通信情報保護學會誌
第 10 券 第 4 號, 2000. 12

RBAC 모델의 Java 2 환경 적용연구 동향 및 활용연구

이 형 효*, 이 동 익**, 노 봉 남***

요 약

자바기술은 플랫폼에 독립적인 프로그래밍 언어와 실행환경 특성으로 인터넷의 확산과 함께 활용분야를 넓혀가고 있다. 초기의 매우 엄격한 자바 보안모델은 네트워크 환경에서 유용한 기능을 수행하는 프로그램 개발에 장애가 되었으나, 현재는 보안정책에 기반한 유연한 접근통제기능을 제공하도록 수정, 보완되었다. 그러나 현재의 자바 보안모델이 코드 기반의 접근통제기능만을 제공하고 있어 네트워크 환경에서 여러 사용자에 의해 수행되는 응용 프로그램 실행환경에 적합하지 않은 문제점이 있다. 본 논문에서는 이와 같은 문제점을 해결하기 위해 자바 환경에 역할기반 접근통제모델을 적용하는 연구들에 대해 살펴보고 역할기반 접근통제모델이 적용된 자바 기술이 활용될 수 있는 분야에 대해 기술한다

Ⅰ. 서 론

1995년 개발된 자바 기술은 운영체제와 하드웨어에 무관하게 프로그램을 동작시키는 특징으로 다양한 플랫폼이 접속되어 있는 인터넷의 확장과 함께 사용분야가 꾸준히 증가되고 있다.[8] 또한 자바 기술은 플랫폼에 독립적인 코드 이동성(code mobility)을 가진 프로그래밍 언어의 제공뿐만 아니라 안전한 프로그램의 개발 및 실행환경을 제공함으로써 기업의 응용 프로그램을 비롯하여 컴포넌트 소프트웨어, 내장형 시스템(embedded system), 전자상거래에 사용되는 스마트카드 등 다양한 분야의 소프트웨어 개발에 활용되고 있다.

자바 기술은 초기에 자바가상기계(JVM: Java Virtual Machine)[9]가 내장된 웹 브라우저에서 동작하는 애플릿 개발에 주로 활용되었으며 신뢰되지 않은 애플릿으로부터 사용자 시스템을 보호하기 위해 제한적 기능을 애플릿에게 허용하는 매우 엄격한 보안기법이 사용되었다. 그러나, 보안을 위해 애플릿의 기능을 제한하는 대신 사용자 시스템에 저장된 정보에 접근하여 보다 유용한 기능을 수행하고자 하는 요구사항이 증대됨에 따라 자바 기술에서 제공하는 보안기술에 대한 수정이 계속되었다. 따라서, Java 1.1, Java 1.2(Java 2) 버전에서는 Java 1.0 버전에 비해 융통성있는 보안기능을 수행할 수 있도록 보안기능이 수정, 보완되었다. 이와 같이 자바기술의 플랫폼 독립성, 코드 이동성과 융통성있는 보안기법을 활용하면 단위기업내의 정보시스템 개발 외에도 인터넷을 통한 기업간 정보시스템의 구축에 효과적으로 활용될 수 있다.

본 논문에서는 먼저 자바의 각 버전에 적용된 보안기법의 특징에 대해 살펴보고, 최근 개발이 진행중인 자바 환경에서의 사용자 인증 및 권한부여 서비스인 JAAS(Java Authentication and Authorization Service)[6]에 대해 기술한다. 그리고, 인터넷과 같은 네트워크 환경에서 많은 사용자에 대한 효과적인 권한관리를 위해 역할기반 접근통제 모델(RBAC: Role-Based Access Control Model)을 자바 환경에 적용하는 관련 연구들에 대해 알아본다. 마지막으로 역할기반 접근통제 모델을 적용한

* 광주과학기술원 BK21 정보기술사업단(hlee@kjist.ac.kr)
** 광주과학기술원 정보통신공학과(dilee@kjist.ac.kr),
*** 전남대학교 컴퓨터정보학부(bongnam@chonnam.ac.kr)
※ 본 연구는 교육부 두뇌한국21 정보기술사업단의 지원금에 의해 수행되었습니다.

자바 기술의 활용방안을 제시한다.

Ⅱ. Java 보안특성

자바 기술은 네트워크를 통해 다운로드된 프로그램이 자바가상기계가 포함된 웹 브라우저와 같은 환경에서, 대부분의 경우 사용자가 인식하지 못한 상태 또는 사용자의 허가없이, 수행되는 특성을 제공하는데 있다. 이와 같이 코드가 이동하여 수행되는 새로운 컴퓨팅 패러다임이 활용되기 위해서 해결되어야 할 가장 중요한 문제는 다운로드된 코드로부터 사용자 시스템이 안전하게 보호되어야 하는 점이며, 따라서 보안에 대한 사항들이 자바 프로그래밍 언어와 자바가상기계 등의 설계단계부터 고려되었다.[8,9] 그러나, 표준 자바가상기계나 웹 브라우저에 내장된 자바가상기계 구현의 오류로 인해 다운로드된 애플릿 코드에 의한 사용자 시스템의 보안침해 사례들이 발생하였으며[10,11], 보안오류가 포함된 자바가상기계의 수정이나 새로운 보안기능을 제공하는 자바 버전의 발표로 문제점을 해결되고 있다[16,17].

이 장에서는 자바의 각 버전에 적용된 보안기술의 주요 특성을 살펴본다.

1. Java 1.0

Java 1.0 버전에서는 자바 프로그래밍 언어 수준에서 제공되는 보안기능 외에 바이트코드 검증기(verifier), 클래스 로더(class loader), 보안관리자(security manager)로 구성된 sandbox 보안모델을 제시하였다.

바이트코드 검증기는 클래스 파일의 구조를 조사하는 매우 기본적인 검사와 함께 theorem prover를 이용한 스택의 오버플로우/언더플로우 발생여부 점검, 불법적인 자료형 변환(cast), 바이트코드 명령별 매개변수 자료형 점검, 클래스와 멤버함수, 멤버변수에 대한 접근허가 점검 등을 통해 바이트코드가 실행되는 과정에서 발생가능한 오류들을 검사한다. 바이트코드 검증을 통과한 바이트코드는 실행과정에서 오류에 대한 조사를 하지 않으므로 바이트코드의 빠른 수행이 가능한 장점을 가진 반면, 모든 바이트코드 오류를 점검해내는 완벽한 바이트코드 검증기의 개발과 시험이 거의 불가능하다는 문제점이 있다.

클래스 로더는 코드의 이동성과 실행시간에 필요한 코드를 동적으로 연결하는 자바의 특성을 지원하는 프로그램으로서 자바가상기계에 의해 요청된 자바 클래스의 적재기능과 이름공간(namespace) 관리기능을 수행한다. 클래스 로더에는 자바 환경의 부트스트래핑에 이용되며 오버라이드가 금지된 primordial 클래스 로더와 자바 프로그램(어플리케이션, 애플릿)의 수행단계에서 요구된 자바 클래스의 적재에 사용되는 일반적인 클래스 로더가 있다. 대표적인 클래스 로더로는 웹 브라우저에 내장되어 애플릿 수행에 필요한 클래스를 적재하는 애플릿 클래스 로더로서 애플릿에 의해 요청된 클래스를 네트워크에서 다운로드하는 대신 primordial 클래스 로더를 통해 먼저 적재함으로써 클래스 스푸핑(spoofing) 공격으로부터 시스템을 보호하는 기능을 제공한다. 그리고, 하나의 자바가상기계에는 여러 개의 클래스 로더가 동작할 수 있으며 각 클래스 로더는 서로 독립적인 이름공간을 유지한다.

보안 관리자는 애플릿이 호출할 수 있는 자바 API(Application Programming Interface)를 제한하는 기능을 수행한다. 즉, 보안 관리자는 자바 프로그램의 신뢰도를 기반으로 시스템 보안과 연관된 연산을 수행할 수 있는지를 판단하는 기능을 제공한다. Java 1.0의 경우 신뢰된 프로그램인 자바 어플리케이션은 보안 관리자의 통제를 받지 않지만, 비신뢰된 자바 프로그램인 애플릿은 보안 관리자에 의해 특정 자바 API 실행가능 여부가 결정된다.

이와 같은 보안속성을 가진 sandbox 보안모델은 자바가상기계가 설치된 시스템으로부터 적재된 어플리케이션을 신뢰된 자바 프로그램으로 간주하여 모든 자바 API를 수행할 수 있는 권한을 부여하는 반면 네트워크로부터 다운로드된 애플릿은 비신뢰된 자바 프로그램으로 처리하여 매우 제한된 자바 API 수행권한만을 부여하는 간단하면서도 매우 엄격한 보안기능을 제공한다. 즉, 모든 애플릿과 CLASSPATH 환경변수에 포함되지 않은 어플리케이션은 바이트코드 검증기, 애플릿 클래스 로더, 보안 관리자를 통한 엄격한 보안점검을 거치는 데 반해 CLASSPATH 환경변수에 저장된 어플리케이션은 보안점검과정을 거치지 않게 된다. 따라서, sandbox 보안모델은 시스템의 보안성 향상을 위해 애플릿의 권한을 축소함으로써 코드 이동성을 통해 융통성있는 시스템 구

축과 운영을 가능하게 하는 자바 기술을 장점을 제한하는 문제점이 있다.

2. Java 1.1

Java 1.1의 보안모델은 네트워크로부터 다운로드된 일부 애플릿에 대해 어플리케이션과 동일한 권한을 부여하는 점을 제외하면 Java 1.0의 보안모델

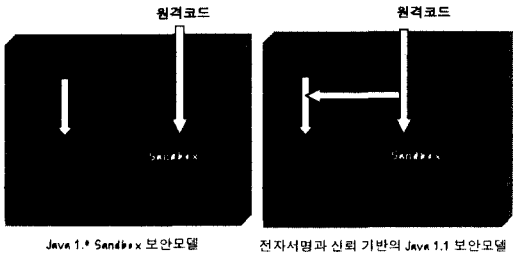


그림 1. Java 1.0, 1.1 보안모델의 동작특성

과 동일하다. 어플리케이션과 동일한 권한이 부여되는 애플릿은 사용자가 신뢰하는 대상에 의해 전자서명된 자바 클래스이다. 이 방식은 이동 코드의 신뢰여부에 따라 코드의 권한을 결정하는 마이크로소프트사의 ActiveX와 같은 보안정책을 지원한다[11]. 전자서명 기능의 제공을 위해 Java 1.1에는 암호암수, 해시함수, X.509 인증서 처리관련 API들이 추가되었다[11,12]. 그러나, Java 1.1의 보안모델 역시 Java 1.0과 같이 모든 자바 프로그램을 신뢰된 자바 프로그램과 비신뢰된 자바 프로그램 두 종류로만 구분하는 특성을 가진다. 그림 1은 Java 1.0, 1.1 보안모델의 동작특성을 나타내고 있다[13].

3. Java 1.2(Java 2)

Java 1.1에 채택된 전자서명된 애플릿 방식은 애플릿에 대한 신뢰여부를 사용자가 결정하여 sandbox 보안모델의 엄격한 제약사항을 보완하고 있으나, 신뢰된 자바 프로그램에게 모든 자바 API를 수행하는 권한을 부여하는 정책을 지원하였다. 따라서, 응용분야의 다양한 보안정책을 지원하기 위해 여러 단계의 신뢰도를 지정하고 각 신뢰도에 따라 권한을 부여할 수 있는 융통성있는 보안모델이 요구되었다.

Java 2의 보안모델은 Java 1.0, 1.1의 두 가지의 신뢰단계대신 사용자가 정의한 보안정책에 따라 여러 단계의 신뢰도를 제공하는 세분화된 보안기능을 제공하며, Java 2 보안모델의 특징은 다음과 같다[15].

· 세부화된 접근통제 기능

자바 프로그램별로 실행가능한 권한을 부여하여 sandbox 보안모델이 지원하는 두 단계 신뢰도에 의한 단순한 접근통제 기능을 보완하였다.

· 사용자에 의해 구성가능한 보안정책

자바 프로그램의 개발자나 사용자가 다양한 보안요구사항을 지원하기 위한 다양한 보안정책을 구성할 수 있다.

· 확장가능한 접근통제 구조

보안정책이나 권한의 논리적 특성에 따라 접근권한을 그룹화하고 재사용함으로써 접근통제 구조의 확장이 용이하다.

· 모든 자바 프로그램에 대한 보안점검 기능

Java 1.0, 1.1에서 정의된 신뢰된 자바 프로그램(CLASSPATH에 지정된 어플리케이션, 전자서명된 애플릿)과 비신뢰된 자바 프로그램(CLASSPATH에 지정되지 않은 어플리케이션, 일반 애플릿)의 구분이 없어지고 모든 자바 프로그램에 대한 보안점검이 이루어진다.

Java 2 보안모델의 가장 큰 특징은 자바 프로그램의 권한이 시스템 관리자와 사용자가 설정한 보안정책에 따라 결정되는 점이다. 다음은 Java 2 보안모델의 주요개념인 코드 소스(code source), 권한(permission), 보호 도메인(protection domain), 그리고 보안정책화일(security policy file)에 대해 기술한다.

3.1 코드 소스, 권한, 보호 도메인

코드 소스는 자바 프로그램의 출처를 나타내는 URL 형식의 코드 베이스(code base)와 자바 프로그램을 전자서명한 서명자로 정의된다. 권한은 파일 이름이나 소켓 번호 등을 나타내는 대상(target)과 대상에 대해 실행할 수 있는 연산(read, write, open, listen 등)으로 구성되며, 경우에 따라 권한 클래스에 대한 전자서명자가 추가될 수 있다. 코드 베이스를 표현하는 하나의 URL이 디렉토리와 호스

트를 정의한 경우, 디렉토리나 호스트에 포함된 모든 자바 프로그램에 대해 권한을 허용하게 된다.

보호 도메인은 Principal에 의해 접근가능한 객체(object)의 집합으로 정의된다. 여기서 Principal는 권한이 부여되는 컴퓨터 시스템내의 객체이며, 보호 도메인에 포함된 객체에 대해 자신에게 부여된 권한을 수행할 수 있다. Java 2에서 보호 도메인은 코드 소스와 코드 소스에 허용된, 즉 코드 소스에

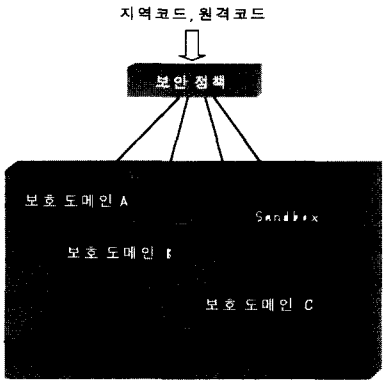


그림 2. Java 2 보안모델의 구조

포함된 자바 프로그램에 허용된, 권한의 집합으로 정의된다. 보호 도메인은 보안정책화일에 정의되며, 보안정책화일은 전용 에디터(policytool)이나 일반 에디터를 이용하여 작성이 가능하다.

비록 시스템내에는 여러 개의 보호 도메인이 존재하지만, 주어진 시점에서 자바 프로그램은 하나의 보호 도메인에 속하게 된다. 네트워크나 로컬 디스크로부터 로드된 자바 프로그램은 시스템에 정의된 어느 보호 도메인에 포함되는지 결정된 후 해당 보호 도메인에 부여된 권한을 수행할 수 있다. 하나의 보호 도메인에는 하나 이상의 자바 클래스가 포함될 수 있고, 보호 도메인은 여러 개의 권한을 가질 수 있다. 그림 2는 Java 2 보안모델의 특성을 나타내고 있다[13].

3.2 보안정책화일

보안정책화일은 보호 도메인을 정의하는 기능을 제공하며, 시스템 관리자에 의해 지정된 시스템 보안정책화일과 사용자들이 각각 지정하는 하나 이상의 사용자 보안정책화일로 구성된다.

보안정책화일을 기술하는 grant 구조체의 문법과 사용 예는 그림 3과 같다.

```
grant [ signedBy signers ] [, codeBase URL ] {
   permission permission_class [ target ] [, action ]
                [, signedBy signers ];
   [ permission ... ]
};
```

(a) grant 구조체 문법

```
grant codeBase "file:${java.home}/lib/ext/-" {
   permission java.security.AllPermission;
};
grant codeBase "file:/C:/java1.2/security" {
   permission java.io.FilePermission
              "C:/datastore/personnel.txt", "read";
};
grant codeBase "http://foo.com/trusted_code" {
   permission java.io.FilePermission
              "C:/datastore/personnel.txt", "read";
   permission java.io.FilePermission
              "C:/datastore/personnel.txt", "write";
};
```

(b) 보안정책화일 예제

그림 3. 보안정책화일 문법 및 사용 예

Java 2에서는 모든 애플릿 코드에 대해 접근권한을 점검하는 보안관리자가 자동적으로 수행되지만 어플리케이션의 수행때에는 보안관리자가 동작하지 않는다. 따라서, 어플리케이션에 대해 애플릿과 같은 보안관리자를 적용하기 위해서는 자바 인터프리터를 이용하여 어플리케이션을 수행할 때 "-Djava.security.manager" 옵션을 추가하거나 어플리케이션 내부에서 "setSecurityManger()" 메소드를 이용하여 보안관리자를 구현하여야 한다.

Java 2 보안모델은 보안정책에 의해 시스템 자원에 대한 융통성있는 접근통제 기능을 제공하여 코드 이동성과 보안정책기반의 접근통제 모델이 결합된 자바 환경이 네트워크 환경에서 안전하고 효과적인 시스템 개발환경으로 사용될 수 있는 기반을 제공한다. 그러나, 잘못된 보안정책화일의 작성이나 여러 사용자에 의해 작성된 보안정책화일들에 의해 시스템 자원 보호에 대한 불일치성이 발생하면 전체 시스템의 보안이 침해되는 문제점을 가지고 있다.

3.3 Java 버전별 주요 보안특성 비교

표 1은 지금까지 살펴본 자바 버전별 주요 보안 특성을 비교하여 나타내고 있다.

표 1. 자바 버전별 주요 보안특성 비교

보안 특성	Java 1.0	Java 1.1	Java 2
전자서명이 않된 지역코드의 접근 통제	무제한 접근	무제한 접근	보안정책 기반
전자서명된 지역 코드의 접근통제	N/A	신뢰된 코드인 경우 무제한 접근 비신뢰된 코드인 경우 sandbox에 의해 통제	보안정책 기반
전자서명이 않된 원격코드의 접근 통제	sandbox 에 의해 통제	sandbox에 의해 통제	보안정책 기반
전자서명된 원격 코드의 접근통제	N/A	신뢰된 코드인 경우 무제한 접근 비신뢰된 코드인 경우 sandbox에 의해 통제	보안정책 기반
비밀성/무결성 지원 암호 서비스	N/A	JCE(Java Cryptographic Extension) 1.1	JCE 1.2
전자서명을 위한 전자서명 서비스	N/A	JCA(Java Cryptographic Architecture) DSA signature	JCA DSA signature

III. RBAC 모델의 Java 2 환경적용 관련연구

Java 2에 채택된 보안정책과 접근통제기술은 로컬 파일시스템이나 네트워크로부터 다운로드된 자바 프로그램이 사용자 시스템의 자원에 접근할 때 코드 중심(code-centric)의 인증 및 권한부여 기능을 수행한다. 한편, JAAS(Java Authentication and Authorization Service)는 자바 프로그램을 수행할 수 있는 사용자에 대한 인증과 접근권한을 부여하는 사용자 중심(user-centric)의 보안기능을 제공함으로써, Java 2 기술이 보안이 필수적인 다중 사용자 환경에서도 활용될 수 있게 되었다[6,7,14].

그러나, 사용자와 자바 프로그램의 수가 많아지고, 자바 프로그램이 수행할 수 있는 권한의 종류가 다양해지게 되면 JAAS에서 제공하고 있는 사용자 기반의 권한부여방식은 사용자, 자바 프로그램, 권한을 효과적으로 관리하는 데 문제점이 발생하게 된다. 따라서, 사용자를 기반으로 하는 방식보다 역할기반의 권한부여 및 관리체계가 효과적이다.

이 장에서는 역할기반 접근통제 모델과 Java 환경에서 사용자 인증 및 사용자 기반의 권한부여 기능을 수행하는 JAAS, 그리고 Java 2 환경에 역할기반 접근통제 모델을 적용하기 위한 관련연구들에 대해 기술한다.

1. 역할기반 접근통제 모델

역할기반 접근통제모델(RBAC: Role-Based Access Control Model)은 사용자의 권한이 개인의 신분이나 보안등급대신 조직내에서 사용자가 수행하는 기능(job function)에 의해 결정되는 특징을 가진다[3,4]. 이를 위하여 RBAC 모델은 사용자에게 권한을 직접 부여하지 않고, 조직의 기능에 따라 정의된 역할에 권한을 먼저 부여한 후, 사용자를 역할에 배정함으로써 사용자가 역할에 부여된 권한을 수행하는 동작구조를 가진다. 그리고, 상위역할이 하나 이상의 하위역할들에 배정된 권한을 상속하는 특징을 지원함으로써 계층구조를 가진 일반적인 기업환경의 관리구조를 자연스럽게 모델링한다(그림 4). 이 밖에도 RBAC 모델의 구성요소나 구성요소들간에 대응수(cardinality)나 임무분리(separation of duty) 등과 같은 제약조건(constraints)을 추가함으로써 응용영역별로 요구되는 다양한 보안정책을 지원하는 보안정책중립적(policy-neutral)인 특성을 가진다.

이와 같은 특성을 가진 RBAC 모델은 인터넷을 통해 많은 수의 사용자들로 구성된 기업환경의 효과적인 권한관리기능과 다양한 보안정책기술에 적합한 보안모델로서 활용되고 있으며, 그 적용범위도 지속적으로 확대되고 있다.

따라서, 보안정책기반의 접근통제기능을 제공하는 플랫폼 독립적인 Java 2 환경에 RBAC 모델을 적용하면, 많은 사용자로 구성된 분산 환경의 응용 프로그램에 대해 효과적인 권한관리 및 접근통제기능을 수행할 수 있다. 그러나, Java 2는 권한을 수행하는 단위가 자바 프로그램(어플리케이션, 애플릿)인데 비해 RBAC 모델은 사용자에 기반한 접근통제 기능을 수행하는 특성을 가지고 있어서 Java 2 환경에 RBAC 모델을 직접 적용하는데 문제점이 있다. 따라서, Java 2 환경에 자바 프로그램을 실

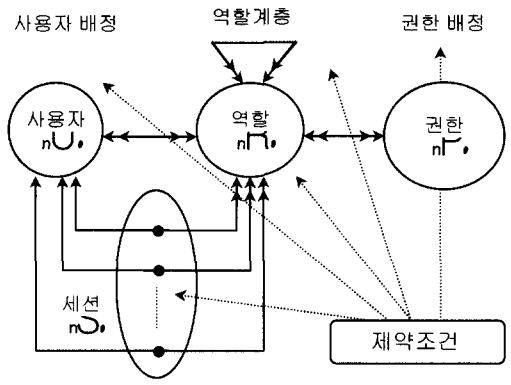


그림 4. RBAC모델 구성요소

행하는 사용자에 대한 인증과 권한부여 기능을 추가하기 위해 JAAS가 개발되었다[6,7].

2. JAAS(Java Authentication and Authorization Service)

JAAS는 Java 2 SDK 버전 1.3의 표준 확장 서비스로서 코드소스 기반 접근통제만을 제공하는 Java 2 보안기능에 코드를 수행하는 사용자에 따라 접근통제를 실행하는 기능을 추가로 제공한다. 여기서 코드는 어플리케이션 애플릿, 서블릿, 빈(bean) 등 모든 자바 프로그램을 의미한다.

JAAS의 인증기능은 자바 응용 프로그램의 수정 없이 인증기술을 변경할 수 있는 PAM(Pluggable Authentication Module) 구조를 가지며, 인증과정은 응용 프로그램의 호출에 의해 수행된다. JAAS의 권한부여 기능을 인증과정을 거친 사용자로부터 보호대상이 되는 시스템 자원을 보호하기 위해 Java 2의 접근통제기술과 함께 사용된다. 즉, 접근통제과정에서 코드소스와 함께 인증된 사용자 정보가 함께 고려된다.

JAAS는 공통, 인증(authentication), 권한부여(authorization) 클래스 집합으로 구성되며, 각 클래스 집합의 기능은 다음과 같다[6].

2.1 공통 클래스 집합

공통 클래스 집합에 포함된 클래스들은 인증과 권한부여 클래스 집합에 의해 공통으로 사용되는 클래스들로, Subject, Principal, Credential 클래스로 구성된다. Subject 클래스는 자원에 대한 접근을 요청한 사람 또는 서비스를 나타내며, 다른 Subject와 구별되는 하나 이상의 Principal을 가진다. 예를 들어, '사용자'를 나타내는 Subject는 '이름' Principal과 'SSN' Principal을 포함할 수 있다. 그리고, Subject는 보안관련 속성인 Credential을 소유할 수 있다. Credential은 공개키나 Kerberos 티켓과 같이 외부에 정보를 제공할 수 있는 공개(public) Credential과 비밀키처럼 외부로부터의 접근을 허용하지 않는 비밀(private) Credential로 구성된다.

특정 Subject가 자원에 대한 연산을 수행할 수 있는지 점검하는 메소드로서 doAs(final Subject subject, final java.security.PrivilegedAction action)가 있다. doAs() 메소드는 현재 쓰레드의 접근통제문맥(AccessControlContext)에 매개변수로 전달되는 'subject'를 결합하여 'action' 메소드를 수행한다. 이 과정을 통해 'subject'가 'action' 메소드를 수행할 수 있는지 여부를 검사하게 된다. 접근통제문맥은 하나 이상의 보호 도메인으로 구성된다. doAS() 메소드와 유사한 메소드로서 doAsPrivileged(final Subject subject, final java.security.PrivilegedAction action, final java.security.AccessControlContext acc)가 있다. 이 메소드는 doAs()에서 현재 쓰레드와 'subject'를 결합하는 대신 매개변수로 주어지는 'acc'에 결합하여 접근권한 여부를 결정하는 특징을 가진다.

2.2 인증 클래스 집합

인증 클래스 집합은 사용자 인증기능을 수행하기 위해 로그인문맥(LoginContext) 클래스와 로그인 모듈(LoginModule), 콜백핸들러(Callback-Handler), 콜백(Callback) 인터페이스로 구성된다. 로그인문맥 클래스는 Subject를 인증하기 위한 기본적인 기능과 인증기술과 무관하게 응용 프로그램을 개발할 수 있는 기능을 제공한다. 로그인모듈 인터페이스는 전통적인 사용자 ID/암호방식으로부터 스마트카드나 생체인식방식 등 다양한 인증기술들이 응용 프로그램과 연동될 수 있도록 하는 기능을 제공한다. 인증과정에 하나 이상의 로그인모듈이 사용될 수 있다.

인증과정에서 사용자와 로그인모듈간 정보의 교환이 필요하게 되는 경우, 콜백핸들러가 이용된다. 응용 프로그램은 콜백핸들러 인터페이스를 구현하여 로그인모듈에게 전달하며, 로그인모듈에서는 구현된

콜백핸들러를 이용하여 사용자로부터 정보를 입력받거나 사용자에게 정보를 제공하게 된다.

JAAS에서 수행되는 인증과정을 개략적으로 기술하면 그림 5와 같다.

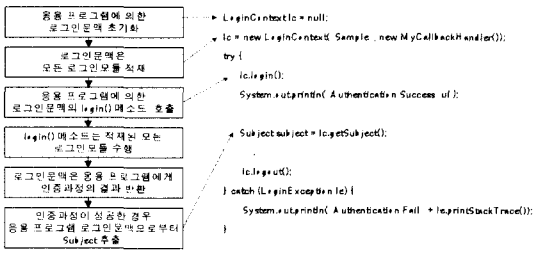


그림 5. JAAS 인증과정

2.3 권한부여 클래스 집합

JAAS 권한부여는 인증과정이 성공적으로 완료된 Subject에 대해 JAAS에서 정의된 보안정책에 따라 세분화된 접근통제를 수행한다. Subject의 자원에 대한 접근은 Subject 클래스의 doAs() 메소드를 통해 이루어진다. 권한부여 클래스 집합은 보안정책을 나타내기 위한 정책(Policy) 클래스, JAAS에서 이용되는 기본적인 권한을 표현을 위한 Auth-Permission 클래스, Subject의 비밀 Credential 보호기능을 제공하는 Private CredentialPermission 클래스로 구성된다.

2.4 로그인 구성정보화일과 JAAS 보안정책화일

JAAS는 코드소스기반의 Java 2 보안정책화일 외에도 로그인 구성정보화일, 사용자 기반의 보안정책을 기술하는 JAAS 보안정책화일들을 참조하여 동작한다[7]. 로그인 구성정보 파일은 사용자 인증에 필요한 하나 이상의 로그인 모듈들에 대한 정보를 기술한다. 그리고 JAAS 보안정책화일은 인증이 성공적으로 완료된 사용자가 수행할 수 있는 권한에 대한 정보를 나타내며, Java 2 보안정책 표기법을 확장한 형태로서 그 문법과 예는 그림 6과 같다. 그림 6의 예제는 "http://foo.com"로부터 다운로드되고 "foo" 사용자에 의해 전자서명된 자바 프로그램이 Principal로서 "admin"을 가진 Subject에 의해 수행되는 경우 "C:/datastore/admin" 디렉토리에 존재하는 모든 파일에 대해 "read", "write"

```
grant codeBase [ "URL" ],
      signedby [ "signer" ],
      Principal [ Principal_Class ]
                  [ "Principal_Name" ],
      [ Principal ... ] {
    permission Permission_class
                  [ "Target_Name" ]
                  [, "Permission_Actions" ]
                  [,
    signedBy "SignerName" ];
    ...
};

grant codeBase "http://foo.com",
       signedBy "foo"
       Principal
        com.sun.security.auth.NTPrincipal
                "admin" {
       permission java.io.FilePermission
                "C:/datastore/admin",
                "read, write";
};
```

그림 6. JAAS 보안정책 문법 및 사용 예

연산이 허용됨을 나타낸다.

로그인 구성정보 파일에서 지정된 모듈에 의해 인증이 완료된 사용자는 하나 이상의 Principle이 부여되며, 사용자가 정보를 접근할 때 인증과정에서 부여된 Principle과 JAAS 보안정책화일에 지정된 Principle을 비교하게 된다. 자바 인터프리터 수행 때 JAAS 보안정책화일과 로그인 구성정보화일의 지정은 "-Djava.security.auth.policy", "-Djava.security.auth.login.config" 옵션을 사용하거나 'java.security' 보안속성 화일의 'auth.policy.url.n', 'login.config.url.n' 속성값을 이용한다.

JAAS는 코드소스를 기반으로 접근통제기능을 수행하는 코드중심의 Java 2 보안기능에 Subject 기반의 세분화된 접근통제기능을 추가함으로써 여러 사용자로 구성된 일반적인 시스템 환경에서도 자바 기술이 유용한 접근통제기능으로 활용될 수 있는 기능을 제공한다.

3. 관련 연구

Java 2 보안모델에 RBAC 모델을 적용하기 위한 주요 연구로는 역할의 지원을 위해 확장된 보호도메인 연구, JAAS 기반의 RBAC 적용연구, 보호도메인 확장연구와 JAAS의 결합 연구 등이 있다.

3.1 보호 도메인을 확장한 RBAC 모델 적용연구

이 연구에서는 Java 2가 여러 응용분야에서 요구되는 새로운 Permission 클래스를 정의할 수 있는 특징을 이용하여 역할을 유일한 이름을 가진 하나의 Permission 클래스로 정의하였다[1]. 그리고 동종의 Permission 객체들로 구성된 여러 개의 PermissionCollection 객체들(Permissions 객체)을 역할에 배정함으로써 RBAC 모델의 역할-권

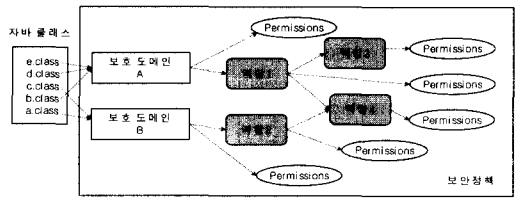


그림 7. 역할을 지원하는 확장된 보호 도메인

한 배정과정을 수행한다. 이 과정에서 역할 Permission이 다른 역할에 배정됨으로써 역할과 역할간에 포함관계(inclusion)가 형성되고, 이 포함관계에 의해 상위 역할은 하위 역할들에 배정된 권한을 상속하게 된다. 이 연구에서 보호 도메인에 배정된 권한은 보호 도메인에 직접 배정된 권한(ProtectionDomain 클래스의 implies() 메소드에 의해 계산)과 역할간 포함관계에 상속된 권한의 합집합으로 정의된다.

그림 7은 역할을 지원하는 확장된 보호 도메인 구조를 기술하고 있다. 자바 프로그램을 구성하는 자바 클래스는 클래스로더에 의해 클래스로 정의되는 시점에서 하나의 보호 도메인과 결합되며, 보호 도메인에 포함된 Permissions와 역할들간 포함관계에 의해 상속되는 Permissions에 포함된 권한을 실행할 수 있다. 예를 들면, 그림 7의 보호 도메인 A에 배정된 권한은 보호 도메인 A에 직접 배정된 Permissions와 도메인 A에 포함된 역할들 중 활성화된 역할에 배정된 Permissions의 합집합으로 구성된다. 여기서 보호 도메인에 포함된 역할들 중 어떤 역할을 언제 활성화시킬 것인지를 결정하는 역할 활성규칙(role activation rule)은 보안정책에 의해 결정된다. 역할 활성규칙의 예로는 보호 도메인에 포함된 모든 역할을 활성화하는 방법, 사용자가 활성화될 역할을 선택하는 방법, 보호 도메인에 지정된 디폴트 역할의 활성화 방법 등이 있다.

역할개념을 도입을 위해 보호 도메인을 확장한 모델의 구현방안으로 현재 Java 2 환경의 수정없이 구현하는 방법과 Java 2 클래스의 확장을 통한 구현방법을 제시하고 있다[1]. 첫 번째 방법은 각 보호 도메인에 포함된 권한의 집합을 보호 도메인에 포함된 역할의 권한집합을 미리 계산해 놓는 방법이다. 이 때 역할간의 포함관계에 의해 권한상속이 고려되어야 한다. 이 방법은 정적으로 보호 도메인의 권한을 계산하기 때문에 임무분리 제약조건의 적용이나 역할의 추가, 변경, 삭제 과정때마다 보호 도메인의 권한을 다시 계산해야 하는 문제점을 가진다. 두 번째 방법은 현재 final 클래스로 지정된 Permissions 클래스를 수정하는 방법으로 표준 Java 2에서는 지원되지 않는 구현방법이다. 이 방식은 역할간 포함관계를 이용하여 보호 도메인에 포함된 권한의 집합을 동적으로 계산할 수 있도록 클래스와 인터페이스를 추가함으로써 첫 번째 구현방법에 비해 RBAC 모델의 특성을 효과적으로 지원한다. 그러나, 이 방식은 현재의 표준 Java 2 환경에 적용이 불가능한 문제점이 있다.

이 연구는 Permission 클래스로서 역할을 정의하고 역할들간 포함관계를 통해 권한상속기능을 제공함으로써 많은 Permissions들로 구성된 시스템에서 효과적인 Permissions 관리기능을 제공한다는 장점을 제공하고 있으나, 자바 프로그램을 실행하는 사용자에 대한 정의와 지원기능을 제공하지 못하여 RBAC 모델의 충실한 적용에는 미흡한 점이 있다.

3.2 JAAS 기반의 RBAC 모델 적용 연구

JAAS는 코드기반의 접근통제기능만을 제공하는 Java 2에 사용자기반의 접근통제기능을 추가하기 위해 개발된 자바 패키지이다. JAAS 보안정책(그림 6 참조)은 코드 소스에 Principal 정보를 추가함으로써 자바 프로그램을 실행하는 사용자 기반의 접근통제기능을 수행한다. 이를 위하여 JAAS는 Java 2 SDK 버전 1.3에 추가된 Domain Combiner 인터페이스를 구현한 SubjectDomain Combiner 클래스를 정의하여 보호 도메인과 Subject를 결합시키는 기능을 수행한다.

한편, JAAS에서는 Principal 클래스(그림 6의 *Principal_Class*)를 이용하여 Principal기반 외

에 그룹 또는 역할기반 접근통제기능을 제공한다. 그룹과 역할을 각각 이름을 가진 Principal 타입으로 간주하고 그림 8과 같은 JAAS 보안정책화일을 이용하여 일반적인 Principal과 동일한 방법으로 그룹과 역할에 부여된 권한을 기술한다.

```
grant Principal foo.Team "GeneralUser" {
   permission java.io.FilePermission
          "C:/datastore/general", "read";
};

grant Principal foo.Role "administrator" {
   permission java.io.FilePermission
          "C:/datastore/admin", "read, write";
};
```

그림 8. JAAS 보안정책화일의 그룹기반 및 역할기반 접근통제 사용 예

그룹간 포함관계 또는 역할간 계층구조의 판단을 위해 JAAS는 PrincipalComparator 인터페이스를 정의하고, Principal을 구현하는 클래스가 PrincipalComparator 인터페이스를 함께 구현하도록 한다. 예를 들면, 그림 9의 역할 'user'가 역할 'administrator'의 하위역할이라 가정하면, 클래스 'foo.Role'은 입력으로 주어진 'subject'에 'administrator' 역할이 결합되어 있으면 'true'를 리턴하도록 PrincipalComparator 인터페이스의 implies() 메소드를 구현하여, 역할 계층구조에 의한 권한의 상속기능을 제공한다.

이 연구는 Principal기반의 접근통제기능을 제공하는 JAAS에서 Principal의 한 형태로 그룹과 역할을 정의하고, 역할로 사용되는 Principal 클래스에서 역할간 계층구조와 권한의 상속기능을 제공하는 방법을 채택하고 있다. 다양한 인증기술이 적용될 수 있는 로그인모듈을 통해 사용자 인증기능이 추가되어 RBAC 모델의 사용자 구성요소를 지원하고 있다. 그러나, 사용자-역할 배정, 임무분리 등과 같은 RBAC 모델의 주요 제약조건의 기술과 수행절차가 제시되지 못한 문제점이 있다.

3.3 보호 도메인 확장연구와 JAAS의 결합

이 연구는 3.1과 3.2 연구를 결합한 방법으로 사

```
public inteface PrincipalComparator {
   boolean implies(Subject subject);
};

grant Principal foo.Role "user" {
   permission java.io.FilePermission
                "C:/tmp/-", "read, write";
};
```

그림 9. JAAS에서 역할간 상속관계 지원

용자 인증기능, 역할-권한, 사용자-권한 배정기능을 JAAS에서 제공하는 인증과 보안정책화일을 이용하고(그림 10의 (b)), 역할계층과 사용자-역할 배정은 Policy 클래스를 확장하여 제공(그림 10의 (a))함으로써 전통적인 RBAC 모델의 구성요소와 동작특성을 대부분 지원하고 있다[2].

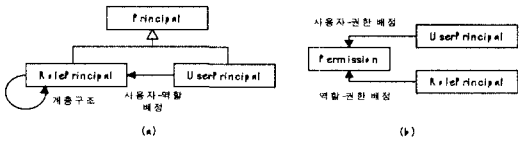


그림 10. UserPrincipal과 RolePrincipal을 이용한 RBAC 모델 지원

먼저 Principal을 사용자와 역할을 의미하는 UserPrincipal과 RolePrincipal로 세분하고, UserPrincipal과 RolePrincipal간의 사용자-역할배정, RolePrincipal들간의 계층구조, Permssion과 UserPrincipal, Permission과 Role Principal 배정에 대한 모델을 제시하였다.

Subject는 하나 이상의 Principal을 포함할 수 있지만, 하나의 UserPrincipal만을 가지도록 하였고, UserPrincipal은 JAAS에서 제공하는 인증과정을 거치도록 지정하였다. 또한, 역할에 배정된 권한들은 주어진 보호 도메인에서 역할이 활성화된 경우 보호 도메인과 결합된 Subject에 의해 사용될 수 권한을 부여하고 있다.

이 밖에도 역할에 배정될 수 있는 사용자의 수, 시간지원 제약조건, 정적 및 동적 임무분리 제약조건을 기술할 수 있는 별도의 클래스를 제시하여 JAAS를 이용한 방법보다 다양한 RBAC 모델의 특성을 지원하고 있다(그림 11).

```
grant [ role "role-name" | user "user-name" ]
{
  role "role-name1" [default];
  ...
  role "role-namen" [default];
};
```

(a) 사용자-역할, 역할계층 지정

```
[ static | dynamic ] mutex
{
   role "role-name1"
   ...
   role "role-namen";
 [ user "user-name1";
   ...
   user "user-namem"; ]
};
```

(b) 임무분리 제약조건 지정

그림 11. 사용자-역할 배정, 역할계층과 임무분리지정 보안정책화일 문법

Ⅳ. 활용방안

지금까지 살펴본 바와 같이 Java 2 보안모델은 보안관리자와 각 사용자에 의해 정의된 보안정책에 따라 유연한 접근통제 기능을 제공함으로써 매우 엄격한 Sandbox 보안모델의 제약점과 신뢰된 사용자에 의해 전자서명된 자바 프로그램에게 모든 권한을 부여하는 문제점을 보완하고 있다. 그러나, 자바 환경은 사용자라는 개념 대신 자바 프로그램을 대상으로 권한을 부여하는 특징을 인하여 여러 사용자가에 의해 수행되는 응용 프로그램을 자바를 이용하여 개발하는데 어려움이 있었다. 한편, RBAC 모델은 상업환경의 보안요구사항을 기초로 정립된 접근통제모델로서 특정 보안정책 대신 모델 구성요소의 변경을 통해 다양한 보안정책을 지원할 수 있는 특징으로 인하여 현재 적용분야가 확대되고 있다. 따라서, 융통성있는 보안정책을 지원하는 RBAC 모델을 분산 응용 프로그램 개발에 적합한 자바 환경에 적용하면 자바 환경에서 사용자 개념의 지원, 여러 사용자로 구성된 응용환경에서의 효과적인 권한관리 및 보안정책기반의 접근통제 등의 장점을 얻을 수 있다.

이 장에서는 RBAC 모델을 지원하는 Java 2 환경이 활용될 수 있는 사례에 대해 살펴본다.

1. 자바, RBAC 기반의 워크플로우 보안서비스

워크플로우 시스템은 다양한 플랫폼으로 구성된 분산환경에서 여러 사용자에 의해 수행되어야 하는 주어진 업무를 자동적으로 실행하는 기능을 제공한다[18,19,20]. 워크플로우 시스템의 구성이나 동작, 관리구조는 워크플로우 시스템이 적용되는 응용분야의 특성에 따라 다양한 구조를 가진다. 그리고, 워크플로우 실행과정에서 발생할 수 있는 보안위협에 대응하는 다양한 보안서비스에 대한 연구도 진행되고 있

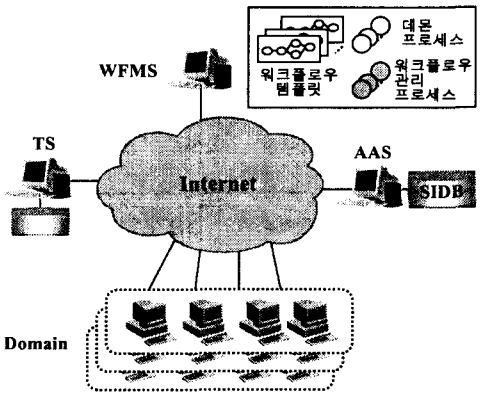


그림 12. 워크플로우 보안서비스 구성 예

다[22]. 따라서, 이질적인 분산환경에서 동작할 수 있는 자바 언어와 보안정책기반의 Java 2 보안모델에 역할기반의 권한부여 및 권한관리기능을 제공하는 RBAC 모델을 결합하면 워크플로우 시스템의 인증과 권한부여 보안서비스 구축에 효과적으로 활용될 수 있다. 그림 12는 RBAC 모델을 적용한 자바 기술이 워크플로우 시스템의 보안서비스에 적용될 수 있는 한 구조를 나타내고 있다.

1.1 구성요소

그림 12에 예제로 제시된 워크플로우 시스템을 구성하는 각 구성요소의 기능과 특징은 다음과 같다.

· 워크플로우 관리시스템

워크플로우 관리시스템(WFMS: Workflow Management System)은 워크플로우 수행의 감독, 관리기능을 수행한다. 각 워크플로우는 워크플로우를 구성하는 태스크(프로세스)들의 수행순서 및 제약조건 등으로 기술된 워크플로우 템플릿으로 명세된다. 그리고, 워크플로우의 각 태스크를 수행할 수 있는 사용자 또는 역할, 그리고 임무분리 제약조건을 명시하여 워크플로우 실행에 필요한 보안요구사항을 위

해 기술할 수 있다[5]. 워크플로우 템플릿의 명세는 워크플로우 설계자와 보안관리자에 의해 수행된다. 워크플로우 관리 프로세스는 현재 실행중인 각 워크플로우에 실행순서, 임무분리 등과 같은 제약사항 수행에 대한 감독 및 관리기능을 수행한다.

· 태스크 서버

태스크 서버(TS: Task Server)는 자바 클래스 형태로 작성된 워크플로우 태스크를 저장, 관리하는 기능을 수행하며, 태스크를 수행하는 워크플로우 엔진의 요청에 의해 태스크를 전송한다. 기존 응용 프로그램을 지원하기 위한 wrapper 프로그램도 태스크 서버에서 관리될 수 있다. 자바 클래스로 작성된 태스크들을 태스크 서버에 의해 관리하면, 태스크에 대한 효과적인 업그레이드와 버전관리가 가능하다.

· 도메인

도메인(Domain)은 보안정보 데이터베이스에 저장된 동일한 보안정책의 통제를 받는 사용자, 역할, 워크플로우 및 워크플로우 엔진으로 구성된다.

· 인증 및 권한부여 서버

인증 및 권한부여 서버(AAS: Authentication and Authorization Server)는 보안정보 데이터베이스에 저장된 인증 및 RBAC 정보를 참조하여 워크플로우 수행에 참여하는 사용자와 프로세스에 대한 인증, 권한부여 기능을 수행한다.

· 보안정보 데이터베이스

보안정보 데이터베이스(SIDB: Security Information Database)는 워크플로우 시스템에 참여하는 사용자의 인증정보(메인, 암호 또는 인증서, 권한부여에 필요한 사용자-역할배정, 역할계층, 역할-권한배정, 제약조건) 등에 대한 정보를 저장, 관리하는 기능을 수행한다. 그리고, SIDB는 보안정책화일들(Java 2, JAAS 보안정책화일 등)은 각 도메인마다 별도로 저장, 관리된다.

워크플로우 엔진은 자신이 속한 도메인의 보안정책화일이나 인증정보를 SIDB로부터 참조하며, SIDB에 저장, 관리되는 보안정책에 따라 전체 워크플로우 시스템의 보안관리기능이 수행한다. 각 도메인의 보안정책화일이나 인증정보를 지정하는 방법으로는 도메인에 포함된 각 시스템의 'java.security' 속성화일의 'policy.url', 'login.config.url', 'auth.policy.url' 속성값을 SIDB시스템과 보안정책화일 이름으로 구성된 URL을 지정하는 중앙관리방법과 각 워크플로우 엔진에 보안정책화일을 저장하고 워크플로우 엔진이 설치된 시스템의 관리자가 보안정책화일을 참조하거나 수정할 수 있게 하는 분산관리방법이 있다.

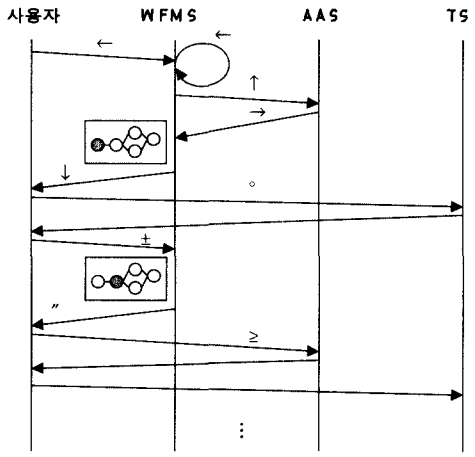


그림 13. 워크플로우 보안서비스 실행 절차

1.2 동작 구조

그림 13은 그림 12와 같이 구성된 워크플로우 시스템에서 워크플로우가 실행되는 순서를 나타내고 있다.

①, ①': 도메인에 소속된 사용자나 WFMS의 데몬 프로세스에 의해 워크플로우 실행요청이 발생한다.

②: 워크플로우 실행권한이 워크플로우를 호출한 사용자에게 부여되어 있는지 확인하기 위해 WFMS로부터 도메인 이름, 사용자 ID/암호, 워크플로우 ID 등의 정보가 AAS로 전달된다.

③, ④: AAS에 의해 인증이 성공적으로 완료되면 워크플로우 실행을 요청한 사용자에게 배정된 역할정보와 함께 워크플로우 템플릿을 구성하는 첫 태스크에 대한 정보를 사용자에게 전달한다.

⑤: 인증과 역할배정에 따른 권한을 부여받은 사용자는 TS로부터 태스크에 해당하는 자바

프로그램을 다운로드 받아 수행한다.

⑥: 태스크의 수행이 완료되면, 완료상태 정보 등을 WFMS에 전송한다.

⑦: WFMS는 수행이 완료된 태스크의 완료상태에 따라 다음에 수행될 태스크와 태스크를 수행할 사용자 또는 역할을 결정하여 선택된 사용자 시스템(워크플로우 엔진)에 통보한다.

⑧: 태스크를 사용할 사용자는 사용자 ID/암호를 AAS를 통해 인증받은 후 자신이 수행할 태스크를 다운로드 받아 수행한다. 이와 같은 과정이 워크플로우를 명세하는 워크플로우 템플릿에 의해 계속된다.

그러나, 이와 같은 워크플로우 보안서비스 구조는 TS나 AAS, SIDB의 장애발생때 전체 시스템의 성능저하 문제점과 서로 다른 도메인간의 인증 및 접근통제 기능에 대한 보완연구가 필요하다.

V. 결 론

본 논문에서는 코드 소스와 보안정책을 기반으로 접근통제 기능을 제공하는 Java 2 보안환경에 RBAC 모델을 적용하는 연구들에 대해 살펴보았다. 그리고, RBAC 모델이 적용된 Java 2 기술의 응용분야에 대한 예를 제시하였다. 이를 위해 먼저 자바 버전별 보안모델의 특성과 문제점을 비교하고, RBAC 모델을 Java 2 환경에 적용하는 연구들의 주요 특징을 기술하였다.

사용자 개념을 추가하여 역할 기반의 권한부여 기능이 보완된 Java 2 기술은 네트워크로부터 다운로드된 프로그램을 플랫폼과 무관하게 웹 브라우저에서 수행할 수 있다는 단편적인 장점을 뛰어넘어, 효과적인 권한관리기능의 제공과 함께 인터넷 환경에서 여러 사용자가 참여하는 다양한 응용 프로그램의 안전한 실행환경 구축하는데 효과적으로 사용될 수 있게 되었다.

그러나, 역할기반 접근통제모델이 적용된 자바 기술이 대규모 인터넷 환경에서의 인증기능 제공을 위해 X.509와 같은 인증서를 이용하거나 속성 인증서(AC: Attribute Certificate)에 기반 권한부여 방법[21], 서로 다른 도메인에 속한 사용자와 역할에 대한 권한부여, 그리고 자바 클래스로 구성된 응용 프로그램에 대한 실행환경 구현에 대한 보완연구가 필요하다.

참 고 문 헌

[1] Giuri L., "Role-Based Access Control in Java," Proceedings of the Third ACM Workshop on Role-Based Access Control, ACM Press, 1998.

[2] Giuri L., "Role-Based Access Control on the Web Using Java," Proceedings of the Fourth ACM Workshop on Role-Based Access Control, ACM Press, 1999.

[3] Ferraiolo D., Cuguni J., Kuhn R., "Role-Based Access Control(RBAC): Features and Motivations," Proceedings of 11th Annual Computer Security Application Conference, New Orleans, LA, December 1995.

[4] Sandhu R. S., Coyne E. J., Feinstein H., Youman C. E., "Role-Based Access Control Models," IEEE Computer, Vol 29, No. 2, February 1996.

[5] HyungHyo Lee, BongNam Noh, "An Integrity Enforcement Application Design and Operation Framework," International Workshop on Security(IWSEC'99) held conjunction with IEEE ICPP'99, 1999.

[6] Java Authentication and Authorization Service(JAAS),http://www.javasoft.com/security/jaas

[7] Java Authentication and Authorization Service(JAAS) 1.0 Developer's Guide, http://java.sun.com/security/jaas/api.html

[8] Gong L., Joy B., Steele G., The Java Language Specification, Addison-Wesley, August, 1996.

[9] Lindholm T., Yellin F., The Java Virtual Machine Specification, Addison- Wesley, 1997.

[10] McGraw G., Felten W. F., Java Security: Hostile Applets, Holes and Antidotes, John Wiley & Sons, 1997.

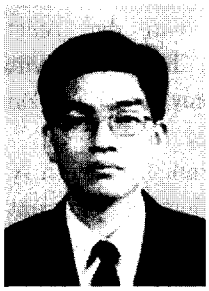
[11] McGraw G., Felten W. F., Securing Java, John Wiley & Sons, 1999.

[12] Scott Oaks, Java Security, O'Reilly, 1998.

[13] Marco P., Dunane F. R., Deepak G., Milind N., Ashok K. R., Java 2 Network

Security, 2nd Ed., Prentice- Hall, 1999.
[14] Balfanx D., Gong L., "Experience with Secure Multi-Processing in Java," Proceedings of ICDCS, May 1998.
[15] Gong L., Mueller M., Prafullchandra H., Schemers R., "Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2," Proceedings of USENIX Symposium on Internet Technologies and Systems, Montrey, 1997.
[16] Sun Microsystems, "Frequently Asked Questions-Java Security," http://java.javasoft.com/sfaq
[17] Princeton Secure Internet Programming Team, "Java Security FAQs," http://www.cs.princeton.edu/sip/java-faq.html
[18] Sushil Jajodia, "Interview: Amit Sheth on Workflow Technology," IEEE Concurrency, April-June 1998.
[19] Amit P. Sheth, Wil van der Aalst, Ismailcem B. Arpinar, "Process Driving the Networked Economy," IEEE Concurrency, July-September 1999.
[20] Georgakopoulos D., Hornik M., Sheth A., "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure," Distributed and Parallel Databases, 3(2): 119-154, 1995.
[21] R. Oppliger, "Authorization Methods for E-Commerce Applications," The Eighteenth Symposium on Reliable Distributed Systems, October 1999.
[22] Mei Fan, "Security for the METEOR Workflow Management System," Master's Thesis, University of Georgia, 1999.

〈著者紹介〉



이 형 효 (HyungHyo Lee)

1987년: 전남대학교 전산학과 졸업
1989년: 한국과학기술원 전산학과 석사
2000년: 전남대학교 전산학과 박사
1990년~1992년: 삼보컴퓨터 기술연구소
1993년~1997년: 한국통신 연구개발원
1995년: 정보처리기술사(전자계산조직응용)
2000년~현재: 광주과학기술원 BK21 정보기술사업단 Post-Doc.
관심분야 : 정보보안, 보안모델, 통신망관리 등

이 동 익 (Dong-Ik Lee)

1985년 : 영남대학교 전기공학과 졸업
1989년 : Osaka Univ. 전자공학과 석사
1993년 : Osaka Univ. 전자공학과 박사
1990년~1995년: Osaka Univ. 전자공학과 Research Associate
1993년~1994년: Univ. of Illinois at Urbana-Champaign Visiting Assistant Professor
1995년~현재: 광주과학기술원 정보통신공학과 부교수
관심분야 : 에이전트 시스템 및 보안, 비동기회로 설계/CAD, Petri net 이론 등

노 봉 남 (BongNam Noh)

1978년: 전남대학교 수학교육과 졸업
1982년: 한국과학기술원 전산학과 석사
1994년: 전북대학교 전산통계학과 박사
1983년~현재 전남대학교 컴퓨터정보학부 교수
관심분야 : 객체지향시스템, 통신망관리, 정보보안, 멀티미디어와 정보사회 등