(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년05월10일
(11) 등록번호 10-1946982
(24) 등록일자 2019년02월01일

(51) 국제특허분류(Int. Cl.)
G06F 21/53 (2013.01) *G06F 21/56* (2013.01)
G06F 9/455 (2018.01)
(52) CPC특허분류
G06F 21/53 (2013.01)
G06F 21/566 (2013.01)
(21) 출원번호 10-2015-7036979
(22) 출원일자(국제) 2014년07월02일
심사청구일자 2018년07월05일
(85) 번역문제출일자 2015년12월29일
(65) 공개번호 10-2016-0030385
(43) 공개일자 2016년03월17일
(86) 국제출원번호 PCT/RO2014/000019
(87) 국제공개번호 WO 2015/152748
국제공개일자 2015년10월08일
(30) 우선권주장
13/936,058 2013년07월05일 미국(US)
(56) 선행기술조사문헌
JP2006155251 A*
KR1020060106655 A*
KR1020110124208 A*
US20120254993 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
비트데펜더 아이피알 매니지먼트 엘티디
사이프러스 니코시아 1076 12 피시 크레온토스
(72) 발명자
루칵스, 산도르
루마니아 주데츠 클루지, 사트 플로레슈티 (코무나 플로레슈티), 이티. 3, 불레바르둘 체타테아 페테이 비엘. 비
토샤, 라울-바실레
루마니아 주데츠 클루지, 클루지-나포카, 에이피. 30,이티. 4, 스트라다 에드가르 키네 엔알. 32
(뒷면에 계속)
(74) 대리인
권영준

전체 청구항 수 : 총 20 항 심사관 : 정성훈

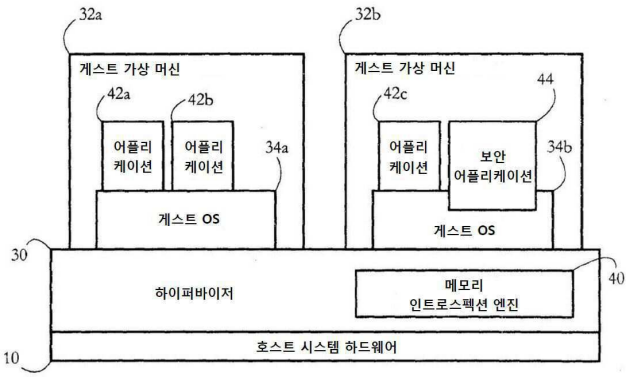
(54) 발명의 명칭 **가상 머신에서 멀웨어 탐지를 위한 프로세스 평가**

(57) 요 약

바이러스와 루트킷과 같은 멀웨어로부터 컴퓨터 시스템을 보호할 수 있는 시스템과 방법이 설명된다. 상기 컴퓨터 시스템 상에서 실행되는 하이퍼바이저에 의해서 노출되는 가상 머신(virtual machine, VM) 내에서 안티-멀웨어 요소가 실행된다. 메모리 인트로스펙션 엔진이 하이퍼바이저의 프로세서 권한 레벨에서 가상 머신 밖에서 실

(뒷면에 계속)

대 표 도 - 도2



행되고 각 프로세스의 메모리 페이지를 쓰기 방지함으로써 가상 머신 내에서 실행되는 프로세스를 보호한다. 각각의 가상 머신의 내부와 외부에서 실행되는 안티 멀웨어 요소를 결합함으로써 본 발명의 일부 실시예들에서는 가상 머신 내부 요소들이 접근권한을 가지는 충분한 행동 데이터를 사용할 수 있고, 한편으로 각각의 가상 머신의 외측으로부터 그와 같은 요소들의 완전성을 보호할 수 있다.

(52) CPC특허분류
G06F 9/45558 (2013.01)
G06F 2009/45587 (2013.01)
G06F 2221/2141 (2013.01)
G06F 2221/2149 (2013.01)

(72) 발명자
보카, 파울-다니엘
루마니아 주데츠 클루지, 클루지-나포카, 에이피. 22, 에스씨. 2, 스트라다 아그리쿨토릴로르 엔알. 20
하지마샨, 게오르게-플로린
루마니아 주데츠 알바, 코무나 룬카 무레술루이, 사트 룬카무레술루이 엔알. 351
루차스, 안드레이-블라드
루마니아 주데츠 사투 마레, 사투 마레, 불레바르 둘 클로슈카 엔알. 111

명 세 서

청구범위

청구항 1

가상 머신을 노출하도록 구성된 하이퍼바이저,

상기 가상 머신 내에서 실행되는 프로세스 평가자(process evaluator),

상기 가상 머신 밖에서 실행되는 메모리 인트로스펙션 엔진(memory introspection engine), 및

프로세스 스코어링 모듈(process scoring module)을 실행하도록 구성된 적어도 하나의 프로세서를 포함하고,

상기 프로세스 평가자는, 상기 가상 머신 내에서 실행되는 피평가 프로세스(evaluated process)가 작업(action)을 수행하는지 결정하고; 응답으로서, 피평가 프로세스가 상기 작업을 수행할 때 상기 피평가 프로세스에 대하여 결정된 제1 프로세스 평가 표시자를 상기 프로세스 스코어링 모듈에 전송하도록 구성되고,

상기 메모리 인트로스펙션 엔진은, 운영 시스템 함수로의 호출을 중간차단해서 상기 가상 머신 내에서 실행되는 피보호 프로세스(protected process)의 개시를 탐지하되, 상기 운영 시스템 함수는 상기 가상 머신 내에서 실행되는 프로세스 리스트를 상기 피보호 프로세스에 추가하고; 상기 개시를 탐지한 응답으로서, 상기 피평가 프로세스가 상기 피보호 프로세스의 메모리 페이지를 수정하려고 시도하는지 결정하고, 응답으로서, 상기 피평가 프로세스가 상기 메모리 페이지를 수정하려고 시도할 때, 상기 피평가 프로세스에 대해서 결정된 제2 프로세스 평가 표시자를 상기 프로세스 스코어링 모듈에 전송하도록 구성되고,

상기 프로세스 스코어링 모듈은, 제1 및 제2 프로세스 평가 표시자들을 수신하고, 응답으로서, 상기 제1 및 제2 프로세스 평가 표시자들에 따라서 상기 피평가 프로세스가 악성인지 여부를 결정하도록 구성되고,

상기 피평가 프로세스가 악성인지를 결정하는 것은, 제1 스코어와 제2 스코어의 가중 합계에 따라서 총합 스코어를 결정하는 것을 포함하되, 상기 제1 스코어와 제2 스코어는 상기 제1 및 제2 프로세스 평가 표시자에 따라서 각각 결정되고,

상기 프로세스 스코어링 모듈은, 제1 가중치(weight)와 제2 가중치를 보안 서버로부터 수신하도록 추가적으로 구성되되, 상기 제1 가중치는 상기 가중 합계에서 상기 제1 스코어에 곱하고, 상기 제2 가중치는 상기 가중 합계에서 상기 제2 스코어에 곱하며,

상기 보안 서버는 호스트 시스템을 포함하는 다수의 컴퓨터 시스템들과 안티-멀웨어 트랜잭션을 수행하도록 구성되는 것을 특징으로 하는 호스트 시스템.

청구항 2

제1항에 있어서,

상기 메모리 인트로스펙션 엔진은,

상기 피보호 프로세스의 개시를 탐지한 것의 응답으로, 상기 가상 머신 내에서 실행되는 보안 어플리케이션에 상기 피보호 프로세스의 표시자를 전송하고; 응답으로, 상기 보안 어플리케이션으로부터 상기 메모리 페이지의 표시자를 수신하도록 추가적으로 구성되는 것을 특징으로 하는 호스트 시스템.

청구항 3

제1항에 있어서,

상기 프로세스 평가자는 사용자 레벨의 프로세서 권한에서 실행되는 사용자 레벨 프로세스 평가자를 포함하되, 상기 사용자 레벨 프로세스 평가자는 상기 피평가 프로세스가 상기 작업을 수행하는지를 결정하도록 구성되는 것을 특징으로 하는 호스트 시스템.

청구항 4

제1항에 있어서,

상기 프로세스 평가자는 커널 레벨의 프로세서 권한에서 실행되는 커널 레벨 프로세스 평가자를 포함하고, 상기 커널 레벨 프로세스 평가자는 상기 피평가 프로세스가 상기 작업을 수행하는지를 결정하도록 구성되는 것을 특징으로 하는 호스트 시스템.

청구항 5

제1항에 있어서,

상기 프로세스 평가자는 상기 피평가 프로세스에 의해서 수행된 시스템 호출을 중단차단하도록 구성된 시스템 호출 평가자를 포함하는 것을 특징으로 하는 호스트 시스템.

청구항 6

제1항에 있어서,

상기 프로세스 스코어링 모듈은 상기 가상 머신 내에서 실행되는 것을 특징으로 하는 호스트 시스템.

청구항 7

제1항에 있어서,

상기 프로세스 스코어링 모듈은 상기 가상 머신 밖에서 실행되는 것을 특징으로 하는 호스트 시스템.

청구항 8

삭제

청구항 9

삭제

청구항 10

제1항에 있어서,

상기 피보호 프로세스는 상기 프로세스 스코어링 모듈을 포함하는 것을 특징으로 하는 호스트 시스템.

청구항 11

제1항에 있어서,

상기 피보호 프로세스는 상기 프로세스 평가자를 포함하는 보안 어플리케이션의 일부를 구성하는 것을 특징으로 하는 호스트 시스템.

청구항 12

적어도 하나의 프로세서를 포함하는 호스트 시스템에서 실행될 때, 상기 호스트 시스템이:

가상 머신을 노출하도록 구성된 하이퍼바이저,

상기 가상 머신 내에서 실행되는 프로세스 평가자(process evaluator),

상기 가상 머신 밖에서 실행되는 메모리 인트로스펙션 엔진(memory introspection engine), 및

프로세스 스코어링 모듈(process scoring module)을

형성하도록 야기하는 명령어(instruction)들을 암호화(encoding)하는 비-일시적 컴퓨터 판독가능 매체로서,

상기 프로세스 평가자는, 상기 가상 머신 내에서 실행되는 피평가 프로세스(evaluated process)가 작업(action)을 수행하는지 결정하고; 응답으로서, 피평가 프로세스가 상기 작업을 수행할 때 상기 피평가 프로세스에 대하여 결정된 제1 프로세스 평가 표시자를 상기 프로세스 스코어링 모듈에 전송하도록 구성되고,

상기 메모리 인트로스펙션 엔진은, 운영 시스템 함수로의 호출을 중간차단해서 상기 가상 머신 내에서 실행되는

피보호 프로세스(protected process)의 개시를 탐지하되, 이 때 상기 운영 시스템 함수는 상기 가상 머신 내에서 실행되고 또한 상기 운영 시스템 함수는 상기 가상 머신 내에서 실행되는 프로세스 리스트를 상기 피보호 프로세스에 추가하도록 구성되고; 상기 개시를 탐지한 응답으로서, 상기 피평가 프로세스가 상기 피보호 프로세스의 메모리 페이지를 수정하려고 시도하는지 결정하고, 응답으로서, 상기 피평가 프로세스가 상기 메모리 페이지를 수정하려고 시도할 때, 상기 피평가 프로세스에 대해서 결정된 제2 프로세스 평가 표시자를 상기 프로세스 스코어링 모듈에 전송하도록 구성되고,

상기 프로세스 스코어링 모듈은, 제1 및 제2 프로세스 평가 표시자들을 수신하고, 응답으로서, 상기 제1 및 제2 프로세스 평가 표시자들에 따라서 상기 피평가 프로세스가 악성인지 여부를 결정하도록 구성되고,

상기 피평가 프로세스가 악성인지를 결정하는 것은, 제1 스코어와 제2 스코어의 가중 합계에 따라서 총합 스코어를 결정하는 것을 포함하되, 상기 제1 스코어와 제2 스코어는 상기 제1 및 제2 프로세스 평가 표시자에 따라서 각각 결정되고,

상기 프로세스 스코어링 모듈은, 제1 가중치(weight)와 제2 가중치를 보안 서버로부터 수신하도록 추가적으로 구성되되, 상기 제1 가중치는 상기 가중 합계에서 상기 제1 스코어에 곱하고, 상기 제2 가중치는 상기 가중 합계에서 상기 제2 스코어에 곱하며,

상기 보안 서버는 상기 호스트 시스템을 포함하는 다수의 컴퓨터 시스템들과 안티-멀웨어 트랜잭션을 수행하도록 구성되는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 매체.

청구항 13

제12항에 있어서,

상기 메모리 인트로스펙션 엔진은,

상기 피보호 프로세스의 개시를 탐지한 것의 응답으로, 상기 가상 머신 내에서 실행되는 보안 어플리케이션에 상기 피보호 프로세스의 표시자를 전송하고; 응답으로, 상기 보안 어플리케이션으로부터 상기 메모리 페이지의 표시자를 수신하도록 추가적으로 구성되는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 매체.

청구항 14

제12항에 있어서,

상기 프로세스 평가자는 사용자 레벨의 프로세서 권한에서 실행되는 사용자 레벨 프로세스 평가자를 포함하되, 상기 사용자 레벨 프로세스 평가자는 상기 피평가 프로세스가 상기 작업을 수행하는지를 결정하도록 구성되는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 매체.

청구항 15

제12항에 있어서,

상기 프로세스 평가자는 커널 레벨의 프로세서 권한에서 실행되는 커널 레벨 프로세스 평가자를 포함하고, 상기 커널 레벨 프로세스 평가자는 상기 피평가 프로세스가 상기 작업을 수행하는지를 결정하도록 구성되는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 매체.

청구항 16

제12항에 있어서,

상기 프로세스 평가자는 상기 피평가 프로세스에 의해서 수행된 시스템 호출을 중단차단하도록 구성된 시스템 호출 평가자를 포함하는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 매체.

청구항 17

제12항에 있어서,

상기 프로세스 스코어링 모듈은 상기 가상 머신 내에서 실행되는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 매체.

청구항 18

제12항에 있어서,

상기 프로세스 스코어링 모듈은 상기 가상 머신 밖에서 실행되는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 매체.

청구항 19

삭제

청구항 20

삭제

청구항 21

제12항에 있어서,

상기 피보호 프로세스는 상기 프로세스 스코어링 모듈을 포함하는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 매체.

청구항 22

제12항에 있어서,

상기 피보호 프로세스는 상기 프로세스 평가자를 실행하도록 구성된 보안 어플리케이션의 일부를 구성하는 것을 특징으로 하는 비-일시적 컴퓨터 판독가능 매체.

청구항 23

호스트 시스템을 포함하는 다수의 컴퓨터 시스템들과 안티-멀웨어 트랜잭션을 수행하도록 구성되는 보안 서버로부터 제1 가중치(weight)와 제2 가중치를 수신하기 위하여 상기 호스트 시스템의 적어도 하나의 프로세서를 채용하는 단계,

상기 호스트 시스템에서 실행되는 하이퍼바이저에 의해서 노출되는 가상 머신 내에서 실행되는 피평가 프로세스에 대해서 결정된 제1 프로세스 평가 표시자를 수신하기 위하여 상기 호스트 시스템의 상기 적어도 하나의 프로세서를 채용하는 단계,

상기 피평가 프로세스에 대해서 결정된 제2 프로세스 평가 표시자를 수신하기 위하여 상기 적어도 하나의 프로세서를 채용하는 단계, 및

상기 제1 및 제2 프로세스 평가 표시자를 수신한 것의 응답으로, 상기 제1 및 제2 프로세스 평가 표시자들에 따라서 상기 피평가 프로세스가 악성인지 결정하기 위하여 상기 적어도 하나의 프로세서를 채용하는 단계를 포함하되,

상기 제1 프로세스 평가 표시자를 결정하는 것은 상기 피평가 프로세스가 제1 작업을 수행하는지를 결정하기 위하여 상기 가상 머신 내에서 실행되는 프로세스 평가자를 채용하는 것을 포함하고,

상기 제2 프로세스 평가 표시자를 결정하는 것은 상기 피평가 프로세스가 제2 작업을 수행하는지를 결정하기 위하여 상기 가상 머신 밖에서 실행되는 메모리 인트로스펙션 엔진을 채용하는 것을 포함하고,

또한, 상기 피평가 프로세스가 악성인지를 결정하는 것은, 제1 스코어와 제2 스코어의 가중 합계에 따라서 총합 스코어를 결정하는 것을 포함하되, 상기 제1 스코어와 제2 스코어는 상기 제1 및 제2 프로세스 평가 표시자에 따라서 각각 결정되고, 상기 제1 가중치는 상기 가중 합계에서 상기 제1 스코어에 곱하고, 상기 제2 가중치는 상기 가중 합계에서 상기 제2 스코어에 곱하는 것을 특징으로 하는 방법.

청구항 24

호스트 시스템을 포함하는 다수의 컴퓨터 시스템들과 안티-멀웨어 트랜잭션을 수행하도록 구성되는 보안 서버로부터 제1 가중치(weight)와 제2 가중치를 수신하기 위하여 상기 호스트 시스템의 적어도 하나의 프로세서를 채

용하는 단계

상기 호스트 시스템에서 실행되는 하이퍼바이저에 의해서 노출되는 가상 머신 밖에서 실행되는 메모리 인트로스펙션 엔진을 실행하기 위하여 상기 호스트 시스템의 상기 적어도 하나의 프로세서를 채용하는 단계(여기서 상기 메모리 인트로스펙션 엔진을 실행하는 것은 상기 가상 머신 내에서 실행되는 프로세스의 개시를 탐지하는 것을 포함),

상기 메모리 인트로스펙션 엔진이 상기 프로세스의 개시를 탐지하는 것에 응답하여, 상기 프로세스의 제1 및 제2 프로세스 평가 표시자를 결정하기 위하여 상기 적어도 하나의 프로세서를 채용하는 단계, 및

상기 제1 및 제2 평가 표시자를 결정하는 것에 응답하여, 상기 제1 및 제2 프로세스 평가 표시자에 따라서 상기 프로세스가 악성인지를 결정하기 위하여 상기 적어도 하나의 프로세서를 채용하는 단계(여기서, 상기 프로세스가 악성인지를 결정하는 것은, 제1 스코어와 제2 스코어의 가중 합계에 따라서 총합 스코어를 결정하는 것을 포함하되, 상기 제1 스코어와 제2 스코어는 상기 제1 및 제2 프로세스 평가 표시자에 따라서 각각 결정되고, 상기 제1 가중치는 상기 가중 합계에서 상기 제1 스코어에 곱하고, 상기 제2 가중치는 상기 가중 합계에서 상기 제2 스코어에 곱하도록 구성됨)를 포함하는 것을 특징으로 하는 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 멀웨어(malware)로부터 컴퓨터 시스템을 보호하기 위한 시스템 및 방법, 특히 하드웨어 가상화 기술을 사용하는 안티-멀웨어 시스템에 관한 것이다.

배 경 기 술

[0002] 멀웨어로도 알려진 악성 소프트웨어는 세계적으로 상당수의 컴퓨터 시스템에 영향을 주고 있다. 멀웨어는 컴퓨터 바이러스, 웜, 및 루트킷(rootkit)과 같은 많은 형태로, 수백만의 컴퓨터 사용자에게 심각한 위협이 되고 있으며, 무엇보다도 데이터 및 민감한 정보의 손실, 신원 도용, 및 생산성 손실에 있어 이들을 취약하게 하고 있다.

[0003] 하드웨어 가상화 기술은, 많은 방면에서 물리적 컴퓨터 시스템으로 거동하는 통상적으로 가상 머신(virtual machine)으로 알려진 시뮬레이티드 컴퓨터 환경의 형성을 허용한다. 서버 통합 및 서비스로서의 인프라스트럭쳐(IAAS)와 같은 통상적인 어플리케이션에서, 몇몇 가상 머신은 동일한 물리적 머신상에서 동시에 실행될 수 있어서, 하드웨어 자원을 그들 사이에서 공유함으로써, 투자 및 운영 비용을 줄일 수 있다. 각각의 가상 머신은 다른 가상 머신들과 별개로 고유의 운영 시스템 및/또는 소프트웨어 어플리케이션을 구동할 수 있다. 멀웨어의 지속적인 확산으로 인해, 이러한 환경에서 작동하는 각각의 가상 머신은 잠재적으로 멀웨어 보호를 필요로 한다.

발명의 내용

해결하려는 과제

[0004] 본 기술분야에서 통상적으로 사용되는 가상화 솔루션은 가상 머신 모니터로도 알려진 하이퍼바이저(hypervisor)를 포함하며, 이는 가상 머신의 운영 시스템(OS)과 컴퓨팅 하드웨어 사이에서 작동하는 소프트웨어의 레이어로 구성되고, 개별 OS보다 더 많은 프로세서 권한(processor privilege)을 갖는다. 루트킷과 같은 일부 멀웨어가 운영 체제 권한 레벨에서 작동하기 때문에 하이퍼바이저의 권한 레벨(privilege level)에서 실행되는 안티-멀웨어 솔루션을 개발할 필요가 있다.

과제의 해결 수단

[0005] 본 발명의 일 태양에 따르면, 호스트 시스템은 다음의 것을 실행하도록 구성된 적어도 하나의 프로세서를 포함한다: 가상 머신을 노출하도록 구성된 하이퍼바이저; 상기 가상 머신 내에서 실행되는 프로세스 평가자(process evaluator); 상기 가상 머신 밖에서 실행되는 메모리 인트로스펙션 엔진(memory introspection engine); 및 프로세스 스코어링 모듈(process-scoring module). 상기 프로세스 평가자는 상기 가상 머신 내에서 실행되는 피평가 프로세스가 작업(action)을 수행하는지 여부를 판단하고, 피평가 프로세스가 작업을 수행할 때에는 응답으로 제1 프로세스 평가 표시자를 상기 프로세스 스코어링 모듈에 전송한다(이 때 상기 제1 프로세스 평가 표시자는 상기 피평가 프로세스에 대하여 결정된다). 상기 메모리 인트로스펙션 엔진은 운영 시스템 함수(operating

system function)로의 호출을 가로채어서 상기 가상 머신 내에서 실행되는 피보호 프로세스(protected process)의 개시(launch)를 탐지하도록 구성되는데, 상기 운영 시스템 함수는 상기 가상 머신 내에서 실행되는 프로세스들의 리스트에 상기 피보호 프로세스를 추가하도록 구성되고, 상기 개시를 탐지한 것에 대한 응답으로, 피평가 프로세스가 상기 피보호 프로세스의 메모리 페이지를 수정하려고 시도하는지 여부를 판단하고, 또한 피평가 프로세스가 상기 메모리 페이지를 수정하려고 시도하는 경우에, 응답으로, 상기 프로세스 스코어링 모듈에 제2 프로세스 평가 표시자를 전송한다(이 때, 상기 제2 프로세스 평가 표시자는 상기 피평가 프로세스에 대하여 결정된다). 상기 프로세스 스코어링 모듈은 제1 및 제2 프로세스 평가 표시자들을 수신하고, 응답으로, 피평가 프로세스가 상기 제1 및 제2 프로세스 평가 표시자에 따라서 악성인지 여부를 결정한다.

[0006] 본 발명의 다른 태양에 따르면, 비-일시적 컴퓨터 판독가능 매체는, 적어도 하나의 프로세서를 포함하는 호스트 시스템에서 실행될 때 호스트 시스템이 다음의 것을 형성하도록 하는 지시들을 암호화(encoding)한다: 가상 머신을 노출하도록 구성된 하이퍼바이저; 상기 가상 머신 내에서 실행되는 프로세스 평가자(process evaluator); 상기 가상 머신 밖에서 실행되는 메모리 인트로스펙션 엔진(memory introspection engine); 및 프로세스 스코어링 모듈(process-scoring module). 상기 프로세스 평가자는 상기 가상 머신 내에서 실행되는 피평가 프로세스가 작업(action)을 수행하는지 여부를 판단하고, 피평가 프로세스가 작업을 수행할 때에는 응답으로 제1 프로세스 평가 표시자를 상기 프로세스 스코어링 모듈에 전송한다(이 때 상기 제1 프로세스 평가 표시자는 상기 피평가 프로세스에 대하여 결정된다). 상기 메모리 인트로스펙션 엔진은 운영 시스템 함수(operating system function)로의 호출을 가로채어서 상기 가상 머신 내에서 실행되는 피보호 프로세스의 개시(launch)를 탐지하도록 구성되는데, 상기 운영 시스템 함수는 상기 가상 머신 내에서 실행되는 프로세스들의 리스트에 상기 피보호 프로세스를 추가하도록 구성되고, 상기 개시를 탐지한 것에 대한 응답으로, 피평가 프로세스가 상기 피보호 프로세스의 메모리 페이지를 수정하려고 시도하는지 여부를 판단하고, 또한 피평가 프로세스가 상기 메모리 페이지를 수정하려고 시도하는 경우에, 응답으로, 상기 프로세스 스코어링 모듈에 제2 프로세스 평가 표시자를 전송한다(이 때, 상기 제2 프로세스 평가 표시자는 상기 피평가 프로세스에 대하여 결정된다). 상기 프로세스 스코어링 모듈은 제1 및 제2 프로세스 평가 표시자들을 수신하고, 응답으로, 피평가 프로세스가 상기 제1 및 제2 프로세스 평가 표시자에 따라서 악성인지 여부를 결정한다.

[0007] 또 다른 태양에 따르면, 본 발명의 방법은 피평가 프로세스에 대해서 결정된 제1 프로세스 평가 표시자를 수신하기 위하여 호스트 시스템의 적어도 하나의 프로세서를 채용하는 것을 포함하고, 이 때 상기 피평가 프로세스는 상기 호스트 시스템에서 실행되는 하이퍼바이저에 의해서 노출되는 가상 머신 내에서 실행된다. 상기 방법은 상기 피평가 프로세스에 대해서 결정된 제2 프로세스 평가 표시자를 수신하기 위하여 상기 적어도 하나의 프로세서를 채용하는 것을 또한 포함하고, 상기 제1 및 제2 프로세스 평가 표시자를 수신하는 것의 응답으로, 상기 피평가 프로세스가 상기 제1 및 제2 프로세스 평가 표시자에 따라서 악성인지 여부를 결정하기 위하여 상기 적어도 하나의 프로세서를 채용하는 것을 포함한다. 상기 제1 프로세스 평가 표시자를 결정하는 것은 상기 피평가 프로세스가 제1 작업을 수행하는 지 여부를 판단하기 위하여 상기 가상 머신 내에서 실행되는 프로세스 평가자를 채용하는 것을 포함한다. 상기 제2 프로세스 평가 표시자를 결정하는 것은 상기 피평가 프로세스가 제2 작업을 수행하는 지 여부를 판단하기 위하여 상기 가상 머신 밖에서 실행되는 메모리 인트로스펙션 엔진을 채용하는 것을 포함한다.

[0008] 또 다른 태양에 따르면, 본 발명의 방법은 메모리 인트로스펙션 엔진을 실행시키기 위하여 호스트 시스템의 적어도 하나의 프로세서를 채용하는 것을 포함하고, 이때 상기 메모리 인트로스펙션 엔진은 상기 호스트 시스템에서 실행되는 하이퍼바이저에 의해서 노출되는 가상 머신 밖에서 실행되고, 이 때 상기 메모리 인트로스펙션 엔진을 실행하는 것은 상기 가상 머신 내에서 실행되는 프로세스의 개시를 탐지하는 것을 포함한다. 상기 방법은, 상기 메모리 인트로스펙션 엔진이 상기 프로세스의 개시를 탐지하는 것의 응답으로, 상기 프로세스의 제1 및 제2 프로세스 평가 표시자를 결정하기 위하여 상기 적어도 하나의 프로세서를 채용하는 것을 또한 포함한다. 상기 방법은, 상기 제1 및 제2 평가 표시자를 결정하는 것의 응답으로, 상기 프로세스가 상기 제1 및 제2 프로세스 평가 표시자에 따라서 악성인지 여부를 결정하기 위하여 상기 적어도 하나의 프로세서를 채용하는 것을 또한 포함한다.

도면의 간단한 설명

[0009] 본 발명의 전술한 태양들 및 장점은 후술하는 상세한 설명 및 도면을 참조로 이해하면 더욱 잘 이해될 것이다.

도 1은 본 발명의 일부 실시예들에 따른, 멀웨어로부터 보호되는 호스트 컴퓨터 시스템의 예시적인 하드웨어 구성을 도시한다.

도 2는 본 발명의 일부 실시예들에 따른, 가상 머신을 보호하기 위한 메모리 인트로스펙션 엔진과 함께 작동하는 보안 어플리케이션과 도 1의 호스트 시스템상에서 실행되는 하이퍼바이저에 의해 노출된 가상 머신의 예시적인 세트를 도시한다.

도 3은 본 발명의 일부 실시예들에 따른 안티-멀웨어 객체들의 세트를 포함하는 호스트 시스템 상에서 다양한 프로세서 권한 레벨들에서 실행되는 소프트웨어 객체(software object)들의 예시적 계층을 보여준다.

도 4는 본 발명의 일부 실시예들에 따른, 다수의 프로세스 평가자에 의하여 프로세스에 대하여 결정된 다수의 프로세스 평가 표시자를 수신하는 예시적인 프로세스 스코어링 모듈을 도시한다.

도 5는 본 발명의 일부 실시예들에 따른, 도 4의 프로세스 스코어링 모듈에 의해서 수행되는 단계들의 예시적인 시퀀스(순서)를 보여준다.

도 6은 본 발명의 일부 실시예들에 따른, 도 2의 시스템 구성 내의 메모리 어드레스들의 예시적인 맵핑을 도시한다.

도 7은 Windows® 환경에서 일련의 프로세스들의 예시적인 실행 흐름도를 도시한다. 실선 화살표는 안티-멀웨어 시스템이 없을 때의 예시적인 실행 흐름도를 가리킨다. 파선 화살표는 본 발명의 일부 실시예에 따라 작동하는 다수의 프로세스 평가자에 의해서 도입되는 상기 실행 흐름도의 변경사항을 가리킨다.

도 8은 본 발명의 일부 실시예에 따른 도 2 내지 도 3의 메모리 인트로스펙션 엔진에 의해서 수행되는 단계들의 예시적 시퀀스를 도시한다.

도 9는 본 발명의 일부 실시예에 따른, 메모리 페이지를 보호하기 위하여 메모리 인트로스펙션 엔진에 의해서 수행되는 단계들의 예시적 시퀀스를 도시한다.

도 10은 컴퓨터 네트워크를 통해서 보안 서버에 연결된 다수의 호스트 시스템을 포함하는 예시적 구성을 도시한다.

도 11은 본 발명의 일부 실시예에 따른, 호스트 시스템과 보안 서버 사이의 예시적 안티-멀웨어 트랜잭션(transaction)을 도시한다.

발명을 실시하기 위한 구체적인 내용

[0010] 이하의 설명에서, 구조들 사이에서 언급된 모든 연결들은 직접적인 동작 연결들 또는 매개 구조들을 통한 간접적인 동작 연결들일 수 있는 것으로 이해된다. 구성 요소들의 세트는 하나 이상의 구성 요소를 포함한다. 구성 요소의 임의의 열거는 적어도 하나의 구성 요소를 언급하는 것으로 이해된다. 복수의 구성 요소는 적어도 2개의 구성 요소를 포함한다. 달리 요구되지 않는다면, 기술된 어떠한 방법 단계들도 설명된 특정 순서로 반드시 실행될 필요는 없다. 제2 구성 요소로부터 유도되는 제1 구성 요소(예컨대, 데이터)는 제2 구성 요소와 동일한 제1 구성 요소는 물론, 제2 구성 요소 그리고 선택적으로는 다른 데이터를 처리하는 것에 의해 생성된 제1 구성 요소를 포함한다. 파라미터에 따라 결정 또는 판정하는 것은 파라미터에 따라 그리고 선택적으로는 다른 데이터에 따라 결정 또는 판정하는 것을 포함한다. 달리 구체화되지 않는다면, 일부 수량/데이터의 표시자는 수량/데이터 그 자체, 또는 수량/데이터 그 자체와 상이한 표시자일 수 있다. 달리 구체화되지 않는다면, 프로세스는 컴퓨터 프로그램의 인스턴스(instance)를 나타내고, 여기서 컴퓨터 프로그램은 컴퓨터 시스템이 특정의 과업을 수행하도록 결정하는 일련의 지시들이다. 달리 구체화되지 않는다면, 페이지는 컴퓨터 시스템의 물리적 메모리로 개별적으로 맵핑된 가상화된 물리적 메모리의 최소 단위를 나타낸다. 컴퓨터 판독 가능 매체는 자성, 광, 및 반도체 저장 매체(예컨대, 하드 드라이브, 광 디스크, 플래시 메모리, DRAM)와 같은 비-일시적 매체(non-transitory medium)를 포함한다. 일부 실시예들에 따르면, 본 발명은, 그 중에서도, 본원에 설명된 방법들을 수행하기 위해 프로그래밍된 하드웨어(예컨대, 하나 이상의 프로세서)는 물론, 본원에서 설명된 방법들을 수행하기 위한 명령들을 인코딩하는 컴퓨터-판독 가능 매체를 포함하는 컴퓨터 시스템을 제공한다.

[0011] 후술하는 설명은 본 발명의 실시예들을 예시적으로 설명하는 것이며, 반드시 제한적인 것은 아니다.

[0012] 도 1은 본 발명의 일부 실시예들에 따른 안티-멀웨어 연산(작업)을 수행하는 호스트 시스템(10)의 예시적인 하드웨어 구성을 도시한다. 호스트 시스템(10)은 특히 엔터프라이즈 서버와 같은 기업용 컴퓨팅 장치, 또는 개인용 컴퓨터나 스마트폰과 같은 엔드-유저 장치를 나타낼 수 있다. 다른 호스트 시스템들은 TV 및 게임 콘솔과 같은 엔터테인먼트 장치, 또는 메모리와 프로세서를 구비하고 가상화를 지원하며 멀웨어 보호를 필요로 하는 임

의의 다른 장치를 포함한다. 도 1은 설명을 위한 컴퓨터 시스템을 도시하며, 휴대 전화 또는 태블릿과 같은 다른 클라이언트 장치들은 상이한 구성을 가질 수 있다. 일부 실시예들에서, 시스템(10)은 버스(24)의 세트에 의해 모두 연결되어 있는 프로세서(12), 메모리 유닛(14), 입력 장치(16) 세트, 출력 장치(18) 세트, 저장 장치(20) 세트, 및 네트워크 어댑터(22) 세트를 포함하는 물리적 장치들의 세트를 포함한다.

[0013] 일부 실시예들에서, 프로세서(12)는 신호 및/또는 데이터의 세트로 산술 및/또는 논리 연산을 실행하도록 구성된 물리적 장치(예컨대, 멀티-코어 집적 회로)를 포함한다. 일부 실시예들에서, 이러한 논리 연산들은 프로세서 명령(예를 들어, 기계 코드 또는 다른 유형의 소프트웨어)의 시퀀스 형태로 프로세서(12)에 전달된다. 메모리 유닛(14)은 명령들을 수행하는 도중에 프로세서(12)에 의해 액세스되거나 생성되는 데이터/신호들을 저장하는 휘발성 컴퓨터-판독 가능 매체(예컨대, RAM)를 포함할 수 있다. 입력 장치(16)는 사용자가 시스템(10)으로 데이터 및/또는 명령들을 도입할 수 있게 하는 개별 하드웨어 인터페이스 및/또는 어댑터를 포함하는, 특히 컴퓨터 키보드, 마우스, 및 마이크를 포함할 수 있다. 출력 장치(18)는 특히 모니터와 같은 디스플레이 장치 및 스피커는 물론, 시스템(10)이 사용자에게 데이터를 통신하게 할 수 있는 그래픽 카드와 같은 하드웨어 인터페이스/어댑터를 포함할 수 있다. 일부 실시예들에서, 입력 장치(16)와 출력 장치(18)는 터치-스크린 장치들의 경우와 같이, 하드웨어의 공통적인 부품을 공유할 수 있다. 저장 장치(20)는 소프트웨어 명령들 및/또는 데이터의 비휘발성 저장, 판독, 및 기록을 가능하게 하는 컴퓨터-판독 가능 매체를 포함한다. 예시적인 저장 장치(20)는 자기 디스크 및 광 디스크 및 플래시 메모리 장치들은 물론, CD 및/또는 DVD 디스크들 및 드라이브들과 같은 소거 가능 매체를 포함한다. 네트워크 어댑터(22) 세트는 시스템(10)이 컴퓨터 네트워크 및/또는 다른 장치들/컴퓨터 시스템들에 연결될 수 있게 한다. 버스(24)는 호스트 시스템(10)의 장치들(12-22)의 상호-통신을 가능하게 하는 복수의 시스템, 주변 장치, 및 칩셋 버스들, 및/또는 다른 모든 회로망을 집합적으로 나타낸다. 예를 들어, 버스(24)는 특히 프로세서(12)를 메모리(14)에 연결시키는 노스브리지, 및/또는 프로세서(12)를 장치들(16-22)에 연결시키는 사우스브리지를 포함할 수 있다.

[0014] 도 2는 본 발명의 일부 실시예들에 따른 하이퍼바이저(30)에 의해 노출되고 호스트 시스템(10) 상에서 실행되는 게스트 가상 머신(32a-b)의 예시적인 세트를 도시한다. 가상 머신(VM)은 본 기술분야에서 통상적으로 다른 VM과 무관한 고유의 운영 시스템 및 소프트웨어를 각각 구동할 수 있는 실제 물리적 머신/컴퓨터 시스템의 소프트웨어 에뮬레이션(emulation)으로 알려져 있다. 하이퍼바이저(30)는 프로세서 동작, 메모리, 저장소, 입력부/출력부, 및 네트워킹 장치들과 같은 호스트 시스템(10)의 하드웨어 자원의 다수의 가상 머신에 의해 멀티플렉싱(공유)을 허용하는 소프트웨어를 포함한다. 일부 실시예들에서, 하이퍼바이저(30)는 다수의 가상 머신들 및/또는 운영 시스템(OS)들이 다양한 고립도(degree of isolation)로, 호스트 시스템(10) 상에서 동시에 실행되게 한다. 이러한 구성을 가능하게 하기 위해, 하이퍼바이저(30)의 일부를 구성하는 소프트웨어는 복수의 가상화된, 즉 소프트웨어-에뮬레이팅된 장치(software-emulated device)들을 생성할 수 있으며, 각각의 가상화된 장치는 특히 프로세서(12) 및 메모리(14)와 같은 시스템(10)의 물리적 하드웨어 장치를 에뮬레이팅한다. 하이퍼바이저(30)는 호스트 시스템(10) 상에서 작동하는 각각의 VM에 대해 가상 장치들의 세트를 또한 할당할 수 있다. 따라서, 각각의 VM(32a-b)은 고유의 물리적 장치 세트를 구비하는 것처럼, 즉 거의 완벽한 컴퓨터 시스템인 것처럼 작동한다. 유명한 하이퍼바이저의 예로는, 특히 VMware Inc.의 VMware vSphere™ 및 오픈 소스 Xen 하이퍼바이저가 있다.

[0015] 일부 실시예들에서, 하이퍼바이저(30)는 이하에서 더욱 설명되는 바와 같은 안티-멀웨어 작업을 수행하도록 구성된 메모리 인트로스펙션 엔진(40)을 포함한다. 엔진(40)은 하이퍼바이저(30)에 통합되거나, 또는 하이퍼바이저(30)와는 구별되고 독립적인 소프트웨어 구성요소로서 전달될 수 있지만, 하이퍼바이저(30)와 실질적으로 유사한 프로세서 권한 레벨로 실행된다. 단일 엔진(40)은 호스트 시스템(10) 상에서 실행되는 다수의 가상머신에 대해서 멀웨어를 보호하도록 구성될 수 있다.

[0016] 도 2가 단순화를 위해 단지 2개의 VM(32a-b)을 도시하고 있더라도, 호스트 시스템(10)은 많은 개수의, 예컨대 수백 개의 VM을 동시에 작동시킬 수 있으며, 이러한 VM의 개수는 호스트 시스템(10)이 작동하는 동안 변경될 수 있다. 일부 실시예들에서, 각각의 VM(32a-b)은 각각 호스트 시스템(10) 상에서 다른 VM과 독립적으로 그리고 동시에 실행되는 게스트 운영 시스템(34a-b) 및/또는 소프트웨어 어플리케이션(42a-b, 42c, 및 44) 세트를 실행한다. 각각의 OS(34a-b)는 개별 VM(32a-b)의 (가상화된) 하드웨어에 대한 인터페이스를 제공하고, 개별 OS 상에서 실행되는 소프트웨어 어플리케이션을 위한 호스트로서의 역할을 하는 소프트웨어를 포함한다. 운영 시스템(34a-b)은 특히 Windows®, MacOS®, Linux®, iOS®, 또는 Android™와 같은 널리 입수가능한 운영 시스템을 포함할 수 있다. 어플리케이션(42a-c)은 특히 워드 프로세싱, 화상 처리, 데이터베이스, 브라우저, 전자 통신 어플리케이션을 포함할 수 있다. 이하의 설명에서, 가상 머신의 가상 프로세서에서 실행되는 소프트웨어는 각각의 가상

머신 내에서 실행되는 것으로 생각될 수 있다. 예를 들어서, 도 2에서, 어플리케이션(42b)은 가상 머신(32a) 내에서 실행되는 것으로 볼 수 있고, 어플리케이션(42c)은 가상 머신(32b) 내에서 실행되는 것으로 볼 수 있다. 반대로, 메모리 인트로스펙션 엔진(40)은 가상 머신(32a-b) 밖에서 실행되는 것으로 볼 수 있다.

[0017] 도 2의 예에서, 보안 어플리케이션(44)은 게스트 OS(34b)에서 실행되고, 메모리 인트로스펙션 엔진(40)과 함께 안티-멀웨어 (AM) 작업을 수행해서는 이하에서 후술하는 바와 같이 멀웨어로부터 가상 머신(32b)을 보호하도록 구성된다. 일부 실시예들에서, 어플리케이션(44)의 인스턴스는 호스트 시스템(10) 상에서 작동하는 다수의 VM들 각각에서 실행될 수 있고, 그러한 인스턴스 각각은 각각의 가상 머신을 보호하기 위하여 인트로스펙션 엔진(40)과 인터페이스하도록 구성된다. 보안 어플리케이션(44)은 자립형 프로그램(standalone program)일 수 있고, 또는 특히, 안티-멀웨어, 안티-스팸, 및 안티-스파이웨어 요소를 포함하는 소프트웨어 군(software suite)의 일부를 형성할 수 있다.

[0018] 도 3은 본 발명의 일부 실시예에 따른 호스트 시스템(10) 상에서 실행되는 소프트웨어 객체들의 계층도를 보여준다. 도 3은 본 기술분야에서 레이어(layer) 또는 보호 링(protection ring)으로 또한 알려진 프로세서 권한 레벨의 관점으로부터 도시되었다. 일부 실시예에서, 각각의 그러한 레이어 또는 보호 링은, 각각의 프로세서 권한 레벨에서 실행되는 소프트웨어 객체가 실행할 수 있는 명령 세트로 특징지어진다. 소프트웨어 객체가 각각의 권한 레벨 내에서 허용되지 않는 명령을 실행하고자 할 때, 해당 시도는 예외(exception), 오류(falut) 또는 가상 머신 종료 이벤트(exit event)와 같은 프로세서 이벤트를 촉발할 수 있다. 일부 실시예에서, 권한 레벨들 사이에서의 전환(switching)은 전용 명령 세트를 통해서 이뤄질 수 있다. 그러한 명령들의 예로는 사용자 레벨에서 커널 레벨(kernel level)로 전환하는 SYSCALL/SYSENTER, 커널 레벨로부터 사용자 레벨로 전환하는 SYSRET/SYSEXIT, 사용자 또는 커널 레벨로부터 루트 레벨로 전환하는 VMCALL, 및 루트 레벨로부터 커널 또는 사용자 레벨로 전환하는 VMRESUME 등이 있다.

[0019] 일부 실시예에서, 하이퍼바이저(30)는 최대 권한 레벨(most privileged level)(링-1 또는 루트 모드로도 알려진 가상화를 지원하는 Intel® 플랫폼의 VMXroot)에서 프로세서(12)의 제어를 획득하고 따라서 호스트 시스템(10)에서 실행되는 다른 소프트웨어에 가상 머신(32)으로서 제공되는 하드웨어 가상화 플랫폼을 생성한다. 도 2에서 OS(34a-b)와 같은 운영 시스템(34)은 VM(32)의 가상 환경 내에서 실행되고, OS(34)는 하이퍼바이저(30)보다 더 낮은 프로세서 권한을 가진다(예를 들어서, 인텔 플랫폼에서 링 0 또는 커널 모드). 어플리케이션(42d-e) 세트는 OS(34)보다 낮은 프로세서 권한(예를 들어서, 링 3 또는 사용자 모드)에서 실행된다.

[0020] 일부 실시예에서는, 보안 어플리케이션(44)의 부분들이 사용자 레벨 프로세서 권한에서, 즉 어플리케이션(42d-e)와 동일한 레벨에서 실행될 수 있다. 예를 들어서, 이러한 부분들은 사용자에게 각각의 VM에서 탐지된 모든 멀웨어 또는 보안 위협들을 통지하고, 사용자가 가리키는 것으로부터의 입력, 예를 들어서 어플리케이션(44)을 위한 바람직한 구성 옵션(configuration option)을 수신하는 그래픽 유저 인터페이스를 포함할 수 있다. 사용자 레벨에서 실행되는 구성요소의 다른 예는 이하에서 상술하는 사용자 레벨 프로세스 평가자이다. 어플리케이션(44)의 다른 부분들은 커널 권한 레벨에서 실행될 수 있다. 예를 들어서, 어플리케이션(44)은 안티-멀웨어 드라이버(36)와 프로세스 스코어링 모듈(38)을 설치할 수 있고, 이 둘은 모두 커널에서 작동한다. 예시적인 AM 드라이버(36)는 안티-멀웨어 어플리케이션(44)에, 예를 들어서 멀웨어 서명을 위한 메모리를 스캔하거나 그리고/또는 프로세스 및/또는 OS(34)에서 실행되는 다른 소프트웨어 객체들에서 멀웨어를 가리키는 거동을 탐지하기 위한 기능(functionality)을 제공한다.

[0021] 일부 실시예에서, 프로세스 스코어링 모듈(38)은 피평가 프로세스에 대해서 결정된 프로세스 평가 데이터를 다수의 소프트웨어 요소들로부터 수신하고, 피평가 프로세스가 각각의 데이터에 따라서 악성인지 여부를 판단하도록 구성된다. 프로세스는 어플리케이션 또는 운영 시스템의 일부와 같은 컴퓨터 프로그램의 인스턴스이고, 최소한 실행 쓰레드(execution thread)와 상기 운영 시스템에 의하여 상기 실행 쓰레드에 할당된 가상 메모리 섹션을 구비하는 것을 특징으로 한다(이 때 각각의 섹션은 실행 코드(executable code)를 포함한다). 일부 실시예에서, 운영 시스템은 호스트 시스템(10)에서(가상화의 경우에는 가상 머신(32) 내에서) 현재 실행되는 프로세스를 관리하는데, 그러한 관리는 특히 각 프로세스에 가상 메모리를 할당하는 것과 각 프로세스 또는 실행을 위하여 이들의 쓰레드를 스케줄링하는 것을 포함한다.

[0022] 도 4는 다수의 프로세스 평가 표시자(52a-d)를 수신하는 예시적인 프로세스 스코어링 모듈(38)을 보여주고, 여기서 각 표시자(52a-d)는 프로세스 평가자 요소에 의해서 결정된다. 도 4에서, 상기 평가자 요소는 예를 들어, 특히 사용자 레벨 프로세스 평가자(50a), 커널 레벨 프로세스 평가자(50b) 및 시스템 호출 평가자(system call evaluator)(50c)를 포함한다. 평가자(50a-c)는 안티-멀웨어 드라이버(36)에 의해서 형성되거나 그 일부를 구성

한다. 그러한 평가자 각각은 다른 평가자들과 독립적으로 실행될 수 있고, 각각은 상기 피평가 프로세스의 다수의 특징적 프로세스 평가 표시자를 결정할 수 있다. 평가자(50a-c)들의 작동은 이하에서 상세히 설명된다. 일부 실시예에서, 도 4의 표시자(52a-c)와 같은 일부 프로세스 평가 표시자들은 VM(32) 내에서 실행되는 요소들에 의해서 결정되고, 한편 도면부호 52d로 표시한 것과 같은 다른 프로세스 평가 표시자들은 VM(32) 밖에서 실행되는 요소들에 의해서(예를 들어서, 메모리 인트로스펙션 엔진(40)에 의해서) 결정된다.

[0023] 일부 평가 표시자들은 멀웨어를 표시할 수 있는데, 즉 피평가 프로세스가 악성이라는 것을 표시할 수 있다. 일부 평가 표시자들은 스스로 멀웨어를 표시할 수 없으나 다른 평가 표시자들과 결합할 때 악성을 표시할 수 있다. 각각의 평가 표시자(52a-d)는 특징적인 방법 또는 기준에 따라서 결정될 수 있다. 피평가 프로세스에 대해서 결정된 예시적인 프로세스 평가 표시자는 예를 들어서 VM(32)의 시스템 레지스터 키(system register key)의 편집 또는 보호된 소프트웨어 객체에 속하는 메모리 페이지에 쓰기와 같은 특정 작업을 피평가 프로세스가 수행하였거나 수행하기 위하여 시도하였는지 여부를 나타내는 행위 표시자(behavioral indicator)를 포함할 수 있다. 또 다른 예시적인 프로세스 평가 표시자는 피평가 프로세스에 속하는 메모리 섹션이 멀웨어를 가리키는 시그니쳐를 포함하는지 여부를 표시할 수 있다. 일부 실시예에서는, 각 프로세스 평가 표시자(52a-d)는 프로세스 ID, 라벨, 또는 해시 인덱스(hash index)와 같은 프로세스 식별 표시자를 포함할 수 있어서 모듈(38)이 각각의 표시자가 나타내는 프로세스를 식별할 수 있다.

[0024] 일부 실시예에서는, 프로세스 평가 표시자는 각 프로세스 평가자에 의해서 결정되고 각 프로세스의 악성의 정도를 가리키는 숫자로된 스코어를 포함할 수 있다. 선택적으로, 상기 스코어들은 프로세스 평가 표시자(52a-d)들에 따라서 모듈(38)에 의해서 결정될 수 있다. 악성 스코어들은 이진수일 수 있고(1/0 또는 예/아니오), 또는 연속적인 범위의 값에 걸쳐서 변화될 수 있다. 소정 값의 범위 내에서 변화할 수 있는 예시적인 악성 스코어는 피평가 프로세스가 악성일 수 있는 개연성(즉, 가능성)을 가리키는 숫자를 포함한다. 그러한 스코어는 예를 들어서 0과 1 사이 또는 0%와 100% 사이에서 변화할 수 있다. 스코어 값들은 행위에 특정적이다. 예를 들어서, 피평가 프로세스는 디스크 파일을 생성했을 때 0.2의 악성 스코어를 받을 수 있고, 윈도우 레지스트리 값(Windows registry value)을 수정했을 때 0.7의 악성 스코어를 받을 수 있다.

[0025] 도 5는 본 발명의 일부 실시예에 따른 프로세스 스코어링 모듈(38)에 의해서 실행되는 단계들의 예시적 순서를 보여준다. 단계(302)에서 모듈(38)은 VM(32)(예를 들어서 도 4의 평가자(50a-c) 참조) 내에서 또는 VM(32)(예를 들어서, 메모리 인트로스펙션 엔진(40)) 밖에서 작동할 수 있는 프로세스 평가자로부터 도 4의 표시자(52a-d)와 같은 프로세스 평가 표시자를 수신할 수 있다. 단계(304)에서, 모듈(38)은 각각의 프로세스 평가 표시자가 결정하는 프로세스를 식별할 수 있다. 일부 실시예에서, 프로세스 스코어링 모듈(38)은 다양한 프로세스 평가자들로부터 수신한 모든 프로세스 평가 표시자들의 프로세스 별 레코드(per-process record)를 보관할 수 있다. 단계(304)는 단계(302)에서 수신된 표시자를 각 프로세스의 레코드에 추가하는 단계를 또한 포함할 수 있다.

[0026] 피평가 프로세스가 악성인지 여부를 결정하기 위하여, 단계(306)에서, 프로세스 스코어링 모듈(38)은 여러 프로세스 평가자들로부터 수신하고 각 프로세스에 대해서 결정된 개별 스코어들을 합쳐서 총합 스코어를 결정할 수 있다. 예시적인 총합 스코어는 개별 스코어들의 가중 합계(weighted sum)와 가중 평균(weighted average)을 포함한다. 일부 실시예에서, 총합 스코어는 다른 프로세스 또는 소프트웨어 객체에 대해서 결정된 프로세스 평가 표시자들/스코어들과 피평가 프로세스에 대해서 결정된 프로세스 평가 표시자들/스코어들을 합칠 수 있다. 예를 들어서, 피평가 프로세스에 대해서 결정된 스코어들은 피평가 프로세스의 자식 프로세스(child process)에 대해서 결정된 스코어들, 그리고/또는 피평가 프로세스의 부모 프로세스(parent process)에 대해서 결정된 스코어들과 합산될 수 있다.

[0027] 단계(308)에서, 모듈(38)은 총합 스코어와 사전결정된 임계값(threshold)을 비교할 수 있다. 총합 스코어가 임계값을 초과하지 않는 경우, 모듈(38)은 앞에서 설명한 단계(302)로 회귀할 수 있다. 일부 실시예에서, 임계값은 각 VM의 사용자로부터(예를 들어서, 보안 어플리케이션(44)에 의해 노출된 사용자 인터페이스를 통해서) 수신된 인풋(input)에 따라서 결정된 값으로 설정될 수 있다. 임계값의 수치는 각각의 사용자의 보안 선호 정도를 반영할 수 있다. 예를 들어서, 사용자가 강력한 보안을 선택할 때 임계값은 상대적으로 낮은 값으로 설정될 수 있다. 사용자가 보다 유연한 보안 설정을 희망하는 경우 임계값은 상대적으로 높은 값으로 설정될 수 있다. 일부 실시예에서, 임계값의 수치는 도 10 내지 도 11과 관련하여 이하에서 설명하는 원격 보안 서버로부터 수신될 수 있다.

[0028] 일부 실시예에서, 단계(306 내지 308)에서, 프로세스 스코어링 모듈(38)은 복수의 총합 스코어들을 결정할 수 있고, 각 총합 스코어를 (가능한 특징적인) 임계값과 비교할 수 있다. 각각의 그러한 총합 스코어는 프로세스

평가 표시자들의 특징적 하위 세트(distinct subset)에 따라서 결정될 수 있다. 예시적 실시예에서, 프로세스 평가 표시자들의 각각의 그러한 세트는 멀웨어(예를 들어, 트로잔, 루트킷 등)의 특정 군(class) 또는 유형을 나타낼 수 있고, 모듈(38)이 탐지된 멀웨어의 분류를 수행할 수 있도록 한다.

[0029] 총합 스코어가 임계값을 초과하는 경우, 단계(310)에서, 모듈(38)은 피평가 프로세스가 악성이라고 결정할 수 있고, 또한 안티-멀웨어 조치를 취할 수 있다. 일부 실시예에서, 그러한 안티-멀웨어 조치는, 특히, 피평가 프로세스를 종료하는 것, 피평가 프로세스를 격리하는 것, 피평가 프로세스의 리소스(파일 또는 메모리 섹션 등)를 제거하거나 불능화(disabling)하는 것을 포함한다. 일부 실시예에서, 안티-멀웨어 조치는 예를 들어서 네트워크 어댑터(들)(22)를 통해서 호스트 시스템(10)에 연결된 컴퓨터 네트워크 상에서 시스템 관리자들에게 메시지를 보냄으로써, 호스트 시스템(10)의 사용자에게 경고하는 것, 그리고/또는 시스템 관리자에게 경고하는 것을 또한 포함할 수 있다. 일부 실시예에서, 안티-멀웨어 조치는 또한 도 10 내지 도 11과 관련하여 이하에서 설명하는 바와 같이 원격 보안 서버에 보안 리포트를 송부하는 것을 또한 포함할 수 있다.

[0030] 도 3 내지 도 4에 도시된 예시적인 프로세스 스코어링 모듈(38)은 OS 프로세서 권한 레벨(예를 들어서, 커널 모드)에서 VM(32) 내에서 작동한다. 선택적 실시예에서, 프로세스 스코어링 모듈(38)은 사용자 모드에서 VM(32) 내에서, 또는 VM(32) 밖에서도, 하이퍼바이저(30)의 프로세서 권한 레벨에서 실행될 수 있다.

[0031] 일부 실시예에서, 인트로스펙션 엔진(40)은 하이퍼바이저(30)와 동일한 권한 레벨에서 실질적으로 실행되고, VM(32)과 같은 가상 머신의 인트로스펙션을 수행하도록 구성된다. VM 또는 각각의 VM에서 실행되는 소프트웨어 객체의 인트로스펙션은, 특히, 소프트웨어 객체의 행동을 분석하는 것, 그러한 소프트웨어 객체들의 메모리 주소들을 결정 및/또는 액세스하는 것, 그러한 주소들에 위치한 메모리의 콘텐트에 특정 프로세스가 접근하는 것을 제한하는 것, 그러한 콘텐트를 분석하는 것, 및 각 소프트웨어 객체들의 프로세스 평가 표시자들(예를 들어, 도 4의 표시자(52d))을 결정하는 것을 포함할 수 있다. 일부 실시예에서, 인트로스펙션 엔진(40)에 의해서 타게팅된 소프트웨어 객체들은, 특히, 프로세스와, 명령 스트림(instruction stream)과, 레지스터(register)와, 각 VM의 페이지 테이블 및 드라이버 객체와 같은 데이터 구조를 포함한다.

[0032] 각 VM의 외부로부터 VM(32)의 인트로스펙션을 수행하기 위하여, 엔진(40)의 일부 실시예에서는 프로세서(12)의 메모리 맵핑 구조와 메커니즘을 채용한다. 가상 머신들은 통상적으로는 가상화된 물리적 메모리, 즉, 호스트 시스템(10)의 실질적인 물리적 메모리(14)의 가상 표현체(virtual representation)와 함께 작동한다. 가상화된 물리적 메모리는 호스트 시스템(10)에서 실행되는 각 게스트 VM에 특정된 가상화된 주소의 인접 공간(contiguous space)과 물리적 메모리(14) 및/또는 물리적 저장 장치(20) 내에서 주소들에 맵핑된 각 공간의 부분들을 함께 포함한다. 가상화를 지원하도록 구성된 시스템들에서, 그러한 맵핑은 확장된 페이지 테이블(extended page table, EPT) 또는 내포된 페이지 테이블(nested page table, NPT)과 같은 프로세서(12)에 의해서 제어되는 전용 데이터 구조들에 의하여 일반적으로 달성된다. 그러한 시스템들에서, 가상화된 물리적 메모리는 당해 기술분야에서 페이지로 알려진 유닛들로 파티션될 수 있다. 페이지는 EPT 및/또는 NPT와 같은 메커니즘을 통해서 물리적 메모리에 개별적으로 맵핑된 가상화된 물리적 메모리의 최소 유닛을 나타내며, 다시말해서 물리적 메모리와 가상화된 물리적 메모리 사이의 맵핑은 페이지 단위(page granularity)로 수행된다. 모든 페이지들은 통상적으로 소정의 크기, 예를 들어 4 킬로바이트, 2 메가바이트 등의 크기를 가진다. 가상화된 물리적 메모리를 페이지들로 파티셔닝하는 것은 대개 하이퍼바이저(30)에 의하여 설정된다. 일부 실시예에서, 하이퍼바이저(30)는 또한 EPT/NPT를 설정하고 따라서 물리적 메모리와 가상화된 물리적 메모리 사이의 맵핑을 설정한다. 가상화된 물리적 메모리 주소를 물리적 메모리 주소로 실질적으로 변환하는 것은 호스트 시스템(10)의 변환 색인 버퍼(translation lookaside buffer, TLB)에서 물리적 메모리 어드레스를 색인(look-up)하는 것을 포함할 수 있다. 일부 실시예에서, 주소 변환은 페이지 테이블 세트에서 연속적인 주소 색인의 세트를 포함하는 페이지 검색(page walk)을 수행하는 것과, 페이지의 오프셋(offset)을 각 페이지에 대한 주소에 추가하는 것과 같은 연산을 수행하는 것을 포함한다.

[0033] 일부 하드웨어 설정은 예를 들어서, 각 페이지에 대하여 읽기/쓰기 접근 권한을 설정함으로써 하이퍼바이저(30)가 각 페이지 내에 저장된 데이터의 접근 권한을 선택적으로 제어할 수 있게 한다. 상기 권한은 예를 들어서, EPT 또는 NPT 내의 각 페이지의 엔트리(entry)를 수정해서 설정될 수 있다. 따라서 하이퍼바이저(30)는 어떤 소프트웨어 객체가 각 페이지 내의 주소들에 저장된 데이터에 액세스할 것인지 결정할 수 있고, 그리고 어떠한 작업(예를 들어서, 읽기, 쓰기 등)이 각 데이터에 대해서 허용되는지 지정할 수 있다. 소프트웨어 객체가 각 권한을 가지지 않는 페이지로부터 데이터를 읽거나 또는 그 페이지에 데이터를 기록하는 것과 같은 작업을 수행하려는 VM 내에서 실행되는 소프트웨어 객체에 의한 시도는 (예를 들어서 인텔 플랫폼에서의 VMExit 이벤트와 같은) 가상 머신 종료 이벤트(exit event)를 촉발할 수 있다. 일부 실시예에서, 가상 머신 종료 이벤트는 프로세서의

제어를 각 소프트웨어 객체를 실행하는 VM으로부터 하이퍼바이저(30) 또는 메모리 인트로스펙션 엔진(40)으로 이동시키고 따라서 하이퍼바이저(30) 및/또는 엔진(40)이 비인가 읽기/쓰기 시도를 가로막고 분석할 수 있게 한다.

[0034] 일부 실시예들에서, OS(34)는 가상 메모리 공간(논리 주소 공간(logical address space)이라고도 함)을 설정하고, 상기 가상 메모리 공간을 도 3의 어플리케이션(42d-e 및 44)과 같은 어플리케이션에 노출시킨다. 이러한 시스템들에서, OS(34)는 예컨대 페이지 테이블 메커니즘을 사용하여 VM(32)의 가상화된 물리적 메모리와 상기 가상 메모리 공간 사이의 맵핑을 설정하고 및 유지한다. 일부 실시예들에서, 상기 가상 메모리 공간은 또한 페이지들로 구획되며, 이러한 페이지들은 OS(34)에 의해 가상화된 물리적 메모리로 개별적으로 맵핑된 가상 메모리의 최소 단위를 나타낸다(가상 메모리에서 가상화된 물리적 메모리로의 맵핑은 페이지 단위(page granularity)로 수행된다).

[0035] 도 6은 도 2에 도시된 바와 같은 실시예의 메모리 주소들의 예시적인 맵핑(변환)을 나타낸다. VM(32a) 내에서 실행되는 프로세스 또는 어플리케이션과 같은 소프트웨어 객체는 게스트 OS(34a)에 의해 가상 주소 공간(214a)을 할당받는다. 개별 소프트웨어 객체가 공간(214a)의 예시적인 메모리 주소(60a)에 대한 액세스를 시도하면, 주소(60a)는 게스트 OS(34a)에 의해 구성 및 제어되는 페이지 테이블들에 따라 게스트 VM(32a)의 가상화된 프로세서에 의해 가상 머신(32a)의 가상화된 물리적 메모리 공간(114a) 내의 주소(60b)로 번역된다. 주소(60b)는 본 기술분야에서 게스트-물리적 주소라고도 알려져 있다. 가상화된 물리적 메모리(114a)를 구성 및 제어하는 하이퍼바이저(30)는 예컨대 상술한 EPT 또는 NPT 수단을 사용하여, 주소(60b)를 호스트 시스템(10)의 물리적 메모리(14) 내의 주소(60c)로 맵핑한다.

[0036] 이와 유사하게, 가상 메모리 공간(214b)은 게스트 VM(32b) 상에서 실행되는 다른 소프트웨어 객체들 또는 어플리케이션들(예컨대, 42c)을 위해 게스트 OS(34b)에 의해 설정된다. 공간(214b) 내의 예시적인 가상 주소(60d)는, 게스트 OS(34b)에 의해 구성 및 제어되는 페이지 테이블에 따라, 게스트 VM(32b)의 가상화된 프로세서에 의해 게스트 VM(32b)의 가상화된 물리적 메모리 공간(114b) 내의 주소(60e)로 번역된다. 주소(60e)는 하이퍼바이저(30)에 의해 물리적 메모리(14) 내의 주소(60f)로 또한 맵핑된다.

[0037] 일부 실시예들에서, 하이퍼바이저(30)는 물리적 메모리(14)의 표현체를 포함하는 그 자신의 가상 메모리 공간(214c)을 설정하고, 공간(214c) 내의 주소를 물리적 메모리(14) 내의 주소로 맵핑하는 변환 메커니즘(예컨대, 페이지 테이블)을 사용한다. 도 6에서, 그러한 예시적인 맵핑은 주소(60g)를 주소(60h)로 번역한다. 이와 유사하게, 물리적 메모리(14) 내의 도면부호 60c 및 60f와 같은 주소들은 하이퍼바이저(30)의 가상 메모리 공간(214c) 내에서 각각 주소들(60k 및 60m)에 대응한다. 상기 변환은 하이퍼바이저(30)과 호스트 시스템(10)에서 작동하는 여러 VM들 내에서 실행되는 소프트웨어 객체들에 속하는 메모리 페이지에 대한 관리(예를 들어, 읽기, 쓰기 및 접근 제어)가 가능하게 한다.

[0038] 도 7은 본 발명의 일부 실시예들에 따른 가상머신(32)에서 실행되는 프로세스(70a-b)들의 세트의 예시적인 실행 흐름도를 도시한다. 도 7의 예는 Windows® 운영체제(OS) 버전에서 작동하는 시스템에서 실행 흐름도를 보여준다. 유사한 다이어그램들이 예를 들어서 리눅스와 같은 다른 운영 체제를 위해서 제공될 수 있다. 실선 화살표는 보안 어플리케이션(44)과 같은 안티-멀웨어 시스템이 없는 경우의 실행 흐름을 나타낸다. 점선 화살표는 본 발명의 일부 실시예들에 따라서 실행되는 프로세스 평가자들이 존재하는 경우의 흐름도가 수정되는 것을 나타낸다.

[0039] 프로세스(70a)는 다수의 DLL(dynamic linked library)(72a-c)을 포함한다. 도 7의 예에서, DLL(72c)은 (악성 가능성이 있는) 프로세스(70b)에 의해서 프로세스(70a)내로 주입된다. 코드 주입(code injection)은 각 프로세스의 원래 기능을 변경하기 위하여 DLL과 같은 코드 시퀀스(code sequence)를 기존 프로세스의 메모리 공간으로 도입하는 방법들의 군을 가리키는 본 기술분야에서 통용되는 용어이다. 프로세스(70a)가, 예를 들어서 디스크 파일에 무언가를 기록하거나, 레지스트리 키를 편집하는 것과 같은 몇 가지 시스템 기능을 요청하는 명령을 실행할 때, 각각의 명령은 KERNEL32.DLL 또는 NTDLL.DLL과 같은 사용자 모드 API를 호출한다. 도 7의 예에서, 각 사용자 모드 API 호출은 사용자 레벨 행동 필터(user behavioral filter,50a)에 의해서 중간차단되고 분석된다. 그와 같은 중간차단(interception)은 특히 DDL 주입 또는 후킹(hooking)과 같은 방법에 의해서 행해질 수 있다. 후킹은 소프트웨어 요소들간에 전달되는 기능 호출, 메시지 또는 이벤트를 가로채는 방법에 대한 본 기술분야에서 통용되는 용어이다. 후킹 방법의 일예는 명령 우회 실행(instruction redirecting execution)을 제2 함수에 삽입하여 타겟 함수(target function)의 엔트리 포인트를 변경하는 것을 포함한다. 그러한 후킹 다음에, 제2 함수가 타겟 함수 대신에, 즉 그 이전에 실행될 수 있다. 도 7의 예에서, 안티-멀웨어 드라이버(36)는

KERNEL32.DLL 또는 NTDLL.DLL의 특정 함수들로 연결(hook)되서 각 함수들이 필터(50a)로 실행되도록 우회 명령할 수 있다. 따라서, 필터(50a)는 프로세스(70a)가 우회를 실행하는 함수에 따라서 식별된 특정의 행동을 수행하는 것을 시도하는 것을 탐지할 수 있다. 필터(50a)가 그러한 행동을 탐지할 때, 필터(50)는 프로세스 평가 표시자(52a)(도 4)를 구축하고 표시자(52a)를 프로세스 스코어링 모듈(38)에 전달할 수 있다.

[0040] 통상적인 실행 흐름도에서는, 사용자 모드 API 함수는 운영 시스템의 커널로부터 서비스를 요청할 수 있다. 일부 실시예에서, 그러한 작업들은 x86 플랫폼에서 SYSCALL 및 SYSENTER와 같은 시스템 호출을 발급함으로써 수행된다. 도 7의 예에서, 그러한 시스템 호출들은 시스템 호출 평가자(50c)에 의해서 중간차단된다. 일부 실시예에서, 그러한 중간차단은 예를 들어서 프로세서(12)의 모델 특정 레지스터(model-specific register, MSR)에 저장된 값을 변경시킴으로써 시스템 호출 핸들러 루틴(system call handler routine)을 수정하는 것을 포함할 수 있고, 이것은 필터(50c)로의 실행을 효과적으로 우회시킨다. 이와 같은 기술들은 MSR 후킹으로서 본 기술분야에서 알려져 있고 시스템 호출 평가자(50c)가 피평가 프로세스가 특정 시스템 호출을 수행하고자 시도하는 하는 것을 탐지할 수 있게 할 수 있다. 그러한 시스템 호출이 중간차단되면 시스템 호출 필터(50c)는 프로세스 평가 표시자(52c)를 구축하고 표시자(52c)를 프로세스 스코어링 모듈(38)에 전달할 수 있다.

[0041] 상기 시스템 호출에 이어서, 프로세스의 제어가 일반적으로 OS(34)의 커널에게 넘어간다. 일부 실시예에서, 커널 레벨 프로세스 평가자(50b)는 OS 커널의 특정 작업을 중간차단하고, 이에 따라서 피평가 프로세스가 악성일 수 있는 특정 작업을 수행시도하는지를 결정하도록 구성된다. 그러한 작동을 중간차단하기 위하여, 일부 실시예에서는 OS(34) 내에 구축되고 OS(34)에 의해서 노출되는 필터링 메커니즘 세트를 채용할 수 있다. 예를 들어서, 윈도우스 OS에서는, FltRegisterFilter가 파일 생성/열기/쓰기/삭제와 같은 작업을 중간차단하기 위하여 사용될 수 있다. 다른 예에서, 평가자(50b)가 ObRegisterCallback을 이용해서 객체-핸들 작업(object-handle operation)을 생성 또는 복사(duplicate)하는 것을 중간차단할 수 있고, 또는 PsSetCreateProcessNotifyRoutine을 사용해서 새로운 프로세스의 생성을 차단할 수 있다. 또 다른 예에서, 레지스트리 키/값을 생성하고 설정하는 것과 같은 윈도우스 레지스트리 작업들은 CmRegisterCallbackEx을 사용하여 중간차단될 수 있다. Linux®과 같은 다른 운영 시스템들을 위한 유사한 필터링 메커니즘이 알려져 있다. 커널-모드 프로세스 평가자(50b)가 그러한 작업을 중간차단할 때, 평가자(50b)는 프로세스 평가 표시자(52b)를 구축하고 표시자(52b)를 프로세스 스코어링 모듈(38)에 전달할 수 있다.

[0042] 프로세스 평가 표시자(52a-c)와 같은 데이터를 평가자(50a-c)로부터 스코어링 모듈(38)로 전달하기 위하여, 통상의 기술자라면 임의의 프로세스간 통신 방법(inter-process communication method)을 사용할수 있다. 예를 들어서, 사용자 모드 요소와 커널 모드 요소 사이의 통신을 위하여 평가자(50a-c)와 모듈(38)은 공유된 메모리 섹션을 사용하도록 구성될 수 있다.

[0043] 도 8은 본 발명의 일부 실시예에 따라서 메모리 인트로스펙션 엔진(40)에 의해서 수행된 단계들의 예시적 시퀀스를 보여준다. 단계(312)에서, 엔진(40)은 멀웨어로부터 보호가 필요한 프로세스(이하에서는 "피보호 프로세스"라 호칭)가 VM(32) 내에서 시작되는 것을 탐지할 수 있다. 일부 실시예에서, 상기 피보호 프로세스는, 특히, 보안 어플리케이션(44)에 속하는 프로세스를 포함한다.

[0044] 피보호 프로세스의 개시를 탐지하기 위하여, 엔진(40)은 OS(34)에 고유한 데이터 구조 및/또는 메커니즘을 채용할 수 있다. 예를 들어서, Windows® OS의 일부 버전들은 커널에 의하여 유지되는 활성 프로세스(active process)의 리스트를 사용하여 프로세스를 관리한다. 프로세스가 생성될 때마다, 각 프로세스의 표시자는 상기 활성 프로세스의 리스트로 삽입된다. 상기 표시자는 상기 각 프로세스의 종료시에 상기 리스트로부터 제거된다. 일부 실시예에서, OS(34)의 커널은, 각 프로세스를 데이터 구조로서 표현하는데, 특히, 각 프로세스의 쓰레드(thread)들 각각에 대한 핸들(handle)과 OS(34)가 다수의 실행 프로세스로부터 각 프로세스를 식별할 수 있도록 하는 고유 프로세스 ID를 포함하는, 예를 들어서 윈도우스에서 실행 프로세스 블록(executive process block)(EPROCESS)로서 표현한다.

[0045] 피보호 프로세스(도 8의 단계 312)의 생성을 탐지하기 위하여, 일부 실시예들은 본 기술분야에서 알려진 임의의 후킹 방법을 사용하여 활성 프로세스의 리스트를 조작하는 커널 함수로 후킹한다. 윈도우스 OS의 그러한 함수들의 예로는 각 프로세스가 개시되어 실행될 때 활성 프로세스의 리스트를 프로세스에 추가하는 PspInsertProcess가 있다. AM 드라이버(36)의 일부 실시예들은 VMCALL 명령 또는 JMP 명령과 같은 각 커널 함수에 우회 패치(re-direction patch)를 적용할 수 있다. 다른 실시예들은 새로운 주소로 지정하기 위하여 각 커널 함수의 EPT 엔트리를 수정할 수 있다. 그러한 패치 및/또는 EPT 후크(hook)의 효과는 고유한 OS 함수의 실행을 메모리 인트로스펙션 엔진(40)에 의해서 제공되는 코드 단편(fragment of code)으로 우회시킬 수 있는 것이다. 후킹 다음으로,

OS(34)가 프로세스를 개시하여 실행하고자 시도할 때, 상기 코드 단편은 각 커널 함수의 코드 전에 또는 그 대신에 실행될 것이고, 따라서 각 프로세스가 실행중임을 메모리 인트로스펙션 엔진(40)에 통보한다. 일부 실시예에서, 엔진(40)은 각 프로세스가 시작될 때 커널 함수로 전달된 파라미터(예를 들어서, 고유 프로세스 ID를 포함하는 EPROCESS 구조)에 따라서 각 프로세스를 식별할 수 있다. 다른 실시예는 (EPT 후크와 같은) 메모리 후크를 사용하여 활성 프로세스의 리스트를 저장하는 메모리 섹션의 주소에 대한 접근을 확보할 수 있고, 각 메모리 섹션의 콘텐트에 따라서 추가적으로 현재 실행 중인 각 프로세스를 설명하는 EPROCESS 구조의 주소를 결정할 수 있다.

[0046] 단계 314에서, 메모리 인트로스펙션 엔진(40)은 AM 드라이버(36)에게 피보호 프로세스가 실행중임을 통보할 수 있다. 예를 들어서, 엔진(40)은 피보호 프로세스의 프로세스 ID와 같은 표시자를 AM 드라이버(36)에 보낼 수 있다. 다음으로, 단계 316에서, 엔진(40)은 드라이버(36)로부터 메모리 페이지의 표시자(예를 들어서, 가상 메모리의 페이지의 주소)를 수신할 수 있고, 이 때 메모리 페이지는 피보호 프로세스의 코드 및/또는 데이터를 저장한다. 일부 실시예에서, 엔진(40)은 단계 314 내지 316을 사용해서 의미 간격(semantic gap)을 연결하는데, 이 의미 간격은 엔진(40)은 VM(32) 밖에서 실행되는 반면에 피보호 프로세스는 VM(32) 내에서 실행되기 때문에 나타난다. VM(32) 내의 커널 모드에서 실행됨으로써 AM 드라이버(36)는 피보호 프로세스의 코드 및/또는 데이터를 저장하는 각 VM의 가상화된 물리적 메모리 내의 페이지 주소(도 6의 스페이스(114a-b) 참조)와 같은 피보호 프로세스에 의해서 사용되는 메모리 주소와 같은 정보에 직접적인 접근권을 가질 수 있다. 하이퍼바이저(30)가 각 VM 내에서 실행되는 활성 프로세스 리스트에 대한 접근권을 얻을 수 있음에도, 각 프로세스에 의해서 로딩(loading)된 (DLL과 같은) 모든 모듈을 결정하기 위하여 상기 리스트를 파싱하는 것과 하이버파이저(30)의 레벨로부터 데이터/코드를 저장하는 메모리 페이지들의 모든 주소를 추가적으로 결정하는 것은 실질적인 연산을 필요로 할 수 있다. 일부 실시예에서, 단계(314 내지 316)들의 시퀀스의 다른 이유는 사용자 모드 프로세스에 속하는 데이터가 물리적 메모리(14)와 다른 컴퓨터 판독가능한 매체, 예를 들어서 저장 장치(20)들과의 사이에서 OS(34)에 의해서 스왑될 수 있다는 것이다. 각 VM의 밖에서 실행되기 때문에 메모리 인트로스펙션 엔진(40)은 데이터가 물리적 메모리의 안과 밖에서 스왑될 때를 탐지할 수 있으나, 데이터가 물리적 메모리 내에 존재하지 않은 동안에는 데이터 접근 및/또는 데이터 보호를 할 수 없다. 반대로, VM(32) 내에서 실행되는 AM 드라이버(36)는 OS(34)가 각 페이지를 로드하도록 함으로써 물리적 메모리 밖에서 스왑되는 페이지에 쉽게 접근할 수 있다. 그래서 AM 드라이버(36)은 피보호 프로세스에 의해서 사용/로딩된 모든 모듈을 효과적으로 리스팅(listing)할 수 있고 VM(32)의 가상화된 물리적 메모리 내의 그러한 모듈의 크기와 위치를 효과적으로 결정할 수 있다.

[0047] 다른 실시예에서, (위 단계 312에서와 같이) 피보호 프로세스의 시작을 활발하게 탐지하는 것 대신에 메모리 인트로스펙션 엔진(40)은 AM 드라이버(36)로부터 피보호 프로세스의 표시자를 수신할 수 있고, 이 때 AM 드라이버(36)는 VM(32) 내로부터 피보호 프로세스의 시작을 실제 탐지할 수 있다. 그러한 실시예에서, 앞서 설명한 단계 314는 더 이상 필요하지 않다. 또 다른 실시예에서, 단계 316에서, 엔진(40)은 상술한 AM 드라이버(36)에 의존하는 것 대신에 피보호 프로세스의 메모리 페이지의 주소를 결정하기 위하여 필요한 연산을 실제 수행할 수 있다.

[0048] 단계 318에서, 메모리 인트로스펙션 엔진은, 예를 들어서 VM(32)을 손상(compromising)하려고 시도하는 악성 소프트웨어에 의하여 타겟 페이지가 원하지 않은 수정이 가해지는 것을 방지한다. 그와 같은 메모리 보호 메커니즘 여럿이 본 기술분야에서 알려져 있다. 메모리 인트로스펙션 엔진(40)에 요청 시 EPT 또는 NPT와 같은 데이터 구조를 이용하여 하이퍼바이저(30)가 그러한 보호가 가능하도록 할 수 있다. 예를 들어서, 하이퍼바이저(30)는 각 페이지들의 EPT/NPT 접근 권한 비트(access right bit)를 수정함으로써 타겟 메모리 페이지를 읽기 전용으로 설정할 수 있다. 일부 실시예에서, 하이퍼바이저(30)는 타겟 객체에 할당된 메모리 페이지에 쓰려는 모든 시도를 중단차단할 수 있고 각 시도를 분석을 위해 메모리 인트로스펙션 엔진(40)으로 우회시킬 수 있다. 단계 318에서 엔진(40)의 작동은 도 9를 참조하여 이하에서 상세히 설명한다.

[0049] 타겟 페이지에 쓰기 보호를 적용하기 위하여, 단계 318은 피보호 프로세스를 위하여 OS(34)가 설정한 가상 메모리 공간(virtual memory space)으로부터 호스트 시스템(10)의 물리적 메모리(14) 전체까지, 또는 각 VM의 가상화된 물리적 메모리 공간으로부터 물리적 메모리(14)까지, 도 6에 설명된 종류의 메모리 주소의 변환을 수행하는 것을 포함할 수 있다. 각각의 변환은 메모리 인트로스펙션 엔진(40)이 AM 드라이버(36)로부터 단계 316에서 받은 표시자에 따라서, 실제 물리적 메모리(14)의 타겟 페이지의 주소를 결정할 수 있도록 해준다. 그러한 변환은 도 6과 관련하여 설명한 바와 같은 EPT/NPT 메커니즘을 채용할 수 있다.

[0050] 단계 320에서, 엔진(40)은 피보호 프로세스의 종료를 탐지할 수 있다. 일부 실시예에서, 단계 320은 상술한 단계 312와 유사한 방법으로 진행될 수 있다. 예를 들어서, 단계 320은 VM(32)의 활성 프로세스들 리스트로부터

프로세스를 제거하도록 구성된 커널 함수로부터의 신호를 수신하는 것을 포함할 수 있는데, 각 함수는 후킹(예를 들어서, VMCALL 명령과 같은 엔진(40)으로의 실행을 우회시키는 패치를 각 함수에 적용하는 것)에 의해서 수정된다. 이와 같은 방식으로 수정될 수 있는 예시적은 윈도우스 함수는 PspDeleteProcess이다. 엔진(40)이 피보호 프로세스의 종료를 탐지할 때, 예를 들어서 하이퍼바이저(30)가 타겟 페이지를 위한 쓰기 허가를 변경하도록 명령함으로써 단계 322은 각 타겟 페이지로부터 보호를 제거한다.

[0051] 도 9는 타겟 페이지를 보호하기 위하여(도 8의 단계 318) 메모리 인트로스펙션 엔진(40)에 의해서 수행되는 단계들의 시퀀스를 보여준다. 단계 332에서, 엔진(40)은 타겟 페이지에 쓰려는 시도를 중간차단할 수 있다. 그러한 시도는 악의적 의도를 나타낼 수 있고 상술한 바와 같이 하이퍼바이저(30)에 의하여 중간차단된다. 단계 334에서, 엔진(40)은 상기 시도를 실행하는 프로세스를 식별할 수 있다. 각 프로세스는 공격 프로세스(offending process)로 호칭된다. 일부 실시예에서, 단계 334를 실행하기 위하여, 엔진(40)은 x86 시스템에서 EIP 및/또는 RIP 레지스터와 같은 명령 포인터 레지스터(instruction pointer register)의 콘텐트를 사용하여 상기 시도를 수행하는 프로세서 명령(또는 그 주소)를 식별할 수 있고, CR3 레지스터 콘텐트를 사용하여 각 명령이 속하는 프로세스를 식별할 수 있다. 선택적으로는 엔진(40)은 x86 프로세서의 FS 레지스터와 GS 레지스터와 같은 세그먼트 레지스터(segment register)의 콘텐트를 사용하여 프로세스 간에 OS(34)가 실행을 전환할 때마다 수정되는 특정 커널 데이터 구조들에 따라서 상기 공격 프로세스를 식별할 수 있다.

[0052] 단계 336에서, 엔진(40)은 공격 프로세스의 프로세스 평가 표시자(52d)(예를 들어 도 4 참조)를 구축하고 표시자(52d)를 프로세스 스코어링 모듈(38)로 전송할 수 있다. 예시적 표시자(52d)는 단계 334에서 식별된 공격 프로세스의 표시자(예를 들어서, 프로세스 ID)와, 공격 프로세스에 의해서 시도되고 단계 332에서 차단된 유형의 행동(예를 들어서 보호된 메모리 페이지에 쓰기 시도)의 표시자를 포함할 수 있다.

[0053] 상술한 방법과 시스템의 일부는 VM(32) 내에서 실행되는 요소들과 각 VM의 밖에서 실행되는 요소들 사이에서 데이터 교환, 및/또는 메시징과 같은 통신이 필요하다. 그러한 통신은 가상화의 기술분야에서 공지된 임의의 방법을 사용하여 수행될 수 있다. 예를 들어서, AM 드라이버(36)와 같은 커널 모드에서 실행되는 요소로부터 메모리 인트로스펙션 엔진(40)으로 데이터를 전송하기 위하여(예를 들어서 도 8의 단계 316 참조) 일부 실시예는 특권 명령(privileged instruction)을 사용하여 프로세서(12)의 제어를 VM(32)으로부터 하이퍼바이저(30)으로 넘긴다. 그러한 특권 명령의 예시는 인텔 플랫폼에서의 VMCALL인데, 이것은 VM(32) 내에서부터 일부 데이터가 전송되는 엔진(40)에 신호를 보내기 위하여 사용될 수 있다. 전송되는 실제 데이터는 드라이버(36)와 엔진(40) 사이에서 공유된 메모리의 소정 섹션에 위치될 수 있다. 메모리 인트로스펙션 엔진(40)으로부터 AM 드라이버(36)로 데이터를 전송하기 위하여(예를 들어서, 도 8의 단계 314와 도 9의 단계 336 참조), 일부 실시예들은 인터럽트 인젝션 메커니즘(interrupt injection mechanism)을 사용하여 데이터가 각 VM의 밖으로부터 전송되는 드라이버(36)에 신호를 보낼 수 있다. 실제 데이터는 예를 들어서 상술한 공유된 메모리 섹션을 통하여 전송될 수 있다.

[0054] 일부 실시예에서, 호스트 시스템(10)은 원격 보안 서버에 의해서, 멀웨어 탐지 이벤트들에 대한 상세와 같은 보안 정보를 교환하도록 구성될 수 있다. 도 10은 그러한 예시적 구성을 보여주는데, 여기서는 다수의 호스트 시스템(10a-c)들이 컴퓨터 네트워크(26)를 통해서 보안 서버(110)에 연결되어 있다. 예시적 실시예에서, 호스트 시스템(10a-c)은 회사의 종업원들에 의하여 사용되는 개별 컴퓨터들인 반면에 보안 서버(110)는 시스템(10a-c)에서 발생하는 멀웨어 위협이나 보안 이벤트들을 모니터링하기 위한 개별 회사의 네트워크 관리자에 의해서 구성된 컴퓨터 시스템을 포함한다. 다른 실시예에서는, 예를 들어서, 각 호스트 시트템(10a-c)이 수십 또는 수백개의 가상 머신을 호스팅하는 서버인 IAAS(Infrastructure-as-a-service) 시스템에서, 보안 서버(110)는 중앙 위치로부터 그러한 모든 VM을 위한 안티-멀웨어 작업을 관리하도록 구성되는 컴퓨터 시스템을 포함할 수 있다. 또 다른 실시예에서는, 보안 서버(110)는 네트워크(26) 주변의 여러 시스템들에서 탐지된 멀웨어에 대한 통계적 그리고/또는 행동 데이터를 수신하기 위하여, (예를 들어서 특히 보안 어플리케이션(44)의 제공자와 같은) 안티-멀웨어 소프트웨어 제공자에 의해서 구성된 컴퓨터 시스템을 포함할 수 있다. 네트워크(26)는 인터넷과 같은 광역 통신망을 포함할 수 있는 한편, 네트워크(26)의 부분들은 LAN(local area network)을 포함할 수 있다.

[0055] 도 11은 도 10에 도시된 바와 같은 실시예에서 호스트 시스템(10)과 보안 서버(10) 사이의 예시적 데이터 교환을 보여준다. 호스트 시스템(10)은 서버(110)에 보안 리포트(80)를 보내도록 구성될 수 있고, 서버(110)로부터 보안 설정(82) 세트를 수신하도록 구성될 수 있다. 일부 실시예에서, 보안 리포트(80)는 특히 프로세스 평가 표시자 및/또는 호스트 시스템(10)에서 실행되는 프로세스 평가자에 의해서 결정된 스코어, 및/또는 프로세스 스코어링 모듈(38)에 의해서 결정된 총합 스코어를 포함한다. 보안 리포트(80)는 또한 각 가상 머신과 피평과 프로세스를 식별하는 데이터(예를 들어서, 프로세스 ID, 이름, 경로, 해쉬, 버전 정보 또는 어플리케이션 및/또는

프로세스의 다른 종류의 식별자)뿐만 아니라 프로세스 평가 표시자/스코어를 이것과 관련되어 결정되는 VM 및 프로세스와 연결하는 표시자를 포함할 수 있다. 일부 실시예에서, 리포트(80)는 호스트 시스템(10)에서 실행되는 프로세스 및/또는 어플리케이션에 대한 통계 데이터 및/또는 행동 데이터를 추가적으로 포함할 수 있다. 시스템(10)은 멀웨어의 탐지 시에, 그리고/또는 스케줄에 따라서(예를 들어서 매 수분, 매 시간 등) 리포트(80)를 보내도록 구성될 수 있다.

[0056] 일부 실시예에서, 보안 설정(82)은 프로세스 평가자의 작업 파라미터(operational parameter)(예를 들어서, 도 4의 필터(50a-c)들의 파라미터들), 및/또는 프로세스 스코어링 모듈(38)의 파라미터들을 포함할 수 있다. 모듈(38)의 작업 파라미터의 예는 피평가 프로세스가 악성인지 여부를 결정하기 위한 임계값이다(도 5의 단계 308와 관련 기재 참조). 프로세스 평가자의 예시적 작업 파라미터는 피평가 프로세스가 특정 행동을 수행할 때 피평가 프로세스에 할당된 악성 스코어의 수치이다. 예를 들어서, 피평가 프로세스는 각 프로세스가 디스크 파일에 쓰기를 할 때 0.1의 악성 스코어를 받을 수 있고, 각 프로세스가 윈도우스 레지스트리 값을 변경할 때 0.7의 악성 스코어를 받을 수 있다.

[0057] 일부 실시예에서, 서버(110)는 멀웨어 탐지 성능을 최대화하기 위하여, 예를 들어서 긍정 오류(false positive)를 감소시키면서 탐지율을 증가시키기 위하여 위와 같은 파라미터들을 동적으로 조정하기 위한 최적화 알고리즘을 작동시킨다. 최적화 알고리즘은 다양한 프로세스 평가자들에 의하여 프로세스 스코어링 모듈(38)에 보고된 프로세스 평가 표시자/스코어들을 포함하여, 다수의 호스트 시스템(10a-c)들에 실행되는 여러 프로세스에 대한 통계적 데이터 및/또는 행동 데이터를 수신할 수 있고, 파라미터들을 위한 최적값을 결정할 수 있다. 다음으로 이들 최적값들은 네트워크(26)를 통해서 각 호스트 시스템으로 전송된다. 일부 실시예에서, 최적 파라미터 값을 결정하기 위하여, 서버(110)는 깨끗한(멀웨어에 영향 받지 않은) 것으로 알려진 프로세스 세트를 이용하여 프로세스 스코어링 모듈(38) 및/또는 프로세스 평가자(50a-c)의 작업을 보정할 수 있다. 예시적인 보정 시나리오에서, 보안 서버(110)는 호스트 시스템(10)에 깨끗한 것으로 알려진 보정 프로세스 세트를 실행하도록 명령하고, 보정 프로세스에 대해서 결정된 프로세스 평가 표시자/스코어들의 세트를 서버(110)에 다시 보내도록 명령할 수 있다. 그리고 나서 서버(110)는 각 가상 머신 및/또는 호스트 시스템에 조정(tailoring)된 파라미터 값을 결정할 수 있다.

[0058] 다른 예에서, 보안 설정(82)은 다양한 프로세스 평가자들로부터 수신한 개별 프로세스 평가 표시자들로부터 피평가 프로세스에 대한 총합 악성 스코어를 결정하기 위하여 프로세스 스코어링 모듈(38)에 의해서 사용되는 가중값의 세트를 포함한다. 상기 총합 스코어가 개별 스코어들의 가중 합계 또는 가중 평균이고 각 스코어가 특징적 멀웨어 탐지 기준 또는 방법(예를 들어서, 각 스코어가 피평가 프로세스가 특정 멀웨어를 나타내는 행동을 수행하는 것을 표시할 때)에 따라서 컴퓨팅 되는 실시예에서, 개별 스코어의 가중(weight)을 변경하는 것이 다른 기준/방법과 비교해서 각 기준 또는 방법의 관련성을 효과적으로 변경할 수 있다. 멀웨어 위협은 통상적으로 연달아 발생하고, 굉장히 많은 전세계의 컴퓨터 시스템이 짧은 시간 간격에서 동일한 멀웨어 에이전트에 영향받는다. 다수의 호스트 시스템으로부터 실시간으로 보안 리포트(80)를 수신함으로써, 보안 서버(110)는 현재의 멀웨어 위협에 대하여 최신의 상태가 될 수 있고, 최적의 보안 설정(82), 예를 들어서 현재의 멀웨어 위협을 탐지하는 데 최적화된 스코어 가중치의 세트를 포함하는 설정(82)을 각 호스트 시스템에 즉각적으로 전달할 수 있다.

[0059] 상술한 예시적 시스템과 방법은 바이러스와 루트킷과 같은 멀웨어로부터 컴퓨터 시스템과 같은 호스트 시스템을 보호할 수 있게 한다. 종래의 안티-멀웨어 시스템들은 통상적으로 운영 시스템의 프로세서 권한 레벨(예를 들어서 커널 모드)에서 실행된다. 루트킷과 같은 일부 멀웨어는 운영 시스템의 상기 레벨에서 또한 작동할 수 있고, 따라서 종래의 안티-멀웨어 시스템을 무력화할 수 있으며 컴퓨터 시스템의 보안을 훼손할 수 있다. 반대로, 본 발명의 일부 실시예에서, 하이퍼바이저는 최고 프로세서 권한 레벨의 컴퓨터 시스템에서 실행되고 운영 시스템을 가상 머신으로 이동(displaying)한다. 본 발명의 일부 실시예에 따른 안티-멀웨어 시스템은 하이퍼바이저 레벨에서 VM 내에서 실행되는 요소와 VM 밖에서 실행되는 요소를 포함한다. 일부 안티-멀웨어 작용은 따라서 운영 시스템의 레벨보다 높은 프로세서 권한 레벨로부터 수행될 수 있으며, 이 때 안티-멀웨어 작용은 VM 내에서 실행되는 멀웨어에 의해서 와해될 수 없다. 일부 실시예에서, 하이버바이저의 레벨에서 실행되는 단일의 메모리 인트로스펙션 엔진이 각 컴퓨터 시스템에서 동시에 실행되는 복수의 가상 머신들을 보호할 수 있다.

[0060] 일부 실시예에서, 메모리 인트로스펙션 엔진의 작용은 그 중에서, 특정 드라이버, 라이브러리, 레지스터, 및 페이지 테이블과 같은 중요한 소프프웨어 객체들을 선택하는 것과, 그러한 객체에 대한 악성적 변화를 방지하는 것을 포함한다. 특히, 일부 실시예들은 따라서 VM 내에서 실행되는 안티-멀웨어 요소들을 악성 공격으로부터 보

호한다.

[0061] 그러한 객체들을 보호하기 위하여, 일부 실시예들은 각 객체에 할당된 메모리 공간에 쓰려는 시도를 중간차단(intercepting)하고, 그러한 시도를 차단(blocking) 또는 우회시켜서 악성적 변화를 방지할 수 있다. 다른 실시예에서는 각 객체에 할당된 메모리 공간에 읽기 전용이라고 표시함으로써 타겟 객체를 보호할 수 있다. 전형적인 하드웨어와 소프트웨어 구성에서, 메모리는 페이지로 알려진 연속 주소(contiguous address)의 개별 블록으로 파티션된다. 가상화를 지원하는 시스템들에서, 페이지 접근 허가는 하이퍼바이저에 의해서, 예를 들어서 확장된 페이지 테이블(EPT)과 같은 전용 데이터 구조(dedicated data structure)를 사용하여 제어된다. 따라서, 타겟 객체의 메모리 공간을 보호하는 것은 예를 들어서 메모리 인트로스펙션 엔진이 하이퍼바이저에게 각 객체에 속하는 데이터를 포함하는 페이지 세트를 읽기 전용으로 표시하도록 명령함으로써 달성될 수 있다.

[0062] 일부 실시예에서, 일부 안티-멀웨어 요소는 피보호 가상 머신 내에서 실행되고 메모리 인트로스펙션 엔진과 협력해서 멀웨어를 탐지한다. 그러한 구성은 가상화를 통해서 나타나는 의미 간격(semantic gap)을 연결함으로써 멀웨어 탐지를 실질적으로 단순화할 수 있다. 통상의 소프트웨어 구성에서, 사용자 모드에서 실행되는 멀웨어 탐지 요소는 피평가 프로세스의 행동에 대한 많은 정보에 접근할 수 있고, 반면에 이러한 정보의 대부분은 커널 레벨 또는 각 VM의 밖에서 실행되는 요소들이 쉽게 이용할 수 있는 것이 아니다. 예를 들어서, 피평가 프로세스가 인터넷으로부터 파일을 다운로드하려고 시도할 때, 사용자 모드 프로세스 평가자는 예를 들어서 DLL 주입과 같은 본 기술분야에서 알려진 방법을 사용하여 어떠한 프로세스가 그러한 동작을 수행하는지 식별할 수 있고, 피평가 프로세스가 파일을 다운로드 하려고 시도하는 것을 탐지할 수 있고, 파일이 다운로드 되는 IP 주소와, 다운로드 되는 파일의 디스크 위치 등을 결정할 수 있다. 한편, 하이퍼바이저의 레벨에서 실행되는 프로세스 평가자는 네트워크 패킷들 세트가 호스트 시스템의 네트워크 어댑터 상에서 순환하는 것을 탐지만 할 수 있다. 피평가 프로세서의 행동에 대한 정보를 하이퍼바이저의 레벨로부터 복구하는 것이 원칙적으로 가능은 하지만, 그러한 작업은 상당한 컴퓨팅 코스트(computational cost)를 수반할 수 있기 때문에 멀웨어 탐지에 현실적이지 않다. 본 발명의 일부 실시예에서는 각 VM 내에서 실행되는 안티-멀웨어 요소를 VM 밖에서 실행되는 메모리 인트로스펙션 엔진과 결합함으로써 VM 내부 요소(inside-VM component)들이 접근권을 가지는 충분한 행동 데이터를 사용할 수 있고, 한편으로는 그러한 요소의 완전성(integrity)을 각 VM의 밖으로부터 보호할 수 있다.

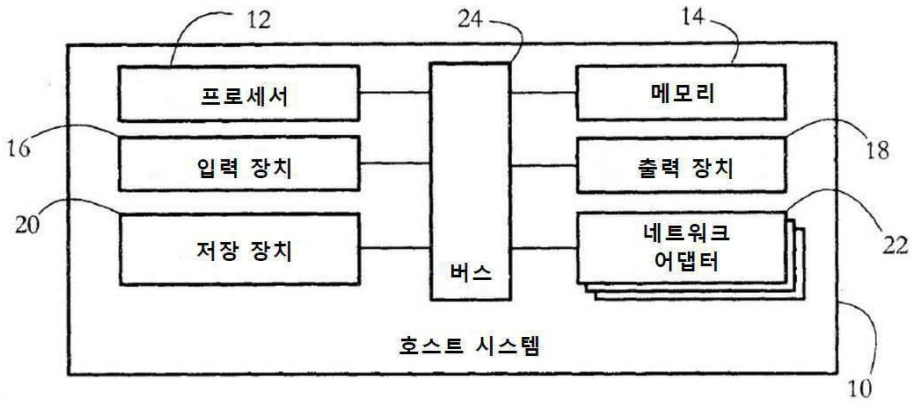
[0063] 종래의 안티-멀웨어 시스템에서, 운영 시스템의 레벨과 비슷한 프로세서 권한 레벨에서 실행되는 소프트웨어 요소는 프로세스가 개시될 때를 탐지하고 다른 안티-멀웨어 요소가 각 프로세스의 행동을 모니터링하도록 명령한다. 일부 멀웨어 에이전트들은 프로세스 개시를 탐지하는 소프트웨어 요소를 불능화함으로써 그러한 안티-멀웨어 시스템을 훼손하도록 작동하고 따라서 안티-멀웨어 시스템이 단지 현재 실행중인 프로세스의 서브세트만을 모니터링하도록 야기한다. 반대로, 본 발명의 일부 실시예에서, 프로세스 개시를 탐지하는 요소는 운영 시스템보다 높은 프로세서 권한 레벨에서 각 가상 머신 밖으로 이동된다. 그러한 구성은 멀웨어가 안티-멀웨어 요소로부터 숨는 것을 방지할 수 있다.

[0064] 일부 실시예에서, 프로세스-스코어링 모듈은 각 VM 내 또는 밖에서 실행되는 다수의 프로세스 평가자로부터 프로세스 별 평가 표시자(per-process evaluation indicator)를 수신한다. 피보호 VM 내에서 실행되는 요소로부터 수신된 프로세스 평가 표시자들은, 예를 들어서 피평가 프로세스가 OS의 레지스트리 값을 수정하려는 시도나 또는 파일을 삭제하려는 시도와 같은 멀웨어 임을 나타내는 행동을 수행한 것을 표시할 수 있다. 각 VM의 밖에서 결정된 프로세스 평가 표시자들은 예를 들어서 피평가 프로세스가 보호된 메모리 섹션에 쓰려고 시도한다는 것을 표시할 수 있다. 프로세서 평가 표시자들은 각 프로세스의 악성 정도를 표시하는 수치로 된 스코어들을 포함할 수 있다. 일부 실시예에서, 프로세스 스코어링 모듈은 다양한 프로세스 평가자들로부터 수신된 다수의 프로세스 평가 표시자/스코어들에 따라서 총합 스코어를 결정하고 총합 스코어에 따라서 피평가 프로세스가 악성인지 여부를 결정한다.

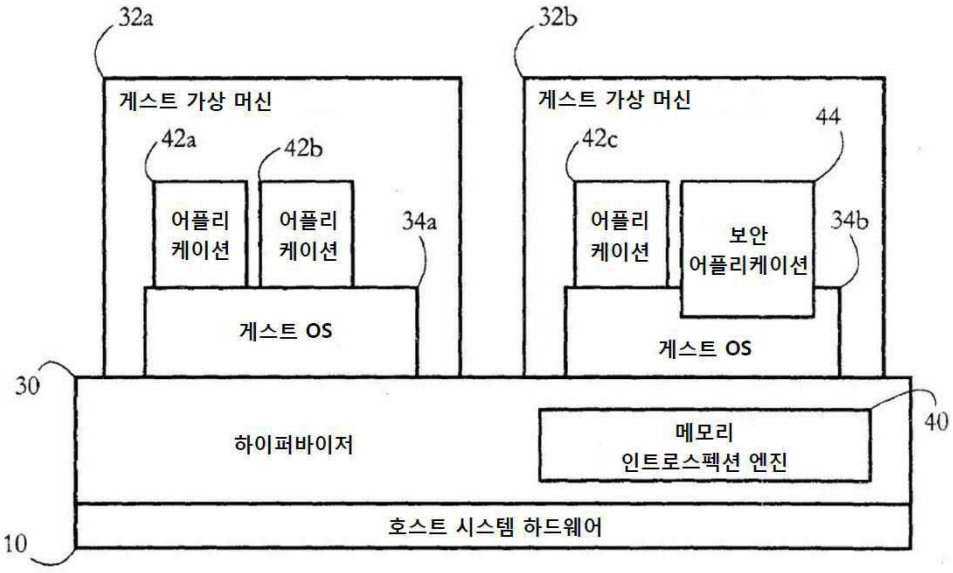
[0065] 상기의 실시예들이 본 발명의 범위를 벗어나지 않는다면 다양한 방법으로 변경될 수 있음은 통상의 기술자에게 당연한 것이다. 따라서 본 발명의 범위는 이하의 청구항과 그들의 법적 균등물에 의해서 결정되어야 한다.

도면

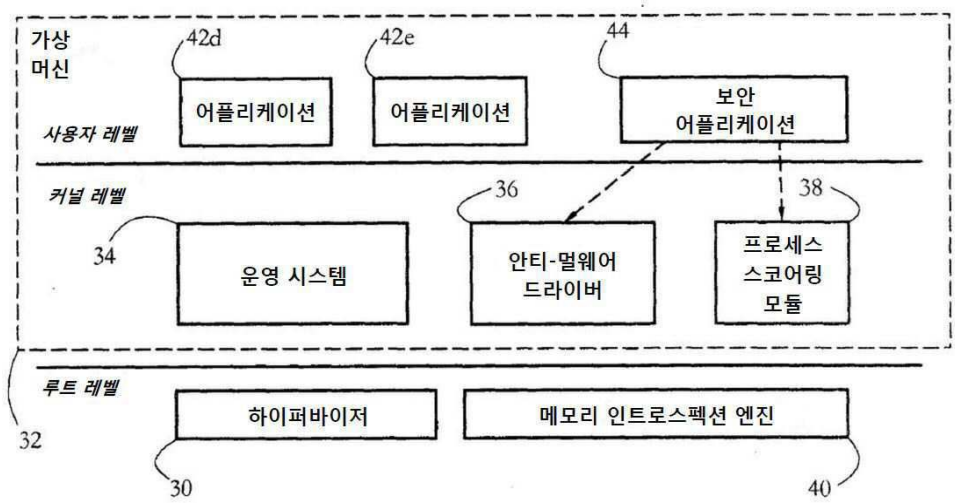
도면1


12
24
14
프로세서
메모리
16
입력 장치
출력 장치
18
20
저장 장치
버스
네트워크
어댑터
22
호스트 시스템
10

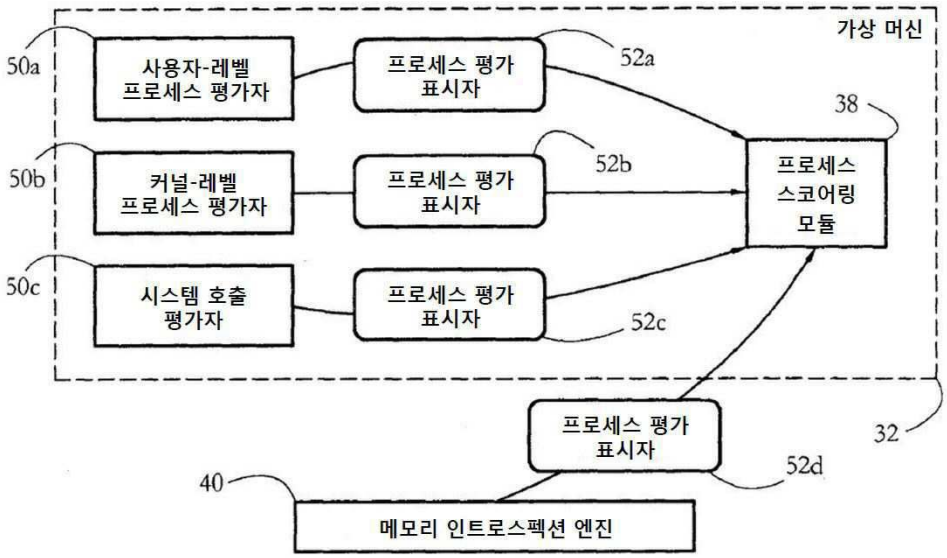
도면2


32a
32b
게스트 가상 머신
게스트 가상 머신
42a
42b
44
42c
어플리
케이션
어플리
케이션
34a
어플리
케이션
보안
어플리케이션
34b
게스트 OS
게스트 OS
30
하이퍼바이저
메모리
인트로스펙션 엔진
40
호스트 시스템 하드웨어
10

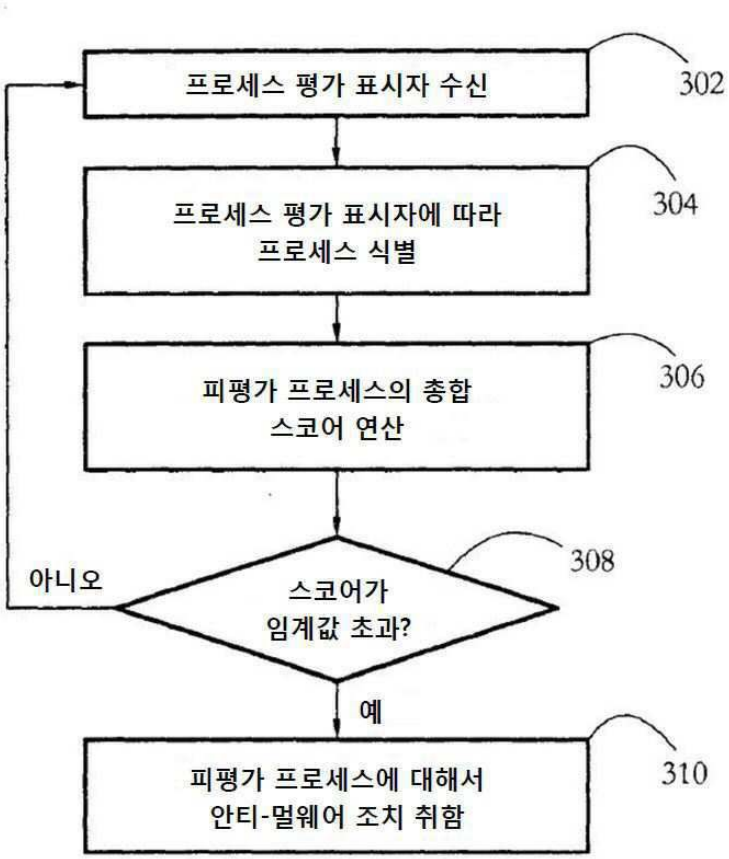
도면3


가상
머신
42d
42e
44
어플리케이션
어플리케이션
보안
어플리케이션
사용자 레벨
커널 레벨
36
38
34
운영 시스템
안티-멀웨어
드라이버
프로세스
스코어링
모듈
루트 레벨
32
하이퍼바이저
메모리 인트로스펙션 엔진
30
40

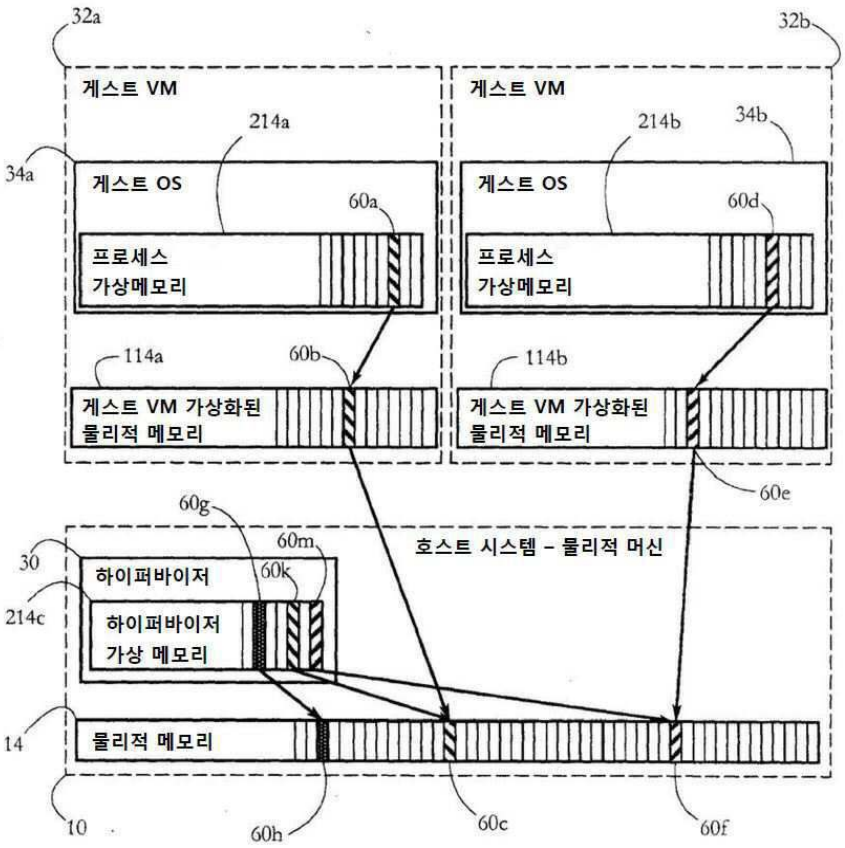
도면4


가상 머신
50a
사용자-레벨
프로세스 평가자
프로세스 평가
표시자
52a
38
50b
커널-레벨
프로세스 평가자
프로세스 평가
표시자
52b
프로세스
스코어링
모듈
50c
시스템 호출
평가자
프로세스 평가
표시자
52c
32
프로세스 평가
표시자
52d
40
메모리 인트로스펙션 엔진

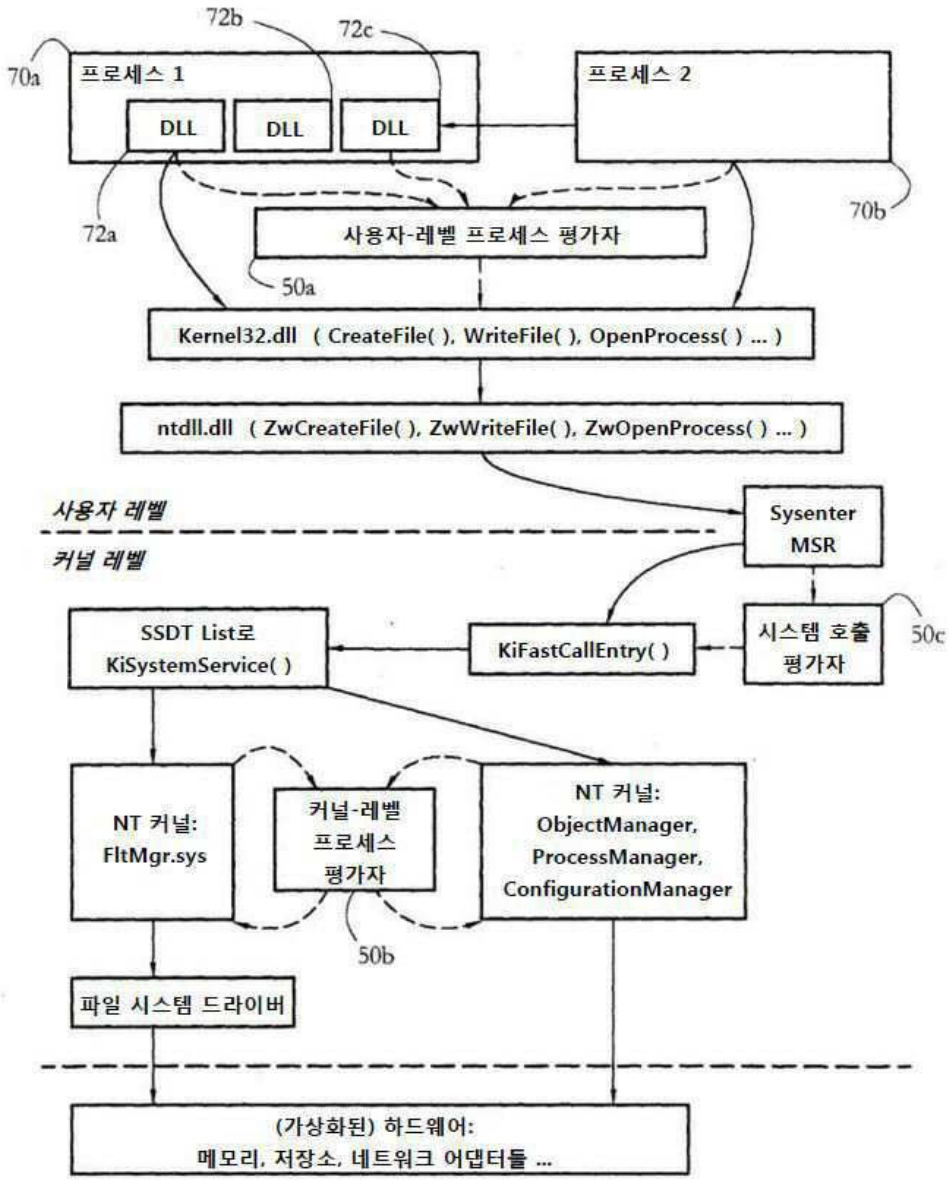
도면5


프로세스 평가 표시자 수신
302
프로세스 평가 표시자에 따라
프로세스 식별
304
피평가 프로세스의 총합
스코어 연산
306
스코어가
임계값 초과?
308
아니오
예
피평가 프로세스에 대해서
안티-멀웨어 조치 취함
310

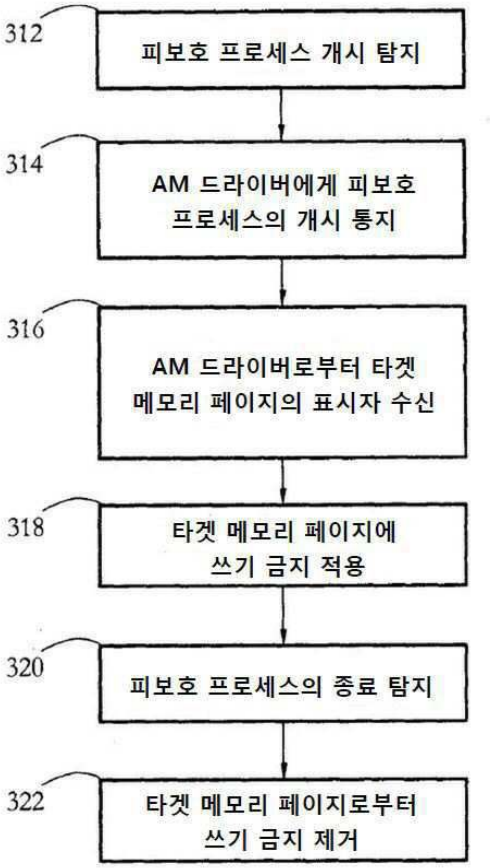
도면6


32a
32b
게스트 VM
게스트 VM
214a
214b
34b
34a
게스트 OS
게스트 OS
60a
60d
프로세스
가상메모리
프로세스
가상메모리
114a
60b
114b
게스트 VM 가상화된
물리적 메모리
게스트 VM 가상화된
물리적 메모리
60e
60g
60m
호스트 시스템 - 물리적 머신
30
하이퍼바이저
60k
214c
하이퍼바이저
가상 메모리
14
물리적 메모리
10
60h
60c
60f

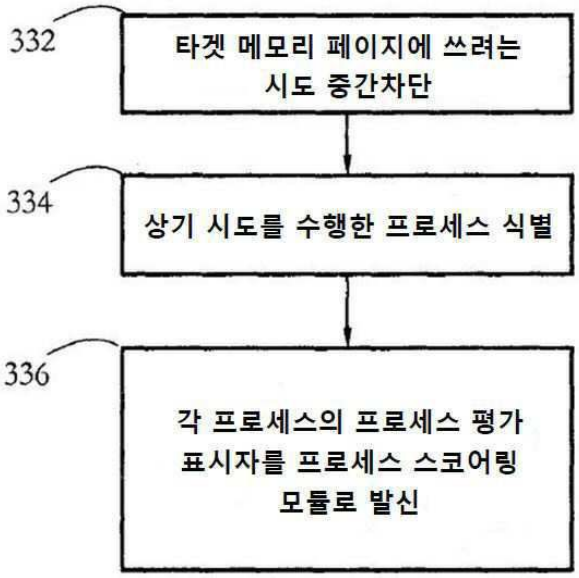
도면7


72b
72c
70a
프로세스 1
DLL
DLL
DLL
프로세스 2
70b
72a
사용자-레벨 프로세스 평가자
50a
Kernel32.dll (CreateFile( ), WriteFile( ), OpenProcess( ) ...)
ntdll.dll (ZwCreateFile( ), ZwWriteFile( ), ZwOpenProcess( ) ...)
사용자 레벨
커널 레벨
Sysenter
MSR
SSDT List로
KiSystemService( )
KiFastCallEntry( )
시스템 호출
평가자
50c
NT 커널:
FltMgr.sys
커널-레벨
프로세스
평가자
NT 커널:
ObjectManager,
ProcessManager,
ConfigurationManager
50b
파일 시스템 드라이버
(가상화된) 하드웨어:
메모리, 저장소, 네트워크 어댑터들 ...

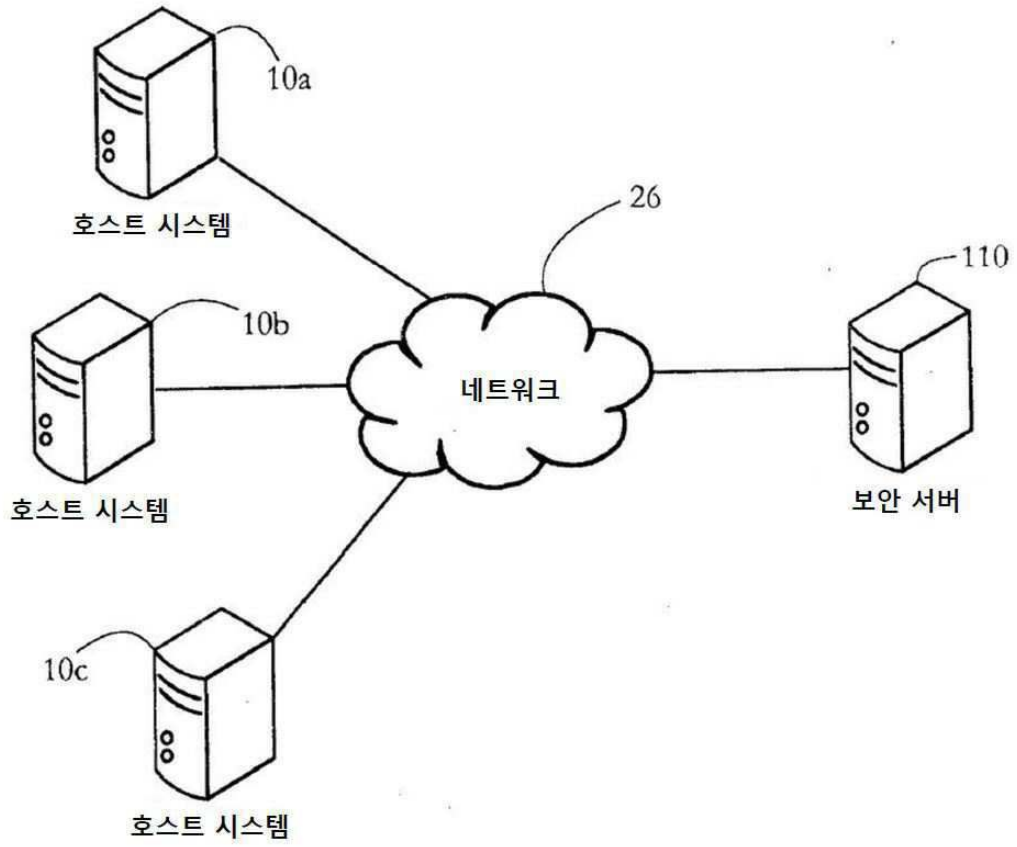
도면8


312
피보호 프로세스 개시 탐지
314
AM 드라이버에게 피보호 프로세스의 개시 통지
316
AM 드라이버로부터 타겟 메모리 페이지의 표시자 수신
318
타겟 메모리 페이지에 쓰기 금지 적용
320
피보호 프로세스의 종료 탐지
322
타겟 메모리 페이지로부터 쓰기 금지 제거

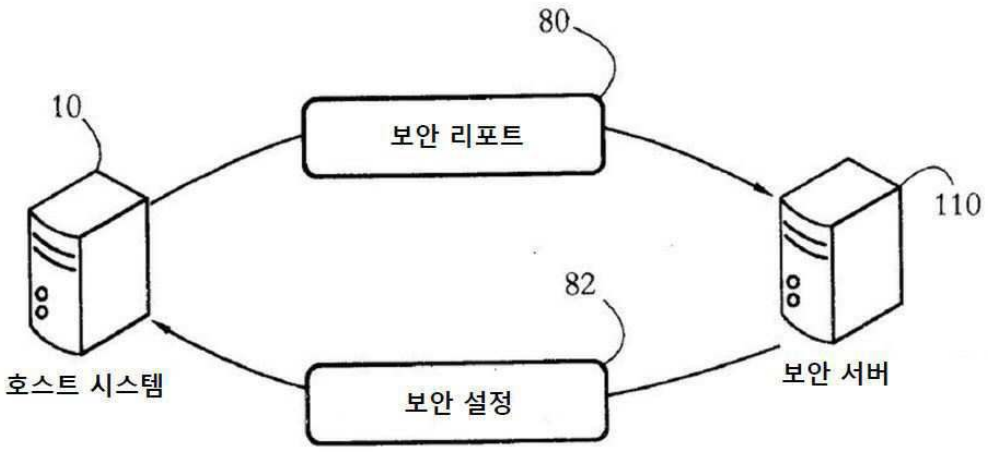
도면9


332
타겟 메모리 페이지에 쓰려는 시도 중간차단
334
상기 시도를 수행한 프로세스 식별
336
각 프로세스의 프로세스 평가 표시자를 프로세스 스코어링 모듈로 발신

도면10



도면11



【심사관 직권보정사항】

【직권보정 1】

【보정항목】 청구범위

【보정세부항목】 청구항 24

【변경전】

상기 피평가 프로세스

【변경후】

상기 프로세스

【직권보정 2】

【보정항목】 청구범위

【보정세부항목】 청구항 1

【변경전】

상기 호스트 시스템

【변경후】

호스트 시스템