

# Jini 서비스를 이용한 안전한 Java 객체 저장소의 개발

유양우<sup>o</sup>, 박희중, 이명준

울산과학기술대학교 컴퓨터디자인학부, 울산대학교 컴퓨터정보통신공학부  
soft@mail.uc.ac.kr<sup>o</sup>, heejong34@mail.ulsan.ac.kr, mjlee@ulsan.ac.kr

## A Secure Repository for Java Objects using Jini Services

Yang-Woo Yu<sup>o</sup>, Hee-Jong Park, Myung-Joon Lee  
School of Computer Design, Ulsan College

### 요 약

Jini 서비스는 개발자에게 분산시스템을 쉽게 개발할 수 있는 하부구조를 제공한다. Jini 서비스 중 하나인 JavaSpace는 자바환경의 분산컴퓨팅 모델로서 Java 객체를 저장하고 저장된 객체에 접근할 수 있는 공간을 말한다. 이러한 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 시스템에 대하여 설명한다.

### 1. 서 론

Java 객체를 이용한 기술은 빠르게 발전하고 있으며 그에 따른 활용할 수 있는 분야도 더욱더 다양해졌다. 네트워크의 개념 또한 다수의 연결된 컴퓨터에서 유용한 서비스를 제공하는 장치로 그 영역이 점점 더 확대되어 가고 있다. 이러한 네트워크의 발전과 함께 시스템들의 상호협력력을 위하여 Java 객체를 공유하고, 동적인 통신을 가능하게 해주는 새로운 분산 기반 구조인 Jini 기술이 등장하였다[1].

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]. 하지만, 기존의 JavaSpace는 보안기능이 취약하여 누구나 Java 객체를 저장하고 저장된 객체를 아무런 제약 없이 누구든지 접근할 수 있었다. 그러므로 보안이 요구되는 정보를 교환하거나 보관할 경우 JavaSpace는 객체 저장소로서의 서비스를 제공할 수 없는 단점을 가지고 있다.

최근에 썬 마이크로시스템즈(Sun Microsystems)에서는 분산 시스템 환경에서 자바코드의 이동성에 따른 보안요소를 충족시키는 Jini2.0을 제시하였다[4]. 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 이용하여, 원격서비스, 트랜잭션서비스 등 Jini서비스와 연계하여 분산처리를 위한 기능들을 제공함으로써 분산 시스템을 쉽게 구축할 수 있도록 한다.

JavaSpace를 이용한 프로그래밍 모델은 매우 간결하다. JavaSpace를 이용하고자 하는 응용프로그램은 Jini의 록업서비스를 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 록업서비스로부터 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 데온 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 이용하여 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 전달하

기 위해 사용된다. (그림 1)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;
    }
}
```

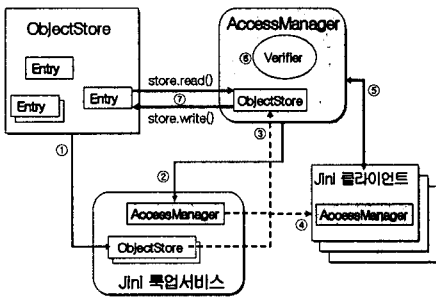
(그림 1) 객체저장소의 엔트리 구조

### 3.2 접근관리자(AccessManager)

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

첫 번째 보안사항은 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 이용하고자 할 때, AccessManager를 통해서만 서비스 받을 수 있다. 즉, 모든 클라이언트들은 ObjectStore 서비스에 직접 접근할 수 없다. 이러한 설계는 클라이언트들에게 안전한 JavaSpace 서비스를 제공하기 위한 새로운 보안모델로서 제시된다.

두 번째 보안사항은 JavaSpace에 저장된 엔트리에 대한 보안으로 인증된 클라이언트라 할지라도 그 엔트리를 접근할 수 있는 수신자가 아니라면 접근할 수 없도록 한 것이다. 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 록업서비스에 자신의 서비스프락시를 등록시킨다. 그리고 자신을 접근할 수 있는 유일한 접근자로서 AccessManager를 지정한다.
- 2) AccessManager 또한 클라이언트들이 자신의 서비스를 받을 수 있도록 록업서비스에 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, 록업서비스를 통하여 AccessManager의 서비스프락시를 다운로드 받는다.
- 5) 클라이언트들은 다운로드 받은 AccessManager의 프락시를 이용하여 원격메소드를 호출한다.
- 6) AccessManager의 검증자(verifier)는 허가된 접근인지 여부를 검사한 후 적절한 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 자바객체이다.

```
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 encrvot 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) LDAP을 이용하는 키관리자에 접근하여 수신자의 id에 해당하는 수신자의 공개키를 얻어온다.[7].

