메모리 트랩기법을 활용한 컨테이너 취약점 침입 탐지 프레임워크

Container Vulnerability Intruder Detection Framework based on Memory Trap Technique

최상훈*, 전우진*,박기웅** 1)
Sang-Hoon Choi, Woo-Jin Jeon and Ki-Woong Park

(05006) 서울특별시 광진구 능동로 209 세종대학교 정보보호학과 시스템보안 연구실 *
(05006) 서울특별시 광진구 능동로 209 세종대학교 정보보호학과 **
{csh0052, woojinjeon929}@gmail.com, woongbak@sejong.ac.kr

요 약

최근 클라우드 플랫폼을 효율적으로 사용하기 위한 컨테이너 기술들이 주목을 받고 있다. 컨테이너 가상화 기술은 기존 하이퍼바이저와 비교하였을 때 이식성이 뛰어나고 집적도가 높다는 장점을 가지고 있다. 하지만 컨테이너 가상화 기술은 하나의 커널을 공유하여 복수개의 인스턴스를 구동하는 운영체제 레벨의 가상화 기술을 사용하기 때문에 인스턴스 간 공유 자원 요소가 많아져 취약성 또한 증가하는 보안 문제를 가지고 있다. 컨테이너는 컴퓨팅 자원의 효율적 운용을 위해 호스트 운영체제의 라이브러리를 공유하는 특성으로 인해 공격자는 커널의 취약점을 이용하여 호스트 운영체제의 루트 권한 획득 공격이 가능하다. 본 논문에서는 컨테이너가 사용하는 특정 메모리 영역의 변화를 감지하고, 감지 시에는 해당 컨테이너의 동작을 중지시키는 메모리 트랩 기법을 사용하여 컨테이너 내부에서 발생되는 호스트 운영체제의 루트 권한 탈취 공격을 효율적으로 탐지 및 대응하기 위한 프레임워크를 제안한다.

Abstract

Recently container technologies have been receiving attention for efficient use of the cloud platform. Container virtualization technology has the advantage of a highly portable, high density when compared with the existing hypervisor. Container virtualization technology, however, uses a virtualization technology at the operating system level, which is shared by a single kernel to run multiple instances. For this reason, the feature of container is that the attacker can obtain the root privilege of the host operating system internal the container. Due to the characteristics of the container, the attacker can attack the root privilege of the host operating system in the container

※ 본 연구는 미래창조과학부의 재원으로 한국연구재단(NRF-2017R1C1B2003957) 및 SW중심대학(R7718-16-1005) 지원사업의 연구결과로 수행되었음

1) 교신저자

논문투고일자 : 2017. 06. 09 심사일자 : 2017. 06. 12 게재확정일자 : 2017. 06. 26

utilizing the vulnerability of the kernel. In this paper, we propose a framework for efficiently detecting and responding to root privilege attacks of a host operating system in a container. This framework uses a memory trap technique to detect changes in a specific memory area of a container and to suspend the operation of the container when it is detected.

키워드: 클라우드 컴퓨팅, 시스템보안, 메모리 포렌식
Keyword: Cloud Computing, System Security, Memory Forensic

1. 서론

최근 클라우드 플랫폼[1, 2]을 더 효율적으로 사용하기 위한 컨테이너 기반의 가상화 기술[3, 4]에 대한 관심이 증가하고 있다. 컨테이너 가상화 기술은 Linux 계열의 운영체제에서 제공하는 LXC(Linux Container)[5]를 기반으로 운영체제 레벨의 가상화 기법을 활용하는 것이다. 컨테이너 기반의 가상화 기술은 기존 하이퍼바이저[6, 7]와 비교하여 이식성이 뛰어나고, 높은 집적도를 가지고 있어 가벼우며, 낮은 사양의 환경에서도 자원 활용 효율이 높다는 장점을 가지고 있다. 최근 컨테이너 기반의 클라우드 환경에서 많이 활용되고 있는 Docker[8], Kubernetes[9, 10]는 기존의 컨테이너 개념에 Devops[11, 12] 환경 친화적인 특징을 더하여 높은 이식성과 여러 컨테이너를 관리하기 위한 오케스트레이션 기능을 확장시켰으며, 기존 컨테이너 기술의 장점인 낮은 오버헤드와 가상화의 장점인 독자적 서버 구성을 가능하게 한다. 컨테이너는 이러한 기술의 확장을 통해 낮은 오버헤드로 다양한 환경의 플랫폼을 구성 및 제공할 수 있게 되었다.

하지만 컨테이너 기술은 커널 영역을 공유하여 낮은 오버헤드로 독립적인 환경을 제공할 수 있는 장점이 있지만, 하나의 커널 영역을 공유함으로써 해당 커널의 취약점까지 공유하게 되어 공격자가 다른 사용자 영역의 컨테이너에 접근이 가능해져 부가적인 보안 문제들이 발생한다. 이에 관한 기존의 연구들은[13, 14] 컨테이너 내부에서 수행되는 연산을 모니터링 하여 보안 메커니즘을 제공한다는 장점을 가지고 있다. 하지만 이와 같은 연구에서 제안된 보안 매커니즘을 컨테이너 플랫폼에 적용하기 위해서는 악성 행위에 대한 사전 분석이 수행되어야 한다.

따라서 본 논문에서는 컨테이너가 공유하는 라이브러리 메모리 영역의 메모리 모니터링을 통해 컨테이너 탈출을 위한 바이너리 인젝션 행위에 관하여 탐지하고, 공유된 메모리 영역 변조 행위가 감지되었을 때 행위의 주체가 된 네임스페이스 분석을 통해 메모리를 변조시킨 컨테이너의 동작을 중지시키는 메모리 트랩 기반의 컨테이너 탈출 탐지 프레임워크 두 가지를 제안한다.

본 논문에서 제안한 두 가지의 메모리 트랩 기술은 먼저 어플리케이션 레벨에서 컨테이너의 탈출을 탐지 할 수 있는 유저 메모리 모니터링 기술과 커널 레벨에서 컨테이너의 탈출을 탐지 할 수 있는 디바이스 메모리 모니터링 기술로 나뉜다.

본 논문의 구성은 다음과 같다. 2장에서는 컨테이너 기술의 취약점과 그에 따른 관련 연구를 서술하고, 3장에서는 컨테이너의 탈출을 효율적으로 탐지 할 수 있는 메모리 트랩 기반 컨테이너 탈출 탐지 프레임워크 두 가지를 제안한다. 4장에서는 컨테이너 탈출 탐지에 관한 탐지 성능 실험결과에 관해 설명하고, 5장에서는 결론을 기술한다.

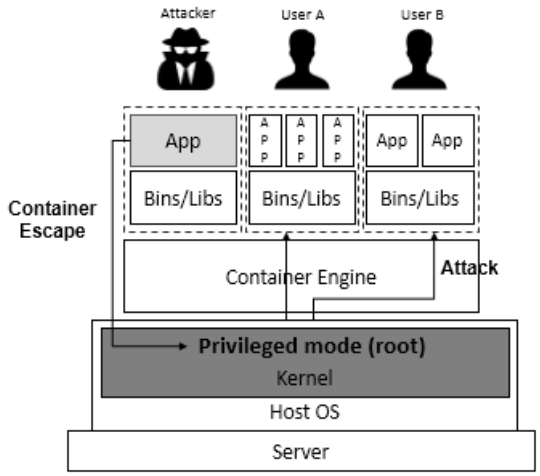
2. 컨테이너 보안 관련연구

본 장에서는 커널, 이미지와 같은 리소스 공유로 인해 발생할 수 있는 컨테이너 가상화의 보안 문제 해결에 관한 기존 연구들에 대해 설명하고 그에 관

한 한계점을 도출한다.

2.1 컨테이너의 리소스 공유에 따른 보안 위협

컨테이너는 파일 시스템, 네트워크 및 커널과 같은 호스트 운영체제의 리소스를 공유함으로써 뛰어난 이식성과 자원에 대한 효율성이 높다는 장점을 가지고 있으나, 공유되는 커널 라이브러리에 따른 부가적인 보안 위험이 발생한다는 단점이 있다. 따라서 컨테이너 보안의 가장 큰 위협은 컨테이너별 커널 구성이 불가능하다는 데서 비롯된다.



(그림 1) 컨테이너 탈출을 통한 특권모드 취득

(그림 1)과 같이 컨테이너는 시스템 운영 레벨에서의 가상화 기술이기 때문에, 호스트 운영체제의 리눅스 커널을 공유하여 사용한다. 따라서 컨테이너 기술에서는 사용자 컨테이너 프로세스가 호스트 운영체제의 관리자 권한에서 프로세싱 되었고, 컨테이너 내부에서 공유된 라이브러리에 대한 변조 또는 함수 실행 순서를 변조하여 컨테이너 탈출에 성공한다면 호스트 운영체제의 관리자 쉘 권한을 쉽게 획득할 수 있다. 호스트 운영체제의 관리자 쉘 권한을 획득한 공격자는 다른 사용자를 공격하거나 컨테이너가 구동되고 있는 서버 시스템에 장애를 일으키는 등의 치명적인 공격을 가할 수 있다.

2.2 LXC 보안 관련 연구

컨테이너의 보안 문제를 해결하기 위한 연구는 꾸준히 진행되어 왔다. 2015년 CNS[13]에서 연구 발표된 프레임워크에서는 리눅스 컨테이너 환경 내부에서 비정상적인 실행이 포착되면, 비정상적인 실행에 대해 분석 후 보안 프로필을 생성하여 보호 메커니즘을 제공한다. 하지만 위와 같은 프레임워크는 악성 행위가 실행된 후 추적하고 프로필을 생성하여 방어하는 기법을 사용하기 때문에, 알려진 공격에 효율적이라는 제한된 환경에 기반하고 있으며, 악성행위 탐지를 위한 사전 테스트가 필요하다는 한계점을 가지고 있다. 같은 해에 GCUX[14]에서 발표된 연구에서는 컨테이너 환경에서 SELinux 또는 AppArmor와 같은 기존 보안 모듈을 사용하여 컨테이너 내에서의 필수 접근 제어 보안 강화를 제안하였다. 하지만 이러한 기술은 여러 보안 요소 중 하나일 뿐이며, 이와 같은 보안 요소들로는 보안 요건에 대한 충족이 한정적이라는 한계점을 갖는다. 이와 같이 기존 연구들은 악성코드 분석이 선행 되어야 하고 컨테이너 환경에 특화되지 않은 기존 보안 모듈을 사용해야하기 때문에 확장성이 떨어진다.

3. 컨테이너 메모리 트랩 프레임워크

본 장에서는 공유된 라이브러리 영역 변조 행위에 대해 효율적으로 탐지 할 수 있는 두 가지 메모리 트랩 시스템에 대해서 제안한다. 첫 번째 시스템은 컨테이너 프로세스의 유저 메모리 영역 모니터링을 통해 메모리 변조에 대해서 탐지 할 수 있는 기법이다. 해당 기법은 구현이 쉽고, 다양한 컨테이너 엔진 내부에 보안 모듈로 적용 될 수 있다는 장점이 있다. 두 번째 시스템은 커널 메모리 영역 모니터링을 통해 공유 라이브러리 변조에 대해서 효율적으로 탐지 할 수 있는 시스템이다. 해당 시스템은 다양한 라이브러리 변조에 대해서 연산 효율적으로 탐지 할 수 있다는 장점이 있다.

3.1 공유 메모리 변조 탐지 기법

리눅스 컨테이너 기술은 단일 호스트 운영체제에서 여러 개의 독립된 리눅스 시스템들을 구동시키기 위한 운영체제 레벨의 가상화 기법이다. 리눅스 컨테이너 기술(LXC)은 리눅스 운영체제에서 제공해주는 cgroups[15]와 namespace[16] 기능을 사용하여 컨테이너 환경을 만든다. LXC 기술에서 프로세스 트리, 네트워크, 사용자 ID, 마운트 된 파일 시스템 등 운영 환경을 완전히 고립시키기 위해 namespace를 활용하고, cgroups를 활용하여 자원(CPU, Memory, Block I/O, Network)의 할당 범위를 운영 시스템 레벨에서 제한하여 사용자에게 격리된 컨테이너 환경을 제공한다. 이와 같이 격리된 컨테이너 환경에서는 컨테이너에서 필요로 하는 라이브러리를 메모리 영역에 중복 할당하는 것을 방지하기 위해 호스트 운영체제의 라이브러리를 공유하여 사용함으로써 컨테이너의 성능을 극대화 시킨다.

따라서 악의적인 목적을 가지고 있는 사용자는 호스트 운영체제 쉘 권한 획득을 위해 공유되는 컨테이너 라이브러리에 사용되지 않는 메모리 영역에 악의적인 바이너리를 인젝션 시키고, 함수의 실행 순서를 변조하여 컨테이너를 탈출 할 수 있다.

공유된 라이브러리의 메모리 변조를 탐지하기 위한 방법에는 크게 두 가지로 나눌 수 있다.

첫 번째는 메모리의 특정영역을 주기적으로 파일 덤프 후 분석하여 변화된 내용에 대해 모니터링 하는 방법이다. 컨테이너는 리눅스 호스트 운영체제 레벨에서 하나의 프로세스로 처리되기 때문에, GDB[17]와 같은 디버그 툴을 사용하면 컨테이너의 메모리 정보를 파일로 추출 가능하다.

두 번째는 여러 컨테이너에서 사용되어지는 공유된 라이브러리의 물리 메모리 영역 주소를 추출 하여, 디바이스 파일로 맵핑시킨 후 리눅스 커널 서브시스템인 inotify[18]를 활용하면 맵핑된 디바이스의 시스템 이벤트에 대해서 효율적으로 모니터링 할 수 있다.

3.2 컨테이너 탈출 취약점 분석

현재 일반적으로 사용되고 있는 컨테이너 기술은 컨테이너를 호스트 운영체제에서 관리자 권한으로 실행한다. 따라서 컨테이너의 공유 된 커널 취약점을 통해 호스트 운영체제의 관리자 권한 획득하여 호스트 운영체제 시스템의 모든 부분에 대한 제어 할 수 있다. 따라서 악의적인 목적을 가진 사용자의 컨테이너 탈출은 호스트 운영체제에서 구동중인 여러 컨테이너에 직접적인 공격을 가하거나, 컨테이너가 구동되고 있는 서버 시스템에 장애를 일으키는 등 치명적인 공격을 가할 수 있다.

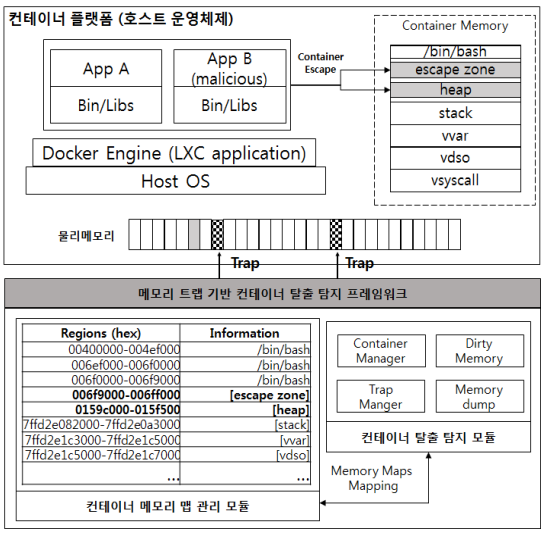
본 논문에서는 공유 라이브러리의 메모리 변조를 활용한 컨테이너 탈출 기법에 대해 효율적으로 탐지하기 위하여 컨테이너 탈출 익스플로잇을 분석하였다. 리눅스 커널의 Dirty-Cow[19, 20] 취약점을 이용하여 컨테이너를 탈출하는 익스플로잇은 컨테이너의 성능을 극대화하기 위해 활용하는 공유된 vDSO(virtual dynamic shared object) 라이브러리로부터 발생한다. vDSO는 컨테이너들의 시간 동기화를 위해 주로 사용되는 라이브러리이다. 이 라이브러리는 리눅스 커널의 취약점인 Dirty-Cow를 통해 변조 가능하다. 컨테이너를 탈출하기 위해 공유 라이브러리를 변조하는 과정은 다음과 같다. vDSO는 라이브러리를 공유하기 위해 메모리를 8kb 사이즈를 고정적으로 할당하여 컨테이너에게 제공해준다. 할당 된 8kb 영역에는 clock_gettime, get_time_of_day, vdso_time 등 컨테이너의 시간 동기화를 위해 사용되는 함수들에 대한 정보가 정의 되어있는데, 이러한 함수들은 8kb 중 4kb에만 정의되어 있고 나머지 메모리 영역은 빈 영역으로 남겨져 있다. 따라서 Dirty-Cow 컨테이너 탈출 취약점은 이러한 점을 활용하여 vDSO 메모리의 마지막 256byte 영역에 컨테이너의 init 네임스페이스(ipc, mnt, net, pid, user, uts) inode 정보를 추출하고, reverse_tcp를 통해 연결 할 수 있는 쉘 코드를 인젝션 하여 clock_gettime 함수의 실행 순서 변조를 통해 해당 쉘 코드를 실행하도록 함으로써

호스트 운영체제의 관리자 권한 쉘을 획득한다.

3.3 유저레벨 메모리 트랩 프레임워크 설계

컨테이너가 할당받아 사용하고 있는 유저 메모리 분석을 통해 컨테이너 탈출 시 변화되는 메모리 정보를 추출하였고, 메모리 로드 정보를 분석한 결과 두 가지 특이사항을 발견할 수 있었다.

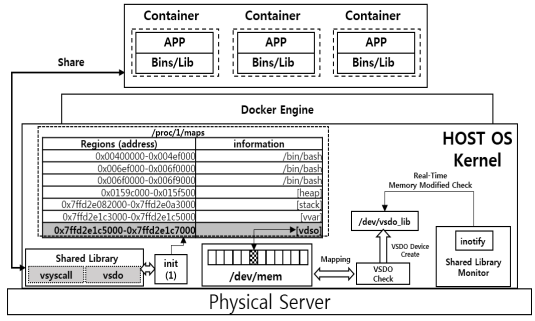
첫 번째는 컨테이너 탈출 시 프로세스(Container APP) 힙 영역의 vDSO 데이터 변조 관련 데이터가 로드 된다는 것이고, 두 번째는 컨테이너 내부에서 호스트 운영체제의 쉘을 실행하기 위하여 메모리의 고정된 데이터 영역(0x006f9000 – 0x006ff000)에 쉘 관련 데이터를 인젝션 시킨다는 것이다. 즉 컨테이너 내부에서 호스트 운영체제의 특권명령 쉘 권한을 획득하기 위해서는 반드시 두 개의 메모리 영역이 변조되어야한다. 본 논문에서는 이러한 메모리 분석결과를 바탕으로 메모리 트랩 영역을 추출하였다.



(그림 2) 컨테이너 탈출 탐지 프레임워크 구조

설계된 메모리 트랩 기반 컨테이너 탈출 탐지 프레임워크는 (그림 2)와 같다. 위와 같이 우리가 제안한 메모리 트랩 기반의 컨테이너 탈출 탐지 프레임워크는 크게 2개의 모듈로 구성되어있다. 컨테이너 메모리 맵 관리 모듈은 호스트 운영체제에서 구동되고 있는 컨테이너 APP에 대한 메모리 할당 정보를 분석하고, 컨테이너 탈출 시 변조되는 메모리 주소를 컨테이너 탈출 탐지 모듈에 전달한다. 컨테이너 탈출 탐지 모듈은 4가지 세부 모듈로 구성되어있다. 첫 번째, Trap Manager 모듈은 메모리 맵 관리 모듈로부터 전달받은 트랩이 설치되어야 할 컨테이너 메모리영역을 추출하며, 두 번째 Memory Dump 모듈은 컨테이너의 메모리 변화 정보를 감시하기 위한 메모리의 일부 영역을 파일로 추출한다. 세 번째, Dirty Memory 모듈은 덤프 된 메모리 정보를 분석하여 트랩 영역에 컨테이너 사용자가 접근하였는지 파악하고, 마지막 Container Manager 모듈은 컨테이너 사용자가 컨테이너 탈출을 시도하였을 때 해당 컨테이너를 강제 종료시키도록 설계하였다.

3.4 커널 레벨 메모리 트랩 프레임워크 설계



(그림 3) 커널 레벨 메모리 모니터링 모듈의 구조

커널 레벨 메모리 트랩 프레임워크는 컨테이너 플랫폼 환경에서 사용 될 수 있는 다양한 라이브러리 변조에 대한 위협을 커널 레벨에서 효율적으로 모니터링 할 수 있도록 (그림 3)과 같이 설계하였다. 컨테이너에서 사용하는 공유 라이브러리는 리눅스의 init 프로세스에서 동일하게 사용하고 있다. 따라서 init의 vDSO 메모리 영역과 컨테이너의 vDSO 메모리 영역은 동일한 물리적 메모리 주소에 맵핑되어있다. 그렇기 때문에 init의 maps 정보를 분석하여 vDSO의 가상 주소를 추출하고, pagemap

을 통해 물리주소로 변환하여 /dev/mem으로부터 vDSO의 물리 주소를 추출한다. 그 후 해당 물리주소를 페이지 크기(4kb)를 갖는 메모리 디바이스로 생성한다. 생성된 vDSO_lib 메모리 디바이스를 커널 서브시스템인 inotify를 통해 파일 시스템에 대한 이벤트 모니터링을 통해서 공유된 라이브러리(vDSO)의 악의적인 메모리 변조에 대해서 탐지하고 변조가 발생하였을 때 해당 메모리 영역을 본래의 라이브러리로 복원 할 수 있도록 설계하였다.

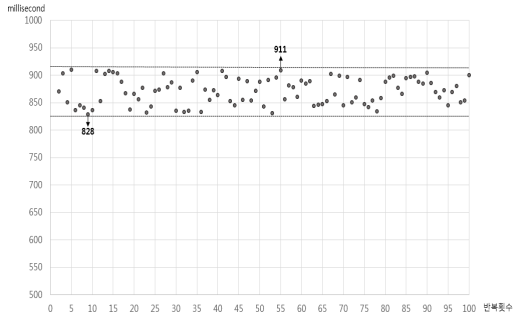
4. 컨테이너 탈출 탐지 성능 평가

본 장에서는 구현된 두 종류의 컨테이너 탈출 탐지 메모리 트랩 시스템에 대한 효용성을 검증하기 위해 메모리 트랩 모듈이 적재된 클라우드 플랫폼에서 컨테이너 탈출을 탐지하는데 소요되는 시간을 측정하였다. 실험은 대표적인 컨테이너 엔진인 Docker 1.3 환경에서 진행하였다.

4.1 실험환경

실험 환경에서는 CPU Xeon E5-2609(2.5GHz)와 32GB의 메모리, SSD 128GB의 하드웨어가 장착되어있는 서버를 사용하였다. Container APP이 구동되는 운영체제는 Ubuntu 14.04-64bit를 사용하였다. 본 실험에서는 vDSO의 취약점을 활용하여 컨테이너를 탈출하였다.

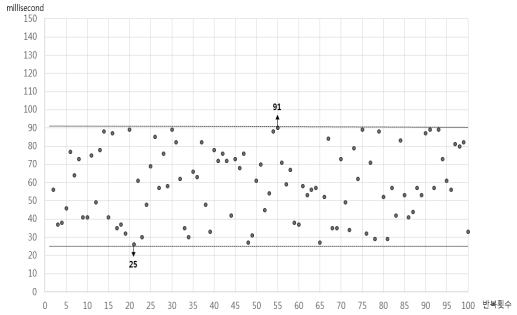
4.2 유저 레벨 컨테이너 탈출 감지 성능평가



(그림 4) 유저 레벨에서의 메모리 트랩 성능평가

본 논문에서 제안한 유저 레벨에서의 메모리 트랩을 통한 컨테이너 탈출 탐지 프레임워크의 성능을 검증하기 위해 컨테이너 탈출 공격을 시도 하였을 때 컨테이너 탈출 공격을 탐지하는데 소요되는 시간을 측정하였다. 총 100번의 컨테이너 탈출 공격을 시도하였고 실험에 대한 결과는 (그림 4)와 같다. 컨테이너 탈출 공격에 대한 탐지 소요 시간은 최소 828ms, 최대 911ms가 소요되었다. 유저 레벨의 모니터링은 컨테이너 탈출 공격이 시도되었을 때 평균 871ms이내에 공격을 탐지하였고 차단할 수 있었다.

4.3 커널 레벨 컨테이너 탈출 감지 성능평가



(그림 5) 커널 레벨에서의 메모리 모니터링 성능평가

본 장에서는 컨테이너 탈출 공격을 시도 하였을 때 커널 메모리 트랩이 모듈이 이를 탐지하는데 소요되는 시간을 측정하였다. 총 100번의 컨테이너 탈출 공격을 시도하였고 실험에 대한 결과는 (그림 5)와 같다. 커널 메모리 모니터링 기반 컨테이너 탈출 탐지 기술은 탈출부터 이를 탐지하는데 까지 최소 25ms, 최대 91ms의 시간이 소요되었다. 커널 메모리 모니터링 기반 컨테이너 탈출 탐지 모듈은 컨테이너 탈출 공격이 시도되었을 때 평균 59ms 이내에 공격을 탐지하고 차단할 수 있었다. 커널 레벨에서의 메모리 모니터링 기술은 유저 레벨에서의 메모리 모니터링 트랩 기술과 비교하여 약 14.7배 이상 시간적 성능 우위를 나타냈다.

5. 결론

본 논문에서는 컨테이너 환경에서 발생할 수 있는 컨테이너 탈출 공격을 효율적으로 탐지하기 위한 메모리 트랩 기법 두 가지를 제안하였다. 우리가 제안한 메모리 트랩 기법은 프로세스의 유저 메모리 영역에 트랩을 만들어 컨테이너의 탈출에 대해서 탐지 할 수 있는 기법과 컨테이너에서 공유되는 라이브러리의 물리 주소를 디바이스 파일로 맵핑시켜 이벤트를 모니터링 하는 기법이다. 본 논문에서 제안한 메모리 트랩 기반 컨테이너 탐지 프레임워크를 실제 컨테이너 실행 환경에 적용하기 위해 Docker 플랫폼에 적재하고 실험을 해보았다. 프로세스의 유저 메모리 영역 모니터링 프레임워크는 컨테이너의 탈출 공격을 탐지하는데 평균 871ms 이 소요되었고, 커널 레벨에서의 메모리 모니터링 기법은 평균 59ms 이내에 컨테이너 탈출을 탐지할 수 있었다. 이와 같은 실험 결과를 통해 우리가 제안한 메모리 트랩 기반의 컨테이너 탈출 탐지 프레임워크가 컨테이너의 보안위협을 탐지하는데 효율적이라는 것을 입증하였다. 추후 연구에서는 컨테이너 기술이 클라우드 플랫폼에서 하이퍼바이저와 융합되지 않고 단일 서비스로 활용 될 수 있도록 시스템 보안을 강화를 위한 부분 가상화 기술에 대해 연구를 수행할 예정이다.

▌참고문헌

〔1〕 Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011).

〔2〕 Armbrust, Michael, et al. "A view of cloud computing." Communications of the ACM 53.4 50-58, (2010).

〔3〕 Celesti, Antonio, et al. "Exploring container virtualization in IoT clouds." Smart Computing (SMARTCOMP), 2016 IEEE International Conference on. IEEE, 2016.

〔4〕 Seo, Kyoung-Taek, et al. "Performance comparison analysis of linux container and virtual machine for building cloud." Advanced Science and Technology Letters 66.105-111 2 (2014).

〔5〕 Helsley, Matt. "LXC: Linux container tools." IBM devloperWorks Technical Library 11. (2009).

〔6〕 Chisnall, David. The definitive guide to the xen hypervisor. Pearson Education, (2008)

〔7〕 Habib, Irfan. "Virtualization with kvm." Linux Journal 2008.166 8. (2008).

〔8〕 Merkel, Dirk. "Docker: lightweight linux containers for consistent development and deployment." Linux Journal 2014.239 2,(2014).

〔9〕 Bernstein, David. "Containers and cloud: From lxc to docker to kubernetes." IEEE Cloud Computing 1.3 81-84, (2014).

〔10〕 Sanchez, Carlos. "Scaling Docker with Kubernetes." Website. Available online at http://www. infoq. com/articles/scaling-docker-with-kubernetes 35. (2015).

〔10〕 Httermann, Michael. DevOps for developers. Apress, (2012).

〔11〕 Bass, Len, Ingo Weber, and Liming Zhu. DevOps: A Software Architect's Perspective. Addison-Wesley Professional, (2015).

〔12〕 Hayden, Major, and Richard Carbone. "Securing Linux Containers." GIAC (GCUX) Gold Certification, Creative Commons Attribution-ShareAlike 4.0 International License (2015).

〔13〕 Mattetti, Massimiliano, et al. "Securing the infrastructure and the workloads of linux containers." Communications and Network Security (CNS), 2015 IEEE Conference on. IEEE, (2015).

〔14〕 Hayden, Major, and Richard Carbone. "Securing Linux Containers." GIAC (GCUX) Gold Certification, Creative Commons Attribution-

ShareAlike 4.0 International License (2015).
〔15〕 Menage, Paul, Paul Jackson, and Christoph Lameter. "Cgroups." Available on-line at: http://www. mjmwired. net/kernel/Documentation/cgroups. txt (2008).
〔16〕 Biederman, Eric W., and Linux Networx. "Multiple instances of the global linux name-spaces." Proceedings of the Linux Symposium. Vol. 1. 2006.
〔17〕 GDB, https://www.gnu.org/s/gdb
〔18〕 Love, Robert. "Kernel korner: Intro to inotify." Linux Journal 2005.139 8. (2005).
〔19〕 Container Escape Code, https://github.com/scumjr/dirtycow-vdso
〔20〕 Dirty Cow, https://dirtycow.ninja/

저자소개

◆ 최상훈

- 2014년 대전대학교 정보보안학과 학사
- 2016년 대전대학교 전산정보보안학과 석사
- 2017년 세종대학교 정보보호학과 박사 과정
- 관심분야: 클라우드 컴퓨팅, 시스템 보안, 디지털 포렌식 등

◆ 전우진

- 2015년 대전대학교 정보보안학과 학사
- 2017년 세종대학교 정보보호학과 석사 과정
- 관심분야: 클라우드 컴퓨팅, 시스템 보안, 데이터베이스 등

◆ 박기웅

- 2005년 연세대학교 Computer Science 학사
- 2007년 KAIST Electrical Engineering 석사
- 2012년 KAIST Electrical Engineering 박사
- 2008년 Microsoft Research Asia, Wireless and Networking Grourp, Research Intern
- 2009년 Microsoft Research, Network Research Group, Graduate Research Fellow
- 2012년 국가보안기술연구소 연구원
- 2012년~2016년 대전대학교 정보보안학과 교수
- 2016년~현재 세종대학교 정보보호학과 교수
- 관심분야: 시스템 보안, 모바일-클라우드 컴퓨팅, 보안 프로토콜, 디지털 포렌식 등