PKDA 를 이용한 Jini Security 의 확장*

°전문광 유지영 송대기 이철훈
충남대학교 컴퓨터공학과
{mgjeon, jyyoo, dksong, chlee}@ce.cun.ac.kr

Extending Jini security with PKDA

Mun-Gwang Jeon° Ji-Young Yoo Dae-Ki Song Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National Univ.

요 약

Sun Microsystems 에 의해서 개발된 Jini 네트워킹 기술은 자바 프로그래밍 언어를 기반으로 하여 분산 환경을 만드는데 좋은 아키텍쳐를 제공한다. 그러나 현재의 자바 security solution 만으로는 분산환경에서의 보안 요구사항을 충족시키기에 충분하지 않다. 이를 해결하기 위한 여러 방법들이 있지만, 대부분이 centralized computing 을 기반으로 하여 분산환경에는 맞지 않다. 이 논문에서는 PKDA(Public key based Kerberos for Distributed Authentication)를 사용하여 Jini 의 security 를 확장함으로써 분산환경에서의 security 요구사항을 충족시킬 수 있음을 보인다.

1. 서 론

최근에는 크기가 작고 정보처리 기능을 갖는 디바이스들이 꾸준히 증가하고 있고 사용자들은 네트워크상의 다른 디바이스에 의해서 제공되는 서비스들을 사용하기 위해서 이들이 네트웍에 연결되기를 바란다. 그러나, 오늘날의 네트워크상의 디바이스들은 매우 다양하여 네트워크 서비스의 사용에 대한 표준이 요구된다. 이러한 요구에 대한 하나의 해결책이 Sun Microsystems 의 Jini 이다.

자바 플랫폼의 상위에 덧붙여진 Jini 는 Sun Microsystems 에 의해서 개발된 것으로, 자바로 쓰여진 proxy-objects 를 통해서 여러 서비스들과 자발적인 통신을 제공한다. 서비스들은 Jini lookup service 를 통해서 찾아지고 Java 로 쓰여진, proxy 들은 동적으로 다운로드 되어 클라이언트의 자바 가상 머신에서 실행된다.

그러나 이와 같은 실행환경에서 컴퓨터 네트워크가 불안정하기 때문에 이를 보완하기 위한 보안 특성들이 요구된다. 예를 들어, 서비스를 제공하는 서버는 서비스를 사용하려는 사용자가 정당한 사용자인지, 어떤 권한을 가지고 있는지, 클라이언트가 자신이 접근하려는 서버에 제대로 접근하고 있는지를 확인하고자 할 것이다. 그러나, 현재의 자바 security 특성들만을 가지고는 이러한 여러 보안 특성들을 만족시키기가 힘들다.

본 논문의 구성은 다음과 같다. 2 장에서는 Jini 의 기본 개념과 주요 구성 요소, 그리고 PKDA(Public Key based Kerberos for Distributed Authentication)에 대하여 설명한다. 3 장에서는 2 장에서 설명한 PKDA 를 사용하여 Jini 의 security 를 확장하는 과정에 대하여 설명을 한다. 마지막으로, 4 장에서는 결론과 향후 연구과제를 기술한다.

2. 관련 연구

2.1 Jini

Jini 아키텍쳐의 목적은 디바이스들의 그룹과 소프트웨어 콤포넌트(software component)들을 하나의 동적인 분산 시스템으로 연합하는 것이다.

Jini 의 기본적인 개념들은, Services, Lookup Service, RMI, Leasing, Transactions, Events 이다.

2.1.1 Services

Jini architecture 의 가장 중요한 요소로써 사람들, 프로그램, 또는 다른 service 에 의해서 사용될 수 있는 service 를 수행한다.

2.1.2 Lookup Service

시스템을 위한 중심적인 bootstrapping 메커니즘이다. Service 들은 *discovery* 와 *join* 이라 불리는 한 쌍의 프로토콜을 통해서 Lookup Service 에 등록되며 클라이언트들은 Lookup Service 를 통해서 원하는 service 를 찾는다.

2.1.3 RMI

RMI(Remote Method Invocation)는 근본적으로 전형적인 원격 프로시져 호출(remote procedure call) 메커니즘들에 대한 자바 프로그래밍 언어의 확장이다. Jini 에서 service 들 사이의 통신은 자바 RMI 를 통해서 이루어진다.

2.1.4 Leasing

* 이 논문은 BK21 대전, 충남 정보통신인력양성 사업단의 RA 연구비 지원에 의한 것임.

Jini 시스템의 service 들에 대한 접근은 lease 기반이다. Service 들이 Lookup Service 에 등록될 때 그들은 임의의 정해진 시간을 가지고 등록이 되고, 이 기간이 지나기 전에 다시 갱신된다. 이 기간이 갱신되지 않은 경우에는 그 자원이 해제 되었다는 것을 의미한다.

2.1.5 Transactions

다중 서비스들을 수반하는 안전한 연산들을 위한 Jini 메커니즘이다.

2.1.6 Events

Jini 아키텍쳐는 분산 이벤트들을 제공하는데, 이것은 event 들이 다른 호스트들에 위치한 서로 다른 가상 머신 사이에 전달되는 것을 허용한다. 예를 들어 Remote Events 는 새로운 service 들이 추가되거나, 변하거나, Jini 네트워크로부터 제거될 때, 클라이언트나 service 들을 인식하는데 사용된다.

Client 가 service 를 사용하는 과정을 살펴보면 아래의 그림 1 과 같다.

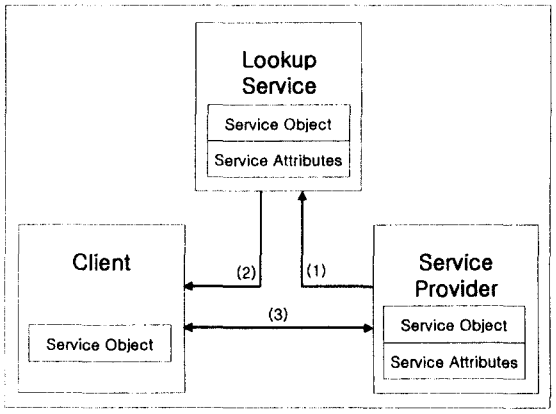

그림 1. Jini Service Architecture

(1)과정에서 우선 service 들은 *discovery, join* 프로토콜을 통해서 Lookup Service 를 찾고 이 곳에 자신의 service object(proxy)와 service attributes 를 등록하게 된다. 다음에 (2)과정에서 클라이언트는 멀티캐스팅을 통한 방법으로 여러 Lookup Service 에게 메시지를 전송하여 자신이 원하는 service 가 등록되어 있는 Lookup Service 를 찾고 이 곳으로부터 service object 를 다운 받는다. 마지막으로 (3)단계에서 클라이언트는 이 service object 를 통해서 자신이 원하는 service 와 통신을 한다.

2.2 PKDA

PKDA 는 Kerberos ticket framework 내에서 public key cryptography 를 사용하는 fully distributed authentication 을 위한 방법으로 제안되었다[2]. 신뢰받고 있는 중간 단계로부터 대부분의 인증작업을 통신 주체들로 분산시킴으로써, Kerberos V5 와 비교하여 security 와 scalability 에 대한 현저한 상승이 달성될 수 있다.

이 방법에서는 전형적인 Kerberos 기법과는 달리 client 는 application server 와 직접적으로 통신하는 기법을 사용하게 된다. 이 단계는 아래와 같다.

1. C → S : SCERT_REQ
2. S → C : SCERT_REP
3. C → S : PKTGS_REQ
4. S → C : PKTGS_REP
5. C → S : AP_REQ

notation :
SCRET_REQ : Request for Server's Certificate
SCERT_REP : Response of Server's Certificate
PKTGS_REQ : Public key based TGS Request
PKTGS_REP : Public key based TGS Response
AP_REQ : Application Service Request

위의 단계를 통해서 이 방법이 완전히 분산된 환경에서 실행될 수 있다는 것을 명백히 볼 수 있다. 초기 인증에서는(단계 3) public key 암호화 기법을 사용하고, 이후의 단계에서는 좀 더 빠른 symmetric cryptography 를 사용한다.

처음의 두 단계는 어떤 암호화를 사용하지 않는다. 이 단계에서 교환되는 정보들은 모두 공개 정보인데, certificate 의 무결성은 다음의 단계에서 검증되기 때문이다. 클라이언트가 certificate 를 저장할 수 있는 능력이 있다면 이들 두 단계는 되풀이되는 인증을 위해서 피할 수 있다. 그러므로, 초기의 인증은 public key 암호화를 사용하여 단계 3 에서 일어나게 된다.

또한 이 방법은 proxiable and forwardable tickets 를 통하여 rights delegation 을 지원할 수 있다.

이 방법은 클라이언트와 자신의 대리자간에 세션을 생성하고, 단계 3 에서 서버에게 proxiable flag 를 설정한 PKTGS_REQ 를 요청하게 된다. 이후에 서버가 proxiable ticket 으로 응답을 하게 되면, 클라이언트는 대리자가 적절한 서버에게 제출할 인증자를 생성하고 proxy key 와 함께 대리자에게 전달한다. 이후에 대리자와 서버와의 통신을 수행하게 된다.

3. 설계

이번 장에서는 PKDA 를 사용하여 Jini 의 service 와 클라이언트 사이의 보안을 확장하는 과정에 대해서 소개한다.

우선 Jini security 를 위한 요구사항을 살펴보고, Jini 의 service 사용 과정에 따라 요구되는 security 를 보장하기 위한 PKDA 의 사용을 설계한다.

3.1 보안 요구사항

이 논문에서는 일반적인 Jini service 를 제공하는 Service provider 와 클라이언트에 필요한 security 에 대해 초점을 맞춘다.

Jini services 를 사용할 때, 아래와 같은 사항들이 요구될 것이다.

- 올바른 service 와 통신하고 있다.
- 다른 어떤 사람도 이 통신을 도청할 수 없다.
- 합법적인 사용자들만이 service 를 접근한다.
- Untrusted code 로부터 보호한다.

이러한 요구 사항들을 충족시키기 위해서 클라이언트는 여러 service 들로부터 자신이 원하는 service 를 구별할 수 있어야 하고, service 를 나타내는 다운로드된 proxy 를 검증할 필요가 있다.

또한 서로의 통신에 사용되는 메시지를 암호화 하기 위한 key 의 생성과 이에 대한 서로의 동의와 인증이 필요하다.

3.2 Design

아래의 그림 2 는 2 장에서 설명한 Jini Service 의 일반적인 과정에 따른 보안 과정을 보인다.

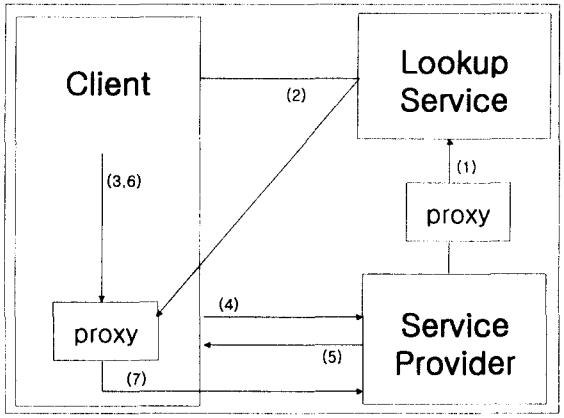

그림 2. Jini Service 의 PKDA 적용 단계

위 그림의 번호는 클라이언트가 service 를 사용하는 순서를 나타내고, 각 단계에 대한 설명은 아래와 같다.

(1) service provider 는 자신의 proxy 를 private key 를 사용하여 sign 한 후에 Lookup Service 에 등록한다.

(2) 클라이언트는 Lookup Service 로부터 원하는 service 를 찾고 해당 proxy 를 다운 받는다.

(3) client 는 proxy 로부터 service 의 public key 를 구하고, proxy 가 이 public key 를 사용하여 sign 되었는지를 검증한다.

(4) 클라이언트는 proxiable flag 를 set 한 PKTGS_REQ 메시지를 생성하여 service 에게 전송한다. 이 메시지는 service 의 public key 로 암호화 되어 있다. 따라서 해당 service 만이 이를 해독하고 인증할 수 있게 되므로, service 는 클라이언트의 identity 를 결정하고, 클라이언트는 service 의 identity 를 납득할 수 있게 된다. 또한 이 메시지에는 service 가 클라이언트에게 응답을 할 때 사용하는 암호화 key 가 포함되어 있다.

(5) service 는 PKTGS_REQ 메시지를 해독하고 정당한지 검증을 한 후에, proxiable ticket 을 생성한다. 이 ticket 은 service 만이 알고 있는 symmetric key 를 사용하여 암호화 되어 있고, 이로써 client 가 이 ticket 을 변화하는 것을 막을 수 있다. 응답 메시지의 나머지 부분은 PKTGS_REQ 메시지 안에 있는 일회용 key 를 사용하여 암호화 된다.

(6) 클라이언트는 proxy 가 service 에게 제출할 authenticator 를 생성한다. 여기에는 proxy 와 service 간에 사용할 session key 가 포함되고 proxy 의 권한에 대한 제한이 포함된다. 클라이언트는 이 authenticator 와 proxiable ticket, 그리고 session key 를 proxy 에게 전달한다.

(7) proxy 는 이 authenticator, proxiable ticket, session key 를 사용하여 service 에게 원하는 작업을 요청한다.

(8) service 는 proxy 로부터의 메시지를 받아서 검증한다.

2 장의 PKDA 의 과정에서 나온 처음 두 단계는 클라이언트와 Lookup Service 간의 통신으로 대체되었다. 클라이언트는 Lookup Service 에서 원하는 service 의 proxy 를 다운 받아서 proxy 를 검증함으로써 이루어진다.

이후의 단계는 2 장에서 설명한 바와 같고, 클라이언트는 proxy 에게 권한을 위임함으로써 proxy 가 해당 service 와 통신하는 것을 가능하게 한다.

클라이언트와 service 의 서로간의 identity 에 대한 납득은 (4)에서 설명한 바와 같다. 그리고 메시지는 서로간의 session key 를 사용하여 암호화 되므로 다른 악의 있는 제 삼자에 의해서 악용될 위험으로부터 보호된다.

또한 proxy 를 service 의 private key 를 사용하여 sign 하고 이를 다운 받은 클라이언트쪽에서 service 의 public key 를 사용하여 검증함으로써 untrusted code 로부터의 보호를 이룰 수 있다.

4. 결론 및 향후 연구과제

이 논문에서는 일반적인 Jini service 를 접근하는 클라이언트와 service 간의 통신에 대해서 요구되는 보안 특성들을 찾고, PKDA 를 사용하여 이러한 요구사항을 충족시킴을 보였다. 하지만, Jini 의 다른 core services 를 위한 문제는 다루지 못하였다.

향후 연구 과제로는 이러한 Jini 의 core services 에 대한 security 를 위해 요구되는 사항들을 살피고, 이들을 만족시키기 위한 방법에 대한 연구가 필요하다.

5. 참고문헌

[1] http://www.sun.com/jini/specs/

[2] T. Wu. *A real world analysis of kerberos password security.* In Proceedings of the 1999 Internet Society Network and Distributed System Security Symposium, 1999.

[3] J. Kohl, C. Neuman. The Kerberos Authentication Service(v5). Internet RFC 1510, September 1993.

[4] Pasi Eronen and Pekka Nikander. *Decentralized Jini security.* To appear in Proceedings of the Network and Distributed System Security Symposium (NDSS 2001), San Diego, California, February 2001.

[5] D. S. Wallach, D. Balfanz, D. Dean, and E. W. Felten. *Extensible security architectures for Java.* Technical report 546-97, Department of Computer Science, Princeton University, Apr. 1997.