Aglets 플랫폼에서 RBAC을 이용한 접근 제어 기법*

한강학, 최정환, 장현수, 엄영익
성균관대학교 정보통신공학부
e-mail: kanghack2@hanmail.net, {themars, jhs4071, yieom}@ece.skku.ac.kr

An Access Control Scheme using RBAC in Aglets Platform

Kang-Hack Han, Jung Hwan Choi, Hyunsu Jang, Young Ik Eom
School of Information and Communications Engineering, Sungkyunkwan Univ.

요 약

Aglets은 IBM 도쿄 연구소에서 자바 언어로 구현된 이동 에이전트 프레임워크이다. Aglets은 악의적인 에이전트로부터 에이전트를 보호하기 위해 접근 제어 기법을 사용한다. 기존의 Aglets의 접근 제어 방식은 각각의 에이전트로부터 수신되는 메시지의 종류에 따라 접근 허용 여부를 결정한다. 그러나, 시스템이 거대화되면 접근 제어의 규모 역시 커지게 되어 시스템에 부담을 주게 된다. 본 논문에서는 기존 Aglets 접근 제어방식에 역할기반 접근 제어 방식을 적용한다. 역할관리자와 역할저장소를 배치하여 기존의 접근 제어보다 상대적으로 변화를 줄여 접근허가 및 할당에 있어 복잡성과 잠재적인 실수 및 비용을 줄인다.

1. 서론

Aglets은 IBM 도쿄 연구소에서 개발된 자바 언어로 구현된 이동 에이전트 프레임워크이다. Aglets은 외부의 악의적인 공격으로부터 시스템을 지키기 위해 여러 보안기법을 사용한다. 그중 악의적인 에이전트로부터 서버와 에이전트를 보호하기 위해 퍼미션(permission)과 프로텍션(protection)이라 명명된 접근 제어 기법을 사용한다 [1,2].

기존 Aglets의 접근 제어 방식에서는 각각의 에이전트로부터 오는 메시지를 종류에 따라 접근을 허용할 것인지 여부를 결정한다 [3]. 이 방식에서는 시스템이 거대화될 경우 접근 제어의 규모 역시 커지게 된다. 따라서 갱신하는데 있어 그 복잡성과 잠재적인 손실도 증대하고, 그에 따른 비용 역시 증가한다.

본 논문에서는 기존 Aglets 접근 제어방식인 프로텍션에 역할 관리자와 역할 저장소를 배치하여 역할기반 접근 제어 방식을 적용하는 기법을 제안한다. 역할기반 접근 제어 방식에서 각 개체는 역할을 부여받고, 각 역할들은 접근허가를 부여받는다. 개체의 역할은 기존의 접근 제어에 비해 상대적으로 변화가 적 접근허가 할당에 있어 복잡성과 잠재적인 실수, 비용 등이 줄어드는 장점이 있다 [3,4].

본 논문은 다음과 같이 구성된다. 2장에서는 Aglets 시스템의 접근 제어 기법과 RBAC 표준모델에 대해 알아보고, 3장에서는 본 논문에서 제안하는 RBAC(Role Based Access Control)기반 Aglets 접근 제어 모델 및 그 알고리즘을 설명한다. 4장에서는 제안한 접근 제어 모델을 시뮬레이션을 통해 검증하고, 마지막 5장에서는 결론과 함께 향후 과제를 제시한다.

2. 관련연구

2.1. Aglets 접근 제어

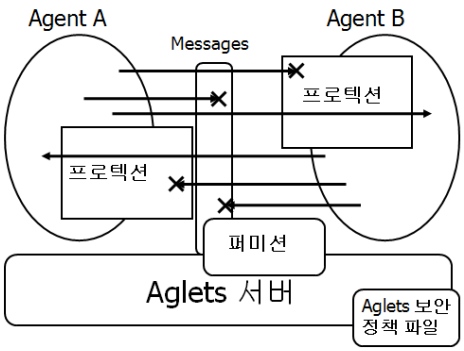
Aglets 안에서 에이전트들은 메시지를 주고받는다. 메시지의 종류로는 에이전트 시스템을 위한 시스템 메시지와 사용자가 지정한 메시지인 사용자 정의 메시지가 있다. Aglets 시스템에서는 메시지들을 통해 에이전트들이 작업을 수행하게 된다. 그 예로 시스템 메시지로는 에이전트의 이동을 명령하는 dispatch, 복제를 명령하는 clone 등이 있다. Aglets에서는 악의적인 목적으로 메시지가 보내지는 것을 막기 위해 접근 제어 방식을 사용한다. Aglets 접근 통제로는 퍼미션과 프로텍션이 있다.

퍼미션은 에이전트 서버가 에이전트로부터 오는 메시지를 제한하는 접근 제어를 말한다. 예를 들어 에이전트 A가 서버 C에 있는 에이전트 B에 메시지를 보낸 경우 서버 C로 진입해야한다. 이 때 서버는 저장되어있는 접근 제어 목록을 참조하여 메시지의 접근을 허가할 것인지를 결정한다. 접근 제어 목록은 서버 내부에 있는 Aglets 보안 정책 파일에 저장되어있다 [2,5].

프로텍션은 에이전트가 다른 에이전트로부터의 접근을

* 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (IITA-2008-(C1090-0801-0046))

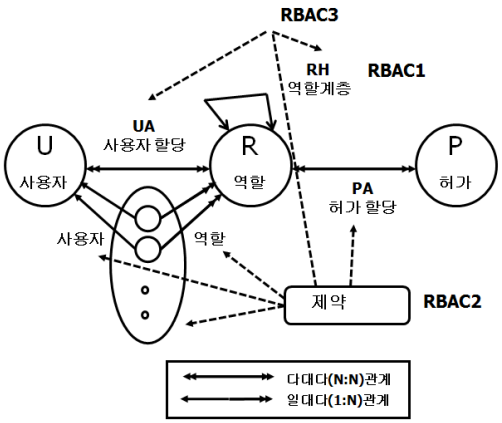
제한하는 방식이다. 에이전트 생성 시 Aglets에서 제공된 메소드를 통해 접근 제어가 이루어진다. 프로텍션에서 Aglets 사용자가 에이전트 소유자에 따라 에이전트에 대한 접근 제어를 지정해야한다. 대형 Aglets 시스템에서는 시스템 메시지 뿐 만 아니라 사용자 정의 메시지의 양이 많아지므로 효율적인 관리가 필요하다. 그림 1은 Aglets에서의 메시지 접근 제어를 나타낸다.



(그림 1) Aglets에서의 메시지 접근 제어

2.2. RBAC 모델

역할기반 접근 제어는 개체의 역할에 기반을 둔 접근 제어 방법으로 Ravi. S. Sandhu의 RBAC 모델이 대표적인 모델이다. 그림 2는 RBAC 모델을 나타낸다 [4].



(그림 2) RBAC 모델

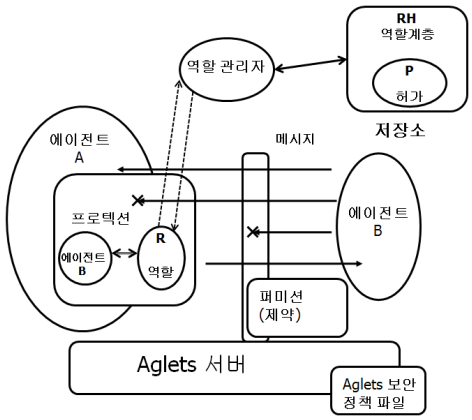
RBAC 모델은 RBAC0, RBAC1, RBAC2, 그리고 RBAC3모델로 구분된다. RBAC0는 가장 기본적인 모델로 사용자에 역할을 배정하고 그 역할에 해당하는 허가가 할당된다. RBAC1은 다른 역할로부터 허가를 상속받을 수 있는 역할 계층의 특성을 추가하였고, RBAC2는 RBAC이 설정된 부분에 특정 제약을 부가할 수 있도록 하였다. RBAC3는 RBAC1과 RBAC2를 통합한 형태이다 [3,4].

3. RBAC를 적용한 Aglets 접근 제어

본 장에서는 역할기반 접근 제어중 RBAC을 적용한 Aglets 접근 제어에 대해 설명한다. 적용하는 RBAC 모델은 Ravi S. Sandhu가 제안한 RBAC 모델 중 하나인 RBAC3으로 한다.

3.1. 접근 제어 구조도

본 절에서는 본 논문에서 제안한 RBAC를 적용한 Aglet 접근 제어의 전체 구조도에 대해서 설명한다. 그림 3은 Aglet 접근 제어에 RBAC 개념을 적용한 구조도이다.



(그림 3) RBAC를 적용한 접근 제어 구조도

프로텍션은 기존 Aglets 접근 제어에서는 에이전트에 대한 메시지 허가를 지정하는 기능을 수행했지만, RBAC을 적용한 본 기법에서는 대상이 되는 에이전트에 역할을 부여하는 기능을 수행한다.

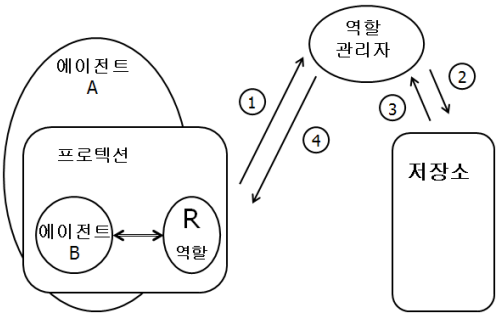
퍼미션은 서버 자체로의 메시지 접근 제어를 수행한다. 또한 역할기반 접근 제어의 적용 측면에서 볼 때 퍼미션은 RBAC3에서의 제약기능을 수행한다. 퍼미션의 기록은 Aglets 개발툴에서 제공되는 Aglet 보안 정책 파일에서 이루어진다.

3.2. 접근 제어 알고리즘

역할관리자는 제안기법을 위해 Aglets 접근 제어에서 추가된 객체이다. 역할관리자는 에이전트가 상대 에이전트에 역할을 부여할 때 그 역할에 해당하는 허가를 부여한다.

역할저장소는 각 역할에 해당하는 허가가 저장된 장소로 역할테이블(role table)과 허가테이블(permission table)로 구성된다. RBAC에서의 역할의 역할계층도 역할 저장소에서 결정된다.

그림 4는 RBAC 적용 Aglets 접근 제어의 동작 순서를 보인다.

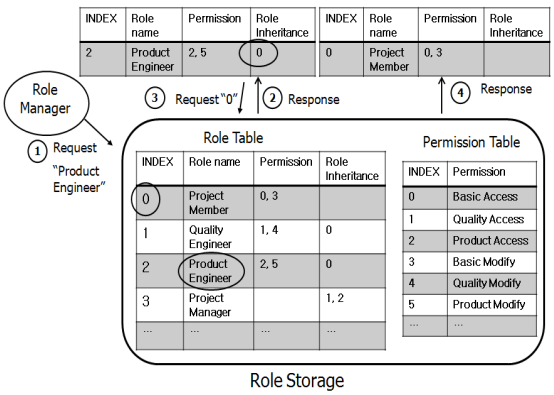


(그림 4) 역할에 대한 허가 부여

에이전트의 프로텍션에서 대상 에이전트에 대해 역할 부여하고, 해당 역할에 대한 허가를 역할관리자에게 요청한다. 역할관리자는 역할저장소에게 역할 테이블에서 해당 역할의 정보를 받는다. 그 후, 그 역할의 허가 번호와 상속한 역할들의 허가 번호를 얻고 허가 테이블을 참조하여 허가 번호에 해당하는 허가로 바꾼다. 그로부터 얻은 허가를 Aglet의 프로텍션에 반환하고, 에이전트는 프로텍션을 활성화함으로써 대상 에이전트에 대한 접근 제어가 완료된다.

이해를 돕기 위해 에이전트 A가 에이전트 B에 제품기술자(Product Engineer)라는 역할을 부여하고 그에 대한 허가를 받는 상황을 예로 들어 알고리즘을 살펴보면 다음과 같다..

에이전트 A는 프로텍션을 통해 에이전트 B의 역할을 제품기술자로 설정하고 역할관리자에게 그에 맞는 허가를 가져올 것을 요청한다. 역할관리자는 역할저장소에게 역할 테이블에서 역할명칭(role name)이 생산기술자인 튜플의 데이터를 가져온다. 그림 5는 역할테이블에서의 튜플 획득을 보인다.

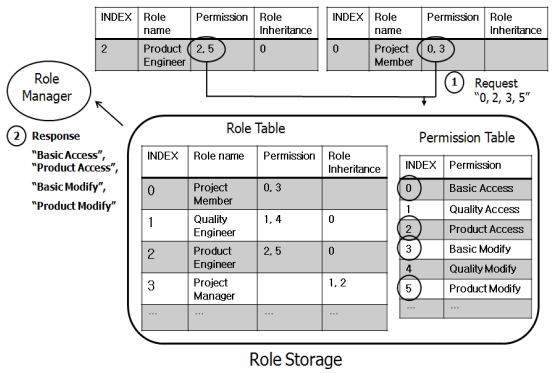


(그림 5) 역할테이블에서의 튜플 획득

역할관리자는 역할저장소에게 역할테이블에서 역할명칭이 제품기술자인 튜플의 데이터를 가져온다. 가져온 튜플에서 역할상속 필드를 확인한다. 이 튜플의 역할 상속 필드에는 0이 기록되어있고 이는 역할 테이블에서 인덱스 필드가 0인 튜플의 역할을 상속한다는 것을 의미하므로 역할 관리자는 역할 테이블에서 인덱스가 0인 튜플의 데이터를 가져온다. 더 이상 상속하는 역할이 없으므로 각 튜플의 허가 필드에 기록된 번호를 확인하고 허가 테이블에서 그 번호와 일치한 인덱스를 가진 튜플의 허가 필드에 기록된 정보를 가져온다.

제품기술자와 프로젝트 멤버의 허가필드에는 0과 3, 2와 5가 기록되어 있다. 그러므로 허가 테이블에서 이 번호들과 일치한 인덱스를 가진 튜플의 허가는 Basic Access, Product Access, Basic Modify, Product Modify이다.

그림 6은 역할에 대한 허가 획득 과정을 보인다.



(그림 6) 역할에 대한 허가 획득

결과로 얻은 허가를 에이전트 A의 프로텍션에 전송한다. 에이전트 A는 에이전트 B에 대한 허가를 부여하고 같은 방법으로 다른 에이전트에 대한 허가도 부여한 후, 프로텍션을 활성화한다.

4. 시뮬레이션

본 논문에서 제안하는 모델에 대한 검증을 위해 시뮬레이션을 수행하였다. 시뮬레이션은 Windows XP 환경에서 CPU 1.7 GHz와 RAM 1024MB을 탑재한 IBM PC에서 수행되었다.

본 기법의 역할저장소는 XML을 이용하여 구현하였다. 표 1은 XML로 구현된 역할테이블과 허가테이블을 나타낸다. 역할테이블에는 인덱스, 역할 이름, 권한 값, 상속 여부 등이 포함되며, 허가테이블에는 인덱스 및 그에 해당하는 허가 이름이 포함된다.

역할관리자는 Role_manager 클래스를 만들고, 클래스 내부의 메소드인 Role_Allocation() 메소드를 통해 에이전트에 지정된 역할에 부여된 메시지 접근 허가를 수행하도록 구현했다.

<표 1> 역할테이블과 허가테이블의 구현

```
<Role-Table>
  <Role>
    <INDEX value="2" />
    <Role-name value="Product Engineer" />
    <Permission value="2,5" />
    <Role-inheritance value="0" />
  </Role>
</Role-Table>
```

```
<Permission-Table>
  <Permission>
    <INDEX value="0"/>
    <Permission-name value="basicAccess"/>
  </Permission>
</Permission-Table>
```

표 2는 에이전트가 생성될 때 자동으로 호출되는 OnCreation() 메소드에서 Role_manager 클래스를 통해 접근 제어를 수행하는 소스 코드이다.

<표 2> 역할 관리자를 이용한 접근 제어

```
public void onCreation(Object init) {
  Protections protections; // 프로텍션

  /* "paul"을 소유자로 하는 에이전트에
   *  "Product Engineer" 역할을 부여하여
   *  그에 맞는 사용자 정의 메시지를
   *  프로텍션에 등록 */
  protections =
  Role_manager.Role_Allocation("paul",
                               "Product Engineer");

  /* 시스템 메시지에 대한 프로텍션 */
  protections.add(new AgletProtection("paul",
  "dispose,clone,dispatch,deactivate,retract"));

  /* 프로텍션 활성화 */
  setProtections(protections);
}
```

구현된 모델이 실제로 Aglets 시스템 상에서 접근 제어를 수행하는 지를 검증하기 위해, 저장소의 데이터를 그림 5와 그림 6에서 보인 데이터로 구성하였으며, 그림 7는 시뮬레이션 후 정상적으로 프로텍션이 활성화되었다고 출력된 결과 메시지를 나타낸다.

```
--ProtectionAgent[id(01490de735249369)]: 생성
Sender에 대한 Role 지정(Product Engineer)
인덱스 : 2       이름 : Product Engineer  허가번호 : 2,5   역할상속 : 0
Product Engineer의 접근 허가 메시지의 인덱스 :  2 5 0 3
SenderAgent(Product Engineer)의 접근 허가 :
        productAccess  productModify  basicAccess  basicModify
=================================================================
--SenderAgent[id(01d2f5a616aaa7f4)]: basicAccess
--ProtectionAgent[id(01490de735249369)]: received 'basicAccess'
```

(그림 7) 시뮬레이션 출력 결과

ProtectionAgent는 생성되면서 SenderAgent에 대해 “Product Engineer”을 부여하여 프로텍션을 활성화한 후, SenderAgent는 ProtectioAgent에게 “basicAccess” 메시지를 보낸다. "Product Engineer"에게 "basicAccess" 메시지 접근허가가 있으므로 ProtectionAgent는 메시지를 제대로 받았다는 메시지를 콘솔창에 출력한다. 만약 접근허가가 나지 않은 메시지가 SenderAgent로부터 전송된다면 프로텍션에 의해 전송이 차단되며 ProtectionAgent는 메시지를 전송 받지 못한다.

5. 결론

Aglets에서는 악의적인 에이전트로부터의 에이전트 보호가 접근 제어를 통해 이루어졌다. 그러나, 이러한 방식은 구조적이지 못해, 시스템이 커짐에 따라 메시지 수가 급속히 증가하여 에이전트의 수에 대한 관리가 효율적이지 못해 잠재적인 실수와 비용의 증가를 유발한다. 이러한 약점을 극복하기위해 본 논문에서는 기존 Aglets 접근 제어 방식에 역할기반 접근 제어를 적용할 것을 제안하였다. 보다 효율적인 접근 제어를 위해 Ravi S. Sandhu가 제안한 RBAC3 모델을 사용하고 역할관리자와 역할저장소를 두어 프로텍션이 역할에 기반한 접근허가 기능을 수행할 수 있도록 하였다. 또한, RBAC3 모델의 특징인 역할계층을 역할저장소의 역할테이블에는 역할상속 필드를 통해 구성하였고, 역할 이외의 제약은 서버의 접근 제어인 퍼미션이 수행하도록 하였다.

향후 NIST RBAC 표준 모델 및 강화모델의 적용, Aglets의 다른 보안기법의 개선방안 등이 연구되어야 할 것이다. Ravi S. Sandhu가 제안한 RBAC 모델 이외에 NIST가 제안한 RBAC 표준 모델의 경우 한 사용자가 두 개의 상충하는 역할을 중복하여 할당 받을 수 없게 하는 의무분리의 개념 등이 추가되어있어 Aglets 시스템에 적용하게 된다면 복잡성을 크게 줄일 수 있을 것으로 예상된다. 또한 Aglets 보안에는 접근 제어 외에 서버 인증이 있으므로 이에 대한 개선방안 역시 연구되어야할 것이다.

참고문헌

[1] Danny B. Lange, Mitsuru Oshima, “Programming and Deploying Java Agents with Aglets,” Addison Wesley, pp. 225, 1998.

[2] G. Karjoth, D. Lange, and M. Oshima, “A Security Model for Aglets,” IEEE Trans. on Internet Computing, Vol. 4, pp. 68-77, 1997.

[3] 김학범, 김동규, “RBAC 표준 참조 모델 연구동향”, 정보통신정보보호학회지, Vol. 10, No. 2, pp. 51-60, 2000.

[4] R.S. Sandhu, E.J. Coyne, H.L. Feinstein and C.E. Youman, “Role-Based Access Control Models,” IEEE Computer, Vol. 29, No. 2, pp. 38-47, 1996.

[5] K. Ono, H. Tai, “A Security Scheme for Aglets,” Software: Practice and Experience, John Wiley & Sons Ltd. Vol. 32, No. 6, pp. 497-514, 2002.