通信情報保護學會論文誌
第 10 券 第 3 號, 2000. 9

워크플로우 응용을 위한 이동 에이전트 시스템에의 역할-행위 기반 접근통제 적용*

신 욱**, 이 동 익***, 윤 석 환****

Role-Behavior Based Access Control on Mobile Agent System for Workflow Management System

Wook Shin**, Dong-Ik Lee***, Seok-Hwan Yoon****

요 약

최근, 기업 및 공공기관에서의 응용을 목적으로 다중 사용자, 계층 구조의 특성을 보이는 상업용 소프트웨어의 개발 사례가 증가하고 있다. 보안 서비스의 제공은 어떤 시스템을 설계하더라도 필수적으로 고려되어야 하는 사항이며, 접근통제는 보안 서비스를 제공함에 있어서 빼놓을 수 없는 서비스 요소이다. 기존 접근통제 정책으로 사용되던 방법으로는 임의 접근통제, 강제 접근통제, 역할 기반 접근통제 방법 등이 존재하나, 다중 사용자, 계층구조 등의 특징을 갖는 상업용 소프트웨어에의 적용에 있어서, 몇 가지 문제점을 안고 있다. 이에 본 논문은 기존의 역할 기반 접근통제 방법을 확장, '행위'라는 개체를 정의함으로써 시스템 자원의 계층구조를 반영하는 새로운 접근통제 방법을 제안하며, 응용에 앞서 기본적인 형태의 접근통제 모델을 제시한다. 또한, 다중 사용자, 계층적인 구조를 갖는 소프트웨어의 대표적 사례인 이동 에이전트 기반 워크플로우 시스템에 제안한 방법을 간단히 구현한 후, 구현결과를 보임으로써 새 접근통제 방법의 실제 시스템 적용 가능성을 타진한다.

ABSTRACT

In these days, it is rapidly increasing that multi-user, multi-layered commercial software developments for companies or public institutions. Security services are necessary for most of systems and the access control service is the essential of security services. Current access control methods that are used as access control policies are classified as Discretionary Access Control, Mandatory Access Control, and Role Based Access Control. However, there are some inefficiencies when those methods are applied to current multi-user, multi-layered systems. Therefore, it is required that a new access control method that takes complex system resources into account from the side of policy. In this paper, extending previous Role Based Access Control, we propose a new access control method that reflects layered system organization by defining a new entity of 'Behavior' and a basic model of the method. And, we simply implement the method on the mobile agent based workflow management system that is a representative example of multi-user, multi-layered softwares and shows implementation results to tap possibilities of real-world application.

keyword : *role-based access control, multi-layered system, behavior, workflow, mobile agent*

* 본 연구는 일부 한국과학재단 특정기초연구과제(98-0102-11-01-3) 지원 및 정보통신부 대학 정보통신 연구센터 지원 사업의 일환으로 수행되었습니다.
** 광주과학기술원 정보통신공학과 병행시스템 연구실(sunihill@geguri.kjist.ac.kr)
*** 광주과학기술원 정보통신공학과 병행시스템 연구실(dilee@kjist.ac.kr)
**** 정보통신 연구 진흥원

Ⅰ. 서 론

최근 기업 및 공공기관에서의 업무 처리를 자동화하기 위하여 많은 소프트웨어들이 개발되고 있다. 이들 소프트웨어들이 가지는 공통적인 특징을 크게 둘로 요약하면 다음과 같다. 첫째, 다수의 사용자를 사용 주체로 갖는다. 일반적으로 기업 및 공공기관의 구성원 수는 수천에 이르므로, 이들 소프트웨어는 이전의 개인용 소프트웨어들과 뚜렷하게 구별되는 다수의 사용자 집합을 가진다. 둘째, 계층적인 구조를 갖는다. 기업용 업무 처리 소프트웨어들은 네트워크 환경에서의 수행을 지원하는 것이 보통이며, 효율적인 처리를 위해 분산 환경에서의 수행을 지원하기도 한다. 따라서, 네트워크 지원 계층 또는 CORBA, DCOM, 이동 에이전트 시스템 등의 분산 객체 시스템을 하위 계층으로 하여 설계되는 경우가 많다. 또한, 단일 솔루션 공급을 통하여 여러 서비스를 통합, 제공하려는 목적으로 소프트웨어의 크기가 점점 거대화하는 추세에 있어, 계층적인 설계 방법을 통해 설계, 관리 측면의 편의를 추구하기도 한다.

소프트웨어 시스템의 보안은 시스템 설계 시 항상 고려되어야 하는 사항이며, 특히 기업 및 공공기관에서 사용되는 이들 소프트웨어의 경우 보안 서비스의 제공은 필수적이다. 접근통제는 보안 서비스를 제공함에 있어서 빼놓을 수 없는 중요한 고려 사항이다. 접근통제란 어떤 시스템 내에서 사용자 또는 시스템 구성 요소의 식별자와 권한에 의거, 시스템 자원에 허가된 접근만을 허용하는 과정을 의미하며, 접근통제 정책 수립 후, 적절한 메커니즘을 사용하여 시스템에 구현함으로써 실현된다.

현존하는 접근통제 방법으로는 임의 접근통제, 강제 접근통제, 역할기반 접근통제가 있는데, 이들 방법을 현재의 다중 사용자, 계층적인 소프트웨어에 적용할 경우, 비효율적인 면이 발생할 가능성이 있다. 따라서 이를 해결하기 위해 새 접근통제 방법의 모색이 필요하다.

이에 본 논문에서는 새 접근통제 방법으로 기존의 역할 기반 접근통제 방법을 확장한 역할-행위 기반 접근통제의 개념과 모델을 제안하고, 이를 다중 사용자, 계층적인 소프트웨어의 대표적인 사례인 이동 에이전트 기반 워크플로우 시스템에 적용, 구현한 결과를 보인다.

논문의 구성은 다음과 같다. 2장에서는 연구 배경으로 기존의 접근통제 방법으로 어떤 것들이 있는가를 살펴보고, 장단점에 대해 알아본다. 3장에서는 2장에서 언급할 접근통제 기법인 역할 기반 접근통제를 확장한 형태인 역할-행위 기반 접근통제에 대해 설명하며, 4장에서 워크플로우 시스템 응용을 위한 이동 에이전트 시스템에 역할-행위 기반 접근통제를 구현한 결과를 제시하고, 5장에서 결론을 맺고 향후 과제에 대하여 언급한다.

Ⅱ. 연구배경

접근이란 접근 주체와 접근 객체 사이의 특별한 형태의 상호작용으로, 이 상호작용의 결과 한 쪽에서 다른 한 쪽으로의 정보의 흐름이 발생하는 것이다. 접근통제란 시스템 자원에의 접근이 발생할 경우, 허가된 프로그램, 사용자 또는 타 시스템에게만 그 접근을 허용하는 과정이다.[1] 현재 다양한 형태의 접근통제 방법이 존재하며, 미 국방성에서는 이들을 크게 임의 접근통제와 강제 접근통제의 두 유형으로 구분했다.[2]

임의 접근통제(DAC: Discretionary Access Control)는 접근 주체 또는 그들이 속한 그룹의 신분에 근거하여 객체에 대한 접근을 통제하는 방법으로, 시스템 자원의 소유자가 자신의 자원에 대한 타인의 접근 행위 허용 여부와 행위의 종류를 임의로 결정할 수 있는 접근통제 방법이다. 이 방법은 접근통제에 있어 유연성을 제공하는 반면, 자원에 대한 접근 권한이 원래의 소유자가 의도하지 않은 경로로 유출될 위험이 있으며, 트로이의 목마 공격에 취약하다는 단점이 있다.

강제 접근통제(MAC: Mandatory Access Control)는 사용자와 시스템 자원을 보안 등급에 따라 분류하고, 접근 행위 발생 시 주체에게 허용된 접근 수준(clearance)과 객체에게 부여된 분류등급(classification)의 비교를 통해 객체에 대한 접근을 통제하는 방법으로, 보안 등급을 명확히 구분할 수 있는 군 관련 정보 등에 적합하나 기업 및 공공기관에 적용하기에는 유연성이 떨어진다는 단점이 있다.

역할 기반 접근통제(RBAC: Role Based Access Control)는 1970년대 다중 사용자, 다중 응용 프로그램 환경에서의 보안 처리 요구를 만족시키기 위해 제안된 방법이다.[1] 역할 기반 접근통제의 요점은 접근 주체인 사용자와 접근 객체인 시스템 자원 사이에 역할 계층을 제공하는 것이다. 역할은 시스템 사용 주체의 조직 구조를 반영하는 개체로서, 시

스템 자원과 자원에 접근하고자 하는 사용자를 잇는 매개체이다. 사용자는 목적 자원에의 접근 권한을 갖는 역할에 속할 때에만 자원에 접근할 수 있다. 즉, 역할 기반 접근통제는 조직 내에서 부여된 개인의 직무에 따라 접근을 통제하는 방법이다. 역할이라는 계층이 사용자와 자원 사용 권한 사이에서 제공하는 사용자 추상성으로 인하여 각각의 사용자에 대한 권한 정의가 단순한 역할 배정으로 완료될 수 있다. 따라서, 역할 기반 접근통제는 사용자가 다수이고 유동적으로 변화하는 환경인 기업 및 공공기관 응용 시스템의 접근통제 방법으로서 현재 가장 선호하고 있다.

2.1 다중 사용자 계층적 구조 시스템 내에서의 접근통제

서론에서 언급한 바와 같이 최근 개발되고 있는 기업 및 공공 기관 응용 소프트웨어들의 대표적인 특징은 크게 다중 사용자, 계층적 구조로 요약된다.

계층적인 구조를 갖는 시스템은 각 독립 계층이 별도로 설계된 후, 계층간 상호작용이 가능하도록 인터페이스를 제공하는 방식으로 구성된다. 별도로 설계되는 각 층은 하위 계층에 관한 구체적인 지식 없이도, 그들이 제공되는 인터페이스의 호출에 의존하여 하위 계층이 제공하는 자원을 사용한다.

이러한 시스템에서 사용자가 요청한 자원 접근은 여러 계층을 통해 이루어지는데, 자원의 접근 경로를 추적해가며, 시스템 전체를 통제하는 유일한 접근통제 시스템을 구성하는 것은 무척 어려운 일이다. 따라서 계층적인 구조를 갖는 복잡한 시스템에서의 접근통제는 각 계층별로 설계되고, 상위 계층의 사용자와 해당 계층 및 하위 계층의 시스템 자원 사이에서 실시된다.

이와 같은 계층 구조, 다중 사용자 시스템에서 임의 접근통제를 실시할 경우, 각각의 접근 주체에 대한 정보를 별도로 관리해야 하므로 비효율적이다. 더구나, 임의 접근통제의 경우, 시스템 자원의 소유권이 접근 주체들에게로 분산되어 있으므로, 기업 및 공공기관 환경에서의 자원 접근통제에 적용하기에는 어려운 점이 있다. 또한, 목적 시스템이 상업용 시스템일 경우, 유연성이 떨어지는 강제 접근통제 방식을 적용하는 것도 현명한 방법이 아니다. 현재 상업용 시스템의 접근통제 방법으로 가장 선호하고 있는 역할 기반 접근통제를 이와 같은 시스템에 그대로 적용할 경우에도 몇 가지 문제점이 있다.

객체에의 접근 권한과 접근 주체의 연관은 보안 관리 업무상 큰 비중을 차지하는 작업이다. 접근통제 시스템에서 이러한 연관은 보안 관리자에 의해 결정된다.

역할 기반 접근통제는 조직 내 구성원의 책임한도, 능력 등을 반영하여 사용자를 추상화한 개체, 즉 역할 개체를 접근 주체와 객체 사이의 매개로 제공하므로, 보안 관리자는 조직에 관한 정보를 토대로 사용자의 위치와 책임한도, 능력 등을 고려하여, 각 역할에의 사용자 할당, 역할과 권한과의 연관을 결정하게 된다. 이때, 보안 관리자는 역할과 사용자와의 연관에 필요한 지식 이외에도, 역할과 자원 접근 권한의 연관을 위해 시스템에 대한 전문적인 지식을 필요로 한다. 그러나, 앞에서 언급한 계층구조를 갖는 시스템의 경우, 시스템은 독립적으로 설계된 이질적인 층들로 구성되므로, 각 계층 사이에는 독립적인 지식의 영역이 존재하기 마련이며, 보안 관리 측면에서 이들을 통괄하는 전문지식을 갖는다는 것은 무척 어려운 일이다.

다중 사용자, 계층적인 구조를 갖는 대표적인 시스템인 이동 에이전트 기반 워크플로우 시스템을 예로 들어 좀 더 구체적으로 설명해 보자. 이동 에이전트 기반 워크플로우 시스템은 하부 이동 에이전트 계층과 상부 워크플로우 시스템 계층으로 구성된다. 이동 에이전트 계층에서 제공되는 접근통제 서비스의 경우, 접근 주체는 기업의 조직 구성원 또는 그들의 권한을 위임받는 워크플로우 시스템 계층의 구성 요소들이며, 접근 객체는 이동 에이전트 객체의 생성, 에이전트 코드 접근, 데이터 접근, 에이전트 객체의 수행 시작 등 이동 에이전트 시스템이 제공하는 인터페이스 형태의 자원이다. 따라서 보안 관리자는 기업 내 조직에 대한 지식과 워크플로우 시스템에 관한 지식 이외에도, 이동 에이전트 시스템에 관한 전문적인 지식을 필요로 한다. 워크플로우 시스템의 사용주체가 기업이고, 보안 관리자는 관리직에 있는 기업의 직원일 것임을 가정할 때, 이 보안 관리자가 기업 구성원과 워크플로우 시스템 구성 요소를 포함하는 접근 주체 집합에서 접근 객체인 이동 에이전트 시스템에 이르기까지 전체 시스템에 대한 전문지식을 갖고 접근통제 시스템을 관리하는 것은 불가능한 일일 것이다. 이러한 경우, 즉 보안 관리자가 접근 객체에 대한 전문적인 지식을 소유하지 않았을 경우, 연관되지 않아야 하는 접근 권한이 어떤 역할에 잘못 할당됨으로써, 예기치 않은 권한의 누설로

인한 접근통제 시스템 전체의 무력화를 유발할 가능성이 존재한다. 예를 들어, 어떤 역할에 에이전트 내의 데이터 갱신을 위한 권한만을 주고자 할 경우, 보안 관리자의 시스템에 대한 지식의 부족은, 그 역할에 에이전트의 생성과 수행 시작의 권한까지 부여하는 요인이 되어, 원래는 의도하지 않았던 에이전트 클론 생성의 권한을 부여하는 결과를 야기할 수 있다. 에이전트 클론 생성은 어떤 경우, 치명적인 결과를 초래할 가능성을 내포하는 권한이다.

시스템이 복잡하게 구성될수록, 이러한 문제는 더욱 심각해지며, 따라서 복잡하고 계층적인 양태를 띠는 시스템 자원을 고려하여 기존 역할 기반 접근통제의 한계를 극복할 수 있는 접근통제 방법이 요구된다.

Ⅲ. 역할-행위 기반 접근통제 방법

앞에서 언급한 바와 같이, 현존하는 접근통제 방법의 유형은 크게 셋으로 나누어 생각할 수 있으며, 그 중 기업 및 공공기관을 대상으로 하는 시스템에 가장 적합한 접근통제 정책은 역할 기반 접근통제 방법이다. 그러나 오늘날의 시스템 형태를 고려해 볼 때, 기존의 역할 기반 접근통제 정책에서 보안 관리자가 수행해야 할 업무의 부담이 늘어나고, 요구되는 지식의 범위가 확대되고 있는 실정이다. 이 장에서는 근래에 개발되고 있는 대규모, 계층 구조를 갖는 시스템에 적용하기 위한 접근통제 방법으로서 기존의 역할 기반 접근통제를 확장한 역할-행위 기반 접근통제를 제안한다.

3.1 역할-행위 기반 접근통제의 개념과 목적

역할-행위 기반 접근통제(RBBAC: Role-Behavior Based Access Control)의 특징은 역할 기반 접근통제에서 유동적인 사용자(User) 집합을 고려하여 역할(Role)이라는 추상계층을 제공했던 것과 마찬가지로, 행위라는 추상계층을 제공하는 것이다. 행위(Behavior)는 기본 권한(Privilege)의 집합이며, 기본 권한은 접근통제가 실시되는 계층의 자원 사용에 관한 권한과, 하위 계층이 제공하는 시스템 자원의 사용에 관한 권한을 의미한다. 기본 권한은 행위 개체로 묶어 추상화되며, 이 때 유연성을 제공하고, 의도하지 않은 권한이 행위 개체에 속하는 것을 방지하기 위하여 궁극적으로 원자성을 가진 기본 권한을 제공하는 것이 바람직하다. 예를 들어, 어떤 파일 시스템에서 기본적인 연산으로 새로운 파일 영역의 생성과 파일의 읽기, 파일의 쓰기 등을 제공한다고 가정하자. 역할-행위 기반 접근통제의 개념에서 기본 권한이란, 각각의 기본적인 연산과 관련하여 주어지는 연산 수행의 권한을 의미한다. 파일 복사는 기존의 파일을 읽고 새로운 파일을 생성한 후, 읽은 내용을 새 파일에 쓰는 파일 시스템의 기본적인 연산들로 구성되는 또 다른 연산이다. 역할-행위 기반 접근통제에서 파일의 복사는 앞에서 열거한 세 개의 기본 권한의 집합인 행위로 정의된다.

역할-행위 기반 접근통제의 목적은 계층적인 보안 관리를 통한 보안 관리상의 편의 추구에 있다. 역할과 행위의 연관은 접근통제가 실시되는 계층의 자원에 접근하는 접근 주체에 대한 배경지식을 갖춘 보안 관리자에 의해 행해지는 것이 바람직하다. 이 때 보안 관리자는 각 역할 고유의 책임 수준을 고려하여 사용자에게 어떤 행위를 허용할 것인가를 결정해야 한다. 행위와 기본 권한의 연관은 접근통제가 실시되는 계층과 하위 계층의 시스템에 관한 전문적인 지식을 가진 보안 관리자에 의해 행해지는 것이 바람직하다. 이 때의 보안 관리자는 어떤 행위에 필요한 최소 기본 권한이 어떤 것인지 고려하여 그 최소 기본 권한 집합만을 행위에 포함시킴으로써, 최소 권한의 원칙을 만족하여 잘못된 연관으로 인한 보안 침해가 일어나지 않도록 주의하여야 한다.

따라서, 역할-행위 기반 접근통제의 또 하나의 목적은 단계적인 보안 시스템 관리를 통해, 접근 주체와 접근 객체간 올바른 접근 권한의 사상을 유도하여 안전한 시스템을 구성하는 것에 있다.

3.2 역할-행위 기반 접근통제 정책

보안 정책이란 어떤 시스템을 보호하기 위한 목적에서 정보가 어떻게 들고나는가, 누가 어떤 조건하에서 어느 정보에 접근할 수 있는가, 시스템 내에서 허용 가능한 정보의 흐름은 무엇인가를 명시하는 기본적인 원칙의 표현이다. 역할 기반 접근통제가 역할 개체를 매개로 정책의 표현이 보다 용이하도록 지원하는 것과 마찬가지로[3] 역할-행위 기반 접근통제 방법 또한 역할, 행위 개체를 통해 정책의 표현을 지원한다. 또한, 행위 개체에 상속의 개념을 도입하거나, 기본 권한과 행위의 관계에 별도의 제약조건을 삽입하여, 보다 정교한 정책을 표현하는 것

도 가능하다.

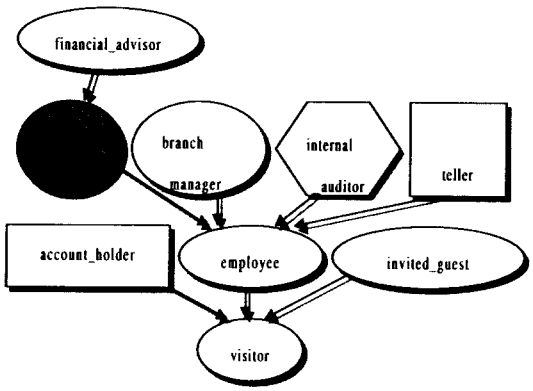
일반적으로 보안 정책은 다음 세 가지 원칙을 만족시킨다.[4]

- 최소 특권의 원칙(Least privilege) : 각 주체는 자신에게 부여된 업무를 완수하는데 필수적인 정보에만 접근할 수 있다.
- 업무 분할(Separation of duties) : 시스템에 위협이 되는 일련의 작업들이 존재한다면, 서로 다른 두 주체에게 작업을 분할하여 수행하도록 한다.
- 역할의 순환(Rotation in role) : 민감한 작업을 한 주체에게 지속적으로 수행하도록 하는 것을 금지한다.

역할-행위 기반 접근통제 방법은 위의 세가지 원칙을 만족시키는 한편, 접근 주체와, 접근 주체에 할당될 수 있는 역할, 각 역할의 수행 가능한 행위, 각 행위를 구성하는 기본 권한과의 관계 설정을 통하여 정보 접근 원칙을 기술하는 용도로 사용할 수 있으므로, 보안 정책으로서의 활용이 가능하다.

역할-행위 기반 접근통제가 위의 세 원칙을 어떻게 만족시킬 수 있는지, 기존연구[5]에서 사용한 은행 지점에서의 접근통제의 예를 사용하여 살펴보기로 하자.

은행에서는 그림 1에서 보는 것과 같이 재무 고문(financial_advisor), 계좌 책임자(account_rep), 지점장(branch_manager), 내부 감사원(internal_auditor), 출납계원(teller), 계좌 소유주(account_holder) 등의 역할이 정의될 수 있다. 화살표는 역할의 상속을 의미하는데, 은행의 계좌 책임자가 각 계좌를 생성, 삭제할 수 있는 경우, 그 역할을 상속받는 재무 고문 역시 각 계좌를 생성, 삭제할 수 있음을 나타낸다.



〔그림 1〕 은행에서의 역할 예

최소 특권의 원칙은 각 사용자에게 주어진 업무를 수행하기 위한 최소한의 역할을 배정하는 동시에 각 역할과 역할을 수행하기 위한 최소한의 행위를 연관시킴으로써 만족될 수 있다. 계정의 생성, 읽기, 쓰기, 삭제의 권한을 포함하는 행위로 계정의 관리를 정의하고, 이미 존재하는 계정의 내용을 읽는 권한만을 포함하는 행위로 계정의 조회를 정의했다고 가정하자. 역할-행위 기반 접근통제에서 계좌 책임자의 역할에는 계정의 관리 행위를 허용하고, 내부 감사의 역할에는 계정의 조회 행위만을 허용함으로써 최소 특권원칙을 만족시킬 수 있다. 내부 감사원은 감사를 위해 기존 계정을 조회하는 것만으로도 책임을 완수할 수 있기 때문에, 계정의 관리 행위를 허용하는 것은 불필요하다. 이와 더불어, 최소 특권의 원칙은 역할-행위 기반 접근통제의 계층 구조 내에서 올바른 행위의 정의를 통해 만족될 수 있다. 만약, 내부 감사원에게 허용된 행위인 계정의 조회에 계정의 삭제를 허용한다면, 내부 감사원에게 불필요한 권한을 허용하는 것이 된다.

업무 분할에는 정적 업무 분할과 동적 업무 분할이 있다. 정적 업무 분할(Static separation of duties)이란, 한 사용자에 대해 두 역할이 동시에 허용되지 않음을 의미한다. 그림 1에서 육각형으로 표시된 내부 감사원은 계좌 책임자와 정적 업무 분할 관계에 있다. 즉, 한 사용자는 계좌 책임자의 역할을 가지면서, 내부 감사원의 역할을 가질 수 없다. 이는 계좌를 책임지는 사람이 계좌를 임의로 조작하는 동시에, 내부 감사원이 되어 자신의 행위를 감추려는 시도를 근본적으로 막기 위한 것이다. 정적 업무 분할은 사용자에게 역할을 허용하는 시점에서 한 사용자에게 두 역할을 허용하지 않도록 하는 동시에, 역할에 행위를 허용하는 시점과 행위에 기본 권한을 허용하는 시점에서 권한의 누설이 생기지 않도록 함으로써 만족된다. 한편, 동적 업무 분할(Dynamic separation of duties)이란, 한 사용자에게 두 역할을 허용하기는 하지만, 동시에 수행하는 것은 금하는 원칙이다. 따라서, 동적 업무 분할은 정적 업무 분할에 비해 상대적으로 약한 업무 분할 원칙이다. 그림 1에서 사각형으로 표시된 계좌 소유주와 출납계원은 각각 계좌 책임자와 동적 업무 분할 관계에 있다. 즉, 계좌 책임자의 역할을 가진 사람은 은행에 계좌를 가질 수는 있으나, 계좌를 처리하는 업무 도중에 자신의 계좌를 처리할 수 없다. 또한, 은행 업무 종료 후, 계좌 책임자가 처리한 당일 입

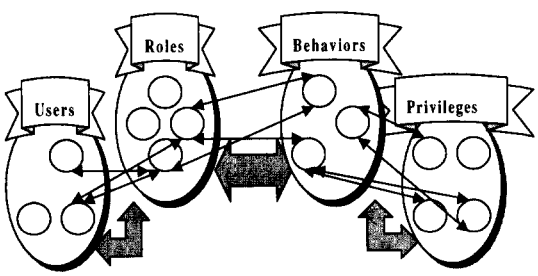
출금의 액수를 확인하고, 현금을 금고로 입금하는 출납계원의 역할과 계좌 책임자의 역할을 한 사용자가 동시에 수행할 수 없다. 동적 업무 분할의 원칙은 수행 시점에서 한 사용자가 두 역할을 수행할 수 없도록 제한을 가하는 동시에, 정적 업무 분할에서와 마찬가지로, 권한이 누설되지 않도록 행위와 역할을 정의함으로써 만족될 수 있다. 계좌 책임자의 역할을 장기간 수행하는 경우, 특정 고객의 프라이버시를 침해하는 경우가 발생할 수 있다. 역할의 순환 원칙은 이러한 상황을 방지하기 위한 원칙이며, 사용자와 역할의 관계에 시간 제약 조건 등 별도의 조건을 부여하여, 일정기간 이상 한 사용자가 동일한 역할을 수행하지 못하도록 원칙적으로 배제하거나, 수행시점에서 사용자의 역할을 정기적으로 재배정함으로써 만족될 수 있다.

따라서, 역할-행위 기반 접근통제는 일반적인 보안 정책의 정의를 만족하는 한편, 앞에서 언급한 세 가지 요구사항을 충족시키는 접근통제 정책으로서의 활용이 가능하다.

3.3 역할-행위 기반 접근통제 모델

이 절에서는 역할-행위 기반 접근통제 방법의 실제 시스템 적용을 위한 단계적 과정으로 역할-행위 기반 접근통제의 모델을 정의한다. 접근통제 모델은 제안한 접근통제 방법의 개념을 좀 더 간결한 형식으로 표현하여, 제안된 방법을 시스템에 구현하는 과정에 발생할 수 있는 혼동을 줄이도록 한다. 단, 현재의 모델은 각 구성 요소간의 관계에 관한 기본적인 정의와 규칙만을 포함하고 있으며, 역할과 행위의 상속, 전이, 제약조건의 삽입 등에 대해서는 정의하고 있지 않다.

역할-행위 기반 접근통제 모델은 사용자(U, User), 역할(R, Role), 행위(B, Behavior), 기본 권한(P, Privilege)의 네 가지 구성요소로 정의된다. 사용자란 접근통제가 실시되는 계층과 그의 하위 계층에서 제공하는 자원을 사용하고자 하는 주체를 의미하며, 사람 또는 사람의 권한을 위임받은 시스템의 구성 요소이다. 역할이란 시스템 내에서 어떤 작업을 할 수 있는 권한을 가진 사용자의 집합이다. 행위란 시스템에서 어떤 역할을 가진 사용자가 수행할 수 있는 작업을 의미하며, 가장 기본적인 연산 또는 시스템 자원에 연관된 사용 권한의 집합이다. 기본 권한이란 시스템이 제공하는 기본적인 연산과 자원 각각의 사용과 연관된 권한이다. 역할-행위 기반 접근통제의 각 요소간의 관계를 도식으로 간략히 보이면 그림 2와 같다.



[그림 2] 역할-행위 기반 접근통제 구성 요소

각 사용자(User)는 하나 이상의 역할(Role)을 소유할 수 있다. 또한 각 역할은 하나 이상의 사용자의 역할일 수 있다. 각 역할은 하나 이상의 행위(Behavior)를 수행할 수 있는 권한을 가질 수 있으며, 각 행위는 하나 이상의 역할에 의해 수행될 수 있다. 각 행위는 하나 이상의 기본 권한(Privilege)을 포함할 수 있으며, 각 기본 권한은 하나 이상의 행위에 포함될 수 있다. 따라서, 각 구성 요소는 다대다(many-to-many) 대응 관계에 있다.

다음은 위에서 논의된 사항을 기존 연구[1]의 형식을 참조하여 간략히 정의한 것이다. 다음 정의들은 집합과 집합간의 관계로 정의되었으며, 구현을 위한 특별한 접근통제 메커니즘을 기술하고 있는 것은 아니다.

Definition RBBAC 모델은 다음과 같은 요소들로 구성된다.

- U, R, B, P : users, roles, behaviors, privileges의 집합
- O : resource objects의 집합
- $RA \subseteq U \times R$: user에의 role 할당
- $BA \subseteq R \times B$: role에의 behavior 할당
- $PA \subseteq B \times P$: behavior에의 privilege 할당
- $ar(u) = \{$activated roles for user u$\}$
- $ra(u) = \{$authorized roles for user u$\}$
- $ab(r) = \{$activated behaviors for role r$\}$
- $ba(r) = \{$authorized behaviors for role r$\}$
- $ap(b) = \{$activated privileges for behavior b$\}$
- $pa(b) = \{$authorized privileges for behavior b$\}$

(단, u는 user, r은 role, b는 behavior를 나타낸다.)

여기서 유념해야 할 사항은 활성화되었다(activated)는 개념과 허용되었다(authorized)는 개념의 차이이다. 어떤 사용자에게 허용된 역할이라는 것은, 그 사용자에게 할당되는 것이 가능한 역할을 말한다. 반면, 어떤 사용자의 활성화된 역할이라는 것은 사용자의 접근 행위가 이루어지는 시점에서 사용자에게 할당되어 있는 역할을 말한다. 따라서, 개체 사이에서 허용된 관계라는 것은 정적으로 결정되는 관계이며, 활성화된 관계라는 것은 상대적으로 동적으로 결정되는 관계라 할 수 있다. 일반적으로 어떤 사용자의 활성화된 역할은 허용된 역할의 부분 집합이다. 따라서, 다음의 세 규칙을 유도할 수 있다.

① 활성화된 역할 : 사용자의 활성화된 역할은 해당 사용자에게 허용된 역할이어야 한다.

$\forall u$에 대해 $\boldsymbol{ar}(u) \subseteq \boldsymbol{ra}(u)$

② 활성화된 행위 : 역할의 활성화된 행위는 해당 역할에게 허용된 행위여야 한다.

$\forall r$에 대해 $\boldsymbol{ab}(r) \subseteq \boldsymbol{ba}(r)$

③ 활성화된 기본 권한 : 행위의 활성화된 기본 권한은 해당 행위에게 허용된 기본 권한이어야 한다.

$\forall b$에 대해 $\boldsymbol{ap}(b) \subseteq \boldsymbol{pa}(b)$

접근 행위란, 궁극적으로는 사용자와 시스템 자원 사이의 관계이다. 그러므로, 다음과 같이 접근 행위를 의미하는 술어인 ***access***(u,o)를 정의할 수 있다.

access(u, o) = ***TRUE*** iff user can access to resource object o
(단, o는 resource object 이다.)

접근 행위의 정의를 이용하여, 사용자의 시스템 자원 접근 행위에 관한 다음의 규칙들을 정의할 수 있다.

역할의 할당 : 올바른 접근 행위를 위하여 사용자는 활성화된 역할을 가져야 한다

$\forall u, \forall o$에 대해서 $\boldsymbol{access}(u,o) \Rightarrow \boldsymbol{ar}(u) \neq \varnothing$

행위의 할당 : 올바른 접근 행위를 위하여 역할은 활성화된 행위를 가져야 한다.

$\forall u, \forall o$에 대해서 $\boldsymbol{access}(u,o) \Rightarrow \exists r \in \boldsymbol{ar}(u)$ such that $\boldsymbol{ab}(r) \neq \varnothing$

기본 권한의 할당 : 올바른 접근 행위를 위하여 행위는 활성화된 기본권한을 가져야 한다.

$\forall u, \forall o$에 대해서 $\boldsymbol{access}(u,o) \Rightarrow \exists r \in \boldsymbol{ar}(u)$ and $\exists b \in \boldsymbol{ab}(r)$ such that $\boldsymbol{ap}(b) \neq \varnothing$

따라서, 최종적으로 접근 행위에 관한 규칙을 다음과 같이 정리할 수 있다.

$\forall u, \forall o$에 대해서 access(u,o) $\Rightarrow \exists r \in ar(u)$ and $\exists b \in ab(r)$ and $\exists p \in ap(b)$

여기서 유의해야 할 것은, 접근행위에 관련된 규칙을 정의하는 과정에서 사용된 충분조건의 표기이다. 접근통제 시스템에서, 접근 행위의 허용 여부를 판별하는 과정에서 여러 제약 조건을 추가로 적용하는 것이 가능하므로, 필요충분조건의 표기는 사용하지 않았다.

3.4 역할 기반 접근통제와의 비교

본 논문에서는 보안 관리자의 역할을 논하는 과정에서, 접근 주체와 접근 객체간의 올바른 연관에 중점을 두었다. 올바른 연관이란, 접근 행렬을 만들었을 때, 접근 주체의 행과 접근 객체의 열이 교차하는 지점에 주체에게 허용된 권한이 정확히 정의되어 있음을 의미한다.

앞에서 이미 언급한 바 있듯, 계층적인 시스템 내에서 역할 기반 접근통제 방법을 적용할 경우, 올바른 연관을 위하여 보안 관리자는 시스템의 구조에 관하여 해박한 지식을 갖추어야만 했다. 그러나, 이러한 지식을 갖추기가 어려울 뿐 아니라, 시스템 전체 구조를 이해하는 전문지식을 갖추고 있더라도, 자원이 복잡하게 정의될 경우 실수로 인한 권한의 누설이 생기거나 연관 설정 시 오류를 범할 위험은

〔표 1〕 가상 은행 시스템에서의 기본권한, 행위, 역할 정의 예

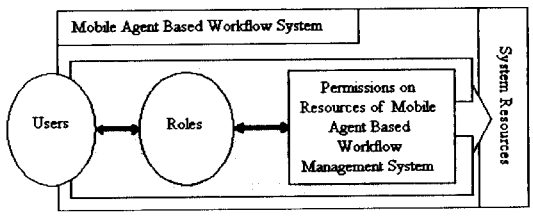
역 할	행 위		기본 권한
출납계원	평사원급 예금 업무	하급 개인입출금	예금 잔고 조회
			예금 입출금 조회
			자금 이체 처리 결과 조회
			예금 입금 데이터 기록
			예금 출금 데이터 기록〈200만원
		어음	수표 사고 조회
			보관 어음 명세 조회
		채권	국채 명세 조회
		하급 대출	대출 거래 계좌 내역 조회
			대출〈1000만원
	평사원급 자금이체	하급 즉시이체	동행내 자금 즉시 이체〈1000만원
			타행간 자금 즉시 이체〈1000만원
		하급 예약이체	동행내 자금 예약 이체〈1000만원
			타행간 자금 예약 이체〈1000만원
	안내	기타 안내	
	…		
과장	간부 사원급 예금 업무	중급 개인입출금	예금 출금 데이터 기록〈1000만원
		중급 대출	대출〈5000만원
	간부 사원급 자금이체	중급 즉시 이체	동행내 자금 즉시 이체〈5000만원
			타행간 자금 즉시 이체〈5000만원
		중급 예약이체	동행간 자금 예약 이체〈5000만원
			타행간 자금 예약 이체〈5000만원
		중급 일괄 이체	동행간 자금 일괄 이체〈5000만원
			타행간 자금 일괄 이체〈5000만원
	…		
부장	간부 사원급 예금 업무	고급 개인 입출금	예금 출금 데이터 기록〈1억원
		고급 대출	대출〈1억원
	간부 사원급 자금 이체	고급 즉시 이체	동행내 자금 즉시 이체〈5000만원
			타행간 자금 즉시 이체〈5000만원
		고급 예약이체	동행내 자금 예약 이체〈5000만원
			타행간 자금 예약 이체〈5000만원
		고급 일괄이체	동행내 자금 일괄 이체〈5000만원
			타행간 자금 일괄 이체〈5000만원
	간부급 신용 조회	개인 신용 조회	개인 신용 조회
	통지	거래 내역 통지	예금 거래 내역 통지
		연락	만기일, 잔고 부족 통지
	…		
지점장	간부 사원급 예금 업무	최고급 개인 입출금	예금 출금 데이터 기록〉=1억원
		최고급 대출	대출〉=1억원
	간부급 자금 이체	최고급 일괄이체	자금 일괄 이체〉=5000만원
	지점장급 신용 조회	신용 정보 조회	모든 고객 신용정보 조회
	지점장급 관리자 조회	관리자 정보 조회	모든 사원 정보 조회
	…		
감사		감사 조회	모든 업무 관련 정보 조회
	…		

여전히 존재한다. 이는 역할 기반 접근통제가 다중 사용자 환경을 염두에 두고 역할 개체를 제안하였기 때문인 것으로 사료된다. '역할'은 주체와 객체간 의미상의 차이를 줄이기 위하여 제안된, 접근 주체를 추상화하는 동시에 접근 객체를 추상화하는 계층이기는 하지만, 기존 연구[7,5,3]에서 나타나듯, 주로 주체의 조직 구조를 반영하는 방식으로 사용되기 때문이다. 주체의 추상화를 담당하는 유일한 추상 계층인 역할 개체 통하여 시스템 자원을 효율적으로 추상화하기 위해서는 의미상의 차이를 감수해야만 한다.

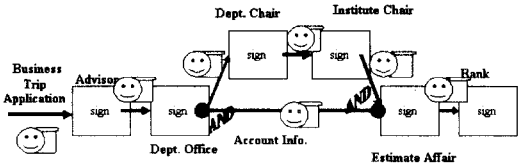
따라서, 기존 역할 기반 접근통제 정책만으로는, 계층 구조화된 복잡하고 다양한 시스템 자원을 효율적으로 분류하기가 어렵다. 이는 곧 보안 관리의 부담으로 나타나며, 올바른 연관을 보장할 가능성을 감소시키는 원인이 된다.

표 1은 참고자료[8]를 토대로, 홈뱅킹 서비스를 제공하는 가상 은행의 업무와 관련된 연산들을 분류한 도표이다. 가장 우측에 열거된 연산들은 시스템에서 처리되는 연산들과 관련한 기본 권한들에 일대일 대응된다고 가정하자. 가상 은행의 한 지점 내에 출납계원, 출납과장, 대출과장, 부장 등의 역할이 존재한다고 할 때, 정의되는 연산의 수와 종류가 많을 경우 보안 관리자가 역할과 권한 사이의 잘못된 연관을 생성할 가능성이 존재한다. 역할-행위 기반 접근통제는 보안 관리상 편의를 제공하기 위하여, 자원을 행위로 추상화한다. 표의 좌측 부분에 기본 권한을 분류한 후, 임의의 이름을 부여하여 행위를 구성한 결과를 열거하였다.

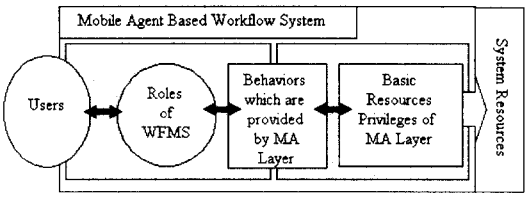
앞에서 사용된 이동 에이전트 기반 워크플로우 시스템의 예를 다시 한 번 떠올려보자. 시스템은 크게 이동 에이전트 하위 계층과, 워크플로우 시스템 계층의 두 계층으로 설계되어 있다. 그림 3은 이 시스템에 기존의 역할 기반 접근통제를 적용한 것을 도식화 한 그림이다. 어둡게 보이는 영역은 시스템에 관한 상세한 이해를 바탕으로 보안 관리가 행해져야



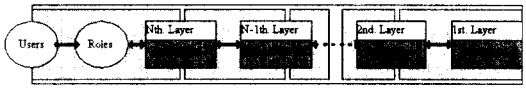
〔그림 3〕 이동 에이전트 기반 워크플로우 시스템의 역할 기반 접근통제



〔그림 4〕 출장신청 워크플로우 예제



〔그림 5〕 이동 에이전트 기반 워크플로우 시스템의 역할-행위 기반 접근통제



〔그림 6〕 계층구조 내에서의 역할-행위 기반 접근통제 확장

하는 부분을 나타낸 것이다. 보안 관리자가 두 시스템에 대한 전문지식을 소유하지 않았을 경우, 발생할 수 있는 잘못된 연관은 다음과 같은 상황을 야기할 가능성이 있다. 그림 4는 출장 신청 워크플로우를 나타낸다. 학생 S1은 출장을 가기 위해 출장 신청서를 작성하고 워크플로우 시스템에 제출한다. 워크플로우 시스템은 이동 에이전트 시스템에 의뢰하여 워크플로우의 수행을 책임질 이동 에이전트 객체를 생성한다. 여기서 학생 S1은 워크플로우의 수행을 의뢰하는 일반 사용자의 역할로 워크플로우 시스템에 참여했다. 보안 관리자가 이 일반 사용자 역할에 해당 역할이 가져야 할 기본 권한과 더불어 이동 에이전트 객체의 데이터 접근과 수정에 관한 권한을 할당했다고 하자. 학생 S1은 예산과에서 은행으로 이동 에이전트가 이동할 때, 이동 에이전트 객체를 가로챈 후, 출장비 항목을 수정하는 것이 가능하다. 이러한 잘못된 권한은 관계 설정 시 직접 할당될 수도 있고, 다른 권한에 포함되어 부적절한 역할에 할당될 수도 있다.

그림 5는 이동 에이전트 기반 워크플로우 시스템에 제안된 역할-행위 기반 접근통제를 적용한 경우를 도식화하여 나타낸 것이다. 그림에서 왼쪽 어두운 영역은 사용자 추상화가 이루어지는 영역으로 사용자와 워크플로우 시스템에 관한 지식을 가진 보안 관리자에 의하여 관리될 수 있는 영역이다. 반면, 오른쪽 영역은 자원의 추상화가 이루어지는 영역으로 이동 에이전트 시스템 계층에 관한 전문지식을 가진 보안 관리자에 의해 관리될 수 있는 영역이다.

이처럼 역할-행위 기반 접근통제 기법은 시스템 자원의 단계적인 분류와 그들 간의 접근통제에 편의를 제공한다. 각 계층에서의 보안 관리는 하나 이상의 보안 관리자에 의해 단계적이고, 독립적인 방식으로 수행될 수 있다. 또한, 역할-행위 기반 접근통제의 '행위'는 계층간 관계의 설정에서 출발하므로 그림 6과 같이 다단계 확장이 가능하다. 그림 6의 목적 시스템은 각 계층이 하위 계층에서 제공하는 자원을 사용하도록 설계되어 있는 일반적인 계층적 구조를 갖는 시스템이다. 자원의 계층적 구조에 의존하는 이와 같은 시스템 조직의 수용은 접근 주체의 추상화에 근거를 두는 기존의 역할 기반 접근통제에서는 개념상의 무리를 감수해야하는 일이었다.

점점 복잡해져 가는 시스템 구조를 고려할 때, 개인에게서 시스템 내부 각 계층에 대한 전문지식을 기대하기 어렵고, 비 형식적인 절차, 단계적이지 않은 협의에 의해 결정된 권한의 연관이 올바를 것이라 확신하기 어렵다. 역할-행위 기반 접근통제는 이와 같은 기존 역할 기반 접근통제의 단점을 극복하고 보안 관리의 부담을 경감시키려는 목적으로, 접근통제 정책적 측면에서 계층적인 보안 관리 업무를 지원하기 위한 방법이다. 즉, 역할-행위 기반 접근통제는 기존의 역할 기반 접근통제에 비하여 자원에 대한 계층적인 접근을 추구하고 있다. 이로써 얻을 수 있는 장점을 정량화 하여 평가하기는 어렵겠지만, 기존의 방법과 비교하여 얻을 수 있는 효율의 개선을 충분히 유추할 수 있다. 네트웍 시스템의 경우, 물리 계층에서 응용 계층까지 여러 계층으로 나뉘어 설계되는 것이 보통이다. 이때, 우리는 수치로 환산할 수 있는 어떤 성능의 향상을 기대하는 것은 아니다. 즉, 설계와 관리의 편의를 위해 추상화와 계층적 접근 방법을 사용한다. 제안된 방법은 이러한 정성적인 효율의 개선에 중점을 두고 있다.

이번 장에서는 기존의 역할 기반 접근통제 방법이 갖는 문제점을 해결하기 위하여, 기존의 역할 기반 접근통제 방법에 행위라는 계층을 추가한 역할-행위 기반 접근통제 방법을 제안하였으며, 제안한 접근통제 방법을 실제 시스템에 적용하기 위한 단계적 절차로서 역할-행위 기반 접근통제 모델을 제안하였다.

역할-행위 기반 접근통제가 실제 시스템에 적용될 경우, 보안 관리는 단계적으로, 보다 정교하게 실시되며, 의도하지 않은 권한의 할당으로 인한 보안 시스템에의 위협을 감소시킨다.

현재, 제안된 역할-행위 기반 접근통제 방법의 가장 기본적인 사항만을 정의한 상태이나, 앞으로 역할 개체의 상속, 행위 개체의 상속을 지원하는 한편, 기존의 연구[1,6]에서 제시한 제약조건 등의 개념을 도입하여 현재의 모델을 점차 발전시켜 나갈 예정이다.

Ⅳ. 구 현

이번 장에서는 3장에서 제안한 접근통제 방법을 다중 사용자, 계층 구조 시스템의 대표적 예인 이동 에이전트 기반 워크플로우 시스템의 이동 에이전트 하부구조에 구현한다.

4.1 이동 에이전트 기반 워크플로우 시스템

워크플로우란 업무에 관련된 문서, 정보, 단위업무 등이 정의된 규칙에 따라 자동으로 전달 되도록 비즈니스 프로세스의 전체 혹은 일부를 자동화한 것이다. 워크플로우 시스템은 이러한 워크플로우의 생성, 수행을 관리하는 시스템으로, 프로세스 정의, 작업 참여자와의 상호작용, 응용프로그램의 호출 등을 담당하는 여러 소프트웨어들로 구성된다.[9]

워크플로우 시스템은 기업의 업무를 전산화하는 과정에서 비즈니스 프로세스 및 정보 프로세스 처리의 자동화와 재사용을 통한 생산성 향상을 목표로 개발되어, 현재 수백에 이르는 워크플로우 지원 도구들이 존재하고 있다. 그러나, 이러한 기존의 워크플로우 시스템들이 상호 운용성, 확장성, 병행성, 가용성등에 취약하다는 단점을 지니고 있다고 지적되어,[10] 현재 이를 해결할 새로운 방법의 모색이 필요하다.

이동 에이전트란 주어진 문제의 해결을 위하여 이질적인 네트워 환경의 호스트와 호스트 사이를 이동하며 사용자 요구 연산을 수행하는 동적 소프트웨어 객체이다. 이동 에이전트 서버는 이동 에이전트 객체를 생성하고, 실행시키며, 다른 이동 에이전트 서버로 전송을 하는 등, 수행 환경을 제공한다. 이동 에이전트 시스템은 이동 에이전트들과 이동 에이전트 서버들의 집합으로 구성된다. 이동 에이전트 기술은 분산환경, 이동 사용자 환경에 적합한 프로그래밍 방식으로 관심이 고조되고 있는 기술이며 이동 에이전트를 이용하면 다음과 같은 장점을 기대할 수 있다.

- 이동 에이전트는 수행하던 일을 멈추고, 원격 서버로 이동하는 시점을 스스로 결정할 수 있으므로, 신뢰성이 떨어지는 네트워 환경에 잘 적응할 수 있다.
- 이동 에이전트는 원격 서버에 메시지를 보내고 결과를 받는 기존의 통신방식과는 달리, 원격 서버로 직접 이동한 후 지역 네트워 안에서 메시지를 보내고 결과를 받으므로, 원격 메시지 교환의 횟수를 줄일 수 있다.
- 이동 에이전트는 서버로 이동한 후, 서버에 의존하여 수행된 후, 반환되는 프로그램이므로 경량화된 사용자 컴퓨팅 환경을 지원하기에 적합하다.
- 이동 에이전트는 이질적인 호스트간의 복잡한 통신 프로토콜을 추상화하는 수단이 된다.
- 비동기 수행을 지원한다.

이동 에이전트의 이런 장점을 이용하여, 기존 워크플로우 시스템의 문제점을 해결하려는 노력이 현재 진행 중이며, 대표적인 예로 Dartflow[11]를 들 수 있다.

본 논문에서 앞으로 언급할 이동 에이전트 기반 워크플로우 시스템이란, 다음과 같이 두 계층으로 구성된 시스템을 의미한다.

- 워크플로우 시스템 계층 : 일반적인 워크플로우 시스템과 마찬가지로, 워크플로우 엔진, 워크리스트 핸들러, 사용자 인터페이스, 워크플로우 정의 도구로 구성되어 있다. 특정 워크플로우의 수행을 워크플로우 사용자가 의뢰하게 되면, 워크플로우 엔진은 해당 작업을 수행할 이동 에이전트의 생성을 이동 에이전트 계층에 의뢰하게 된다. 수행 중인 작업의 진행 상황 모니터링이 요구되는 경우, 작업 종료 시, 동적 변경 시에는 이동 에이전트와의 정보교환을 통해 요청 작업을 수행한다.
- 이동 에이전트 계층 : 워크플로우 계층의 작업 수행 요구에 따라 사용자 의뢰 작업을 담당할 이동 에이전트를 생성하며, 필요에 따라 부 에이전트를 생성하거나 원격 이동 에이전트 서버로 이

동하며 사용자 요청 작업을 수행한다.

이동 에이전트 기반 워크플로우 시스템에서 역시 일반적인 이동 에이전트 기반 시스템과 마찬가지로 이동 에이전트의 보안 문제를 떼어 생각할 수 없다. 이동 에이전트의 보안 문제를 해결하기 어렵게 하는 문제로 만드는 가장 큰 원인은 믿을 수 없는 이동 에이전트 서버의 존재에 있다. 이동 에이전트가 개방된 네트웍 환경에서 수행되는 경우, 인증 받지 않은 이동 에이전트 서버를 방문할 가능성을 염두에 두고 있으므로, 이동 에이전트를 보호하려 할 때 심각한 문제에 부딪히게 된다. 워크플로우 시스템은 전술했듯 어떤 기업, 또는 공공기관 등에서 수행되는 응용프로그램이다. 따라서 워크플로우 시스템이 신뢰할 수 있는 서버의 집합으로 구성된 닫힌 네트웍 환경 안에서 수행됨을 가정할 수 있다. 이동 에이전트 서버가 믿을 수 있다고 가정하면, 악의를 가진 인증 받지 않은 이동 에이전트 서버에 의한 지속적 관찰의 결과 에이전트가 공격당하는 경우와 같은 상황을 배제하고 이동 에이전트의 보안 문제를 고려할 수 있게 된다. 따라서, 워크플로우 시스템 안에서 이동 에이전트 시스템에서의 보안 문제가 축소된다.

위의 같은 가정을 하더라도 안전한 수행을 위하여 이동 에이전트 기반 워크플로우 시스템에서 제공해야 할 보안 서비스는 다양하다.[12] 하지만, 본 논문의 목적이 역할-행위 기반 접근통제에 한정되어 있는 만큼, 접근통제를 제외한 여타의 보안 서비스들은 별도의 도구에 의해 지원된다고 가정한다.

4.2 기존 에이전트 시스템의 접근통제 방법

현재까지 많은 이동 에이전트 시스템이 기업 및 대학들에 의해 개발되어 왔다. 그러나, 아직까지 자원 접근을 통제하는 시스템이 구현된 이동 에이전트 시스템은 그리 많지 않은 실정이다.[13] 또한, 현존하는 에이전트 시스템은 범용으로 설계되어 광범위한 응용을 목적으로 하고 있으므로, ACL을 바탕으로 접근 규칙 기술 언어 등을 도구로써 제공하는 것과 같은 접근통제 서비스를 제공하고 있다. 범용 이동 에이전트 시스템의 대표적인 예로 IBM사의 Aglet™ 시스템을 들 수 있으며, Aglet의 접근통제 시스템은 다음과 같이 접근통제 규칙을 기술하는 Authorization Language를 정의하고 있다.[13]

```
TRUSTED :
manufacturer=Athena OR master=Hades ->
File / tmp read, write
Net TCP Underworld Accept
Top level window 3
Aglet owner=Leda retract
GUEST :
manufacturer=Hermes ->
Net message Olympus receive
Aglet Property get,set
REJECT :
manufacturer=Hermes, Titans ->
Context NOT enter
```

이러한 방법은 접근 주체와 객체 각각에 대한 권한의 정의가 이루어지는 임의 접근통제에 비해 좀 더 유연한 구조를 갖는 접근통제 정책의 형식이긴 하지만, 이 또한 보안 정책을 정의하는 사람이 접근 행위에 참여하는 모든 개체들에 관한 특성을 파악한 후, 그들의 식별자나, 주소, 관리자 식별자 등의 특징을 이용하여 규칙을 표현할 수 있는 경우에만 가능한 방법이다. 그러나, 워크플로우 시스템에서 접근 가능성을 가진 주체의 집합, 접근 객체 자원이란 한 번에 파악되고 쉽게 분류되는 대상이 아니다. 따라서, 일반적인 이동 에이전트 시스템의 접근통제 시스템을 워크플로우 시스템 구성에 이용하기에는 불합리한 점이 있다.

따라서, 본 논문에서는 이동 에이전트 기반 워크플로우의 접근통제 방법으로 3장에서 제안한 역할-행위 기반 접근통제를 적용한다. 역할-행위 기반 접근통제는 이동 에이전트 기반 워크플로우 시스템의 사용주체인 기업 및 공공기관 내의 조직 구조를 반영하며, 계층적으로 설계되는 시스템 환경에서 접근통제를 실시할 경우 발생 가능한 오류 감소, 관리 편의 등의 이점을 제공한다.

4.3 X-MAS 이동 에이전트 시스템

이동 에이전트 시스템 X-MAS는 이동 에이전트 기반 워크플로우 시스템을 위해 광주과학기술원에서 현재 개발 중인 에이전트 시스템이다.[15]

X-MAS의 이동 에이전트와 이동 에이전트 서버의 기능은 일반적인 이동 에이전트 시스템과 크게 다르지 않으며, 각각의 기능을 간략히 적으면 다음과 같다.

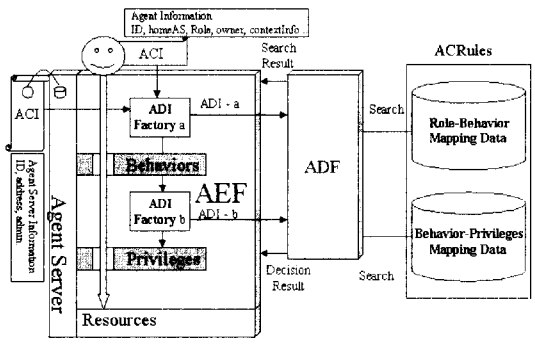
- 이동 에이전트 : 이동 가능한 형태의 프로그램으로 코드와 데이터 부분으로 분리되어 있다. 서버에서 생성이 완료되면 Thread의 형태로 수행된다.
- 이동 에이전트 서버 : 이동 에이전트 객체를 생성하고, 다른 서버로 전달하거나, 다른 서버로부터 받아 수행한다.

사용자가 X-MAS를 사용하여 사용자 에이전트를 만들고자 할 경우에는, 이동 에이전트 객체를 상속하여, 에이전트 고유의 내용을 결정하는 action() 함수의 부분에 원하는 행위를 추가한다. 현재 X-MAS는 Proxy, 스케줄러 모듈 등 이동 에이전트 기반 워크플로우 시스템의 구성에 필요한 요소들이 추가되어 확장되는 중이나, 본 논문에서는 이동 에이전트와 이동 에이전트 시스템으로 범위를 한정하여, 접근통제 모델을 설계하였다. 추후, 확장을 위한 설계가 종료되고 구현이 완료된 이후, 접근통제 구조 역시 확장될 계획이다.

4.4 접근통제 구조

X-MAS의 접근통제 시스템은 ISO의 표준안[14]을 기반으로 시스템에 맞게 변형, 그림 7과 같이 구성된다.

- AEF(Access Enforcement Function) : AEF는 접근통제에 필요한 함수들을 제공한다. AEF가 제공하는 함수들은 check function이라 하며, 이동 에이전트 시스템 내부에서 접근 행위가 발생할 때, 호출되어 접근 행위가 타당한지 여부를 판단하는 역할을 한다. 접근 행위가 허용되는 행위인 경우, check function은 아무런 결과도 내놓지 않지만, 허용되지 않는 접근 행위인 경우, 예외상황을 발생시킨다. 에이전트 시스템은 check function호출 후, 예외상황이 반환되는지를 검사하여 불가한 접근 행위를 탐지할 수 있게 된다.
- ACI(Access Control Information) : ACI는 이동 에이전트와 이동 에이전트 서버가 소유한 고유정보로 접근통제 시스템에서 사용하기 위하여 추출되는 정보이다.
- ADIFactory(Access Decision Information Factory) : ADIFactory는 접근 주체와 객체로부터 접근통제에 필요한 정보(ACI)를 추출하여 접근통제 구조 내에서 사용되는 정보의 형태인 ADI를 반환한다.
- ADI(Access Decision Information) : 접근통제 과정에서 사용되는 정보로 접근 주체와 접근 객체에 관한 정보, 접근이 발생하는 시점의 시간등 부가정보를 포함한다.
- ADF(Access Decision Function) : ADF는 AEF로부터 넘겨받은 ADI를 바탕으로 해당 접근 행위가 허용되는지 여부를 판별하는 부분이다.
- ACRules(Access Control Rules) : ACRules는 접근통제 과정에서 사용되는 규칙들을 접근하기 위한 인터페이스이다. 현재의 구현에서 각 규칙들은 단순히 파일을 통해 저장된다.

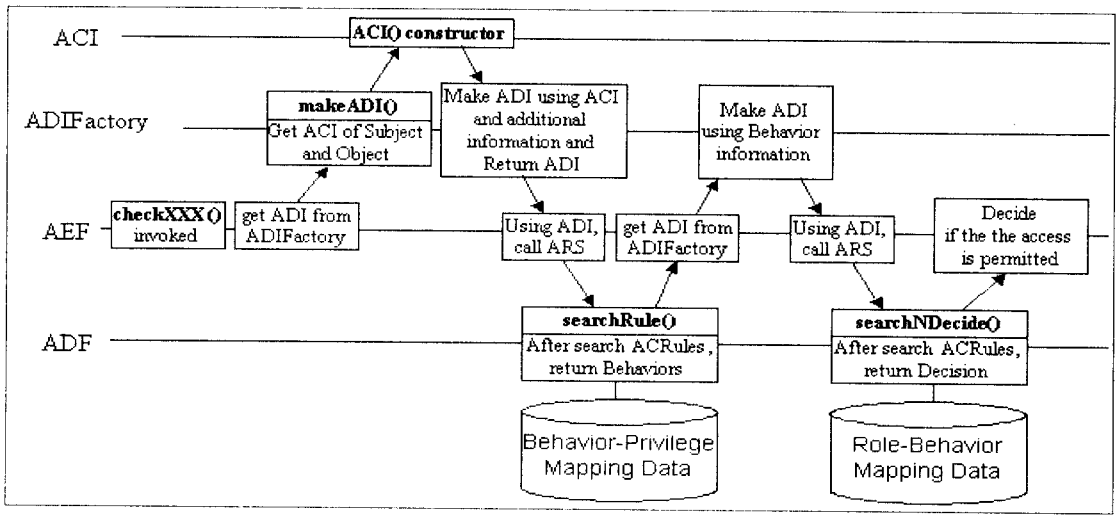


〔그림 7〕 접근통제 시스템의 구조

RBBAC은 앞에서 설명했듯 두 단계의 mapping을 거쳐 행해진다. 따라서 위의 그림과 같이 AEF는 ADF를 두번 호출하게 되며, 각 단계에 사용되는 정보가 다르므로 ADI는 두번의 과정을 거쳐 조합된다.

4.5 접근통제 처리 과정

접근통제가 일어나는 과정을 설명하면 다음과 같다. AEF가 제공하는 check function이 에이전트 시스템 수행 과정에서 호출되면 AEF는 ADIFactory에게 ADI를 구성하도록 요구한다. ADIFactory는 접근 행위에 참여하는 개체들의 정보를 ACI형태로 변환하여 얻은 후, 접근 행위가 어떤 Privilege에 해당하는지를 토대로 접근 주체와 접근 객체를 분류하고, 접근 시간 등의 정보와 함께 조합하여 ADI를 구성한다. AEF는 얻은 ADI를 ADF에게 전달 하며, ADF는 전달 받은 ADI의 정보를 토대로 접근 행위



〔그림 8〕 접근통제 수행 과정

에 해당하는 Privilege가 존재하는지를 검사하고, 해당 Privilege를 포함하는 Behavior를 반환한다. AEF는 ADI가 반환한 Behavior와 이전의 ADI를 ADIFactory에게 넘겨 다시 ADI를 구성하고, ADF에게 넘긴다. ADF는 ADI에 포함된 Behavior와 접근 주체의 Role, 그 밖의 접근 행위의 허용을 결정하는 정보를 토대로 주체에게 Behavior에의 접근이 허용되는지를 검사하여 접근 행위의 허용 여부를 반환한다. AEF는 반환된 최종결과를 확인한 후, 불허된 접근 행위일 경우 예외상황을 발생시킨다. 위의 과정은 그림 8에 나타나 있다.

4.6 구현 결과

구현 결과는 다음과 같은 시나리오를 가정한 후 얻어진 결과다.

시나리오 :

① 사용자 U1은 서버 S1의 워크플로우 시스템에 접속, 출장 신청 워크플로우 WF의 수행을 의뢰한다.

② 서버 S1의 워크플로우 엔진은 하위 이동 에이전트 시스템 AS1으로 하여금, 출장 신청 워크플로우 WF의 수행을 책임질 이동 에이전트 객체를 생성하도록 요구한다.

③ AS1은 템플릿 파일의 형태로 데이터 저장소에 보관되어 있는 출장 신청 워크플로우 관련 에이전트 정보를 읽어 메모리에 로드한다. 메모리에 로드된 이동 에이전트 M1은 에이전트 코드의 내용에 따라, 서버 AS1에서 출발, AS2, AS3, AS4등으로 이동하면서 사용자가 의뢰한 워크플로우 WF를 수행한다. 사용자 U2는 사용자 U1이 신청한 워크플로우의 participant로서, M1이 WF를 수행하기 위하여 AS2를 방문할 때, 이동 에이전트 데이터에 서명을 함으로써 작업에 참여한다.

④ 워크플로우의 수행이 완료되면 M1은 AS1으로 돌아와 결과를 보고한 후 소멸한다.

위의 시나리오에서 다음과 같은 역할들을 정의할 수 있다.

- User U1 : WorkflowExecutionRequester
- User U2 : WorkflowTaskPerfomer
- 이동 에이전트 M1 : WorkflowExecutionAgent

워크플로우 WF의 수행 도중, 이동 에이전트 서버 AS1과 서버 AS2는 사용자 U1과 U2의 권한을 위임받게 된다.

위의 시나리오에서 보듯, 처음 워크플로우의 수행 요구가 일어나는 서버는 이동 에이전트 파일을 읽어 새로운 이동 에이전트를 생성할 수 있다. 그러나, 중간의 participant 서버는 새로운 이동 에이전트 파일을 읽어 에이전트를 생성할 수 없다.

새로운 이동 에이전트 객체를 만들기 위해 이동 에이전트 템플릿으로부터 정보를 읽는 권한은 Agent-FileRead이라는 권한으로 정의된다. 읽어들인 중간언어 형태의 에이전트 코드로부터 에이전트 객체를 생성하는 권한은 AgentCreate이다. 따라서, Agent-Create은 에이전트 LifeCycle을 제어하기 위한 필수적인 권한이다. 이들 권한은 에이전트 시스템에 관한 전문 지식이 없는 보안 관리자에게 혼동을 야기할 수 있다. 에이전트 서버에게 필요한 권한이 이들 외에도 AgentInitiate, AgentBorn, AgentExport, AgentImport등 다양하고 복잡함을 생각할 때 이러한 가능성은 충분하다. 현재 시스템에 역할 기반 접근통제가 실시되고 있을 경우, 위에서 언급한 혼동에 의하여 participant 서버가 AgentFileRead의 권한을 배정받을 수 있으며, 따라서 에이전트 수행 도중, 다른 사용자 U3가 participant 서버 AS3에서 AgentFileRead의 권한을 이용하여 자신의 이동 에이전트를 생성, 복수 실행시킨 후, 결과를 가로채는 결과를 낳을 수 있다. 그러나, 역할-행위 기반 접근통제가 적용될 경우, 이러한 권한들은 이동 에이전트 시스템에 관한 전문지식을 소유한 보안 관리자에 의해 NewAgent-FileReadAndCreate와 AgentInstanceCreate라는 Behavior로 나뉘어 제공되어, 혼동의 위험을 방지하게 되며, 워크플로우 시스템 계층의 보안 관리자가 필요한 최소 권한 집합만을 제공하도록 수단을 제공하게 된다.

그림 9, 그림 10은 TravelingAgent 프로그램을 수행하고, 접근통제 상황을 출력한 결과이다. TravelingAgent는 이동 에이전트 서버 SUNI-HILL에서 생성된 후 "My name is Traveling Agent"라는 메시지를 출력하고, 다른 이동 에이전트 서버 NADIA를 방문하여 "I just arrived at this machine, my name is Traveling Agent"를 출력한 후, 다시 원래의 홈 서버인 SUNIHILL로 돌아와 "Here is my home, Bye"라는 메시지를 출력하는 이동 에이전트 프로그램이다. 그림 9는 Work-flowExecutionRequester 라는 Role을 위임받은 에이전트 서버 SUNIHILL이 TravelingAgent를

```
RBBAC info - This Server, 203.237.51.74 : WorkflowExecutionRequester
reading an AgentFile
sInfo.getServerRole() WorkflowExecutionRequester
[TEST] Searching Privilege is : AgentFileRead

sunihill.kjist.ac.kr : Server Start...
---------- Access Control Information ----------
@ACMode : 3
@Agent Info. -- [ID] : sunihill.kjist.ac.kr2048test [Role] : AgentExecutor
[Owner] : owner01 [OwnerRole] : Requester
@AgentServer Info. -- [Name] : sunihill.kjist.ac.kr [IP] : 203.237.51.74
------------------------------
[TEST] Searching Privilege is : AgentInitiate
sunihill.kjist.ac.kr2048test : waiting message
------------------------------
My name is TravelingAgent
Unbind sunihill.kjist.ac.kr2048test
sunihill.kjist.ac.kr2048test : listenThread is stopped
---------- Access Control Information ----------
@ACMode : 5
@Agent Info. -- [ID] : sunihill.kjist.ac.kr2048test [Role] : AgentExecutor
[Owner] : owner01 [OwnerRole] : Requester
@AgentServer Info. -- [Name] : sunihill.kjist.ac.kr [IP] : 203.237.51.74
------------------------------
[TEST] Searching Privilege is : AgentStop
---------- Access Control Information ----------
@ACMode : 20
@Agent Info. -- [ID] : sunihill.kjist.ac.kr2048test [Role] : AgentExecutor
[Owner] : owner01 [OwnerRole] : Requester
@AgentServer Info. -- [Name] : sunihill.kjist.ac.kr [IP] : 203.237.51.74
------------------------------
[TEST] Searching Privilege is : AgentExport
---------- Access Control Information ----------
@ACMode : 21
@Agent Info. -- [ID] : sunihill.kjist.ac.kr2048test [Role] : AgentExecutor
[Owner] : owner01 [OwnerRole] : Requester
@AgentServer Info. -- [Name] : sunihill.kjist.ac.kr [IP] : 203.237.51.74
------------------------------
[TEST] Searching Privilege is : AgentImport
sunihill.kjist.ac.kr2048 : registerAgent is called
---------- Access Control Information ----------
@ACMode : 3
@Agent Info. -- [ID] : sunihill.kjist.ac.kr2048test [Role] : AgentExecutor
[Owner] : owner01 [OwnerRole] : Requester
@AgentServer Info. -- [Name] : sunihill.kjist.ac.kr [IP] : 203.237.51.74
------------------------------
[TEST] Searching Privilege is : AgentInitiate
sunihill.kjist.ac.kr2048test : waiting message
TravelingAgent : Here is my home, Bye
---------- Access Control Information ----------
@ACMode : 9
@Agent Info. -- [ID] : sunihill.kjist.ac.kr2048test [Role] : AgentExecutor
[Owner] : owner01 [OwnerRole] : Requester
@AgentServer Info. -- [Name] : sunihill.kjist.ac.kr [IP] : 203.237.51.74
------------------------------
[TEST] Searching Privilege is : AgentDestroy
Unbind sunihill.kjist.ac.kr2048test
sunihill.kjist.ac.kr2048test : listenThread is stopped
kill sunihill.kjist.ac.kr2048test
---------- Access Control Information ----------
@ACMode : 5
@Agent Info. -- [ID] : sunihill.kjist.ac.kr2048test [Role] : AgentExecutor
[Owner] : owner01 [OwnerRole] : Requester
@AgentServer Info. -- [Name] : sunihill.kjist.ac.kr [IP] : 203.237.51.74
------------------------------
[TEST] Searching Privilege is : AgentStop
```

[그림 9] 에이전트 서버 SUNIHILL에서의 수행 결과

```
nadia.kjist.ac.kr : Server Start...
---------- Access Control Information ----------
@ACMode : 21
@Agent Info. -- [ID] : sunihill.kjist.ac.kr2048test [Role] : AgentExecutor
[Owner] : owner01 [OwnerRole] : Requester
@AgentServer Info. -- [Name] : nadia.kjist.ac.kr [IP] : 203.237.51.29
------------------------------
[TEST] Searching Privilege is : AgentImport
nadia.kjist.ac.kr2048 : registerAgent is called
---------- Access Control Information ----------
@ACMode : 3
@Agent Info. -- [ID] : sunihill.kjist.ac.kr2048test [Role] : AgentExecutor
[Owner] : owner01 [OwnerRole] : Requester
@AgentServer Info. -- [Name] : nadia.kjist.ac.kr [IP] : 203.237.51.29
------------------------------
[TEST] Searching Privilege is : AgentInitiate
sunihill.kjist.ac.kr2048test : waiting message
------------------------------
Traveling : I just arrived at this machine, my name is TravelingAgent
Unbind sunihill.kjist.ac.kr2048test
sunihill.kjist.ac.kr2048test : listenThread is stopped
---------- Access Control Information ----------
@ACMode : 5
@Agent Info. -- [ID] : sunihill.kjist.ac.kr2048test [Role] : AgentExecutor
[Owner] : owner01 [OwnerRole] : Requester
@AgentServer Info. -- [Name] : nadia.kjist.ac.kr [IP] : 203.237.51.29
------------------------------
[TEST] Searching Privilege is : AgentStop
---------- Access Control Information ----------
@ACMode : 20
@Agent Info. -- [ID] : sunihill.kjist.ac.kr2048test [Role] : AgentExecutor
[Owner] : owner01 [OwnerRole] : Requester
@AgentServer Info. -- [Name] : nadia.kjist.ac.kr [IP] : 203.237.51.29
------------------------------
[TEST] Searching Privilege is : AgentExport
```

[그림 10] 에이전트 서버 NADIA에서의 수행 결과

```
RBBAC info - This Server, 203.237.51.74 : WorkflowTaskPerformer
reading an Agent File sInfo.getServerRole() WorkflowTaskPerformer
[TEST] Searching Privilege is : AgentFileRead
[RBBAC Exception] Cannot Read An Agent File!!
m_agent.ac.RBBACException: Role-Behavior Access Control Exception
```

〔그림 11〕 에이전트 서버 SUNIHILL에서의 수행 결과

생성하고, 생성된 이동 에이전트가 WorkflowTask-Performer의 역할을 위임받는 에이전트 서버 NADIA를 방문한 다음, Home Agent Server인 SUNIHILL로 돌아오는 과정을 보인다. 결과에서 보듯, 일련의 checkFunction들이 올바른 접근 권한을 확인한 후 아무런 예외상황 없이 완료되었음을 알 수 있다.

그림 11은 WorkflowTaskPerfomer 역할을 위임 받은 시점에서 에이전트 서버가 이동 에이전트 파일을 읽어들이지 못하는 경우를 보인다. 접근규칙을 정의함에 있어서, WorkflowTaskPerformer는 AgentFileRead를 포함하는 행위를 소유하지 못하도록 하였으므로 checkFunction은 올바른 권한이 아님을 확인한 후 Exception을 발생시키고 프로세스를 종료한다.

V. 결론 및 향후 과제

본 논문은 현재의 소프트웨어 개발 추세인 다중 사용자, 다층 구조 시스템의 경우에 기존의 접근통제 방법이 적용될 때의 문제점 고찰에서 출발하여, 시스템 자원의 복잡성을 고려하고, 계층적인 보안관리를 유도하는 접근통제 방법인 역할-행위 기반 접근통제 방법을 제안하였다. 또한, 역할-행위 기반 접근통제 방법과 관련, 접근통제 정책으로서의 활용 타당성을 언급하고, 기본적인 형태의 모델을 제시하였으며, 다중 사용자, 계층 구조 시스템의 대표적인 예인 이동 에이전트 기반 워크플로우 시스템에 적용, 구현하고, 간단한 결과를 제시하였다. 논문에서 제시한 역할-행위 기반 접근통제 방법은 보안 관리자에게서 요구되는 지식의 범위를 한정시켜 보안 관리상의 부담을 감소시키고, 계층적으로 설계되는 시스템 내에서 보안관리가 단계적으로 이루어지도록 하는 방법이며, 구현 결과에서 보듯 에이전트 계층의 Behavior 제공을 통해 워크플로우 계층의 보안관리 시 편의와 안전성을 제공할 수 있음을 알 수 있다.

접근통제 규칙이 데이터 베이스에 저장되어 있다고 가정할 때, 기존의 역할 기반 접근통제의 경우, 역할과 권한 사이의 접근통제 규칙 검색을 위해 데이터를 최소 한 번 접근하는 데에 반해, 새 접근통제 방법에서는 최소 두 번 접근해야 한다는 것은 역할-행위 기반 접근통제의 구현 시 오버헤드의 요인이 될 수 있다. 그러나 직관적으로 생각해 볼 때, 역할-행위 기반 접근통제는 두 단계의 추상화를 통해 보다 효율적인 권한 연관을 보장한다. 사용자의 수를 |U|, 기본 권한의 수를 |P|라하고, 역할의 수를 |U|보다 적은 |R|, 행위의 수를 |P|보다 적은 |B|라 하자. 시스템 내의 한 job position에 대하여 보안관리자가 고려해야할 경우의 수는 |U|×|P| 이다. 그러나, |R|<|U|, |B|<|P|가 보장될 경우, 보다 적은 경우의 수 |R|×|B|를 얻게 된다. 즉, 시스템 내 모든 job position 수를 n이라 할 때, 고려해야할 경우의 수는

$$\sum_{j=1}^{n} |R| * |B|$$

이며, 이는 역할 기반 접근통제의

$$\sum_{j=1}^{n} |R| * |P|$$

보다 적은 경우의 수임을 알 수 있다. 또한, 빠른 검색을 지원하는 저장도구를 통해 접근 규칙을 접근하여, 기존 방법과의 검색 효율 격차를 줄일 수 있는 성능의 시스템을 구현하는 것도 기대해볼 수 있다. 역할-행위 기반 접근통제 모델은 현재 가장 기본적인 형태가 제안된 상태이며, 앞으로 상속과 부가조건의 개념을 삽입하여 좀 더 높은 수준의 모델을 구성하고자 한다.

역할-행위 기반 접근통제의 구현 과정에서, 가정한 것은 이동 에이전트 기반 워크플로우 시스템이 믿을 수 있는 호스트들 사이에서 동작한다는 것이었으며, 기타 인증, 암호화 등의 서비스는 별도의 도구에 의하여 지원 받는다고 가정하였다. 신뢰할 수 없는 호스트를 가정할 경우, AEF의 check function호출을 bypassing하는 형태의 공격 또는, 직접적인 메모리 접근을 통한 에이전트 공격을 예상할 수 있다. 그러나, 가정이 허용되는 상황하에서 접근통제 시스템은 별다른 무리 없이 수행되었으며, 이동 에이전트의 수행 시간에도 큰 변화를 주지 않았다.

참 고 문 헌

[1] R. S. Sandhu, E. J. Coyne, H. L. Fein-Stein, and C. E. Youngman, "Role-Based Access Control Models", *IEEE Computer*, Vol. 29, No. 2, pp. 38~37, February 1996.

[2] DoD, "Department of Defense Trusted Computer System Evaluation Criteria", TCSEC DOD 5200.28-STD, August 1983.

[3] D. F. Ferraiolo, J. F. Barkley, and D. R. Kuhn, "A Role Based Access Control Model and Reference Implementation within a Corporate Intranet", *ACM Transactions on Information Systems Security*, Vol. 1, No. 2, February 1999.

[4] M. Milenkovic, *Operating Systems - Concepts and Design*, McGraw Hill, 2nd Ed., 1995.

[5] J. F. Barkely, V. Cincotta, D. F. Ferraiolo, S. Gavrilla, D. R. Kuhn, "Role Based Access Control for the World Wide Web", NIST, 20th National Computer Security Conference, 1997.

[6] 이철원, 이병각, 김기현, 박정호, 이홍섭, 최용락, "역할 속성을 이용한 역할기반 접근통제 메커니즘", *통신정보보호학회 논문지*, 제8권, 제4호, 1998.

[7] J. F. Barkely, "Application Engineering in Health Care", Second Annual CHIN Summit, 1995. http://hissa.ncsl.nist.gov/rbac/

[8] 조이남, 차병주, "우리나라의 홈뱅킹 현황과 발전방향", *정보화 저널*, 제1권, 제3호, 1994.

[9] WFMC, "Workflow Management Coalition, Technology and Glossary", WFMC-TC-1011, Jun., 1996.

[10] G.. Alonso, H. Scheck, "Database Technology in Workflow Environment, Informatik", Jan., 1996.

[11] T. Cai, P. A. Gloor, S. Nog, "Dartflow: A Workflow Management System on the Web using Transportable Agents", Technical Report PCS-TR96-283, May., 1996.

[12] 신욱, 이동익, "이동 에이전트 기반 워크플로우 시스템의 보안 고려사항", *KISS Proceedings of The 11th Spring Conference*, 제6권, 제1호, pp. 1137~1140, 1999.

[13] G. Karjoth, D. B. Lange, M. Oshima, "A Security Model For Aglets", *IEEE Internet Computing*, pp. 68~77, July August 1997.

[14] ISO/IEC DIS 10181-3 Information Technology Open Systems Interconnection Security Frameworks in Open Systems Part 3 : Access Control, 1993.

[15] Jeong-Joon Yoo and Dong-Ik Lee, "X-MAS: Mobile Agent Platform for Workflow Systems with Time Constraints", 4th. International Symposium on Autonomous Decentralized Systems, pp. 370~373, 1999.

[16] 신욱, 이동익, "역할-행위 기반 접근통제", 1999년 한국 정보처리학회 추계 학술발표 논문집, 제6권, 제2호, SEC 57-63, 1999.

〈著者紹介〉

신 욱 (Wook Shin)
1998년 2월 : 동국대학교 컴퓨터공학과 졸업
2000년 2월 : 광주과학기술원 정보통신공학과 석사
2000년 2월~현재 : 광주과학기술원 정보통신공학과 박사과정
〈관심분야〉 정보보호, 분산 시스템, 병행 시스템 검증 이론 등

이 동 익 (Dong-Ik Lee) 정회원
1985년 : 영남대학교 전기공학과 졸업
1989년 : Osaka Univ. 전자공학과 석사
1993년 : Osaka Univ. 전자공학과 박사
1990~95년 : Osaka Univ. 전자공학과 Research Associate
1993~94년 : Univ. of Illinois at Urbana-Champaign, Visiting Assistant Professor
1995년~현재 : 광주과학기술원 정보통신공학과 부교수
〈관심분야〉 정보보호, 분산/병행 시스템, 비동기 회로 설계/CAD, PetriNet 이론 등

윤 석 환 (Seok-Hwan Yoon)
1982년 : 아주대학교 산업공학과 졸업
1984년 : 건국대학교 산업공학과 석사
1992년 : 품질관리 기술사 자격취득(등록번호:92137010342V)
1996년 : 아주대학교 산업공학과 박사
1986~97년 : 한국전자통신연구원 책임연구원
1998년~현재 : 정보통신연구진흥원 책임연구원
1998년~현재 : 한국정보처리학회 이사 겸 학회지 편집위원장
2000년~현재 : 대전산업대학교 정보통신, 컴퓨터공학과 겸임교수
〈관심분야〉 소프트웨어공학, 품질공학, 개발방법론, 그룹웨어 등