Abstract— Security is one of the most important quality attributes in software architecture. Previous modeling approaches provide insufficient support for an in-depth treatment of security. They lack the ability to model important security concepts. Also they are based on formal syntaxes such as using ADLs. This paper presents a more comprehensive treatment of an important security aspect, access control, at the architecture level. Our approach models security user, role, permission, two many-to-many relations for assigning a role to user and permission to role, and policy of architectural constituents. We base our new modeling language on UML 2.0 and try to model security concepts by extending UML. We also provide mechanisms for checking UML models to detect architectural vulnerabilities and assure correct access control at an early design stage.

Index Terms— Secure software architecture, Architectural Access Control, UML 2.0.

I. INTRODUCTION

Security means correct behavior in face of an intelligent adversary or adversaries [1]. Security is a property of a computer system that prevents unauthorized access to resources, modification of information and data, and use of resources. Application security refers to security controls and mechanisms that are included within a software system. Three important properties in secure systems are confidentiality, integrity, and availability. Each secure system can be called secure in three mentioned view points.

In software engineering, security is considered a non-functional requirement [2],[3],[4]. Therefore, it should be considered in all phases of software development. Currently, security is an afterthought and developers usually attempt to remove the vulnerable points, after attacks against the system. This method is called “penetrate and patch” [5] and it usually imposes heavy costs and defects on software projects.

The first phase in software development in which non-functional requirements and specifically security should be addressed is building software architecture [6], [7]. The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [8].

Software Architecture consists of early design decisions and is a basis for design, implementation, and deployment of a system. Thus, in order to address and evaluate security from early stages of software development, security requirements should be modeled at architecture level. To achieve security, there exist various mechanisms that control interactions between components of the system to prevent unauthorized access to system resources and services. At architecture level, these mechanisms are used as architectural tactics [8]. One of the most important tactics to be considered at architecture level is access control, which controls how protected computational resource can be accessed. Access control is arguably the most dominant security assurance mechanism.

In this paper we propose an extended architectural connector, called secure connector. Using secure connector we can model access control at architecture level. In our work, we choose Role Based Access Control (RBAC) [9] from wide range of access control methods. While our approach cannot fully solve the general software security problem, it can complement and possibly guide other solutions that operate on the mathematical properties and low-level implementations to collectively provide the comprehensive solution that is necessary for a complex, componentized, and networked software system.

In this work we have chosen Unified Modeling Language (UML) as our base modeling language and extend it to model important security abstractions. We believe that UML is one of the most complete and flexible modeling languages. In contrast with formal languages, UML is easy to understand and use. It is also accepted as an industrial modeling language between software developers and lots of tools exist to build UML models.

The rest of this paper is organized as follows. Section 2 surveys related work. Section 3 outlines our approach, introducing security abstractions to be modeled by UML and the modeling extensions necessary for security development. Section 4 gives an example of applying the approach to a case study system. The paper is concluded with Section 5 which contains brief recapitulation of the main points and further works.

Pejman Salehi is a M.Sc. student at the Amirkabir University of Technology, Tehran, Iran. (e-mail: psalehi@ce.aut.ac.ir).

Pooya Jaferian is a M.Sc. student at the Amirkabir University of Technology, Tehran, Iran. (e-mail: jaferian@ce.aut.ac.ir).

Ahmad Abdollahzadeh Barforoush is an associate professor at the Amirkabir University of Technology, Tehran, Iran. (e-mail: ahamad@ce.aut.ac.ir).
II. RELATED WORKS

In this section, we overview existing attempts and works in the field of secure software engineering, especially secure software architecture. In [14] the authors present a structured and flexible way for describing security system architectures using the Software Architecture Model (SAM). They also introduce the concept of security constraint patterns, which provides a generic form to formally specify security policies that should be enforced. They present a technique to decompose system-wide constraint patterns onto individual components of the system based on the security architecture model and to verify the consistency between global and component constraints. These constraint patterns define what conditions or properties each component must satisfy under the system architecture. In concert with the architecture model and constraint propagation, they present a flexible and scalable technique to verify whether the security system architecture satisfies the required security constraints. They integrate the above aspects into a systematic and incremental process of security system architecture modeling and verification.

In [27], authors concentrate on formal methods to describe security at architecture level and try to inject subject policy and object by the means of formal methods. In [23], some methods are proposed in order to extend UML for modeling software architecture. These methods are based on a special ADL called C2. In [26] authors focused on architectural connectors and try to propose a new method for documenting general connector using UML 2.0. Ren and Tylor specify a model [15], [16] that centered on software connectors and provides a suitable vehicle to model, capture, and enforce access control. They addressed security problem from an architectural viewpoint that can be used during design and analysis of secure software systems. They also provide a secure software architecture description language for describing architectural access control. This language enables specifying security contracts of components and connectors.

III. SECURITY AND ACCESS CONTROL

Security is usually defined by a security model and security policy. Security policy captures the security requirement of an enterprise or describes the steps that have to be taken to achieve security. It discriminates authorized and unauthorized as considered in a secure system. Security model is an abstraction of security policy. It identifies the relation among the subjects and objects of a system in a formal manner. This relationship specifies the access rights the subject has over system’s resources. Access control is one of the most common types of security model.

Access control, which controls how a protected computational resource can be accessed, is arguably the most dominant security assurance mechanism. In the classic access control model [18],[19] a system contains a set of subjects that has permissions and a set of objects (also called resources) on which these permissions can be exercised. An access matrix specifies what permission a subject has on a particular object. The rows of the matrix correspond to the subjects, the columns correspond to the objects, and each cell lists the allowed permissions that the subject has over the object.

More recent models, such as the role based access control (RBAC) model [9] are often based on employee functions rather than data ownership. In these models, access control decisions are usually determined by the roles that individual users take as part of organization. A recent study by NIST demonstrates that RBAC addresses many needs of the commercial and government sectors. A major purpose of RBAC is to facilitate security administration and review. RBAC introduces the concept of roles as an indirect reference to organize the permissions assignments to subjects. Instead of assigning permissions directly to subjects, the permissions are assigned to roles and users are made members of appropriate roles. This greatly simplifies management of permissions.

In our approach, we use RBAC as our base access control method and try to model in at architecture level using secure connectors.

IV. OUR APPROACH TO SECURE ARCHITECTURE

This section details the elements of the security modeling approach we are taking. We first give an overview on extension mechanisms of the UML and discuss suitability of each for modeling security abstractions. Then we propose a method for modeling architectural connectors with UML. After that we outline the new modeling capabilities we proposed to help assuring correct architectural access control.

A. UML Extension Mechanisms

UML does not support modeling of important security abstractions. As a result, we need to extend UML in well-defined way in order to capture required modeling concerns. UML provides four important extension mechanisms that allow designers to customize and extend the semantic of model elements. Constraints place added semantic restrictions on model elements. Tagged Values allow attributes to be associated with model elements. Stereotypes allow grouping of constraints and tagged values and assigning them a descriptive name to create a new form of meta-class for models. Profiles are predefined sets of stereotypes, tagged values, constraints, and icons to support modeling in specific domains [17].

B. User, Roles, Permission and their relation

We introduce the following core components that are necessary to model access control at architecture level: User, Role, Permission, and Security Policy. We need to introduce two many-to-many relations, first for assigning a role to a user and second for assigning permission to a role.

Users are assigned roles based on their responsibilities and qualifications. Users can be easily reassigned from one role to another. The users can not pass access permission on to other users at their discretion.

Roles are created for the various job functions in an
organization Roles can be granted new permissions as new applications and systems are incorporated, and permissions can be revoked from roles as needed. A role is properly viewed as a semantic construct around which access control policy is formulated. The particular collection of users and permissions brought together by a role is transitory. The role is more stable because an organization's activities or functions usually change less frequently.

Permission is an approval of a particular mode of access to one or more objects in to the system. Permission confers the ability to the holder of the permission to perform some action(s) in the system. Permissions to modify the sets U, R, and P and relations PA and UA are called administrative permissions.

User to role assignment (UA relation) and permission to role assignment (PA relation) are both a many to many relations. The model expected that each role to be assigned to at least one permission and each user to be assigned to at least one role. The key to access control lies in these two relations. Ultimately, it is a user to exercises permissions. The placement of a role as an intermediary to enable a user to exercise permission provides much greater control over access configuration and review than does directly relating user to permission.

Policy is a powerful mechanism for laying out higher-level organizational Constraints. Policy can apply to the UA and PA relations. Policies are predicates which, applied to these relations, return a value of “acceptable” or “not acceptable.” Access control policy is embodied in various components of RBAC such as role-permission, user-role relation. These components collectively determine whether a particular user will be allowed to access a particular piece of data in the system. There are several ways to describe and model policies; one way is using formal languages. We use OCL as a basis architectural security policy modeling.

C. Modeling Architectural Connectors with UML

There are different approaches available to model software architecture views such as components and connectors [8]. In this paper we model components and connectors view with UML 2.0 with focus on architectural connectors.

There are some reasons which made UML appropriate for software architecture which first mentioned by Garlan and associates [24]. These reasons are as follows:

- Semantic match: The UML constructs should map intuitively to the architectural features being documented.
- Visual clarity: The UML description should bring conceptual clarity to a system design, avoid visual clutter, and highlight key design details.
- Completeness: All relevant architectural features for the design should be represented in the UML model. The ability to capture this information is also referred to as expressiveness.

Also there is another reason for using UML as a basis for documenting software architecture which motioned in [23]:

- Tool support: Not all uses of UML are supported equally by all tools (particularly when specializing UML), which can limit documentation options.

Unfortunately UML 2.0 has not a built-in model for documenting software architecture despite its improvements in compare with UML 1.4. Therefore, we must use UML extension mechanisms to extend it for architectural documentations especially component and connector view with respect to security considerations.

Some works has been done to model component and connector view with UML. In [22], two strategies for supporting architectural concerns within UML are presented. One strategy involves using UML “as is,” while the other incorporates useful features of existing ADLs as UML extensions. But the strategies are customized for a specific ADL (C2) architectural style. In [23], some strategies are presented to model components and connectors with UML 2.0 and strengths and weaknesses of each approach are discussed.

In our work, we model architectural components with UML component, and architectural connectors with UML classes. Also to document connector’s properties we extend meta-model of the UML to allow new semantics (properties) to be incorporated into UML classes.

D. Secure Connector

In order to model software architecture from security viewpoint we use component and connector view which shows the system as a set of cooperating units of runtime behavior.

In this approach, each component is running on behalf of a User. It sends its message to the connected secure connector without considering the destination to which the message is routed. Secure connectors play an important role in our approach. They will perform all access control activities and decide on whether the message should be routed to the destination or not. To achieve this goal we should model the mentioned security abstractions such as User, Role, Permission, and Policy at connector level.

With respect to the roles of the secure connector, some information about users, roles, permissions, and their relationship should be embedded in it. Furthermore, it should contain some semantic information based on which access control can be performed.

The UML 2.0 connector concept, which is new, is too lacking in expressiveness to be a good solution for documenting our secure connectors. In particular, it lacks any ability to associate semantic information with a connector (e.g., a behavioral description) or to clearly document C&C connector roles which is an important requirement for our secure connector. Therefore with respect to the requirements of secure connector, we use UML classes in order to model secure connectors. We extend UML 2.0 meta-model and construct a meta-class named connector (Fig. 1).
To embed security abstractions in connectors, we introduce them as properties. Therefore, we extend UML meta-model by inheriting a meta-class named <<secure-connector>> from <<connector>> meta-class. We also define four UML tags with the type of set for Users, Roles-Users, and Permissions-Roles, and Valid-Permissions. The Users set defines the list of users that may interact using the connector. The Roles-Users set contains ordered pair of Users and their assigned role. The Permissions-Roles set contains ordered pair of Roles and their assigned Permissions. The Valid-Permissions set contains those permissions for which the connector will route the message. When modeling software architecture with secure-connectors, we can assign value to the mentioned tags as tagged-values.

We should also define a policy for connector to route a message. We specify secure-connector’s policy using OCL. The default policy is that the message will be routed if it came from a user with a valid permission. The default policy can be overridden by extending <<secure-connector>> meta-class and build a specific secure-connector. We show the extended meta-class in figure 2.

![Fig. 2. Secure connector meta-class and its associated policy](image)

V. CONCLUSION

Software architecture is the first stage to realize quality attributes in software. Security is one of the most important quality attributes in software systems; therefore, it should be modeled and evaluated at architecture level. With respect to the importance of security, a specific method is required to model different security tactics at architecture level.

In this paper we extend UML 2.0 to model secure architectural connectors using which core security concepts in RBAC (users, roles, permissions, and policy) can be specified. In our approach, Component compositions are handled by connectors, which regulate the desired access control property. Our extended modeling language can describe the security properties of software architecture, specify intended access control policy, and facilitate security design and analysis at the architecture level.

The contributions of this research lie in that 1) we address the security problem from an architectural viewpoint. Our use of an architecture model can guide the design and analysis of secure software systems and help security assurance from an early development stage; 2) we provide an extension on the UML 2.0 for describing architectural access control, arguably the most important aspect of security and increase the expressiveness of UML models by modeling users, roles and permissions; 3) using UML for modeling secure connectors, increases the usability, and portability of models in comparison with using ADLs or formal models for architectural access control.

For next phases of our work, we will apply our approach on some case study systems and evaluate and improve our work based on experiences gained from modeling case study systems. Also we will develop an algorithm using which we can check architectural models for correct access control.

REFERENCES