OSGi 서비스 프레임워크 환경에서의 서비스 보안 모델

박대하[0*], 백두권**
*㈜시큐리티 테크놀로지스 (dhpark@stitec.com)
**고려대학교 컴퓨터학과 (baik@swsys2.korea.ac.kr)

A Service Security Model for the OSGi Service Framework Environment

Dae-Ha Park[0*], Doo-Kwon Baik**
*STI, SECURITY Technologies Inc.
**Dept. of Computer Science and Engineering, Korea University

요 약

본 논문에서는 OSGi 서비스 프레임워크 환경에서 서비스의 접근 통제 기법을 구현할 수 있는 정형화된 보안 모델을 제시하고 있다. OSSEM(*OSGi Service Security Extension Model*)이라고 하는 이 보안 모델은 위임, 허가, 거부의 3 가지 권한 엔트리를 정의하여 동적으로 설치되는 번들에서 독립적인 보안 정책의 설정을 지원하며, 다른 번들의 정책과 결합될 때 안전한 서비스의 수행을 보장한다. 이 모델은 유연성있는 보안 정책의 설정의 가능하고 OSGi 서비스 프레임워크 환경에서 기존의 Java 2 보안 아키텍처와 쉽게 통합이 가능하므로 번들과 서비스에 대한 접근 통제 뿐만 아니라 OSG 내부 자원의 접근 통제 방안으로도 확장할 수 있다.

1. 서 론

OSGi(*Open Services Gateway Initiative*)는 다양한 서비스를 WAN을 거쳐 LAN과 디바이스로 전달하기 위한 개방형 스펙을 개발하고, 이 스펙을 기반으로 한 상품과 서비스를 정보가전과 같은 임베디드(embedded) 시장에 확산시키기 위한 목적으로 조직된 그룹이다. 1999년에 설립되어 현재 전세계적으로 80 개 이상의 회사가 참여하고 있다.

OSGi 서비스 게이트웨이 스펙(릴리즈 1.0)[1]은 개방형 서비스 게이트웨이(OSG)에 준거하는 서비스를 개발하는데 필요한 핵심적인 API와 선택적인 API를 Java 언어를 사용하여 정의하고 있다. 이 스펙에는 서비스의 생명주기 관리, 서비스 간의 의존성, 데이터 관리, 디바이스 관리, 클라이언트 접근, 자원 관리, 보안 등에 관한 API가 포함되어 있다.

OSGi 서비스 프레임워크(service framework)는 소형 메모리 자원을 가진 디바이스에 작동하는 어플리케이션을 개발하여 동적으로 배치할 수 있도록 플랫폼에 독립적인 Java 프로그래밍 언어의 동적 코드 로딩 능력을 사용하고 있다. 따라서 개발자는 동일한 서비스 인터페이스를 갖는 다수의 구현을 제공할 수 있으며, 다른 개발자는 구현에 상관없이 서비스 인터페이스를 이용할 수 있다. 또한 번들(bundle)이라는 자체 인스톨이 가능한 컴포넌트 형태로 어플리케이션을 분할할 수 있도록 생명주기(life-cycle) 관리가 가능하여 전체 시스템에 영향을 주지 않고 새로운 서비스를 추가하거나 갱신할 수 있게 해준다.

번들에 포함된 서비스는 생명주기에 따라서 동적으로 서비스 게이트웨이에 배치되어 다른 번들의 서비스와 상호 작용하며, 일부 서비스는 서비스 게이트웨이 자체의 자원을 직접 사용할 수도 있다. 따라서 서비스 게이트웨이의 관리자(또는 오퍼레이터, operator)와 번들의 개발자 및 제공자(또는 서비스 제공자, service provider)는 프레임워크 환경에서 발생할 수 있는 보안상의 문제점을 인식하고 이를 해결하기 위한 방안을 마련해야 한다.

본 논문에서는 OSGi 서비스 프레임워크 환경에서 발생할 수 있는 보안 요구사항 중에서 서로 다른 번들의 서비스 간의 상호 작용에서 발생할 수 있는 접근 통제(access control) 문제를 설명하고 이를 만족시킬 수 있는 구체적인 서비스 보안 모델인 OSSEM을 제시하고자 한다. 특히 번들이나 서비스에 대한 접근 통제를 유연성과 확장성이 보장되는 방식으로 처리할 수 있도록 번들에 포함된 파일 형식으로 명시될 보안 정책(security policy)의 구성요소와 의미론을 정의한다. 또한 제시된 보안 모델을 기존의 Java 보안 아키텍처과 결합하여 프레임워크 상에서 작동하는 보안 서비스를 구현하는 방안을 설명하기로 한다.

2. 관련 연구

프레임워크 자체는 JVM(*Java Virtual Machine*) 상에서 작동하는 어플리케이션으로 간주할 수 있으므로 OSG에 설치되는 서비스에 대한 접근 통제는 호스트의 운영체제 상에 설치된 JVM에서 제공하는 보안 모델을 기초로 이루어져야 한다. Java 언어는 네트워크 상에서 이동하며 실행되는 모바일 코드(mobile code)의 보안성을 염두에 두고 개발되었으며, 특히 Java 2의 보안 아키텍처에서는 보호 도메인(protection domain)의 개념과 스택 조사(stack inspection) 알고리즘을 기반으로 안전한 보안 관리기(security manager)와 확장성 있는 클래스 로더(class loader)를 제공하고 있다[3].

Java 2 보안 아키텍처의 중앙 집중형 보안 정책[4]은 유연성있는 고수준의 보안 관리(예: 주체 그룹화, 권한 위임, 예외 설정 등)가 어렵고, 서로 다른 위치에서 전송된 모바일 코드 간의 상호 작용에 필요한 보안 관계 설정이 불가능한 단점을 가지고 있다. 이를 극복하기 위하여 시스템 권한에 거부, 예외를 결합한 복합형 정책 설정[5], 키 지향 인증서(key-oriented certificate)를 이용한 권한 위임[6], 모바일 에이전트(mobile agent)에 대한 확장된 보안 정책 설정[7] 등의 연구가 기존의 Java 2 보안 아키텍처 상에서 이루어졌다.

본 논문에서 제안하는 보안 모델은 OSGi 서비스 프레임워크 환경에 설치되는 번들이 독자적인 보안 정책을 수립하고 다른 번들의 정책과 결합하여 서비스에 대한 안전한 접근 통제를 이룰 수 있도록 지원하며, 기존의 보안 모델보다 더욱 향상된 유연성과 확장성을 제공할 수 있는 장점을 갖는다.

3. OSGi 서비스 보안 모델

OSGi 서비스 프레임워크 환경에 대한 발생 가능한 보안 문제를 해결하기 위하여 필요한 요구사항으로 다음의 4 가지를 들 수 있다.

- 사용자로부터 OSG 및 서비스의 보호
- 서비스로부터 OSG의 보호
- OSG로부터 서비스의 보호
- 다른 번들의 서비스로부터 서비스의 보호

첫번째와 두번째 요구사항은 모바일 코드에 대한 기존의 보안 모델에서 자주 다루고 있으나(논문 지면의 제약으로 인해 자세한 설명은 생략하며 4.3 절에서 간단히 언급할 예정임), 세번째와 네번째 요구사항은 OSGi 서비스 프레임 환경에서 새롭게 등장한 것이다.

서비스 제공자는 자신이 제공하는 서비스에 접근할 수 있는 권한을 다른 번들의 서비스에 부여하도록 독자적인 보안 정책을 가질 수 있다. OSG에서 작동하는 모든 서비스에 대한 정책을 오퍼레이터가 책임을 지고 설정하는 것은 서비스의 세부적인 구현에 대한 지식을 요구하며 번들 내에서 필요에 따라 오퍼레이터의 인식 밖에 있는 다른 번들을 인스톨하여 사용할 수 있는 유연성을 해치게 되므로 바람직하지 않다.

실제로 OSG는 오퍼레이터가 설치한 보안 서비스 번들의 정책에 의존하고 있으므로 보안 서비스 번들에서 서비스를 공격하는 경우에 막는 것은 불가능하다. 그러나, 오퍼레이터의 부주의로 인한 과다한 권한의 허가로 다른 번들이 공격을 당할 수 있는 위협은 막아야 한다.

본 논문에서는 OSGi 서비스 프레임워크 환경에서 OSG로부터 또는 다른 번들의 서비스로부터 자신의 서비스를 보호하려는 요구사항을 만족시킬 수 있는 보안 모델인 OSSEM을 제시하기로 한다.

3.1 정책 구성요소

OSSEM에서 각각의 번들이 포함하고 있는 서비스를 보호하기 위하여 설정되는 보안 정책의 구성요소는 다음과 같다.

- 주체(subject) – 번들 또는 서비스에 대한 접근을 요청하기 위하여 현재 정책을 포함하고 있는 번들에 의해 설치되는 다른 번들의 서비스 제공자 명칭과 번들의 URL 위치(즉, Java 2 보안 아키텍처에서의 codeSource에 해당함)
- 객체(object) –주체에 대한 접근 통제의 대상이 되는 번들 또는 서비스(URL로 표현되는 명칭공간을 가지며, 현재 도메인에서 가용한 번들이나 서비스만 해당됨)
- 도메인(domain) – 접근 통제에 대한 요청이 발생하는 보안 문맥(번들 A가 번들 B를 설치하면 B의 도메인은 A의 하위 도메인이 됨)
- 액션(action) – 주어진 주체와 객체에 대해서 위임이나 허가 또는 거부될 수 있는 퍼미션(Java에서 Permission을 정의할 때 인자로 사용되는 액션을 포함하는 개념임)

OSSEM은 하나의 번들(또는 번들 내의 서비스)에 유연성과 확장성을 보장하면서 독립적인 권한을 부여할 수 있도록 다음의 3 가지 종류의 엔트리(entry)를 정의한다.

- 위임(delegate) – 위임 엔트리에 포함된 권한은 자신의 도메인 내에서 설치한 하위 도메인에 상속된다.
- 허가(grant) – 도메인 내에 주어진 긍정형(positive) 권한이며, 상위 도메인에서 거부 엔트리가 정의되어 있지 않은 경우에만 허용되고 하위 도메인으로 상속되지 않는다.
- 거부(deny) – 도메인 내에 주어진 부정형(negative) 권한이며, 상위 도메인의 위임 엔트리로부터 상속된 권한을 무효화할 수 있다.

3.2 모델 의미론

OSSEM에 대한 정형화된 의미론(semantics)을 정의하기 위하여 인증 명세 언어인 ASL(*Authorization Specification Language*)[8]의 변형된 버전을 사용한다. ASL은 접근 통제 정책을 기술하고 접근 통제 결정을 지원하는 논리적 언어이며, [5]에서 복합 정책의 설정에 사용되고 있다.

보안 정책은 4-tuple인 (s, o, d, a)의 집합으로 표현되며, 각각은 주체(s), 객체(o), 도메인(d), 액션(a)을 나타낸다. ASL로 표현된 모든 규칙은 우측의 조건이 모두 true이면 좌측의 규칙이 생성됨을 의미한다. 리터럴(literal)인 $\mathbf{in}(d_1, d_2)$는 상위 도메인인 d_2의 보안 문맥 내에서 하위 도메인인 d_1가 설치되었음을 나타낸다.

다음의 **cando** 규칙은 정책 파일에 명시된 엔트리를 표현하는데 사용된다. 액션은 위임('#'), 허가('+'), 거부('-')를 정의할 수 있도록 부호(sign)를 갖는다. 이 규칙에서 우측에 오는 조건은 없다.

$\mathbf{cando}(s, o, d, \langle \# \mid + \mid - \rangle a) \leftarrow .$

다음의 **dercando** 규칙은 OSG 내에 설치된 번들이 자신의 도메인 내에서 확장 엔트리를 생성하는데 사용된다. ¬는 not을 의미하며 명시된 **cando** 규칙이 존재하지 않으면 true가 된다(**dercando** 규칙에는 ¬을 표시할 수 없음).

$\mathbf{dercando}(s, o, d_1, \#a) \leftarrow \mathbf{cando}(s, o, d_1, \#a)\ \&\ \mathbf{dercando}(s, o, d_2, \#a)\ \&\ \mathbf{in}(d_1, d_2).$

$\mathbf{dercando}(s, o, d, +a) \leftarrow \mathbf{cando}(s, o, d, \langle \# \mid + \rangle a).$

$\mathbf{dercando}(s, o, d_1, +a) \leftarrow \neg\mathbf{cando}(s, o, d_1, \langle \# \mid + \rangle a)\ \&\ \mathbf{dercando}(s, o, d_2, \#a)\ \&\ \mathbf{in}(d_1, d_2).$

$\mathbf{dercando}(s, o, d, -a) \leftarrow \mathbf{cando}(s, o, d, -a).$

$\mathbf{dercando}(s, o, d_1, -a) \leftarrow \neg\mathbf{cando}(s, o, d_1, -a)\ \&\ \mathbf{dercando}(s, o, d_2, -a)\ \&\ \mathbf{in}(d_1, d_2).$

다음의 **do** 규칙은 도메인의 확장 엔트리에서 발생할 수 있는 허가 권한과 거부 권한 간의 모순 해결(conflict resolution)을 위하여 거부 우선순위(denials take precedence)[8] 규칙을 정의하고 있다.

$\mathbf{do}(s, o, d, +a) \leftarrow \mathbf{dercando}(s, o, d, +a)\ \&\ \neg\mathbf{dercando}(s, o, d, -a).$

$\mathbf{do}(s, o, d, -a) \leftarrow \mathbf{dercando}(s, o, d, -a)$

마지막으로 **grant** 규칙을 이용하여 주어진 번들이나 서비스에서 도메인으로 요청된 접근 통제를 수행한다. 리터럴인 $\mathbf{implies}(t_1, t_2)$는 요청에 포함된 주체, 객체, 액션과 도메인에 포함된 주체, 객체, 액션 간의 의미적인 내포 관계[3]를 의미한다.

$\mathbf{grant}(s', o', d, +a') \leftarrow \mathbf{do}(s, o, d, +a)\ \&\ \neg\mathbf{do}(s, o, d, +a)\ \&\ \mathbf{implies}(s, s')\ \&\ \mathbf{implies}(o, o')\ \&\ \mathbf{implies}(a, a').$

4. OSGi 보안 서비스의 구현 방안

OSSEM은 OSGi 서비스 프레임워크의 보안 관리 서비스 내에서 Java의 java.lang.SecurityManager 클래스를 확장한 보안 관리기 클래스로 구현한다. 그리고, 프레임워크의 관리에 사용되는 연산(예: 번들의 설치, 시작, 갱신 등 또는 서비스의 등록, 획득 등)이 수행되는 메쏘드 내에서 필요한 퍼미션을 검사(SecurityManager.chekPermission 메쏘드를 사용함)하는 코드를 추가한다.

4.1 OSGi 퍼미션과 관리 연산

OSGi 스펙[1]에서 정의하고 있는 퍼미션은 다음과 같다.

- AdminPermission – 번들에 대한 생명주기와 관련되는 관리적인 기능을 수행할 수 있도록 해주는 퍼미션이다. 현재 OSGi 스펙에서는 이 퍼미션에 타겟이나 액션을 부여하고 있지 않지만 본 논문의 모델에서는 위임에 따른 권한의 축소가 가능하도록 번들의 위치를 타겟(target)으로 두 가지 액션인 install과 start을 지원하도록 확장한다.
- ServicePermission – 서비스의 등록과 접근을 통제할 수 있도록 해주는 퍼미션이다. 서비스에 대한 인터페이스 명칭을 타겟으로 두 가지 액션인 register와 get을 지원한다.
- PackagePermission – 번들이 패키지를 수입(import)하거나 수출(export)할 수 있도록 해주는 퍼미션이다. 패키지에 대한 명칭을 타겟으로 두 가지 액션인 import와 export을 지원한다.

OSGi 서비스 프레임워크에서 접근 통제를 요구하는 관리 연산(및 API)과 필요한 퍼미션(및 액션)은 다음과 같다.

- 번들의 설치(BundleContext.installBundle) – AdminPermission: "install"
- 번들의 분석 – PackagePermission: "import", "export"
- 번들의 시작(Bundle.start) – AdminPermission: "start"
- 번들의 중지(Bundle.stop) – AdminPermission: "start"
- 번들의 갱신(Bundle.update) – AdminPermission: "install"
- 번들의 제거(Bundle.uninstall) – AdminPermission: "install"
- 서비스의 등록(BundleContext.registerService) – ServicePermission: "register"
- 서비스의 획득(BundleContext.getService) – ServicePermission: "get"
- 서비스의 방출(BundleContext.ungetService) – ServicePermission: "get"
- 서비스의 해지(ServiceRegistration.unregister) – ServicePermission: "register"

다음은 번들의 설치를 요청하기 위하여 BundleContext.installBundle 메쏘드를 호출할 때 접근 권한을 검사하는 코드이다.

```
public Bundle installBundle(String location)
throws BundleException {
    try {
      SecurityManager sm = System.getSecurityManager();
      if (sm != null)
        sm.checkPermission(new AdminPermission(location, "install"));
    } catch (SecurityException ex) { /* throws BundleException */ }
    // Bundle installation starts here.
}
```

4.2 OSGi 보안 정책 파일

OSSEM에 따른 정책을 명시하기 위하여 작성되는 보안 정책 파일은 XML 구문법을 사용한다. XML을 사용하는 이유는 높은 이식성과 가독성(readability)을 제공하며, DTD를 사용하여 자동화된 도구에서 간단히 의미를 파악할 수 있기 때문이다.

그림 1과 그림 2는 각각 오퍼레이터와 서비스 제공자가 자신이 소유한 번들 내에 작성한 보안 정책 파일의 예를 보여준다. 태그(tag)인 delegate, grant, deny를 사용하여 위임, 허가, 거부 엔트리를 표현하며, policy 태그의 bundle 속성은 도메인을 나타낸다. 나머지 태그와 속성은 Java 보안 아키텍처의 정책 파일[3]에서 정의하고 있는 엔트리의 표현 방법과 거의 동일하다.

```
<policy bundle="http://www.stitec.com/osgi/ossem.jar">
  <delegate signedBy="Admin" codeBase="http://www.stitec.com/osgi/-">
   <permission class="org.osgi.framework.AdminPermission">
    <target="*"/>
    <action="install start"/>
   </permission>
  </delegate>
  <delegate signedBy="SP1" codeBase="http://www.sp1.com/osgi/-">
   <permission class="org.osgi.framework.AdminPermission">
    <target="http://www.sp2.com/-"/>
    <action="start"/>
   </permission>
  </delegate>
  <grant signedBy="SP2" codeBase="http://www.sp2.com/-">
   <permission class="org.osgi.framework.ServicePermission">
    <target="yyy.light.service.Light"/>
    <action="register"/>
   </permission>
   <permission class="osg.osgi.framework.PackagePermission">
    <target="xxx.device.light"/>
    <action="import"/>
   </permission>
  </grant>
  <deny signedBy="SP3" codeBase="http://www.sp3.com/-">
   <permission class="org.osgi.framework.AdminPermission">
    <target="http://www.stitec.com/osgi/-"/>
    <action="install"/>
   </permission>
  </deny>
</policy>
```

그림 1. 오퍼레이터 정책 파일의 예

```
<policy bundle="http://www.sp1.com/osgi/monitor.jar">
  <grant signedBy="SP3" codeBase="http://www.sp3.com/-">
   <permission class="org.osgi.framework.AdminPermission">
    <target="*"/>
    <action="install"/>
   </permission>
   <permission class="org.osgi.framework.ServicePermission">
    <target="yyy.light.service.Light"/>
    <action="register"/>
   </permission>
  </grant>
  <deny signedBy="SP2" codeBase="http://www.sp2.com/alarm.jar">
   <permission class="org.osgi.framework.ServicePermission">
    <target="yyy.light.service.Light"/>
    <action="get"/>
   </permission>
  </deny>
</policy>
```

그림 2. 서비스 제공자 정책 파일의 예

4.3 보안 모델의 확장

OSSEM은 3 절의 도입부에서 열거한 보안 요구사항 중에서 사용자로부터 서비스와 OSG를 보호하거나 프레임워크에 설치된 번들의 서비스로부터 OSG 내부 자원을 보호할 수 있도록 확장이 가능하다.

사용자로부터 서비스와 OSG를 보호하려면 주체를 표현하는 서비스 제공자 명칭과 번들의 URL 위치 대신에 사용자 명칭만 갖는 엔트리를 정의할 수 있도록 해준다. 서비스 제공자 명칭과 사용자 명칭은 실제로는 해당하는 실체를 증명할 수 있는 인증서(certificate)의 키 저장소 내부에 등록되는 별명(alias)에 해당한다. 번들에 보안 정책을 포함시켜서 다운로드하는 방식은 JAAS[9]의 principal 기반 정책을 이용하는 방식과 유사하므로 사용자에게 전달되는 접속용 프로그램(예: login applet)에 동적으로 해당 사용자의 권한 엔트리를 추가하여 전송할 수 있다. 사용자는 인증서에 대응하는 개인키(private key)를 소유하고 있어야 한다.

서비스로부터 OSG 내부 자원을 보호하려면 OSG의 내부 자원에 대한 접근이 발생하는 순간에 쓰레드에 대한 스택 조사를 통하여 관련된 모든 클래스가 가진 퍼미션의 교집합(intersection) 내에 해당 자원의 접근 퍼미션이 존재하는지 검사해야 한다.

OSG는 오퍼레이터가 제공하는 보안 서비스 번들에 포함된 서비스 게이트웨이 정책을 기반으로 내부 자원에 대한 접근 통제를 수행할 수 있다. 보안 서비스 번들은 OSG의 구현 과정에서 기본 내장형 서비스로 장착되거나 부트스트래핑(bootstrapping) 단계에서 오퍼레이터의 인증[2]을 거쳐서 설치될 수 있다.

5. 결론 및 추후 연구

OSGi 서비스 프레임워크 환경은 여러 서비스 제공자가 독립적으로 개발한 번들의 서비스를 동적으로 연결하여 하나의 어플리케이션 형태로 작동한다. 따라서 OSG에서 작동하는 서비스에 적용되는 보안 모델은 분산된 정책 설정을 지원하면서 런타임에서 동적인 보안 도메인의 설정을 보장해야 한다. 본 논문에서 제안하는 보안 모델인 OSSEM은 번들이나 서비스의 접근 권한을 위임, 승인 또는 거부할 수 있는 엔트리를 제공하여 정책 설정의 유연성과 확장성을 높이는 한편, 구체적인 의미론을 통하여 안전한 접근 통제 결정이 가능하도록 하였다.

현재 OSSEM의 안전성을 충분히 증명할 수 있는 정형 기법을 연구하고 있으며 Java의 스택 조사와 유사하게 동적으로 연결되는 여러 서비스를 거쳐 접근하는 권한 요청을 처리할 수 있도록 도메인 행렬(domain matrix)[4] 기반의 접근 제어기(access controller)를 구현하고 있다.

또한 OSGi 스펙의 퍼미션과 관리 연산 이외에 번들 내에서 독립적으로 정의한 퍼미션과 이를 검사하는 서비스에 대한 권한도 정책에 포함시킬 수 있는 방안을 연구 중이다. 독립적으로 정의된 퍼미션과 관련 서비스를 다른 번들의 권한 설정에 포함시키려면 서비스 제공자 간에 정책을 교환할 수 있는 기반이 마련되어야 한다. 추후에는 특정 OSG에 설치된 번들의 정책을 오퍼레이터의 호스트에 등록하는 보안 정책 등록기(security policy registry)를 구현하고, 새로운 번들의 개발 과정에서 등록된 정책을 기반으로 적절한 정책을 수립할 수 있는 도구를 개발하고자 한다.

[참고 문헌]

[1] OSGi, "OSGi Service Gateway Specification - Release 1.0", http://www.osgi.org, May 2000.

[2] OSGi, "RFC 18 - Security Architecture Specification", Draft, http://www.osgi.org/member, February 2001.

[3] M. Pistoia, et al., "Java 2 Network Security", Second edition, *Prentice Hall*, 1999.

[4] L. Kassab et al., "Towards Formalizing the Java Security Architecture of JDK 1.2", *Proc. of the ERORICS'98*, Leuven-la-Neuve, Belgium, September 1998.

[5] M. Hauswirth et al., "A Secure Execution Framework for Java", *Proc. of the 7th ACM conference on computer and communications security* (CCS 2000), pp. 43-52, Athens, Greece, November 2000.

[6] P. Nikander et al., "Distributed Policy Management for JDK 1.2", *Proc. of the 1999 Network and Distributed Systems Security Symposium*, pp. 91-102, San Diego, CA, February 1999.

[7] G. Karjoth et al., "A Security Model for Aglets", *IEEE Internet Computing*, 1(4), July 1997.

[8] S. Jajodia et al., "A Logical Language for Expressing Authorization", *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 1997.

[9] Charlie Lai et al., "User Authentication and Authorization in the Java Platform", *Proc. of the 15th Annual Computer Security Application Conference*, Phoenix, AZ, December 1999.