OSGi 플랫폼에서 서비스 인젝션 공격 및 대응책

Preventing Service Injection Attack on OSGi Platform

김인태*, 정경용**, 임기욱***, 이정현*
인하대학교 정보공학과*, 상지대학교 컴퓨터정보공학부**, 선문대학교 컴퓨터정보학부***

In-Tae Kim(inking007@gmail.com)*, Kyung-Yong Chung(cyjung@sangji.ac.kr)**,
Kee-Wook Rim(rim@sunmoon.ac.kr)***, Jung-Hyun Lee(jhlee@inha.ac.kr)*

요약

OSGi 플랫폼은 자바 기반의 컴포넌트 플랫폼으로써 애플리케이션 개발 환경에서부터 엔터프라이즈 소프트웨어까지 다양한 분야에 사용되고 있다. OSGi 플랫폼은 개방형 소프트웨어 환경에서 컴포넌트를 동적으로 설치하고 갱신하기 위한 기능을 제공한다. 그러나 이러한 환경으로 인해 새로운 보안상 취약점이 발생하고 있으며 최근 보안문제를 해결하기 위한 연구들이 이루어지고 있다. OSGi 플랫폼에서 취약점은 자바의 멀티 애플리케이션 플랫폼의 한계로 인한 취약점과 OSGi 플랫폼 자체의 취약점으로 분류할 수 있다. 우리는 OSGi 플랫폼 자체의 취약점 중에서 서비스 인젝션에 관한 취약점을 식별하고 이를 방어할 수 있는 매커니즘을 제안한다. 또한 오픈 소스중 하나인 Knopflerfish를 이용하여 제안된 매커니즘을 구현하고 다른 오픈소스들과 비교 평가한다.

■ 중심어 : | OSGi | 서비스 인젝션 | 보안 |

Abstract

The OSGi platform is a Java-based component platform that is being widely used from environments for the application development to enterprise software. The OSGi platform provides dynamic and transparent installation for open environments. However, it open new attacks so that many researches try to solve OSGi vulnerability. Security flaws in OSGi platform are categorized two parts: the JVM and the OSGi platform itself. We focus on vulnerability by OSGi platform itself, particularly service injection. We identify the service injection attack and suggest secure mechanisms to prevent the attack. Those mechanisms are implemented, providing a few modification to the Knopflerfish OSGi implementation and are evaluated through comparing with existing mechanisms.

■ keyword : | OSGi | Service injection | Security |

I. 서 론

OSGi[1] 플랫폼은 자바 기반의 컴포넌트 플랫폼으로써 초창기 홈 게이트웨이와 셋톱박스처럼 리소스가 제한된 장치를 위한 플랫폼으로 디자인되었다. 그러나 최근에는 Eclipse[3]와 같은 통합 개발 환경에서부터 엔터

* "본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음" (NIPA-2010-C1090-1031-0004)

접수번호 : #100517-001
접수일자 : 2010년 05월 17일
심사완료일 : 2010년 07월 13일
교신저자 : 김인태, e-mail : inking007@gmail.com

프라이즈 소프트웨어 [10]까지 다양한 분야에 사용되고 있다.

오늘날, 소프트웨어의 배포는 통합된 패키지 형태가 아닌 제 3자에 의해 제작되고 배포된 다양한 서비스를 설치하여 사용하는 형태로 변해가고 있다 [2]. OSGi 플랫폼은 실행 중에 코드 의존성을 해결하고 실행 중지 없이 컴포넌트를 설치하고 갱신할 수 있는 특성을 가지고 있어서 이러한 소프트웨어의 배포 및 관리 형태에서 중요한 역할을 수행할 수 있다.

그러나 제 3자에 의해 제작된 소프트웨어가 동적으로 설치되고 갱신되는 환경으로 인해 새로운 보안상 취약점이 발생될 수 있다. OSGi 플랫폼에서 발생할 수 있는 취약점은 자바의 멀티 애플리케이션 플랫폼의 한계로 인한 취약점과 OSGi 플랫폼 자체의 취약점으로 분류할 수 있다. 자바의 멀티 애플리케이션 플랫폼 한계로 인한 보안 문제는 CPU 또는 메모리 등을 고갈시키는 서비스 거부 공격 및 정적 변수를 변경하는 공격에 대한 취약점으로써 자바 리소스를 프로세스별로 고립하는 방식으로 해결하고 있다 [6][7]. 최근 연구에서 현재 구현되어 있는 오픈 소스 OSGi 플랫폼을 분석하여 25개의 취약점을 분류하고 17개의 OSGi 플랫폼 자체의 취약점들에 대한 해결책을 제시하고 있다 [4]. 다른 연구에서는 취약한 컴포넌트의 무결성을 보장하는 방법으로 전자서명을 사용하는 방법을 제안하고 있다 [5]. 본 논문에서는 [4]에 기술되지 않은 OSGi 플랫폼 취약점 중 컴포넌트를 갱신할 때 발생할 수 있는 서비스 인젝션 공격을 식별하고 이를 방어할 수 있는 매커니즘을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 OSGi 플랫폼의 기본 구조 및 보안 취약점에 대해서 기술하고 3장에서는 OSGi 플랫폼에서 발생될 수 새로운 취약점을 식별하고 이를 보호하기 위한 매커니즘을 제안한다. 4장에서는 오픈 소스 OSGi 중 하나인 Knopflerfish[8]를 이용하여 제안된 매커니즘을 구현하여 다른 오픈소스들과 비교 평가한다. 5장에서는 결론을 맺는다.

II. 관련 연구

본 장에서는 OSGi 플랫폼의 모듈화 및 서비스에 대하여 기술하고 현재 알려진 OSGi 플랫폼 자체 취약점들에 대해 기술한다.

1. OSGi 플랫폼

OSGi 플랫폼은 개념적으로 모듈계층, 라이프사이클 계층, 서비스 계층으로 나눌 수 있다. 모듈 계층은 번들이라 불리는 컴포넌트를 정의하고 컴포넌트들끼리 공유와 제약을 가능케 한다. 라이프사이클 계층은 번들의 라이프 사이클을 정의한다. 서비스계층은 서비스를 자바 인터페이스 형태로 등록하고 LDAP[11] 요청 형식으로 검색할 수 있게 한다.

OSGi 번들은 자바클래스파일, 메타 파일 (META-INF/MANIFEST.MF), 리소스 파일등을 포함한 JAR 파일 [12] 형태이다. 메타파일에는 모듈화를 위한 다음과 같은 헤더정보를 포함하여 번들들간 공유 및 접근제한을 설정할 수 있다.

- Export-Package: 다른 번들에게 자신의 패키지를 공유하기 위해 정의한다.
- Import-Package:다른 번들에서 공유한 패키지를 사용하기 위해 정의한다.

라이프 사이클 계층에서 정의하고 있는 번들의 라이프 사이클은 [그림 1]에 나타나 있다.

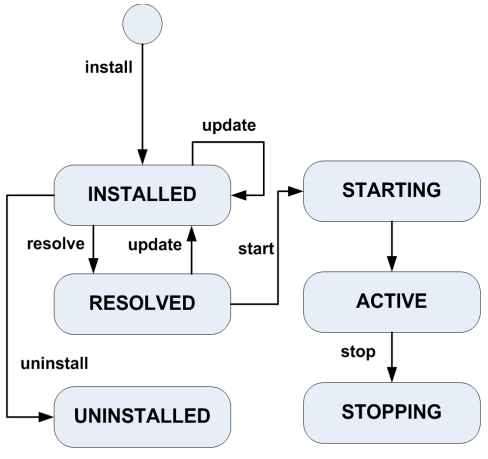


그림 1. 번들 갱신 전 서비스 호출.

메타 파일에 번들 라이프 사이클을 후킹하기 위한 클

래스를 Bundle-Activator 헤더에 정의한다. 이 클래스를 일명 Activator 클래스라고 부른다. RESOLVED 상태에서 STARTING 상태로 전이되면서 Activator 클래스의 start() 메소드가 호출되고 start() 메소드가 정상적으로 실행되면 ACTIVE 상태로 전이된다. 번들이 STOP되면 Activator 클래스의 stop() 메소드가 호출된다.

2. OSGi 보안 취약점 및 대응방법

[4]에서 현재 구현되어 있는 오픈소스 OSGi에서 발생할 수 있는 다양한 취약점에 대해서 분류하고 대응책에 대해 기술하고 있다. [표 1]은 악의적인 번들에 의해 발생 할 수 있는 공격 유형들을 보여주고 있다.

표 1. 악의적인 번들에 의해 발생 할 수 있는 공격 유형

공격명	공격 위치
번들 크기	번들 Archive
패키지 중복	Manifest
Activator 블럭	Activator
플팻폼 종료	애플리케이션 코드 (네이티브 코드)
무한 반복문	애플리케이션 코드 (자바 코드)
반복적 쓰레드생성	애플리케이션 코드 (자바 API)
과중한 서비스 등록	애플리케이션 코드 (OSGi API)
번들 조각 교체	번들 조각

각 계층의 대표적인 취약점에 대한 대응책을 살펴보면 다음과 같다.

1) 패키지를 중복되게 임포트하여 정상적인 번들 설치를 거부하게 만드는 공격
   대응책: 중복된 임포트를 무시하여 방어한다.
2) 파일 크기가 큰 번들을 설치하여 디스크 용량을 가득 채우는 공격
   대응책: 번들 다운로드하기 전 번들의 크기를 제한하여 방어한다.
3) 번들이 시작할 때 Activator 클래스의 start() 메소드에 무한 반복문을 수행하여 번들 관리 유틸리티를 동작하지 않게 하는 공격
   대응책: 번들 관리 유틸리티에서 명령을 수행하는 부분을 쓰레드로 작성하여 방어한다.
4) 많은 서비스를 등록해서 정상적인 서비스 등록을 막는 공격
   대응책: 번들별로 서비스 등록 횟수에 제한을 두어 방어한다.

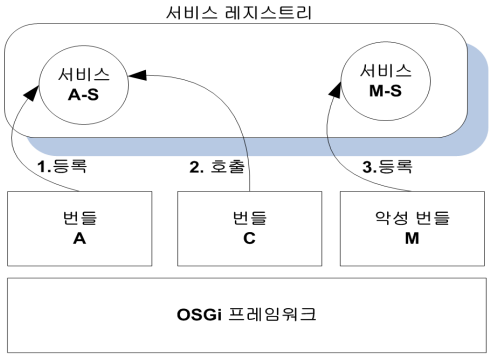
III. 새로운 취약점 및 대응방법

1. 서비스 인젝션 취약점

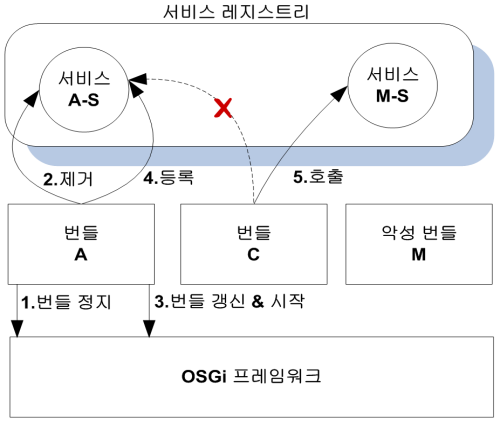
OSGi 플랫폼은 서비스 실행 중에 서비스중지 없이 번들을 갱신하는 매커니즘을 가지고 있다. 이러한 매커니즘은 OSGi 플랫폼의 최대 장점 중에 하나이지만 악의적인 서비스가 인젝션 될 수 있는 문제점을 가지고 있다. 우리는 공격자 번들이 다른 번들의 라이프 사이클을 간섭할 수 있는 권한, 즉 번들 매니저 권한을 가지고 있지 않다고 가정한다. 다른 번들의 라이프사이클을 조작하여 공격하는 유형은 자바 퍼미션을 통해 막을 수 있다. 우리는 악의적인 번들이 설령 번들 매니저의 권한을 가지고 있지 않더라도 일어날 수 있는 서비스 인젝션 공격에 초점을 두고 있다.

[그림 2]는 서비스 인젝션이 일어나는 예를 보여주고 있다. [그림 2-a]는 번들이 갱신되기 전 정상적인 서비스 상태를 보여주고 있다. 정상적인 번들A는 서비스 A-S를 등록하고 번들 C는 서비스 A-S를 호출하여 이용하고 있다. 악의적인 번들 M은 서비스 A-S 서비스를 교체할 악의적인 서비스 M-S를 등록한다.

[그림 2-b]는 번들 갱신 후 정상적인 서비스가 악의적인 서비스와 교체되는 것을 보여주고 있다. 번들 A의 새로운 버전을 설치하기 위해 갱신 요청받는다. 번들이 갱신되는 동안 서비스 A-S를 잠시 중지하고 이전 번들의 JAR 파일을 새로운 JAR 파일로 교체한 후 서비스 A-S 룰 다시 등록한다. 서비스 A-S를 이용하던 호출자들은 번들 A가 갱신되는 동안 서비스 레퍼런스가 초기화된 것을 확인하고 다시 서비스 레퍼런스 검색을 시도한다. 이때 바인딩 되는 서비스는 번들 M에 의해 등록된 서비스 M-S가 되고 이후 사용되는 서비스는 A-S가 아니라 M-S가 된다.



a) 번들 갱신 전 서비스 호출



b) 번들 갱신 후 서비스 호출

그림 2. OSGi에서 서비스 인젝션

[그림 2]에서 살펴본 것과 같이 번들 갱신 중에 정상적인 서비스가 악의적인 서비스로 교체되는 공격이 일어날 수 있다. 우리는 이러한 서비스 인젝션 공격을 방어하기 위한 방법을 제안한다.

2. 번들의 전자서명을 통해 보호하기

기존연구에서는 번들에 전자서명을 함으로써 번들의 무결성만을 보장하고 있다. 그러나 우리는 전자 서명된 번들을 통해 서비스의 인젝션 공격을 방어하기 위한 강화된 서비스 등록 및 검색 방법을 제안한다.

OSGi 플랫폼에서는 서비스 등록을 위해 BundleContext 클래스에 다음과 같은 API를 정의하고 있다.

registerService(Service Name, Service Object, Filter);

서비스 레지스트리의 등록과정은 다음과 같이 동작한다. registerService 메소드의 두 번째 인수인 서비스 객체 대한 클래스 타입을 확인한 후 이미 등록되어 있는 클래스 타입이 있는지 확인한다. 있다면 해당 타입의 서비스로 추가 등록하고 없다면 새로운 타입의 서비스로 생성하여 등록한다. 서비스 인젝션 공격을 막기 위하여 이러한 등록 과정을 다음과 같이 수정한다.

첫째, 전자서명된 번들을 설치할 때 번들의 무결성을 보장하기 위하여 유효성 검사를 한다.

둘째, 유효성 검사가 정상적이라면 번들 관리자는 해당 서명정보를 번들 컨텍스트 정보에 포함시킨다.

셋째, 해당 번들로 부터 서비스 등록 요청을 받으면 전자서명에 쓰인 정보를 검색 필터링으로 포함하여 등록한다.

우리는 서비스를 등록할 때 번들의 전자서명 정보를 포함시켜서 정상적인 서비스를 악의적인 번들에 의한 서비스와 구분한다. 그리고 구분된 서비스를 얻어오기 위해 API를 확장한다. 기존 OSGi에서는 서비스 객체를 얻기 위해 다음과 같은 API를 정의하고 있다.

public ServiceReference getServiceReference(String clazz)

public ServiceReference[] getServiceReferences(String clazz, String filter)

getServiceReference 메소드는 해당 클래스 타입을 구현한 서비스들 중 가장 우선순위가 높은 서비스를 얻어온다.

getServiceReferences 메소드는 해당 클래스 타입을 구현한 서비스들 중에서 필터 값으로 필터링하여 서비스들을 배열형태로 얻어온다.

우리는 전자서명된 번들이 등록한 서비스를 검색하기 위해 첫 번째 API에 필터 파라미터를 받도록 다음과 같이 확장한다.

public ServiceReference getServiceReference(String clazz, String filter)

getServiceReferences 메소드에서 사용된 필터와의 차이점은 검색대상이 되는 속성을 전자서명 속성에 제한하여 필터링한다는 것이다.

3. 전자서명되지 않은 번들의 서비스 보호하기

전자서명된 번들에 대한 서비스 인젝션 공격에 대한 보호는 서비스 등록과 검색 과정에서 전자서명 정보를 이용함으로써 이루어질 수 있다. 그러나 실제적으로 모든 번들들이 전자서명되어 사용된다고 볼 수 없기 때문에 전자서명되지 않은 번들들을 위한 추가적인 보호 매커니즘이 필요하다. 우리는 서비스 인젝션 공격을 방어하기 위한 보안이 강화된 번들 갱신 매커니즘을 제공한다.

우선 기존 번들 갱신 매커니즘을 살펴보면 다음과 같다.

1) 현재 번들이 ACTIVE 상태이면 번들을 멈춘다.
2) 메타 파일에 정의되어 있는 Bundle-UpdateLocation 헤더로 부터 URL 정보를 가져와 번들을 다운받는다.
3) 다운받은 번들의 메타파일을 검사하고 기존 번들 JAR 파일을 새로운 번들 JAR 파일로 교체한다.
4) 기존 번들에 의해 로드된 클래스들을 제거한다.
5) 기존 번들이 ACTIVE 상태였다면 새로 받은 번들을 시작한다.

기존 번들 갱신 매커니즘에서 악의적인 서비스가 인젝션 되는 시점은 번들이 멈추는 순간이다. 번들이 멈추면 프레임워크는 번들의 Activator 클래스의 stop() 메소드를 호출한다. stop() 메소드는 일반적으로 ServiceRegistration클래스의 unregister 메소드를 호출하여 등록된 서비스를 서비스레지스트리에서 삭제하도록 구현한다. 정상적인 서비스가 서비스 레지스트리에서 삭제되면 악의적인 서비스가 그 다음 우선순위를 가지고 대기 중이다. 이때 기존에 정상적인 서비스를 사용 중이던 호출자가 다시 getServiceReference 메소드를 호출하면 악의적인 서비스와 바인딩된다. 우리는 번들 라이프사이클에 UPDATING 상태를 추가하여 악의적인 서비스 인젝션 막는다. UPDATING 상태를 추가한 후 안정한 번들을 갱신 과정은 다음과 같다.

번들이 갱신되는 동안 해당 번들의 상태는 UPDATING으로 변경된다. UPDATING 상태는 서비스를 할 수 있는 상태이며 서비스를 사용하는 입장에서는 아무런 변화가 없다. 새로운 번들 JAR를 다운받아 설치 시 BundleContext에 자신의 이전 번들 정보를 포함한다. 이전 번들이 등록한 서비스들의 우선순위 정보도 포함한다. 새로 설치된 번들을 ACTIVE 상태로 전환하고 서비스들은 이전 번들의 우선순위 정보에 따라 서비스를 등록한다. 새로운 번들이 ACTIVE 상태가 되면 이전 번들은 STOP 상태로 전환한다. 이렇게 함으로써 번들이 갱신이 완료 될 때 까지는 이전 서비스를 계속 제공하다가 번들이 갱신이 완료되면 악의적인 서비스 인젝션없이 새로운 서비스로 교체될 수 있다.

변경된 번들 라이프 사이클은 [그림 3]에 나타나 있다.

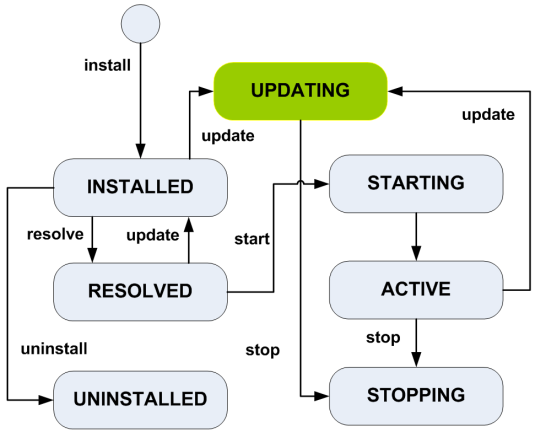


그림 3. 변경된 번들의 라이프사이클

IV. 구현 및 평가

우리는 OSGi 오픈 소스 프로젝트인 Knopflerfish를 수정하여 제안한 서비스 인젝션 방어 매커니즘을 구현하였다. 서명된 번들을 위한 서비스 인젝션 방어는 SF-Jarsigner[5]를 이용하여 적용하였다.

SF-Jarsinger는 다음과 같이 번들의 전자서명 유효성 검사를 수행한다.

1) 서명된 jar 파일에 포함된 리소스들이 순서가 맞는지 검사한다.
2) 서명 블록 파일이 유효한지 검사한다.
3) 서명 파일이 유효한지 검사한다.
4) 메타 파일이 유효한지 검사한다.

전자서명 유효성 검사를 할 때 사용되는 서명 블록파일, 메타 다이제스트, 파일 다이제스트 정보는 번들이 갱신되면 변경되는 정보이므로 서비스 유효성검사에는

불필요한 정보라고 할 수 있다. 우리는 전자서명자 정보를 공인된 인증기관에서 받은 고유한 이름이며 번들이 갱신되더라도 변하지 않는 값으로 가정한다. 그래서 우리는 전자서명자 정보를 사용하여 서비스가 동일한 서명자에 의해 서명된 번들에서 등록된 것임을 검증하는데 사용하였다.

제안된 서비스 인젝션 방법은 Knopflerfish에서 Bundle 인터페이스를 구현한 BundleImpl 클래스를 수정하여 번들 라이프사이클에 UPDATING 상태를 추가하고 번들을 갱신할 때 번들을 정지-갱신-재시작 하던 기존 과정을 재시작 후 정지하도록 수정하여 구현되었다. 기존 Services 클래스에 ArrayList와 HashMap 이용하여 구현된 서비스레지스트리에 서비스별 우선순위를 부여하기 위하여 ServiceRegistrationImpl 클래스에 priority 속성을 추가하고 Collection.sort 함수를 이용하여 정렬하였다.

우리는 오픈소스로 구현된 OSGi 프레임워크인 Equinox[3], Felix[9], Knopflerfish에 대해 번들을 갱신할 때 서비스 인젝션 공격을 방어하는지 실험하였다. 실험을 위하여 다음과 같은 4개의 번들을 작성하였다.

- Good Bundle: 서비스의 인터페스를 가지고 있는 번들이다.
- GoodImpl Bundle: 서비스를 등록하고 실제 기능 구현을 포함하고 있는 번들로써 서비스가 호출되면 'good'이라는 메시지를 출력한다.
- BadImpl Bundle: 악의적인 서비스를 등록하고 기능을 구현을 포함하고 있는 번들로써 서비스가 호출되면 'malicious' 라는 메시지를 출력한다.
- Consumer Bundle: 서비스를 이용하는 번들로써 Activator클래스의 start() 메소드에서 쓰레드를 생성하여 주기적으로 서비스를 호출한다.

번들 설치 순서는 Good, GoodImpl BadImpl, Consumer 이며 실행 순서는 GoodImpl, BadImpl, Consumer로 하였다. 모든 번들이 Active 상태가 된 후 GoodImpl 번들 갱신 명령을 수행하였다. 서비스 인젝션에 대한 보안성이 취약한지 안전한지에 대한 판단은 번들을 갱신한 후 Consumer 번들에 의해 호출되는 서비스의 출력문을 보고 판단하였다. [표 2]는 현재 널리 사용되고 있는 오픈소스 OSGi들과 본 논문에서 제안한 시스템을 비교한 것이다.

표 2. 오픈소스 OSGi들과 서비스 인젝션 취약점 비교평가

이름	출력 결과	서비스 인젝션
Equinox 3.6	malicous	취약함
Felix 2.05	malicous	취약함
Knopflerfish	malicous	취약함
제안된 시스템	good	안전함

[표 2]에서 살펴본 바와 같이 현재 구현되어 있는 OSGi 프레임워크들은 번들 갱신 시 악의적인 번들에 의한 서비스 인젝션이 발생하고 있다. 그러나 기존 연구에서는 번들이 갱신할 때 발생할 수 있는 서비스 인젝션에 대한 취약점을 도출하지 못했다. 이에 우리는 새로운 취약점에 대한 특징화 항목을 도출하여 [표 3]에 기술하였다.

표 3. 취약점 특징화

항목	내용
주체	서비스 인젝션
엔티티	악의적인 번들
취약점 위치	번들의 라이프사이클 관리
대상	갱신되는 번들의 서비스
취약점이 나타나는 시점	번들이 갱신되는 시점

[표 4]는 서비스 인젝션에 대한 취약점에 대해 OSGi 프레임워크에 보안을 강화하기 위한 기존연구들과 제안한 논문을 비교한 것이다. [4]에서는 OSGi프레임워크에서 발생할 수 있는 많은 취약점들을 발견하고 이를 해결하기 위한 보안이 강화된 OSGi 프레임워크를 제시하고 있지만 본 연구에서 도출한 새로운 취약점과 해결책에대해 기술하고 있지 않다. 또한 [5]에서는 전자서명을 사용한 번들의 무결성을 보장하고 있지만 서비스 인젝션에 대한 방어에 대해 기술하고 있지 않다.

표 4. 기존 연구들과 비교평가

이름	번들 무결성	취약점	서비스 인젝션 방어
연구 [4]	지원	미발견	지원 X
연구 [5]	지원	미발견	지원 X
제안된 시스템	지원	발견	지원

V. 결론

우리는 OSGi 플랫폼에서 번들 갱신 시점에 일어날 수 있는 서비스 인젝션 취약점을 식별하고 이를 방어하기 위한 두 가지 매커니즘을 제안하였다. 첫째는 전사 서명정보를 이용한 서비스 인젝션 방어 방법이며 둘째는 번들 주기에 UPDATING을 추가하여 번들 갱신 매커니즘을 개선하는 방법이다. 우리는 기존 오픈 소스 OSGi 플랫폼과 비교 평가하여 제안된 매커니즘이 서비스 인젝션 공격에 대해 안전함을 증명하였다.

참 고 문 헌

[1] OSGi Alliance. OSGi service platform, core specification release 4.2. release 03 2010.

[2] Y. Royon and S. Fr´enot. Multiservice home gateways: business model, execution environment, management infrastructure. IEEE Communications Magazine, Vol.45, No.10, pp.122-128, 2007(10).

[3] Equinox. http://www.eclipse.org/equinox.

[4] P. Parrend and S. Fr´enot. Security benchmarks of OSGi platforms: toward hardened OSGi. Software: Practice and Experience, Vol.39, No.5, pp.471-499, 2009(4).

[5] P. Parrend, S. Frenot, Supporting the secure deployment of OSGi Bundles. First IEEE WoWMoM Workshop on Adaptive and DependAble Mission and bUsiness Critical Mobile Systems, Helsinki, Finland, 2007.

[6] G. Czajkowski and L. Dayn‘es. Multitasking without compromise: a virtual machine evolution. In Proceedings of the Object Oriented Programming, Systems, Languages, and Applications Conference, pages 125-138, Tampa Bay, USA, October 2001. ACM.

[7] GEOFFRAY, N., THOMAS, G., MULLER, G., ET AL. I-JVM: a Java virtual machine for component isolation in OSGi. In DSN'09 (Estoril, Portugal), p.10, 2009(4).

[8] Knopflerfish OSGi - Open Source OSGi service platform. http://knopflerfish.org/

[9] Apache felix. http://felix.apache.org/site/ index.html

[10] Spring Dynamic Modules for OSGi(tm) Service Platforms http://www.springsource.org /osgi

[11] Howes T. The String Representation of LDAP Search Filters. IETF RFC, Network Working Group, Request for Comments: 2254, 1997

[12] Sun Microsystems, Inc. JAR File Specification, Sun Java Specifications, 2003.

저 자 소 개

김 인 태(In-Tae Kim) 정회원

- 1997년 2월 : 인하대학교 전자계산공학과(공학사)
- 1999년 2월 : 인하대학교 전자계산공학과(공학석사)
- 2005년 3월 ~ 현재 : 인하대학교 컴퓨터 정보공학과 박사과정

<관심분야> : 유비쿼터스 컴퓨팅 보안, 클라우드 컴퓨팅 보안

정 경 용(Kyung-Yong Chung) 정회원

- 2000년 2월 : 인하대학교 전자계산공학과(공학사)
- 2002년 2월 : 인하대학교 컴퓨터정보공학과(공학석사)
- 2005년 8월 : 인하대학교 컴퓨터정보공학과(공학박사)
- 2005년 9월 ~ 2006년 2월 : 한세대학교 IT학부 교수
- 2006년 3월 ~ 현재 : 상지대학교 컴퓨터정보공학부 교수

<관심분야> : 유비쿼터스 컴퓨팅, 인공지능시스템, 데이터마이닝, U-CRM

임 기 욱(Kee-Wook Rim) 정회원

- 1977년 2월 : 인하대학교 전자공학과(공학사)
- 1987년 2월 : 한양대학교 전자계산학(공학석사)
- 1994년 8월 : 인하대학교 전자계산학(공학박사)
- 1977년 ~ 1988년 : 한국전자통신연구소 시스템소프트웨어 연구실장
- 1989년 10월 ~ 1996년 12월 : 한국전자통신연구원 시스템연구부장, 주전산기(타이컴)Ⅲ,Ⅳ 개발사업 책임자
- 2001년 7월 ~ 1999년 12월 : 한국전자통신연구원 컴퓨터소프트웨어 연구소장
- 2000년 ~ 현재 : 선문대학교 컴퓨터정보학부 교수

<관심분야> : 실시간데이터베이스시스템, 운영체제, 시스템구조

이 정 현(Jung-Hyun Lee) 정회원

- 1977년 2월 : 인하대학교 전자과(공학사)
- 1980년 9월 : 인하대학교 전자공학과(공학석사)
- 1988년 2월 : 인하대학교 전자공학과(공학박사)
- 1979년 ~ 1981년 : 한국전자기술연구소 연구원
- 1984년 ~ 1989년 : 경기대학교 전자계산학과 교수
- 1989년 1월 ~ 현재 : 인하대학교 컴퓨터공학부 교수

<관심분야> : 자연어처리, HCI, 음성인식, 정보검색, 고성능 컴퓨터구조