(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2025년05월27일
(11) 등록번호 10-2813096
(24) 등록일자 2025년05월22일

(51) 국제특허분류(Int. Cl.)
G06F 21/53 (2013.01) *G06F 21/64* (2013.01)
G06F 9/30 (2018.01) *G06F 9/4401* (2018.01)
G06F 9/455 (2018.01)
(52) CPC특허분류
G06F 21/53 (2013.01)
G06F 21/64 (2013.01)
(21) 출원번호 10-2022-0162199
(22) 출원일자 2022년11월29일
심사청구일자 2022년11월29일
(65) 공개번호 10-2024-0080245
(43) 공개일자 2024년06월07일
(56) 선행기술조사문헌
Richard Li et al., "Fluorescence: Detecting Kernel-Resident Malware in Clouds"(2019.)*
Nick L. Petroni, Timothy Fraser, Jesus Molina, and William A. Arbaugh, "Copilot - a coprocessor-based kernel runtime integrity monitor"(2004.08.)*
최상훈, 박기웅, "클라우드 메모리 덤프 가속을 위한 API 설계 및 구현"(2015.11.)*
Asit More and Shashikala Tapaswi, "Virtual machine introspection: towards bridging the semantic gap"(2014.10.)*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
세종대학교산학협력단
서울특별시 광진구 능동로 209 (군자동, 세종대학교)
(72) 발명자
박기웅
서울특별시 광진구 능동로17길 21, 304호(화양동)
조여름
경기도 의정부시 호국로1183번길 45, 2층(가능동)
최상훈
서울특별시 광진구 긴고랑로2길 38-2, 301호(중곡동)
(74) 대리인
양성보

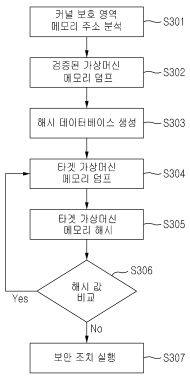
전체 청구항 수 : 총 2 항 심사관 : 정성훈

(54) 발명의 명칭 **클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 시스템**

(57) 요 약

클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 시스템이 개시된다. 가상머신 커널 런타임 보호 시스템은, VMI(Virtual Machine Introspection)를 이용하여 클라우드 환경의 가상머신 커널 런타임 무결성 검증을 수행하는 것으로, 리눅스 커널이 구동되는 가상머신의 메모리 주소 정보를 분석하는 과정; 및 상기 메모리 주소 정보를 이용하여 가상머신의 커널 변조 무결성을 검증하는 과정을 처리할 수 있다.

대 표 도 - 도3



(52) CPC특허분류
G06F 9/3012 (2013.01)
G06F 9/4406 (2013.01)
G06F 9/45504 (2013.01)
G06F 9/45545 (2013.01)
G06F 9/45554 (2013.01)
G06F 9/45558 (2013.01)
G06F 2009/45587 (2019.08)
G06F 2009/45591 (2019.08)

이 발명을 지원한 국가연구개발사업
과제고유번호 1711152732
과제번호 IITP-2021-0-01816-002
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보통신방송혁신인재양성
연구과제명 메타버스 자율트윈 핵심기술 연구
기 여 율 40/100
과제수행기관명 세종대학교 산학협력단
연구기간 2022.01.01 ~ 2022.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711161570
과제번호 NRF-2020R1A2C4002737
부처명 과학기술정보통신부
과제관리(전문)기관명 한국연구재단
연구사업명 중견연계연구지원사업
연구과제명 IoT 침해사고 대응을 위한 지능형 분석 플랫폼 기술 연구
기 여 율 20/100
과제수행기관명 세종대학교 산학협력단
연구기간 2020.03.01 ~ 2023.02.28
이 발명을 지원한 국가연구개발사업
과제고유번호 1711174187
과제번호 001657941G0003072
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보통신방송기술국제공동연구(R&D)
연구과제명 랜섬웨어 침해사고 전주기적 능동대응을 위한 다각적 수집-분석-대응 플랫폼 개발
기 여 율 20/100
과제수행기관명 세종대학교 산학협력단
연구기간 2022.07.01 ~ 2024.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711093714
과제번호 2019-0-00426
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보보호핵심원천기술개발사업
연구과제명 IoT 기반 이식-침습형 고위험 의료장치를 위한 능동형 킬 스위치 및 바이오 마커 활용 방어 시스템 개발
기 여 율 20/100
과제수행기관명 국민대학교산학협력단
연구기간 2019.04.01 ~ 2022.12.31

공지예외적용 : 있음

명 세 서

청구범위

청구항 1

컴퓨터 시스템으로 구현되는 가상머신 커널 런타임 보호 시스템에 있어서,

메모리에 포함된 컴퓨터 판독가능한 명령들을 실행하도록 구성된 적어도 하나의 프로세서

를 포함하고,

상기 적어도 하나의 프로세서는,

VMI(Virtual Machine Introspection)를 이용하여 클라우드 환경의 가상머신 커널 런타임 무결성 검증을 수행하는 것으로,

리눅스 커널이 구동되는 가상머신의 메모리 주소 정보를 분석하는 과정; 및

상기 메모리 주소 정보를 이용하여 가상머신의 커널 변조 무결성을 검증하는 과정

을 처리하고,

상기 적어도 하나의 프로세서는,

리눅스 운영체제에서 구동 중인 타겟 가상머신의 커널 영역 메모리에서 커널 보호 영역의 메모리 주소를 획득하는 것으로,

상기 커널 보호 영역 중 하나인 섹션(section)의 메모리 주소 및 크기를 획득하기 위해 커널 코드인 vmlinux을 추출하고,

상기 커널 보호 영역 중 다른 하나인 IDT(interrupt descriptor table)의 메모리 주소 및 크기를 획득하기 위해 VMCS(Virtual Machine Control Structure)의 IDTR(Interrupt Descriptor Table Register)의 값을 읽어오고,

상기 커널 보호 영역은 커널 코드가 저장된 영역인 <.text section>, 읽기전용 초기화(read-only initialized) 데이터가 존재하는 영역인 <.rodata section>, 예외를 처리할 함수의 테이블이 저장된 영역인 <__ex_table section>, 및 인터럽트를 처리할 함수의 테이블이 저장된 영역인 <IDT>를 포함하고,

상기 적어도 하나의 프로세서는,

가상머신의 부팅이 완료되면 상기 메모리 주소 정보에 따라 물리 메모리 영역의 데이터를 덤프(dump)하여 덤프된 메모리 데이터에 대한 해시 값을 무결성 검증 기준이 되는 해시 값과 비교하여 커널 변조 여부를 확인하는 것으로,

상기 획득한 메모리 주소로 검증된 가상머신의 물리 메모리 데이터를 읽어오는 제1 과정;

상기 읽어온 메모리 데이터에 대한 해시 값을 계산하여 데이터베이스에 저장하는 제2 과정;

상기 타겟 가상머신의 부팅이 완료되면 상기 타겟 가상머신으로부터 읽어온 커널 보호 영역의 메모리 데이터에 대한 해시 값을 계산하는 제3 과정;

상기 데이터베이스에 저장된 해시 값과 상기 타겟 가상머신의 메모리 데이터에 대한 해시 값을 비교하는 제4 과정;

상기 타겟 가상머신의 메모리 데이터에 대한 해시 값이 상기 데이터베이스에 저장된 해시 값과 일치하지 않으면 상기 타겟 가상머신의 커널이 변조된 것으로 판단하여 사전에 정해진 보안 조치를 실행하는 제5 과정; 및

상기 타겟 가상머신의 메모리 데이터에 대한 해시 값이 상기 데이터베이스에 저장된 해시 값과 일치하면 가상머신이 동작하는 런타임 동안 상기 제3 과정과 상기 제4 과정을 반복하는 과정

을 처리하고,

상기 적어도 하나의 프로세서는,

가상머신 데이터를 덤프하는 과정에서 디스크 저장 방식이 아닌 인-메모리(in-memory) 방식을 통해 호스트 메모리 영역에 버퍼를 할당하여 해당 버퍼에 메모리 데이터를 복사하는 것으로,

가상머신의 커널 보호 영역에 대한 주소 정보를 게스트 가상주소의 커널 공간(kernel space)으로 매핑하고,

상기 게스트 가상주소를 게스트 물리주소로 번역하고,

EPT(Extended Page Tables)를 통해 상기 게스트 물리주소를 호스트 물리주소로 번역하고,

상기 호스트 물리주소에서 물리 메모리 데이터를 덤프하는 것

을 특징으로 하는 가상머신 커널 런타임 보호 시스템.

청구항 2

삭제

청구항 3

삭제

청구항 4

삭제

청구항 5

삭제

청구항 6

삭제

청구항 7

삭제

청구항 8

삭제

청구항 9

삭제

청구항 10

삭제

청구항 11

컴퓨터 시스템에서 수행되는 가상머신 커널 런타임 보호 방법에 있어서,

상기 컴퓨터 시스템은 메모리에 포함된 컴퓨터 판독가능한 명령들을 실행하도록 구성된 적어도 하나의 프로세서를 포함하고,

상기 가상머신 커널 런타임 보호 방법은,

VMI(Virtual Machine Introspection)를 이용하여 클라우드 환경의 가상머신 커널 런타임 무결성 검증을 수행하는 것으로,

상기 적어도 하나의 프로세서의 의해, 리눅스 커널이 구동되는 가상머신의 메모리 주소 정보를 분석하는 단계; 및

상기 적어도 하나의 프로세서의 의해, 상기 메모리 주소 정보를 이용하여 가상머신의 커널 변조 무결성을 검증하는 단계

를 포함하고,

상기 분석하는 단계는,

리눅스 운영체제에서 구동 중인 타겟 가상머신의 커널 영역 메모리에서 커널 보호 영역의 메모리 주소를 획득하는 것으로,

상기 커널 보호 영역 중 하나인 섹션(section)의 메모리 주소 및 크기를 획득하기 위해 커널 코드인 vmlinux을 추출하는 단계; 및

상기 커널 보호 영역 중 다른 하나인 IDT(interrupt descriptor table)의 메모리 주소 및 크기를 획득하기 위해 VMCS(Virtual Machine Control Structure)의 IDTR(Interrupt Descriptor Table Register)의 값을 읽어오는 단계

를 포함하고,

상기 커널 보호 영역은 커널 코드가 저장된 영역인 <.text section>, 읽기전용 초기화(read-only initialized) 데이터가 존재하는 영역인 <.rodata section>, 예외를 처리할 함수의 테이블이 저장된 영역인 <__ex_table section>, 및 인터럽트를 처리할 함수의 테이블이 저장된 영역인 <IDT>를 포함하고,

상기 검증하는 단계는,

가상머신의 부팅이 완료되면 상기 메모리 주소 정보에 따라 물리 메모리 영역의 데이터를 덤프(dump)하여 덤프된 메모리 데이터에 대한 해시 값을 무결성 검증 기준이 되는 해시 값과 비교하여 커널 변조 여부를 확인하는 것으로,

상기 획득한 메모리 주소로 검증된 가상머신의 물리 메모리 데이터를 읽어오는 제1 단계;

상기 읽어온 메모리 데이터에 대한 해시 값을 계산하여 데이터베이스에 저장하는 제2 단계;

상기 타겟 가상머신의 부팅이 완료되면 상기 타겟 가상머신으로부터 읽어온 커널 보호 영역의 메모리 데이터에 대한 해시 값을 계산하는 제3 단계;

상기 데이터베이스에 저장된 해시 값과 상기 타겟 가상머신의 메모리 데이터에 대한 해시 값을 비교하는 제4 단계;

상기 타겟 가상머신의 메모리 데이터에 대한 해시 값이 상기 데이터베이스에 저장된 해시 값과 일치하지 않으면 상기 타겟 가상머신의 커널이 변조된 것으로 판단하여 사전에 정해진 보안 조치를 실행하는 제5 단계; 및

상기 타겟 가상머신의 메모리 데이터에 대한 해시 값이 상기 데이터베이스에 저장된 해시 값과 일치하면 가상머신이 동작하는 런타임 동안 상기 제3 단계와 상기 제4 단계를 반복하는 단계

를 포함하고,

상기 검증하는 단계는,

가상머신 데이터를 덤프하는 과정에서 디스크 저장 방식이 아닌 인-메모리(in-memory) 방식을 통해 호스트 메모리 영역에 버퍼를 할당하여 해당 버퍼에 메모리 데이터를 복사하는 것으로,

가상머신의 커널 보호 영역에 대한 주소 정보를 게스트 가상주소의 커널 공간(kernel space)으로 매핑하고,

상기 게스트 가상주소를 게스트 물리주소로 번역하고,

EPT(Extended Page Tables)를 통해 상기 게스트 물리주소를 호스트 물리주소로 번역하고,

상기 호스트 물리주소에서 물리 메모리 데이터를 덤프하는 것

을 특징으로 하는 가상머신 커널 런타임 보호 방법.

청구항 12

삭제

청구항 13

삭제

청구항 14

삭제

청구항 15

삭제

발명의 설명

기 술 분 야

[0001] 아래의 설명은 클라우드 플랫폼 환경에서 가상머신 커널 런타임을 보호하기 위한 기술에 관한 것이다.

배 경 기 술

[0002] 클라우드 컴퓨팅의 기반 기술인 가상화 기술이 발전함에 따라 클라우드 컴퓨팅 기술이 함께 발전하고 있으며, 클라우드 컴퓨팅의 연속성과 효율성과 같은 장점으로 인해 사용자의 편의성이 증가하고 있다.

[0003] 그러나, 보안 관점에서는 가상머신을 커널 영역과 관련된 공격으로부터 보호하기 위한 솔루션을 그대로 클라우드 환경에 적용할 수 없다. 기존의 커널 런타임 보호 소프트웨어를 가상머신마다 내부에 설치되는 에이전트(Agent) 방식으로 구동한다면 클라우드 환경에서는 리소스 효율성이 저하된다. 또한, 일부 커널 보호 솔루션은 커널 공격 프로그램과 같은 권한을 가지며 공격이 성공했을 경우 이 솔루션은 중지되거나 방해를 받기 때문에 정상적인 보안 동작을 수행할 수 없게 된다.

[0004] 따라서, 가상머신 외부에서 가상머신의 커널의 변조를 탐지하는 기술이 필요하다. 호스트에서 가상머신의 런타임 상태를 투명하게 관찰할 수 있는 방법으로 VMI(Virtual Machine Introspection)가 있으며, 가상머신에 할당된 메모리 데이터와 vCPU 레지스터 등 바이너리 형태로 모두 수집할 수 있다.

발명의 내용

해결하려는 과제

[0005] 클라우드 플랫폼 환경에서 가상머신의 커널을 보호하기 위해 가상머신 내에 설치되는 에이전트 방식이 아닌, 호스트에서 보호하는 VMI 기반의 방식을 사용하기 위해서는 여러 가지 요구사항이 따른다. 먼저, 커널의 보호 영역에 대한 무결성 검증이 필요하다. 두 번째로, 가상머신이 사용하는 커널 이미지가 이미 변조되었을 경우에도 커널 변조를 탐지할 수 있어야 한다. 마지막으로, 실시간 VMI로 인해 가상머신은 성능 저하를 겪게 되는데, 이를 최소화해야 한다.

[0006] VMI 방법을 활용한 클라우드 플랫폼 환경에서 구동되는 가상머신의 커널 변조를 정확하고 효율적으로 탐지할 수 있는 기술적 사상을 제공한다.

과제의 해결 수단

[0007] 컴퓨터 시스템으로 구현되는 가상머신 커널 런타임 보호 시스템에 있어서, 메모리에 포함된 컴퓨터 판독가능한 명령들을 실행하도록 구성된 적어도 하나의 프로세서를 포함하고, 상기 적어도 하나의 프로세서는, VMI(Virtual Machine Introspection)를 이용하여 클라우드 환경의 가상머신 커널 런타임 무결성 검증을 수행하는 것으로, 리눅스 커널이 구동되는 가상머신의 메모리 주소 정보를 분석하는 과정; 및 상기 메모리 주소 정보를 이용하여 가상머신의 커널 변조 무결성을 검증하는 과정을 처리하는 가상머신 커널 런타임 보호 시스템을 제공한다.

[0008] 일 측면에 따르면, 상기 적어도 하나의 프로세서는, 리눅스 운영체제에서 구동 중인 타겟 가상머신의 커널 영역 메모리에서 커널 보호 영역의 메모리 주소를 획득할 수 있다.

[0009] 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 기준 해시 값을 이용하여 가상머신의 커널 보호 영역의 메모리 값의 변조 여부를 확인할 수 있다.

[0010] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 가상머신의 부팅이 완료되면 상기 메모리 주소 정보에 따라 물리 메모리 영역의 데이터를 덤프(dump)하고, 덤프된 메모리 데이터에 대한 해시 값을 무결성 검증 기준이 되는 해시 값과 비교하여 커널 변조 여부를 확인할 수 있다.

[0011] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 커널 코드인 vmlinux에 포함된 섹션 정보로 커널 보호 영역에 대한 주소 정보를 분석하고, 분석된 주소를 게스트 가상주소의 커널 공간(kernel space)으로 매핑하고, 상기 게스트 가상주소를 게스트 물리주소로 번역하고, EPT(Extended Page Tables)를 통해 상기 게스트 물리주소를 호스트 물리주소로 번역하고, 상기 호스트 물리주소에서 물리 메모리 데이터를 덤프할 수 있다.

[0012] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 가상머신 데이터를 덤프하는 과정에서 디스크 저장 방식이 아닌 인-메모리(in-memory) 방식을 통해 호스트 메모리 영역에 버퍼를 할당하여 해당 버퍼에 메모리 데이터를 복사할 수 있다.

[0013] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 가상머신의 커널 보호 영역의 메모리 주소를 획득하고, 상기 획득한 메모리 주소로 검증된 가상머신의 물리 메모리 데이터를 읽어오고, 상기 읽어온 메모리 데이터에 대한 해시 값을 계산하여 데이터베이스에 저장하고, 타겟 가상머신으로부터 읽어온 커널 보호 영역의 메모리 데이터에 대한 해시 값을 계산하고, 상기 데이터베이스에 저장된 해시 값과 상기 타겟 가상머신의 메모리 데이터에 대한 해시 값을 비교하여 변조 여부를 결정할 수 있다.

[0014] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 가상머신의 커널 보호 영역 중 하나인 섹션(section)의 메모리 주소 및 크기를 획득하기 위해 커널 코드인 vmlinux을 추출할 수 있다.

[0015] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 가상머신의 커널 보호 영역 중 하나인 IDT(interrupt descriptor table)의 메모리 주소 및 크기를 획득하기 위해 VMCS(Virtual Machine Control Structure)의 IDTR(Interrupt Descriptor Table Register)의 값을 읽어올 수 있다.

[0016] 또 다른 측면에 따르면, 상기 적어도 하나의 프로세서는, 커널 코드가 저장된 영역인 <.text section>, 읽기전용 초기화(read-only initialized) 데이터가 존재하는 영역인 <.rodata section>, 예외를 처리할 함수의 테이블이 저장된 영역인 <__ex_table section>, 및 인터럽트를 처리할 함수의 테이블이 저장된 영역인 <IDT> 중 적어도 하나의 커널 보호 영역을 대상으로 커널 변조 무결성 검증을 수행할 수 있다.

[0017] 또 다른 측면에 따르면, 컴퓨터 시스템에서 수행되는 가상머신 커널 런타임 보호 방법에 있어서, 상기 컴퓨터 시스템은 메모리에 포함된 컴퓨터 판독가능한 명령들을 실행하도록 구성된 적어도 하나의 프로세서를 포함하고, 상기 가상머신 커널 런타임 보호 방법은, VMI(Virtual Machine Introspection)를 이용하여 클라우드 환경의 가상머신 커널 런타임 무결성 검증을 수행하는 것으로, 상기 적어도 하나의 프로세서의 의해, 리눅스 커널이 구동되는 가상머신의 메모리 주소 정보를 분석하는 단계; 및 상기 적어도 하나의 프로세서의 의해, 상기 메모리 주소 정보를 이용하여 가상머신의 커널 변조 무결성을 검증하는 단계를 포함하는 가상머신 커널 런타임 보호 방법을 제공한다.

발명의 효과

[0018] 본 발명의 실시예에 따르면, 가상머신이 동작하기 전에 미리 구성된 해시 데이터베이스로 동작하는 런타임 동안은 물론이고, 가상머신이 부팅한 후에도 커널이 변조되었는지 검증할 수 있다.

[0019] 본 발명의 실시예에 따르면, 가상머신 커널 런타임 보호를 위한 점검 동작을 호스트에서 수행함에 따라 클라우드 환경에서 다수의 가상머신에 대한 커널 변조에 대한 보호 동작을 효율적으로 수행할 수 있다.

도면의 간단한 설명

[0020] 도 1은 본 발명의 일실시예에 있어서 가상머신 커널 런타임 보호 시스템이 적용되는 환경 예시를 도시한 것이다.

도 2는 본 발명의 일실시예에 있어서 클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 시스템의 내부 모듈을 도시한 것이다.

도 3은 본 발명의 일실시예에 있어서 클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 과정의 일례를 도시한 순서도이다.

도 4는 본 발명의 일실시예에 있어서 클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 시스템에서 가상

머신 물리 데이터 덤프하는 과정 예시를 도시한 것이다.

도 5는 본 발명의 일실시예에 있어서 클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 시스템의 무결성 검증 대상이 되는 커널 보호 영역 예시를 도시한 것이다.

도 6은 본 발명의 일실시예에 있어서 클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 시스템을 구현하기 위한 컴퓨터 시스템의 예를 도시한 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0021] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0023] 본 발명의 실시예들은 클라우드 플랫폼 환경에서 가상머신 커널 런타임을 보호하기 위한 기술에 관한 것이다.

[0024] 본 명세서에서 구체적으로 개시되는 것들을 포함하는 실시예들은 호스트의 KVM(Kernel-based Virtual Machine)/QEMU(Quick Emulator)에서 구동되는 리눅스 환경의 가상머신의 커널을 보호하기 위한 것으로써 커널 영역에 악의적인 코드를 삽입하여 변조하거나 커널 데이터 구조를 변경하여 악의적인 코드로 향하도록 하는, 커널 영역을 대상으로 하는 악성코드로부터 보호할 수 있다.

[0025] 본 명세서에서 구체적으로 개시되는 것들을 포함하는 실시예들은 루트킷(rootkit) 탐지, 가상머신 모니터링, 디지털 포렌식(digital forensics) 등에 활용될 수 있다.

[0026] 본 발명의 실시예들에 따른 가상머신 커널 런타임 보호 시스템은 적어도 하나의 컴퓨터 장치에 의해 구현될 수 있으며, 본 발명의 실시예들에 따른 가상머신 커널 런타임 보호 방법은 가상머신 커널 런타임 보호 시스템에 포함되는 적어도 하나의 컴퓨터 장치를 통해 수행될 수 있다. 이때, 컴퓨터 장치에는 본 발명의 일실시예에 따른 컴퓨터 프로그램이 설치 및 구동될 수 있고, 컴퓨터 장치는 구동된 컴퓨터 프로그램의 제어에 따라 본 발명의 실시예들에 따른 가상머신 커널 런타임 보호 방법을 수행할 수 있다. 상술한 컴퓨터 프로그램은 컴퓨터 장치와 결합되어 가상머신 커널 런타임 보호 방법을 컴퓨터에 실행시키기 위해 컴퓨터 판독 가능한 기록매체에 저장될 수 있다.

[0027] 도 1은 본 발명의 일실시예에 있어서 가상머신 커널 런타임 보호 시스템이 적용되는 환경 예시를 도시한 것이다. 도 1은 본 발명의 실시 예에 따른 가상머신 커널 런타임 보호 시스템이 적용되는 환경을 나타내고 있다.

[0028] 도 1을 참조하면, 본 발명의 실시예에 따른 가상머신 커널 런타임 보호 시스템은 클라우드 플랫폼(101)과 클라우드 플랫폼 서비스를 관리하는 관리자(102)가 존재하는 환경에서 적용될 수 있다.

[0029] 본 실시예들은 클라우드 환경에서 커널 공격에 대응하기 위한 가상머신 커널 런타임 무결성 검증을 호스트 수준에서 VMI를 통해 수행할 수 있다.

[0030] 도 2는 본 발명의 일실시예에 있어서 클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 시스템의 내부 모듈을 도시한 것이다.

[0031] 도 2를 참조하면, 본 발명의 실시예에 따른 클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 시스템은 호스트 커널 레벨에서 동작하는 커널 보호 영역 메모리 주소 분석 모듈(201)과 가상머신 메모리 덤프 모듈(202)을 포함할 수 있고, 호스트 사용자 레벨에서 동작하는 해시 데이터베이스(203)와 무결성 검증 모듈(204)을 포함할 수 있다.

[0032] 커널 보호 영역 메모리 주소 분석 모듈(201)은 가상머신의 커널 영역 메모리에서 무결성 검증 대상이 되는 영역의 주소 정보(시작 주소, 크기)를 획득하기 위해 사용된다.

[0033] 가상머신 메모리 덤프 모듈(202)은 메모리 주소 분석 모듈(201)로부터 획득한 주소 정보에 해당하는 물리 메모리 영역의 데이터를 덤프(dump)한다.

[0034] 해시 데이터베이스(203)는 타겟 가상머신의 커널 보호 영역의 메모리 값의 변조 여부를 확인하기 위한 기준 해시 값으로 사용된다.

[0035] 무결성 검증 모듈(204)은 해시 값 비교를 통해 커널 변조 여부를 확인하기 위해 사용된다.

[0036] 도 3은 본 발명의 일실시예에 있어서 클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 과정의 일례를 도시한 순서도이다.

[0037] 본 발명에 따른 가상머신 커널 런타임 보호 시스템은 가상머신의 커널 보호 영역의 메모리 주소를 획득하는 단계, 획득한 메모리 주소로 검증된 가상머신으로부터 정상적인 메모리 데이터를 읽어오는 단계, 읽어온 메모리 데이터에 대한 해시 값을 계산하고 해시 데이터베이스에 저장하는 단계, 타겟 가상머신에 이전에 획득한 커널 보호 영역의 메모리 데이터를 읽어오는 단계, 타겟 가상머신으로부터 읽은 메모리 데이터에 대한 해시 값을 계산하는 단계, 및 해시 데이터베이스에 저장된 해시 값과 타겟 가상머신의 메모리 데이터의 해시 값을 비교하여 변조 여부를 결정하는 단계를 포함할 수 있다.

[0038] 도 3을 참조하면, 먼저, 가상머신 커널 런타임 보호 시스템은 커널 보호 영역 메모리 주소 분석 모듈(201)을 통해 현재 클라우드 플랫폼(리눅스 운영체제)에서 구동 중인 타겟 가상머신의 커널 보호 영역의 가상 메모리 주소 정보(시작주소, 크기)를 분석한다(S301).

[0039] 가상머신 커널 런타임 보호 시스템은 가상머신 메모리 덤프 모듈(202)을 통해 단계(S301)에서 분석한 정보를 바탕으로 검증된 가상머신의 물리 메모리 데이터를 덤프한다(S302).

[0040] 다음으로, 가상머신 커널 런타임 보호 시스템은 덤프된 메모리에 대한 해시 값을 계산하여 무결성 검증의 기준이 되는 해시 데이터베이스를 생성한다(S303).

[0041] 가상머신 커널 런타임 보호 시스템은 타겟 가상머신의 부팅이 완료되면 단계(S301)에서 획득한 주소 정보를 통해 물리 메모리를 덤프한다(S304).

[0042] 가상머신 커널 런타임 보호 시스템은 타겟 가상머신에서 덤프된 데이터를 해시한다(S305).

[0043] 이후, 가상머신 커널 런타임 보호 시스템은 검증된 가상머신으로부터 생성된 해시 데이터베이스와 타겟 가상머신으로부터 도출된 해시 값을 각각 비교한다(S306).

[0044] 가상머신 커널 런타임 보호 시스템은 해시 값이 일치하지 않으면 타겟 가상머신의 커널이 변조되었다고 판단하여 보안 조치(일례로, 가상머신 실행 중지, 심층 분석 등)를 실행한다(S307).

[0045] 가상머신 커널 런타임 보호 시스템은 해시 값이 일치한다면 단계(S304)로 돌아가 가상머신이 동작하는 런타임 동안 상기한 과정의 검사를 반복한다.

[0046] 도 4는 본 발명의 일실시예에 있어서 클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 시스템에서 가상머신 물리 데이터 덤프하는 과정 예시를 도시한 것이다.

[0047] 본 발명에 따른 가상머신 커널 런타임 보호 시스템은 가상머신의 커널 보호 영역인 섹션의 메모리 주소 및 크기를 획득하기 위해 vmlinux 파일을 추출하는 과정을 포함할 수 있다. 여기서, vmlinux 파일은 커널을 컴파일할 때 생성되는 ELF(Excutable Linking Format) 이미지 파일로써 압축되지 않은 커널 코드를 의미한다. vmlinux를 분석하면 심볼 및 자료구조 정보를 얻을 수 있다.

[0048] 도 4를 참조하면, 가상머신 커널 런타임 보호 시스템은 가상머신 커널 버전에 대한 섹션 정보를 분석하기 위해 vmlinux 파일을 추출한다(S401).

[0049] 가상머신 커널 런타임 보호 시스템은 vmlinux에 나타난 섹션(sections) 정보로 보호 영역에 대한 주소 정보(시작주소, 크기)를 계산한다(S402).

[0050] 가상머신 커널 런타임 보호 시스템은 단계(S402)에서 분석한 주소를 게스트 가상주소의 커널 공간(kernel space)으로 매핑한다(S403).

[0051] 다음으로, 가상머신 커널 런타임 보호 시스템은 게스트 가상주소를 게스트 물리주소로 번역한다(S404).

[0052] 가상머신 커널 런타임 보호 시스템은 EPT(Extended Page Tables)로 게스트 물리주소를 호스트 물리주소로 번역한다(S405). 여기서, EPT는 가상화 환경에서 새도우 페이지 테이블(shadow page table)의 복잡성을 줄이고, TLB(Translation lookaside buffer) 플러시(flush) 수를 줄이기 위해 사용되는 하드웨어 지원 페이징을 의미한다.

[0053] 마지막으로, 가상머신 커널 런타임 보호 시스템은 번역된 메모리 영역에서 메모리를 읽어오기 위해 호스트 메모리 영역에 버퍼를 할당하여 복사한다(S406).

[0054] VMI에서는 커널 보호 영역의 주소에 해당하는 게스트 가상머신에 대한 메모리 데이터를 읽기 위해 이전에 메모리 시작 주소와 크기를 매개변수로 하여 QEMU에 가상머신 메모리 읽기를 요청한다. QEMU는 요청된 메모리 영역

을 덤프하기 위해 가상머신의 가상주소를 GVA(Guest Virtual Address), GPA(Guest Physical Address), HPA(Host Physical Address) 순으로 번역한다. 그리고, 물리 메모리에서 해당 영역의 데이터를 덤프한다.

[0055] 상기한 과정에서 가상머신 커널 런타임 보호 시스템은 가상머신 데이터를 덤프하는 과정에서 디스크에 저장하는 것이 아닌 인-메모리(in-memory) 방식을 통해 호스트 영역에 메모리 버퍼를 할당하여 해당 버퍼에 저장함으로써 메모리 덤프에 소요되는 시간, 즉 데이터 읽기 소요시간을 단축할 수 있다.

[0056] 도 5는 본 발명의 일실시예에 있어서 클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 시스템의 무결성 검증 대상이 되는 커널 보호 영역 예시를 도시한 것이다.

[0057] 도 5를 참조하면, 일례로, 무결성 검증 대상이 되는 커널 보호 영역 중 하나인 <.text section>(501)은 커널 코드가 저장된 영역을 의미한다. <.rodata section>(502)은 읽기전용 초기화(read-only initialized) 데이터가 존재하는 영역을 의미한다. <__ex_table section>(503)은 예외가 발생했을 때 예외를 처리할 함수의 테이블이 저장된 영역을 의미한다. <IDT>(504)는 인터럽트가 발생했을 때 인터럽트를 처리할 함수의 테이블이 저장된 영역을 의미한다. IDT(interrupt descriptor table)는 인터럽트 벡터 테이블을 구현하기 위해 X86 아키텍처에서 사용되는 데이터 구조체이다. IDT는 프로세서가 인터럽트와 예외를 처리하기 위해 사용된다. IDT 테이블에는 정의되어 있는 인터럽트들의 번호와 실행 코드를 가리키는 주소들이 저장된다. 가상머신 커널 런타임 보호 시스템은 가상머신의 커널 보호 영역인 IDT의 메모리 주소 및 크기를 획득하기 위해 VMCS(Virtual Machine Control Structure)의 IDTR(Interrupt Descriptor Table Register)의 값을 읽어오는 과정을 포함할 수 있다.

[0058] 상기한 커널 영역들(501 내지 504)은 코드 삽입 및 변조의 일반적인 대상이 되는 영역이다.

[0059] 도 6은 본 발명의 일실시예에 따른 컴퓨터 시스템의 예를 도시한 블록도이다. 본 발명에 따른 클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 시스템은 도 6과 같이 구성된 컴퓨터 시스템(600)에 의해 구현될 수 있다.

[0060] 도 6에 도시된 바와 같이 컴퓨터 시스템(600)은 본 발명의 실시예들에 따른 클라우드 플랫폼 환경에서의 가상머신 커널 런타임 보호 방법을 실행하기 위한 구성요소로서, 메모리(610), 프로세서(620), 통신 인터페이스(630) 그리고 입출력 인터페이스(640)를 포함할 수 있다.

[0061] 메모리(610)는 컴퓨터에서 판독 가능한 기록매체로서, RAM(random access memory), ROM(read only memory) 및 디스크 드라이브와 같은 비소멸성 대용량 기록장치(permanent mass storage device)를 포함할 수 있다. 여기서 ROM과 디스크 드라이브와 같은 비소멸성 대용량 기록장치는 메모리(610)와는 구분되는 별도의 영구 저장 장치로서 컴퓨터 시스템(600)에 포함될 수도 있다. 또한, 메모리(610)에는 운영체제와 적어도 하나의 프로그램 코드가 저장될 수 있다. 이러한 소프트웨어 구성요소들은 메모리(610)와는 별도의 컴퓨터에서 판독 가능한 기록매체로부터 메모리(610)로 로딩될 수 있다. 이러한 별도의 컴퓨터에서 판독 가능한 기록매체는 플로피 드라이브, 디스크, 테이프, DVD/CD-ROM 드라이브, 메모리 카드 등의 컴퓨터에서 판독 가능한 기록매체를 포함할 수 있다. 다른 실시예에서 소프트웨어 구성요소들은 컴퓨터에서 판독 가능한 기록매체가 아닌 통신 인터페이스(630)를 통해 메모리(610)에 로딩될 수도 있다. 예를 들어, 소프트웨어 구성요소들은 네트워크(660)를 통해 수신되는 파일들에 의해 설치되는 컴퓨터 프로그램에 기반하여 컴퓨터 시스템(600)의 메모리(610)에 로딩될 수 있다.

[0062] 프로세서(620)는 기본적인 산술, 로직 및 입출력 연산을 수행함으로써, 컴퓨터 프로그램의 명령을 처리하도록 구성될 수 있다. 명령은 메모리(610) 또는 통신 인터페이스(630)에 의해 프로세서(620)로 제공될 수 있다. 예를 들어 프로세서(620)는 메모리(610)와 같은 기록 장치에 저장된 프로그램 코드에 따라 수신되는 명령을 실행하도록 구성될 수 있다.

[0063] 통신 인터페이스(630)는 네트워크(660)를 통해 컴퓨터 시스템(600)이 다른 장치와 서로 통신하기 위한 기능을 제공할 수 있다. 일례로, 컴퓨터 시스템(600)의 프로세서(620)가 메모리(610)와 같은 기록 장치에 저장된 프로그램 코드에 따라 생성한 요청이나 명령, 데이터, 파일 등이 통신 인터페이스(630)의 제어에 따라 네트워크(660)를 통해 다른 장치들로 전달될 수 있다. 역으로, 다른 장치로부터의 신호나 명령, 데이터, 파일 등이 네트워크(660)를 거쳐 컴퓨터 시스템(600)의 통신 인터페이스(630)를 통해 컴퓨터 시스템(600)으로 수신될 수 있다. 통신 인터페이스(630)를 통해 수신된 신호나 명령, 데이터 등은 프로세서(620)나 메모리(610)로 전달될 수 있고, 파일 등은 컴퓨터 시스템(600)이 더 포함할 수 있는 저장 매체(상술한 영구 저장 장치)로 저장될 수 있다.

[0064] 통신 방식은 제한되지 않으며, 네트워크(660)가 포함할 수 있는 통신망(일례로, 이동통신망, 유선 인터넷, 무선 인터넷, 방송망)을 활용하는 통신 방식뿐만 아니라 기기들 간의 근거리 유선/무선 통신 역시 포함될 수 있다.

예를 들어, 네트워크(660)는, PAN(personal area network), LAN(local area network), CAN(campus area network), MAN(metropolitan area network), WAN(wide area network), BBN(broadband network), 인터넷 등의 네트워크 중 하나 이상의 임의의 네트워크를 포함할 수 있다. 또한, 네트워크(660)는 버스 네트워크, 스타 네트워크, 링 네트워크, 메쉬 네트워크, 스타-버스 네트워크, 트리 또는 계층적(hierarchical) 네트워크 등을 포함하는 네트워크 토폴로지 중 임의의 하나 이상을 포함할 수 있으나, 이에 제한되지 않는다.

[0065] 입출력 인터페이스(640)는 입출력 장치(650)와의 인터페이스를 위한 수단일 수 있다. 예를 들어, 입력 장치는 마이크, 키보드, 카메라 또는 마우스 등의 장치를, 그리고 출력 장치는 디스플레이, 스피커와 같은 장치를 포함할 수 있다. 다른 예로 입출력 인터페이스(640)는 터치스크린과 같이 입력과 출력을 위한 기능이 하나로 통합된 장치와의 인터페이스를 위한 수단일 수도 있다. 입출력 장치(650)는 컴퓨터 시스템(600)과 하나의 장치로 구성될 수도 있다.

[0066] 또한, 다른 실시예들에서 컴퓨터 시스템(600)은 도 6의 구성요소들보다 더 적은 혹은 더 많은 구성요소들을 포함할 수도 있다. 그러나, 대부분의 종래기술적 구성요소들을 명확하게 도시할 필요성은 없다. 예를 들어, 컴퓨터 시스템(600)은 상술한 입출력 장치(650) 중 적어도 일부를 포함하도록 구현되거나 또는 트랜시버(transceiver), 각종 데이터베이스 등과 같은 다른 구성요소들을 더 포함할 수도 있다.

[0067] 이처럼 본 발명의 실시예들에 따르면, 가상머신이 동작하기 전에 미리 구성된 해시 데이터베이스로 동작하는 런타임 동안은 물론이고, 가상머신이 부팅한 후에도 커널이 변조되었는지 검증할 수 있다. 그리고, 본 발명의 실시예들에 따르면, 본 발명의 실시예에 따르면, 가상머신 커널 런타임 보호를 위한 점검 동작을 호스트에서 수행함에 따라 클라우드 환경에서 다수의 가상머신에 대한 커널 변조에 대한 보호 동작을 효율적으로 수행할 수 있다.

[0068] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 어플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0069] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 컴퓨터 저장 매체 또는 장치에 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0070] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 이때, 매체는 컴퓨터로 실행 가능한 프로그램을 계속 저장하거나, 실행 또는 다운로드를 위해 임시 저장하는 것일 수도 있다. 또한, 매체는 단일 또는 수 개의 하드웨어가 결합된 형태의 다양한 기록수단 또는 저장수단일 수 있는데, 어떤 컴퓨터 시스템에 직접 접속되는 매체에 한정되지 않고, 네트워크 상에 분산 존재하는 것일 수도 있다. 매체의 예시로는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM 및 DVD와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical medium), 및 ROM, RAM, 플래시 메모리 등을 포함하여 프로그램 명령어가 저장되도록 구성된 것이 있을 수 있다. 또한, 다른 매체의 예시로, 어플리케이션을 유통하는 앱 스토어나 기타 다양한 소프트웨어를 공급 내지 유통하는 사이트, 서버 등에서 관리하는 기록매체 내지 저장매체도 들 수 있다.

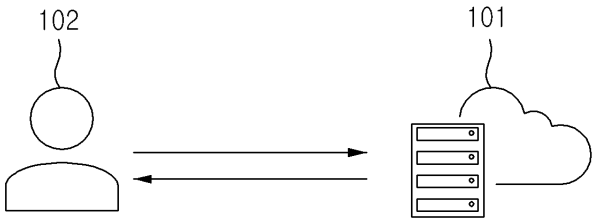
[0071] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가

진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

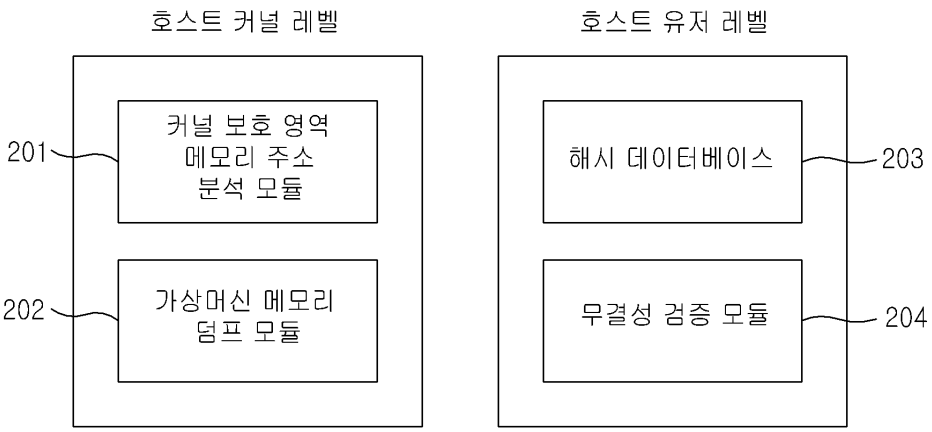
[0072] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

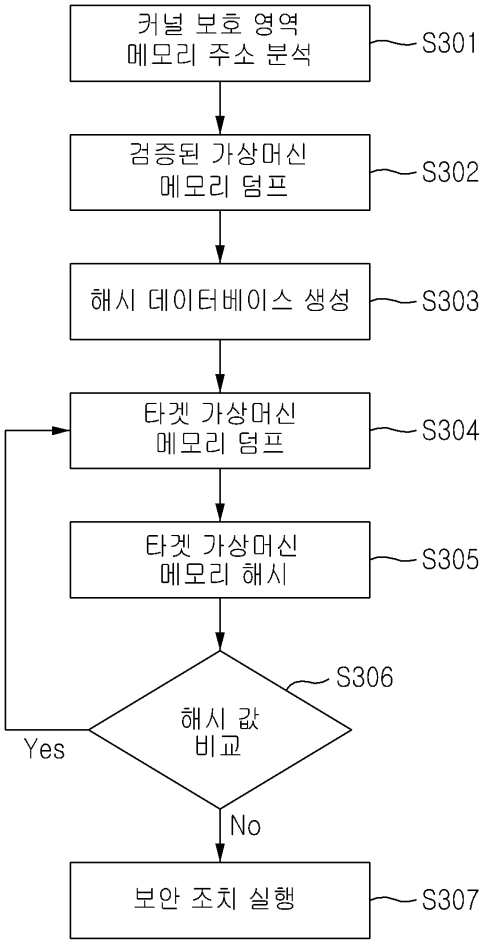
도면1



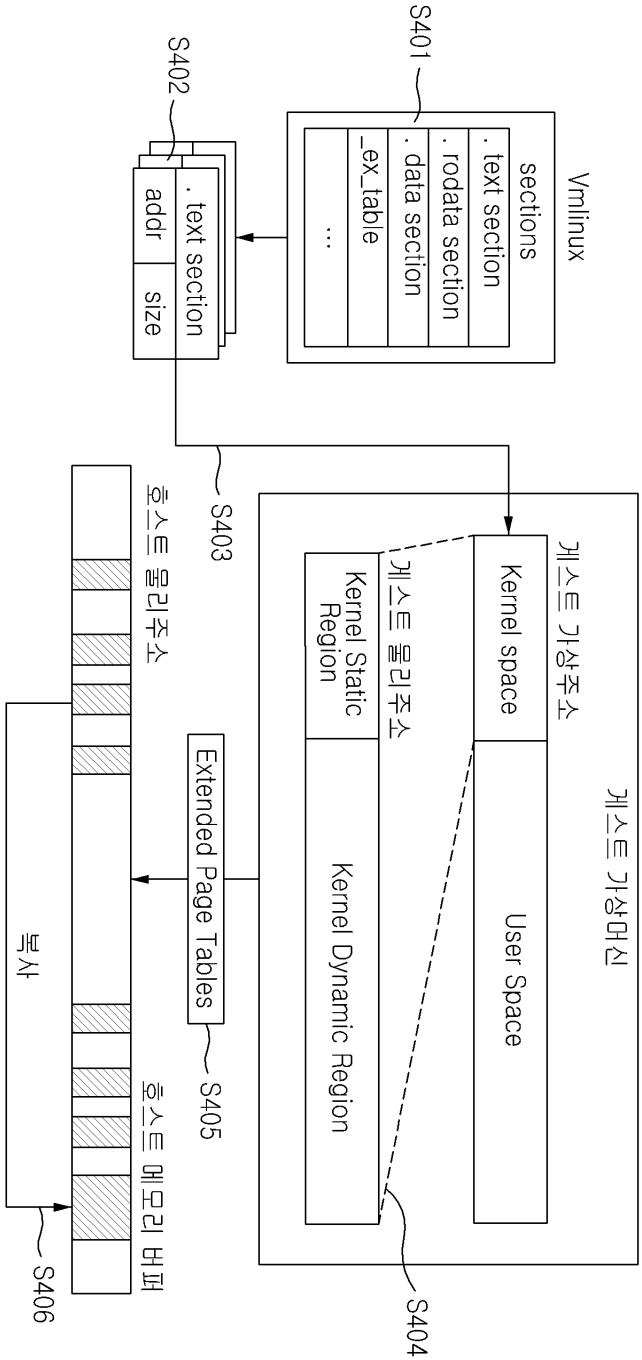
도면2



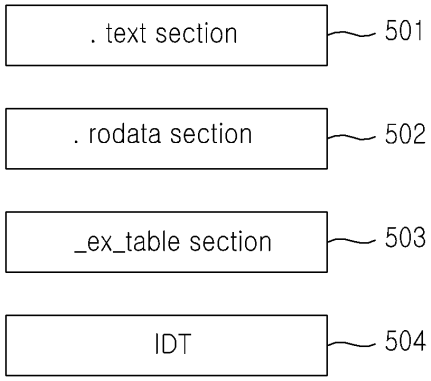
도면3


커널 보호 영역
메모리 주소 분석
S301
검증된 가상머신
메모리 덤프
S302
해시 데이터베이스 생성
S303
타겟 가상머신
메모리 덤프
S304
타겟 가상머신
메모리 해시
S305
해시 값
비교
S306
Yes
No
보안 조치 실행
S307

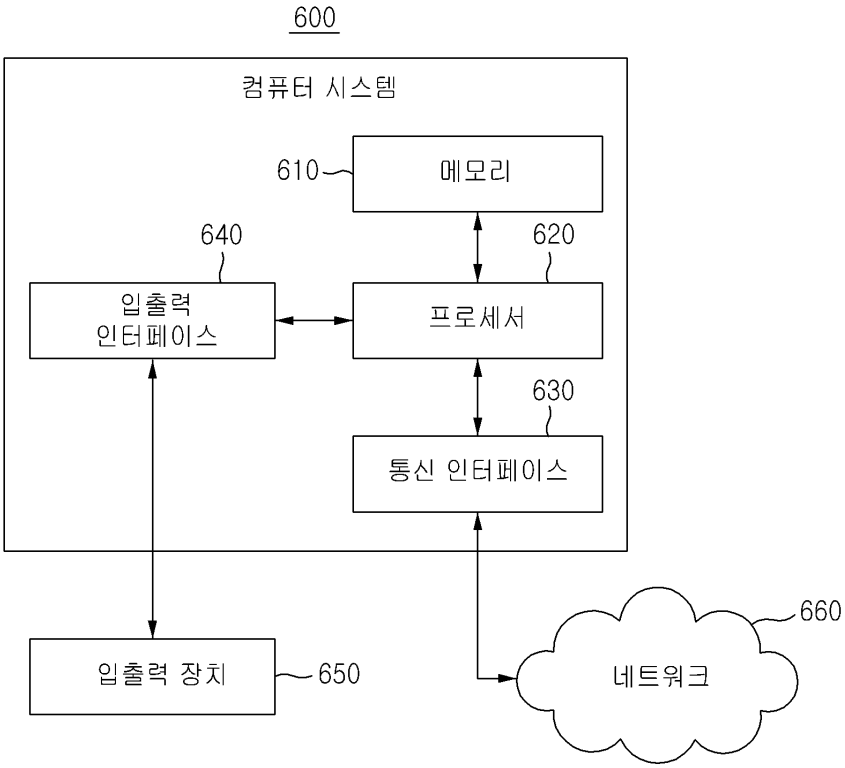
도면4


Vmlinux
sections
. text section
. rodata section
. data section
_ex_table
...
S401
S402
addr
size
. text section
S403
게스트 가상주소
Kernel space
User Space
게스트 가상머신
게스트 물리주소
Kernel Static Region
Kernel Dynamic Region
S404
호스트 물리주소
Extended Page Tables
S405
복사
호스트 메모리 버퍼
S406

도면5


. text section
501
. rodata section
502
_ex_table section
503
IDT
504

도면6


600
컴퓨터 시스템
610
메모리
640
620
입출력
인터페이스
프로세서
630
통신 인터페이스
660
입출력 장치
650
네트워크