OSGi 서비스 프레임워크 환경에서의 서비스 번들 인증 메커니즘

김영갑°, 문창주, 박대하, 백두권
고려대학교 컴퓨터학과 소프트웨어 시스템 연구실
{ygkim°, mcj, dhpark, baik}@software.korea.ac.kr

Service Bundle Authentication Mechanisms in the OSGi Service Framework Environment

Young-Gab Kim°, Chang-Joo Moon, Dae-Ha Park, Doo-Kwon Baik
Software System Lab. Dept. of Computer Science and Engineering, Korea University

요 약

기존의 네트워크 환경의 서비스와 달리 OSGi 프레임워크 환경에서의 서비스는 생명 주기에 따라 동적이고, 다른 서비스와의 상호 작용 등이 일어나기 때문에 거짓 오퍼레이터(OP)에 의해 인증되지 않은 악의적인 서비스가 배치될 수 있고 또한 서비스가 변질 될 수도 있다. 그러므로, 개방형 서비스 게이트웨이(OSG)를 시동하였을 때 초기화하는 과정인 부트스트랩핑(Bootstrapping) 단계에서 공유 비밀키(shared secret)를 생성, 이로부터 유도된 MAC(Message Authentication Codes) Key를 기반으로한 서비스 번들 인증을 통하여 안전한 서비스를 제공받을 수 있어야 한다. 본 논문에서는 서비스 번들을 인증하기 위한 키 교환 방법과 서비스 인증 메커니즘을 제시하였다. 대칭키, 공개키의 키 교환 메커니즘은 대칭키를 이용한 키 교환 메커니즘이 공개키를 이용한 것보다 더 효율적이며, 이를 이용한 MAC 기반 서비스 번들 인증 또한 기존의 PKI 기반 서비스 번들 인증보다 인증 속도가 빠르다.

1. 서론

OSGi(Open Services Gateway Initiative)는 15개 회사가 미들웨어와 응용 프로그램간의 API를 정의하는 것을 목적으로 시작한 그룹이다.

OSGi 서비스 프레임워크(service framework)[1]는 Java 프로그래밍 언어의 플랫폼 독립성과 동적 코드 로딩 능력을 이용하여 소형 메모리 디바이스에 적합한 어플리케이션을 쉽게 개발하고 동적으로 배치할 수 있도록 한다. 또한 번들(bundle)이라는 자체 설치(self installable)가 가능한 컴포넌트 형태로 어플리케이션을 분할할 수 있도록 생명주기(life-cycle) 관리가 가능하여 전체 시스템에 영향을 주지 않고 새로운 서비스를 추가하거나 갱신할 수 있게 해준다

기존의 네트워크 서비스는 서비스를 제공하는 서버와 제공받는 클라이언트 사이의 안전한 전송 채널 하의 정적인 환경에서 컴포넌트 형태로 제공된다. 그러나, OSGi 프레임워크 환경하에서의 번들에 포함된 서비스는 게이트웨이 관리자(또는 오퍼레이터, operator, OP)와 생명주기에 따라서 동적으로 서비스 게이트웨이(Open Service Gateway, OSG)에 배치되며 다른 번들의 서비스와도 상호 작용한다. 그렇기 때문에 실제 OP가 아닌 거짓 OP에 의해 인증되지 않는 악의적인 서비스가 배치될 수도 있으며, 서비스가 변질 될 수도 있다.

그래서, OSG가 시동하면서, 장치의 인식하고 초기설정을 하는 초기화 작업인 OSGi 부트스트래핑(bootstrapping) 단계에서의 OP와 OSG는 OSGi 프레임워크 환경에서 발생할 수 있는 이러한 보안상의 문제점을 인식하여 이를 해결하고 인증 후의 안전한 서비스 번들의 전송을 위한 메커니즘이 필요하다.

본 논문에서는 OSGi 부트스트래핑 단계에서의 OSG와 OP와의 안전한 인증을 위한 2가지 인증 메커니즘을 제시하고, 인증 후의 안전한 서비스 번들 전송을 위한 전송 메커니즘을 제시하였다.

2. 연구 배경

2.1 OSGi 서비스 프레임워크

그림 1은 전체적인 OSGi 서비스 프레임워크 아키텍쳐를 나타내고 있다. OSGi는 특정 기능을 수행하는 자바 인터페이스와 실제 구현 객체인 서비스(service), 서비스를 제공하기 위한 기능적 배치단위인 번들(bundle), 프레임워크 내에서 번들들의 실행 환경인 번들 문맥(bundle context)로 구성되어 있다.

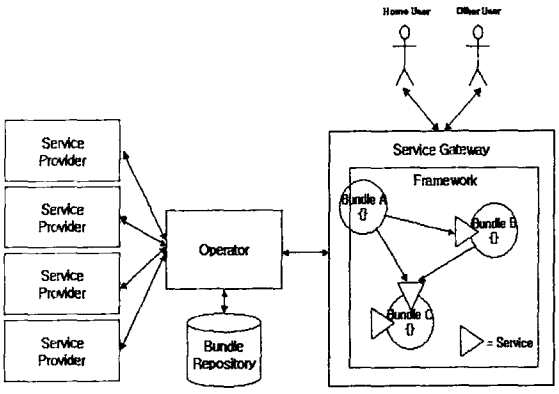


그림 1. OSGi 서비스 프레임워크 아키텍쳐

서비스는 미리 정의된 서비스 인터페이스를 통해 접근이 가능한 독립적인 컴포넌트다. 하나의 애플리케이션은 여러 개의 서비스의 협동작업을 통해 구성되고, 런타임시에 필요한 서비스를 요청할 수도 있다.

프레임워크는 각 서비스와 그 서비스에 해당하는 실제구현에 대한 매핑을 가지고 있고, 각 서비스간의 상호 의존관계를 관리한다.

번들은 여러 서비스의 구현을 하나의 패키지로 묶은 JAR 파일의 형태로 존재하며 프레임워크 내부에서 설치되거나 활성화될 때 하나의 기능적인 컴포넌트를 나타낸다.

2.2 OSGi 보안 아키텍쳐

OSGi 보안 모델은 공개키 기반구조(PKI : Public Key infrastructure) 솔루션에 기반을 두고 있다[2]. 인증 기관은 모든 OSGi 개체에 인증서를 전달하는 책임을 지게 된다. 인증서와 연관된 키는 인증, 무결성, 기밀성에 사용된다.

각각의 OSG 개체는 특정 OP 의 PKI 중 일부로서 개체를 식별하기 위하여 단 하나의 키와 인증서를 갖는다. 서로 다른

OSGi 개체 간의 모든 타입의 전송에 인증을 권고한다. 권한 부여(authorization)는 필수사항이며, 이전 인증의 결과를 기반으로 수행되어야 한다. 또한 권한부여는 각각의 요청마다 수행되어야 한다.

기밀성은 선택사항이며, 전송자의 임의대로 사용한다. 그러나 안전하지 않는 통신 채널을 거치며 보안 침해에 사용될 수 있는 정보는 원하는 수신자만 읽어볼 수 있도록 항상 암호화되어야 한다.

부트스트래핑 단계에서의 번들의 인증은 번들을 다운로드하고 수용하기 이전에 OP는 특정 OSG를 인증해야 한다. OSGi 프레임워크에서의 부트스트랩 프로세스는 현재 다른 작업 그룹에서 개발 중이며 아직 최종적으로 결정되지 않는 상태이다.

2.3 인증(Authentication)

인증이란, 메시지 혹은 사용자의 신뢰성을 결정하는 과정이다. 사용자나 메시지의 신원을 입증하는데 쓰인다[3][4]

- 메시지 인증 코드(Message Authentication codes)

메시지 인증 코드는 키와 함께 생성된 메시지 다이제스트이다. 보내고 받는 쪽 모두 동일한 대칭키가 있어야 한다. 가장 흔한 MAC 알고리즘은 HMAC(Hashed MAC)이다.

- 전자 인증서(Digital Certificates)

전자서명은 개인키로 계산하고, 공개키로 검증한다. 서명해야 하는 메시지의 메시지 다이제스트를 생성하고, 자신의 개인키로 암호화하면 된다.

공개키와 전자 서명 이외에도 인증서가 생성된 호스트 이름의 정보를 가지고 있을 수 있다.

인증서 체인(certificate chaining)은 다른 공개 키 인증서가 있는 개인키로 서명하는 것이다. 이런 식으로 해서 한 인증서 승인 기관은 루트 CA라고 불리는 다른 인증서 기관에 의해 인가된다.

2.4 Signed JAR 파일

그림 2는 서명 JAR(singed JAR) 파일의 구조[5]를 나타낸 것이다. 그림에서 보는 것과 같이 Singed JAR 파일은 기존의 JAR 파일에 서명 파일(signature file)과 서명 블록 파일(signature block file)을 가지고 있다. 서명 파일은 manifest 파일에 목록에 대응하는 목록을 가지고 있으며, 각 manifest 파일 목록에 해당하는 다이제스트 값을 가지고 있다. 서명 블럭 파일은 JAR 파일을 검증하기 위한 디지털 서명(digital signature)과 인증서(certificate)를 포함하고 있다.

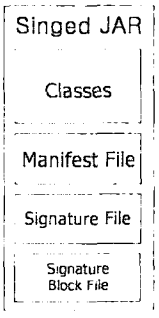


그림 2. Singed JAR 파일구조

3. OSGi 서비스 번들의 인증 메커니즘

OSGi 프레임워크 환경에서의 서비스 번들 인증은 번들이 SP에서 OP로 등록하는 때와, OP에서 OSG로의 번들 설치시로 나누어 볼 수 있다. 그림 3은 SP, OP, OSG 사이의 번들 퍼미션 할당에 대한 전체적인 구조다. 그림에서 보면, OP는 SP로 전송 받은 번들에 대해 인증이 이루어지고, OSG에서는 실제 서비스가 실행되는 시기에 인증이 이루어짐을 알 수 있다.

그림 4는 이 논문에서 제시하는 서비스 번들 인증을 위해서 OSG와 OP간의 부트스트래핑 단계에서의 키 교환 메커니즘으로써 OSG와 OP에서 생성한 각각의 random nonce (Nsg,Nop)을 공개키 또는 대칭키를 이용하여 암호화, 복호화 할 수 있다.

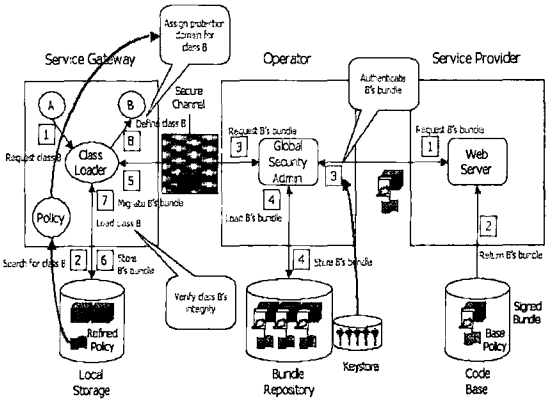
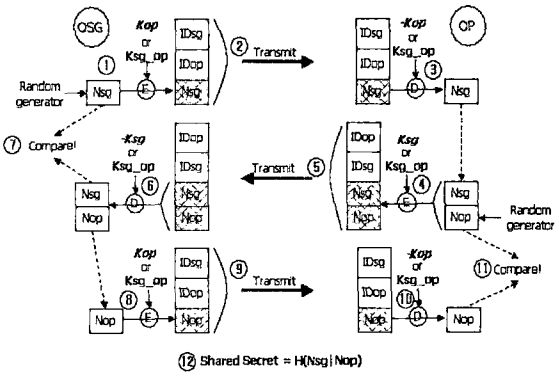


그림 3. 번들 퍼미션 할당



IDsg : OSG의 id
IDop : OP의 id
Nsg : OSG가 생성한 random nonce
IDsg : OSG의 id
IDop : OP의 id
Nsg : OSG가 생성한 random nonce
Nop : OP가 생성한 random nonce
Kop, -Kop : OP의 공개키, 개인키
Ksg, -Ksg : OSG의 공개키, 개인키
Ksg_op : OSG, OP가 공유하고 있는 대칭키

그림 4. 부트스트래핑 단계에서의 키 교환 메커니즘

3.1 키 교환 메커니즘

3.1.1 공개키를 이용한 키 교환

공개키를 이용한 키 교환 메커니즘을 자세히 설명하면 다음과 같다.

① OSG의 Random generator에 의해 random nonce 인 Nsg를 생성한다.
② 생성된 Nsg를 OP의 공개키(Kop)로 암호화하고 IDsg, IDop와 함께 OP로 전송한다.
③ OP에서는 OP의 개인키(-Kop)로 Nsg를 복호화한다.
④ 복호화한 Nsg와 OP에서 생성한 Nop를 함께 OSG의 공개키(Ksg)로 암호화한다.
⑤ 암호화된 [Nsg,Nop]를 IDop, IDsg 와 함께 OSG로 전송한다.
⑥ 전송 받은 [Nsg,Nop]를 OSG의 개인키(-Ksg)로 복호화한다.

⑦ 자신의 Nsg와 전송 받은 Nsg와 같은지 비교하여 일치하면 OP가 인증되고 만약 일치하지 않으면 더 이상의 처리는 이루어지지 않는다.
⑧ 전송 받은 Nop를 OP의 개인키(Kop)로 암호화한다.
⑨ 암호화한 Nop와 IDsg, IDop를 함께 OP로 전송한다.
⑩ 전송 받은 Nop를 OP의 개인키로 복호화한다.
⑪ 복호화한 Nop를 자신의 Nop와 비교하여 일치하면 OSG가 인증되고 결국 상호 인증이 이루어진다. 만약 일치하지 않으면 더 이상의 처리는 이루어지지 않는다.
⑫ OP, OSG가 서로 인증되면 그 결과로 공유 비밀키(shared secret)가 생성된다.

3.1.2 대칭키를 이용한 키 교환

대칭키를 이용한 키 교환은 공개키를 이용한 키 교환과 거의 비슷하나 OSG와 OP에서 생성하는 random nonce (Nsg, Nop)를 암호, 복호화 할 때 OSG와 OP가 공통으로 가지고 있는 대칭키(Ksg_op)를 가지고 한다.

3.1.3 두 가지 키 교환 메커니즘의 비교

공개키는 대칭키에 비해 키 관리 및 분배, 갱신이 쉽지만 OSG, OP가 각각 다른 키를 가지고 있어서 암호, 복호 속도가 느리고 키 길이가 길어 전송 속도가 느리다. 또한, 인증서 자체 검증과, CA(Certification Authority,인증기관)연동 시에 시간이 많이 걸리며 연산 로드가 많이 걸려 자원이 많이 소모된다. 반면에 대칭키는 암호, 복호화 속도가 빠르며, 키 길이가 짧아 전송 속도가 빠르다. 따라서, 연산의 속도나 네트워크 상에 전송되어야 하는 데이터의 크기를 고려해 볼 때, 제한적인 자원을 가지고 있는 OSGi 프레임워크 환경하에서는 대칭키를 적용한 키 교환이 더 효율적이다.

3.2 서비스 번들의 인증 메커니즘

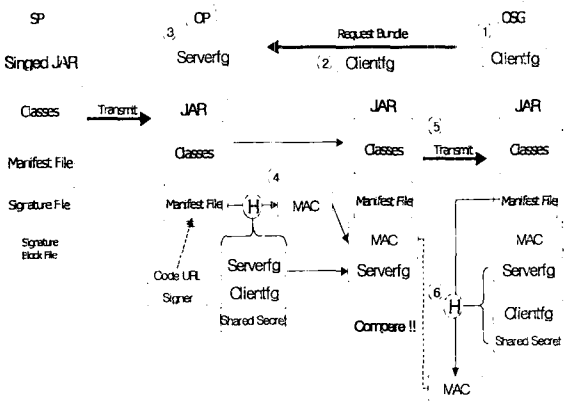
3.2.1 PKI 기반 서비스 번들 인증

기존의 OSGi 서비스 번들은 PKI 기반으로 OP가 서비스 번들을 검증하지 않고 SP의 서비스 번들이 OSG에게로 바로 전달된다. 서명된 JAR 파일을 통한 인증이 이루어지며 서명된 JAR 파일 안에는 인증에 필요한 각종 정보들이 담겨져 있다.

3.2.2 MAC 기반 서비스 번들 인증

MAC 기반 서비스 번들 인증은 부트스트래핑 단계에서 생성된 공유 비밀키(shared secret)로부터 유도된 MAC을 인증의 기반으로 삼는다. 정상적인 HTTP의 Get 응답 / 요청 (Response / Request) 에 기반을 두며 HMAC 계산의 입력으로 OSG와 OP에서의 임의의 랜덤값인 Clientfg, Servefg와 Menifest 파일 그리고 공유 비밀키를 사용한다. 좀더 자세한 과정을 보면 다음과 같다.

① OSG는 Clientfg라는 하나의 nonce를 생성하여 저장한다.
② OSG는 Clientfg를 OP에 전송한다.
③ OP는 Serverfg라는 하나의 nonce를 생성한다.
④ OP는 HMAC 함수와 공유 비밀키 , Clientfg, Serverfg, Menifest를 기반으로 MAC을 계산한다.
⑤ OP는 serverfg와 MAC 및 서비스 번들을 포함하는 응답을 OSG에 전송한다.
⑥ OSG는 OP와 동일한 방식으로 MAC 값을 계산하고 수신한 MAC 값과 일치하는지 검사한다. 일치하면 OSG는 OP에서 전달된 서비스 번들을 믿고 사용할 수 있다. 만약 일치하지 않으면 더 이상의 처리는 이루어지지 않는다. 계산에는 수신한 응답 내부의 Serverfg 값이 사용된다.



3.2.3 서비스 번들 인증 메커니즘 비교

기존 PKI 기반의 OSGi 서비스 번들 인증은 특정 SP의 정보, SP의 공개키와 CA의 서명이 들어있는 인증서를 포함하고 여러 사람의 서명이 가능하지만 이러한 이유로 이동하는 번들의 사이즈가 커져 전송 속도 및 인증 속도가 느리다. 반면, MAC 기반 서비스 번들 인증은 OP가 검증한 번들을 인증하므로 기존 서명 JAR 파일의 서명 부분을 제외하고 Menifest 파일을 붙여 전송이 이루어지므로 서명 크기가 작고 인증이 빠르다. 또한, 서비스 번들 요청 시마다 MAC 키가 생성되므로 키의 신선도 및 보안을 높일 수 있다. 그러나, OP가 모든 것에 관여를 하므로 OP에 대한 보안이 필요하며, 대책으로는 방화벽(firewall)을 설치하거나, MAC 키에 대한 암호, 복호 처리를 할 수 있다.

4. 결론 및 향후 과제

본 논문에서는 OSGi 프레임워크 환경에서의 부트스트래핑 단계에서 공개키, 대칭키를 이용한 키 교환 방법과 이를 이용한 서비스 번들의 인증 메커니즘을 제시하였다. 향후에는 OSGi 스펙에 맞는 적절한 구현이 이루어져야겠고, 서비스 번들의 안전한 이동 채널과 OSGi 서비스 프레임워크 환경에서의 사용자 인증에 대한 연구가 필요하다.

5. Reference

[1] OSGi, "OSGi Service Gateway Specification - Release 2.0" http://www.osgi.org, 2001
[2] OSGi, "RFC 18 - Security Architecture Specification" Draft, http://www.osgi.org/member, 2001
[3] Jess Garms, Daniel Somerfield, "Professional Java Security", WROX, 2001
[4] M. Pistoia, et al. "Java 2 Network Security", Second edition, Prentice Hall, 1999.
[5] Sun, JAR Feature , http://java.sun.com/j2se/1.4/docs/guide/jar/, 2001
[6] OSGi, "Secure Provisioning Data Transport using Http", Confidential, Draft, RFC36, http://www.osgi.org/ , 2002