# SecureJS: Jini2.0 기반의 안전한 JavaSpace의 구현

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

# SecureJS: A Secure JavaSpace based on Jini2.0

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

**요 약**

Jini 서비스 중 하나인 JavaSpace는 자바환경의 분산 컴퓨팅 모델로서 객체를 저장하고 저장된 객체에 접근할 수 있는 공간을 말한다. 이러한 JavaSpace 서비스는 객체를 공유하는 방법으로 매우 유용하게 사용되고 있 만, 보안성이 취약하여 객체정보에 대한 접근 보안이 요구되는 분산시스템의 개발에는 적합하 않다. 본 논문에서는 JavaSpace의 취약한 보안성을 강화시켜 안전한 JavaSpace 서비스를 제공하는 SecureJS 시스템에 대하여 설명한다.

## 1. 서 론

인터넷 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 상호협력을 위하여 객체를 공유하고, 동적인 통신을 가능하게 해주는 새로운 분산 기반 구조인 Jini 기술이 등장하였다[1].

Jini 서비스 중에 하나인 JavaSpace는 표준 인터페이스를 통하여 객체를 저장하고, 저장된 객체를 그 클래스의 템플릿을 이용하여 읽거나 가져오는 새로운 패러다임을 갖고 있다[2, 3]. 이러한 특징을 이용하여 분산객체의 영속성과 데이터 교환 기능을 원하는 분산시스템을 구현하는데 많이 활용되고 있다. 하 만, 기존의 JavaSpace는 보안기능이 취약하여 누구나 객체를 저장하고 저장된 객체를 아무런 제약 없이 누구든 접근할 수 있었다. 그러므로 보안이 요구되는 정보를 교환하거나 보관할 경우 JavaSpace는 객체 저장소로서의 서비스를 제공할 수 없는 단점을 가 고 있다.

최근에 썬 마이크로시스템즈(Sun Microsystems)에서는 분산시스템 환경에서 자바코드의 이동성에 따른 보안요소를 충족시키는 Jini2.0을 제시하였다[2]. 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 특징을 소개하고, 보안요소의 필요성을 기술한다. 3장에서는 안전한 JavaSpace 서비스를 제공하기 위하여 Jini2.0 기반의 SecureJS 시스템의 설계 및 구현에 관하여 자세히 설명한다. 마 막으로 4장에서 결론 및 향후 연구방향에 대해 살펴본다.

## 2. JavaSpace 서비스에서 보안요소의 필요성

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 적합하 않다.

따라서, 본 연구에서는 기존의 JavaSpace의 기능에 Jini2.0 보안요소를[4] 추가하여 안전한 JavaSpace 서비스를 제공하고자 한다. 안전한 JavaSpace 서비스를 제공하기 위해 다음의 두 가 사항을 고려한다. 첫째는 신원이 허용되고 확인된 클라이언트만이 룩업서비스로부터 JavaSpace의 서비스프락시를 구할 수 있도록 제한한다. 둘째는 JavaSpace에 객체를 저장할 때 객체에 접근할 수 있는 수신자를 정하고, 정된 수신자만이 객체에 접근할 수 있도록 제한한다. 서비스프락시에 대한 접근을 제한하기 위해 Jini2.0 기반의 JERI 모델과 JAAS 인증기능을 이용하였다[4]. 그리고 JavaSpace 내의 저장된 객체에 대한 접근을 제한하기 위하여 SecureJS 시스템을 개발하였다. SecureJS 시스템은 AccessManager와 ObjectStore로 구성되어 있으며, 시스템에 관한 자세한

설명은 3장에서 논의할 것이다.

## 3. SecureJS 시스템

### 3.1 SecureJS 시스템의 보안정책

SecureJS 시스템은 AccessManager(접근관리자)와 ObjectStore(객체저장소) 그리고 KeyManager(키 관리자)로 구성되어 있다. ObjectStore는 JavaSpace 서비스를 제공하는 객체저장소이며, AccessManager는 인증된 클라이언트들에게 안전한 JavaSpace 서비스를 제공하는 데몬 형태로 동작하는 Jini2.0 서비스이다. 그리고 KeyManager는 AccessManager에서 올바른 수신자인 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. 그림 1)은 SecureJS의 구조를 보여주고 있다.

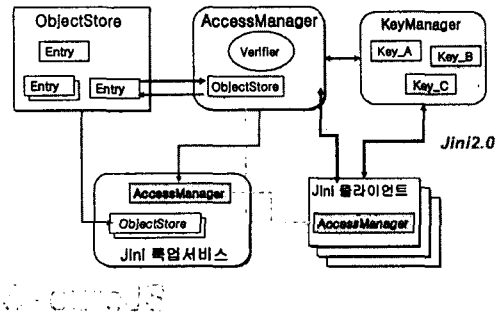


그림 1) SecureJS 시스템의 구조

### 3.2 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.

ObjectStore는 하나의 JavaSpace이며 Jini2.0 보안정책을 이용하여 JavaSpace에 접근할 수 있는 모든 권한을 AccessManager에게만 부여한다. 이러한 작업을 위해 ObjectStore는 Jini2.0 환경설정 파일을 작성해야 하며 그림 2)에서 ①, ② 코드는 JavaSpace의 접근권한을 AccessManager에게만 부여하는 환경설정의 일부분이다.

```
Server ObjectStore {
 private static AccessManagerKey = KeyStores.getX500
   Principal("AccessManager".caTruststore); ----- ①
                    :
 AuthenticationPermission(AccessManagerKey,
       ObjectStoreKey, "connect");          ----- ②
}
```

그림 2) ObjectStore의 환경설정

ObjectStore에 저장되는 자바객체는 엔트리와 수신자 id 정보를 포함한다. 수신자 id는 저장된 엔트리에 대하여 올바른 수신자에게 정확히 요청한 객체를 검색하여 전달하기 위해 사용된다.

### 3.3 AccessManager(접근관리자)

SecureJS 시스템은 JavaSpace 서비스에 대한 두 가지 보안사항을 적용하였다.

- JavaSpace에 접근할 수 있는 프로세스를 AccessManager로 한정시켜 클라이언트들이 JavaSpace에 접근할 경우 AccessManager를 통해서만 접근 할 수 있다.
- JavaSpace에 저장된 엔트리에 대하여 클라이언트들이 접근하고자 할 때, AccessManager는 적법한 수신자임을 검증하여 그 클라이언트에게 서비스를 제공한다.

SecureJS 시스템은 자바객체를 저장할 수 있는 공간으로 ObjectStore를 사용하고 있으며, ObjectStore에 대한 접근은 AccessManager만이 권한을 부여받아 서비스를 제공할 수 있다. 이는 그림 3)의 환경설정 파일에서 정한다. ObjectStore에 접근할 수 있는 인증된 주체를 "AccessManager"로만 한정시켰다. 그래서 클라이언트들은 안전한 JavaSpace의 서비스 이용하고자할 때, AccessManager를 통해서만 서비스 받을 수 있다. 즉, 클라이언트들은 ObjectStore 서비스에 직접 접근 할 수 없다. 이러한 설계는 클라이언트들에게 안전한 JavaSpace 서비스를 제공하기 위한 새로운 보안모델로서 제시된다.

두 번째 보안사항은 JavaSpace에 저장된 엔트리에 대한 보안으로 인증된 클라이언트라 할 라도 그 엔트리를 접근할 수 있는 수신자가 아니라면 접근할 수 없도록 한 것이다. 이는 <*올바른 수신자의 검증 알고리즘*>에서 자세히 설명할 것이다. 그림 3)은 AccessManager의 동작을 설명하고 있다.

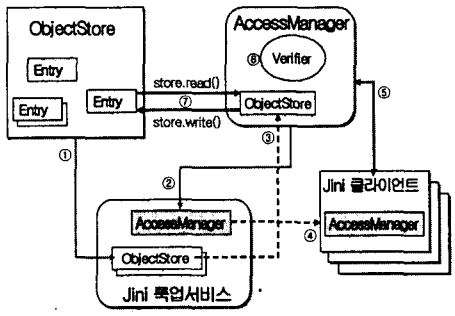


그림 3) AccessManager의 동작

1) 자바객체를 저장하는 ObjectStore는 Jini 서비스로 동작하기 위해 룩업서비스에 자신의 서비스프락시를 등록시킨다. 그리고 자신을 접근할 수 있는 유일한 접근자로서 AccessManager를 정한다.
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 접근인 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 시스템의 주요 오퍼레이션들은 AccessManager_Impl 클래스 내에 정의되어 있으며, 이를 이용하여 엔트리를 저장하고 검증된 수신자가 저장된 엔트리를 읽는 과정을 설명한다.

< 올바른 수신자의 검증 알고리즘 >

1) 인증된 클라이언트는 엔트리(Entry)를 SecureJS 내에 write() 메소드를 이용하여 저장할 수 있다. 일반적인 JavaSpace의 write() 메소드와는 달리 SecureJS 에서는 그 엔트리에 접근할 수 있는 수신자의 id(receiver_id)를 write() 메소드에 매개변수로 지정한다. 이는 올바른 수신자를 검증하기 위해서이다.
2) 엔트리에 수신자를 지정하여 SecureJS 시스템의 write(receiver_id, entry, txn, lease) 메소드를 호출하여 엔트리를 저장한다.
3) 클라이언트는 read() 메소드를 이용하여 ObjectStore 내에 저장된 엔트리에 접근을 시도할 때, AccessManager는 저장된 객체를 서비스하기 전에 그 클라이언트가 해당 객체에 접근할 수 있는 올바른 수신자인지 검증해야한다.
4) 먼저, 수신자는 자신이 올바른 수신자임을 증명하기 위하여 자신의 id를 개인키로 암호화시킨 값과 자신의 id를 read() 메소드를 이용하여 매개변수로 전달한다.

예) `read(private_encrypt_id, id, id_tmpl, txn, lease)`

5) AccessManager는 수신자의 개인키로 암호화시킨 값과 수신자 id 그리고 KeyManager를 이용한 수신자의 공개키 정보를 이용하여 올바른 수신자임을 검증할 수 있다.
6) KeyManager에서 구한 수신자의 공개키를 사용하여 수신자의 개인키로 암호화된 값을 해독하여 id 값을 구한다. 이 값과 수신자가 보낸 id가 일치하면, 이는 올바른 수신자임이 검증된 것이다.

### 3.4 KeyManager(키 관리자)를 이용한 공개키 관리

SecureJS 시스템에서 AccessManager는 사용자의 신원을 인증하기 위한 방법으로 JDK에서 지원하는 DSA 보안 알고리즘을 사용하며[5], 각 사용자들에 대한 공개키와 개인키를 구할 수 있다. 그리고 DSA 알고리즘을 이용하여 얻은 공개키는 LDAP를 이용하는 KeyManager에서 관리된다[6]. AccessManager와 클라이언트는 상호 인증을 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) : 디렉토리 구조에서 관리되는 각 엔트리를 추가, 변경, 삭제할 수 있다.

## 4. 결론 및 추후연구

본 논문에서는 새로운 Jini2.0 보안모델과 프로그래밍 모델을 적용하여 JavaSpace 서비스를 안전하게 제공할 수 있는 SecureJS 시스템을 개발하였다. SecureJS 시스템은 AccessManager와 ObjectStore 그리고 KeyManager로 구성되어 있다. 개발된 Jini2.0 기반의 SecureJS 시스템은 분산 컴퓨팅 환경에서 객체를 안전하게 공유하고 저장할 수 있는 매우 유용한 기술을 제공하고 있다. 그 결과 본 시스템의 활용 및 응용영역은 매우 다양할 것으로 여겨진다.

추후연구는 이전 연구에서 개발된 이동 에이전트시스템인 JMoblet 시스템[7]을 보안성이 강화된 Jini2.0 시스템 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] 김진홍, 구형서, 유양우, 이명준, *"JMoblet: Jini 기반의 이동에이전트시스템"*, 한국정보처리학회논문 B 제8-B 권 제6호, pp.292-312, 2001.

[8] 유양우, 이명준, *"분산응용프로그램을 위한 안전한 JavaSpace."*, 한국정보과학회 춘계학술발표회, pp.352-354, 2003.