情報保護學會論文誌
第17卷 第6號, 2007. 12

WIPI에 적합한 Specification 기반의 침입탐지시스템

김 익 재†, 이 수 진‡
국방대학교

Specification-based Intrusion Detection System for WIPI

Ik-jae Kim†, Soojin Lee‡
National Defense University

요 약

본 논문에서는 국내 무선인터넷 표준 플랫폼 규격인 WIPI(Wireless Internet Platform for Interoperability)에 적합한 Specification 기반의 침입탐지시스템을 제안한다. 시스템을 제안함에 있어서는 경량의 단순화된 코드, C와 자바 등의 다양한 언어 수용성 그리고 하드웨어 독립성을 고려하였다. 제안한 침입탐지시스템은 WIPI HAL(Handset Adaptation Layer)의 주요 API를 대상으로 침입을 탐지하는 알고리즘에 기반하고 있으며, HAL API에 추가 가능한 형태의 mIDS(mobile Intrusion Detection System) API 그룹을 프로토타입으로 정의하였고, 자바 라이브러리와 WIPI 에뮬레이터를 이용한 시뮬레이션을 통해 적용 가능성을 증명하였다.

ABSTRACT

In this paper, we propose a specification-based intrusion detection system for WIPI(Wireless Internet Platform for Interoperability). In proposing the system, we focused on providing lightweight code, supporting multiple languages and hardware independence. The proposed system is based on an algorithm which detects an intrusion to main API of WIPI-HAL(Handset Adaptation Layer) and defines the prototype of mIDS(mobile IDS) API group that it can be added on the HAL. Moreover, we prove apply possibility through a WIPI emulator using java library.

Keywords : WIPI, Intrusion Detection System(IDS), HAL, mIDS, Mobile, Handset Device

Ⅰ. 서 론

WIPI는 한국 무선인터넷 표준화 포럼(Korea Wireless Internet Standardization Forum : KWISF)에서 제정한 한국 무선인터넷 표준 플랫폼으로 이동통신 단말기에 탑재되어 응용프로그램을 수행할 수 있는 환경을 제공한다. WIPI는 2001년도에 LGT, SKT, KTF 등 국내 이동통신사가 공동으로 플랫폼 요구사항을 도출하여 기존에 서비스되고 있던 다양한 플랫폼들(BREW, MAP, GVM, SK-VM, KVM)의 장점을 수용하고, 차세대 서비스를 위해 필요한 기능 추가와 기존 플랫폼의 단점을 보완하였다. 이러한 WIPI는 응용프로그램 개발자에게는 플랫폼 간 콘텐츠 호환성을 보장하고, 단말기 개발자에게는 플랫폼 이식의 용이성을 제공하며, 일반 사용자에게는 다양하고 풍부한 콘텐츠를 제공하게 되었다.

그러나 이러한 무선인터넷 플랫폼의 표준은 반드시 순기능만 있는 것은 아니다. 역기능으로는 표준 플랫폼이 제정됨에 따라 악의적 사용자의 공격 방향이 WIPI로 단일화되었다는 점을 들 수 있다. 이미 2000년에 모

접수일: 2007년 9월 20일; 채택일: 2007년 10월 30일
† 주저자, nodo22@hanmail.net
‡ 교신저자, cyberkma@kndu.ac.kr

바일 장비를 공격하는 트로이 목마가 발견되었고, 휴대용 단말기의 고기능화에 따라 운영체제 및 응용프로그램의 취약점이 점차 증대되어 악성코드가 더욱 기승을 부릴 것으로 전망되고 있는 가운데 WIPI에 대한 보안 필요성은 사회적으로 더욱 증가하고 있는 추세라고 할 수 있다[1].

따라서 악성코드를 탐지하여 사용자를 보호할 수 있는 보다 적극적인 방법이 필요하다고 할 수 있다. 본 논문에서는 WIPI에 적합한 Specification 기반의 침입탐지시스템을 제안하고, WIPI HAL에 추가할 수 있도록 설계된 mIDS API의 프로토타입을 정의함에 있어서 WIPI HAL 표준 규격에 추가 가능한 형태를 유지하였다. 그러나, HAL은 블랙박스화 되어 있어 일반 개발자가 접근할 수 없는 영역이므로 제안한 시스템에 대한 적용 가능성은 구현 모델을 별도로 제시하고, 자바 라이브러리 형태로 탐지 로직을 구현하여 가상의 공격 시나리오를 통해 WIPI 에뮬레이터 기반의 시뮬레이션을 수행했다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 침입탐지 시스템의 침입탐지 기법을 고찰하고, WIPI의 개념과 구조를 분석한다. Ⅲ장에서는 침입탐지 알고리즘을 도출하고, WIPI HAL에 추가 가능한 mIDS API 그룹을 제안하며, 제안된 API에 대한 프로토타입을 정의한다. Ⅳ장에서는 가상의 공격 시나리오를 통한 공격과 탐지의 시뮬레이션 결과를 기술한다. 마지막으로 Ⅴ장에서는 향후 연구방향을 제시하고 결론을 맺는다.

Ⅱ. 관련 연구

2.1. 침입탐지 기법

침입탐지시스템은 침입을 탐지하는 방법에 따라 Misuse Detection, Anomaly Detection 및 Specification-based Detection으로 구분한다.

2.1.1. Misuse Detection

Misuse Detection은 정보시스템의 취약점에서 침입이 발생하는 경우 이를 탐지하는 기능으로 침입 사고가 일어날 때 나타나는 정형적인 패턴들을 감시함으로써 탐지가 이루어진다. 침입에 사용되는 정형적인 패턴들을 미리 가지고 있어 침입패턴만 찾으면 되므로 빠르게 검색할 수 있고, 정상행위를 잘못된 행위로 판단하는 오탐이 매우 낮은 장점이 있다. 그러나 이미 알려진 취약점에 대해 침입탐지의 정확도가 매우 높지만 새로운 유형의 침입에 대해서는 탐지하기 어려운 단점이 있다[2]. Misuse Detection에서 쓰이는 기법은 [표 1]과 같다[3].

〔표 1〕 Misuse Detection의 이용 기법

이용 기법	내 용
패턴 매칭 (pattern matching)	알려진 침입 유형들에 대한 데이터를 가지고 일어나는 행위들을 침입 시나리오로 설정된 패턴들과 비교하여 탐지
조건부 확률 (conditional probability)	내부로 유입되는 데이터 중 특정 데이터가 침입일 확률을 공식에 의해 계산 후 탐지
전문가 시스템 (production / expert system)	침입 패턴들에 대한 데이터를 가지고 그 행위를 명시하고, 그와 일치된 침입행위를 발견하는 경우 IF-THEN 규칙에 따라 탐지
상태전이 분석 (state transition analysis)	침입행위를 특정 시스템의 상태 전이의 순서로 표현하여, 하나의 상태가 주어진 조건을 분석하여 탐지

2.1.2 Anomaly Detection

Anomaly Detection은 일반적인 시스템 사용 패턴에서 벗어나는 비정상 행위들을 탐지한다. 이러한 비정상적 행위는 외부의 침입뿐 아니라 내부의 시스템 남용

〔표 2〕 Anomaly Detection의 이용 기법

이용 기법	내 용
통계적 접근 (statistical approach)	과거의 경험적인 자료를 토대로 처리하는 방식으로 사용자나 프로세스의 행위를 관찰해서 각각의 행위에 대한 프로파일을 생성하고, 주기적으로 관찰하여 통계적 이상 여부에 따라 처리하는 기법
특징 추출 (feature selection)	특정 침입의 패턴을 추출하는 방법으로 경험적인 침입탐지 측정도구의 집합을 설정하여 침입의 예측, 분류 가능한 침입탐지 도구의 부분집합을 결정해서 침입을 예측하고 분류하는 기법
신경망 (neural network)	명령의 순서를 신경망으로 학습시켜 다음에 수행될 명령어를 미리 예측하는 기법

으로도 발생할 수 있으며, 알려져 있는 침입, 알려지지 않은 침입 모두 탐지할 수 있지만 Misuse Detection과 달리 일정한 패턴이 있지 않기 때문에 감사 자료를 분석하여 판단하여야 한다[2]. Anomaly Detection에서 쓰이는 기법은 [표 2]와 같다[3].

2.1.3. Specification-based Detection

Specification 기반의 탐지 기법은 정상행위에서 벗어난 공격을 탐지한다는 점에서는 Anomaly Detection과 유사하다. 그러나 기계 학습 기법에 의존하지 않고, 적법한 시스템 행위들을 캡쳐 해 수동으로 개발한 명세를 기반으로 하며, 개발한 명세들에 보안 규칙을 적용시켜 그 규칙을 위험한 객체의 실제 동작과 비교해 탐지하는 특징이 있다.

이러한 Specification 기반의 탐지 기법은 Misuse Detection의 단점인 알려지지 않은 침입에 대응할 수 있고, Anomaly Detection의 단점인 오탐율의 비율을 낮출 수 있는 장점이 있다[4].

Specification 기반의 탐지 기법은 Anomaly Detection의 장점과 Misuse Detection의 장점을 가지며 프로파일 작성이나 알려진 공격 패턴의 수집이 필요 없기 때문에 휴대용 단말기 상에서의 침입탐지에 적합할 것으로 판단되어 본 논문에서는 Specification 기반의 침입탐지 방법을 채택하였다. 그러나 효율적인 침입탐지를 위해서는 상세한 명세의 개발이 필요하기 때문에 WIPI HAL의 명세를 바탕으로 침입탐지 시스템을 설계한다.

2.2. WIPI의 개념

WIPI는 이동통신 단말기에 탑재되어 응용프로그램의 실행 환경을 제공하는데 필요한 한국 표준규격으로 2002년 5월 한국정보통신기술협회(TTA) 단체 표준인 TTAS-KO-06.0036(모바일 표준 플랫폼 규격)으로 채택되었다[5].

WIPI의 지원 언어는 C와 자바가 동시에 지원되는 구조이며, 플랫폼과 응용프로그램은 하드웨어에 독립적인 구현이 가능하도록 CPU, LCD 및 메모리 등이 단말기 하드웨어나 운영체제에 관계없이 실행과 이식이 용이하다. 또한 응용프로그램이 이동통신사업자 및 단말기 제조사의 비밀이나 단말기 사용자 개인정보를 마음대로 접근할 수 없도록 하는 보안 규격도 포함되어 있다[6].

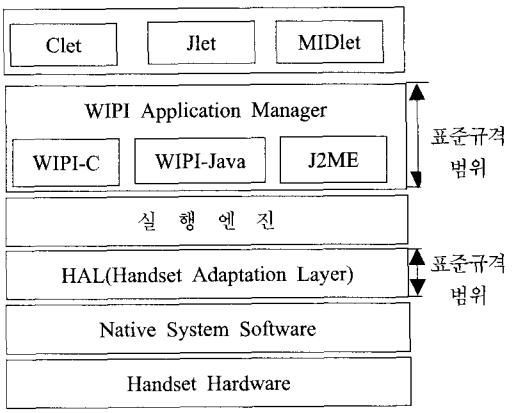
2.2.1. 구조

WIPI 2.0 플랫폼의 구조는 [그림 1]과 같다. 하위에 있는 Native System Software는 간단한 단말기 운영체제 기능과 통신 기능 및 각종 디바이스 드라이버가 포함된다. HAL은 단말기 제조사를 위한 API를 정의한 것으로 단말기 제조사마다 서로 다른 기기에 공통된 플랫폼을 지원하기 위한 추상화 계층을 도입하였으며, WIPI에서 획기적인 것으로 받아들여지고 있다[7]. WIPI 표준 규격의 범위는 HAL 계층과 WIPI Application Manager이고 표준 API의 전형적인 동작을 추상화하여 정의하고 있다[8].

2.2.2. 주요기능

WIPI 플랫폼은 바이너리 응용프로그램을 서버로부터 다운로드 받아 수행하고, 동시에 여러 개의 응용프로그램을 메모리에 적재하여 다중 수행할 수 있는 환경을 제공한다.

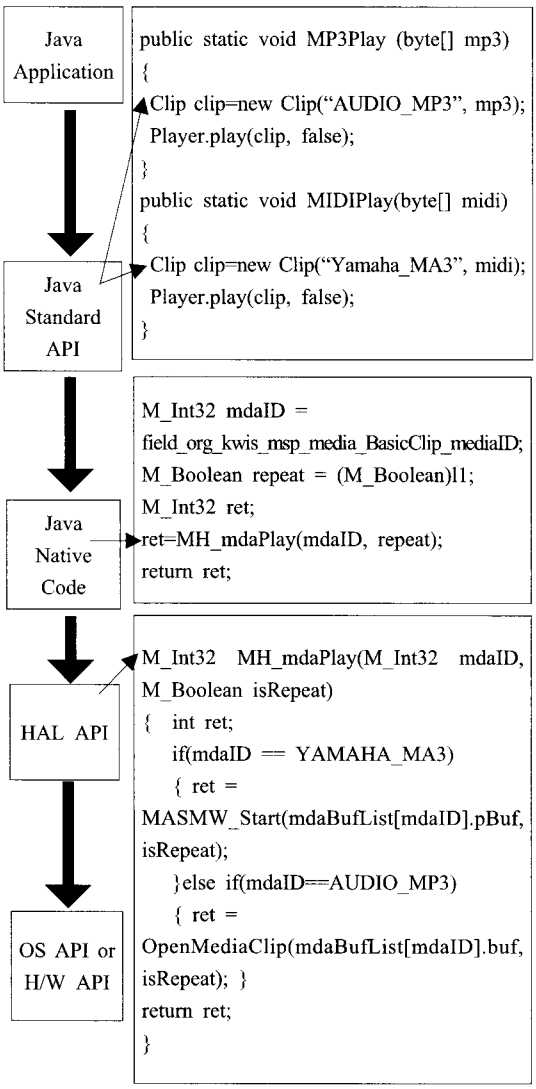
현재 지원하는 언어는 C 언어와 자바 언어뿐이지만 마이크로소프트는 자사의 C#을 채택하려는 노력을 진행하고 있다. 플랫폼은 일반 수준, 콘텐츠 개발자 수준, 시스템 수준의 세 가지 보안 수준을 지원한다. 또한 API별로 접근수준을 지정할 수 있으며, 플랫폼은 자동 메모리 해제, 메모리 컴팩션, 자바 가비지 컬렉션, 공유 메모리 지원 등 메모리 관리 기능과 응용프로그램 관리 기능을 가지고 있다.



〔그림 1〕 WIPI 플랫폼 구조

2.2.3. 자바 API의 HAL API 호출

WIPI에서 동작하는 응용프로그램이 자바로 만들어진 경우 개발자가 사용한 자바 표준 API는 자바 Native 코드로 변경되면서 HAL API를 호출한다. 호출된 HAL API는 단말기의 하드웨어 API나 운영체제 API를 호출하여 단말기를 제어하고, 단말기 리소스에 접근하게 된다. 본 논문에서는 이런 응용프로그램 API가 HAL API를 호출할 때 HAL API의 명세에 근거해 침입을 탐지하는 알고리즘을 제안한다. 자바 API의 HAL API 호출 과정은 [그림 2]와 같다[9].



[그림 2] 자바 API의 HAL API 호출 과정

Ⅲ. Specification 기반의 침입탐지시스템 설계

본 논문에서는 WIPI 플랫폼의 보안 요구사항을 분석하여 침입탐지 알고리즘을 제안하며, 이를 수행하기 위한 세부 명세를 개발하고, 제안한 mIDS API의 프로토타입을 정의한다.

3.1. WIPI 보안 요구사항 분석

WIPI 규격은 플랫폼의 보안을 정의하고, 보안 수준과 보안의 대상이 되는 자원을 식별하고 있으므로 보안의 정의, 보안의 수준 및 보안 수준에 따른 접근 허용을 분석하여 침입탐지 알고리즘을 도출한다.

3.1.1. 보안의 정의

플랫폼에서의 보안은, 한 응용프로그램이 플랫폼의 자원(API나 저장 공간, 공유 메모리 등)을 통해서 다른 엔티티(다른 응용프로그램 혹은 응용프로그램이 동작하고 있는 플랫폼이나 플랫폼이 탑재된 단말기 혹은 다른 단말기가 연동하고 있는 네트워크/서버 시스템 등)의 동작에 영향을 끼치거나, 다른 엔티티가 소유하고 있는 정보에 접근하는 경우에 대한 정책적, 기술적 대책을 의미한다. 즉 플랫폼은 보안 관련 규격에 따라 응용프로그램의 플랫폼 자원에 대한 접근을 제어할 수 있으며, 플랫폼의 자원과 보안 수준에 대한 규격을 정의하고, 응용프로그램의 자원 접근 제어를 위한 테이블을 가진다. [표 3]은 플랫폼에서 정의하는 자원별 보안 수준이다.

3.1.2. 보안 수준

플랫폼의 보안 기법은 플랫폼의 각 자원별로 정의된 보안 수준에 따른 접근 허용 여부와 응용프로그램의 보안 수준에 대한 정보를 바탕으로 수행된다.

[표 3] 자원별 보안 수준

구 분	Resource 1	.....	Resource N
Application 1	System	.....	System
.....	.....	.....	.....
Application N	CP	.....	Public

〔표 4〕 플랫폼 보안 수준

보안 수준	내 용
PUBLIC	플랫폼에 존재하는 보안 레벨 중 가장 신뢰할 수 없는 수준으로 가장 제약 사항이 많은 수준을 뜻한다. 플랫폼은 PUBLIC 수준의 응용프로그램이 실행될 때 단말기 시스템에 영향을 줄 수 있거나 개인 정보에 접근하는 등 보안 문제를 야기할 소지가 있는 플랫폼 자원에 대한 접근을 허용해서는 안된다.
CP	플랫폼이 응용프로그램을 일정 수준 이상 혹은 전폭 신뢰가 가능하여 PUBLIC 수준에서의 자원에 대한 제약 사항을 일부 혹은 모두 해제한 수준을 의미한다.
SYSTEM	SYSTEM 보안 수준은 사실 CP 보안 수준의 부분 집합으로써 플랫폼이 응용프로그램을 완전히 신뢰 가능한 것으로 간주하고, 모든 자원에 대한 접근을 자동 허용하는 것을 말한다.

플랫폼의 보안 수준의 종류 혹은 개수는 최소한 하나 이상 정의되어야 하며, 그 내용은 구현에 관한 사항으로 정의하고 있다. 모바일 표준 플랫폼 규격 V2.0.1에서는 최소한 PUBLIC, CP, SYSTEM으로 구분할 것을 권장하고 있다. 플랫폼 보안 수준의 세부적인 내용은 [표 4]와 같다.

〔표 5〕 자원 접근 제어

권 한 구 분		내 용
허용(ALLOW)		사용자의 개입 없이 자동으로 자원에 대한 접근을 허용한다.
거부(DENY)		사용자의 개입 없이 자동으로 자원에 대한 접근을 거부한다.
사용자 허락에 의한 접근 허용 (USER)	계속거부	응용프로그램이 Uninstall 될 때까지 거부한다.
	종료 시 까지 거부	응용프로그램의 실행이 종료될 때까지 거부한다.
	이번만 거부	응용프로그램은 이번 호출에 한해 자원에 접근할 수 있어서는 안되며, 다음 번에는 다시 사용자에게 문의한다.
	이번만 허용	단 한번의 함수 호출만을 허용해야 하며, 다음 번에는 다시 사용자에게 문의한다.
	종료 시 까지 허용	응용프로그램의 실행이 종료될 때까지 허용해야 한다.
	계속 허용	응용프로그램이 Uninstall 될 때까지 허용해야 한다.

3.1.3. 자원의 보안 수준별 접근 허용 여부

WIPI 플랫폼에는 각 자원별로 보안 수준에 따라 접근허용 여부가 정의된 표가 존재한다. 또한 응용프로그램별로 보안 수준이 지정되어 있다.

플랫폼은 응용프로그램이 어떤 자원을 접근하고자 할 경우 응용프로그램의 보안 수준과 자원의 보안 수준별 접근인가 조건에 따라 응용프로그램의 자원접근 시도에 대해 [표 5]와 같은 제어를 수행할 수 있다.

이러한 자원의 보안 수준별 접근인가표는 단말기 플랫폼에 존재할 수도 있고, 응용프로그램을 공급하는 서버에 존재할 수도 있다. [표 6]은 보안 수준별 접근인가표를 보여주고 있다.

3.1.4. 침입탐지 알고리즘

분석한 보안 요구사항에 근거하여 설계된 침입탐지 알고리즘의 실행 순서도는 [그림 3]과 같다.

먼저 [그림 3]의 (1)에서 WIPI 응용프로그램이 구동되면 플랫폼은 구동된 응용프로그램이 Clet, Jlet, MIDlet 중 어느 것이냐에 따라 WIPI-C, WIPI-자바, J2ME(Java 2 Platform, Micro Edition)에 매핑되며, 그 외 다른 프로그램은 실행이 거부된다.

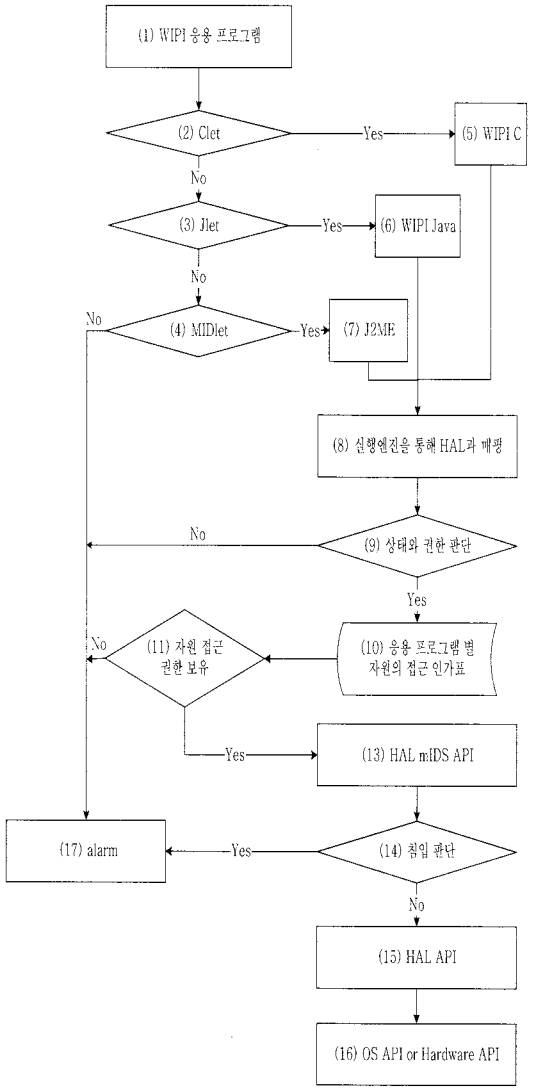
이후 (9)의 단계에서는 응용프로그램의 상태와 권한을 판단하여 자원의 접근인가표를 이용해 접근권한의 보유여부를 판단하게 된다.

접근권한이 없는 경우는 실행 거부 및 경보를 발생하며, 정상적인 권한을 보유한 경우는 HAL 계층의 mIDS API와 명세를 통해 침입을 판단한다. 모든 침입의 경우 실행거부와 경보를 발생하고, 정상적인 경우는 단말기 운영체제나 Hardware API를 실행한다.

즉, 제안된 침입탐지시스템은 응용프로그램의 상태와 권한을 검증하며, HAL 계층에 mIDS API 그룹을

〔표 6〕 보안 수준별 접근인가표

구 분	Graphic	Sound	Persona l Info	System Property
Public	Allow	Allow	Deny	Deny
CP	Allow	Allow	User	Deny
System	Allow	Allow	Allow	Allow



〔그림 3〕 침입탐지 알고리즘

둠으로써 HAL API들이 명세에 어긋나는 행위를 할 경우 침입으로 간주하여 탐지한다. 이는 WIPI 플랫폼의 하위 수준인 HAL 계층에서 응용프로그램(Clet, Jlet, MIDlet)과 무관하게 탐지할 수 있는 장점이 있으며, 응용프로그램의 권한을 적극 통제함으로써 보다 세부적인 권한관리를 수행할 수 있고, 호출되는 API별로 명세를 개발하여 탐지하므로 알려지지 않은 공격과 성능이 제한적인 휴대용 단말기에 적합하다고 할 수 있다.

3.2. mIDS API 그룹 설계

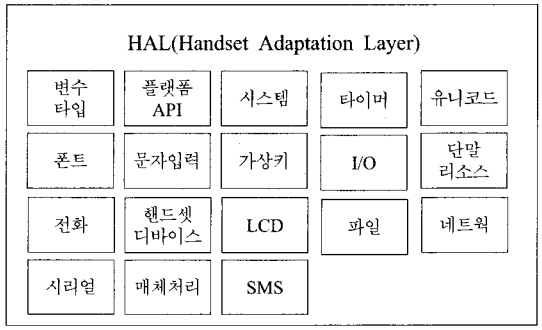
WIPI HAL은 플랫폼 하드웨어의 독립성을 유지하기 위한 추상화 계층으로 플랫폼의 상위 계층은 HAL API만을 호출해서 하드웨어 자원에 접근한다. 이를 통해 단말기의 네이티브 시스템에 대한 추상화가 이루어지고 독립적으로 플랫폼을 구성할 수 있다.

HAL의 목적은 운영체제에 대한 이식성 향상에 있으며, 각 단말의 네이티브 시스템에 근거하여 HAL API만 구현하면 플랫폼 내부의 프로그램 코드를 변경하지 않고도 플랫폼 전체의 이식이 가능하다.

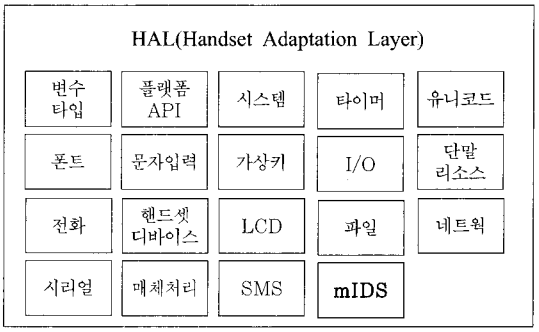
본 논문에서는 모든 응용프로그램이 HAL을 통해서만 하드웨어와 단말 리소스에 접근할 수 있는 특성을 이용해 mIDS API 그룹을 설계한다. HAL에서 정의하고 있는 API는 세부적으로 시스템 API를 포함하여 18개 항목으로 [그림 4]와 같이 구성되어 있다. HAL은 접근대상이 되는 자원에 따라 구분되어 있으며, 각 자원에 대한 전형적인 사용방식을 추상화한 형태로 정의되어 있다.

본 논문에서 제안하는 mIDS API 그룹이 추가된 HAL API의 개념적 구성도는 [그림 5]와 같다.

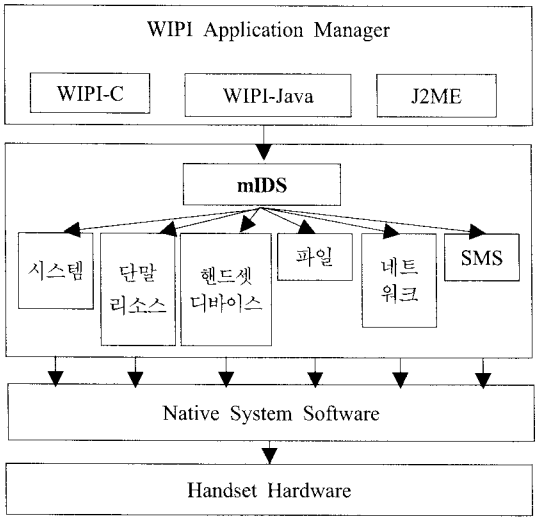
mIDS API 그룹은 WIPI Application Manager 계층에서 요청한 HAL API를 감시하는 역할을 수행하며



〔그림 4〕 HAL API 구성도



〔그림 5〕 mIDS API 그룹이 추가된 HAL API 구성도



〔그림 6〕 운영 개념도

mIDS API 그룹의 운영 개념도는 [그림 6]과 같다.

mIDS API 그룹은 HAL API 중 시스템, 단말리소스, 핸드셋 디바이스, 파일, 네트워크 및 SMS API를 감시하고 침입이 탐지된 경우에 경보를 발생한다. HAL API의 호출 시 제일 우선하여 프로그램의 현재 상태를 확인하는데 이는 Active 상태면서 접근하고자 하는 자원에 대한 권한이 유효해야 한다.

if(ApplicationState(ApplicationName)=="ACTIVE" && ApplicationAuth(ApplicationName,ResourceName) =="TRUE") mIDS API call else abort( )

3.3. mIDS API 설계

mIDS API 그룹에 대해 시스템, 단말리소스, 핸드셋 디바이스, 파일, 네트워크 및 SMS API를 감시하고 검증할 수 있는 세부 API들을 설계한다.

3.3.1. 시스템

모바일 표준 플랫폼 규격 v2.0.1에서는 시스템 API를 정의하고 있는데 시스템 API는 표준 플랫폼 커널에서 단말기의 정보 또는 이벤트를 입력받아 다음 수행 동작을 결정하게 된다. 시스템 API를 기능별로 구분하면 커널의 동작 수행 상태를 확인하기 위해 디버깅하는 콘솔을 지원하는 함수와 크리티컬 섹션을 보호하기 위해 단말기 운영체제에서 지원하는 잠금 메커니즘을 구현한 함수, 커널이 사용할 메모리 영역을 잡아주는 함수 등으로 구분할 수 있다. 시스템 API에 대한 정의를 바탕으로 비정상 행위를 다음의 세 가지 방법에 따라 탐지한다.

3.3.1.1. 플랫폼을 제외한 타 응용프로그램 접근 탐지

HAL 계층에서는 플랫폼을 위한 API들이 다수 존재한다. 그러므로 플랫폼 이외의 응용프로그램에서 플랫폼을 위한 API를 호출하려 할 경우 이는 악의적인 사용으로 간주할 수 있다.

플랫폼을 위한 API는 PlatformAPIList를 새롭게 정의하여 그 목록 정보를 갖도록 한다. 즉, 플랫폼을 제외한 타 응용프로그램의 모듈이 사용해서는 안 되는 시스템 API의 목록을 정의하고, 호출된 API가 해당 목록에 포함되어 있으면 침입으로 탐지한다.

PlatformAPIList = { MH_sysGetHeapBlock }
if (called API ∈ PlatformAPIList) alarm( )

3.3.1.2. 너무 잦은 호출에 의한 탐지

HAL 계층은 하드웨어의 접근을 위한 통로로 사용된다. 그러나 너무 잦은 호출이 발생할 경우 이는 다분히 악의적 DOS(Denial Of Service) 공격이라고 판단할 수 있다. 그러므로 시스템 정보를 얻으려는 API를 너무 자주 호출하는 경우 경보를 발생한다.

if (API == MH_sysGetInformation &&
tooManyAttempts( )) alarm( )

3.3.1.3. 크리티컬 섹션 진입 탐지

단말기의 운영체제는 플랫폼에 이벤트를 전달하기 위해 MH_pltEvent( ) 함수를 호출하는데 이때 호출된 MH_pltEvent( ) 함수와 플랫폼의 context에서 사용하는 영역 간에 크리티컬 섹션이 발생한다. 이러한 크리티컬 섹션은 플랫폼 외부 문맥이 진입하여 악의적 코드가 삽입될 수 있다. 그러므로 HAL이 MH_sysLock( ) 함수로 크리티컬 섹션을 막은 경우 MH_pltEvent( ) 함수가 호출되어 크리티컬 섹션으로 코드 삽입을 시도한다면 이는 악의적인 코드 삽입이라고 판단한다.

if(MH_sysLock( )) { MH_pltEvent( ) call } alarm( )

3.3.2. 단말 리소스

이미지, 사운드, 주소록 등 특정 데이터 포맷을 가지면서 단말 영역에 저장된 데이터들을 통칭하여 단말 리소스라고 한다. 단말 리소스 API들은 WIPI 응용프로그램이 리소스 그룹과 리소스에 접근하는 통로를 제공한다. 단말 리소스 API는 그 기능에 따라 단말 리소스 관리, 단말 리소스 그룹 관리, 단말 리소스 보안, 단말 리소스 기타로 그룹 짓는다. 각 그룹에 속하는 함수들은 다음과 같다.

```
ResourceManagement =
{ MH_termResGetFormat, MH_termResGetSize,
  MH_termResGetUIName,
  MH_termResRead, MH_termResWrite,
  MH_termResDelete, MH_termResRegister,
  MH_termResGetRegisteredInfo,
  MH_termResGetInfo, MH_termResSearch }
ResourceGroupManagement =
{ MH_termResGetSupportedGroups,
  MH_termResGetCount, MH_termResGetList,
  MH_termResGetGroupInfo }
ResourceSecurity =
{ MH_termResGetGroupLockState,
  MH_termResGetLockState,
  MH_termResSetLockState,
  MH_termResSetLockState,
  MH_termResCheckPassword }
ResourceETC = { MH_termResGetFreeSpace,
  MH_termResExecuteCmd }
```

3.3.2.1. 정의되지 않은 리소스 탐지

단말 리소스는 MIME 타입과 데이터 포맷을 가지고 있다. 이미지, 동영상, 사운드, 폰북, BLACKLIST, SMS 등의 단말 리소스는 각각의 MIME 타입에 해당되지 않는 데이터를 안전하지 않다고 판단한다. 즉, 정해지지 않은 데이터 포맷은 미확인된 리소스로 간주하여 경보를 발생한다.

```
if (resource ∉ (imageMIMEList ||
animationMIMEList || videoMIMEList ||
soundMIMEList || phonebookMIMEList ||
blacklistMIMEList || SMSMIMEList) ) alarm( )
```

또한 모든 리소스는 리소스 그룹 중 하나에 속하도록 구현되어야 하므로 리소스가 어떠한 리소스 그룹에도 속하지 않은 경우도 동일하게 경보를 발생한다.

```
if( resource ∉ ResourceGroupList)  alarm( )
```

3.3.2.2. 보안 기능 수행 여부에 의한 탐지

단말 리소스는 보안 기능을 강화하여 단말 리소스 보안 API를 별도로 정의하고 있는데 이때 그룹 Lock된 리소스 데이터를 읽고 쓰고 지우거나, 개별 Lock된 리소스의 데이터를 읽고 쓰고 지우는 경우에는 단말기의 비밀번호를 확인하고 리소스에 접근할 수 있도록 규정하고 있다. 그러므로 리소스 관리 그룹에 속하는 API가 호출될 경우 비밀번호를 점검하는 루틴을 수행하지 않으면 인증되지 않은 접근으로 탐지한다.

```
if( resource || resourceGroup is locked)
 { if( (API ∈ ResourceManagement) && Not
Called MH_termResCheckPassword( ) ) alarm( ) }
```

3.3.3. 핸드셋 디바이스

HAL의 핸드셋 디바이스 API는 단말기에 장착된 장치를 제어하는 함수들이다. 핸드셋 디바이스는 부정 작동이나 잦은 호출로 인한 상태 변경이 일어날 경우 배터리 소모를 통한 DOS 공격으로 예측할 수 있다. 그러므로 핸드셋 디바이스 API 목록을 만든 후 API가 핸드셋 디바이스의 제한치를 초과하는 동작을 요청하거나 너무 잦은 호출로 상태 변경을 시도하는 경우 이를 배터리 소모를 통한 DOS 공격으로 탐지한다.

```
HandsetDeviceMethodList =
{ MH_devVBacklight, MH_devVibrator,
  MH_devLedControl }
if(API ∈ HandsetDeviceMethodList &&
  tooManyAttempts( )) alarm( )
```

3.3.4. 파일

WIPI에서 모든 API의 파일 경로는 절대 경로로 접근되며 파일 관련 함수는 쓰기, 읽기, 속성 변경, 속성 확인 등의 큰 그룹으로 구분할 수 있다. 파일 쓰기 함수 그룹은 WriteOperationGroup에 정보를 저장하고, 읽기 함수 그룹은 ReadOperationGroup에 정보를 저장한다. 속성 변경

함수 그룹은 FileAttributeChangeOperationsGroup에 정의하고, 속성 확인 함수는 FileAttributeCheckOperations에 각각 그 정보를 저장한다.

WriteOperationsGroup(path) = { MH_fileClose, MH_fileMkDir, MH_fileOpen, MH_fileRemove, MH_fileRename, MH_fileRmDir, MH_fileWrite }
ReadOperationsGroup(path) = { MH_fileList, MH_fileRead, MH_fileTotalSpace, MH_fileGetCounts }
FileAttributeChangeOperationsGroup(path) = { MH_fileSeek, MH_fileSetMode }
FileAttributeCheckOperationsGroup(path)= {MH_fileAttribute, MH_fileAvailable, MH_fileIsExist, MH_fileTell}

이렇게 정의된 각 그룹에 속하는 API의 목록을 작성하고, 쓰기 동작에서는 절대경로가 사용자 영역이 아닌 경우에 탐지하고, 읽기 동작에서는 절대경로가 시스템 영역인 경우에 침입으로 탐지한다.

if(API ∈ WriteOperationsGroup(path) && path ∉ userArea) alarm( )
if(API ∈ ReadOperationsGroup(path) && path ∈ systemArea) alarm( )

3.3.5. 네트워크

네트워크 API에서는 TCP/IP 인터넷 통신을 지원하는 함수들을 정의하고 있다. 플랫폼에서는 MH_netConnect( ) 함수의 호출에 의해 인터넷 사용이 가능해지며 MH_netClose( ) 함수가 호출된 이후에는 인터넷을 통한 데이터 통신이 불가능해야 한다. 그러므로 인터넷 통신이 종료된 이후 데이터 통신을 요청하는 경우 경보를 발생한다.

if(API == MH_netClose( ) &&
called MH_NETEV_NETWORK_OPEN) alarm( )

3.3.6. SMS

WIPI 플랫폼은 SMS를 이용하여 문자 메시지를 보내기 위한 함수를 제공한다. 핸드셋 장비를 대상으로 하는 SMS 관련 공격은 이미 상당수 식별된 바 있다. 그러므로 본 논문에서는 이러한 경우를 탐지하기 위해 지속적인 SMS 호출이 발생할 경우 이는 휴대용 단말기의 배터리 소모를 통한 DOS 공격으로 탐지한다[10].

if(API==MH_smsSend && tooManyAttempts( ))
alarm( )

3.4. 프로토타입 정의

WIPI HAL에 적용할 mIDS API 그룹의 구성은 [표 7]과 같다.

지금까지 설명한 mIDS API의 기본적인 적용방식을 추상화한 프로토타입은 [표 8]과 같이 정의할 수 있다.

Ⅳ. mIDS 구현 및 동작 시뮬레이션

WIPI HAL은 블랙박스화되어 있어 일반 개발자가 접근할 수 없는 영역이다. 이러한 영역에 새로운 mIDS API를 그대로 구현하는 것은 상당한 어려움이 있다. 따라서 본 논문에서는 mIDS API의 프로토타입을 토대로 이를 검증하기 위한 제한적인 구현을 위해 자바 라이브러리를 이용한다.

4.1. 구현 모델

〔표 7〕 mIDS APIs 구성

구분	이 름
자료 정의	PlatformAPIList
	ResourceGroupList
	HandsetDeviceMethodList
	ResourceManagement
	ResourceGroupManagement
	ResourceSecurity
	ResourceETC
	ResourceGroup
API	ApplicationState(M_Char ApplicationName)
	ApplicationAuth(M_Char ApplicationName)
	alarm()
	tooManyAttempts()
	WriteOperationsGroup(path)
	ReadOperationsGroup(path)
	FileAttributeChangeOperationsGroup(path)
	FileAttributeCheckOperationsGroup(path)
	mIDfileAuth(String pName, String path, int auth)
	abort()

〔표 8〕 mIDS API 프로토타입 정의

■ String ApplicationState(M_Char ApplicationName)
프로토타입 :
String ApplicationState(M_Char ApplicationName)
설명 : 이 함수는 응용프로그램의 상태를 반환한다.
상태의 종류는 Active, Destroyed, Paused로 반환된다.
매개 변수 : ApplicationName은 현재 상태를 반환할 응용프로그램의 이름
반환값 : Active || Destroyed || Paused
참고사항 : API 호출시 프로그램의 상태를 반환하며 Active 이외의 경우에 비정상 상황으로 판단한다.

■ String ApplicationAuth(M_Char ApplicationName, M_Char ResourceName)
프로토타입 : ApplicationAuth(M_Char ApplicationName, M_Char ResourceName)
설명 : 이 함수는 응용프로그램이 리소스에 대해 갖고 있는 접근 권한을 반환한다. 접근 권한은 Allow, Deny, User의 형태로 반환된다.
User인 경우는 사용자가 선택한 바에 따라 해당 값을 반환하게 된다.
매개 변수 : ApplicationName은 현재 상태를 반환할 응용프로그램의 이름
ResourceName은 접근 권한을 확인 할 대상 리소스의 이름
반환값 : Allow || Deny || User
{ User-allow // 이번만 허용
User-deny // 이번만 거부
User-all-allow // 계속거부
User-all-deny // 계속허용
App-End-allow // 실행종료 시 까지 허용
App-End-deny // 실행종료 시 까지 거부 }

■ void alarm(String text)
프로토타입 : void alarm( ) / void alarm(path)
설명 : 이 함수는 침입탐지 mIDS API에서 침입을 탐지한 경우에 로그 및 화면을 이용해 사용자에게 경고 메시지를 전달하거나 path의 경로에 로그를 작성한다.
매개 변수 : 매개 변수가 없는 경우는 단순 경보만 발생시키고 절대경로 path의 형태로 매개 변수를 전달하는 경우는 절대경로의 위치에 로그를 만든다. 문자열 매개 변수는 그대로 출력한다.

■ void tooManyAttempts( )
프로토타입 : void tooManyAttemps( )
설명 : 이 함수는 WIPI HAL API의 잦은 호출 발생 시 악의적인 행동으로 간주한다. SMS인 경우는 트래픽 양을 모니터한다.
반환값 : 1초 이내에 3번 이상의 동일한 API 호출 발생 시 TRUE를 반환하고, 발생하지 않을 경우 FALSE를 반환한다.

■ WriteOperationsGroup(path)
프로토타입 : WriteOperationsGroup(path)
설명 : 이 함수는 path 파일에 대한 쓰기 API 실행을 탐지한다.
매개 변수 : path는 파일이 존재하는 절대경로를 나타낸다.
반환값 : 개인정보에 대한 쓰기 API가 실행될 때 TRUE를 반환한다. 그 외에는 FALSE를 반환한다.

■ ReadOperationsGroup(path)
프로토타입 : ReadOperationsGroup(path)
설명 : path 파일에 대한 읽기 API 실행 탐지
반환값 : 시스템 정보에 대한 읽기 API가 실행되면 TRUE를 반환한다. 그 외에는 FALSE를 반환한다.

■ FileAttributeChangeOperationsGroup(path)
프로토타입 :
FileAttributeChangeOperationsGroup(path)
설명 : path 파일에 대한 속성 변경 API 실행 탐지
반환값 : 개인 정보 파일 속성이 변경되려 할 때 TRUE를 반환한다. 그 외에는 FALSE를 반환한다.

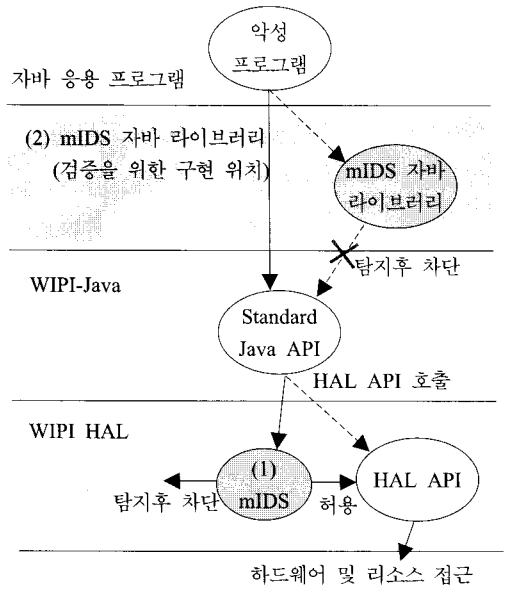
■ FileAttributeCheckOperationsGroup(path)
프로토타입 : FileAttributeCheckOperationsGroup(path)
설명 : path 파일에 대한 속성 읽기 API 실행 탐지
반환값 : 시스템 파일에 대한 속성을 읽으려고 할 때 TRUE를 반환한다. 그 외에는 FALSE를 반환한다.

■ boolean mIDfileAuth(String pName, String path, int auth)
프로토타입 : boolean mIDfileAuth(String pName, String path, int auth)
설명 : 파일에 대한 접근 권한을 검증한다.
매개 변수 : pName은 프로그램 이름, path는 파일의 이름을 포함한 절대 경로, auth는 수행하려는 명령어를 나타낸다.
반환값 : 미리 정의된 접근권한표에서 응용프로그램이 파일에 대한 수행 권한이 있는 경우 TRUE를 반환하고, 권한이 없는 경우는 FALSE를 반환한다.

■ void abort()
프로토타입 : void abort( )
설명 : 즉시 응용프로그램의 실행을 중지한다.

mIDS를 자바 라이브러리로 구현하는 논리적 구현 모델은 [그림 7]과 같다. 이러한 구현 모델은 HAL에 대한 접근이 불가능하기 때문에 제시되었다. 따라서 최초에 제안했던 HAL 계층인 (1)의 위치에 mIDS를 직접 구현할 수 없으므로 (2)의 위치에 (3)의 mIDS 자바 라이브러리를 구현한다. 이를 이용하면 제안한 침입탐지 알고리즘을 부분적으로 적용할 수 있다.

이는 WIPI가 콘텐츠 개발자를 위해 WIPI Application



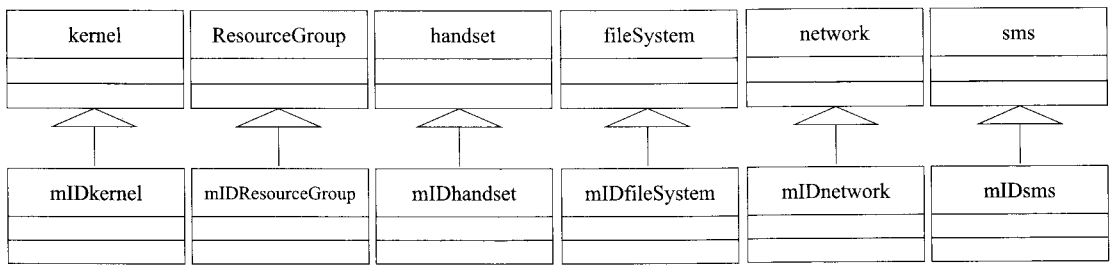
〔그림 7〕 mIDS의 논리적 구현 모델

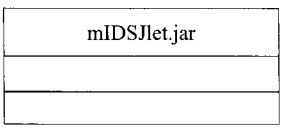
Manager 계층에서 WIPI-Java의 API를 개념적으로 정의하고 있으며, 이동통신사는 이를 구현하여 개발자가 활용할 수 있도록 일부를 제공하고 있기 때문이다. 본 논문에서는 이렇게 공개된 WIPI-Java 기본 API를 활용한다.

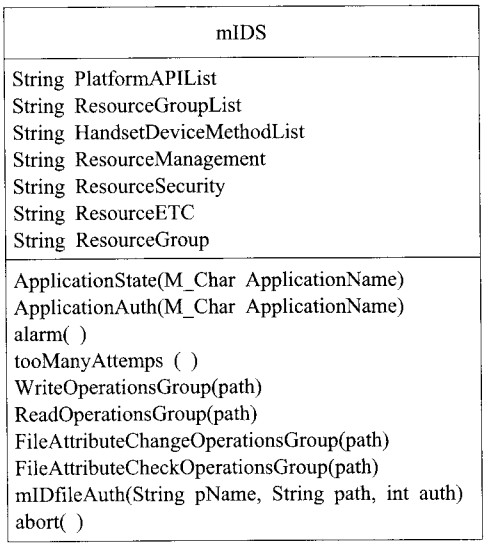
HAL의 시스템, 단말리소스, 핸드셋 디바이스, 파일, 네트워크, SMS API는 WIPI-Java의 kernel, ResourceGroup, handset, fileSystem, network, sms 클래스와 매핑 된다.

따라서 mIDS 라이브러리는 기존의 클래스를 상속받아 명세 기반의 탐지 코드를 추가하여 mIDkernel, mIDResourceGroup, mIDhandset, mIDfileSystem, mIDnetwork, mIDsms라고 이름 짓고 여기에 mIDS를 추가하여 침입 탐지 코드를 구현한다.

이때 악성코드를 실행할 프로그램은 Jlet으로 구현하여 mIDSJlet.jar라고 이름 붙이고 WIPI 에뮬레이터를 이용해 실행하며, 구현 시에는 mIDS 라이브러리를







〔그림 8〕 클래스 다이어그램

이용한다. WIPI 에뮬레이터를 이용해 프로그램을 실행하면, 참조된 mIDS 라이브러리 및 mIDkernel, mIDResourceGroup, mIDhandset, mIDfileSystem, mIDnetwork, mIDsms는 mIDSJlet.jar의 침입을 탐지하여 경보를 발생하고 그 결과를 로그로 작성한다.

이러한 구조를 클래스 다이어그램으로 작성하면 [그림 8]과 같이 나타낼 수 있다. 여기에는 기존 클래스와의 상속 관계와 공격 프로그램과 새롭게 작성된 클래스 간의 연관관계를 나타내고 있다.

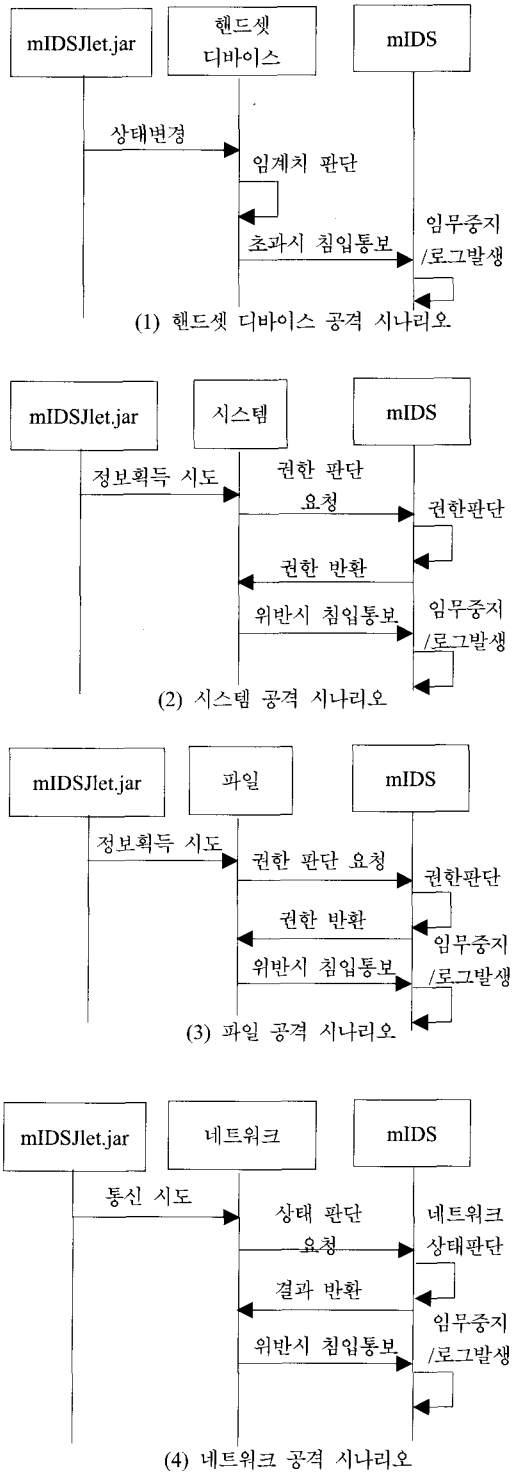
4.2. 공격 시나리오

공격 시나리오는 자바 라이브러리로 구현이 가능한 부분만을 선택하여 작성한다. 본 논문에서 공격할 대상은 핸드셋 디바이스, 시스템 명령어, 파일 및 네트워크로 정한다. 공격 방법은 핸드셋 디바이스를 조작한 배터리 소모 공격과 시스템 명령어를 이용한 단말기 주요 정보 획득 시도 및 주요 파일에 대한 접근 시도와 임의적인 네트워크 통신 시도로 한다.

배터리 소모 공격은 단말기의 백라이트 조명과 진동 모드의 지속시간을 임의적으로 과도하게 설정하여 배터리 소모 공격을 실시하며, 시스템 명령어를 이용한 공격은 사용자 및 단말기의 중요 정보를 획득한다. 파일 공격은 개인 파일로 설정된 파일을 접속하여 중요 정보를 획득한다. 네트워크 공격은 정상적인 통신이 종료된 이후에 임의로 데이터 통신을 시도한다.

공격에 대한 탐지 방법은 핸드셋 디바이스에서는 백라이트 및 진동 모드의 설정 값을 검색하여 임계치를 둠으로써 공격을 방어하고, 임계치를 벗어나는 경우에 경보를 발생한다. 시스템 명령어는 응용프로그램의 상태와 권한을 검증하여 시스템 명령어에 대한 접근을 통제한다. 이와 유사한 방법으로 파일에 대한 접근도 미리 정해진 권한을 검증하는 방식으로 통제하며, 이러한 규칙을 위반하는 사항에 대해 침입으로 간주하고 경보를 발생한다. 네트워크 공격에 대한 탐지는 전체 통신 상태가 종료된 이후에 발생하는 어떠한 임의의 통신 시도도 차단하여 공격을 방어한다.

이를 본 논문에서 수행할 핸드셋 디바이스, 시스템, 파일, 네트워크에 대한 공격 시나리오로 정의하며, 그 관계를 도식화해서 나타내면 [그림 9]와 같고, 개략적인 코드는 [표 9]의 형태로 나타낼 수 있다.



[그림 9] 공격 시나리오

〔표 9〕 공격 형태 및 탐지 방법

순번	공 격 형 태	탐 지 방 법

1	mIDbackLight a = new mIDbackLight( );
	// 조명을 이용한 배터리 소모 공격
	a.on(0,0x0000ff,20000000);
	// 단말기 조명을 지속적으로 켬
	a.alwaysOn( );
		public void on(int id, int color, int duration)
		{ if(duration < 600000) BackLight.on(id,color,duration);
		else alarm("mIDbackLight detect1 "); }
		public void alwaysOn( )
		{ BackLight.on(0,0xff0000,600000);
		alarm("mIDbackLight detect2 "); }

	// 진동을 이용한 배터리 소모 공격
	mIDvibrator b = new mIDvibrator( );
	b.on(99,0);
		public void on(int level, int duration)
		{ if (duration==0) {
		alarm("mIDvibration detect1 ");
		}else if(duration > 60000) {
		alarm("mIDvibration detect2 ");
		duration = 60000;
		Vibrator.on(level,duration); }
		else Vibrator.on(level,duration); }

2	// 시스템 명령어를 통한 정보 획득 시도
	mIDhandsetProperty h =
	new mIDhandsetProperty( );
	h.mIDgetSystemProperty("ESN");
	h.mIDgetSystemProperty("NID");
	h.mIDgetSystemProperty("SID");
	h.mIDgetSystemProperty("BASEID");
	h.mIDgetSystemProperty("CURRENTCH");
	h.mIDgetSystemProperty("PHONENUMBER");
	h.mIDgetSystemProperty("BATTERYLEVEL");
		public String mIDgetSystemProperty(String command)
		{ mIDS m = new mIDS( );
		String PrgName = Kernel.getPrgName( );
		if(m.ApplicationState(PrgName).equals("active") &&
		m.ApplicationAuth(PrgName,"SYSTEMCOMMAND")) {
		String tmp
		= HandsetProperty.getSystemProperty(command);
		alarm("mIDhandsetProperty detect1 ");
		return tmp; }else {
		alarm(" mIDhandsetProperty detect2 "); abort( ); } }

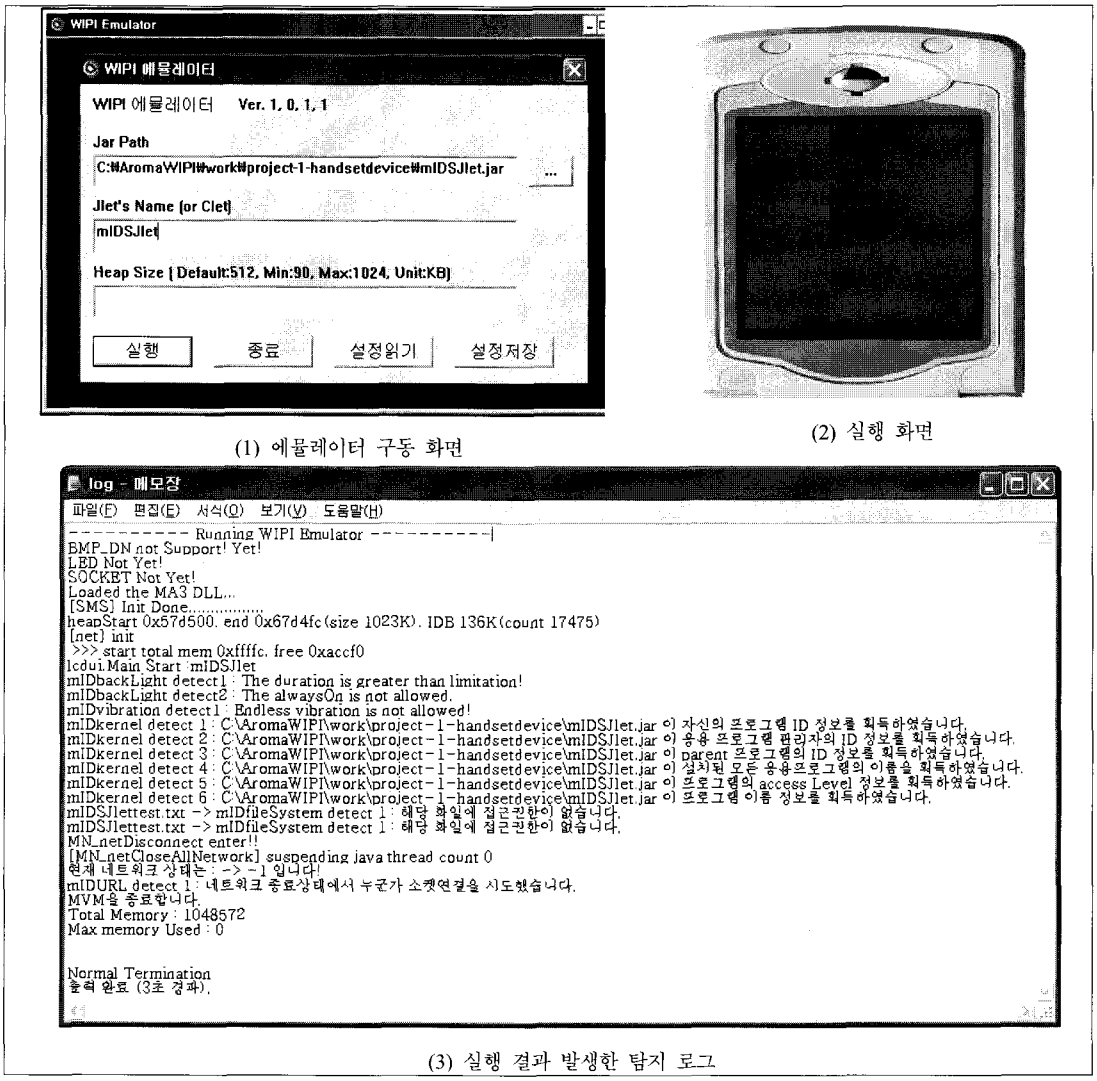
3	mIDfileSystem f = new mIDfileSystem( );
	// 개인 파일 접근 시도
	f.exists("secret.txt",f.PRIVATE_ACCESS,"mIDSJlet");
	// 파일 정보 획득 시도
	f.isFile("secret.txt",f.PRIVATE_ACCESS,"mIDSJlet");
		public boolean
		exists(String name, int flag, String pName)
		throws IOException, SecurityException
		{ boolean ret =false;
		if(mIDfileAuth(pName,name,flag)){
		try{ ret = FileSystem.exists(name,flag); alarm(ret); }
		catch (SecurityException e){ alarm("mIDfileSystem detect
		1");} } return ret;}

4	//임의의 네트워크 통신 시도
	mIDnetwork n = new
	mIDnetwork( );
	n.connect( );
	n.disconnect( );
	mIDURL U = new mIDURL( );
	String tmp = "http://www.naver.com";
	U.find(tmp);
		public Socket find(java.lang.String url)
		throws SchemeNotFoundException
		{ if (n.getNetworkStatus( )==0)
		//현재의 네트워크 상태를 통한 침입판단
		{ return URL.find(url); }else {
		alarm("mIDURL detect 1 ");
		return null; } }

4.3. 탐지 결과 및 분석

본 논문에서 제시한 mIDS의 논리적 구현 모델을 기반으로 시나리오에 따라 공격을 수행하고, 이를 탐지하는 과정을 에뮬레이터 상에서 시뮬레이션 했다.

시뮬레이션에 사용한 도구는 문서 편집기인 Edit Plus와 (주)아로마소프트에서 지원하는 AromaWIPI 에뮬레이터 WIPIEmul.exe를 사용하였다.



(1) 에뮬레이터 구동 화면

(2) 실행 화면

(3) 실행 결과 발생한 탐지 로그

〔그림 10〕 mIDS 동작 시뮬레이션 결과

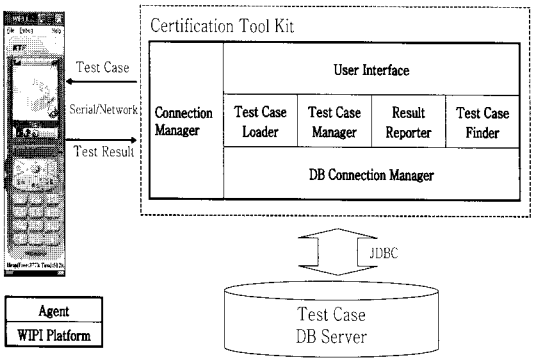
시뮬레이션 결과는 그림 10의 (1)과 같이 WIPI 에뮬레이터를 구동하여 공격 프로그램을 실행하여 (2)와 같은 화면을 얻었으며, 그 결과 (3)의 탐지 로그가 출력되는 것을 확인 할 수 있었다.

V. 결 론

본 논문에서는 향후 해킹 및 악성코드의 주 공격대상이 휴대용 단말기가 될 것에 대비하여 국내 무선인터넷 표준 플랫폼인 WIPI에 적합한 Specification 기반의 침입탐지시스템을 제안하였다. 보다 근본적으로 하위 수준에서 정보와 자원을 보호하기 위해 WIPI HAL의 시스템, 단말리소스, 핸드셋 디바이스,파일, 네트워크 및 SMS API를 대상으로 HAL API의 동작을 감시하여 악의적 행동을 탐지할 수 있는 알고리즘을 도출하였으며, HAL에 mIDS API 그룹을 추가하는 세부 프로토타입을 정의함으로써 본 연구 결과를 WIPI HAL 표준에 채택 가능한 형태로 추상화하였다.

따라서 mIDS API가 향후 보완될 WIPI 표준에 채택될 경우 보다 신뢰성이 향상되고 보안에 강화된 모바일

플랫폼을 구현할 수 있을 것으로 기대된다. 그러나 본 논문에서 다룬 HAL 계층은 블랙박스화 되어있기 때문에 HAL API를 직접 개발하고 그 성능을 검증할 수는 없다. 이에 그 대안으로 자바 라이브러리를 이용하는 구현모델을 제시하였고, 부분적으로 적용 가능한 일부분만을 구현하여 검증하는 제약사항이 있었다. 향후 연구에서는 필요한 모든 부분을 구현하고, 그 결과를 HAL에 적용하여 [그림 11]과 같은 플랫폼 인증 툴킷(PCT : Platform Certification Toolkit)[11]을 이용해 그 기능과 성능을 검증할 필요가 있을 것으로 판단된다.



〔그림 11〕 PCT overview

참고문헌

[1] 윤승노, "Study on the Possibility of Malicious Code Infection on WIPI Mobile Devices", 연세대학교 석사학위논문, 2006.

[2] 한국정보보호학회, "차세대 네트워크 보안기술", 한국정보보호진흥원, 2002. 11.

[3] 박진용, "정책기반 망에서 침입탐지에 대한 연구", 대구카톨릭대학교 석사학위논문, 2002.

[4] 민욱기, 장혜영, 최종천, 조성제 "다단계 구조를 가진 침입 탐지 및 방어 시스템의 구현", *정보과학학회 한국컴퓨터종합학술대회 논문집*, pp. 136-138, 2005.

[5] "TTA.KO-06.0036 모바일 표준화 규격", 정보통신단체표준, 2004. 6.

[6] 이상윤, 김선자, 김홍남 "한국 무선 인터넷 표준 플랫폼(WIPI)의 표준화 현황 및 발전 전망", *정보과학회지*, 제22권 제1호, pp. 16-23, 2004. 1.

[7] 이상윤, 이환구, 김우식, 이재호, 김선자, 김홍남 "무선 인터넷 표준 플랫폼 WIPI 2.0의 표준화 동향", *전자통신동향분석* 제19권 제5호, pp. 143-149, 2004. 10.

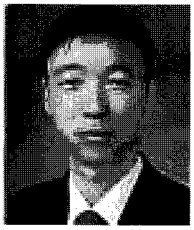
[8] 이상윤, 김선자, 김홍남, "무선인터넷 표준 플랫폼 WIPI2.0", *TTA저널*, 제92호, pp. 97-102, 2004. 3.

[9] 김학수, "WIPI 단말 구조 및 HAL의 이해", 2006 KWISF & WIDEF Technique Conference, 2006. 9.

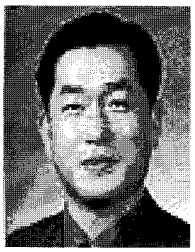
[10] Radmilo Racic, Denys Ma, Hao Chen, "Exploiting MMS Vulnerabilities to Stealthily Exhaust Mobile Phone's Battery", *Securecomm and Workshops*, pp. 1-10, 2006. 8.

[11] 이재호, 김선자, 이상윤, "Embedded Linux-based smartphone platform for sharing WIPI contents", *IT-SOC2004(Seoul, korea)*, pp. 550-553, 2004. 10.

〈著者紹介〉



김 익 재 (Ik-jae Kim) 학생회원
1995년 2월 : 공군사관학교 전산과학과 졸업
2006년 1월~현재 : 국방대학교 전산정보학과 석사과정
<관심분야> 침입탐지 시스템, 모바일 보안



이 수 진 (Soojin Lee) 종신회원
1992년 2월 : 육군사관학교 전산학과 졸업
1996년 2월 : 연세대학교 컴퓨터과학과 석사
2006년 2월 : 한국과학기술원 전산학과 박사
2006년 9월~현재 : 국방대학교 전산정보학과 교수
<관심분야> 침입탐지 시스템, 모바일 웹 보안, USN 보안