권한의 위임을 위한 역할-기반 접근 제어 모델의 설계 및 구현

나 상 엽*

요 약

현재의 컴퓨팅 환경에서 기업이나 조직내의 사용자들은 다른 사용자와 자원을 공유하며 상호작용을 통하여 보다 효율적으로 작업을 수행하게 된다. 이 경우 공유하는 자원이나 정보의 불법적인 사용을 막고 데이터의 무결성을 유지하기 위하여 사용자의 인증과정이 필요하며, 또한 사용자의 작업에 대한 접근 제어(Access Control)의 필요성이 더욱 중요시되고 있다.

임의적 접근 제어(DAC)는 객체의 소유자에게 모든 위임의 권한이 주어지고 강제적 접근 제어(MAC)의 경우에는 주체와 객체단위의 정책 적용이 어려운 단점이 있다. 최근에는 역할-기반 접근 제어를 이용하여 조직의 보안 정책을 보다 효율적이고 일관성 있게 관리하고자 하는 시도가 있다. 하지만 역할-기반 접근 제어의 경우 각 역할의 계층에 의하여 권한의 상속이 결정되는 문제가 발생한다. 하지만, 하위 개념의 역할이 상위 개념이 가지는 역할의 권한을 수행 하여야 하는 경우가 발생하는데 기존의 역할-기반 접근 제어 모델은 상기의 문제를 해결하기 위한 방법이 존재하지 않고 역할에 새로운 권한을 추가해야만 하는 문제가 존재한다. 따라서 본 논문에서는 이러한 문제를 해결하기 위하여 역할이 가지는 권한의 일시적인 위임을 통하여 이를 해결하고자 한다. 이를 위하여 권한의 위임을 수행하기 위한 새로운 모델과 새로운 역할-권한 관계의 정의, 권한 위임을 수행하는 위임 서버, 그리고 이를 수행하는 프로토콜을 제시한다.

Design and Implementation of Permission Delegation in Role-Based Access Control Model

Sang Yeob Na*

ABSTRACT

In the distributed-computing environment, applications or users have to share resources and communicate with each other in order to perform their jobs more efficiently. In this case, it is important to keep resources and information integrity from the unexpected use by the unauthorized user. Therefore, there is a steady increase in need for a reasonable way to authentication and access control of distributed-shared resources. In RBAC, there are role hierarchies in which a higher case role can perform permissions of a lower case role. No vise versa. Actually, however, it is necessary for a lower case role to perform a higher case role's permission, which is not allowed to a lower case role basically. In this paper, we will propose a permission delegation method, which is a permission delegation server, and a permission delegation protocols with the secret key system. As the result of a permission delegation, junior roles can perform senior role's permissions or senior role itself on the exceptional condition in a dedicated interval.

* 남서울대학교 컴퓨터학과

1. 서 론

오늘날 분산 컴퓨팅 환경이 보편화됨에 따라 기업이나 조직 내의 사용자들은 서로의 자원을 공유하며 상호 작용을 통하여 보다 효율적으로 작업을 수행한다. 여러 사용자들이 상호 작용하는 분산 컴퓨팅 환경에서 공유하는 자원이나 정보가 증가함에 따라 허가되지 않은 정보로의 접근이 발생할 수 있고 불법적인 사용으로 인한 정보의 누출이 발생한다. 따라서 분산 컴퓨팅 환경의 정보를 보호하기 위하여 사용자의 인증이나 사용자의 작업에 대한 접근 제어 정책을 통한 정보 보안의 필요성이 증가하고 있으며 이러한 접근 제어 정책은 시스템 사용의 편리성을 위하여 사용자나 응용 프로그램에 투명하게 제공되어야 한다[1, 6].

접근 제어 정책은 정보의 소유자에 의하여 접근 통제 관계가 정의되는 임의적 접근 제어(Discretionary Access Control : DAC), 정보의 내용(보안등급)과 사용자나 그가 속한 그룹에 의하여 접근을 제어하는 강제적 접근 제어(Mandatory Access Control : MAC), 그리고 시스템 내에 필요한 역할(Role)과 그 역할이 수행할 수 있는 권한(연산)을 정의하고 각 사용자에게 역할을 할당하는 역할-기반 접근 제어(Role-Based Access Control : RBAC)기법으로 나눌 수 있다[2, 3, 7].

강제적 접근 제어의 경우 정보(객체)의 보안등급에 의한 접근 제어를 수행하므로 각각의 사용자와 객체단위로 접근 제어의 설정이 불가능하고 임의적 접근 제어의 경우 객체의 소유자에 의한 접근 정책의 임의변경과 다른 주체에게 자신의 허가된 권한의 임의위임이 가능하여 정보의 효율적 제어가 어렵다.

역할-기반 접근 제어의 경우 미리 정의된 역할, 역할이 수행할 수 있는 접근 권한(Permission)을 명시하고 사용자에게 역할을 부여하므로 사용자는 자신에게 할당된 역할에 의하여 객체를 접근할 수 있다. 따라서 조직은 조직의 특성에 적합한 접근제어 정책을 일관성 있게 유지할 수 있을 뿐 아니라 주체와 자원의 접근 권한 관계를 독립적으로 유지하므로 접근 권한이 변경될 때 새로운 권한을 사용자가 아닌 역할에만 적용하면 되고 복잡한 보안 정책도 추상화하여 효율적으로 관리할 수 있다[4, 5].

역할-기반 접근 제어에서 역할은 조직 내에서 객체에의 접근이 허가된 권한과 책임들의 집합으로 볼 수 있다[4]. 역할-기반 접근 제어에서 각각의 역할은 조직의 접근 정책에 따라 다른 역할과의 상관관계를 가지고 계층구조로 표현되며 상위개념의 역할은 하위개념의 역할의 권한을 상속(Inheritance)받는다[5, 7]. 이 경우 하위개념의 역할이나 계층구조로 포함되지 않은 역할은 상위개념의 역할이 가지는 권한을 수행할 수 없는 문제가 있다.

본 논문에서는 역할의 위임을 통하여 하위개념의 역할이 일시적으로 상위 개념의 역할이 가지는 권한을 수행할 수 있는 방법을 제시하며 위임을 권한의 특성에 의하여 묵시적으로 수행 가능한 내부 위임(Internal Delegation)과 자신에게 부여된 권한을 명시적으로 다른 주체에게 위임하여 주는 외부 위임(External Delegation)으로 구분 하였다. 권한의 위임을 수행하기 위한 새로운 모델과 새로운 역할-권한 관계의 정의, 권한 위임을 수행하는 위임 서버, 그리고 이를 수행하는 위임 프로토콜을 제시한다.

2. 접근 제어 모델(Access Control Model)

2.1 임의적 접근 제어와 강제적 접근 제어

임의적 접근 제어(Discretionary Access Control : DAC) 정책은 정보 소유자의 임의적 판단에 의하여 사용자나 사용자가 속한 그룹의 신원

정보에 근거하여 접근 통제 관계가 정의되는 접근 제어 방법으로, 정보에 대한 접근 권한의 허가 여부가 소유자에 의해 결정되므로 유연한 보안관리 기능을 제공하는 특징을 가지며, 정보에 내포된 임의적 접근 제어 관계가 복사된 정보의 소유자에 의하여 재정의(Redefine) 되므로, 각각의 정보와 사용자에 대하여 접근 권한을 설정할 수 있다.

임의적 접근 제어의 경우 자신이 허가권을 가진 정보의 접근 관계를 임의의 다른 사용자에게 부여할 수 있는 장점을 가지지만 정보의 소유자에 의한 접근 정책의 임의 변경이 가능하고 자신에게 허가된 권한을 다른 사용자에게 임의로 위임할 수 있으므로 정보의 효율적 제어가 어렵다. 또한, 접근 제어가 사용자의 신원 정보에만 근거하므로 정보의 등급에 의한 접근 관계의 설정이 불가능하여 같은 정보라 할지라도 정보 소유자에 의하여 동일한 사용자에게 각기 다른 접근 권한이 적용될 수 있다.

강제적 접근 제어(Discretionary Access Control : DAC)의 경우 각 정보에 부여된 보안 등급과 사용자에 부여된 인가 등급을 조직 내에서 규정된 접근 규칙과 비교하여 규칙에 만족하는 사용자에게만 접근 권한을 부여하는 접근 제어 방법이다. 규칙에는 정보의 소유자에 의하여 변경될 수 없는 사용자와 정보의 접근 통제 관계가 명시되어야 한다. 정보에 대한 접근 권한은 복사된 정보에 상속되는 특성을 가지고 있어 모든 정보와 사용자에게 일관된 접근 관계를 정의할 수 있다.

그러나 각각의 정보와 사용자단위로 접근 권한의 설정이 불가능하고 시스템의 보안 관리자가 모든 정보와 사용자에 대하여 보안등급을 부여, 관리하여야 한다. 또한, 정보의 소유자도 접근 권한의 변경이 불가능하며 정보의 보안 등급에 의하여 접근 권한 관계가 설정된다. 강제적 접근 제어는 군사 환경과 같은 정보의 비밀성 위주의 매우 통제된 환경에서 주로 사용되고 있다[2, 8].

일반적인 기업 조직의 경우 다른 조직들과 각각 상이한 보안 정책을 필요로 하므로 TCSEC에서 정의한 임의적 접근 제어나 강제적 접근 제어만으로 기업의 보안 요구를 충족시킬 수 없으며, 강제적 접근 제어 정책은 정보의 기밀성을 위한 정보의 보안에 중점을 두고 있으며, 임의적 접근 제어 정책은 접근 정책 제어 권한의 제어가 정보의 소유자에 의존하므로 정보의 효율적인 제어가 어렵고, 순수한 임의적 접근 제어 정책은 기업 환경에 적용하기에는 적합하지 않다[5, 9].

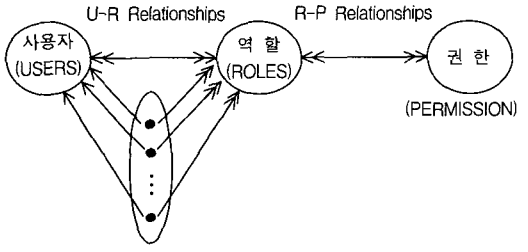
2.2 역할-기반 접근 제어 모델

역할-기반 접근 제어(Role-Based Access Control) 정책은 정보에의 접근 여부를 미리 정의된 규칙이나 사용자의 신원 정보에 의하여 판단하지 않고, 조직 내에서 사용자가 수행하는 역할(Role)에 의하여 결정되는 특징을 가진다.

조직의 보안 관리자는 조직 내에서 필요한 역할과 역할이 수행할 수 있는 접근 권한(Permission)을 명시하고 사용자에게 필요한 역할을 부여한다. 사용자는 자신에게 부여된 역할에 의하여 정보에 접근하므로 접근 권한이 사용자에게 직접 부여되지 않고 조직 내에 정의된 역할에게 부여된다. 따라서 조직은 조직의 특성에 적합한 접근 제어 정책을 일관성 있게 유지할 수 있으며, 사용자와 정보의 접근 권한 관계를 서로 독립적으로 관리하므로 접근 권한의 변경이 필요한 경우 새로운 권한을 사용자가 아닌 역할에만 적용하면 된다. 역할-기반 접근 제어의 경우 복잡한 보안 정책을 추상화하여 관리할 수 있으므로 사용자가 많은 실제 기업 환경에 효과적으로 적용할 수 있다[5, 9, 10].

역할-기반 접근 제어는 비교적 최근의 접근 제어 기법으로 복잡한 조직의 형태를 추상화하여 효율적으로 표현할 수 있는 장점은 가지고 있지

만 아직까지 정형화된 모델은 제시되지 않고 있다. 많은 역할-기반 접근 제어 모델이 제시되어 연구되고 있지만 기본적인 구성 요소로는 사용자(User), 역할(Role), 역할 계층(Role Hierarchy), 권한(Permission), 사용자-역할 관계(User-Role Relationship), 그리고 역할-권한 관계(Role-Permission relationship) 등이 있다.



(그림 1) 역할-기반 접근 제어 모델

2.2.1 사용자(User)

사용자는 시스템 내의 응용 프로그램이나 시스템을 사용하는 권한을 가진 사람으로 보안을 필요로 하는 객체에의 접근을 요구하는 주체이다. 사용자가 시스템에 접근하기 위해서는 일차적으로 사용자의 계정과 패스워드 같은 인증을 위한 방법이 제공되어야 한다. 사용자가 합법적으로 인증을 받게 되면 역할-기반 접근 제어 시스템은 사용자가 시스템에서 수행할 수 있는 임무에 의하여 미리 정의된 역할을 부여한다.

2.2.2 역할(Role)

역할은 역할-기반 접근 제어 모델의 가장 중요한 구성 요소로, 조직 내에서 사용자에게 주어지는 의미적 구조체로 객체에의 접근이 허가된 책임과 의무의 집합으로 볼 수 있다. 각 역할은 해당 역할에서 수행가능 한 권한을 부여받는다. 역할은 조직의 보안 정책에 따라 다른 역할과의 상관관계에 의하여 계층 구조(Hierarchy)를 가지고 표현된다[5].

2.2.3 권한(Permission)

권한은 시스템 내의 객체에 대해 수행 가능한 접근 모드의 집합으로 구성되며, 접근 권한(Access Right), 특권(Privilege), 허가(Authorization) 등으로 표현된다. 이는 시스템 내에서 하나 이상의 객체에 대한 특별한 접근 모드(Access Mode)의 승인으로 권한의 소유자는 시스템에서 행동을 수행할 능력을 가진다. 예를 들어, 파일에 대한 권한은 '읽기', '쓰기', '실행' 등이 있을 수 있는데, 역할-기반 접근 제어 정책은 다른 접근 제어 정책과는 달리 '읽기', '쓰기' 등의 저 수준의 권한 대신 접근 제어 정책을 구현하는 시스템의 특성에 따라 '조제', '입금' 등의 추상적인 권한을 허용하여 상업 환경에서의 보안 정책의 적용을 보다 용이하게 한다.

2.2.4 역할 계층(Role-Hierarchy)

역할 계층은 역할이 수행할 수 있는 권한들 사이에 포함관계를 가지는 부분 순서(Partial Ordering) 관계로 역할의 계층 구조에서 역할의 위치에 따라 상위 역할(Senior Role)이 하위 역할(Junior Role)의 권한을 가지는 상속(Inheritance) 관계와 다중 상속 관계가 성립한다. 역할 계층의 부분 순서 관계는 반사적(Reflexive), 이행적(Transitive), 그리고 비대칭적(Anti-Symmetric) 관계를 가진다[5, 13].

2.2.5 사용자-역할 관계(User-Role Relationship)

사용자-역할 관계는 다대다 관계이며 사용자가 시스템에 인증 과정을 거쳐 시스템 내에서 작업을 진행하기 시작하면 시스템 보안 관리자는 사용자가 시스템 내에서 수행할 수 있는 역할을 배정해 주어야 한다. 사용자-역할 관계는 다대다 관계이므로 사용자는 여러 역할에 배정될 수 있고 동일한 역할에 다수의 사용자가 배정될 수 있다.

2.2.6 역할-권한 관계(Role-Permission Relationship)

역할-권한 관계는 해당 역할이 수행할 수 있는 권한을 명시하여 준다. 역할-권한 관계에 의하여 시스템 내에서 사용할 수 있는 권한은 역할에게만 할당되고 사용자는 권한과 직접적인 관계를 가지지 않는다. 즉, 사용자가 정보 객체에 실행할 수 있는 연산들이 사용자에게 직접 부여되는 것이 아니라 역할-권한 관계에 의하여 해당 역할에 필요한 권한을 배정하고 사용자는 해당 역할의 구성원이 됨으로써 객체에 원하는 연산을 수행하도록 한다[12].

〈표 1〉 역할-기반 접근 제어 모델 기술을 위한 함수

$user : S \rightarrow U,\ user(s_i) = u$
$R \rightarrow 2^U,\ user(r_i) = \{u \in U | (u, r_i)\ UA\}$

$roles : S \rightarrow 2^R,\ roles(s_i) \subseteq \{r \in R | (user(s_i), r) \in UA\}$
$U \uplus P \rightarrow 2R,\ roles(u_i) = \{r \in R | (u_i, r) \in UA\}$
$roles(p_i) = \{r \in R | (p_i, r) \in PA\}$

$sessions : U \rightarrow 2^S,\ sessions(u_i) = \{s \in S\}$

$permissions : R \rightarrow 2^P,$
$permissions(r_i) = \{p \in P | (p, r_i) \in PA\}$

$operations : R \times OBJ \rightarrow 2^{OP},$
$operations(r^i, obj_i) = \{op \in OP | (op, obj_i, r_i) \in PA\}$

3. 위임 정책

현재까지의 역할-기반 접근 제어 모델에 대한 연구는 모델 구성 요소들에 대한 관리와 객체에 대한 사용자 접근 요청의 허가여부이며, 주로 모델의 정립과 모델 구성 요소의 정형화, 그리고 의무 분리, 최소 권한 원칙 등의 보안특성에 관한 연구가 주를 이루었다. 현재의 역할-기반 접근 제어 모델은 사용자가 시스템을 사용하는 시점에서 세션이 사용자에 의하여 실행되고, 세션에 의하여 활성화된 역할과 해당 역할에 할당된 권한만을 사용자가 수행할 수 있도록 제어하는 기능을 제공한다. 즉, 세션이 사용자의 임의적인 결정에 의하여 수행되는 사용자 중심의 접근 통제 방식이다.

역할-기반 접근 제어 모델의 주요 구성 요소인 역할 계층의 특징 중 하위 역할에 배정된 권한들이 상위 역할로 상속되는 특징은 동일한 역할 그룹 내에서 계층관계를 가지는 역할들 간의 효율적인 권한 배정과 조직의 권한체계를 자연스럽게 모델링 하는 장점을 가진다. 하지만, 현재 역할-기반 접근 제어 모델에서 제안하는 역할의 정의를 실제 환경에 직접 적용시키는 데는 문제점이 있어 최소 권한 원칙의 수행과 불필요한 상속의 권한을 제한하는 연구와 필요성도 제시되고 있다[11].

역할-기반 접근 제어 정책을 사용하는 시스템의 경우 역할 또는 역할에 주어진 권한을 다른 역할이나 사용자에게 위임 또는 전가를 필요로 하는 경우가 발생한다[1, 11]. 역할-기반 접근 제어 시스템에서 접근 제어 정책을 적용하기 위하여 시스템의 보안 관리자는 조직 내에서 기능상 필요한 역할을 정의하고 역할 수행에 필요한 권한을 정의하여야 한다. 또한 해당 역할을 수행할 자격을 가진 사용자를 역할에 할당하여 접근 제어가 필요한 객체에의 접근은 사용자에게 할당된 역할에 의해서만 접근하게 하여야 한다. 이러한 정적인 역할의 생성과 할당을 하는 역할-기반 접근 제어 모델은 다음과 같은 두 가지 문제점을 지니고 있다.

첫째, 역할의 생성과 배정, 권한의 부여 등 시스템의 보안정책을 관리하기 위한 모든 작업이 강제적 접근 제어 정책과 같이 시스템 보안 관리자에 부과되므로 보안관리자의 작업이 증가하

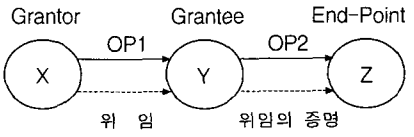
고, 접근 제어 정책의 변화 시 역할-기반 접근 제어 정책이 가지는 융통성을 감소시킨다.

둘째, 계층적 역할-기반 접근 제어의 경우 역할 계층의 상속 관계에 의하여 상위 역할은 하위 역할이 가지는 권한을 묵시적으로 소유하게 되므로 상위 역할은 하위 역할이 가지는 권한을 수행할 수 있으나 계층 구조 내에서 하위 역할은 상위 역할이 가지는 권한을 수행하는 방법은 제공되지 않는다. 이때, 하위 역할이 자신의 기능을 올바르게 수행하지 못하는 경우에는 권한 상속에 의하여 상위 역할이 묵시적으로 소유하는 하위 역할의 권한을 수행하여 조직의 기능을 정지시키지 않고 작업을 진행할 수 있지만, 그 반대의 경우 조직의 기능이 올바르게 실행되지 않는 문제가 발생한다.

위와 같은 문제를 해결하기 위하여, 이 논문에서는 위임을 통한 문제 해결 방안을 제시하고 이를 위하여 권한 위임의 개념을 설명하고 권한 위임을 수행하기 위한 위임 서버의 기능과 위임 프로토콜에 대하여 기술한다.

4. 권한 위임(Permission Delegation)

비임의적 접근 제어 모델에서 사용자가 접근 제어를 받는 객체에 접근하기 위해서는 해당 객체의 소유자일지라도 시스템 보안 관리자의 작업에 의하여 해당 정보 객체의 접근 권한을 부여받아야 한다. 보안 관리자는 자신은 접근할 수 없는 정보 객체일지라도 조직 내에서 적절한 기능을 가진 사용자에게 해당 객체에 접근할 수 있는 권한을 부여하여 주는 기능을 가지는데 이러한 관리자의 작업은 자신이 가지고 있지 않은 권한 위임의 특별한 경우로 볼 수 있다. 비임의적 접근 제어 모델에서 시스템 사용자들은 자신에게 할당된 권한을 다른 사용자에게 임의로 허가할 수 없는 정책이지만, 실제 기업 환경에서는 자신이 가지고 있는 권한을 다른 사용자에게 전이해야 하는 경우가 빈번히 발생한다[1, 11]. 이 경우, 자신이 부여받은 권한을 다른 사용자에게 위임하는 방법이 필요하며 이러한 권한 위임의 기본 모델은 (그림 2)로 표현된다.



(그림 2) 위임 정책의 기본 모델

4.1 내부 위임(Internal Delegation)과 외부 위임(External Delegation)

권한은 해당 권한이 가지는 특성에 의하여 내부 위임과 외부 위임으로 구분할 수 있다.

예를 들어, 파일이라는 객체에 대한 권한은 읽기(Read), 쓰기(Write), 첨가(Append) 등이 있을 수 있으며 이러한 권한이 가지는 특성상 쓰기 연산은 읽기나 첨가 연산보다 상위에 있다고 할 수 있다. 이때, 쓰기 권한을 가지는 역할은 묵시적으로 읽기나 첨가 권한을 수행 할 수 있으며, 쓰기 연산에는 읽기, 첨가 연산이 포함된다 할 수 있다. 이처럼 연산이 가지는 특징에 의하여 명시되지 않은 권한을 수행할 수 있는 경우를 내부 위임(Internal Delegation)으로 정의할 수 있다.

이에 반하여, 어떠한 객체에 대하여 접근할 수 있는 권한과 자신에게 허가된 접근 권한을 다른 주체에게 승인할 수 있는 권한을 분리하는 경우에는 내부 위임과 다른 위임 방법을 정의하여야 한다. 즉, 어떠한 객체에 대하여 접근 원한을 가지고 있다 하더라도 이는 해당 주체에 접근할 수 있는 권한이 주어진 것이고 자신에게 주어진 권한을 다른 주체에게 위임할 수는 없게 제한해야 하는 것이다. 이를 위해 해당 권한을 x 권한과 xc 권한으로 구분하여야 하는데 x 권한은 해당 주체에게만 접근 권한이 주어진 경우이고 xc권한은 다른 주체에게 위임 권한까지 주어진 경우

를 의미하고 이 경우 xc 권한을 다른 주체에게 위임하는 것을 외부 위임(External Delegation)으로 정의 한다. 일반적으로 xc 권한은 x 권한을 포함하고 있다. 이때 다음과 같이 두 가지 경우가 발생할 수 있다.

어떠한 객체에 대하여 xc 권한을 가진 주체는 해당 객체의 x 권한을 다른 주체에게 위임할 수 있다.

어떠한 객체에 대하여 xc 권한을 가지고 있는 주체는 해당 객체의 xc 권한이나 x 권한을 다른 주체에게 위임할 수 있다.

이 두 가지 경우는 위임의 정도를 나타내기 위하여 구분되는데, 첫 번째의 경우에는 xc 권한을 가진 주체에 의하여 객체에의 접근 여부만을 나타내는 x 권한만 위임되는 경우이므로 권한을 위임받은 주체에 의한 다른 위임은 발생하지 않는다. 하지만, 두 번째 경우에는 xc 권한을 가진 주체에 의하여 xc 권한이나 x 권한을 다른 주체에게 위임할 수 있으며 xc 권한을 위임받은 주체는 또 다른 주체에게 xc 권한이나 x 권한을 위임할 수도 있다. 이때 위임된 xc 권한의 위임 정도(Delegation Degree)를 나타내야 하는데 이는 xc*로 표시할 수 있다.

〈표 2〉 권한 위임의 정형화된 표현

생성(create)
=> cc : Subject → 2^{Object}

권한(right)
=> right : Subject × Object → 2^{right}

내부-위임(int-delegation)
=> int-dele : Subject × Object × right → 2^{rirght}

외부-위임(ext-delegation)
=> ext-dele :
Subject × Subject × Object right → 2^{right}

xc* 권한은 xc 권한이나 x 권한으로 위임이 가능하고 xc 권한은 x 권한으로만 위임이 가능하다. 이때, xc 권한은 일단계 위임만이 가능한 권한이고 xc* 권한은 무제한 위임이 가능한 형태이다. 또한, 위임의 정도를 제한하는 경우도 가능한데 이는 xc^n으로 표현된다. 즉, 권한의 위임이 진행됨에 따라 xc^n 권한은 xc^{n-1}, xc^{n-2}, …, xc^1, x로 차수가 감소한다. 이때 위임의 정도는 권한을 가지는 주체나 연산이 수행되는 객체의 특징에 따라 보안 관리자에 의하여 결정된다. 예를 들어, 같은 그룹에 속하는 역할들 사이의 위임은 무제한 위임이 가능하게 하고 다른 역할 그룹에 속하는 역할에 권한을 위임하는 경우에는 일 단계 위임만이 가능하게 제한할 수 있다. 이를 정형화하여 나타내면 <표 2>와 같다.

내부-위임을 나타내는 int-dele 함수는 int-dele(u, o, x) = {x_1, x_2, …, x_n}으로 나타나는데 하나의 주체 u가 객체 o에 대하여 x 권한을 가지고 있는 경우 주체 u는 내부-위임을 통하여 x_1, x_2, …, x_n의 권한을 부여받을 수 있음을 나타낸다. <표 2>를 이용하여 파일 객체의 내부-위임 권한을 표현하면 다음과 같다.

int-dele(user, file, w) = {a, r}
int-dele(user, file, a) = {r}
int-dele(user, file, r) = ∅

이는 어떠한 파일에 대하여 쓰기(Write) 권한을 가지는 사용자는 내부 위임을 통하여 첨가(Append)와 읽기(Read) 권한을 위임받을 수 있으며, 첨가(Append) 권한을 가지고 있는 사용자는 읽기(Read) 권한을 위임받을 수 있다는 것을 나타낸다.

외부-위임을 나타내는 ext-dele 함수는 ext-dele(u, v, o, x) = {x_1, x_2, …, x_n}으로 나타나는데 하나의 주체 u가 객체 o에 대하여 x 권한을 가지고 있는 경우 주체 u는 외부-위임을 통하여 x_1, x_2, …, x_n의 권한 중 일부를 또 다른 객체 v에게 위임하는 것을 나타낸다.

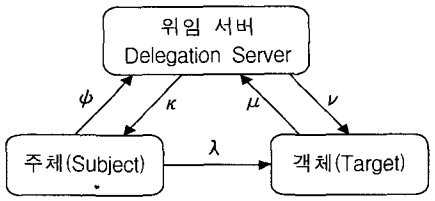
ext-dele(user, user, file, xc*)
= {xc*, xc1, x}
ext-dele(user, user, file, xc1) = {x}
ext-dele(user, user, file, x) = ∅

4.2 권한 위임 서버(Permission Delegation Server)

권한 위임 서버는 역할-역할 관계, 역할-권한 관계, 역할 계층, 그리고 예외 조건 등을 종합하여 역할 위임의 가능 여부를 최종적으로 결정해 주는 기능을 수행한다. 역할 위임이 필요한 경우 위임을 원하는 주체(Subject)는 위임을 원하는 역할과 대상을 명시하여 위임 서버에게 위임 증명의 발급을 요구한다. 이 정보는 자신과 위임 서버 사이에 공유하는 비밀 키로 암호화하여 전달한다, 위임 서버는 주체가 전달한 정보를 자신이 가지는 비밀 키로 복호화 후에 자신이 가지는 위임 가능 정보와 비교하여 위임 가능 여부를 판단하고 위임서(Certificate)를 다시 암호화하여 주체에게 전달한다. 주체는 서버로부터 받은 위임서를 자신과 객체 사이의 공유키로 암호화하여 객체에게 전달한다. 위임서를 전달받은 객체는 이를 위임 서버에게 보내 검증을 요구하고 서버는 위임서가 올바른 것이라는 것을 증명하여 준다. 위와 같은 과정을 거치면 객체는 주체에 의하여 해당 역할을 위임받을 수 있으며 역할-권한 관계에 명시되어진 권한을 수행한다. 이때 위임서에는 위임이 유효한 시간을 명시하여 주어진 시간만큼 객체가 일시적으로 역할의 권한을 수행하게 한다. 프로토콜은 (그림 3)으로 표현되고, 위임 서버와 주체, 객체 사이에는 공개 키와 비밀 키 알고리즘이 사용된다.

4.3 권한 위임 프로토콜(Permission Delegation Protocol)

권한 위임을 위한 프로토콜은 다음의 (그림 3)로 표현된다.



Message φ : {주체, 역할, 객체, 조건}Ks-ds
(Delegate Request)
Message κ : {Certificate}Ks-ds {Certificate}Kds
(Delegation Request Result : Certificate)
Message λ : {Certificate}Kds (Certificate)
Message μ : {객체, {Certificate}Kds}Kt-ds
(Certificate Authentication)
Message ν : {Certificate}Kt-ds (Authentication Result)

Certificate : <객체, 역할, 기간>
Ki-j : i 와 j 가 공유하는 비밀키
Kk : k의 비밀키

(그림 3) 권한 위임 프로토콜

권한 위임을 수행하기 위해서 권한 위임을 수행하고자 하는 주체가 위임 요청(Delegate Request : Message ①) 메시지를 위임 서버에게 전송 하여야하는데 메시지에는 권한 위임의 주체, 위임하는 권한, 위임받는 객체, 위임을 위한 조건 등의 정보를 가지고 위임 서버와 역할 위임을 요청하는 객체 사이의 공유 비밀 키로 암호화된다. ①의 메시지를 받은 위임 서버는 위임 가능 여부를 판단하여 그 결과를 위임 서버의 비밀 키, 위임 서버와 객체사이의 공유키로 각각 암호화하여 인증서의 형태(Certificate : Message ②)로 다시 주체에게 전송한다.

위임서에는 위임받는 객체, 위임받은 권한, 위임 기간을 명시하여야한다. 인증서를 받은 주체는 자신과 위임 서버 사이에 공유하는 비밀 키로 인증서의 내용을 확인하고 위임 서버의 비밀 키로 암호화된 메시지(Message ③)를 객체에게 전송한다. 주체로부터 인증서를 전달받은 객체는 자기 자신의 id와 전달받은 인증서를 인증 서버와 자신이 공유하는 비밀 키로 암호화하여(Certificate Delegation : Message ④) 위임 서버에게 전송한다.

④의 메시지를 전송 받은 인증 서버는 인증서의 내용이 변조되었는지 확인하고 이상이 없으면 이를 다시 자신과 객체 사이의 공유키로 암호화하여 객체에게(Message ⑤) 전송한다. 이러한 과정을 통하여 위임은 이루어지고 위임 서버로부터 인증서를 전송받은 객체는 인증서에 나와 있는 위임받은 권한을 제한된 기간동안 수행함으로써 권한의 위임이 이루어진다.

5. 결 론

접근 제어 정책은 승인된 주체의 보안 객체에 대한 합법적인 접근 허가를 나타내는 정책으로 사용자의 접근 권한 소유 여부를 판단하는 기술이다. 대표적인 접근 제어 정책으로는 강제적 접근 제어 정책, 임의적 접근 제어 정책 그리고 역할-기반 접근 제어 정책 등이 있다. 강제적 접근 제어는 정보 객체에 대한 사용자의 접근 권한 부여를 시스템 보안 관리자에 의해 정의된 규칙에 의하여 결정되며, 이러한 규칙은 보안 관리자에 의해서만 변경되는 중앙 집중형 보안 관리가 제공된다. 임의적 접근 제어는 정보 객체에 대한 접근 권한의 허가 및 취소를 일방적으로 정보의 소유자가 수행하게 함으로써 강제적 접근 제어 정책 보다 유연한 보안 관리 기능을 제공하는 특징을 지닌다.

역할-기반 접근 제어 정책은 기존의 강제적 접근 제어 정책이나 임의적 접근 제어 기법보다 상업적 목적을 가지는 거대한 조직의 형태를 보다 효율적으로 표현할 수 있는 장점을 가지며 정보에 대한 추상적인 접근 통제와 효율적인 접근 권한 관리를 수행할 수 있는 특징을 지닌다.

역할-기반 접근 제어 기법은 다양한 분야에 적용할 수 있는 장점을 가지고 있으며 최소 권한 원칙, 의무 분리 원칙 등을 지원하여, 하나의 보안 정책을 지원하는 것이 아니라 모델의 구성 요소와 구성 요소간의 관계를 설정하여 여러 가지 다양한 형태의 보안 정책을 지원하는 특징을 가진다.

역할-기반 접근 제어 모델은 사용자, 역할, 권한, 역할 계층, 의무 분리 등의 여러 가지 구성 요소를 가진다. 역할-기반 접근 제어 모델의 주요 구성 요소인 역할 계층의 특징 중 하위 역할에 배정된 권한들이 상위역할로 상속되는 특징은 동일한 역할 그룹 내에서 계층관계를 가지는 역할들 간의 효율적인 권한 배정과 조직의 권한 체계를 자연스럽게 모델링 하는 장점을 가진다.

그러나 계층적 역할-기반 접근 제어의 경우 역할 계층의 상속 관계가 단 방향으로만 고정되어 있고 계층 구조의 형태가 역할의 생성 시에 정적으로 확정되는 단점을 가지고 있다.

이러한 문제를 해결하기 위하여 본 논문에서는 하위 역할이나 다른 그룹에 속하는 역할이 상위 역할이 가지는 권한을 일시적으로 수행할 수 있게 하는 위임 방법을 제시하였다. 권한 위임은 자신이 가지는 권한의 전부 또는 일부를 다른 사용자에게 전이하여 다른 사용자가 자신이 가진 권한을 수행할 수 있도록 하는 것으로, 위임이 수행되면 위임을 받은 역할이나 사용자는 위임된 역할이 가지는 권한을 수행함으로써 상위 역할이 어떠한 이유에 의하여 해당 역할의 기능을 수행할 수 없더라도 조직의 기능이 계속하여 실행될 수 있다. 또한, 위임 가능 여부를 시스템 관리자가 수행하지 않고 자동화된 서버에 의하여 수행함으로써 시스템 보안 관리자에게 부가되는 작업을 분산하는 효과가 있을 수 있으며 시스템이 동작중 위임 서버에 의하여 위임이 수행되므로 사용자는 동적으로 권한을 할당받고 부여된 권한을 수행 할 수 있다.

이밖에도 권한 위임을 수행하기 위하여 위임의 수행 여부를 판단하는 위임 서버를 제안하였고 확장된 권한의 형태를 제시하였다. 또한, 권한이 가지는 특성에 의하여 위임을 내부 위임과 외부 위임으로 구분하였으며 이의 개념을 제시한 권한과 연계하여 위임에 대하여 설명하였다.

본 논문에서 제안한 권한 위임은 위임 서버에

의하여 역할의 할당이 수행되므로 시스템의 수행 중에 동적으로 역할을 할당받아 권한을 수행하게 함으로서 역할-기반 접근 제어 정책의 역할 계층 구조에 의한 정적으로 표현되는 역할 상속을 해결할 수 있고, 조직 내에서 역할들의 상호 작용에서 역할에 부여되지 않은 권한을 수행하여야 하는 경우 권한 위임을 통하여 일시적으로 필요한 권한을 부여해주는 방법을 제시하였다.

권한 위임의 경우 역할-기반 접근 제어 모델에서 의무 분리 조건을 만족하지 않는 역할 들 사이의 권한 위임을 위한 지속적인 연구가 필요하고 위임의 기간을 명시하는 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] N. Yialelis, M. S. Sloman, "A Security Framework Supporting Domain Based Access Control in Distributed Systems", ISOC Symposium on Network and Distributed System Security(SNDSS96), 1996.

[2] Department of Defence(USA), Department of Defence Trusted Computer System Evaluation Criteria, DoD 5200-78-STD, DoD, 1985.

[3] David F. Ferraiolo, J. Cugini and Richard Kuhn, "Role-Based Access Control : Features and Motivations", National Institute of standards and technology, 1995.

[4] E. C. Lupu, D. A. Marriott, M. S. Sloman, and N. Yialelis, "A Policy Based Role Framework for Access Control", First ACM/NIST Role Based Access Control Workshop, 1995.

[5] Ravi S. sandhu, Edward J. Coyne, Hal L. Feinstein and Charles E. Youman, "Role_-Based Access Control Models", IEEE computer, Vol.29, No.2, pp.38-47, 1996.

[6] B, Lampson, M, Abadi, and R, Needham, "A Logic of Authentication", ACM Transaction on Computer System, Vol.8, No.1, 1990.

[7] David F. Ferraiolo, J. Cugini and Richard Kuhn, "Role-Based Access Control : Features and Motivations", National Institute of standards and technology, 1995.

[8] Warwick Ford, "Computer Communication Security", Prentice-Hall, 1994.

[9] Ravi S. Sandhu, David Ferraiolo and Richard Kuhn, "The NIST Model for Role-Based Access Control : Towards A Unified Standard", Proceedings of the 5th ACM RBAC Workshops, pp.47-63, 2000.

[10] Ravi S. Sandhu and Pierangela Samarati, "Access Control : Principles and Practice", IEEE Computer, pp.40-48, 1994.

[11] Cheh Goh and Adrian Baldwin, "Towards a more complete model of role", Proceedings of the 3rdACM RBAC Workshops, pp.55-62, 1998.

[12] John Barkley and Anthony Cincotta, "Managing role/permission relationships using object access types", Proceedings of the 3rd ACM RBAC Workshops, pp.73-80, 1998.

[13] Jonathan D. Moffett and Emil C. Lupu, "The uses of role hierarchies in access control", Proceedings of the 3rd ACM RBAC Workshops, pp.153-160, 1999.

나 상 엽

1992년 동국대학교 전자계산학과 (공학사)

1995년 동국대학교 컴퓨터공학과 (공학석사)

2001년 동국대학교 컴퓨터공학과 (공학박사)

1996년~현재 남서울대학교 컴퓨터학과 교수