OSGi 기반 동적 RBAC 모델

Dynamic RBAC Model based on OSGi

김인태*, 정경용**, 임기욱***, 이정현*
인하대학교 정보공학과*, 상지대학교 컴퓨터정보공학부**, 선문대학교 컴퓨터정보학부***

In-Tae Kim(inking007@gmail.com)*, Kyung-Yong Chung(cyjung@sangji.ac.kr)**,
Kee-Wook Rim(rim@sunmoon.ac.kr)***, Jung-Hyun Lee(jhlee@inha.ac.kr)*

요약

홈 네트워크 환경에서 사용자 인증 및 권한 부여 메커니즘은 사용자의 정보와 편의성에 밀접한 관련을 가지고 있는 중요한 보안 이슈이다. 현재 홈 게이트웨이로써 가장 널리 사용되고 있는 OSGi 서비스 플랫폼도 이러한 메커니즘을 제공하고 있다. OSGi 제공하는 기존의 권한 부여방식은 기본적인 역할 기반 접근 제어(RBAC) 모델 기반으로 설계 되어 이는 효과적이지 못하다는 문제점이 있다. 본 논문에서는 OSGi 기반의 동적 역할 기반 접근 제어 모델을 제안하였다. 제안된 방법에서는 기존 프레임워크를 절대적 롤뿐 아니라 상대적 롤을 지원하도록 확장하고 롤 검사를 서비스 번들에게 위임함으로써 효과적인 접근제어 메커니즘을 제안한다. 마지막으로 AspectJ와 자바 어노테이션을 이용하여 제안한 프레임워크를 구현한다.

■ 중심어 : | OSGi | 접근제어 | 보안 |

Abstract

In home network environments, the user authentication and authorization associated user's information and usability may be important security issue. The OSGi service platform, a well-known home network gateway already specifies the mechanism of that. The traditional authority method provided OSGi implements simple RBAC(Role Based Access Control) model. This is difficult to support efficient access control. In this paper, we propose the dynamic RBAC model based on OSGi. The proposed method describes the extended framework that manage two roles named as absolute role and relative role, extend existed framework with relative role and propose programming model to enable dynamic access control. Finally, we implement the proposed framework using AspectJ and Java annotation.

■ keyword : | OSGi| RBAC| Authority |

I. 서 론

오늘날 마크와이저[1]가 제안한 것처럼 많은 정보가 서로 공유되고 다양한 장치들이 연결되어 있는 유비쿼터스 시대로 변해가고 있다. 이러한 시대 흐름에 맞추어 다른 환경에서 이기종 장치들을 상호 연결할 수 있는 개방형 표준 인터페이스에 대한 많은 연구들이 진행되고 있다. 그 중에서도 OSGi[2]는 자바기반의 개방형

* "본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었습니다."
접수번호 : #081022-003
접수일자 : 2008년 10월 22일
심사완료일 : 2009년 01월 19일
교신저자 : 김인태, e-mail : inking007@gmail.com

표준 프로그래밍 인터페이스를 제공하여 현재 홈 네트워크에서 서비스 게이트웨이 역할로 많이 사용되고 있다. 사용 초기에는 홈 네트워크 게이트웨이에 국한되었으나 최근에는 모바일, 텔레메틱스와 같은 다양한 유비쿼터스 환경으로 확장되어 가고 있다. 홈게이트웨이는 사용자가 등록된 원격 서비스에 접속하여 댁내 장치들을 이용하고 관리할 수 있게 해준다. 이러한 기능을 수행하는데 보안은 필수적인 요소라 할 수 있다. OSGi 보안은 기본적으로 자바 보안[3]과 자바프로그래밍 모델을 기반으로 하고 있으며 몇 가지 권한을 관리하여 번들간 상호작용을 검사한다. 서비스 객체의 등록, 접근을 제어 위한 ServicePermission, 패키지들을 임포트, 익스포트를 제어를 위한 PackagePermission 그리고 관리자 서비스에 접근하기 위한 AdminPermission을 정의하고 있다. 또한 사용자 인증과 사용자 권한 부여 기능을 컴포넌트 명세서에 정의하고 있다.

사용자 인증 및 권한 부여는 사용자의 정보와 편의성에 밀접한 관련을 가지고 있는 중요한 보안 이슈이다. 사용자 인증은 아이디/패스워드 방식, 인증서 그리고 지문과 같은 바이오 정보를 이용하여 다양하게 처리하고 있다. 그리고 권한 부여 방법은 전통적으로 MAC(Mandatory Access Control)과 DAC (Discretionary Access Control) 방식이 사용되어 오다가 1990대 RBAC (Role based Access Control)[4]가 소개되면서 최근 가장 많이 쓰이고 있다. OSGi에서도 RBAC 모델에 기반한 접근 제어방식을 제공하고 있다. 하지만 기본적인 롤-사용자-권한 매핑 방식으로만 접근 제어를 수행하여 효과적이지 못하다는 문제점이 있다. 여러 연구에서 OSGi에 대한 보다 나은 접근제어 방법을 제안[5][6]하고 있으나 이는 단순히 XML 설정을 통한 접근제어를 적용하거나 XACML[7]을 그대로 적용함으로써 유연하고 동적인 접근제어가 어렵다. 아직까지 대부분 홈 네트워크 환경에서는 서비스들을 통합 형태로 설치하고 관리하지만 점차 다양한 서비스와 장치가 늘어남에 따라 통합 관리 기능을 수행하기 어려워지고 있다. 본 논문에서는 기존 OSGi에서 사용하고 있는 접근제어 방법을 확장하여 효과적이며 정보 변화에 적응이 뛰어난 접근제어 방법을 제안하고 관점 지향 프로그래밍(Aspect-oriented Programming)[8] 기법을 이용하여 구현한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 OSGi 접근제어 방법과 관점 지향 프로그래밍 방법에 대해 설명하며 3장에서는 본 논문에서 제안하는 접근제어 방법과 구현에 대해 기술한다. 4장은 기존 방법과 비교 평가 하며 5장에서는 본 논문의 결론을 맺는다.

II. 관련 연구

본 장에서는 OSGi의 기본 구조와 접근 제어 방법에 대해 살펴 보고 관점 지향 프로그래밍의 기본 개념에 대해서 설명한다.

1. OSGi 프레임워크

OSGi 프레임워크는 번들이라고 불리는 컴포넌트를 설치하고 서비스를 등록 실행하기 위한 프로그래밍 모델을 지원한다. 프레임워크 구조는 [그림 1]과 같이 몇 개의 계층으로 이루어 져있다. 실행 환경은 자바 가상머신 이며 기본적으로 프레임워크를 구성하는 시스템 번들이 있다. 그리고 응용 서비스에 따라 적당한 서비스 번들이 배치된다.

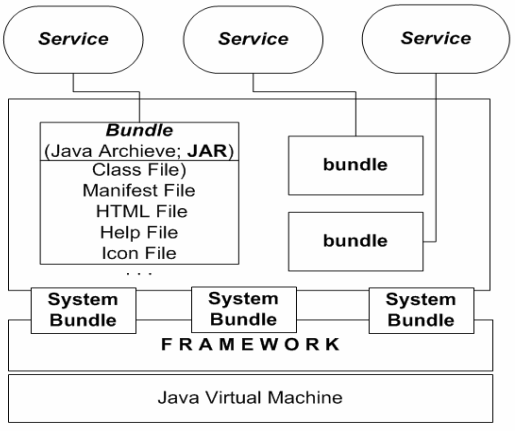

그림 1. OSGi 구조.

OSGi는 User Admin 서비스를 통해 사용자 인증 및

권한 부여를 수행한다. 이 서비스는 현재 OSGi V4 에 User Admin Service Spec 1.1이 정의 되어 있다. 권한 부여 방식은 RBAC 구조를 기반으로 하고 있다. Role 인터페이스를 상속받은 User, Group 인터페이스를 이용하여 사용자 정보와 집합을 표현하며 액션 그룹에 Group 객체를 할당하여 권한 검사를 하는 방식이다.

다음은 Spec 1.1 에 기술되어 있는 액션 그룹과 그룹을 정의 하는 예이다.

Administrators = { Elmer, Pepe, Bugs }
Family = { Elmer, Pepe, Daffy }
AlarmSystemActivation
required = { Administrators }
basic = { Family }

Administrators, Family라는 그룹을 정의하고 각각 사용자가 할당되어 있다. 또한 AlaramSystemActivation 라는 액션 그룹을 정의하고 Administrator를 required로 정의하고 Family는 basic으로 정의하였다. 이런 경우 AlarmSystemActivation을 접근할 수 있는 사용자는 Elimer와 Pepe가 된다. 이러한 매커니즘은 기존 RBAC 방식에서 사용되고 있는 롤에 사용자를 할당하고 퍼미션에 롤을 할당하는 사용자-롤-퍼미션 매핑 방식이다.

2. 관점 지향 프로그래밍

소프트웨어 시스템은 다양한 역할을 수행하는 모듈들의 연계를 통해 통합적인 기능을 제공한다. 그러한 모듈들 중에는 주요 업무와 관련 되어 있는 것뿐 아니라 로그를 남기거나 보안을 검사하는 것과 같은 부가적인 업무를 위해 구현된 모듈이 있다. 일반적으로 주요 업무 모듈에서 부가적인 함수를 호출하는 구조를 가지고 있으며 결국 주요 업무와 관련 없는 코드가 포함되어 생산성 및 품질이 떨어지고 유지보수 비용 많이 들게 된다. 관점 지향 프로그래밍이 이러한 문제점을 보완하기 위하여 제안된 새로운 프로그래밍 패러다임이다. 기본 개념은 주요 모듈과 부가적인 모듈을 각각 개발하고 나중에 두 개의 코드를 조합(weaving)하여 완벽한 프로그램을 작성하는 것이다. 프로그래밍 언어마다 이러한 조합을 지원하는 툴들이 제공되며 자바 프로그래밍 언어에서는 AspectJ[9] 라는 대표적인 툴이 있다. 자바 프로그램 안에서 어떻게 조합할지 결정하는데 몇 가지 룰이 사용된다. [그림 2]는 각 룰에 의미를 보여준다. 첫째는 애플리케이션의 행동의 변화를 줄 조건과 지점을 정의하는 하는 포인트 컷(point-cut)이며 둘째는 어떤 변경을 시킬지 정의하는 어드바이스(advice)가 있다. 프로그램 수행 중 포인트 컷에서 지정한 조건에 부합되는 경우 어드바이스로 실행 컨텍스트가 넘겨 어드바이스 코드를 수행 한 후 다시 원래 실행 컨텍스트를 진행하거나 중단 한다.

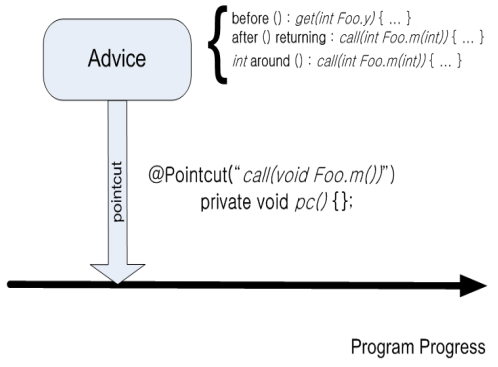

그림 2. 관점 지향프로그래밍에서 조합을 위한 기본 룰

II. 설계 및 구현

본 장에서는 확장된 OSGi 접근제어 방법을 제안하고 어떻게 관점 지향 프로그래밍을 이용하여 구현하였는지 기술한다.

1. 롤에 할당된 사용자 검사

본 연구에서는 롤의 종류를 크게 두 가지로 구분한다. 그 중 하나는 기존 RBAC에서 사용되던 롤의 형태로써 특정 대상이 없이 사용자들이 할당 되어 있는 롤이고 다른 하나는 특정 대상이 있는 롤이다. 우리는 이를 각각 절대적 롤과 상대적 롤이라고 명명하였다. 우리는 절대적 롤 검사만 지원하고 있는 기존 OSGi 권한 부여 매커니즘에 상대적인 롤 검사를 지원하도록 확장한다. 또한 기존 롤-사용자 매핑과 권한-롤의 매핑 형태로 관

리하던 롤 관리를 확장하여 상대적 롤 적용 시 발생할 수 문제점을 보완하고 동적인 권한 검사가 가능한 프로그래밍 모델을 제안한다.

OSGi에서는 현재 사용자가 해당 액션을 수행 할 권한이 있는지 확인할 때 Authorization 클래스의 hasRole() 메소드를 사용하도록 명세화하고 있다. 우리는 절대적 롤 검사만 가능한 구조를 상대적 롤을 사용 가능 하도록 확장하고 동적 권한 검사가 하도록 변경한다. 우선 상대적 롤에 대한 동적 권한 검사를 수행할 수 있도록 대상(target) 인수를 추가한다. 이 대상 인수는 주로 권한 검사 대상이 되는 메소드의 특정 인수가 넘겨 진다. 그리고 Role 클래스 마다 RoleBehaviour 클래스를 등록하여 관리한다. RoleBehaviour는 기존 Authorization 클래스에서 수행하던 권한 검사를 해당 Role에 한해서 대신 검사할 수 있도록 새로 추가한 위임 클래스이다. 우선 관리자가 해당 롤에 설정한 정책이 있다면 이를 검사한 후 Role에 등록된 RoleBehaviour 에게 권한 검사를 위임하여 수행한다. RoleBehaviour는 IRoleBehaviour 인터페이스를 구현한 클래스이며 Role과 RoleBehaviour의 관계는 Map으로 관리한다. 권한 검사를 호출자 관점이 아닌 수행자 관점에서 정의하여 기존 hasRole()메소드는 인수로 액션 그룹이름을 문자열로 받았지만 변경된 메소드는 Role 객체를 받아 처리한다. 각 클래스들간 관계도는 [그림 3]에 나타나 있다.

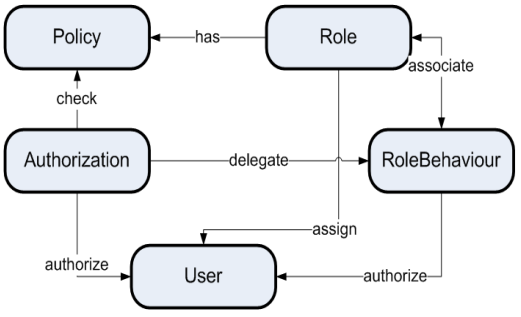

그림 3. 클래스 관계도

RoleBehaviour 위임 모델은 애플리케이션 번들 개발자가 자신만의 롤 검사를 동적으로 구현하여 수행할 수 있게 해준다. 대상 인수를 기본 롤 검사처럼 단순하게 사용자-롤-대상 매핑을 결정하기 위한 인수로도 사용할 수 있지만 보다 다양한 검사를 위한 인수로 사용할 수 있다. 가령 장치마다 소유자가 있고 해당 장치들은 소유자만이 조작할 수 있게 하고 싶다면 DEVICE_OWNER라는 Role을 생성하고 DEVICE_OWNER를 위한 RoleBehaviour를 코드 1과 같은 생성할 수 있다. 즉 모든 장치마다 매핑 테이블에 별도로 추가할 필요 없이 디바이스에 등록된 소유자의 정보만으로 롤 검사를 수행할 수 있다. 또한 장치의 소유자가 변하더라도 정책 변경 없이 롤 검사를 수행 할 수 있다.

코드 1. RoleBehaviour 예

```
boolean isInRole(user, role, target) {
    return user.equals(getDiviceOwner(target));
}
```

2. 롤 관리

적절한 접근 제어를 제공하기 위해서 롤 관리는 중요한 기능 중에 하나이다. 롤 관리는 롤에 사용자 또는 그룹을 할당하는 기능을 수행한다. 이전에 기술한 것과 같이 롤 검사는 Autorization 클래스가 수행하는 기본 검사와 RoleBehaviour에 위임하여 검사하는 유형이 있다. 기본 롤 관리는 사용자-롤-대상 매핑을 저장하며 스키마는 다음과 같다.

UserOrGroup	Role	Type	PermissionType	Target

UserOrGroup은 사용자 또는 그룹의 식별자를 저장하고, Role은 롤에 이름을 저장한다. Type은 사용자인지 그룹인지 식별하기 위한 구분자를 나타내며 PermissionType은 사용자를 DENY 할지 ALLOW 할지 결정하는 검사 타입이다. 마지막으로 Target은 상대적 롤에 사용되는 대상을 나타낸다.

대상이 존재 하지 않은 경우는 Role에 사용자를 할당하기만 하면 롤 관리가 가능하므로 OSGi 시스템 번들

만으로도 처리가 가능하다. 그러나 대상이 있는 경우 대상정보가 롤을 생성한 번들에 의존적이므로 서비스 번들을 개발할 때 롤에 대한 대상 목록을 제공하는 메소드를 구현해야 한다. 우리는 이를 위해 Role 인터페이스에 getTargetList()라는 메소드를 추가하여 정의한다. 예를 들어 DEVICE_OWNER 롤인 경우 각 장치의 ID를 반환하도록 구현하면 된다.

RoleBehaviour에 위임하는 경우 관리자가 롤 관리를 위해 직접 위임된 코드를 변경하는 것은 사실 불가능한 일이다. 그렇다고 롤에 사용자를 추가, 삭제하지 못하는 것은 관리상 문제가 발생 할 수 있다. 제안된 모델에서 완전히 이러한 문제를 해결하지는 못하지만 상호 보완적인 방법을 적용하였다. 우선 관리자에게 RoleBehaviour를 비활성화 할 수 있는 기능을 제공한다. RoleBehaviour를 비활성화하면 기본 롤 관리처럼 위임된 클래스에 의한 동적 롤 검사를 수행할 수 없다. 이러한 이유로 우리는 추가로 RoleBehaviour가 활성화 상태일 때도 롤을 관리하기 위하여 롤 DENY와 ALLOW 검사 타입을 두었다. 이는 ACL(Access Control List)에서 허가 타입을 두어 접근제어를 하는 방법과 유사하다.예를 들어 관리자가 특정 사용자를 DENY 했다면 설령 RoleBehaviour 위임 로직에서 해당 사용자를 허가하였더라도 그 사용자는 접근이 거부된다.

3. 접근제어 구현

번들에서 직접 Authorization.hasRole 메소드를 호출하는 것이 아니라 Reference Monitor[10]의 방식처럼 별도의 레이어가 접근 제어를 수행 한다. 본 논문에서는 관점 지향 프로그래밍 기법을 적용하여 자바 어노테이션과 AspectJ를 사용하여 구현한다. XML 설정파일을 통해 접근 제어[7]를 할 수 있겠지만 본 연구에서는 적용하지 않았다.

접근 제어가 필요한 경우 검사를 수행해야 하는 클래스 또는 메소드 상단에 @RequiredPermission 어노테이션을 정의하기만 하면 된다. @RequiredPermission 어노테이션은 코드 2에 기술되어 있다. 어노테이션 인수로는 Role의 이름을 배열 형식으로 받는다.

코드 2. @RequirePermission 어노테이션

```
@Retention(RetentionPolicy.RUNTIME)
public @interface RequiredPermission {
    String[] value() default {};
}
```

클래스 권한에 할당 할 수 있는 롤은 절대적 롤이며 메소드 권한에는 절대적, 상대적 롤 모두 할당 할 수 있다. [표 1]은 롤 종류와 사용 예를 보여주고 있다.

표 1. 롤 종류와 사용 예

대상	롤 종류	예제
클래스	절대적 롤	@RequiredPermission({role}) public class A {...}
메소드	절대적 롤	@RequiredPermission({role}) a() {}
메소드	상대적 롤	@RequiredPermission({role}) a (@RoleTargetParam String id) {}

@RoleTargetParam 어노테이션은 상대 롤 검사를 위해 사용되는 어노테이션이다. 인수로는 targetField를 받는다. targetField는 대상으로 지정한 인수의 내부 필드값을 권한 검사에 이용하고자 할 때 사용된다. @RoleTargetParam 어노테이션은 코드3에 기술되어 있다.

코드 3. @RoleTargetParam 어노테이션

```
@Target(ElementType.PARAMETER)
@Retention(RetentionPolicy.RUNTIME)
public @interface RoleTargetParam {
   String targetField() default "";
}
```

AuthorizationManager는 @RequiredPermission 어노테이션이 정의된 클래스와 메소드에 대하여 수행 흐름을 가로채 권한 검사를 수행한다. 우선 해당 @RequiredPermission 어노테이션을 포인트컷으로 지정하고 롤 검사 검사를 위해 Authorization.hasRole 메소드를 호출하는 어드바이스를 지정한다.

클래스와 메소드 권한 검사를 위한 포인트컷과 어드바이스는 코드4과 코드5에 기술되어 있다.

코드 4. 클래스 권한검사를 위한 포인트컷과 어드바이스

```
@Around("@target(RequiredPermission)")
 public Object doAccessCheck
                        (ProceedingJoinPoint jp)
                        throws Throwable {
  RequiredPermission requiredPermission =
            jp.getTarget().getClass().
            getAnnotation(RequiredPermission.class);
      return AuthorizationUtil.
            accessCheckAndProceed(jp, requiredPermission);
 }
```

코드 5. 메소드 권한검사를 위한 포인트컷과 어드바이스

```
@Around(value="@annotation(requiredPermission)",argNames
="requiredPermission")
public Object doMethodAccessCheck
                        (ProceedingJoinPoint jp,
  RequiredPermission requiredPermission) throws Throwable
{
      return AuthorizationUtil
            .accessCheckAndProceed(jp, requiredPermission);
 }
```

accessCheckAndProceed 메소드 알고리즘.

1) @RequiredPermission 어노테이션으로 부터 Role 집합을 가져온다.
2) 메소드 인수에 @RoleTargetParam 어노테이션이 정의되어 있는지 확인하고 있다면 이를 통해 대상 객체 정보를 가져온다.
3) Authorization.hasRole 메소드를 호출하여 해당 사용자가 Role에 속하는지 확인한다.
4) 3)을 Role 집합의 원소 개수만큼 반복한다.
5) 해당 Role이 하나라도 있다면 계속 진행한다. 그렇지 않으면 예외상황을 발생시킨다.

VI. 평가 및 고찰

셋톱 박스를 위한 서비스를 개발하고 접근제어를 어떻게 적용하는지 간단한 시나리오로 설명한다.

1) 번들 개발자는 TV 를 OSGi를 통해 조작하는 서비스를 개발한다. 그 중 요청을 받으면 셋톱박스에 켜고 채널 변경 명령을 수행하는 간단한 서비스를 개발 한다.
2) 서비스 개발 후 접근 제어를 위한 모듈을 개발 한다. LIMITED_VIEW_ROLE 이라는 Role과 RoleBehaviour를 추가한다. RoleBehaviour는 현재 시간과 사용자의 나이를 고려하여 접근제어를 수행하도록 구현하여 채널을 변경할 때 나이에 따라 선택할 수 있는 채널과 시청 시간대를 제한 한다. 제한 시간과 채널을 별도의 관리 프로그램으로 관리할 수 있다.
3) 배포된 번들은 운영자에 의해 OSGi 에 배치되고 Role 저장소에 Role과 RoleBehaviour를 등록한다.
4) 운영자는 Role 관리기를 이용하여 등록된 Role과 RoleBehaviour를 확인하고 등록된 RoleBehaviour를 활성화 시킬지 결정한다.
5) 사용자가 로그인 후 채널 변경을 요청하면 권한 체크 모듈이 이를 가로채어 Role에 활성화된 RoleBehaviour가 접근제어를 수행한다.
6) 사용자가 해당 롤에 속해있다면 요청한 서비스를 진행시키고 아니라면 오류를 발생시킨다.

기존 접근제어 방식으로 처리하려면 채널 그룹별 액션 그룹을 생성하여 각 사용자를 액션 그룹에 추가해 주어야 한다. 그러나 제안된 모델에서는 위임된 RoleBehaviour가 사용자의 나이 정보로 권한을 검사하도록 하여 별도의 추가 없이 수행 가능하다. 또한 홈 네트워크 환경에서 사용자가 추가 또는 삭제되는 것과 같은 동적 상황에서도 기존 방식처럼 액션 그룹에 사용자를 추가하고 삭제하는 하는 등의 별도의 관리가 필요 없게 된다.

제안한 접근 제어 방법은 기존 OSGi 접근제어 방법과 비교하여 다음과 같은 특징을 가지고 있다. 첫째로 절대적 롤뿐 만 아니라 상대적 롤에 대한 접근제어를 지원한다. 이를 통해 단순하게 특정 액션에 대한 접근제어를 하는 것 이 아니라 특정 액션에 특정 대상까지 고려한 접근 제어가 가능하다. 둘째로 동적인 접근제어 매커니즘을 구성할 수 있다. 롤에 할당된 사용자에 대한 검사를 무조건 롤 매핑 방식으로 시스템 번들이 수행하는 것이 아니라 서비스 번들에게 상황에 따라 위임

하여 적응성이 뛰어난 접근제어가 가능하다.

OSGi에 제안한 매커니즘을 적용할 때 몇 가지 고려할 것이 있다. 우선 개발의 편의를 위해서 어노테이션과 AspectJ 기능을 사용했지만 현재 OSGi는 Java 1.3 호환 버전으로 어노테이션을 지원하지 않는다. 하지만 OSGi 명세서에서도 나와 있듯이 결국 Java 1.3의 호환성유지보다는 더 높은 버전을 지원하는 것으로 변화할 것으로 보인다. AspectJ 경우 컴파일타임에 적절한 조합을 위한 작업이 이루어진다. 즉, 서비스 번들을 만들 때부터 이를 고려한 개발환경이 구축되어야 적용이 가능하다. 그러나 아직 OSGi에 AOP를 적용하는 연구는 진행 중에 있으며 본 연구에서는 오픈소스 OSGi 프레임워크인 Knopflerfish[11]를 AspectJ 컴파일러로 다시 컴파일하여 적용하였다.

V. 결론

홈네트워크에서는 엔터프라이즈 서버환경과 같이 필요에 따라 기능을 구현하고 때론 커스터마이징하면서 운영하기 어렵다. 초창기에는 통합된 패키지에 의해서 모든 기능을 관리할 수 있겠지만 다양한 서비스와 장치가 나옴에 따라 현재 OSGi에서 제공하는 단순한 통합 RBAC 방식으로는 적절한 접근 제어를 수행하기 어렵다. 제안된 모델은 기존 OSGi 접근제어 방법에 상대적 롤 개념을 추가하고 롤에 속한 사용자 검사를 위임 클래스가 수행할 수 있게 함으로써 다양하고 적응성 뛰어난 접근 제어 수행할 수 있게 하였다. 이러한 특징은 기존 정보가 변하더라도 위임 클래스의 구현에 따라 롤 매핑을 수정하지 않고 효과적인 롤 검사를 할 수 있었다.

참 고 문 헌

[1] W. Mark, "The computer for the twenty-first century," Scientific American, Vol.265, No.3, pp.94-104, 1991.

[2] OSGi Alliance, About the OSGi Service Platform, Technical Whitepaper Revision 4.1, 2005.

[3] S. Oaks, *Java Security*, 2/e O'Reilly, 2001.

[4] D. F. Ferraiolo and D. R. Kuhn, "Role Based Access Control," (PDF) 15th National Computer Security Conference: pp.554-563, 1992.

[5] 이준호, 임경식, 원유재, "XACML 기반 홈 네트워크 접근제어 시스템의 설계 및 구현", 정보처리학회논문지, 제13-C권, 제5호.

[6] 조은애, 문창주, 백두권, "OSGi 서비스 플랫폼에서 RBAC기반의 사용자 접근제어 프레임워크", 정보과학회논문지, 제34권, 제5호, 2007.

[7] OASIS, XACML 1.0 Specification, http://www.oasis-open.org/committees/download.php/2046/oasis-xacml-1.0.pdf, 2003.

[8] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-Oriented Programming.In ECOOP, pp.220-242, 1997.

[9] The AspectJ Project. http://www.eclipse.org/aspectj/.

[10] J. P. Anderson, Computer Security technology planning study, Technical report, Deputy for Command and Management System, USA, 1972.

[11] Knopflerfish OSGi - Open Source OSGi service platform, http://knopflerfish.org/

저 자 소 개

김 인 태(In-Tae Kim) 정회원

- 1997년 2월 : 인하대학교 전자계산공학과(공학사)
- 1999년 2월 : 인하대학교 전자계산공학과(공학석사)
- 2005년 3월 ~ 현재 : 인하대학교 컴퓨터 정보공학과 박사과정

<관심분야> : 유비쿼터스 컴퓨팅, 상황인식

정 경 용(Kyung-Yong Chung) 정회원

- 2000년 2월 : 인하대학교 전자계산공학과(공학사)
- 2002년 2월 : 인하대학교 컴퓨터 정보공학과(공학석사)
- 2005년 8월 : 인하대학교 컴퓨터 정보공학과(공학박사)
- 2005년 9월 ~ 2006년 2월 : 한세대학교 IT학부 교수
- 2006년 3월 ~ 현재 : 상지대학교 컴퓨터정보공학부 교수

<관심분야> : 유비쿼터스 컴퓨팅, 인공지능시스템, 데이터마이닝, U-CRM

임 기 욱(Kee-Wook Rim) 정회원

- 1977년 2월 : 인하대학교 전자공학과(공학사)
- 1987년 2월 : 한양대학교 전자계산학(공학석사)
- 1994년 8월 : 인하대학교 전자계산학(공학박사)
- 1977년 ~ 1988년 : 한국전자통신연구소 시스템소프트웨어 연구실장
- 1989년 10월 ~ 1996년 12월 : 한국전자통신연구원 시스템연구부장, 주전산기(타이컴)Ⅲ,Ⅳ 개발사업 책임자
- 2001년 7월 ~ 1999년 12월 : 한국전자통신연구원 컴퓨터소프트웨어 연구소장
- 2000년 ~ 현재 : 선문대학교 컴퓨터정보학부 교수

<관심분야> : 실시간데이터베이스시스템, 운영체제, 시스템구조

이 정 현(Jung-Hyun Lee) 정회원

- 1977년 2월 : 인하대학교 전자과(공학사)
- 1980년 9월 : 인하대학교 전자공학과(공학석사)
- 1988년 2월 : 인하대학교 전자공학과(공학박사)
- 1979년 ~ 1981년 : 한국전자기술연구소 연구원
- 1984년 ~ 1989년 : 경기대학교 전자계산학과 교수
- 1989년 1월 ~ 현재 : 인하대학교 컴퓨터공학부 교수

<관심분야> : 자연어처리, HCI, 음성인식, 정보검색, 고성능 컴퓨터구조