2001년 한국멀티미디어학회 춘계학술발표논문집

Java 기반 이동 에이전트 시스템의 실행 환경 보호 방안

한명진†, 신정화†, 신 원†, 이경현‡
† 부경대학교 전자계산학과, ‡ 부경대학교 전자컴퓨터정보통신공학부

A Protection of Execution Environment in Java-based Mobile Agent System

Myung-Jin Han†, Jung-Hwa Shin†, Weon Shin† and Kyung-Hyune Rhee‡
† Dept. of Computer Science, PKNU
‡ Division of Electronic, Computer and Telecommunication Engineering, PKNU

요 약

사용자의 권한을 위임받아 다양한 네트워크 환경에서 원하는 작업을 수행하는 이동 에이전트 시스템이 현재 Java 언어를 기반으로 구축되고 있다. 본 논문에서는 악의적인 에이전트 동작으로부터 시스템을 보호하고 그 동작을 제한·추적하기 위한 이동 에이전트 실행 환경 보호 모델을 제안하고 구현 환경에 대하여 논의한다.

1. 서론

최근 들어 네트웍 기술의 급속한 발전으로 데이터 이동만으로 서비스를 제공하던 기존의 통신 응용 프로그램들은 서버의 과부하를 줄이면서 필요한 서비스를 사용자에게 지원하기 위한 방법의 하나로 데이터와 코드를 결합한 객체의 이동 즉, 이동 에이전트(mobile agent)를 통하여 서비스를 지원하고 있다. 이동 에이전트는 코드 자체가 여러 호스트 사이를 이동하도록 구현되므로, 이동하는 코드 자체와 이동 코드를 받아들여 수행하는 호스트 측 모두에게 새로운 보안 문제를 야기한다. 따라서, 본 논문에서는 Java 기반 이동 에이전트를 안전하게 동작시킬 수 있는 실행 환경 보호 모델을 제안하고자 한다. 2장에서 이동 에이전트 시스템을 기술하고 3장에서 이동 에이전트 시스템과 역할 기반 접근 제어에 관해서 살펴본다. 4장에서는 Java를 기반으로 하는 이동 에이전트에 적용 가능한 Java Security 구조를 분석하고, 5장에서는 Java 기반 이동 에이전트 시스템의 실행 환경 보호를 살펴보고, 6장에서 결론을 맺는다.

2. 이동 에이전트 시스템

2.1 이동 에이전트 소개

이동 컴퓨팅 환경에서 사용자의 요구에 따라 여행 정보나 주식 정보 등을 인터넷에서 검색하여 보여주거나 사용자를 대신하여 업무를 처리해주는 자율적인 프로세서를 일반적으로 "에이전트"라고 한다. 여러 유형의 소프트웨어 에이전트들이 존재할 수 있으나 자신이 생성된 시스템에 머물러 있지 않고 주어진 작업을 수행하기 위해 네트워크로 연결된 시스템 사이를 스스로의 판단 하에서 이동하는 에이전트를 특히 "이동 에이전트"라고 한다[1]. 이것은 사용자를 위해 자동적으로 행동하는 프로세스이다. 이러한 이동 에이전트는 기존 통신 체계 하에서 발생하는 막

대한 통신 비용의 절감, 인터넷을 통해 자유롭게 제품을 사고 파는 전자 상거래의 대두, 이동 컴퓨팅 및 분산 컴퓨팅 환경의 급속한 성장 등의 요구를 수용하는 해결책으로서 등장하게 되었다.

2.2 이동 에이전트 시스템 시큐리티

이동 에이전트는 이동성, 자율성, 지능성, 반응성, 사회성, 협동성 등의 다양한 특성을 함께 가지고 있다[2]. 그 중 이동 에이전트의 가장 두드러진 특징인 이동성은 이동 에이전트 작성을 위해 사용하는 이동 코드에 기반을 두고 있으므로 이동 에이전트의 보안 문제는 여러 시스템에서 실행 가능한 이동 코드 자체에 대한 보안 문제에 기반 한다고 볼 수 있다[1].

호스트 입장에서 바라본다면 이동 에이전트는 외부에서 유입되는 신뢰할 수 없는 코드이므로 악의적인 목적을 가지고 호스트의 중요 정보를 임의로 삭제, 수정할 수 있을 뿐만 아니라 시스템 자원을 독점하여 다른 정당한 사용자 및 에이전트에게 피해를 끼칠 수도 있다. 따라서, 호스트는 각 에이전트가 수행되는 동안 자원 접근을 감시하고, 부여받은 권한에 맞도록 동작하는지, 시스템 자원에 불법적으로 액세스하지 않는지를 검사하고 추적할 수 있는 방안이 제공되어야 한다.

Java 환경에서 외부에서의 악의적인 코드 유입으로 인해 발생할 수 있는 공격은 개략적으로 다음과 같이 4가지로 분류할 수 있다[3].

- 시스템 수정 : Java 코드가 읽기 및 쓰기 권한을 얻어서 시스템을 변경할 수 있다.
- 프라이버시 침해 : Java 코드가 읽기 권한을 얻어 시스템 정보를 훔치는 것이 가능하다.
- 서비스 거부 : 시스템 자원을 점유하여 정당한 수행을 방해할 수 있다.
- 위장 : 시스템의 실제 정당한 사용자로 위장할 수 있다.

일반적인 Java 기반 에이전트 시스템은 코드 자체에 대한 안전성을 위해 Class Loader, Bytecode Verifier를 적용하여 사용하고, 시큐리티 정책에 따른 허가를 수행하기 위해 Security Manager를 확장하거나 수정하여 사용한다. 또한, 실행 환경에서 코드에 대한 액세스 제어를 위해 JVM의 Sandbox 모델, Signed Applet 모델, Fine-grained Access Control 모델을 적용하고 있다[3]. 이 Java 2의 Fine-grained Access Control 모델은 설정된 보안 정책을 기초로 Applet 및 Application이 로드될 때 각 애플릿에 허가권(Permission)을 할당하고 할당된 허가권(read, write)에 따라 사용자 자원에 접근이 가능하며 시스템 영역 내에 "Protected Domain"을 설정하여 애플릿의 특성에 따라 다중 보안 정책 적용이 가능한 모델이다.

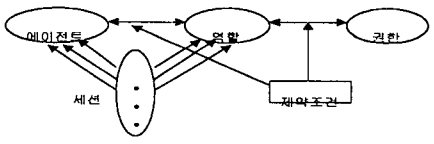
3. 이동 에이전트 시스템과 역할기반 접근 제어

3.1 RBAC 정의

RBAC(Role Based Access Control)은 1970년대 다중 사용자, 다중 프로그래밍 환경에서 제안된 방식으로 각 주체가 역할이라는 추상화 단계를 거쳐 자원 접근 권한과 연관되는 방식이다. 즉 접근 제어는 자원에 허가받지 않은 접근을 차단하여 시스템의 안전성을 보장해 주는 것이다. 허가받지 않은 접근으로부터 자원을 보호하기 위한 접근 제어 메커니즘으로 임의적 접근 제어(Discretionary Access Control)와 강제적 접근제어(Mandatory Access Control)가 있다[4]. RBAC은 이러한 MAC와 DAC를 대체할 수 있는 효과적인 방법으로 중심적인 개념은 사용자가 기업이나 조직의 정보 자원을 임의로 접근할 수 없도록 하는 것이다. 대신에 접근 권한이 역할에 부여되고 사용자는 적절한 역할에 소속됨으로서 역할의 수행에 필요한 최소 자원만을 접근할 수 있도록 한다. 접근을 제어하기 위한 역할의 사용은 조직의 특정한 보안 정책들을 개발하고 강화하기 위한, 그리고 보안 관리 과정을 능률적으로 처리하기 위한 효율적인 수단이 될 수 있다.

3.2 이동 에이전트 시스템에서의 RBAC

이동 에이전트를 RBAC에 적용할 경우 기능적 측면에서 RBAC의 중심 개념은 역할, 역할의 대상인 에이전트, 그리고 연관된 행동을 나타내는 권한으로 구성된다. 즉, RBAC을 사용한 접근 제어에서, 접근 결정은 개인 사용자들이 조직의 부분으로서 가지는 역할에 기반을 두고 있다. 역할, 에이전트, 권한 사이의 관계는 다음 <그림 1>과 같다[5].



<그림 1> 역할 기반 접근 제어 모델

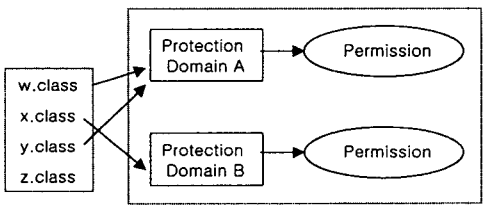
여기서 두 개의 화살표는 다-대-다(many-to-many) 관계를 나타낸다. 이것은 하나의 에이전트가 하나

혹은 그 이상의 역할과 연관될 수 있고 하나의 역할이 하나 혹은 그 이상의 에이전트를 가질 수 있다는 것을 의미한다. 역할과 권한의 경우도 동일하다. 필요한 권한을 직접 에이전트에게 부여하지 않고 수행에 필요한 역할에 부여하였다. 이를 통하여 에이전트를 쉽게 역할에 추가하거나 제거할 수 있고, 역할도 단지 권한에만 관련이 있으므로 에이전트에게 할당된 동작의 기능이 바뀌거나 삭제될 때 권한만을 수정하면 된다. 에이전트는 세션을 통하여 할당된 역할을 수행할 수 있고 하나의 에이전트는 여러 개의 세션을 동시에 수행할 수 있다. 세션에 의해 수행되는 역할을 활성화 역할(active role)이라고 한다. 그리고 제약 조건은 모든 구성요소에 대하여 적용될 수 있다. 역할들은 서로 상호 중첩되는 책임과 특권을 가질 수 있다. 즉, 다른 역할들과 연관된 에이전트가 일반적인 연산들을 수행할 필요가 있을 수도 있다.

4. 적용 가능한 Java Security 구조

4.1 Protection Domain[3][6]

현재 Java 2에서 각 권한은 Protection Domain에 의해 허가되고, 각 Java 클래스와 객체들은 Protection Domain에 소속된다. 이러한 Protection Domain은 사용자 인증에 의해 구분되어 수행할 수 있고, 다른 사용자에 의해 수행되는 같은 코드도 서로 다른 권한을 부여할 수 있게 된다. Protection Domain은 필요시 생성되고, 소스 코드에 기반하는데, 여기서 코드 소스는 코드 서명에 사용되었던 코드 근원지와 공개키 집합에 기반한다. 다음 <그림 2>는 Java 클래스와 Protection Domain, Permission의 관계를 보여준다[7].



<그림 2> Protection Domain

즉, Java 코드의 근원지 URL 및 서명자의 신분에 의해 Protection Domain이 결정되고 이를 기반으로 Permission에 따른 액세스 제어를 수행한다. 여기서 각 클래스 및 객체는 오직 하나의 도메인에 소속되어 유지되며, 각 도메인은 자체 도메인 범위 내에 부가적인 내부 자원 보호를 구현하고 있다. 그러나 Protection Domain은 코드의 근원지에 대해서 도메인으로 분류하여 수동적인 보호 기능만을 제공하므로 불법적인 코드의 동작에 대해서는 보호할 수 없는 단점이 있다. 이를 위하여 Java 2에서는 Security Manager를 구현하여 코드 동작에 있어 허가 및 추적을 제공하고 있다.

4.2 Java 2 Security Manager

Java 2의 Security Manager는 각 실행 코드마다 Protection Domain을 적용하여 Domain-based Access Control을 수행하여 시스템 수정, 프라이버시 침해를 막을 수 있고, Security Manager 자체를 확장·수정함으로써 서비스 거부, 위장과 같은 공격도 막는 것이 가능하다[3].

기본적으로 Java Application은 제한된 Sandbox에서 수행되는 Java Applet과는 달리 Security Manager를 가지지 않는다. 그러나 Java 기반 이동 에이전트 시스템에서 실행 환경을 보호하기 위해서는 에이전트 코드 실행동안 자체 Security Manager를 가지도록 구현해야 하고 이를 통하여 코드 실행 동안의 시큐리티 허가를 수행할 수 있게 된다. 구체적인 방안으로 Security Manager를 통한 권한 제어는 Java 2 SecurityManager 클래스를 상속받는 별도의 클래스를 생성하여 사용하는 방법과 개발자가 Java 2 SecurityManager 클래스의 소스 코드를 목적에 맞도록 직접 수정하여 사용하는 방법이 있다[7]. 전자의 경우는 손쉽게 시스템에 적용할 수 있는 장점이 있지만, 시스템 목적에 맞는 유연한 처리가 어렵다는 단점이 있다. 반면에 후자의 경우는 구현 시스템에 따라 그 목적을 따르는 시큐리티를 구현할 수 있지만, 소스 코드를 수정해야 하는 방대한 작업과 다른 시스템과는 호환성이 없다는 단점이 존재한다.

5. Java 기반 이동 에이전트 시스템의 실행 환경 보호

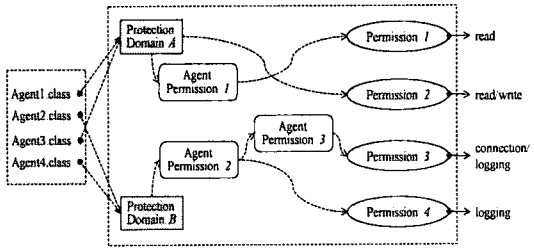
현재 Java 2 권한 허가 구조는 근원지 기반의 코드 소스를 허가하고 시스템 자원에 대한 어떤 행위에 대한 권한을 부여하도록 구현되어 있다. 이 구조는 부가적인 시스템 변경 없이 시큐리티 정책 수정만을 요구하므로 다양한 어플리케이션에 유연하게 적용할 수 있다. 그러나, 시스템 구현 환경 또는 목적에 따라 권한 허가 구조를 수정하여 확장하거나 그 기능을 제한할 필요가 있다. Java 2 권한 허가

구조에서는 SecurityManager 클래스를 상속받아 서브클래스를 설계하고 필수적인 메소드를 재정의(Override)하는 "SecurityManager Extension" 방법을 제공하는데, 다음과 같은 2가지 과정을 거친다[7].

① 새로운 SecurityManager 생성 :
기존의 SecurityManager를 상속받아 에이전트 실행을 위해 새로 설계된 SecurityManager를 생성한다.

② 필요한 메소드 재정의 :
시큐리티 특성에 따라 시스템 구현시 해당하는 check*XXX*() 메소드들을 재정의한다.

이렇게 확장된 SecurityManager는 Protection Domain과 함께 근원지 및 서명자 인증을 제공하고, 각 Permission에서 기존 기능을 시스템 구현시 요구되는 형태로 수정·확장하여 허가 권한 설정 및 실행 로깅 기능을 제공하여 코드 실행으로부터 실행 환경을 보호할 수 있는 장점을 가진다. 다음 <그림 3>은 이를 적용하여 각 에이전트 실행에 대한 "Agent Permission" 개념을 새롭게 도입하여 기존의 Java 2 권한 허가 구조를 확장한 것을 보여준다.



<그림 3> Java 2 구조의 확장

여기서, 기존 Java 2 구조에서 에이전트 환경을 고려하여 추가된 Agent Permission은 다음과 같은 특징을 가진다[7].

- Agent Permission은 실제 허가에 따른 실행을 위한 Permission 수행 이전의 Pre-Permission이다.
- Agent Permission은 이름에 의해 유일하게 구별 가능하다.
- Agent Permission은 한 Permission에서 다른 Permission으로 전이가 가능하지만, 사이클은 허용되지 않는다.
- Agent Permission에 RBAC을 적용하면 에이전트 동작에 역할 행위 기반의 액세스 제어 수행이 가능하다.

6. 결론

본 논문에서는 이동 에이전트 시스템에 대한 소개 및 Java 기반 에이전트에 적용 가능한 Java Security 구조에 대해서 살펴보았다. 이동 에이전트는 주어진 작업을 수행하기 위해 네트워크로 연결된 여러 시스템 사이를 스스로의 판단 하에 이동하는 지능적인 프로그램으로 본 논문에서는 Java Security 구조에서 각 실행코드마다 Protection Domain을 적용하여 Domain-based Access Control을 수행하는 Security Manager의 일부를 수정하거나 새로운 Security Manager를 생성하여 시스템 수정, 프라이버시 침해 등을 막을 수 있는 모델을 제안하였다. 제안된 모델은 악의적인 에이전트 동작으로부터 시스템을 보호하고 그 동작을 제한·추적하기 위한 이동 에이전트 실행 환경에 적용될 수 있다.

[참고문헌]

[1] 석황희, 김인철, "이동에이전트의 개념과 응용", 「한국멀티미디어학회지」,제3권,제2호, pp.29-39, 1999. 10.

[2] 이상열, 정성호, 황병곤, "KQML(Knowledge Query and Manipulation Language)을 이용한 에이전트의 사용화", 「한국멀티미디어학회지」 제3권 제2호, pp.53-61, 1999. 10.

[3] Pistoia, Reller, Gupta, Nagnur, and Ramani, "Java 2 Network Security Second Edition"

[4] 이철원, 이병각, 김기현, 박정호, 이홍섭, 최용락, "역할 속성을 이용한 역할기반 접근통제 매커니즘", 「통신정보보호학회논문지」, 제8권, 제4호, 1998. 12.

[5] 이희규, 조한진, 김봉한, 이재광, "WWW에서 안전한 역할 기반 접근 제어 시스템 구현", 「통신정보보호학회논문지」, 제10권, 제1호, 2000. 3.

[6] 이완석, 김홍근, "자바 보안 모델", 「정보과학회지」, 제16권, 제4호, pp.29-35, 1998. 4.

[7] Luigi Giuri, "Role-Based Access Control in Java", ACM Workshop on Role Based Access Control, pp.91-100