하드웨어 지원 가상화 환경에서의 시스템 콜 인터셉트 기법1)

이동우, 김인혁, 엄영익
성균관대학교 정보통신공학부
{cowsboys, kkojiband, yieom}@ece.skku.ac.kr

A Method of Intercepting System Calls on Hardware-assisted Virtualization Environment

Dongwoo Lee, Inhyuk Kim and Young Ik Eom
Sunkyunkwan Univ. School of Information and Communication Eng.

요 약

최근 많은 보안 공격 도구들은 커널 레벨에서 동작하도록 설계 되고 있다. 악의적인 행동이 시스템 모니터링 프로그램과 같은 권한 레벨에서 이루어지기 때문에 공격 도구들은 공격 대상으로부터 자신의 존재를 숨기고 활동할 수 있다. 이러한 커널 레벨 악성코드들에 대한 대처 방법으로 가상화 기술을 이용하는 더 높은 권한 레벨을 가진 가상머신모니터(VMM)에서의 시스템 모니터링 방법이 소개 되었다. 그러나 많은 가상화 기반 모니터링 도구들이 주로 페이지 폴트 예외를 이용하여 시스템 콜 호출을 감시하기 때문에 신뢰성이 떨어지고 확장 페이지 테이블 (EPT) 과 같은 최신 하드웨어 가상화 기술 지원을 받지 못한다. 이에 본 논문에서는 일반 보호 예외를 이용하는 새로운 시스템 콜 인터셉트 기법을 제안함으로써 기존 기법의 기술적 한계를 극복하고 신뢰도 높은 모니터링 기법을 제공하고자 한다. 또 해당 기법의 구체적 구현 방법을 서술하고 구현 결과를 실험함으로서 제안 기법을 검증한다.

1. 서 론

최근의 시스템 보안 공격 도구 및 악성코드들은 커널 레벨에서 동작하도록 설계 되고 있다. 이러한 악의적인 프로그램들은 루트킷을 이용해 커널의 주요 부분을 수정함으로써 자신의 동작을 숨기고 운영체제 전반의 통제권을 갖는다. 이를 통해 응용레벨에서 동작하는 보안 시스템을 회피함은 물론 커널레벨에서 동작하는 보안 시스템까지 간단히 무력화 시킬 수 있게 된다.

커널 레벨에서 이루어지는 공격에 대한 대응책으로 가상화 기술을 사용하는 시스템 모니터링 기법[1]들이 많이 연구 되었다. 이와 같은 모니터링 툴에서는 시스템 콜을 가로채기 위한 수단으로 페이지 폴트 예외를 이용[2]한다. 시스템 콜의 실제 시작 지점(Entry)대신 유효하지 않은 주소 영역에서 해당 시스템 콜이 동작 하도록 하여 페이지 폴트를 유발 시키고, 이를 통해 가상머신 모니터로 실행 모드를 전달하는 방법이다. 그러나 페이지 폴트 예외를 일으키기 위해 지정한 임의의 주소가 유효한 주소 영역인 경우 모니터링 시스템이 올바르게 동작하지 않음은 물론 게스트 시스템 전체가 오작동을 일으키게 된다.

또한 인텔의 최근 네할렘(Nehalem)[3] 마이크로아키텍처 기반 프로세서에는 확장 페이지 테이블 (Extended Page Table, EPT)[4]이라는 새로운 하드웨어 지원 가상화 기술이 포함 되었다. EPT 지원을 받는 가상화 환경에서는 게스트 시스템이 자체적으로 페이지 폴트 예외를 처리하기 때문에 이를 이용한 가상 머신 모니터로의 실행 전환이 불가능하다.

본 논문에서는 페이지 폴트 예외를 이용한 시스템 콜 인터셉트 기법의 문제점을 해결 할 수 일반 보호 예외를 이용한 새로운 기법을 제안한다. 이 방법은 인텔 ISA(Instruction Set Architecture)에서 정의한 SYSENTER 명령의 동작을 이용하기 때문에 정확하게 시스템 콜을 가로챌 수 있으며, 부가적으로 EPT기술의 지원도 받을 수 있다.

우리는 제안 기법을 하드웨어 지원 가상화를 이용하는 오픈소스 가상머신모니터, KVM[5] 환경에서 구현 하였다. 구현 결과를 통해 페이지 폴트 예외를 이용한 방식과 일반 보호 예외를 이용한 방법의 시스템 콜 호출 성능을 비교하고, SPEC INT 2006 벤치마킹 소프트웨어를 통해 전체 시스템의 성능을 비교 한다.

2. 관련 연구

2.1 가상화 기반 모니터링 기법

가상화 환경에서 가상머신 모니터 게스트 시스템보다 높은 실행 권한을 갖기 때문에 게스트 시스템에 대한 메모리 보호와 시스템 이벤트 선점이 가능하다. 이러한 특징을 이용하여 모니터링 대상이 되는 가상머신을 더 높

1) 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음" (NIPA-2010-(C1090-1021-0008))

은 권한을 통해 감시함으로써 악의적 공격으로부터 보호할 수 있다.

가상화 환경을 이용한 모니터링 프로그램은 크게 수동적 모니터링 방법[6, 7]과 능동적 모니터링 방법[8, 9]으로 나눌 수 있다. 수동적 모니터링 방법은 게스트 가상머신이 악의적 프로그램에 의해 공격이 성공 했을 경우에만 탐지가 가능한 사후 탐지 기법이다. 보안 프로그램에 있어 사후 탐지는 큰 의미를 갖지 못하므로 게스트 시스템의 동작을 가로채어 미리 악의적 공격을 차단하는 방식의 능동적 모니터링 방법이 널리 쓰이고 있다.

능동적 모니터링 방법은 시스템 콜 호출과 같은 시스템 보안에 영향을 줄 수 있는 동작이 실행되기 전에 중간에서 가로채어 해당 동작이 악의적 특성이 없는지 확인 후 실행 여부를 판단 할 수 있게 한다.

2.2 XenFIT

XenFIT은 Xen기반의 파일 무결성 보호/탐지 도구 이다. 실시간으로 동작하는 모니터링 도구이다. 모니터링 대상이 되는 도메인에는 어떠한 모듈이나 프로그램도 설치하지 않기 때문에 해당 도메인이 스스로 탐지되고 있는지 확인하는 것이 매우 어렵다. 이를 통해 악의적 공격으로부터 효과적으로 해당 도메인을 보호 할 수 있다.

XenFIT은 시스템 동작을 모니터링 하기 위해 중단점(Breakpoint)을 사용한다. 커널 메모리 영역에 중단점 설정하여 시스템 콜 혹은 인터럽트 등에 의해 이 영역에 대한 접근 시도가 발생하면 해당 동작을 탐지하고 이를 신뢰 영역에 있는 xenfitd라는 프로그램에 의해 탐지 및 기록 된다. 그러나 매 커널영역에 대한 접근마다 중단점에 의해 시스템의 수행이 중단 되므로 모니터링 동작에 대한 성능 부하가 매우 크다.

2.3 VMFence

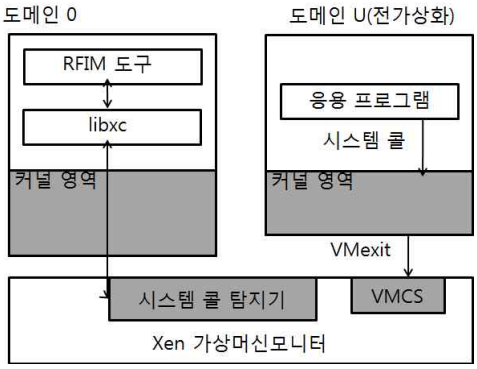

그림 1 VMFence 동작 구조도

VMFence는 XenFIT과 유사한 Xen 가상머신 모니터를 기반으로 하는 모니터링 프로그램으로 어떤 악의적인 공격도 파일 시스템에 접근하는 것에 초점을 둔다. 이 프로그램은 리눅스에서 사용되고 있는 사용자/그룹/일반의 파일 권한과 별개로 중요/확인/허용 세 가지로 파일을 분류하여 관리하는 하는 새로운 파일 관리 정책을 제공한다.

Breakpoint를 사용하는 XenFIT과는 달리 하드웨어 가상화 기술의 지원을 통해 시스템 콜을 가로채어 작동한다. 시스템 초기화 과정에서 SYSENTER명령 수행에 필요한 MSR_SYSENTER_EIP를 0으로 설정하여 시스템 콜이 호출 되었을 때 의도적으로 페이지 폴트 예외를 발생 시킨다. 이때 실행 모드가 Xen으로 넘겨지게 되고 이 정보를 신뢰할 수 있는 도메인에 전달하여 모니터링 대상이 되는 도메인 레지스터 정보를 확인한다. 확인 결과가 파일 관련 시스템 콜일 경우 인자들을 확인 하고 미리 정의 해둔 세 가지 형태의 파일 관리 정책을 바탕으로 해당 시스템 콜을 처리한다. 해당 파일이 중요 정책인 경우에는 해당 시스템 콜의 수행을 중단 시키며, 확인 정책인 경우에는 해당 시스템 콜의 수행을 로그에 기록한다. 허용 정책인 파일의 경우에는 아무 제약 없이 그대로 수행 한다.

2.4 SIM(Secure In-VM Monitoring)

SIM은 KVM을 기반으로 하는 모니터링 도구이다. SIM의 가장 큰 특징은 모니터링 대상이 되는 시스템에 직접 모니터링 도구를 삽입하여 가상머신간의 전환 부하를 줄인데 있다. 게스트 시스템 내부에 메모리를 할당하여 이 부분에 모니터링 도구를 설치하고 가상머신 모니터를 이용해 이 메모리 부분을 감추고 보호한다.

시스템에 보안 관련 동작이 탐지 되면 VMEXIT를 발생시켜 시스템에 대한 제어 권을 이양하는 것이 아니라, 시스템 내부에서 시작/종료 관문(Entry/Exit Gate)이라는 모듈 코드를 통해 시스템을 모니터링 할 수 있다. 가상머신 간의 전환에 따른 시스템 부하를 현격히 줄였지만, 모니터링 시스템에 모듈을 설치해야 한다는 점에서 악의적 공격자로부터 시스템 공격 지점을 제공한다는 단점이 있다.

3. 제안기법

3.1 SYSENTER 동작

x86 아키텍처는 인터럽트를 이용한 전통적인 방식과 펜티엄2 이후부터 지원된 SYSENTER[11]명령을 이용한 방식의 두 가지 시스템 콜 호출 방법을 제공한다. SYSENTER 명령은 MSR(Model Specific Register)을 사용하여 빠르게 시스템 콜을 호출하기 위한 동작을 한다. 시스템 콜에 사용되는 MSR은 MSR_SYSENTER_CS, MSR_SYSENTER_EIP, MSR_SYSENTER_ESP가 있다.

이전의 많은 시스템 모니터링 프로그램들은 시스템 콜을 가로채기 위해 시스템 콜 핸들러의 시작 주소가 담긴 MSR_SYSENTER_EIP을 임의의 값으로 변경한다. 이 후에 게스트 시스템에서 시스템 콜이 호출 되면 페이지 폴트 예외가 발생하고 VMEXIT를 통해 가상머신 모니터로 실행 모드가 넘어오게 된다. 가상머신 모니터는 게스트 시스템이 저장해놓은 시스템 상태 정보를 가지고 어떠한 인자를 가지고 어떠한 시스템 콜이 호출 되었는지를 알 수 있으므로, 이 정보를 바탕으로 시스템 모니터링 동작을 수행 한다.

그러나 페이지 폴트 예외를 발생시키기 위한 임의의

MSR_SYSENTER_EIP값은 유효한 주소 범위 내의 값이어야 하므로 시스템 콜을 항상 정확하게 가로 챌 것이라는 보장을 할 수 없다. 이에 우리는 페이지 폴트 예외대신 일반 보호 예외를 이용하여 시스템 콜을 가로챌 수 있는 기법을 제안한다.

```
SYSENTER:

if CR.0.PE == 0 then #GP(0);
if SYSENTER_CS_MSR == 0 then #GP(0);

EFLAGS.VM <- 0;
EFLAGS.IF <- 0;
…
SS.ATTR.P <- 1;

ESP <- SYSENTER_ESP_MSR;
EIP <- SYSENTER_EIP_MSR;
```

그림 2 SYSENTER 명령어 동작 과정

그림 2에 명시된 바와 같이 SYSENTER명령이 호출 되었을 때 SYSENTER_CS_MSR의 값이 0일 경우 일반 보호 오류를 발생시킨다. 이를 이용해 페이지 폴트 예외를 이용한 방법과 유사하게 시스템 콜을 인터셉트 한다. 자세한 인터셉트 과정은 3.3절에 기술 하도록 한다.

3.2 확장 페이지 테이블 지원

게스트 시스템의 물리 주소 공간은 호스트 시스템 입장에서는 또 하나의 가상 주소 공간이다. 이 때문에 게스트 시스템은 물리 주소를 사용하는 CR3 레지스터 및 페이지 테이블을 그대로 사용 할 수 없다. 이를 해결하기 위해 매 페이지 폴트마다 가상머신 모니터로 실행을 전환하여 섀도우 페이징(Shadow Paging)기법을 이용하여 게스트 시스템의 페이징을 지원 하였다.

최신 인텔 ISA 아키텍처인 네할렘 부터는 하드웨어 지원 가상화 기술의 일환으로 EPT를 지원하기 시작했다. EPT는 프로세서가 게스트 시스템을 동작시키고 있을 경우 게스트 시스템의 물리 주소를 호스트의 물리 주소로 변환해 주는 기능을 한다. 이를 통해 게스트 운영체제는 가상머신 모니터를 거치지 않고도 직접 페이지 폴트 예외를 처리 할 수 있다.

이처럼 EPT지원을 받는 가상화 환경에서는 페이지 폴트 예외를 통한 가상머신 모니터로 실행 전환이 불가능 하기 때문에 이를 이용한 시스템 콜 인터셉트가 불가능 하다. 본 논문에서 제안하는 기법은 페이지 폴트 예외를 이용하지 않기 때문에 EPT를 사용하는 시스템에서도 시스템 콜을 가로챌 수 있다.

3.3 일반 보호 예외를 이용한 시스템 콜 인터셉트

앞서 설명한 바와 같이 SYSENTER_CS_MSR의 값이 0일 때 SYSENTER 명령어를 수행하려고 하면 일반 보호 예외가 발생한다. 이를 이용해 게스트 시스템을 초기화 하는 과정에서 레지스터의 값을 0으로 설정한다. 모니터링이 끝난 후 정상적으로 SYSENTER를 수행하기 위해서는 MSR_SYSENTER_CS의 값을 복원해야 하므로 초기화 데이터를 저장해 둔다.

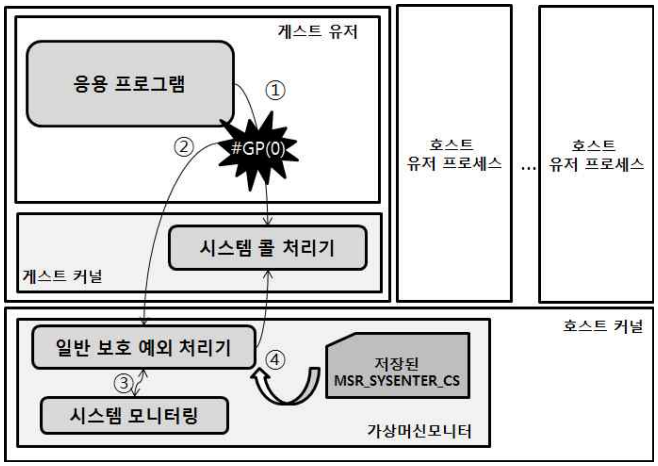

그림 3 제안 기법의 시스템 콜 인터셉트 과정

시스템이 초기화 된 이후 게스트 시스템에서 실제로 시스템 콜이 호출 되면 그림 5와 같이 일반 보호 예외가 발생하게 되고 가상머신 모니터로 실행 모드가 전환 된다. 가상머신 모니터에서는 게스트 시스템의 상태 정보를 통해 게스트 시스템이 어떤 시스템 콜 요청을 했는지 확인 하여 시스템 모니터링 동작을 수행한다. 해당 시스템 콜에 대한 모니터링 동작이 끝나고 해당 시스템 콜이 유효한 것으로 판단된 경우 해당 시스템 콜 요청을 올바르게 수행해야 하므로 저장된 데이터를 이용하여 MSR_SYSENTER_CS 레지스터를 복원 한 후 다시 게스트 시스템으로 실행 모드를 전환 한다. 게스트 시스템은 다시 SYSENTER 명령을 수행하여 시스템 콜 호출을 완료한다.

4. 구현

우리는 본 논문에서 제안하는 시스템 콜 인터셉트 기법의 성능과 안정성을 평가하기 위해 가상머신 모니터에 구현 하였다. 가상머신 모니터는 리눅스에 빌트-인(Built-in)형태로 제공되는 KVM을 이용하였다.

4.1 초기화 과정

운영체제가 초기화 되는 과정에서 WRMSR 명령을 사용하여 각종 MSR 레지스터를 설정 한다. WRMSR 명령어는 최상위 특권 레벨에서만 사용 가능 하므로 게스트 시스템에서 WRMSR 명령을 수행 하려고 할 때 실행모드가 가상머신 모니터로 넘어와 이를 처리해 주게 된다. 우리는 이 과정을 가로 채어 MSR_SYSENTER_CS 레지스터의 값을 0으로 설정하고 별도의 저장공간에 운영체제가 설정한 레지스터 값을 저장해 둔다.

또한 KVM의 초기 예외 벡터는 일반 보호 예외를 포함하고 있지 않으므로 일반 보호 예외가 발생 하더라도 가상머신으로 실행 모드가 전환되지 않는다. 그러므로 초기화 과

정에서 예외 벡터에 GP_VECTOR를 포함 시켜 일반 보호 예외가 발생했을 때 가상 머신 모니터로 실행 모드가 전환 되도록 한다. 이 때 별도의 임시 공간에 초기 예외 벡터 값을 별도의 저장공간에 저장하여 모니터링이 끝났을 경우 설정을 복원 할 수 있도록 한다.

4.2 일반 보호 예외 처리

초기화 과정을 거친 후 게스트 시스템이 초기화 되면, 처음 시스템 콜을 호출 할 때 일반 보호 예외가 발생한다. 일반 보호 예외 역시 예외 벡터에 등록해 두었으므로 가상 머신 모니터로 실행 모드가 전환되고 KVM의 일반 보호 예외 처리 함수가 호출 된다. 이 과정에 시스템 모니터링 관련 동작을 추가 하여 시스템 보안 동작을 수행 할 수 있다.

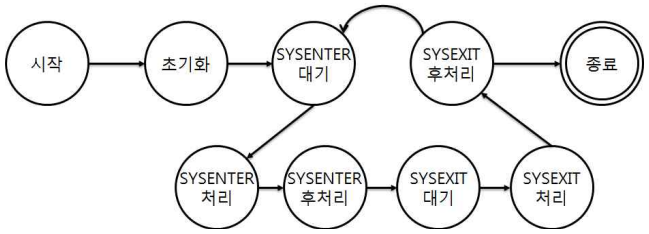

그림 6 시스템 콜 인터셉트 상태 천이도

SYSENTER를 수행한 경우를 제외하고도 의도하지 않게 일반 보호 예외가 발생 할 수 있다. 이러한 경우를 처리하기 위해 구현에는 인터셉트 단계를 6단계로 구분 하였다. 위 그림 3과 같은 상태 천이에서 실제 일반 보호 예외를 확인 하는 상태는 SYSENTER 대기와, SYSEXIT 대기뿐으로 실제 게스트 시스템의 동작 시간에 아주 짧은 부분만을 차지한다.

그림 6에서 SYSENTER 뿐만 아니라 SYSEXIT도 처리를 해주는데 이것은 시스템 콜 호출에 있어 SYSENTER와 SYSEXIT가 항상 쌍으로 수행되기 때문이다. SYSEXIT명령도 SYSENTER와 동일하게 동작 과정에서 MSR_SYSENTER_CS의 유효성을 검사하기 때문에 SYSENTER와 같은 처리 과정이 필요하다.

4.3 시스템 콜 인터셉트

SYSENTER명령에 의해 일반 보호 예외가 발생 한 경우 이를 처리 하고 게스트 시스템에서 다시 명령을 수행 할 때 이미 MSR_SYSENTER_CS가 복구 된 뒤 이므로 다시 SYSENTER를 가로챌 수 없게 된다. 이를 해결하기 위해 트랩 플래그(TF)를 이용한다. 게스트 시스템에 다시 실행 권한을 넘겨 줄 때 트랩 플래그를 설정하여 SYSENTER 명령이 수행된 후에 다시 한 번 가상 모니터로 실행을 전환 한다. 이때는 이미 SYSENTER명령이 성공적으로 수행된 후 이므로 다시 MSR_SYSENTER_CS를 0으로 설정할 수 있다. 이를 통해 다음 SYSEXIT가 수행 되었을 때 이를 가상머신 모니터에서 탐지 할 수 있으며, 같은 과정을 통해 계속 해서 SYSENTER를 탐지 할 수 있다.

5. 실험 및 평가

본 논문에서 제안하는 시스템 콜 인터셉트 기법의 성능 부하를 측정하기 위해 자주 쓰이는 시스템 콜 몇 가지를 선정하여 인터셉트를 하지 않았을 경우와 각각의 방법을

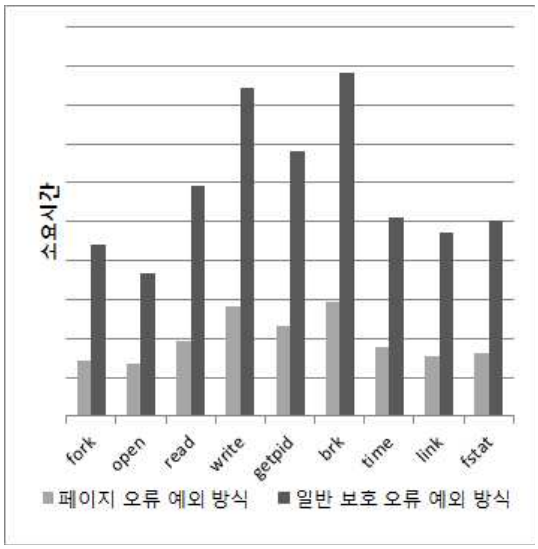

그림 4 시스템 콜 호출 부하 실험 결과

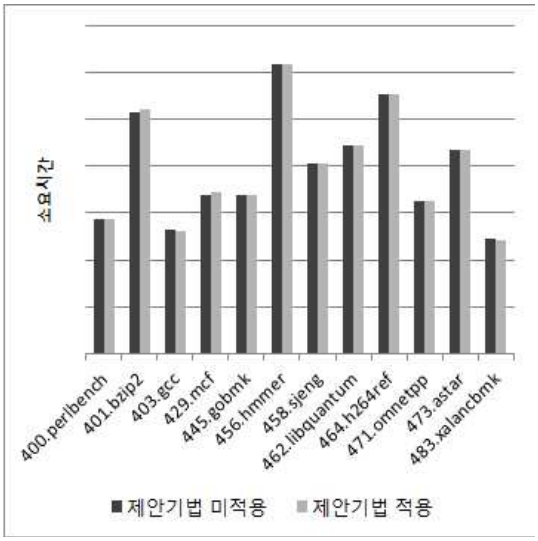

그림 5 SPEC CPU 2006 벤치마크 결과

이용해 인터셉트를 했을 때의 수행 시간을 측정 해 보았다. 각각의 시스템 콜을 100000회 호출했을 때 소요된 시간을 측정 하였으며 그 결과는 그림 4와 같다.

인터셉트를 하지 않았을 경우에는 EPT 사용 여부에 상관없이 아주 적은 시간이 소모 되었다. 인터셉트를 하는 경우에 제안 기법을 사용한 인터셉트는 페이지 폴트 예외를 사용한 기존의 방법에 비해 평균 294%의 수행 시간을 더 소모 하였다. 이것은 가상 머신 모니터로의 실행 전환이 한번만 필요한 기존 기법에 비하여 두 번의 전환 시간을 필요로 하는 제안 기법의 특징에 기인 한다.

본 제안 기법의 시스템 콜 호출에 따른 부하가 3배가량 더 큰 것으로 확인 되었다. 실제로 이러한 부하가 전체 시스템에 미치는 영향을 확인하기 위해 SPEC CPUT 2006 벤치마크 소프트웨어를 이용해 SPEC INT 2006 테스트를 진행 하였다. 그림 5의 결과에서 확인 할 수 있듯이, BZIP2와 MCF와 같이 일부 시스템 콜 호출이 많은 응용 프로그램을 제외 하고는 전체 성능 부하는 3% 미만으로 성능 차이가 거의 나타나지 않았다. EPT기술에 의한 성능 향상을 감안 할 때 이러한 부하는 무시 할 만큼 작다고 할 수 있다.

6. 결론

기존의 시스템 모니터링 프로그램은 하드웨어 중단점이나 페이지 폴트를 사용하여 시스템 콜을 인터셉트 했다. 이러한 기법들은 낮은 성능과 안정성이 문제가 되었다. 이에 본 논문에서는 일반 보호 예외를 이용한 시스템 콜 인터셉트 기법을 제안하여 최신 하드웨어 가상화 기술의 지원을 받을 수 있도록 하였다. 또한 이를 실제 환경에 구현 하고 실험 해 봄으로써 제안 기법의 낮은 성능 부하와 안정성을 검증 할 수 있었다.

그러나 기존 페이지 폴트 예외를 사용한 기법에 비해 현저히 떨어지는 인터셉트 성능은 개선의 여지가 있으며, 최신 하드웨어 가상화 기술을 지원하지 않는 시스템에 대한 보완이 필요하다. 향후 연구로는 본 제안 기법을 적용한 시스템 모니터링 기법에 관한 연구를 제안한다.

참고문헌

[1] T. Garfinkel and M. Rosenblum "A virtual machine introspection based architecture for intrusion detection," In Proc. of the Network and Distributed Systems Security Symposium, 2003.

[2] H. Jin, G. Xiang, D. Zou, F. Zhao, M. Li and C. Yu "A guest-transparent file integrity monitoring method in virtualization environment" Computers & Mathematics with Applications, Vol. 60, No. 2, pp.256-266, July 2010.

[3] Intel® Microarchitecture, Codenamed Nehalem. http://www.intel.com/technology/architecture-silicon/next-gen/

[4] Intel® Virtualization Technology: Hardware Support for Efficient Processor Virtualization http://www.intel.com/technology/itj/2006/v10i3/1-hardware/8-virtualization-future.htm

[5] Kernel based Virtual Machine. http://www.linux-kvm.org/page/Main_page

[6] X. Jiang, D. Xu, and X. Wang, "Stealthy malware detection through vmm-based 'out-of-the-box' semantic view reconstruction," In Proc. of the ACM conference on Computer and Communications Security, 2007.

[7] N. L. Petroni and M. Hicks. "Automated detection of persistent kernel control-flow attacks," In Proc. of the ACM conference on Computer and Communications Security, 2007.

[8] N. Anh Quynh and Y. Takefuji "A novel approach for a file-system integrity monitor tool of Xen virtual machine" In Proc. of ACM symposium on Information, computer and communications security, pp.194-202, 2007

[9] B.D. Payne, M. Carbone, M. Sharif, and Wenke Lee "Lares: An Architecture for Secure Active Monitoring Using Virtualization" In Proc. of IEEE Symposium on SP 2008., pp. 233-247, May 2008

[10] Intel(r) 64 and IA-32 Intel Architecture Software Developer's Manual Volume 2B: Instruction Set Information