Journal of Korea Multimedia Society Vol. 13, No. 8, August 2010(pp. 1194-1201)

유비쿼터스 컴퓨팅을 위한 임베디드 운영체제 보안 기술 연구

박 종 혁†

요 약

현재 임베디드 시스템은 디지털 셋톱박스, 모바일 폰, USN 등 유비쿼터스 환경에 응용을 위해 널리 이용되고 있으며 필수적으로 내장된 것인 만큼 보안의 중요성 또한 점점 확대되어가고 있는 추세이다. 본 논문은 지난 2006년 12월에 제정된 국내 TTA 표준인 임베디드 운영체제 보안 참조 모델을 기반으로 주요 보안 요구사항 중 하나인 타겟시스템에서의 새로운 무결성 검증 방법을 제안하고자 한다. 제안된 방법은 기존 [2,6]에서 제안한 방법에 비해 향상된 효율성을 제공한다.

A Study on Embedded Operating System Security Technology for Ubiquitous Computing

Jong Hyuk Park†

ABSTRACT

Currently embedded system has been widespreadly used in digital Set-top box, mobile phone, USN, etc and the significance of security has been increased due to be necessarily embedded in these all system. In this paper we propose new integrity verification scheme among the main security requirements in target system based on the korea TTA standard, security reference model for embedded operating system, published in december 2006. Moreover the proposed scheme is more effective than the previous scheme, Jung, et al.[2,6].

Key words: Embedded System(임베디드 시스템), Operating system(운영체제), Ubiquitous Computing (유비쿼터스 컴퓨팅)

1. 서 론

임베디드 시스템은 어떤 제품이나 솔루션에 추가로 탑재되어 그 제품 안에서 특정한 작업을 수행하도록 하는 솔루션을 말한다. 예를 들어 주된 용도가 통화인 휴대폰에 텔레비전 기능이 들어가 있다면, 텔레비전 기능(시스템)이 바로 임베디드 시스템이다. 첨단 기능이 들어 있는 컴퓨터, 가전제품, 공장자동화 시스템, 엘리베이터, 휴대폰 등 현대의 각종 전자·정보·통신 기기는 대부분 임베디드 시스템을 갖추고 있다. 대개의 경우 그 자체로 작동할 수도 있지만, 다른 제품과 결합해 부수적인 기능을 수행할 때에 한해 임베디드 시스템이라고 한다. 컴퓨터의 경우에는 전용 동작을 수행하거나 특정 임베디드 소프트웨어 응용 프로그램과 함께 사용되도록 디자인된 맞춤형 컴퓨터 시스템 또는 컴퓨팅 장치를 일컫는다. 컴퓨터 외에 텔레비전·전기밥솥·냉장고·자동차 등에 내장되어 있는 웹 기능 등도 모두 임베디드 시스템이다[1].

※ 교신저자(Corresponding Author) : 박종혁, 주소 : 서울 노원구 공릉2동 서울과학기술대학교 컴퓨터공학과 325호 박종혁교수연구실(139-742), 전화 : 02)970-6702, FAX : 02)977- 9441, E-mail : jhpark1@snut.ac.kr

접수일 : 2009년 7월 18일, 수정일 : 2010년 3월 25일
완료일 : 2010년 5월 11일
†종신회원, 서울과학기술대학교 컴퓨터공학과 조교수

이러한 임베디드 시스템은 일상생활과 매우 밀접한 관계에 있으며 그 응용범위가 매우 넓어 현재 이용되고 있는 데스크탑 PC에서의 홈쇼핑이나 인터넷 뱅킹 이용 시에도 이에 따른 중요 개인 정보가 임베디드 시스템 내에 저장되고 있는 실정이다. 따라서 이러한 임베디드 시스템 내에서 개인 정보보호를 위한 임베디드 운영체제의 보안 기술은 그 사용의 증가와 비례하여 매우 중요해 질 것으로 예상된다[2].

임베디드 시스템의 구성은 크게 임베디드 운영체제, 임베디드 미들웨어 및 보안 시스템, 임베디드 기본/공통 응용 소프트웨어, 그리고 임베디드 소프트웨어 개발 도구로 나뉠 수 있다. 여기서 임베디드 시스템을 보안 관점에서 고려해 보면 임베디드 운영체제에서 커널에 대한 보안과 임베디드 미들웨어 부분에서 호스트와 타겟 임베디드시스템 간의 안전한 통신을 위한 보안시스템이 그 대상이라 할 수 있다[3]. 본 논문에서도 이들 부분에 대한 보안성을 보다 효율적으로 강화하기위한 새로운 방법을 개발하고자 하는 것이 그 목적이다.

한편 임베디드 시스템은 기존의 서버나 데스크탑 PC에서의 보안 요구사항과는 달리 제한된 성능 및 가용자원으로 인해 특성상 특정 응용에 맞는 시스템 개발이 각기 요구되며 보안 요구사항 또한 이에 맞추어 재구성해야 한다. 그러나 특정 응용 분야와 관계없이 기본적으로 요구되는 보안 요구사항은 아래와 같이 크게 세 가지로 요약될 수 있다[4,5].

- 인증(Authentication) : 네트워크 상에 원격지에 있는 개체(서버, 호스트, 또는 타겟 임베디드 시스템 등)가 정당하게 허가받은 개체인지를 확인할 수 있어야 함.
- 데이터 무결성(Data integrity) : 전달받은 데이터를 인가되지 않은 방법으로 변경할 수 없도록 증명할 수 있어야 함.
- 기밀성(Confidentiality) : 각 개체 간에 전달되는 데이터를 당사자 이외에는 볼 수 없어야 함.

본 논문에서는 위의 세 가지 보안 요구사항과 관련하여 인증과 기밀성에 대한 부분은 기존에 제안된 [6]의 방법을 그대로 이용하고 있다. 따라서 임베디드 시스템 즉 타겟 시스템 의 커널 내부에서 다운로드 된 응용 프로그램의 무결성을 보장하기위한 것이 본 연구의 범위이자 목적이다.

현재 이용되고 있는 임베디드 운영체제는 기존의 모바일 임베디드 운영체제와는 달리 PC 및 서버에서 주로 사용된 범용 운영체제를 시스템에 맞게 적용한 까닭에 기존에 존재하던 보안 위협들이 그대로 옮겨질 수 있다. 특히, 임베디드 리눅스를 위협하는 악성 프로그램들은 주로 ELF(Excutable and Linking Format) 형식의 바이너리 파일을 변형시켜 악성 코드를 삽입, 실행하려 한다. 이러한 방법은 다른 파일을 감염시켜 자가 복제를 계속 발생시키는 고전적인 형태의 바이러스외에 웜, 트로이 목마, 루트킷 등 다양한 공격에서 활용된다[7]. 이러한 위협에 대비하여 기존의 논문 [7]에서는 임베디드 시스템에 쉽게 적용이 가능하며 오버헤드를 최소한으로 줄일 수 있는 ELF 바이너리 파일 대상 악성 프로그램 실시간 감시 방법으로 커널 레벨에서 서명검증 기법을 이용한 악성 프로그램 차단 시스템을 제안한 바 있다. 이 방법은 ELF 파일의 변조 여부로 공격을 판단하기 때문에 인증을 받은 정상적인 프로그램 내부에 존재하는 취약점을 통해 직접 쉘 코드를 실행시키는 형태의 공격을 방어하지 못한다는 단점을 내포하고 있다. 제안하는 방법은 이러한 위협에 대비하면서도 기존 논문 [7]의 단점을 극복할 수 있는 방안을 제시하고자 한다.

지난 2006년 12월 정보통신기술협회인 TTA (Telecommunication Technology Association)에서는 임베디드 시스템에서 이러한 보안 요구사항들을 모두 만족하는 임베디드 운영체제 보안 참조모델 표준[8]을 제정하여 배포한 바가 있다.

표준규격에서는 임베디드 보안 프레임워크의 환경을 타겟 시스템에서 운영되는 보안 커널 모듈 및 타겟 시스템 보안 프로그램 모듈과 호스트 시스템에서 실행되는 호스트 시스템 보안 프로그램 모듈로 구성하여 각 모듈별 기능 요구사항들에 대해 기술하고 있다. 이들에 대한 구체적인 사항들은 [8]을 참조하기 바라며, 본 논문에서는 이들 가운데 호스트 시스템에서 응용 바이너리(프로그램)의 서명과 서명된 응용 바이너리를 타겟으로 전송하는 호스트 시스템 보안 프로그램 모듈과 타겟 시스템의 보안 커널 모듈에서 서명된 응용프로그램에 기록된 서명을 검증하여 무결성을 보장하는 서명 검증 모듈의 기능에 대해 그 범위를 한정한다.

그동안 운영체제를 위한 보안 기술과 관련하여 서버급이나 데스크탑 PC에서 기존의 운영체제에 보안

성을 강화한 사례(예, NSA의 범용 SELinux, (주)시큐브레인과 한국전자통신연구원의 SecureOS 등)는 국내외적으로 깊이 있게 진행되어 왔으나 임베디드 운영체제를 위한 보안 기술과 관련하여서는 특별히 국내외적으로 관심을 가지고 개발된 사례가 없었으나 지난 2006년 한국전자통신연구원에서 앞서 언급한 국내 TTA의 표준 규격을 만족하면서 동시에 임베디드 운영체제 보안 시스템 전반에 관해 솔루션을 제공하는 기술이 개발된 바 있다[2,6,9].

본 논문에서는 한국전자통신연구원에서 개발한 [2]와 [6]의 보안 프레임워크를 기반으로 그들이 제안한 임베디드 운영체제 보안 서비스 미들웨어와 관련, 타겟 시스템에서 응용 바이너리의 무결성을 보장하기 위한 방법에 대해 보다 효율적이고 새로운 방법을 제안하고자 한다. 또한 제안 방식은 [2]와 [6]에서 제안한 방법과 동일하게 국내 표준 규격을 만족한다.

본 논문의 구성은 다음과 같다. 2장에서 TTA 표준인 임베디드 운영체제 보안 참조 모델과 기존에 한국전자통신연구소에서 개발한 무결성 보장 방법에 대해 살펴보고 3장에서 보다 효율적인 방법을 제안, 분석한 다음 4장에서 제안 방식에 대한 유비쿼터스 환경에의 응용 방안에 대해 고찰한 후, 5장을 끝으로 결론을 맺고자 한다.

2. 관련 연구

2.1 임베디드 운영체제 보안 참조 모델

본 모델은 그림 1에 있는 바와 같이 호스트 시스템과 타겟 시스템으로 구성되며 호스트 시스템에서는 타겟 시스템에서 동작할 응용 프로그램을 생성하고 타겟시스템에 전송해 주는 역할을 수행한다[7]. 이때 호스트 시스템 내 보안 프로그램 모듈은 타겟 시스템에서 동작하는 응용 프로그램을 생성 한 후, 그 응용 프로그램에 서명과 태그를 부여하는 작업을 수행한다. 또한 타겟 시스템의 보안 프로그램 모듈은 서명된 응용 프로그램을 다운로드하고 해당 서명된 응용 프로그램에 대한 접근 제어에 필요한 정보를 커널 측에 전달하는 역할을 수행한다.

보안 커널 모듈은 임베디드 시스템용 운영체제 커널에 접근제어모듈과 서명검증 모듈이 추가되며 이때 서명 검증 모듈은 서명된 응용 프로그램이 실행되는 시점에 그 서명된 응용 프로그램에 기록된 서명을 검증하여 무결성을 보장한다.

호스트 시스템의 보안 프로그램 모듈은 그림 2에 나타난 바와 같이 호스트에서 개발된 응용 프로그램에 대한 전자서명과 서명된 응용 프로그램에 대한 접근제어에 필요한 정보인 태그를 추가해 주는 기능을 제공하며 보안 프로그램에 의해 서명된 응용 프로그램 형태로 제작되며 이 형태 그대로 타겟 시스템 쪽으로 전송된다.

타겟 시스템의 보안 프로그램 모듈은 그림 3에 나타난 바와 같이 호스트 시스템 보안 프로그램과 연동하여 호스트로부터 서명된 응용 프로그램 및 공개키를 다운로드 받고 커널이 사용 가능한 포맷으로 접근제어테이블을 작성하여 커널에 제공한다. 다시 말해 서명된 응용 프로그램을 호스트 시스템 보안 프로그램으로부터 다운로드 받는 일과 네트워크 단의 무결성을 검증하기 위한 검증기능, 비휘발성 미디어에 저

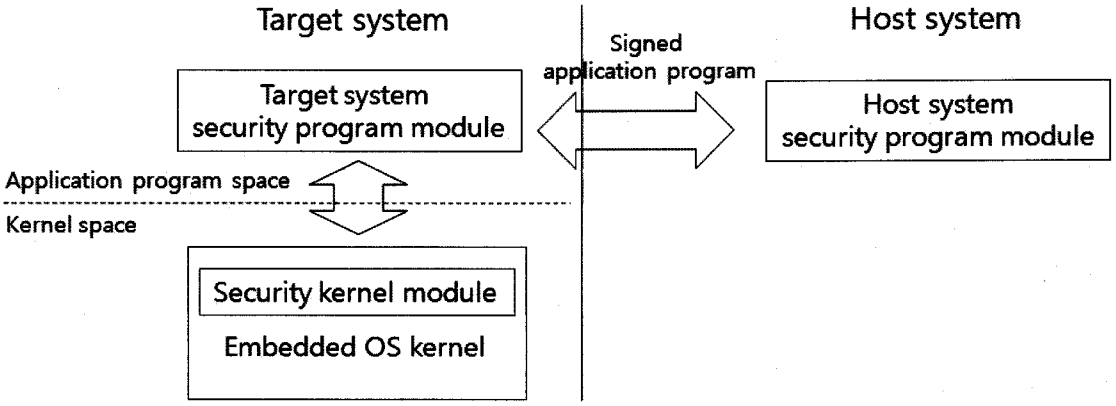

그림 1. 임베디드 보안 참조 모델

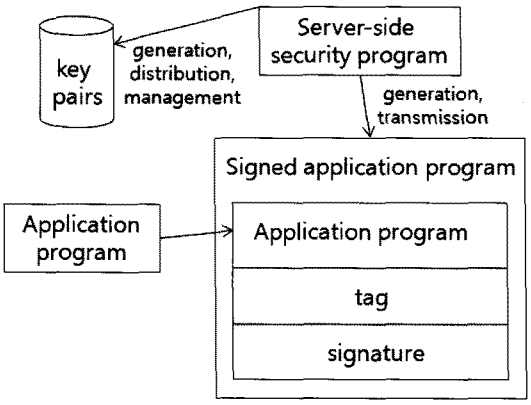

그림 2. 호스트 시스템 보안 기능

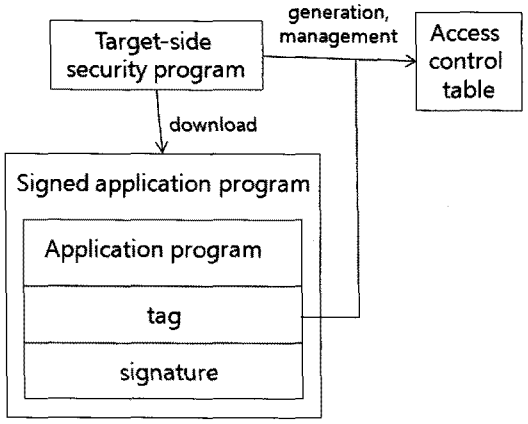

그림 3. 타겟 시스템 보안 기능

장된 접근제어정보테이블을 관리하는 기능, 이러한 모든 정보를 커널에 전달해 주는 기능을 제공한다.

본 논문에서는 이러한 참조모델을 기반으로 특히 무결성 검증 기능에 초점을 맞추어 새로운 방법을 제안하고자 한다.

2.2 응용 프로그램 사이닝/태깅

그림 4는 한국전자통신연구소에서 개발한 임베디드 운영체제 보안 기술에 관한 개략도이다. 본 논문은 그림 4에서 임베디드 보안 서비스 미들웨어 가운데 응용 프로그램 사이닝/태깅과 관련하여 보다 중점적으로 다루고자 한다.

한편 응용 프로그램 사이닝/태깅은 그림 5와 같이 임베디드 시스템에서 응용 프로그램을 응용 컨텐츠 서버에서 다운로드 받거나 임베디드 시스템에서 구동 시에 위조나 변조에 대응하여 무결성을 검증 할 수 있도록 하여 보안성을 극대화한 것이다[2,6].

개발 호스트에서 크로스 컴파일된 응용 프로그램은 응용 프로그램 바이너리 사이닝/태깅 모듈을 통해서 서명과 태그를 포함한 서명된 바이너리로 변화하게 된다. 먼저 사이닝/태깅 모듈을 통해서 임베디드 보안 운영체제의 접근제어 커널에 제공할 접근제어 정보를 새로운 ELF 섹션으로 바이너리에 기록하게 된다. 이 기록된 정보는 타겟에서 다운로드 받을 때 해석되어 임베디드 보안 운영체제의 접근제어 커널 모듈에 전달된다. 이는 그림 5의 ①의 순서에 해당한다. 태깅 작업이 끝나게 되면 서명 작업이 수행된다. 먼저 태깅이 완료된 바이너리의 해시 값을 생성한다. SHA-1(Secure Hash Algorithm) 해시 알고리듬을 이용하여 생성된 해시는 이미 호스트에서 생성된 개인키를 이용하여 RSA 전자서명 과정을 수행한다. 이 결과 값은 추가된 ELF 섹션으로 바이너리에 기록된다. 이는 그림 5의 ②의 순서에 해당한다. 개발 호스트에서 타겟으로 다운로드 된 프로그램은 검증/설정 모듈을 이용하여 설치 과정을 거친다. 설치 과정을 통하여 응용의 역할을 설정하고 다운로드 된 프로그램의 안전성을 확인하기 위함이다. 이는 그림 5의 ③의 순서에 해당한다. 이때 바이너리 검증/설정 모듈은 다운로드 된 프로그램의 안전성을 확인하기 위해 전송 받은 프로그램에서 ELF 헤더를 검색하여 추가된 ELF 섹션을 추출해 낸다. 먼저 전자 서명이 포함된 섹션을 제외한 파일 전체에 대한 현재의 SHA-1 해시 값을 구하고, 바이너리에 저장된 전자서명 값을 타겟 시스템에 저장된 공개키를 이용하여 RSA 전자서명 검증 과정을 거쳐서 앞에서 구한 현재의 SHA-1 해시와 비교한다. 검증과정을 통하여 전송과정에서 있을 수 있는 오류와 바이너리의 변조를 탐지해 낼 수 있게 된다(그림 5의 ④와 ⑤의 과정). 검증과정이 완료되면 바이너리에서 추출해낸 접근제어 태그의 정보는 해석되어 접근제어 커널 모듈에서 제공하는 시스템 콜을 통하여 전달된다(그림 5의 ⑥의 과정).

무결성 검증 모듈은 커널 모듈에 포함되어 프로그램의 실행 시에 바이너리에 포함된 서명을 검증하여 훼손된 바이너리의 실행을 차단하는 역할을 수행하는 그림 5에서 ⑦과 ⑧의 과정으로, 바이너리가 exec 시스템 콜을 이용하여 수행되면 커널 내의 LSM hook [10,11]이 불리게 된다. 이 때 무결성 검증모듈

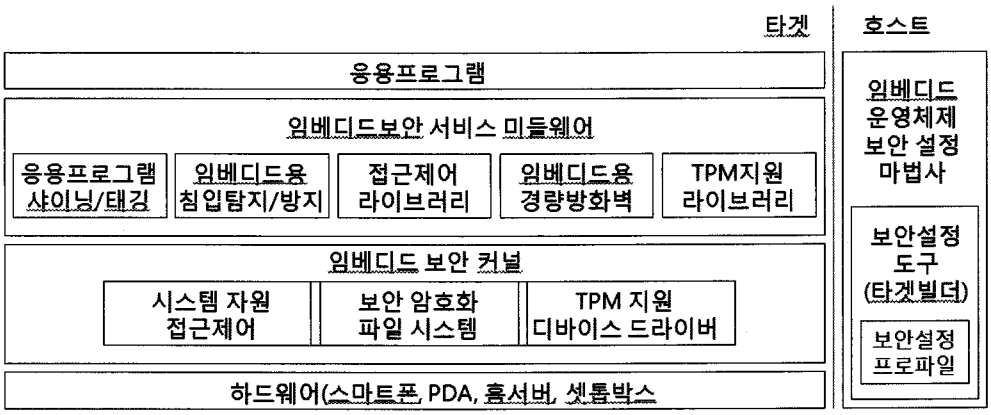

그림 4. 임베디드 운영체제 보안 기술 개략도

은 전자서명이 포함된 섹션을 제외한 파일 전체에 대한 현재의 SHA-1 해시값을 구하고, 바이너리에 저장된 전자서명 값을 커널에 로딩된 공개 키를 이용하여 RSA 전자서명 검증 과정을 과정을 거쳐서 앞에서 구한 현재의 SHA-1 해시와 비교하고, 변조된 응용은 실행을 차단하도록 한다. 이러한 검증 과정은 런타임에 수행하기에 비교적 오버헤드가 큰 작업이다. 그러므로 inode 구조에 서명과 검증 결과를 캐싱(caching)하는 방법을 사용하여 그 오버헤드를 줄일 수 있다. 시스템이 부팅된 후에 최초의 바이너리 실행에 대해서는 검증 과정을 수행한다. 그리고 바이너리의 현재의 SHA-1 해시 값과 그 결과는 inode에 캐시된다. 그 다음의 실행 시에는 inode의 변경이 없는 경우 캐시 값을 읽어서 실행/차단 여부를 결정하게 된다. 그리고 inode나 파일의 내용에 변화가 생기는 연산의 수행 시에는 파일의 변경 여부를 기록하고 캐시를 invalid로 변경하여 재실행 시에 서명 검증 과정을 수행하도록 하여 오버헤드를 줄일 수 있다.

그러나 제시된 방법은 기본적으로 호스트 시스템으로부터 타겟 시스템으로 서명된 응용 바이너리가 다운로드 될 때 타겟 시스템의 바이너리 검증/설정 모듈에서 이미 동일한 방법으로 해시와 서명 검증 과정이 한번 일어난 바 있다. 즉, 타겟 시스템의 바이너리 검증/설정 모듈을 통해 다운로드 된 응용 바이너리에 대한 위 변조 여부를 1차로 검증 한 후, 커널 내에서 응용 바이너리가 실행 시 동일한 방법으로 무결성 검증 모듈을 통해 2차로 실행된 바이너리의 위 변조 여부를 검증하고 있다. 물론 앞서 언급한 바

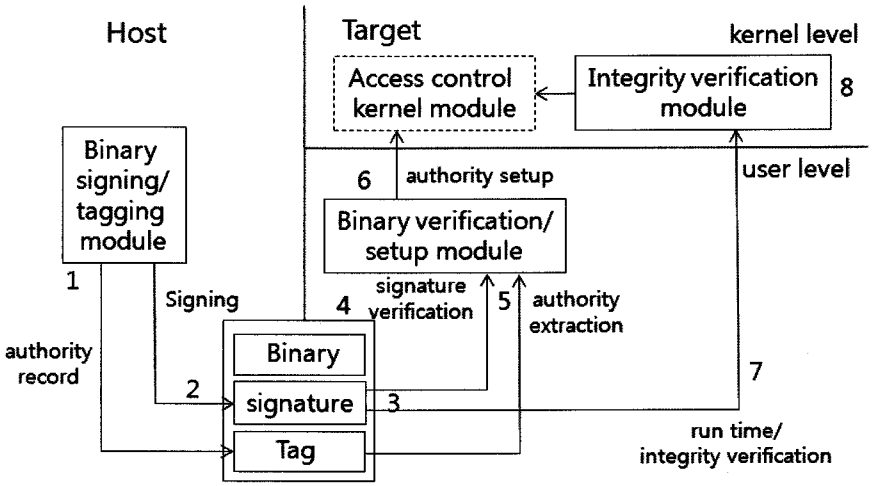

그림 5. 응용 프로그램 사이닝/태킹 메커니즘

대로 런타임 수행에 따른 오버헤드를 줄이기 위해 inode 구조에 캐싱하는 방법을 소개하고 있지만, 만일 inode나 파일의 내용에 잦은 변화가 발생할 경우 실행 시마다 서명 검증과정(해쉬, 서명검증, 비교연산을 매회 1번씩 수행)을 수행해야하기 때문에 많은 오버헤드가 발생하는 것을 피할 수 없다. 우리는 3장에서 위의 [6]의 방법을 기반으로 이러한 오버헤드를 줄이는 새로운 방법을 제안하고자 한다.

3. 제안 방식

3.1 무결성 검증을 위한 효율적인 방법

무결성 검증의 목적은 앞서 언급한 바대로 커널 내부에서 실행된 바이너리의 변조 여부를 검증하는 데 있다. 이전 [6]의 방법에서는 무결성 검증시 바이너리 검증과 동일한 방법으로 해시연산과 서명 검증연산을 통해 둘의 결과 값이 동일한 가를 비교하고 있다. 그러나 당초 서명에 대한 값의 경우 바이너리 값에 대한 무결성을 검증하기 보다는 사용자를 인증하는 목적이 강하다. 따라서 무결성 검증을 위해 제안 방식의 아이디어는 서명에 대한 검증 연산을 제외하는 대신 이전의 바이너리 검증/설정 모듈에서 전자서명이 포함된 섹션을 제외한 파일 전체에 대해 계산한 해시값(이를 H하자)을 바이너리 검증/설정 모듈에 미리 저장해 두었다가 커널 내부에서 바이너리가 실행시 무결성 검증 모듈에서 위와 동일하게 해쉬 값(이를 H′이라 하자)을 계산하여 값 H와 H′이 같은 가를 비교함으로서 연산량을 효율적으로 줄이고자 한다.

제안 방식은 아래와 같다(그림 6 참조)

[단계 1] Precomputation

먼저 바이너리 검증/설정 모듈을 통해 호스트 시스템으로부터 다운로드 된 바이너리의 검증이 완료(본 과정까지는 이전 방법 [6] 즉, 그림 5의 ①에서 ⑤과정, 제안한 방법인 그림 6의 ①에서 ⑤과정과 동일)된 후 현재의 SHA-1 해쉬 값(이를 H라 하자)을 바이너리 검증/설정 모듈에 저장(그림 6의 ⑥의 과정)한 후 그림 6의 ⑦의 과정(이전 방법 [6] 즉, 그림 5의 ⑥과정과 동일)을 계속 진행한다.

[단계 2] Integrity Verification

바이너리가 LSM hook을 통해 커널 내부에서 실행 시 무결성 검증 모듈은 전자서명이 포함된 섹션을 제외한 파일 전체에 대한 현재의 SHA-1 해쉬 값(이를 H′이라 하자)을 계산(그림 6의 ⑧과 ⑨의 과정)한 후 [단계 1]에서 저장된 값 H를 로딩하여 값 H=H′인지를 비교(그림 6의 ⑩의 과정)하여 만일 값이 다를 경우 변조된 응용이므로 차단한다.

위의 방법은 [6]에서 제안한 방법과 동일하게 런타임 수행의 오버헤드를 줄이기 위해 inode 구조에 서명대신 해쉬 값(H′)과 검증결과를 캐싱하는 방법을 이용하여 오버헤드를 줄일 수 있다. 응용에 따라 경량의 보안특성을 요구할 경우 무결성에 대한 검증과정을 매 실행시마다 수행하지 않고 랜덤하게 혹은

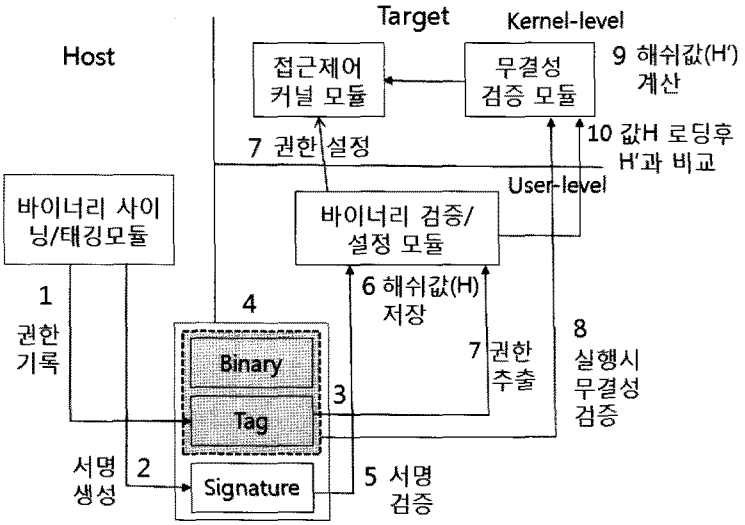

그림 6. 제안하는 무결성 검증 방법

일정 간격을 두고 실행할 수도 있을 것이다. 또한 기존 방법의 경우 inode 구조에 서명 값 대신 해쉬 값만을 캐싱하기 때문에 저장공간에 대한 오버헤드를 절약할 수 있을 뿐만 아니라, 특히 본 논문은 응용에 따라 inode 구조나 파일 내용에 변화가 잦을 경우 기존 방법에 비해 훨씬 효과적인 검증 성능을 제공할 수 있다.

3.2 안전성 및 효율성에 대한 고찰

일반적으로 임베디드 시스템에서 요구되는 보안 요구사항은 앞서 1장 서론 부분에서 언급된 바와 같이 인증, 무결성, 기밀성 세 가지이다. 이들 중 인증과 기밀성의 경우 호스트 시스템으로부터 타겟 시스템으로 바이너리를 전송하여 다운로드하는 과정에서 발생하는 것으로 본 논문에서는 앞서 [6]의 방법에서 제안한 방식(그림 5와 그림 6의 ①~⑤과정에 해당)을 그대로 적용하고 있다. 따라서 이들에 대한 안전성은 [6]을 참조하기 바란다.

본 논문에서 다루고 있는 무결성과 관련하여 제안 방식은 응용 바이너리에 대한 무결성을 보장하기위해 커널 내 무결성 검증 모듈을 통해 전자서명을 제외한 섹션에 대해 SHA-1 해쉬 값을 계산하여 기존에 바이너리 검증/설정 모듈에 저장된 SHA-1 값과 비교함으로써 그 무결성을 보장하고 있다. 따라서 무결성에 대한 보장은 암호학적인 해쉬함수의 안전성에 비추어 볼 때 이론적으로 안전한 것으로 증명되어 있다.

한편 타겟 시스템 측에서 바이너리 실행 시 무결성 과정에 대해 제안 방식을 기존의 [6]에서 제안한 방법과 계산량을 기준으로 비교하면 표 1과 같다. 표 1에서 나타난 바와 같이 제안 방식은 기존 방법 [6]에 비해 동일한 안전성을 보장하면서도 RSA를 이용한 전자서명의 과정을 생략할 수 있어 매우 효율적이다. 다만 그림 6의 ⑩의 과정에서 바이너리 검증/설정 모듈로부터 저장된 해쉬 값을 무결성 검증 모듈로 한번 더 로딩하는 과정이 추가되지만 이는 임베디드 시스템 특성상 각 모듈이 하드웨어로 구성되어 있기 때문에 기존의 RSA 서명 검증과정에 걸리는 시간과 비교해 볼 때 무시할 수 있을 정도이다. 또한 기존 방법 [6]의 경우 inode 구조에 저장되던 서명 값 대신 해쉬 값만을 캐싱하기 때문에 저장공간에 대한 오버헤드를 절약할 수 있다.

4. 제안 방식의 유비쿼터스 환경에의 응용 방안

임베디드 시스템은 컴퓨터 하드웨어와 소프트웨어 특히, 실시간 운영체제를 조합한 전자제어시스템으로 자동차나 컴퓨터, 가전, 혹은 특정 용도의 센서나 칩에 내장될 수 있다. 일례로 돼지고기에 컴퓨터 칩이 심어지고 이 칩이 스스로 전자레인지의 온도와 시간을 조절해 최적의 상태로 요리를 하거나, 스마트 센서가 달린 알약은 우리 몸속의 지정된 위치까지 정확하게 약을 운반해 주며, 냉장고가 남은 식품의 재고 상황을 파악하여 쇼핑리스트를 만들고 스스로 주문하거나 TV와의 커뮤니케이션을 통해 필요한 물품 광고를 가족들에게 보여주고 직접 선택하도록 하는 등 이러한 일들은 현재 실제 모습이거나 가까운 미래의 모습이다. 임베디드 시스템은 이러한 유비쿼터스 세상을 가능하게 만드는 요소기술로서 이 임베디드 시스템의 중추적인 역할을 담당하는 소프트웨어의 근간이 바로 임베디드 운영체제이다.

본 논문에서는 이러한 임베디드 시스템에서 핵심 기술 중 하나인 보안 기술로 임베디드 시스템 자체의 보안성을 강화함과 동시에 시스템 효율성을 향상 시킬 수 있는 방안에 대해 제시하였다. 이러한 향상된

표 1. 기존 방법 [6]과 제안 방식과의 비교

수행 시점	기존 방법 [6]	제안 방식
커널내부 바이너리 수행시	전자서명이 포함된 섹션을 제외한 파일 전체에 대한 SHA-1 해쉬 계산 1회	전자서명이 포함된 섹션을 제외한 파일 전체에 대한 SHA-1 해쉬 계산 1회
	RSA 전자서명 검증계산 1회	
	SHA-1해쉬 값과 전자서명 검증 값과의 비교 1회	SHA-1해쉬 값과 바이너리 검증/설정 모듈로부터 저장되어 로딩된 해쉬 값과의 비교 1회
inode에 저장되는 캐싱정보	서명값과 검증결과	해쉬값과 검증결과

방법은 임베디드 시스템에 탑재되어 앞서 언급한 바와 같이 유비쿼터스 세상을 이루는 다양한 분야에 응용될 것이라 확신한다.

5. 결 론

본 논문에서 우리는 임베디드 운영체제 시스템에서의 보안 기술과 관련하여 기존에 제시된 보안 요구사항들 중 타겟 시스템 특히, 보안 커널 내부에서 다운로드된 응용 프로그램인 바이너리의 무결성을 보장하기위한 효율적인 방법을 제시하였다. 제안한 방법은 이전에 제안된 [6]의 방법에 비해 RSA를 통한 서명 검증 과정을 제외하는 대신 단 한 번의 해쉬 연산과 비교만으로 동일한 안전성과 향상된 효율성을 제공하는 우수성을 가지고 있다. 즉, 제안하는 논문은 기존 방법의 RSA 서명에 대한 검증 연산에 비해 한 번의 해쉬연산 만으로 동일한 보안성을 가지면서도 보다 향상된 효율성을 제공하고 있다.

TTA 표준에 따르면 커널 내부에서의 보안 기능의 하나로 응용 바이너리의 무결성을 제시하고는 있으나 한편으로 타겟 시스템에서 서명된 응용 프로그램에 기록된 서명을 검증하는 것으로 무결성을 보장해야한다고 기술하고 있다. 따라서 향후 연구방향으로 이러한 표준 요구사항을 만족하기위한 보다 새로운 방법을 개발하는 것으로 그 연구를 진행하고자 한다. 현재는 기존 [6]의 방법에서 제시하고 있는 중량의 RSA의 서명 검증 방법을 보다 개선하기 위해 경량의 임베디드 시스템에 탑재 가능한 Rabin, NtruEncrypt, ECDSA, XTR 암호 프리미티브 등을 이용하여 가장 적합하고 효율적인 보안 프로토콜을 설계 중에 있다.

참 고 문 헌

[1] 네이버 백과사전, http://100.naver.com/100.nhn?docid=791706.

[2] 정영준, 임동혁, 서영빈, 김재명, "임베디드 운영체제 보안 기술 동향," 전자통신동향분석 제 23권, 제1호, 2008.

[3] "임베디드 시스템 구성요소 및 기능", *IT SoC Magazine*, 제19호, 2007.

[4] Syed Gilani, "Embedded OS: A Foundation for Secure Networking," *Embedded Computer Design*, 2007 OpenSystems Publishing, http://www.mentor.com, 2007.

[5] Hwang D.D, Schaumont P, Tiri K, and Verbauwhede I, "Securing embedded systems," *Security & Privacy, IEEE,* Vol. 4, Issue 2, pp 40-49, 2006

[6] 임동혁, 임용관, 정영준, 김재명, "임베디드 보안 운영체제를 위한 바이너리 보안 프레임워크의 설계," NCS, 2006. 12

[7] 이종석, 정기영, 정다니엘, 김태형, 김유나, 김종, "임베디드 리눅스에서 서명 검증 방식을 이용한 악성 프로그램 차단 시스템," 2007년 한국정보과학회 추계학술발표논문집, 제34권, 제2호, 2007.

[8] "임베디드 운영체제 보안 참조 모델", TTAS.KO-11.0054, 정보통신기술협회, 2006.

[9] 이재홍, 허준영, 박재민, 조유근, 홍지만, "Buffer Cache Level Encryption for Embedded Secure Operating System," EUC, 2007.

[10] WireX Communications, Linux Security Module, http://lsm.immunix.org/, 2001.

[11] Chris Wright, Crispin Cowan, Stephen Smalley, James Morris, and Greg Kroah Hartman, "Linux security module framework", In 2002 Ottawa Linux Symposium, 2002.

박 종 혁

2001년 2월 순천향대학교 컴퓨터공학부(공학사)

2003년 2월 고려대학교 정보보호대학원 정보보호학과(공학석사)

2007년 2월 고려대학교 정보보호대학원 정보보호학과 (공학박사)

2002년 12월~2007년 7월 한화에스앤씨(주) 기술연구소 선임연구원

2007년 9월~2009년 8월 경남대학교 컴퓨터공학부 전임강사

2009년 9월~현재 서울과학기술대학교 컴퓨터공학과 조교수

관심분야 : 디지털포렌식, DRM, 접근제어, 유비쿼터스 컴퓨팅 & 보안, 지능형 홈 서비스, 멀티미디어 보안 및 서비스, ISMS