Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.25, NO.4, Aug. 2015 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2015.25.4.863

TrustZone의 시큐어 타이머를 이용한 효율적인 커널 검사 시스템*

김 진 목,[1,2†] 김 동 욱,[1‡] 박 진 범,[1] 김 지 훈,[1] 김 형 식[2]
[1]삼성전자 소프트웨어센터, [2]성균관대학교

An Efficient Kernel Introspection System using a Secure Timer on TrustZone*

Jinmok Kim,[1,2†] Donguk Kim,[1‡] Jinbum Park,[1] Jihoon Kim,[1] Hyoungshick Kim[2]
[1]Software R&D Center, Samsung Electronics, [2]Sungkyunkwan University

요 약

커널 루트킷은 운영체제의 컴포넌트 사이의 통신을 가로채거나 수정할 수 있기 때문에, 운영 체제의 무결성을 훼손시킬 수 있는 가장 위협적이고 널리 퍼진 위협 중 하나로 인식되고 있다. 커널 루트킷이 이미 커널 권한을 획득하였기 때문에 루트킷이 설치된 공간에서 커널을 보호하는 것은 안전하지 않다. 따라서 커널보호 시스템은 커널과 동일한 공간으로부터 독립적이어야만 한다. 루트킷을 탐지하기 위해 많은 연구들이 수행되어 왔지만 다른 연구들과 달리 TrustZone 기반 연구는 커널과 동일한 공간으로부터 분리되고, 독립된 공간에서 커널을 보호하는 것이 가능하다. 하지만 제안된 방법들은 커널보호 시스템을 완전히 독립시킬 수 없는 단점이 있다. 이러한 이유로, 우리는 TrustZone의 시큐어 타이머를 이용한 효율적인 커널 검사 시스템을 제안한다. 이 시스템은 레퍼런스의 무결성을 보장하기 위해 커널 원본 이미지인 vmlinux을 활용하여 측정하였다. 또한, 보호영역 크기에 대한 유연성을 제공함으로써 효율적으로 커널보호 시스템을 운영하는 것이 가능하다. 실험 결과들은 제안된 커널보호 시스템이 완전히 독립되어 운영되고, 런타임동안 최대 6%정도의 성능만 저하시킨다는 것을 보여준다.

ABSTRACT

Kernel rootkit is recognized as one of the most severe and widespread threats to corrupt the integrity of an operating system. Without an external monitor as a root of trust, it is not easy to detect kernel rootkits which can intercept and modify communications at the interfaces between operating system components. To provide such a monitor isolated from an operating system that can be compromised, most existing solutions are based on external hardware. Unlike those solutions, we develop a kernel introspection system based on the ARM TrustZone technology without incurring extra hardware cost, which can provide a secure memory space in isolation from the rest of the system. We particularly use a secure timer to implement an autonomous switch between secure and non-secure modes. To ensure integrity of reference, this system measured reference from vmlinux which is a kernel original image. In addition, the flexibility of monitoring block size can be configured for efficient kernel introspection system. The experimental results show that a secure kernel introspection system is provided without incurring any significant performance penalty (maximum 6% decrease in execution time compared with the normal operating system).

Keywords: Platform security, System security, TrustZone, Code injection, System call table hooking

접수일(2015년 7월 2일), 수정일(1차: 2015년 7월 27일, 2차: 2015년 8월 4일), 게재확정일(2015년 8월 4일)
* 본 연구는 삼성전자 소프트웨어센터의 지원 및 관리로 수행하였습니다.
† 주저자, jinmok.kim@samsung.com
‡ 교신저자, donguk14.kim@samsung.com(Corresponding author)

I. 서 론

커널 루트킷은 노출되지 않은 상태로 운영체제를 무력화시킬 수 있기 때문에, 운영체제의 발달에도 불구하고, 가장 위협적이고 널리 퍼진 커널 위협 중에 하나로 인식되어 왔다. 루트킷 코드는 보안 취약점을 이용하여 이전에 삽입되었던 디바이스 드라이버에 의해 커널 메모리에 삽입된다. 삽입된 루트킷이 실행되면 커널 코드, 데이터, 구조체의 변조 및 함수 포인터 후킹을 통해 공격자가 의도한대로 오작동이 유발된다. 불행하게도, 커널 루트킷은 스마트폰, 스마트 TV등과 같은 스마트 기기까지 설치가 가능하고 그 위험성이 증가되어 가고 있다. 따라서 루트킷의 위협을 탐지하기 위해, 스마트 기기 상에서도 커널보호 시스템의 필요성이 대두되고 있다.

커널 루트킷이 이미 커널 권한을 획득하였기 때문에 루트킷이 설치된 공간에서 커널을 보호하는 것은 안전하지 않다. 따라서 커널보호 시스템은 커널과 동일한 공간으로부터 독립적이어야만 한다. 이를 위해, 많은 연구들이 수행되어 왔으며, 다음과 같이 분류할 수 있다. 첫째로, 하이퍼바이저 기반 연구[1,2], 둘째로, 외부 하드웨어 모니터 기반 연구[3,4], 마지막으로 TrustZone 기반 연구[5,6]이다.

하이퍼바이저 기반 연구에서는 가상화를 통해 커널과 격리된 환경에 커널보호 시스템을 구성할 수 있으나, 하이퍼바이저 역시 소프트웨어 취약점을 가질 수 있으며[7,8], 하이퍼바이저에 대한 무결성을 어떻게 보장할 수 있는지 등에 대한 문제점을 가진다. 이와는 달리, 외부 하드웨어 모니터 기반 연구는 독립된 공간에서 커널 무결성을 체크하기 위해 외부 하드웨어 모니터를 사용하였다. 하지만 외부 하드웨어 모니터를 사용하기 위해서는 특별한 하드웨어 모듈이나 SoC가 필요하며, 이는 다른 시스템과의 연결을 위해 하드웨어 디자인의 복잡성을 증가시키거나 [4]에서 제기한 바와 같이 전력소모를 증가시킨다. 지금까지 소개된 방법들과는 달리 TrustZone 기반 연구는 normal world로부터 검사 요청을 받고, secure world에서 커널 메모리를 검사하는 방법을 사용하였다. 이 방법은 메모리 검사 시, 커널보호 시스템이 커널과 동일한 공간으로부터 분리되고, 독립된 공간에서 커널을 보호하는 것이 가능하다.

하지만, 이미 제안된 방법인 TZ-RKP[5]와 SPROBES[6]는 커널보호 시스템을 완전히 독립시킬 수 없다. Normal world에서 secure world로 메시지를 보냈을 때, 루트킷이 man-in-the middle 공격을 통해 이 메시지를 변조시킬 수 있기 때문이다[9,10]. 이러한 이유로, 우리는 TrustZone의 시큐어 타이머를 이용한 효율적인 커널 검사 시스템(EKI, Efficient Kernel Introspection system using secure timer on TrustZone)을 제안한다. EKI는 TrustZone의 시큐어 타이머를 활용하여, 외부 하드웨어 모니터와 같은 형태의 active monitoring을 수행한다. 시큐어 타이머는 TrustZone상에서 FIQ(Fast Interrupt reQuest)를 발생시켜, 커널 메모리 검사를 수행한다. 이는 normal world로부터 검사 요청을 받아서, TrustZone상에서 커널 메모리를 검사하는 방법을 사용하지 않으므로, 커널보호 시스템을 완벽하게, 독립적으로 분리시키는 것이 가능하다. 본 연구와 마찬가지로 TrustZone을 이용한 커널 인트로스펙션 기법[11]에서는 시큐어 타이머를 이용하는 아이디어가 제안되었으나, 실제 단말에서의 시큐어 타이머 구현 없이 실험을 진행하였다.

본 연구에서는 [11]의 아이디어를 확장하여 실제 단말 상에서 실험을 진행하였고, 단말의 보안과 성능을 효율적으로 관리하기 위해 세 가지 기능적 요소를 고려하였다. 첫째로, 커널의 원본 메모리 값을 나타내는 레퍼런스 측정값의 무결성 보장을 위해 커널 빌드 시, 커널 원본 이미지인 vmlinux를 이용하여 레퍼런스를 측정하였다. 또한, 한 번에 검사할 수 있는 블록의 크기에 대한 유연성을 제공하기 위해, 각 섹션의 보호 영역을 설정 가능한 크기의 여러 모니터링 블록으로 나누어 측정하였다. 이와 같이, 무결성과 유연성을 제공함으로써 단말의 용도, 성능과 커널의 크기를 고려하여 효율적으로 커널 보호 시스템을 운영하는 것이 가능하다. 둘째로, vmlinux를 이용한 레퍼런스 측정 시, 각 보호영역의 가상주소를 물리주소로 변환하는 방법을 제시하였다. 따라서 secure world에서도 normal world의 커널 메모리 데이터에 직접 접근하는 것이 가능하다. 셋째로, EKI의 독립적인 커널 검사 수행을 위해 TrustZone의 시큐어 타이머를 이용한 상태 모니터링을 수행하였다.

II. 배경지식

Fig. 1.에서 보는 바와 같이, TrustZone은 normal world와 secure world라는 2개의 가상 CPU 코어 제공 및 모니터를 통해 이 두 개의

world사이의 안전한 컨텍스트 스위칭 방법을 제공한다. 이 방법을 위한 보안정책이 프로세서, 버스, 메모리, 주변장치에까지 적용되어 있다. 이렇게 보안이 적용된 하드웨어들의 자원들은 normal world에서 동작할 자원과 secure world에서 동작할 자원이 사전에 구분되고, 하드웨어에 의해 각 world에 선택적으로 제공된다. 따라서 normal world에서 secure world로 모드를 전환하기 위해서는 SMC 인스트럭션을 이용하여 현재 수행되는 모드를 모니터 모드로 변경해야 한다. SMC 인스트럭션을 이용하는 방법 외에도, IRQ(Interrupt ReQuest)나 FIQ를 이용하여 모니터 모드로의 변경이 가능하다[12]. Secure world에서는 전체 메모리 영역을 확인할 수 있지만 normal world에서는 secure world쪽의 메모리 영역이 하드웨어에 의해 차단되기 때문에, 이를 통해, TrustZone이라는 안전한 실행환경을 제공할 수 있다.

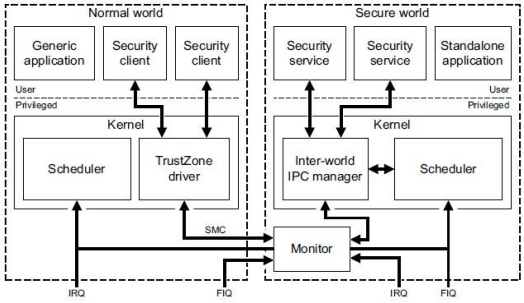

Fig. 1. ARM TrustZone Software Architecture[12]

III. 공격 모델

본 연구에서는 이미 단말에 커널 루트킷이 설치되어 있고, 이 사실을 유추할 수 없도록 루트킷의 존재를 숨긴다는 것으로 가정한다. 설치된 커널 루트킷은 normal world의 커널을 변조하여 운영체제를 완전히 제어할 수 있으나, ARM TrustZone에 의해 보호되는 secure world를 제어할 수 는 없다. 이들을 효과적으로 방어하기 위해서는 사전 분석을 통해 공격특성을 밝혀내고 공통적인 탐지 방법을 찾아내야 한다. 커널 루트킷들의 특성을 분석한 연구들 중 하나인, Poker[13]에서는 커널 루트킷을 다양한 측면에 프로파일링 하였다. 프로파일링된 결과에서 커널 루트킷의 기본적인 공격특성은 코드 인젝션을 통한 커널 코드 변조와 시스템콜 후킹인 것으로 확인되었다.

코드 인젝션의 경우, 실제 커널코드에 악성코드를 삽입하는 방법으로써, 삽입된 코드를 이용하여 직접적인 공격을 수행하거나 공격자가 만든 공격코드를 호출하게끔 한다. 이 방법은 초기 루트킷에서도 많이 사용한 방식으로써, 기본적인 공격특성이라고 볼 수 있다. 시스템콜 후킹 공격은 시스템콜 테이블 엔트리를 후킹하여 원래의 함수가 아닌 공격자가 만든 함수가 호출되도록 하는 공격이다. 이 공격은 실행 난이도가 낮으면서도 사용자 레벨에서 사용하는 함수를 직접적으로 공격 코드가 들어있는 함수로 대체할 수 있다는 점에서 공격효과가 크다고 볼 수 있다.

따라서 EKI는 대부분의 커널 루트킷들에서 가장 기본적이고 널리 사용되는, 코드 인젝션과 시스템콜 후킹에 대한 탐지를 목적으로 한다.

IV. 탐지 방법 설계

본 연구에서는 앞서 얘기한 외부 하드웨어 모니터와 TrustZone 기반 커널보호 방법의 장점 활용 및 기존 연구의 단점을 보완한 TrustZone에 기반을 둔 상태 모니터링 시스템을 설계하였다. EKI는 항상 ARM 기반 아키텍처의 TrustZone 환경 안에서 동작하기 때문에 normal world와 독립적으로 커널 무결성을 검증할 수 있으며, 이는 외부 하드웨어 모니터를 이용하는 것과 같은 장점을 가질 수 있게 해준다.

Fig. 2.는 EKI의 전반적인 아키텍처를 나타낸다. 빌드 시, 레퍼런스 해쉬 도구가 vmlinux를 이용하여 레퍼런스 측정값을 만들어 낸다. 런타임 시에는 secure world내에 trusted application(EKI application)과 디바이스 드라이버(EKI Driver) 형태로 EKI가 존재하여 모니터링을 수행한다. 단말의 운영체제는 secure boot을 통해서 안전하게 부팅이 완료되었다는 것을 가정하고, 부팅 이후에는 EKI에 의해 리눅스 커널 메모리에 대한 무결성을 검증 받게 된다. EKI는 이러한 3개의 주요 컴포넌트로 구성되어 커널에 대한 메모리 보호 영역을 지속적으로 모니터링 한다. 주요 컴포넌트의 기능은 다음과 같다.

(1) EKI 어플리케이션은 시큐어 타이머 동작 주기를 등록하는 커널 모니터링 시간 설정 및 커널 메모리의 보호 영역을 설정한다. 또한, 보호 영역의 데이터 값을 읽어서 런타임 측정값을 생성

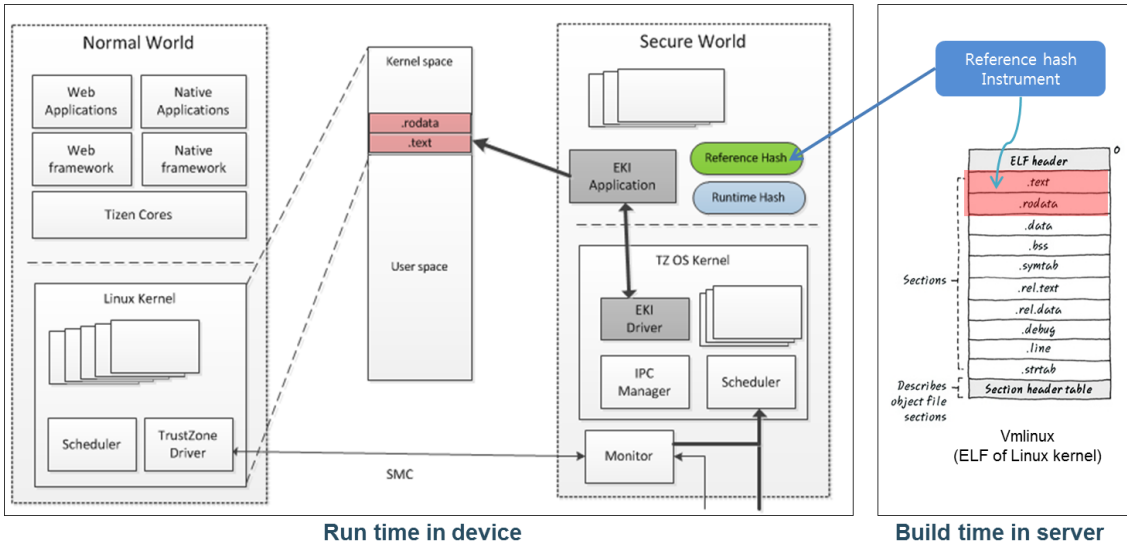

Fig. 2. EKI Architecture

하고, 이를 레퍼런스 측정값과 비교하는 기능을 수행한다.

(2) EKI 드라이버는 secure world내의 페이지 생성을 통해 물리메모리 주소를 secure world내의 가상메모리 주소로 변환하는 기능을 수행한다. EKI 어플리케이션은 보호 영역의 물리메모리 주소를 저장하고 있으나, 직접 접근하는 것은 불가능하기 때문에, EKI 드라이버를 통해 가상메모리 주소로 변환하는 과정이 필요하다. 또한, EKI 드라이버에서는 시큐어 타이머 기능이 구현되어 있어, 주기적인 FIQ를 발생시킨다.

(3) 레퍼런스 해쉬 생성도구는 커널 빌드 시점에 동작한다. 빌드 시, 커널 이미지(vmlinux), 커널 빌드 옵션(.config), 디바이스 트리 바이너리(dtb.bin)를 이용하여 레퍼런스 측정값을 만드는 기능을 수행한다.

이러한 컴포넌트들로 이루어진 EKI는 3가지 주요 기술적 요소를 고려하여 설계되었다. 아래는 각 주요 기술적 요소를 설명한 내용이다.

4.1 메모리 값 측정

첫 번째 기술적 요소는 측정값들을 얻고 이를 비교하는 방법이다. EKI는 커널 메모리 영역을 검사하기 위해서 두 가지 측정값이 필요하다. 하나는 커널의 원본 메모리를 나타내는 레퍼런스 측정값으로써, 완벽히 보안이 적용된 환경에서 생성되어야 한다. 그렇지 않다면, 측정값이 루트킷에 의해 변조될 수 있고, 이로 인해, EKI의 기능은 무력화 될 수 있다. 다른 하나는 런타임 측정값으로써, 이는 효율적으로 측정되어야 한다. EKI는 외부 하드웨어가 아니기 때문에, 오랜 시간 동안 런타임을 측정값을 생성하여 커널 메모리를 검사한다면, 운영체제의 성능저하를 크게 일으킬 수 있다. 따라서 런타임 측정값을 생성할 시에, 최소한의 성능저하만 일으키도록 효율적인 커널 검사를 수행해 한다. 이를 위해, 커널 보호 영역의 측정값 생성 시에 필요한 정보들을 미리 생성해 놓는 것이 중요하다. 커널 보호 영역에 대한 레퍼런스 측정값, 크기, 물리메모리 주소를 미리 저장해 놓는다면, 적은 계산량으로 커널 검사를 수행할 수 있다.

EKI에서는 위의 두 문제를 해결하기 위해 vmlinux를 활용하는 방법을 제안하고자 한다. Vmlinux를 활용하여 빌드환경에서 안전하게 레퍼런스 측정값을 생성하여 첫 번째 문제를 해결할 수 있다. 두 번째 문제를 해결하기 위해, 빌드 시, vmlinux를 통해 검사할 영역에 대한 전체크기를 사전에 파악함으로써, 런타임 시, 단말의 성능을 고려한 모니터링 블록의 크기설정을 가능하게 해준다. Vmlinux는 리눅스 커널 이미지가 포함된 ELF 실행파일이다. 리눅스 커널의 동작은 부트로더가

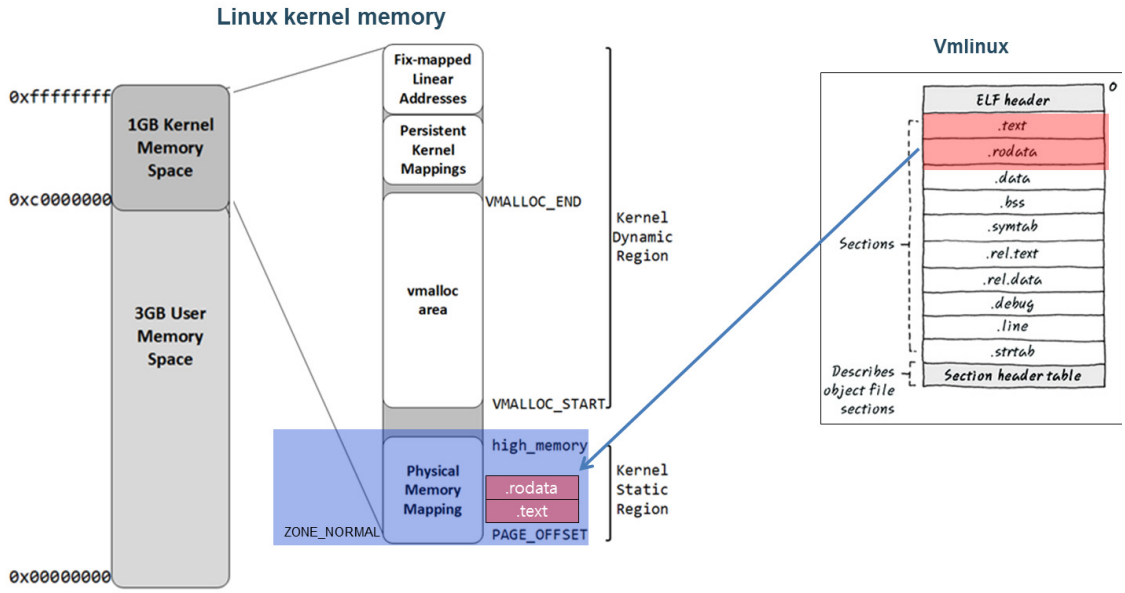

Fig. 3. Linux kernel memory layout

vmlinux를 압축한 uImage를 로딩 함으로써 시작하게 된다. Vmlinux는 일반적인 ELF 파일 포맷과 동일하게 구성되어 있다. 데이터의 특성에 맞게 여러 개의 섹션으로 구성되어 있고, 각 섹션별로 이름, 가상시작주소, 오프셋, 크기, 접근 권한 정보도 포함되어 있다. 섹션별 타입을 살펴보면 text 섹션은 실행 코드들이, rodata 섹션은 read-only initialized 데이터가 존재하는 영역이다. 또한, data와 bss 섹션은 각각 initialized/uninitialized read/write 데이터가 존재하는 영역이다. ELF 로딩 시, 각 섹션은 가상메모리 주소에 맞추어 메모리에 로딩 된다. 섹션들 중에서 코드 인젝션과 시스템콜 후킹 공격의 대상이 되는 영역은 text와 rodata 섹션이고, 이 부분에 대한 무결성을 보장하는 것을 EKI에서 수행한다.

Fig. 3.은 vmlinux 이미지와 런타임 시, 단말에서의 커널 메모리 배치의 관계를 나타낸다. Vmlinux의 text와 rodata 섹션은 리눅스 커널의 정적 영역인 ZONE_NORMAL 영역 안에서 연속적으로 로딩 된다. 리눅스는 가상 메모리 시스템이기 때문에 일반 프로세스의 주소영역은 페이지 단위로 관리된다. 따라서 가상메모리 주소가 연속적으로 이어져 있어도 실제 물리메모리 주소가 연속적으로 이어져 있다고 보장할 수 없다. 하지만 리눅스 커널의 ZONE_NORMAL 영역은 특별한 메모리 관리 영역으로, 가상메모리와 물리메모리가 일대일로 매칭되는 영역이다. 이 속성에 의해, vmlinux의 text와 rodata 섹션의 데이터는 런타임 시에도 페이지 별로 물리메모리 주소가 분산되어 있지 않고 연속적으로 위치하게 된다. Normal world 내의 리눅스 커널 메모리 영역과 분리되어 있는 TrustZone내의 EKI에서도 각 섹션의 물리메모리 시작 주소를 저장하고 있으면 동일한 데이터에 접근 가능하다. 따라서 vmlinux의 text와 rodata 섹션에 대한 레퍼런스 측정값을 해쉬 알고리즘을 활용하여 미리 생성하고, 런타임 시, 각 섹션이 로딩 되어 있는 물리메모리에 접근하여 런타임 해쉬 측정값을 생성, 이를 레퍼런스 측정값과 비교하여 무결성 검증을 수행할 수 있다.

런타임 측정값을 효과적으로 측정하기 위해서, EKI는 보호영역을 여러 모니터링 블록으로 나누고, 각 모니터링 주기마다 한 블록씩만 측정값을 생성한다. 만약 블록의 크기가 크다면, 짧은 시간에 많은 영역을 검증할 수 있지만 운영체제의 성능저하는 커지게 된다. 보안성과 성능간의 tradeoff 관계가 형성되는데, 블록의 크기는 미리 EKI에서 설정할 수 있으므로, 각 운영체제의 환경에 따라 효율적인 측정이 가능하다.

레퍼런스 측정값을 실제 단말에 적용하기 위해서 레퍼런스 해쉬 생성도구라는 컴포넌트를 구현하였다. 이 컴포넌트는 커널 빌드 시스템에 삽입되어, 리눅스

커널 빌드가 완료된 후에, 자동으로 vmlinux의 레퍼런스 측정값을 생성할 수 있도록 하였고, 작동 과정은 다음과 같다.

(1) Vmlinux를 분석하여 text와 rodata 섹션의 크기를 파악하고 미리 설정된 크기의 모니터링 블록으로 나눔
(2) 각 블록에 대한 가상메모리 시작 주소를 물리메모리 시작 주소로 변환
(3) 각 블록의 데이터에 대한 해쉬 측정값을 생성
(4) 각 블록에 대한 물리메모리 시작주소, 크기, 해쉬 측정값을 레퍼런스 측정값으로 저장

런타임 시에는 EKI 어플리케이션에서 이 정보를 바탕으로 리눅스 커널 메모리의 블록 단위로 무결성 검증을 수행하게 된다.

4.2 메모리 주소 변환

리눅스 커널의 ZONE_NORMAL 영역은 가상메모리와 물리메모리가 일대일로 매칭된다는 속성을 이용하여, 변환할 가상메모리 주소, 리눅스 커널의 물리메모리 시작 주소인 PHYS_OFFSET, 리눅스 커널의 가상메모리 시작 주소인 PAGE_OFFSET를 이용하면 수식 (1)에 대입하여 가상메모리 주소와 매칭되는 물리메모리 주소를 계산해 낼 수 있다.

$$\text{물리주소} = \text{가상주소} - \text{PAGE\_OFFSET} + \text{PHYS\_OFFSET} \quad (1)$$

PAGE_OFFSET은 커널 빌드 옵션 설정 파일인 .config에 존재하고, PHYS_OFFSET 정보는 하드웨어 정보가 포함된 dtb.bin에 존재한다. 위의 정보를 바탕으로 레퍼런스 해쉬 생성도구에서 각 블록에 대한 가상메모리 시작 주소를 물리메모리 시작 주소로 변환하여 레퍼런스 측정값에 다른 값들과 함께 저장하게 된다.

레퍼런스 측정값 생성 시뿐만 아니라 런타임 시에도 메모리 주소 변환과정이 필요하다. ARM에서는 각 world별로 메모리 변환 테이블을 따로 가지고 있고, secure world내의 trusted application은 물리메모리에 바로 접근할 수 없다. 따라서 EKI 어플리케이션이 물리메모리 주소에 접근하여 데이터를 읽으려면, 먼저 물리메모리 주소를 secure world내에 있는 페이지로 매칭시켜 가상메모리 주소로 변환시켜야 한다. 이를 위해, EKI 드라이버는 보호 영역의 물리메모리 주소를 secure world내의 가상메모리 주소로 매칭하고 이에 대한 페이지를 생성한다. 또한 생성된 가상주소를 EKI 어플리케이션으로 전달하여 보호영역의 런타임 측정값을 생성하도록 도와준다.

4.3 시큐어 타이머를 이용한 상태 모니터링

세 번째 기술적 요소는 TrustZone상에서의 상태 모니터링 방법이다. 비록, secure world의 메모리가 normal world와 독립적으로 나뉘어져 있지만, TrustZone 특성에 따라 secure world상의 프로세스가 동작하기 위해서는 normal world상의 프로세스의 호출이 있어야 한다. 이와 같은 의존성 때문에 trusted application은 독립적으로 동작하는 외부 하드웨어 모니터와 같은 효과를 낼 수 없다. 이를 해결하기 위해 EKI에서는 다음과 같은 시큐어 타이머를 구현하였다.

TrustZone에서는 secure world에 제공되는 하드웨어 타이머가 포함될 수 있다. EKI 드라이버에서 구현한 시큐어 타이머는 타이머 주기를 설정하기 위한 기능과 FIQ 발생 시, 이를 처리하기 위한 타이머 서비스 루틴을 등록하는 기능으로 구성된다. Secure world 부팅 시, secure world내의 EKI 어플리케이션이 시큐어 타이머를 이용하여 모니터링 주기를 설정하고, 타이머 서비스 루틴을 등록한다. 각 주기마다 하드웨어 타이머가 FIQ를 발생시키고, 시큐어 타이머가 등록된 타이머 서비스 루틴에 따라 이를 처리하여 EKI 어플리케이션을 호출한다. 호출을 받은 EKI 어플리케이션에서는 커널 메모리의 보호영역에 대한 검사를 수행하게 된다. 이렇게 구현된 시큐어 타이머를 통해 EKI는 독립된 환경에서 동작될 수 있고, 커널 루트킷의 공격으로부터 안전할 수 있다.

V. 평 가

EKI의 성능을 평가하기 위해서 앞서 제시한 threat model의 공격에 대한 보호여부와 운영체제의 성능저하 측면에서 이를 평가하였다. 모든 실험은 리눅스 3.10.30 커널과 쿼드코어 ARMv7 CPU가 내장되어 있는 실제 단말인, Samsung Tizen TV

상에서 수행되었다.

5.1 보안성 평가

본 실험에서는 threat model에서 가정한 대로, Samsung Tizen TV용 공격 모듈을 직접 구현하고, 이를 미리 설치하여 커널 변조를 진행하였다. 이 모듈을 실행하면 text와 rodata 섹션의 메모리 값을 변조하거나 시스템콜 테이블 엔트리를 후킹하여 원하는 공격함수를 호출한다. 시스템콜 후킹 공격의 경우, TV에서 일반 웹사이트에 로그인 시, 아이디와 패스워드가 어플리케이션에 저장되도록 설계되었다. EKI가 동작하는 상태에서 두 가지 공격에 대한 탐지여부를 측정한 결과, EKI가 코드 인젝션과 시스템콜 후킹 공격을 정상적으로 탐지한다는 것을 확인할 수 있었다.

Table 1.은 공격모듈이 실행되고 검출될 때까지의 평균시간을 측정한 결과이다. 가로축은 모니터링 주기를 나타내고, 세로축은 모니터링 블록크기를 나타낸다. 주기가 50ms이내의 경우에는 모든 공격을 1초 이내에 검출 가능하였고, 주기가 100ms인 경우에도 블록크기가 256Kb인 경우를 제외하고는 1초 이내에 모든 공격을 검출하였다. 이 결과를 통해, 제안된 시스템은 실시간으로 동작하지 못하여 주기적으로 탐지기능을 수행한다는 단점이 있으나 짧은 시간 내에 모든 공격을 탐지할 수 있음을 확인하였다.

5.2 성능 평가

운영체제의 성능저하를 측정하기 위해서 EKI의 모니터링 주기와 각 주기마다 검증하는 블록 크기 별로 benchmark 테스트를 수행하였다. ARM에서 동작되고, 널리 사용되는 Unixbench[14], Linpack[15], CoreMark[16], nbench[17]등 4개의 benchmark를 사용하여 측정방법의 차이를 고려하였고, EKI의 동작유무에 따른 운영체제의 성능저하를 비교하였다. Fig. 4.부터 Fig. 7.은 각 benchmark를 사용하였을 때의 성능을 나타낸 그래프로써 가로축은 모니터링 주기, 세로축은 성능, 각 그래프의 색깔은 블록크기를 나타낸다. 모든 benchmark 테스트 결과에서 모니터링 주기가 길어지고, 블록크기가 작을수록 높은 성능 결과를 나타내었는데 이는 주기가 길수록 EKI의 동작 횟수가 적어지고, 블록크기가 작을수록 연산하는 양이 적기 때문이라고 판단된다.

주기적인 검증방법의 worst case는 가장 짧은 모니터링 주기이면서, 한 주기당 검증하는 블록을 가장 크게 설정하는 경우이다. 이 경우, 다른 설정에

Table 1. Detection Time (in milliseconds)

	10 ms	50 ms	100 ms	500 ms	1000 ms
256Kb	542	939	1491	5205	9987
512Kb	432	634	885	2876	5384
1Mb	359	453	581	1578	2833
2Mb	353	410	558	1085	1836

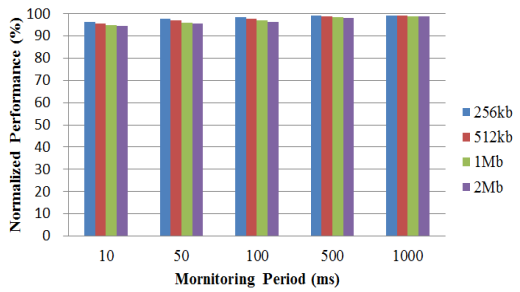

Fig. 4. Performance that measured by Unixbench

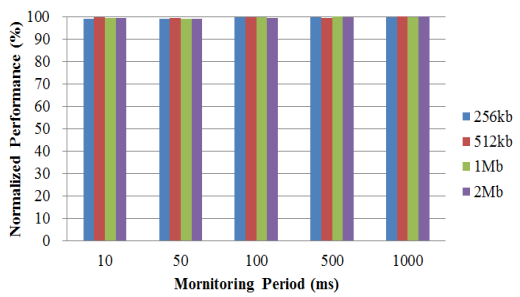

Fig. 5. Performance that measured by Linpack

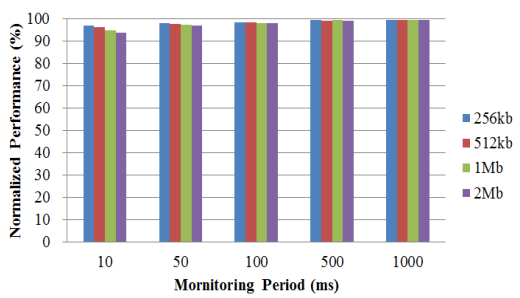

Fig. 6. Performance that measured by CoreMark

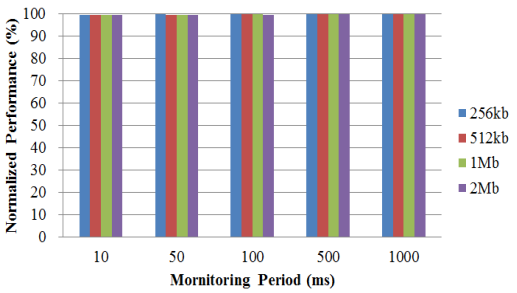

Fig. 7. Performance that measured by nbench

Table 2. Maximum Overhead per Monitor

Monitor	Defense Type	Protection Area	Maximum Overhead
Copilot [3]	Active monitoring	Static memory	9.92%
TZ-RKP [5]	Event driven	Static and Dynamic memory	7.65%
EKI	Active monitoring	Static memory	6.09%

비해 성능저하가 더 많이 발생한다는 단점이 있으나, 실시간으로 검증하는 것과 비슷한 효과를 나타내기 때문에 Vigilare[4]에서 소개된 transient 공격에 대해서도 탐지가 가능하다. 실험결과를 통해, EKI는 최대 6.09%정도의 성능저하만 발생시킨다는 것을 확인할 수 있었다. Table 2.의 연구들의 경우, 실험 환경의 차이가 있으나 Copilot[3]이 최대 9.92%, TZ-RKP[5]가 최대 7.65%의 성능저하를 나타내었고, 이는 EKI가 성능저하 측면에서 우수하다는 것을 보여준다.

VI. 토 의

6.1 Transient 공격

EKI는 실시간으로 모니터링하지 못하고, 일정한 주기를 가지는 시큐어 타이머를 기반으로 동작한다. 또한, 시스템의 효율성을 위해서 각 주기마다 미리 정해진 모니터링 블록영역만 검사하기 때문에, 짧은 시간 동안 공격을 한 후 변조된 데이터를 원본으로 복구하는, Vigilare[4]에서 소개된 transient 공격에 대해서는 약점을 가질 수 있다.

Transient 공격에 대한 탐지율을 높이기 위해서는 시큐어 타이머의 주기를 짧게 설정하고, 모니터링 블록크기를 크게 설정해주면 실시간 모니터링과 같은 효과를 낼 수 있기 때문에 탐지가 가능하다. 하지만 5장의 실험결과와 같이, 시스템의 성능저하가 발생하게 때문에, 각 단말의 성능을 고려하여 transient 공격에 대한 대응수준을 정하고 이에 따른 시큐어 타이머 주기와 모니터링 블록크기를 설정하면 transient 공격에 대한 탐지율을 높일 수 있다.

6.2 DKOM 공격

Poker[13]에서의 커널 루트킷 분석에 의하면 최근에는 Direct Kernel Object Manipulation (DKOM)이라는 복잡한 형태의 공격이 추가되고 있다. DKOM 공격 영역은 Fig. 3.에 표시된 커널 동적영역이다. 이 공격방법은 커널의 자료구조를 직접 생성하거나 제거할 수 있으며 커널 상태정보의 변경하여 통하여 악의적인 동작을 수행한다. 따라서 본 연구에서 제시한 방법으로는 이 공격을 직접적으로 탐지할 수 없으나, 동적영역의 변조를 통해 text와 rodata 섹션의 메모리 값을 변조하거나 시스템콜 테이블 엔트리를 후킹하는 행위를 간접적으로 탐지할 수 있다.

VII. 결론 및 향후 계획

본 연구에서, 우리는 TrustZone의 시큐어 타이머를 이용한 효율적인 커널 검사 시스템을 제안하였다. 이 시스템은 외부 하드웨어 모니터와 TrustZone 기반 커널보호 방법의 장점 활용 및 기존 연구의 단점을 보완하였고, 커널과 동일한 공간으로부터 분리되고, 독립된 공간에서 커널을 보호하는 것이 가능하도록 설계하였다. 이를 위해, 3가지 주요 기술적 요소를 제시하고 이를 구현하였다. 실험을 통해, EKI가 제시된 threat model의 공격탐지가 가능하며, worst case에서도 운영체제의 정상적인 동작을 방해할만한 성능저하를 발생시키지 않는다는 것을 증명하였다. 향후에는 6장에서 설명한 내용들을 고려하여 커널 동적영역을 공격하는 kernel rootkit에 대한 분석 및 악의적인 변조를 검출해낼 수 있는 시스템을 개발할 것이다.

References

[1] O.S. Hofmann, A.M. Dunn, Sangman Kim, I. Roy, and E.Witchel, "Ensuring operating system kernel integrity with OSck," Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems, pp. 279-290, Mar. 2011

[2] Z. Wang, X. Jiang, W. Cui, and P. Ning, "Countering kernel rootkits with lightweight hook protection," Proceedings of the 16th ACM conference on Computer and communications security, pp. 545-554, Nov. 2009

[3] N.L. Petroni Jr., T. Fraser, J. Molina, and W. Arbaugh, W. A. "Copilot - a coprocessor-based kernel runtime integrity monitor," Proceedings of the 13th USENIX Security Symposium, pp. 179-194, Aug. 2004

[4] Hyungon Moon, Hojoon Lee, Jihoon Lee, Kihwan Kim, Yunheung Paek, and Brent Byunghoon Kang, "Vigilare: toward snoop-based kernel integrity monitor," Proceedings of the 2012 ACM conference on Computer and communications security, pp. 28-37, Oct. 2012

[5] A.M. Azab, P. Ning, J. Shah, Q. Chen, R. Bhutkar, G. Ganesh, J. Ma, and W. Shen, "Hypervision across worlds: real-time kernel protection from the ARM TrustZone secure world," Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 90-102, Nov. 2014

[6] X. Ge, H. Vijayakumar, and T. Jaeger, "Sprobes: enforcing kernel code integrity on the TrustZone architecture," Proceedings of the Third Workshop on Mobile Security Technologies (MoST) 2014, May. 2014

[7] "Vmware : Vulnerability statistics," http://www.cvedetails.com/vendor/252/Vmware.html

[8] "Xen : Security vulnerabilities," http://www.cvedetails.com/vulnerability-list/vendor_id-6276/XEN.html

[9] "Sensepost," http://www.sensepost.com/blog/9114.html

[10] "Unlocking the motorola bootloader," http://blog.azimuthsecurity.com/2013/04/unlocking-motorola-bootloader.html

[11] Sunjune Kong and Brent Byunghoon Kang, "Kernel introspection methods based on TrustZone," CISC-W'14, Dec. 2014

[12] "ARM TrustZone Software Architecture," http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/EABGFFIC.html

[13] R. Riley, X. Jiang, D. Xu, "Multi-aspect profiling of kernel rootkit behavior," Proceedings of the 4th ACM European conference on Computer systems, pp. 47-60, Apr. 2009

[14] "Unix bench," https://code.google.com/p/byte-unixbench/

[15] "Linpack," http://www.netlib.org/benchmark/hpl/

[16] "CoreMark," http://www.eembc.org/coremark/download_coremark.php

[17] "nbench," http://www.tux.org/~mayer/linux/bmark.html

〈저 자 소 개〉

김 진 목 (Jinmok Kim) 정회원
2006년 2월: 숭실대학교 정보통신공학부 졸업
2005년 12월~현재: 삼성전자 소프트웨어센터 재직
2015년 3월~현재: 성균관대학교 정보통신대학 석사과정 재학
〈관심분야〉 정보보호

김 동 욱 (Donguk Kim) 정회원
2007년 8월: 고려대학교 산업시스템정보공학과 졸업
2007년 7월~2008년 8월: 한국생산성본부 연구원
2014년 2월: KAIST 산업및시스템공학과 박사
2014년 3월~현재: 삼성전자 소프트웨어센터 재직
〈관심분야〉 정보보호, 시스템 보안

박 진 범 (Jinbum Park) 정회원
2013년 2월: 경원대학교 컴퓨터소프트웨어학과 졸업
2013년 3월~현재: 삼성전자 소프트웨어센터 재직
〈관심분야〉 컴퓨터공학, 소프트웨어 보안, 리눅스 커널 보안

김 지 훈 (Jihoon Kim) 정회원
2014년 2월: 광운대학교 컴퓨터소프트웨어학과 졸업
2014년 3월~현재: 삼성전자 소프트웨어센터 재직
〈관심분야〉 정보보호, 시스템 소프트웨어

김 형 식 (Hyoungshick Kim) 종신회원
1999년 2월: 성균관대학교 정보공학과 졸업
2001년 2월: KAIST 전산학과 석사
2012년 2월: University of Cambridge 컴퓨터공학과 박사
2013년 3월~현재: 성균관대학교 컴퓨터공학과 조교수
〈관심분야〉 정보보호