유비쿼터스 환경에서 Jini 서비스를 이용한 안전한 Java 객체 저장소

유양우°, 이명준
울산과학대학 컴퓨터디자인학부, 울산대학교 컴퓨터정보통신공학부
soft@mail.uc.ac.kr°, mjlee@ulsan.ac.kr

A Secure Repository for Java Objects using Jini Services

Yang-Woo Yu°, Myung-Joon Lee
School of Computer Design, Ulsan College

요 약

유비쿼터스 환경에서 많은 정보는 공유되고 누구나 쉽게 접근하여 정보를 교환할 수 있어야한다. 이를 위하여 Jini 서비스 중 하나인 JavaSpace는 Java 객체를 저장하고 저장된 객체에 접근할 수 있는 공간을 제공한다. JavaSpace 서비스는 객체를 공유하는 방법으로 매우 유용하게 사용되고 있지만, 보안성이 취약하여 객체정보에 대한 접근 보안이 요구되는 분산시스템의 개발에는 적합하지 않다. 본 논문에서는 JavaSpace의 취약한 보안성을 강화시켜 안전한 JavaSpace 서비스를 제공하는 SecureJS 시스템을 개발하여 보안 정책과 그 구현에 대하여 설명한다.

1. 서 론

인터넷기술의 발전과 빠른 확산으로 인하여 컴퓨터는 우리 생활에서 매우 중요한 역할을 담당하고 있다. 그 결과 컴퓨터를 이용하는 사용자들은 언제 어디서나 네트워크에 접속할 수 있는, 즉 우리의 일상이 네트워크로 연결되어 모든 장소에서 누구나 컴퓨터를 자유로이 이용할 수 있는 유비쿼터스(Ubiquitous) 컴퓨팅 환경을 요구하고 있다. 유비쿼터스 환경에서 많은 정보는 공유되고 누구나 쉽게 접근하여 정보를 교환할 수 있다. 이러한 환경에서 다양한 이기종 장치 상호간 통신을 지원하기 플랫폼으로 Jini 기반구조가 강력히 대두되고 있다. Jini 기술은 네트워크 상의 지능형 기기 또는 소프트웨어들이 Jini 네트워크에 접속과 동시에 서비스할 수 있는 기능을 제공하고 있다. 이러한 특징은 유비쿼터스 컴퓨팅을 지원하기 위한 필수적인 기반구조와 미들웨어를 개발하는 플랫폼으로 이용되고 있다.

Jini 서비스 중에 하나인 JavaSpace는 표준 인터페이스를 통하여 Java 객체를 저장하고, 저장된 객체를 그 클래스의 템플릿을 이용하여 읽거나 가져오는 새로운 패러다임을 갖고 있다[2, 3].

본 논문에서는 새로운 Jini2.0 보안모델을 적용하여 서비스에 대한 상호인증과 객체에 대한 접근제어 기능을 구현하였으며, JavaSpace 서비스를 안전하게 접근할 수 있도록 SecureJS 시스템을 개발하였다.

본 논문의 구성은 다음과 같다. 2장에서는 JavaSpace 서비스의 특징에 대하여 소개하고, 보안요소의 필요성을 기술한다. 3장에서는 안전한 Java 객체 저장소를 제공하기 위하여 Jini2.0 기반의 SecureJS 시스템의 설계 및 구현에 관하여 자세히 설명한다. 4장에서는 SecureJS의 실행방법을 소개하고, 마지막으로 5장에서 결론 및 향후 연구방향에 대해 살펴본다.

2. JavaSpace의 소개

JavaSpace 기술은 자바 환경의 새로운 분산 컴퓨팅 모델로서 자바객체를 저장하고 접근할 수 있는 공간을 말한다. 이는 자바의 원격 객체 호출 시스템인 RMI와 직렬화를 이용하며, 룩업서비스, 트랜잭션서비스 등 Jini서비스와 연계하여 분산처리를 위한 기능들을 제공함으로서 분산 시스템을 쉽게 구축할 수 있도록 한다.

JavaSpace를 이용한 프로그래밍 모델은 매우 간결하다. JavaSpace를 이용하고자 하는 응용프로그램은 Jini의 룩업서비스를 이용하여 JavaSpace에 접근할 수 있는 서비스프락시를 다운로드 한다. 그런 다음 서비스프락시를 이용하여 객체를 저장하고, 원하는 객체를 검색하여 그 객체를 읽거나 가져갈 수 있다. 이러한 패러다임은 객체를 공유하는 차원에서는 매우 유용하게 사용될 수 있다. 그러나 JavaSpace의 프락시를 룩업서비스로부터 구하기만 하면 어떠한 목적의 응용프로그램도 JavaSpace 내의 공간에 객체를 저장하거나 가져갈 수 있기 때문에 이러한 객체정보들에 대한 접근 보안이 요구되는 분산 응용프로그램 개발에는 적합하지 않다.

따라서, 본 연구에서는 기존의 JavaSpace의 기능에 Jini2.0 보안요소를[4] 추가하여 안전한 JavaSpace 서비스를 제공하고자 한다. JavaSpace 내의 객체에 대한 접근을 제한하기 위하여 SecureJS 시스템을 개발하였다. SecureJS 시스템은 접근관리자와 객체저장소 그리고 키관리자로 구성되어 있으며, 시스템에 관한 자세한 설명은 3장에서 논의할 것이다.

3. SecureJS: 안전한 Java 객체 저장소

SecureJS 시스템은 접근관리자(AccessManager)와 객체저장소(ObjectStore) 그리고 키 관리자(KeyManager)로 구성되어 있다. 객체저장소는 JavaSpace 서비스를 제공하는 객체저장소이며, 접근관리자는 인증된 클라이언트들에게 안전한 JavaSpace 서비스를 제공하는 데몬 형태로 동작하는 Jini 서비스이다. 그리고 키 관리자는 접근관리자에서 올바른 수신자인지 여부를 검사할 때 사용하는 공개키들을 관리한다[5]. 이러한 구성요소를 갖는 SecureJS 시스템은 분산컴퓨팅 환경에서 객체를 안전하게 공유하고 저장할 수 있는 매우 유용한 기술을 제공하고 있다.

3.1 ObjectStore(객체저장소)

JavaSpace는 엔트리(net.jini.core.entry.Entry) 인터페이스를 구현한 자바객체를 저장하는 Jini 서비스이다. 엔트리는 java.io.Serializable을 확장한 인터페이스이며, 엔트리 객체의 모든 속성은 직렬화할 수 있는 객체 참조이어야 한다. 사용자는 JavaSpace에 엔트리를 구현한 객체를 저장할 수 있으며, 엔트리 객체의 템플릿을 이용하여 매칭되는 엔트리 객체를 구할 수 있다.

객체저장소는 하나의 JavaSpace이며 Jini2.0 보안정책을 이

용하여 JavaSpace에 접근할 수 있는 모든 권한을 접근관리자에게만 부여한다. 객체저장소에 저장되는 자바객체는 엔트리와 수신자 id 정보를 포함한다. 수신자 id는 저장된 엔트리에 대하여 올바른 수신자에게 정확히 요청한 객체를 검색하여 전달하기 위해 사용된다. (그림1)의 코드는 SecureJS 시스템에서 사용되는 엔트리의 구조이다.

```
import net.jini.core.entry.Entry;
public class Id_Entry implements Entry {
    private String receiver_id;
    private Entry entry;

    public Id_Entry(String  rcv_id, Entry e) {
      receiver_id = rcv_id;
      entry = e;
    }
    public String get_id(){
      return receiver_id;
    }
    public Entry get_entry(){
      return entry;
    }
}
```

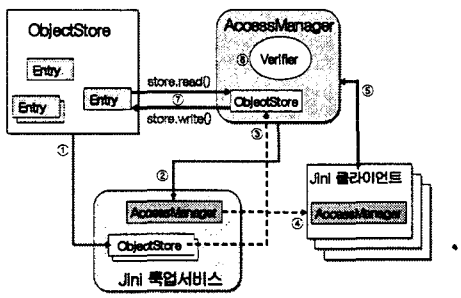
(그림 1) 객체저장소의 엔트리 구조

3.2 접근관리자(AccessManager)

일반적으로 클라이언트들은 JavaSpace 서비스를 이용하기 위하여 Jini 룩업서비스로부터 서비스프락시를 다운로드 받아야 한다. SecureJS 시스템은 JavaSpace 서비스에 대한 두 가지 보안사항을 적용하였다.

*첫 번째 보안사항*은 객체저장소에 대한 접근은 AccessManager만이 권한을 부여받아 서비스를 제공할 수 있다. 이는 객체저장소에 접근할 수 있는 인증된 주체를 "AccessManager"로만 한정시켰다. 그래서 클라이언트들은 안전한 JavaSpace의 서비스 이용하고자할 때, AccessManager를 통해서만 서비스 받을 수 있다. 즉, 모든 클라이언트들은 ObjectStore 서비스에 직접 접근 할 수 없다. 이러한 설계는 클라이언트들에게 안전한 JavaSpace 서비스를 제공하기 위한 새로운 보안모델로서 제시된다.

*두 번째 보안사항*은 JavaSpace에 저장된 엔트리에 대한 보안으로 인증된 클라이언트라 할지라도 그 엔트리를 접근할 수 있는 수신자가 아니라면 접근할 수 없도록 한것이다. 이는 *<올바른 수신자의 검증 알고리즘>*에서 자세히 설명할 것이다[8]. (그림 2)는 AccessManager의 동작을 설명하고 있다.



(그림 2) AccessManager의 동작

1) 자바객체를 저장하는 ObjectStore는 Jini 서비스로 동작하기 위해 룩업서비스에 자신의 서비스프락시를 등록시킨다. 그리고 자신을 접근할 수 있는 유일한 접근자로서 AccessManager를 지정한다.
2) AccessManager 또한 클라이언트들이 자신의 서비스를 받을 수 있도록 룩업서비스에 서비스프락시를 등록시킨다.
3) AccessManager는 Jini2.0 보안모델을 이용하여 ObjectStore의 프락시를 다운로드 한다. AccessManager를 제외한 다른 클라이언트들은 ObjectStore의 프락시를 다운로드 받을 수 없다.
4) 안전한 JavaSpace 서비스를 이용하기 위하여, 먼저 클라이언트들은 인증과정을 수행한 후, 룩업서비스를 통하여 AccessManager의 서비스프락시를 다운로드 받는다.
5) 클라이언트들은 다운로드 받은 AccessManager의 프락시를 이용하여 원격메소드를 호출한다.
6) AccessManager의 검증자(verifier)는 허가된 접근인지 여부를 검사한 후 적법한 클라이언트임이 검증되면 ObjectStore의 서비스프락시 내에 메소드를 호출하여 안전한 JavaSpace 서비스를 제공한다.
7) ObjectStore는 AccessManager에서 요구한 JavaSpace 서비스를 수행한다. 그리고 그 결과 값을 클라이언트에게 반환한다.

SecureJS 시스템은 기존의 JavaSpace에서 제공하는 오퍼레이션을 모두 지원하고 있으며, 클라이언트는 정의된 오퍼레이션을 수행함으로써 안전한 JavaSpace 서비스를 제공받을 수 있다. 접근관리자는 AccessManager_Impl 클래스를 이용하여 구현된다. 클라이언트는 write() 메소드를 호출하여 객체저장소에 엔트리를 저장한다. receiver_id인자는 엔트리에 접근할 수 있는 수신자의 id이다. 그리고 entry 인자는 객체저장소 내에 저장하고자 하는 자바객체 이다.

```
public Lease write(receiver_id, entry, txn, lease) {
   Id_Entry ie = new Id_Entry(receiver_id, entry);
   object_store.write(Id_Entry, txn, lease);
}
```

(그림 3) 접근관리자의 write() 메소드

객체저장소에 저장된 엔트리에 접근하기 위하여 read() 또는 take() 메소드를 이용할 수 있다. 접근관리자는 클라이언트가 이들 오퍼레이션을 호출할 때, 호출한 클라이언트가 올바른 수신자인지를 검증하고, 검증된 수신자라면 해당 오퍼레이션에 따른 객체저장소 서비스를 제공한다. (그림 4)는 read() 메소드 설명하고 있다.

```
public Entry read(encrypt_id, id, tmpl, txn, lease) throws
                                                NotTrusted {
  boolean trust;
  ldap_pub_key = ldapCtx.search(id, PK_FILTER.
                 constraints);        ---------- ①

  trust = verifier(ldap_pub_key, private_encrypt_id, id);
                                      ---------- ②
  if (trust) {
    Id_Entry  id_tmpl = new Id_Entry(id, tmpl);
                                      ---------- ③
    Id_Entry  ie = object_store.read(id_tmpl, txn, lease);
                                      ---------- ④
    return ie.get_entry();            ---------- ⑤
  }
                  :
```

(그림 4) 접근관리자의 read() 메소드

read() 오퍼레이션에서 첫 번째 인자 encrypt_id는 올바른 수신자임을 검증하기 위해 수신자 자신의 id를 수신자 개인키(private key)로 암호화시킨 값이다. 그리고 두 번째 인자는 엔트리에 접근을 허용할 수신자의 id이다. tmpl인자는 수신자가 접근하고자하는 엔트리를 검색하기 위한 템플릿이다. read()

오퍼레이션의 동작은 다음과 같다.

1) LDAP를 이용하는 키관리자에 접근하여 수신자의 id에 해당하는 수신자의 공개키를 얻어온다[7].
2) verifier(ldap_pub_key, encrypt_id, id) 메소드를 이용하여 수신자가 올바른 수신자인지를 검사한다.
   : ldap_pub_key인자는 키관리자의 search() 오퍼레이션 을 통하여 얻은 수신자의 공개키이며, encrypt_id는 수신자 id를 수신자 자신의 개인키로 암호화시킨 값이다. 그리고 id는 수신자 자신의 id이다.
   : 접근관리자는 수신자의 공개키를 가지고 수신자의 개인키로 암호화시킨 encrypt_id를 해독한다. 해독한 값이 read() 메소드의 두 번째 인자인 id값과 일치하면 올바른 수신자임이 검증된 것이다.
3) id_tmpl 인자는 객체저장소에 저장된 엔트리를 검색하기 위한 템플릿이다.
4) 템플릿에 정확히 매칭되는 엔트리를 객체저장소의 read() 메소드를 호출하여 가져온다.
5) 검색된 엔트리에서 실제 수신자가 요구하는 엔트리만을 수신자에게 전달한다.

3.3 KeyManager(키 관리자)를 이용한 공개키 관리

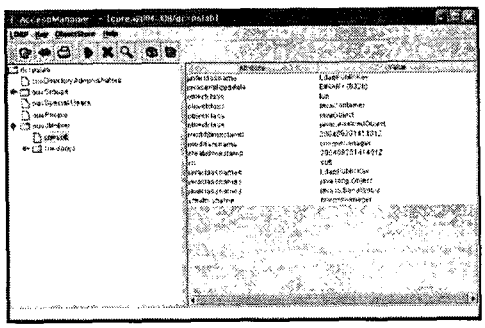
SecureJS 시스템에서 접근관리자는 사용자의 신원을 인증하기 위한 방법으로 JDK에서 지원하는 DSA 보안 알고리즘을 사용하며[6], 각 사용자들에 대한 공개키와 개인키를 구할 수 있다. 그리고 DSA 알고리즘을 이용하여 얻은 공개키는 LDAP를 이용하는 KeyManager에서 관리된다[7]. AccessManager와 클라이언트는 상호인증을 위해 KeyManager에서 관리되는 공개키를 사용한다.

SecureJS 시스템에서 KeyManager는 공개키 관리를 위하여 LDAP 디렉토리를 사용한다. 디렉토리는 검색할 정보의 커다란 집합으로 볼 수 있다. 전화번호부와 같이 정보에 거의 변화가 없고, 매우 자주 검색되는 정보는 디렉토리의 영역이라고 볼 수 있다. KeyManager의 주요 기능은 다음과 같다.

- **바인딩(Binding)** : LDAP는 바인딩과 인증을 구별하지 않으며, 디렉토리 서버로 바인딩할 때 원하는 서버와 자신에 대한 정보(계정, 암호)를 함께 지정할 수 있다.
- **검색(Searching)** : 검색을 하려면 검색의 범위를 지정하여 search() 메소드를 호출해야 한다.
- **엔트리 추가(Adding Entries), 엔트리 변경(Modifying Entries), 엔트리 제거(Deleting Entries)**

4. SecureJS의 실행

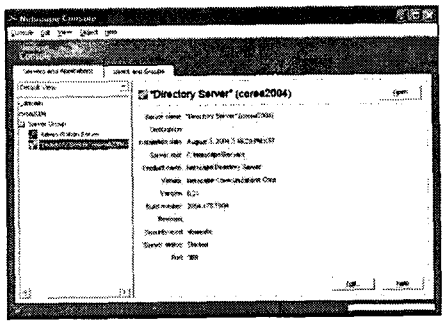
SecureJS는 인증된 클라이언트에게 안전한 JavaSpace 서비스를 제공한다. SecureJS 시스템을 기동시키기 위한 절차는 다음과 같다.



(그림 5) SecureJS의 메인화면

1) JavaSpace인 객체저장소를 먼저 실행시킨다.
2) Netscape 사의 디렉토리 서버를 실행한다.
3) 접근관리자를 실행시킨다.

(그림 5)은 SecureJS 시스템을 실행시키기 위한 메인화면이다. LDAP 서버와 연결된 모습을 보여주고 있다. (그림 5)의 실행화면에서 왼쪽의 디렉토리 구조는 접근관리자와 연결된 LDAP 서버의 정보를 보여준다. 화면의 오른쪽 테이블은 현재 DC(domain component)가 pslab이고, OU(organization unit)가 JMoblet이며, CN(common name)이 soft인 사용자의 공개키를 포함하는 정보를 보여준다. (그림 6)은 SecureJS에서 이용하는 LDAP 서버의 현재 실행되고 있는 모습을 보여주고 있다. LDAP 디렉토리 서버는 Netscape사에서 제공하는 디렉토리 서버 6.2를 이용하였다.



(그림 6) Netscape사의 디렉토리 서버

5. 결론 및 추후연구

본 논문에서는 새로운 Jini2.0 보안모델과 프로그래밍 모델을 적용하여 JavaSpace 서비스를 안전하게 제공할 수 있는 SecureJS 시스템을 개발하였다. SecureJS 시스템은 접근관리자와 객체저장소 그리고 키관리자로 구성되어 있다. 개발된 Jini2.0 기반의 SecureJS 시스템은 유비쿼터스 컴퓨팅 환경에서 Java 객체를 안전하게 공유하고 저장할 수 있는 매우 유용한 기술을 제공하고 있다. 그 결과 본 시스템의 활용 및 응용 영역은 매우 다양할 것으로 여겨진다.

추후연구는 J2ME 환경에서 Jini 클라이언트를 작성하여 모바일기기에서도 SecureJS 시스템을 이용할 수 있는 환경으로 개선할 것이다.

[참고문헌]

[1] Sun Microsystems, *"Jini™ Architecture Specification."*, Published Specification, http://java.sun.com/products/jini/2.0/doc/specs/html/jini-spec.html, 2003.

[2] Sun Microsystems, *"Jini™ Technology Starter Kit Overview v2.0."*, Published Specification, http://java.sun.com/developer/products/jini/arch2_0.html, 2003.

[3] Sun Microsystems, *"JavaSpaces™ Service Specification."*, Published Specification, http://wwws.sun.com/software/jini/specs/jini1.2html/js-title.html, 2002.

[4] Frank Sommers, *"Jini Starter Kit 2.0 tightens Jini's security framework"*, Los Alamitos, CA., IEEE Computer Society Press, 2003.

[5] Sun Microsystems, *"Security enhancements for the Java2 SDK"*,http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html, 2003.

[6] Rob Weltman, Tony Dahbura, *"LDAP Programming with Java."*, Addison-Wesley, 2000.

[7] Sun Microsystems, Inc. JNDI Implementor Guidelines for LDAP Service Providers Draft 0.2.

[8] 유양우, 문남두, 정혜영, 이명준, "SecureJS: Jini2.0 기반의 안전한 JavaSpace", 한국정보처리학회, 제11-C권, 제6호, 2004. 12.