Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.33, NO.6, Dec. 2023 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2023.33.6.929

ARM PMU 이벤트를 활용한 TrustZone 루트킷 탐지에 대한 연구

최 지 민,[1†] 신 영 주[2‡]
[1,2]고려대학교 정보보호대학원 (대학원생, 교수)

Detection of TrustZone Rootkits Using ARM PMU Events

Jimin Choi,[1†] Youngjoo Shin[2‡]
[1,2]School of Cybersecurity, Korea University (Graduate student, Professor)

요 약

모바일 장치에서 사용되는 ARM 프로세서는 하드웨어 기반의 격리 실행 환경인 TrustZone 개념을 도입하여 신뢰 실행 환경인 Secure World와 비신뢰 실행 환경인 Normal World를 구현하였다. 악성 소프트웨어의 종류 중 루트킷은 관리자 권한을 획득하고 자신의 존재를 숨기면서 백도어를 만든다. Secure World에서 동작하는 프로세스는 메모리 접근에 제한이 없고, 격리되어 있어 Secure World에서 루트킷이 실행되었을 때 탐지하기 어렵다. 본 논문에서는 하드웨어 기반의 성능 측정 모니터인 Performance Monitoring Unit을 활용하여 Secure World 루트킷의 이벤트를 측정하고 딥러닝 기반으로 루트킷을 탐지하는 기법을 제시한다.

ABSTRACT

ARM processors, utilized in mobile devices, have integrated the hardware isolation framework, TrustZone technology, to implement two execution environments: the trusted domain “Secure World” and the untrusted domain “Normal World”. Rootkit is a type of malicious software that gains administrative access and hide its presence to create backdoors. Detecting the presence of a rootkit in a Secure World is difficult since processes running within the Secure World have no memory access restrictions and are isolated. This paper proposes a technique that leverages the hardware based PMU(Performance Monitoring Unit) to measure events of the Secure World rootkit and to detect the rootkit using deep learning.

Keywords: ARM TrustZone, Rootkit, PMU, Deep learning, Detection

I. 서 론

스마트폰 시장의 급속한 성장으로 스마트폰은 업무와 사생활에서 필수적인 기기로 자리 잡았다. 다양한 모바일 앱과 함께 카메라, 마이크, 그리고 여러 물리 센서들을 통해 스마트폰은 디지털 정보뿐만 아니라 실제 세계의 물리적 정보까지 처리하게 되었다[1][2]. 그리하여 스마트폰을 악성 소프트웨어의 주요 대상으로 만들었으며, 스마트폰의 보안 중요성은 더욱 강조되고 있다.

루트킷(Rootkit)은 악성 소프트웨어로, 시스템의 보안 메커니즘을 우회하여 관리자 권한을 획득하거나 자신의 행동을 숨기는데 사용된다. 이는 공격자에게 시스템에 대한 지속적인 엑세스 권한을 제공하며, 사용자나 시스템 관리자가 감지하기 어렵게 만든다.

악성 소프트웨어로부터 시스템을 보호하기 위해 하드웨어 기반의 격리 실행 환경이 도입되었고, ARM 프로세서는 트러스트존(TrustZone) 아키텍처를 구성하였다. 트러스트존은 사용자가 접근하는 노멀월드(Normal World)와 별도로 시큐어월드

Received(10. 16. 2023), Modified(10. 30. 2023), Accepted(11. 08. 2023)
† 주저자, cjm4leaf@korea.ac.kr
‡ 교신저자, syoungjoo@korea.ac.kr(Corresponding author)

(Secure World)라는 격리된 실행 환경을 제공하여 악성 소프트웨어의 공격으로부터 보호한다.

트러스트존 아키텍처를 따르면, 시큐어월드는 노멀월드 메모리에 접근할 수 있지만, 반대로 노멀월드는 시큐어월드의 메모리에 접근할 수 없다[3]. 시큐어월드에서 실행되는 TAs(Trusted Applications)는 그 특성상 트러스트존을 보호하기 위한 다양한 보안 메커니즘[4]이 존재하고 인증된 코드만 실행하도록 서명검증을 한다. 하지만 드라이버나 TAs의 개발과정에서 생기는 보안 결함, 공급업체의 보안 관련 미흡, 트러스트존의 역공학적 분석[5][6] 등으로 인한 취약점이 발생할 위험이 있다. 이러한 취약점들을 활용하면 강력한 루트킷을 개발할 수 있으며, 이를 기반한 루트킷 프레임워크도 공개되었다[7].

기존의 루트킷 탐지 방법들은 시그니처, 행동, 무결성을 기반으로 하거나 추가적인 하드웨어를 이용해 탐지한다[8]. 그러나 시큐어월드의 OS권한을 가진 루트킷[7]은 모든 메모리 영역에 접근할 수 있고, 기존의 탐지 프로세스들을 우회할 수 있다. 따라서 본 논문은 하드웨어 기반의 성능 측정 모니터인 PMU(Performance Monitoring Unit)을 활용하여 루트킷 특유의 발생 이벤트를 탐지한다. PMU는 실시간으로 프로세서의 작동 사이클, 캐시, 메모리 접근 등을 실시간으로 측정할 수 있으며, 오버헤드가 크지 않은 장점이 있다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문의 배경에 대해 설명하고, 3장에서는 시큐어월드에서 동작하는 루트킷을 탐지 방법을 제시한다. 4장에서는 제안된 방법의 구현과 검증 과정을 설명하며, 마지막으로 5장에서 탐지의 한계와 앞으로 향후 연구에 대해서 언급하고 결론을 맺는다.

II. 배 경

루트킷은 관리자 권한을 획득하여 자신의 존재를 숨기고 백도어를 생성하여 지속적으로 루트권한을 유지한다. 기존에 알려진 루트킷의 종류는 응용 프로그램 레벨, 라이브러리 레벨, 펌웨어 레벨 및 가상화 레벨 이 있다[8]. DKOM(Direct Kernel Object Manipulation)[9] 기법이 나오면서 커널 레벨 루트킷들은 커널 오브젝트를 직접 조작해 후킹없이 루트킷을 숨기면서 대상 시스템을 자유롭게 제어할 수 있게 되었고, 많은 루트킷들이 이 방법을 사용하기 시작했다. 이러한 루트킷을 탐지하기 위해 시그니처, 행동 , 학습, 외부 하드웨어, 무결성, 교차검증 기반 등의 탐지기법들이 나왔다[10][11][12].

트러스트존의 시큐어월드를 활용한 루트킷 탐지 기법들도 나왔는데 TAs을 사용하여 메모리 무결성을 검증하거나 리눅스의 task_struct에서 숨겨진 태스크를 찾는다[13]. SPROBES[14]는 리눅스 커널의 취약점에서 ROP 공격을 대응하기 위해 시큐어월드에서 승인된 코드만 실행되도록 제한하여 루트킷의 위협을 방지한다. 다양한 시큐어월드 기반의 루트킷 탐지 방법들이 있지만, 이들은 주로 시큐어월드의 높은 권한을 활용하는 방식이다.

2.1 ARM 트러스트존 기반의 루트킷

ARMv8-A아키텍처[3]는 ELs(Exception Levels)을 지원하여, 최대 4개의 권한 레벨로 구성되어 있다. 어플리케이션은 EL0에서 실행되고, Rich OS는 EL1에서 동작하며, 하이퍼바이저는 EL2에서 작동한다. EL3는 노멀월드와 시큐어월드를 전환시켜주는 시큐어모니터(Secure Monitor) 권한으로 설정되어 있다. Fig. 1.에서 이 구조와 각 권한 레벨을 확인할 수 있다.

이전의 루트킷 탐지 방법들은 시큐어월드의 특성을 이용해, 노멀월드에서 루트킷이 실행되어도 시큐

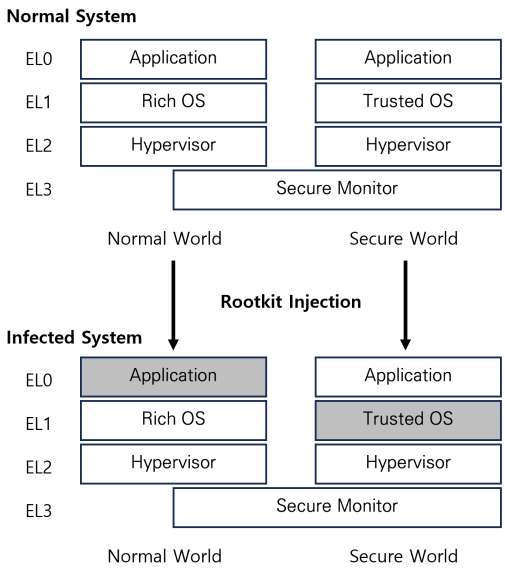

Fig. 1. ARMv8 ELs and their components in the infection process

어월드의 무결성을 침해할 수 없는 것에 기반을 두었다. 그러나 시큐어월드 기반의 루트킷은 탐지 솔루션과 같은 권한을 가지므로 이를 보장하지 못한다. 공개된 시큐어월드 기반 루트킷[7]은 기본적으로 시큐어월드의 EL1(Trusted OS) 권한과 노멀월드의 EL0(Client Application) 권한을 획득한 상태로 수행된다.

리눅스 커널은 task_struct 구조체를 활용하여 프로세스를 관리한다. task_struct는 더블 링크드 리스트 형식으로 구성되어 있어, 시스템 내의 모든 테스크를 추적하고 관리할 수 있다. 이 구조체 내부에는 프로세스 이름을 나타내는 comm, 프로세스의 상태를 표현하는 state, 프로세스의 ID를 나타내는 pid 필드가 존재한다. 또한 list_head의 next, prev 필드를 통해 앞뒤 프로세스의 포인터정보를 가지고 있다.

시큐어월드 기반의 루트킷은 노멀월드의 메모리를 차례대로 탐색하며 커널에서 최초 실행되는 “swapper“ 프로세스를 task_struct의 comm 필드에서 찾는다. SMP(Symmetrical Multiprocessing) 환경에서는 첫 번째 태스크 이름에 “/0”가 접미사가 붙어 “swapper/0”을 찾으면 이후에 task_struct의 next와 prev 필드를 활용해 모든 태스크 리스트를 확보할 수 있다. 이를 통해 루트킷은 원하는 프로세스의 권한을 변경하여 관리자 권한을 획득한다.

2.2 PMU(Performance Monitoring Unit)

PMU[3]는 프로세서 성능을 모니터링하고 분석하기 위한 하드웨어 구성 요소이다. 다양한 이벤트를 측정하여 소프트웨어나 시스템 성능 분석에 도움을 준다. CPU의 사이클, 버스, 캐시 등의 이벤트를 카운팅하여 분석하고, 독립적인 하드웨어 구성으로 되어 있어 보안 분야에서 활용도가 높다. 예를들어, PMU을 활용해 ROP공격을 탐지하고[15], Rowhammer 공격을 탐지하거나[16]. Specter 공격을 탐지[17][18], 그리고 CFIMon은 제어 흐름 무결성을 검증하기 위해 PMU을 활용했다[19].

그러나 PMU을 활용한 이런 방법들은 주로 노멀월드에서의 공격을 탐지하는 방법들로, 시큐어월드 내의 루트킷은 탐지하지 못한다. 본 논문은 시큐어월드에서 동작하는 루트킷의 특징을 PMU 이벤트를 통해 파악하고, 그 특성을 기반으로 루트킷을 탐지하는 새로운 방법을 제시한다.

III. 제안기법

시큐어월드 기반의 높은 권한에서 실행되는 루트킷이 있어도, PMU는 하드웨어 수준에서 독립적으로 분리되어 있어 장치에 대한 무결성이 유지된다. Fig. 2.는 루트킷 소스코드의 시작 부분으로 메모리 액세스를 많이 시도하는 것을 확인할 수 있다. 이러한 특성을 이용하면, 시큐어월드에서 루트킷이 루트 권한을 획득하기 전에 PMU 이벤트를 통해 루트킷의 활동을 식별하고 탐지할 수 있다.

```
paddr_t find_kernel_entrypoint(void)
{
  for (paddr_t page = NS_BASE_OFFSET; page < SDRAM_SIZE; page += PAGE_SIZE)
  {
    struct mobj *mobj = load_page(page);
    if (!mobj)
      continue;
    ...
  return uefi_header_addr;
}

paddr_t find_init_task_name(paddr_t uefi_header_addr)
{
  for (paddr_t page = uefi_header_addr; page < SDRAM_SIZE; page += PAGE_SIZE)
  {
    struct mobj *mobj = load_page(page);
    if (!mobj)
      continue;
    uint64_t *vaddr = mobj_get_va(mobj, 0);
    void *process_name = find(vaddr, PAGE_SIZE,
                  INIT_TASK_COMM, strlen(INIT_TASK_COMM) + 1);
    free_page(mobj);
    if (process_name)
    {
      paddr_t process_name_offset = (uint8_t *)process_name - (uint8_t *)vaddr;
      paddr_t process_name_pa = page + process_name_offset;
      return process_name_pa;
    }
  }
  return ERROR_ADDR;
}
```

Fig. 2. Rootkit source code to find the kernel's start address and task_struct

3.1 공격모델

시큐어월드에서는 서명 검증이 완료된 어플리케이션만 로드하는 보호 매커니즘이 존재한다[3]. 그러나 TAs을 제작하는 벤더나 개발자가 공격을 받거나 악의적으로 행동을 할 위험이 있으며, 실제 기기를 대상으로 시큐어월드에서 임의코드가 실행되는 취약점들도 발견되었다[20][21].

본 공격시나리오는 TAs를 제작하는 벤더나 개발자가 악성코드에 감염되거나 악의적인 의도, 시큐어월드의 EL1에 임의의 코드 실행이 가능한 취약점이

있다는 가정 하에 진행된다. 시큐어월드의 EL1과 노멀월드의 EL0의 권한을 획득하는건 연구 범위에서 다루지 않는다. 또한 시큐어월드 EL1의 권한을 갖더라도, 독립적 실행은 불가능하며, 공격 모델은 노멀월드의 EL0에서 실행되면 시큐어월드의 루트킷을 실행되는 것으로 공격모델을 제한한다.

시큐어월드의 EL1 권한은 시큐어월드의 OS 레벨의 권한으로 메모리 접근에 제약이 없으며, PMU 이벤트를 비활성화 하거나 조작할 수 있는 권한을 갖는다. 이러한 권한으로 본 연구에서 제시하는 모델은 공격자로부터 공격을 보호하는 것에 중점이 아닌 노멀월드의 관리자 권한을 획득하기 전에 이상 행위를 탐지하는 것을 목적으로 둔다.

이 연구의 공격 시나리오에서는 노멀월드의 커널의 구조를 알 수 없다고 가정한다. 또한 커널의 소스코드 또는 컴파일 아티팩트에 대한 접근이 불가능하다는 것을 전제로 한다. 함수 및 데이터 구조와 같은 심볼의 위치정보는 확인할 수 없다. 공격시나리오의 목적은 시큐어월드를 활용해 노멀월드 메모리에 접근하여 노멀월드의 관리자 권한을 획득하는 것을 목적으로 한다.

3.2 탐지기법

시큐어월드에서 실행되는 루트킷[7]은 시큐어월드의 EL1 권한으로 노멀월드의 모든 메모리에 접근할 수 있으나, 리눅스는 KASLR[29] 및 randstruct 기법으로 메모리를 보호하고 있다. 이러한 보호기법을 우회하기 위해서, 루트킷은 노멀월드 메모리의 시작 주소부터 페이지 단위로 순차적으로 탐색하며 커널 이미지를 찾고 task_struct를 찾는다.(Fig. 2.) 이 과정에서 루트킷은 많은 메모리 이벤트가 발생하고 PMU로 이벤트를 측정하면 루트킷 고유의 이벤트를 탐지할 수 있다.

루트킷을 차단하기 위해서는 루트 권한을 획득하기 전에 미리 탐지해야 한다. 루트킷은 초기 단계에서 많은 이벤트를 발생시키므로, PMU 측정을 프로세스 실행 직후 짧은 간격으로 측정하여 루트킷이 루트 권한을 얻기 전에 루트킷을 판별한다.

PMU를 활용하여 동적으로 탐지하는 방법으로는 딥러닝(Deep learning)을 이용하여 악성코드를 탐지하는 방법[25], 사이드채널 공격을 탐지하는 방법[26][27]등 다양한 연구 방법들이 제시되고 비교되었다.

본 논문에서는 딥러닝을 활용하여 루트킷 탐지를 진행하였다. 딥러닝 기반의 탐지모델은 데이터를 기반으로 학습하여 높은 정확도를 보이며, 새로운 이벤트를 지속적으로 학습에 포함해 확장성을 갖출 수 있는 장점이 있다.

학습 데이터를 확보하기 위해 공격모델의 루트킷과 OP-TEE[22]에서 제공하는 TA example[23], xtest, MbedTLS[24]의 PMU 이벤트를 측정한다. MbedTLS는 임베디드 환경을 위한 경량화된 오픈 소스 SSL/TLS 라이브러리로, MbedTLS의 벤치마크와 암호화, 복호화 기능들을 측정하였다.

PMU 이벤트 측정 데이터는 시계열 데이터로 딥러닝 모델 중 RNN, LSTM, GRU, LSTM-GRU 모델로 학습하여 성능을 평가하였다. 학습 데이터들을 기반으로 가장 높은 정확도를 보인 모델을 최종적으로 선택한다.

머신러닝을 학습하기 위해 Tensorflow[28] 프레임워크를 사용하고, Tensorflow Lite를 활용해 모델을 변환하여 실제 장치에 배포한다. Tensorflow Lite는 머신러닝 모델을 경량화하고 최적화하여 모바일 및 IoT장비에서 운영될 수 있게 만드는 크로스 플랫폼 딥러닝 프레임워크이다. 이 프레임워크는 제한된 메모리와 연산 능력을 갖춘 임베디드 시스템의 자원 제약 문제에 효과적으로 대응할 수 있도록 설계되었다[32]. Tensorflow Lite를 적용한 예로 SmartHat[33]은 라즈베리파이를 활용해 실시간 이미지 개체 분류 작업을 수행하고, DL-HIDS[34]는 딥러닝 모델을 활용해 IDS를 구성하여 IoT장치에서 사용한다.

IV. 구현 및 검증

4.1 제안기법 구현

본 논문에서는 트러스트존 환경에서 하드웨어 기반 PMU 이벤트를 활용하여 시큐어월드의 루트킷을 탐지하는 기법을 제안한다. 제안기법은 Hikey960 개발 보드에서 구현한다. Hikey960은 ARMv8-A 기반의 Cortex-A73과 Cortex-A53, 2개의 CPU로 구성되어 있다. ARMv8-A 계열의 CPU는 트러스트존 환경과 PMU를 지원한다.

4.1.1 OP-TEE

OP-TEE[22]는 Linaro 그룹이 개발하고 유지 관리하는 신뢰실행환경(Trusted Execution Enviroment)[3]이다. ARM 트러스트존 기술을 기반으로, 현재 오픈소스로 운영되고 있다. ARMv8 기반의 여러 개발보드, 예를 들어 Hikey, STM, Raspberry Pi 등 다양한 개발보드와 QEMU를 지원하고 있다. 오픈소스화로 인해 많은 IoT 장비들이 신뢰실행환경 구성을 위해 OP-TEE를 사용한다.

Hikey960은 Linaro와 개발자 커뮤니티(96Boards)와의 협업으로 제작되었고, 이로 인해 OP-TEE 환경을 테스트하기에 적합하다. 공격모델과 환경을 일치시키기 위해 OP-TEE는 3.11 버전으로, 커널은 리눅스 5.4 버전으로 설정하였다. 또한 KASLR[29]을 활성화하여 노멀월드에서 커널 이미지가 랜덤 위치에 로드되도록 설정하였다. 나머지 설정은 기본으로 설정하여 구성하였다.

4.1.2 루트킷 구성

공개된 시큐어월드 기반의 루트킷[7]은 QEMU 환경에 맞춰 설정하고 테스트를 했기 때문에, 실제 장치에서 테스트하기 위해서는 몇 가지 설정이 필요하다. 루트킷이 노멀월드 메모리의 시작 베이스 주소부터 순차적으로 탐색하기 때문에, 베이스 주소 및 PHYS_OFFSET 등을 Hikey960 메모리 구조에 맞게 수정한다. Hikey960 장치는 4GRAM로 구성되어 있으며 메모리 구조는 Fig. 3.과 같다. KASLR이 설정되어 있어 커널이 로드되는 주소는 DDR_BASE부터 BL1전까지(0x0~0x1AC00000), 그리고 BL2와 Non Secure DRAM 사이에서 (0x1AC58000~0x3E000000) 랜덤하게 결정된다.

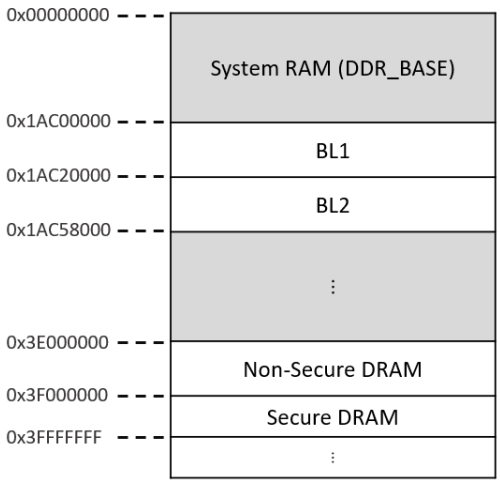

Fig. 3. Memory Structure of Hikey960

구현된 루트킷[7]은 노멀월드에서 관리자 권한 획득, 프로세스 보호를 위한 기아 프로세스로 변환, 시큐어월드 권한을 활용한 메모리 카빙 기능이 존재한다. 세 가지 기능 모두 시큐어월드의 메모리 접근 권한을 기반으로 구현되었다.

4.1.3 루트킷 탐지 구현

구성된 환경에서 PMU는 총 33가지의 이벤트를 지원하나, 동시에 활성화할 수 있는 이벤트는 6가지로 제한된다. 루트킷을 100회씩 실행하면서 PMU 이벤트 중 표준편차가 높은 6개 이벤트를 선별하였다. 6개의 이벤트는 Fig. 4.와 같다. 그래프를 관찰하면, 최초 실행과 10초 경과 후에 이벤트 카운터가 높은 것을 확인할 수 있다.

공격모델의 루트킷[7]은 평균 299ms에 루트권한을 획득한다. 따라서 본 논문에서 제안된 기법은 299ms이내 이상여부를 탐지하여야 한다. 이를 위해 PMU 이벤트 측정 간격을 30ms로 설정하고, 시퀀스를 2개만 획득하도록 하였다. 측정 간격이 30ms로 짧지만, 2개의 시퀀스만 획득하기 때문에 오버헤드에는 영향이 없을 것으로 예상된다.

Table 1.에 제시된 프로세스들을 각각 100회 실행하여 PMU 이벤트 데이터를 수집한다. 수집된 데

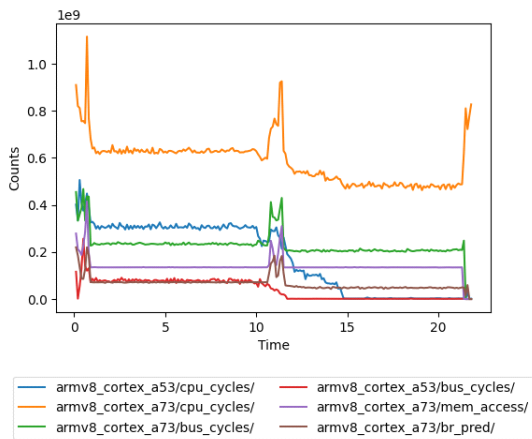

Fig. 4. Graph of the top 6 standard deviations of PMU events during rootkit execution

Table 1. Deep learning training data targets and PMU event types to capture

Process	PMU Event Type
Rootkit	Cortext A53/cpu_cycle Cortext A53/bus_cycle Cortext A73/cpu_cycle Cortext A73/bus_cycle Cortext A73/mem_access Cortext A73/br_pred
OP-TEE Example	
OP-TEE xtest	
MbedTLS	

이터는 8:2의 비율로 학습과 테스트 데이터로 나누었다. 3.2절에서 언급한 딥러닝 모델별로 학습해 모델 성능을 평가를 진행한다.

모든 모델에서 출력은 이진 분류를 위한 sigmoid 활성화 함수를 사용했다. 각 모델은 2개의 층으로 설계되었고, 마스킹 레이어와 드랍아웃기법을 사용해 데이터의 불규칙성을 처리하고 오버피팅을 방지하였다.

Fig. 4.에서 초기에 관찰된 높은 이벤트 발생을 고려하여 시퀀스 크기를 2로 설정하였다. 모델에 더 많은 층을 쌓으면 성능 향상이 있을 수 있지만, IoT 장비에서 작동 시 오버헤드를 최소화하기 위해 모델 구조를 간결하게 유지하였다.

Fig. 5.는 각 모델의 데이터 분류 정확도와 손실율을 나타내는 그래프이다. RNN 모델은 정확도가 99%로 나머지 세 모델보다 정확도가 높고 loss 또한 낮게 나타났다. Table 2.는 테스트 데이터로 모델들의 성능평가 결과로 모든 모델이 우수한 성능을 보였다. 이는 개체 분류를 이진 분류로 단순화하였기 때문에 이런 결과가 나온 것으로 판단된다.

Fig. 5. 결과를 바탕으로 루트킷 탐지에는 RNN 모델을 사용한다. 학습된 RNN 모델은 Tensorflow Lite를 통해 바이너리 형태로 변환한 후, 개발 보드

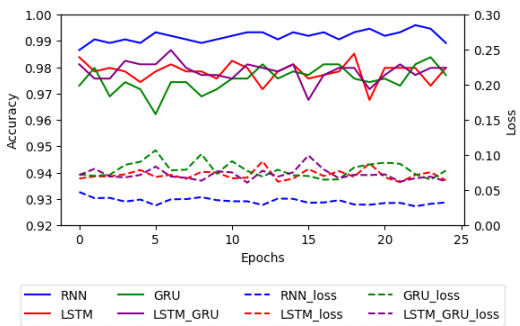

Fig. 5. Compare rootkit classification accuracy and loss rates by model

Table 2. Model-specific evaluation results table with test data

Model	Accuracy	Precision	Recall	F1 Score
RNN	0.99	0.94	1.00	0.96
LSTM	0.98	0.95	0.97	0.96
GRU	0.98	0.97	0.95	0.96
LSTM-GRU	0.98	0.96	1.00	0.97

에 배포하여 테스트한다.

개발 보드에서 루트킷을 실행할 때, 학습된 모델을 활용하여 PMU 이벤트를 캡처하고 탐지율을 측정하였다. 100회의 루트킷 실행 시, 탐지율은 94%가 나타났다. 또한 탐지모델은 데이터 시퀀스 길이가 2개로 짧아 루트킷을 탐지하는데 평균 251ms가 소요되었다. 이는 루트킷이 루트 권한을 획득하는 299ms보다 빠르게 탐지하였고, 탐지모델이 노멀월드의 루트킷 프로세스를 중단시키면, 루트권한을 획득하지 못하고 프로세스가 종료되고 시큐어월드의 루트킷도 종료되었다.

4.2 제안기법 성능 평가

본 절에서는 제안된 기법의 성능 오버헤드를 평가한다. 분류 작업에 활용된 루트킷, OP-TEE Example, MbedTLS 프로세스들의 실행시간을 탐지 모델의 활성화 유무에 따라 측정하였다. 이를 통해, 탐지 모델의 활성화 시 발생하는 오버헤드를 정량적으로 평가하였다.

PMU 이벤트의 측정은 측정 간격에 따라 성능 변

Table 3. Execution time performance evaluation results by disabling/enabling the detection model (unit: seconds)

type	Process	Model		Over head	Over head(%)
		Disable	Enable		
rootkit	rootkit	21.555	21.407	-0.148	-0.69%
OPTEE example	acipher	5.871	6.087	0.216	3.68%
	xtest	64.771	64.235	-0.536	-0.83%
MbedTLS	selftest	1.158	1.144	-0.014	-1.21%
	crpyt_and_hash	3.276	3.281	0.005	0.15%
	benchmark	181.072	181.866	0.794	0.44%

동이 있다. 본 연구에서는 30ms 간격으로 이벤트를 측정하였다. 측정 간격은 탐지 기법 및 성능 평가에 영향을 미칠 것으로 예상된다.

Table 3.은 프로세스를 반복적으로 실행할 때, 탐지모델을 비활성화한 상태와 활성화한 상태에서의 실행시간의 평균값이다. 각 프로세스마다 발생하는 오버헤드가 다르지만, 전반적으로 3% 내외의 오버헤드가 확인되었다. 이러한 결과를 통해, 제안된 루트킷 탐지모델은 시스템 성능에 많은 영향을 주지 않으면서 높은 정확도로 루트킷을 탐지할 수 있음을 알 수 있다.

V. 결 론

다수의 연구에서는 노멀월드의 루트킷 탐지 방법들에 대해 소개되었다. 그러나 본 논문은 시큐어월드에서 작동하는 루트킷의 탐지에 중점을 두었고, 이 방법을 통해 높은 정확도와 낮은 오버헤드로 탐지가 가능함을 확인하였다.

5.1 탐지모델 한계

제시한 탐지모델은 한계점을 가지고 있다. 첫째, PMU를 기반한 탐지 방식이기 때문에, 루트킷 실행 속도를 느리게 조절하거나, 이벤트 발생을 늦추는 등의 루트킷 행동을 변조한다면 탐지하기 어렵다. 둘째, 루트킷은 시큐어월드의 EL1 권한으로 실행되기 때문에, PMU로 측정된 메모리를 변조하거나, PMU 측정 자체를 비활성화 시키는 등의 우회 기법이 존재한다. 마지막으로 탐지모델의 탐지속도보다 루트킷의 공격 성공 이후에 탐지하거나 탐지에 실패할 경우가 존재한다.

변조되는 루트킷을 대응하기 위해 룰 기반이 아닌 머신러닝을 통해 루트킷을 탐지하였다. 변조된 루트킷의 데이터를 확보한다면 재학습하여 탐지모델에 반영하여 위협을 줄일 수 있고, 다른 루트킷들에 대한 탐지도 가능하다.

시큐어월드 루트킷의 EL1 권한으로 인한 문제는 ReZone[31]과 Trusted Monitor[30]을 활용하여 해결할 수 있다. REZONE은 시큐어월드의 과도한 권한을 보호하기 위해 EL0와 EL1을 도메인별로 격리시켜 존으로 분리하였다. 존 내에서 실행되는 코드의 메모리 접근 권한을 제한하여 다른 존에 접근할 수 없도록 구현하였다. Trusted Monitor는 Trusted Firmware-A(EL3)와 시큐어월드의 EL1사이에 Watchdog과 PMU를 드라이버로 구현하여 시큐어월드 루트킷의 공격에서 방어하도록 구현되어 있다. 이 두 기법을 참고하여 시큐어월드에 두 개의 존을 구성하고, 탐지모델을 다른존에서 드라이버 형태로 구현한다면, REZONE의 격리된 환경으로 인해 루트킷의 시큐어월드 EL1 권한의 공격으로부터 보호받을 수 있다. 존으로 시큐어월드의 EL0과 EL1을 구성한다면 메모리에 대한 접근 권한도 분리되어 있기에 무결성을 보장받을 수 있다.

탐지모델이 루트킷이 공격이 성공한 후에 탐지하거나, 탐지에 실패하였을 경우에도 앞서 언급한 ReZone[31]으로 인해 노멀월드가 보호될 수 있다. ReZone은 메모리 권한을 분리하여 시큐어월드의 EL1 권한이라도 노멀월드 EL1의 메모리를 보호할 수 있도록 구현되어 있다. 따라서 탐지모델이 실패하더라도 노멀월드의 EL1은 보호할 수 있다.

딥러닝 모델에서 정상과 비정상으로 분류하는 이진 분류는 정확도가 높게 나오지만, 각각의 타입을 분류하는 다중 분류의 경우 정확도가 높게 나오지 않는다. 본 연구에서도 다중 분류를 적용하려 했으나, 정확도가 낮아 적용하지 못하였다.

5.2 향후연구

향후 연구로 본 논문에서 제안한 탐지모델을 발전시켜, 노멀월드의 루트킷과 악성코드, 그리고 시큐어월드 기반의 다른 루트킷들을 학습하여 범용적인 루트킷 탐지모델을 만들고자 한다. 추가적으로, ReZone[31]과 Trusted Monitor[30]방식을 참고해 시큐어월드의 EL0와 EL1을 두 개의 존으로 분리하여 탐지모델을 위한 존을 구성하고 Trusted Firmware-A와 시큐어월드 사이에 탐지모델을 배치해, 노멀월드와 시큐어월드에서의 공격을 방어하도록 구현할 계획이다. 또한, 탐지모델은 다중 분류가 가능하도록 모델 구조를 최적화하여 실행되는 프로세스의 구별능력을 향상시킬 계획이다.

5.3 결론

모바일 및 IoT기기의 사용이 확산되면서, 기기 내의 데이터의 중요도는 증가하고 있다. 본 논문은 하드웨어 기반의 성능 측정 도구인 PMU를 활용하여 시큐어월드의 루트킷 탐지 방법을 제안하였다. 기존

의 PMU 기반 탐지 모델은 노멀월드의 루트킷에 초점을 맞추었지만, 본 연구에서는 시큐어월드에서의 루트킷의 탐지 방법을 제시하였다.

제시한 탐지모델은 딥러닝을 기반으로 확장 가능하면서도 루트 권한을 획득하기 전에 루트킷을 조기에 탐지할 수 있고 그 결과는 매우 높은 정확도를 보여준다. 또한 이 모델은 정상적인 동작을 방해할만한 성능저하를 발생시키지 않는다는 것을 증명하였다. 앞으로는 이 방법의 범용성을 높이고, 시큐어월드에서 탐지모델에 대한 공격도 방어할 수 있도록 연구를 확장할 계획이다.

References

[1] A. Khatoon, P. Corcoran, "Privacy concerns on android devices," 2017 IEEE International Conference on Consumer Electronics (ICCE), pp. 149-152, Jan. 2017.

[2] P. Ketelaar, M. van Balen, "The smartphone as your follower: the role of smartphone literacy in the relation between privacy concerns, attitude and behaviours towards phone-embedded tracking", Computers in Human Behavior, vol. 78, pp. 174-182, Jan. 2018.

[3] ARM, et al, "ARM Cortext-A Series : Programmer's Guide for ARMv8-A", 2015.

[4] "GlobalPlatform Technology : TEE Protection Profile", Public release v1.3, Jul. 2020.

[5] D. Cerdeira, N. Santos, P. Fonseca, and S. Pinto, "Sok: understanding the prevailing security vulnerabilities in trustzone-assisted tee systems," 2020 IEEE Symposium on Security and Privacy (SP), pp.1416-1432, May. 2020.

[6] F. Fleischer, M. Busch, and P. Kuhrt, "Memory corruption attacks within android tees: a case study based on op-tee.", Proceedings of the 15th International Conference on Availability, Reliability and Security., pp.1-9, Aug. 2020.

[7] Marth, Daniel, et al, "Abusing trust: mobile kernel subversion via trustzone rootkits.", 2022 IEEE Security and Privacy Workshops (SPW), pp. 26-276, May. 2022

[8] Nadim, Mohammad, Wonjun Lee, David Akopian. "Kernel-level rootkit detection, prevention and behavior profiling: a taxonomy and survey.", arXiv preprint arXiv:2304.00473, Mar. 2023

[9] Greg Hoglund, Jamie Butler Author, "Rootkit : Subverting the Windows Kernel ", Addison Wesley, pp. 199-242, 2005

[10] Wang, Jianxiong, "A rule-based approach for rootkit detection.", 2nd IEEE International Conference on Information Management and Engineering. IEEE, pp. 11-21, Apr. 2010

[11] Song, Chengyu, et al, "Enforcing kernel security invariants with data flow integrity.", NDSS, Feb. 2016.

[12] Joy, Jestin, Anita John, James Joy, "Rootkit detection mechanism: A survey.", International Conference on Parallel Distributed Computing Technologies and Applications, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 366-374, Sep. 2011.

[13] Dong, Shangwu, et al, "Kims: kernel integrity measuring system based on trustzone.", 2020 6th International Conference on Big Data Computing and Communications (BIGCOM). IEEE, pp . 103-107 , Jul. 2020.

[14] Ge, Xinyang, Hayawardh Vijayakumar, Trent Jaeger, "Sprobes: Enforcing kernel code integrity on the trustzone architecture." arXiv preprint arXiv:1410.7747, Oct. 2014

[15] Pappas, Vasilis, Michalis Polychronakis,

A.D. Keromytis, "Transparent {ROP} exploit mitigation using indirect branch tracing.", 22nd USENIX Security Symposium (USENIX Security 13), pp. 447-462, Aug. 2013.

[16] Kuruvila, Abraham Peedikayil, et al, " Explainable machine learning for intrusion detection via hardware performance counters.", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 41.11, pp. 4952-4964, Feb. 2022

[17] Li, Congmiao, Jean-Luc Gaudiot, "Detecting spectre attacks using hardware performance counters." IEEE Transactions on Computers 71.6, pp. 1320-1331, Jun. 2022.

[18] Zhang, Yunjie, Yiorgos Makris, "Hardware-based detection of spectre attacks: a machine learning approach.", 2020 Asian hardware oriented security and trust symposium (AsianHOST) IEEE, pp. 1-6, Dec 2020.

[19] Xia, Yubin, et al, "CFIMon: Detecting violation of control flow integrity using performance counters.", IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012). IEEE, pp. 1-12, Jun 2012.

[20] Shakevsky, Alon, Eyal Ronen, Avishai Wool, "Trust Dies in Darkness: Shedding Light on Samsung's {TrustZone} Keymaster Design.", 31st USENIX Security Symposium (USENIX Security 22), pp. 251-268, Aug. 2022.

[21] Ryan, Keegan, "Hardware-backed heist : Extracting ECDSA keys from qualcomm's trustzone.", Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 181-194, Nov. 2019.

[22] About OP-TEE, "Open portable trusted execution environment op-tee", https://optee.readthedocs.io/en/latest/general/about.html, Oct. 2023

[23] optee_example, "OP-TEE documentation - optee_examples", https://optee.readthedocs.io/en/latest/building/gits/optee_examples/optee_examples.html, Oct. 2023

[24] Mbed TLS, "Mbed TLS", https://www.trustedfirmware.org/projects/mbed-tls/, Oct. 2023

[25] Chaganti, Rajasekhar, Vinayakumar Ravi, Tuan D. Pham, "Deep learning based cross architecture internet of things malware detection and classification." Computers & Security 120, Sep. 2022.

[26] Li, Xinyao, Akhilesh Tyagi, "Cross-World Covert Channel on ARM Trustzone through PMU.", Sensors 22.19, Sep. 2022

[27] Mushtaq, Maria, et al, "Nights-watch: a cache-based side-channel intrusion detector using hardware performance counters.", Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy, pp. 1-8, Jun. 2018.

[28] Abadi, M. et al, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems". https://www.tensorflow.org, Oct. 2023

[29] Cook, Kees, "Kernel address space layout randomization.", Linux Security Summit, May. 2013

[30] Jung, Benedikt, et al, "Trusted monitor: Tee-based system monitoring.", 2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC). IEEE, pp. 1-8, Nov. 2022.

[31] Cerdeira, D., Martins, J., Santos, N., & Pinto, S, "{ReZone}: Disarming

{TrustZone} with {TEE} Privilege Reduction.", 31st USENIX Security Symposium (USENIX Security 22), pp. 2261-2279, Aug. 2022.

[32] David, Robert, et al, "Tensorflow lite micro: Embedded machine learning for tinyml systems.",Proceedings of Machine Learning and Systems, vol 3, pp. 800-811, Apr. 2021

[33] Konaite, Matshehla, et al, "Smart Hat for the blind with Real-Time Object Detection using Raspberry Pi and TensorFlow Lite.", Proceedings of the International Conference on Artificial Intelligence and its Applications, pp.1-6, Dec. 2021

[34] Idrissi, Idriss, Mostafa Mostafa Azizi, and Omar Moussaoui, "A lightweight optimized deep learning-based host-intrusion detection system deployed on the edge for IoT", International Journal of Computing and Digital System, Oct. 2022.

〈저자소개〉

최 지 민 (Jimin Choi) 정회원
2011년 2월: 대구가톨릭대학교 컴퓨터공학과 졸업
2013년 8월~현재: 삼성SDS 근무
2021년 3월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 시스템보안, 정보보호, 모바일보안

신 영 주 (Youngjoo Shin) 종신회원
2006년 2월: 고려대학교 컴퓨터학과 학사
2008년 2월: KAIST 전산학과 석사
2014년 8월: KAIST 전산학과 박사
2008년 4월~2017년 2월: 국가보안기술연구소 선임연구원
2017년 3월~2020년 8월: 광운대학교 컴퓨터정보공학부 조교수
2020년 9월~2022년 2월: 고려대학교 정보보호대학원 정보보호학과 조교수
2022년 3월~현재: 고려대학교 정보보호대학원 정보보호학과 부교수
〈관심분야〉 시스템 보안, CPU 마이크로아키텍처 취약점 분석, 클라우드 보안