(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년06월01일
(11) 등록번호 10-1626424
(24) 등록일자 2016년05월26일

(51) 국제특허분류(Int. Cl.)
G06F 21/53 (2013.01) *G06F 11/30* (2006.01)
G06F 9/06 (2006.01)
(21) 출원번호 10-2013-7025864
(22) 출원일자(국제) 2012년03월27일
심사청구일자 2013년09월30일
(85) 번역문제출일자 2013년09월30일
(65) 공개번호 10-2014-0033349
(43) 공개일자 2014년03월18일
(86) 국제출원번호 PCT/US2012/030702
(87) 국제공개번호 WO 2012/135192
국제공개일자 2012년10월04일
(30) 우선권주장
13/073,791 2011년03월28일 미국(US)
(뒷면에 계속)
(56) 선행기술조사문헌
KR1020070081362 A
KR1020090068833 A
US20090328195 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
맥아피 인코퍼레이티드
미국 95054 캘리포니아 산타클라라 미션컬리지 블러바드 2821
(72) 발명자
살람 아메드 사이드
미국 캘리포니아주 95014 쿠퍼티노 빅스버그 드라이브 10171
(74) 대리인
제일특허법인

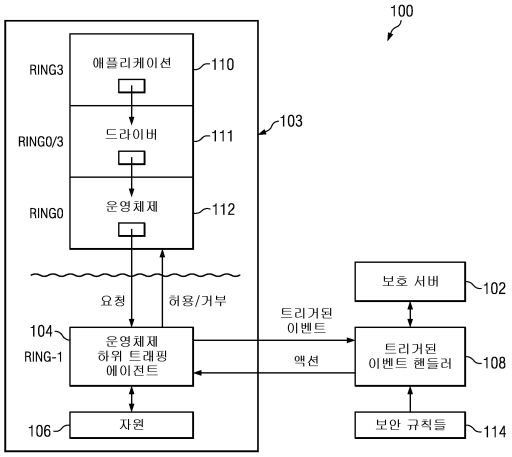
전체 청구항 수 : 총 23 항 심사관 : 구본재

(54) 발명의 명칭 **가상 머신 모니터 기반 안티 악성 소프트웨어 보안 시스템 및 방법**

(57) 요 약

전자 장치를 보호하기 위한 시스템은 메모리, 프로세서, 프로세서에 의해 실행하기 위해 메모리에 상주하는 하나 이상의 운영체제들, 운영체제와 통신 가능하게 결합된 전자 장치의 자원, 전자 장치 상에서 자원을 액세스하는 전자 장치의 전체 운영체제들 아래의 레벨에서 실행되도록 구성된 가상 머신 모니터, 및 전자 장치 상에서 자원을 액세스하는 전자 장치의 모든 운영체제들 아래의 레벨에서 실행되도록 구성된 보안 에이전트를 포함한다. 가상 머신 모니터는 가상 머신 모니터 위의 레벨에서 이루어진 자원 요청을 인터셉트하여 그 요청을 보안 에이전트에 알리도록 구성된다. 보안 에이전트는 요청이 악성 소프트웨어를 나타내는지 여부를 판단하도록 구성된다.

대 표 도 - 도1



(30) 우선권주장

13/073,810 2011년03월28일 미국(US)
13/073,842 2011년03월28일 미국(US)
13/073,853 2011년03월28일 미국(US)
13/073,864 2011년03월28일 미국(US)
13/074,741 2011년03월29일 미국(US)
13/074,831 2011년03월29일 미국(US)
13/074,925 2011년03월29일 미국(US)
13/074,947 2011년03월29일 미국(US)
13/075,049 2011년03월29일 미국(US)
13/075,072 2011년03월29일 미국(US)
13/075,101 2011년03월29일 미국(US)
13/076,473 2011년03월31일 미국(US)
13/076,480 2011년03월31일 미국(US)
13/076,493 2011년03월31일 미국(US)
13/076,512 2011년03월31일 미국(US)
13/076,537 2011년03월31일 미국(US)
13/077,227 2011년03월31일 미국(US)
13/077,270 2011년03월31일 미국(US)
13/077,305 2011년03월31일 미국(US)

명 세 서

청구범위

청구항 1

악성 소프트웨어로부터 전자 장치를 보호하기 위한 시스템으로서,

하드웨어 프로세서,

상기 프로세서에 통신가능하게 결합된 메모리,

운영 체제 - 상기 운영 체제는 상기 운영 체제 내의 드라이버를 로딩 및 언로딩함 -,

상기 프로세서에 의해 실행되기 위한 상기 메모리에 저장된 명령어를 포함하고 상기 운영 체제의 하나 이상의 자원에 대해 시도된 액세스를 트래핑하도록 구성되는 트래핑 에이전트 - 상기 시도된 액세스는 상기 운영 체제 내의 상기 드라이버에 대해 시도된 로딩 및 언로딩을 포함하고, 상기 시도된 액세스는 상기 드라이버를 로딩 또는 언로딩하기 위한 시스템 함수에 관한 코드를 포함하는 메모리 페이지의 실행을 트래핑함으로써 트래핑됨 -, 및

상기 프로세서에 의해 실행되기 위한 상기 메모리에 저장된 명령어를 포함하는 트리거된 이벤트 핸들러

를 포함하되,

상기 트래핑 에이전트는 또한 상기 드라이버의 상기 로딩 또는 언로딩을 포함하는 상기 트래핑된 시도에 관한 정보를 상기 트리거된 이벤트 핸들러로 전송하고,

상기 트리거된 이벤트 핸들러는

상기 정보에 기초하여 하나 이상의 보안 규칙에 액세스하고,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하며,

상기 트래핑 에이전트로 평가를 전송하도록 구성되고,

상기 트래핑 에이전트는 또한

상기 평가가 상기 드라이버에 대해 시도된 로딩 또는 언로딩이 악성 소프트웨어를 나타내는 것을 포함하는 경우 교정 액션을 수행하고,

상기 평가가 상기 드라이버에 대해 시도된 로딩 또는 언로딩이 안전함을 나타내는 것을 포함하는 경우 상기 드라이버에 대해 시도된 로딩 또는 언로딩을 허용하도록 구성되며,

상기 트래핑 에이전트와 상기 트리거된 이벤트 핸들러는 또한 운영 체제를 사용하지 않고 상기 시스템의 프로세서 상에서 실행되는 것을 포함하여 상기 하나 이상의 자원에 액세스하는 상기 전자 장치의 모든 운영 체제들보다 하위 레벨에서 동작하는

전자 장치 보호 시스템.

청구항 2

제 1 항에 있어서,

상기 트래핑 에이전트는 또한 상기 드라이버를 로딩 또는 언로딩하기 위한 상기 시스템 함수의 서브 함수의 시도된 실행을 트래핑하도록 구성되는 - 상기 시스템 함수는 상기 운영 체제에 의해 제공됨 -

전자 장치 보호 시스템.

청구항 3

제 1 항에 있어서,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 것은 로딩될 드라이버의 아이디를 판별하고 평가하는 것을 포함하는

전자 장치 보호 시스템.

청구항 4

제 1 항에 있어서,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 것은 상기 드라이버를 로딩 또는 언로딩하도록 시도한 개체를 판별하고 평가하는 것을 포함하는

전자 장치 보호 시스템.

청구항 5

악성 소프트웨어로부터 전자 장치를 보호하기 위한 시스템으로서,

하드웨어 프로세서,

상기 프로세서에 통신가능하게 결합된 메모리,

운영 체제 - 상기 운영 체제는 상기 운영 체제 내의 드라이버를 로딩 및 언로딩함 -,

상기 프로세서에 의해 실행되기 위한 상기 메모리에 저장된 명령어를 포함하고 상기 운영 체제의 하나 이상의 자원에 대해 시도된 액세스를 트래핑하도록 구성되는 트래핑 에이전트 - 상기 시도된 액세스는 상기 운영 체제 내의 상기 드라이버에 대해 시도된 로딩 및 언로딩을 포함하고, 상기 시도된 액세스는 상기 드라이버를 로딩 또는 언로딩하기 위한 시스템 함수에 관한 코드를 포함하는 물리적 메모리 주소의 실행을 트래핑함으로써 트래핑됨 -, 및

상기 프로세서에 의해 실행되기 위한 상기 메모리에 저장된 명령어를 포함하는 트리거된 이벤트 핸들러

를 포함하되,

상기 트래핑 에이전트는 또한 상기 드라이버의 상기 로딩 또는 언로딩을 포함하는 상기 트래핑된 시도에 관한 정보를 상기 트리거된 이벤트 핸들러로 전송하고,

상기 트리거된 이벤트 핸들러는

상기 정보에 기초하여 하나 이상의 보안 규칙에 액세스하고,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하며,

상기 트래핑 에이전트로 평가를 전송하도록 구성되고,

상기 트래핑 에이전트는 또한

상기 평가가 상기 드라이버에 대해 시도된 로딩 또는 언로딩이 악성 소프트웨어를 나타내는 것을 포함하는 경우 교정 액션을 수행하고,

상기 평가가 상기 드라이버에 대해 시도된 로딩 또는 언로딩이 안전함을 나타내는 것을 포함하는 경우 상기 드라이버에 대해 시도된 로딩 또는 언로딩을 허용하도록 구성되며,

상기 트래핑 에이전트와 상기 트리거된 이벤트 핸들러는 또한 운영 체제를 사용하지 않고 상기 시스템의 프로세서에 액세스하는 것을 포함하여 상기 전자 장치의 모든 운영 체제들보다 하위 레벨에서 동작하는

전자 장치 보호 시스템.

청구항 6

제 5 항에 있어서,

상기 트래핑 에이전트는 또한 상기 드라이버를 로딩 또는 언로딩하기 위한 상기 시스템 함수의 서브 함수의 시도된 실행을 트래핑하도록 구성되는 - 상기 시스템 함수는 상기 운영 체제에 의해 제공됨 -

전자 장치 보호 시스템.

청구항 7

삭제

청구항 8

제 5 항에 있어서,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 것은 상기 드라이버를 로딩 또는 언로딩하도록 시도한 개체를 판별하고 평가하는 것을 포함하는

전자 장치 보호 시스템.

청구항 9

악성 소프트웨어로부터 전자 장치를 보호하기 위한 방법으로서,

운영 체제의 하나 이상의 자원에 대해 시도된 액세스를 트래핑하는 단계 - 상기 운영 체제는 드라이버를 로딩 및 언로딩하고, 상기 시도된 액세스는 상기 운영 체제 내의 드라이버에 대해 시도된 로딩 또는 언로딩을 포함하며, 상기 시도된 액세스는 상기 드라이버를 로딩 또는 언로딩하기 위한 시스템 함수에 관한 코드를 포함하는 메모리 페이지의 실행을 트래핑함으로써 트래핑됨 -,

상기 시도된 액세스에 기초하여 하나 이상의 보안 규칙에 액세스하는 단계,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 단계,

상기 평가가 상기 드라이버에 대해 시도된 로딩 또는 언로딩이 악성 소프트웨어를 나타내는 것을 포함하는 경우 교정 액션을 수행하는 단계, 및

상기 평가가 상기 드라이버에 대해 시도된 로딩 또는 언로딩이 안전함을 나타내는 것을 포함하는 경우 상기 드라이버에 대해 시도된 로딩 또는 언로딩을 허용하는 단계

를 포함하되,

상기 시도된 액세스를 트래핑하는 단계 및 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 단계는, 운영 체제를 사용하지 않고 상기 전자 장치의 프로세서에 액세스하는 것을 포함하여 상기 전자 장치의 모든 운영 체제들보다 하위 레벨에서 동작하는

전자 장치 보호 방법.

청구항 10

제 9 항에 있어서,

상기 시도된 액세스를 트래핑하는 단계는 상기 드라이버를 로딩 또는 언로딩하기 위한 상기 시스템 함수의 서브 함수의 시도된 실행을 트래핑하는 단계를 더 포함하는 - 상기 시스템 함수는 상기 운영 체제에 의해 제공됨 -

전자 장치 보호 방법.

청구항 11

제 9 항에 있어서,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 단계는 로딩될 드라이버의 아이디를 판별하고 평가하는 단계를 포함하는

전자 장치 보호 방법.

청구항 12

제 9 항에 있어서,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 단계는 상기 드라이버를 로딩 또는 언로딩하도록 시도한 개체를 판별하고 평가하는 단계를 포함하는

전자 장치 보호 방법.

청구항 13

악성 소프트웨어로부터 전자 장치를 보호하기 위한 방법으로서,

운영 체제의 하나 이상의 자원에 대해 시도된 액세스를 트래핑하는 단계 - 상기 운영 체제는 드라이버를 로딩 및 언로딩하고, 상기 시도된 액세스는 상기 운영 체제 내의 드라이버에 대해 시도된 로딩 또는 언로딩을 포함하며, 상기 시도된 액세스는 상기 드라이버를 로딩 또는 언로딩하기 위한 시스템 함수에 관한 코드를 포함하는 물리적 메모리 주소의 실행을 트래핑함으로써 트래핑됨 -,

상기 시도된 액세스에 기초하여 하나 이상의 보안 규칙에 액세스하는 단계,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 단계,

상기 평가가 상기 드라이버에 대해 시도된 로딩 또는 언로딩이 악성 소프트웨어를 나타내는 것을 포함하는 경우 교정 액션을 수행하는 단계, 및

상기 평가가 상기 드라이버에 대해 시도된 로딩 또는 언로딩이 안전함을 나타내는 것을 포함하는 경우 상기 드라이버에 대해 시도된 로딩 또는 언로딩을 허용하는 단계

를 포함하되,

상기 시도된 액세스를 트래핑하는 단계 및 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 단계는, 운영 체제를 사용하지 않고 상기 전자 장치의 프로세서에 액세스하는 것을 포함하여 상기 전자 장치의 모든 운영 체제들보다 하위 레벨에서 동작하는

전자 장치 보호 방법.

청구항 14

제 13 항에 있어서,

상기 시도된 액세스를 트래핑하는 단계는 상기 드라이버를 로딩 또는 언로딩하기 위한 상기 시스템 함수의 서브 함수의 시도된 실행을 트래핑하는 단계를 더 포함하는 - 상기 시스템 함수는 상기 운영 체제에 의해 제공됨 -

전자 장치 보호 방법.

청구항 15

제 13 항에 있어서,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 단계는 로딩될 드라이버의 아이디를 판별하고 평가하는 단계를 포함하는

전자 장치 보호 방법.

청구항 16

제 13 항에 있어서,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 단계는 상기 드라이버를 로딩 또는 언로딩하도록 시도한 개체를 판별하고 평가하는 단계를 포함하는

전자 장치 보호 방법.

청구항 17

제조 물품으로서,

컴퓨터 판독가능 매체와,

상기 컴퓨터 판독가능 매체 상에 저장되는 컴퓨터 실행가능 명령어를 포함하되,

상기 명령어는 프로세서에 의해 판독가능하고, 상기 명령어가 판독 및 실행될 경우 상기 프로세서로 하여금,

운영 체제의 하나 이상의 자원에 대해 시도된 액세스를 트래핑하게 하고 - 상기 운영 체제는 드라이버를 로딩 및 언로딩하고, 상기 시도된 액세스는 상기 운영 체제 내의 드라이버에 대해 시도된 로딩 또는 언로딩을 포함하며, 상기 시도된 액세스는 상기 드라이버를 로딩 또는 언로딩하기 위한 시스템 함수에 관한 코드를 포함하는 메모리 페이지의 실행을 트래핑함으로써 트래핑됨 -,

상기 시도된 액세스에 기초하여 하나 이상의 보안 규칙에 액세스하게 하며,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하게 하고,

상기 평가가 상기 드라이버에 대해 시도된 로딩 또는 언로딩이 악성 소프트웨어를 나타내는 것을 포함하는 경우 교정 액션을 수행하게 하며,

상기 평가가 상기 드라이버에 대해 시도된 로딩 또는 언로딩이 안전함을 나타내는 것을 포함하는 경우 상기 드라이버에 대해 시도된 로딩 또는 언로딩을 허용하게 하되,

상기 시도된 액세스를 트래핑하는 것 및 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 것은, 운영 체제를 사용하지 않고 전자 장치의 프로세서에 액세스하는 것을 포함하여 상기 전자 장치의 모든 운영 체제들보다 하위 레벨에서 동작하는 것인

제조 물품.

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

청구항 21

삭제

청구항 22

삭제

청구항 23

삭제

청구항 24

삭제

청구항 25

삭제

청구항 26

삭제

청구항 27

삭제

청구항 28

삭제

청구항 29

삭제

청구항 30

삭제

청구항 31

삭제

청구항 32

삭제

청구항 33

삭제

청구항 34

삭제

청구항 35

삭제

청구항 36

삭제

청구항 37

삭제

청구항 38

삭제

청구항 39

삭제

청구항 40

삭제

청구항 41

삭제

청구항 42

삭제

청구항 43

삭제

청구항 44

삭제

청구항 45

삭제

청구항 46

삭제

청구항 47

삭제

청구항 48

삭제

청구항 49

삭제

청구항 50

삭제

청구항 51

삭제

청구항 52

삭제

청구항 53

삭제

청구항 54

삭제

청구항 55

삭제

청구항 56

삭제

청구항 57

삭제

청구항 58

삭제

청구항 59

삭제

청구항 60

삭제

청구항 61

삭제

청구항 62

삭제

청구항 63

삭제

청구항 64

삭제

청구항 65

삭제

청구항 66

삭제

청구항 67

삭제

청구항 68

삭제

청구항 69

삭제

청구항 70

삭제

청구항 71

삭제

청구항 72

삭제

청구항 73

삭제

청구항 74

삭제

청구항 75

삭제

청구항 76

삭제

청구항 77

삭제

청구항 78

삭제

청구항 79

삭제

청구항 80

삭제

청구항 81

삭제

청구항 82

삭제

청구항 83

삭제

청구항 84

삭제

청구항 85

삭제

청구항 86

삭제

청구항 87

삭제

청구항 88

삭제

청구항 89

삭제

청구항 90

삭제

청구항 91

삭제

청구항 92

삭제

청구항 93

삭제

청구항 94

삭제

청구항 95

삭제

청구항 96

삭제

청구항 97

삭제

청구항 98

삭제

청구항 99

삭제

청구항 100

삭제

청구항 101

삭제

청구항 102

삭제

청구항 103

삭제

청구항 104

삭제

청구항 105

삭제

청구항 106

삭제

청구항 107

삭제

청구항 108

삭제

청구항 109

삭제

청구항 110

삭제

청구항 111

삭제

청구항 112

삭제

청구항 113

삭제

청구항 114

삭제

청구항 115

삭제

청구항 116

삭제

청구항 117

삭제

청구항 118

삭제

청구항 119

삭제

청구항 120

삭제

청구항 121

삭제

청구항 122

삭제

청구항 123

삭제

청구항 124

삭제

청구항 125

삭제

청구항 126

삭제

청구항 127

삭제

청구항 128

삭제

청구항 129

삭제

청구항 130

삭제

청구항 131

삭제

청구항 132

삭제

청구항 133

삭제

청구항 134

삭제

청구항 135

삭제

청구항 136

삭제

청구항 137

삭제

청구항 138

삭제

청구항 139

삭제

청구항 140

삭제

청구항 141

삭제

청구항 142

삭제

청구항 143

삭제

청구항 144

삭제

청구항 145

삭제

청구항 146

제 5 항에 있어서,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 것은 로딩될 드라이버의 아이디를 판별하고 평가하는 것을 포함하는

전자 장치 보호 시스템.

청구항 147

제 17 항에 있어서,

상기 시도된 액세스를 트래핑하는 것은 상기 드라이버를 로딩 또는 언로딩하기 위한 상기 시스템 함수의 서브 함수의 시도된 실행을 트래핑하는 것을 더 포함하는 - 상기 시스템 함수는 상기 운영 체제에 의해 제공됨 -

제조 물품.

청구항 148

제 17 항에 있어서,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 것은 로딩될 드라이버의 아이디를 판별하고 평가하는 것을 포함하는

제조 물품.

청구항 149

제 17 항에 있어서,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 것은 상기 드라이버를 로딩 또는 언로딩하도록 시도한 개체를 판별하고 평가하는 것을 포함하는

제조 물품.

청구항 150

제조 물품으로서,

컴퓨터 판독가능 매체와,

상기 컴퓨터 판독가능 매체 상에 저장되는 컴퓨터 실행가능 명령어를 포함하되,

상기 명령어는 프로세서에 의해 판독가능하고, 상기 명령어가 판독 및 실행될 경우 상기 프로세서로 하여금,

운영 체제의 하나 이상의 자원에 대해 시도된 액세스를 트래핑하게 하고 - 상기 운영 체제는 드라이버를 로딩 및 언로딩하고, 상기 시도된 액세스는 상기 운영 체제 내의 드라이버에 대해 시도된 로딩 또는 언로딩을 포함하며, 상기 시도된 액세스는 상기 드라이버를 로딩 또는 언로딩하기 위한 시스템 함수에 관한 코드를 포함하는 물리적 메모리 주소의 실행을 트래핑함으로써 트래핑됨 -,

상기 시도된 액세스에 기초하여 하나 이상의 보안 규칙에 액세스하게 하며,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하게 하고,

상기 평가가 상기 드라이버에 대해 시도된 로딩 또는 언로딩이 악성 소프트웨어를 나타내는 것을 포함하는 경우 교정 액션을 수행하게 하며,

상기 평가가 상기 드라이버에 대해 시도된 로딩 또는 언로딩이 안전함을 나타내는 것을 포함하는 경우 상기 드라이버에 대해 시도된 로딩 또는 언로딩을 허용하게 하되,

상기 시도된 액세스를 트래핑하는 것 및 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 것은, 운영 체제를 사용하지 않고 전자 장치의 프로세서에 액세스하는 것을 포함하여 상기 전자 장치의 모든 운영 체제들보다 하위 레벨에서 동작하는 것인

제조 물품.

청구항 151

제 150 항에 있어서,

상기 시도된 액세스를 트래핑하는 것은 상기 드라이버를 로딩 또는 언로딩하기 위한 상기 시스템 함수의 서브 함수의 시도된 실행을 트래핑하는 것을 더 포함하는 - 상기 시스템 함수는 상기 운영 체제에 의해 제공됨 -

제조 물품.

청구항 152

제 150 항에 있어서,

상기 보안 규칙을 고려하여 상기 드라이버에 대해 상기 시도된 로딩 또는 언로딩을 평가하는 것은 로딩될 드라이버의 아이디를 판별하고 평가하는 것을 포함하는

제조 물품.

발명의 설명

기 술 분 야

[0001] 우선권 출원

[0002] 본 출원은 2011년 3월 28일 출원된 미국출원 제13/073,791, 2011년 3월 28일 출원된 미국출원 제13/073,810, 2011년 3월 28일 출원된 미국출원 제13/073,842, 2011년 3월 31일 출원된 미국출원 제13/077,227, 2011년 3월 28일 출원된 미국출원 제13/073,853, 2011년 3월 29일 출원된 미국출원 제13/075,049, 2011년 3월 31일 출원된 미국출원 제13/076,493, 2011년 3월 29일 출원된 미국출원 제13/074,741, 2011년 3월 31일 출원된 미국출원 제13/077,305, 2011년 3월 29일 출원된 미국출원 제13/074,831, 2011년 3월 29일 출원된 미국출원 제13/074,925, 2011년 3월 29일 출원된 미국출원 제13/074,947, 2011년 3월 31일 출원된 미국출원 제13/077,270, 2011년 3월 31일 출원된 미국출원 제13/076,537, 2011년 3월 28일 출원된 미국출원 제13/073,864, 2011년 3월 29일 출원된 미국출원 제13/075,072, 2011년 3월 29일 출원된 미국출원 제13/075,101, 2011년 3월 31일 출원된 미국출원 제13/076,512, 2011년 3월 31일 출원된 미국출원 제13/076,480, 및 2011년 3월 31일 출원된 미국출원 제13/076,473에 대한 우선권을 주장하며, 이들의 내용은 이로써 그 전체적으로 참조로서 통합된다.

[0003] 기술 분야

[0004] 본 발명은 일반적으로 컴퓨터 보안 및 악성 소프트웨어 방지에 관한 것으로서, 특히 가상 머신 모니터 기반의 안티 악성 소프트웨어 보안에 관한 것이다.

배 경 기 술

[0005] 네이티브 운영체제 서비스들은 보안 소프트웨어가 운영체제들의 커널 안에 임의의 후킹을 설치하는 것을 방지할 수 있다. 따라서 보안 소프트웨어가 악성 소프트웨어에 의한 악의적인 액션들을 잠재적으로 포함하는 전자 장치의 모든 행위들을 필터링하는 것이 금지된다. 악성 소프트웨어는 스파이웨어, 루트킷(rootkits), 패스워드 탈취, 스팸, 피싱 공격 소스, 도스(denial-of-service, 서비스 거부) 공격 소스, 바이러스, 로거(loggers), 트로이 목마(Trojans), 애드웨어, 또는 악의적 활동을 낳는 임의의 다른 디지털 콘텐츠를 포함할 수 있으나 그들에 국한되지 않는다.

[0006] 운영체제에 의해 제공되는 필터링 기능은 제한될 수 있으며 운영체제 판매자에 의해 결정된 일정에만 이용될 수 있다. 악성 소프트웨어는 특히 운영체제 커널에서 보안 소프트웨어와 동일한 수준으로 동작하고 상주할 수 있으며, 그에 따라 운영체제 및 보안 소프트웨어 자체의 무결성 모두를 손상시킬 수 있다.

[0007] 여러 형식의 공격적 커널 모드 악성 소프트웨어는 동적인 악성 코드 주입과 같은 악의적 작업들을 수행하기 위해 사용자 모드를 변조하여 실행 경로를 변경하고 악성코드로 방향을 돌리도록 사용자 모드 섹션들을 변경하며, 보안 소프트웨어를 무력화하기 위해 사용자 모드 데이터 구조들을 변경한다. 또한, 어떤 악성 소프트웨어는 검

출 로직을 속이기 위해 프로세스 메모리 코드 및 데이터 섹션들을 변조함으로써 커널로부터 안티 악성 소프트웨어 애플리케이션들과 프로세스들을 공격할 수 있다.

[0008] 커널 모드 루트킷과 기타 악성 소프트웨어는 사용자 모드 애플리케이션들과 커널 모드 장치 드라이버들로부터 자신들의 존재를 감추기 위한 다양한 방법들을 채용한다. 사용되는 기법들은 감염이 어디에서 발생하는지에 따라 달라질 수 있다. 예를 들어, 악성 소프트웨어는 루트킷이나 기타 악성 소프트웨어 프로세스를 삭제하거나 링크 해제하기 위해 운영체제의 커널 액티브 프로세스 리스트를 공격한다. 다른 악성 소프트웨어는 프로세스 액세스의 코드 섹션들과 열거 기능들을 변조할 수 있다.

발명의 내용

과제의 해결 수단

[0009] 일 실시예에서, 전자 장치를 보호하기 위한 시스템은 메모리, 프로세서, 상기 프로세서에 의해 실행하기 위해 상기 메모리에 상주하는 하나 이상의 운영체제들, 상기 운영체제와 통신 가능하게 결합된 상기 전자 장치의 자원, 상기 전자 장치 상에서 상기 자원을 액세스하는 상기 전자 장치의 상기 운영체제들 전체 아래의 레벨에서 실행되도록 구성된 가상 머신 모니터, 및 상기 전자 장치 상에서 상기 자원을 액세스하는 상기 전자 장치의 모든 운영체제들 아래의 레벨에서 실행되도록 구성된 보안 에이전트를 포함한다. 가상 머신 모니터는 가상 머신 모니터 위의 레벨에서 이루어진 자원 요청을 인터셉트하여 그 요청을 보안 에이전트에 알리도록 구성된다. 보안 에이전트는 요청이 악성 소프트웨어를 나타내는지 여부를 판단하도록 구성된다.

[0010] 다른 실시예에서, 전자 장치를 보호하기 위한 시스템은 메모리, 프로세서, 상기 프로세서에 의해 실행하기 위해 상기 메모리에 상주하는 하나 이상의 운영체제들, 상기 운영체제와 통신 가능하게 결합된 상기 전자 장치의 자원, 상기 전자 장치 상에서 상기 자원을 액세스하는 상기 전자 장치의 상기 운영체제들 전체 보다 높은 우선순위로 실행되도록 구성된 가상 머신 모니터, 및 상기 전자 장치 상에서 상기 자원을 액세스하는 상기 전자 장치의 모든 운영체제들 보다 높은 우선순위로 실행되도록 구성된 보안 에이전트를 포함한다. 상기 우선순위는 프로세서에 의해 정의된다. 가상 머신 모니터는 가상 머신 모니터 보다 낮은 우선순위를 가진 개체에서 이루어진 자원 요청을 인터셉트하여 그 요청을 보안 에이전트에 알리도록 구성된다. 보안 에이전트는 요청이 악성 소프트웨어를 나타내는지 여부를 판단하도록 구성된다.

[0011] 또 다른 실시예에서, 전자 장치를 보호하기 위한 시스템은 메모리, 프로세서, 상기 프로세서에 의해 실행하기 위해 상기 메모리에 상주하는 하나 이상의 운영체제들, 상기 운영체제와 결합된 상기 전자 장치의 자원, 상기 전자 장치 상에서 상기 자원을 액세스하는 상기 전자 장치의 상기 운영체제들 전체 보다 많은 특권을 가진 실행 링 위에서 실행되도록 구성된 가상 머신 모니터, 및 상기 전자 장치의 모든 운영체제들 보다 많은 특권을 가진 실행 링 위에서 실행되도록 구성된 보안 에이전트를 포함한다. 가상 머신 모니터는 가상 머신 모니터보다 적은 특권을 가진 실행 링에서 이루어진 자원 요청을 인터셉트하여 그 요청을 보안 에이전트에 알리도록 구성된다. 보안 에이전트는 요청이 악성 소프트웨어를 나타내는지 여부를 판단하도록 구성된다.

[0012] 또 다른 실시예에서, 전자 장치를 보호하는 방법은 자원을 액세스하는 전자 장치의 운영체제들 전체의 아래 레벨에서, 상위 레벨에서 이루어진 상기 전자 장치의 자원 요청을 인터셉트하는 단계 및 상기 요청이 악성 소프트웨어를 나타내는지 여부를 판단하는 단계를 포함한다. 상기 자원은 상기 운영체제와 통신 가능하게 결합된다.

[0013] 또 다른 실시예에서, 전자 장치를 보호하는 방법은 자원을 액세스하는 전자 장치의 운영체제들 전체 보다 높은 우선순위로, 보다 낮은 우선순위를 가진 개체에서 이루어진 자원 요청을 인터셉트하는 단계 및 상기 요청이 악성 소프트웨어를 나타내는지 여부를 판단하는 단계를 포함한다. 그러한 우선순위는 상기 전자장치의 프로세서에 의해 정의된다.

[0014] 또 다른 실시예에서, 전자 장치를 보호하는 방법은 자원을 액세스하는 전자 장치의 운영체제들 전체 보다 더 많은 특권을 가진 실행 링 상에서, 자원 요청을 인터셉트하는 단계 및 상기 요청이 악성 소프트웨어를 나타내는지 여부를 판단하는 단계를 포함한다. 상기 요청은 보다 적은 특권을 가진 실행 링에서 이루어진다.

[0015] 또 다른 실시예에서, 제조 물품이 컴퓨터 판독 가능 매체 및 상기 컴퓨터 판독 가능 매체 상에 포함되는 컴퓨터 실행 가능 명령어들을 포함한다. 상기 명령어들은 프로세서에 의해 판독 가능하다. 상기 명령어들은 판독되어

실행될 때, 상기 프로세서가 자원을 액세스하는 전자 장치의 운영체제 전체의 아래 레벨에서, 보다 상위 레벨에서 이루어진 상기 전자 장치의 자원 요청을 인터셉트하고 상기 요청이 악성 소프트웨어를 나타내는지 여부를 판단하게 한다. 상기 자원은 상기 운영체제와 통신 가능하게 결합된다.

[0016] 또 다른 실시예에서, 제조 물품이 컴퓨터 판독 가능 매체 및 상기 컴퓨터 판독 가능 매체 상에 포함되는 컴퓨터 실행 가능 명령어들을 포함한다. 상기 명령어들은 프로세서에 의해 판독 가능하다. 상기 명령어들은 판독되어 실행될 때, 상기 프로세서가 자원을 액세스하는 전자 장치의 운영체제들 전체 보다 높은 우선순위로, 보다 낮은 우선순위를 가진 개체에서 이루어진 자원 요청을 인터셉트하고 상기 요청이 악성 소프트웨어를 나타내는지 여부를 판단하게 한다. 상기 우선순위는 프로세서에 의해 정의된다.

[0017] 추가 실시예에서, 제조 물품이 컴퓨터 판독 가능 매체 및 상기 컴퓨터 판독 가능 매체 상에 포함되는 컴퓨터 실행 가능 명령어들을 포함한다. 상기 명령어들은 프로세서에 의해 판독 가능하다. 상기 명령어들은 판독되어 실행될 때, 상기 프로세스가 자원을 액세스하는 전자 장치의 운영체제들 전체 보다 더 많은 특권을 가진 실행 링 상에서, 자원 요청을 인터셉트하고 상기 요청이 악성 소프트웨어를 나타내는지 여부를 판단하게 한다. 상기 요청은 보다 적은 특권을 가진 실행 링에서 이루어진다.

도면의 간단한 설명

[0018] 본 발명과 그에 따른 이점에 대한 보다 완전한 이해를 위해, 이제 첨부된 도면과 함께 해석되는 이하의 설명을 참조한다.

도 1은 악성 소프트웨어로부터 전자 장치를 보호하기 위한 시스템의 실시예이다.

도 2는 악성 소프트웨어로부터 전자 장치를 보호하기 위한 가상 머신 모니터 기반 및 보안 규칙 기반의 실행 가능 보안 솔루션을 위한 시스템의 실시예이다.

도 3은 악성 소프트웨어로부터 가상 머신 모니터 기반의 전자 장치 보호를 위한 방법의 실시예이다.

도 4는 악성 소프트웨어로부터 전자 장치를 보호하기 위한 펌웨어 기반 및 보안 규칙 기반 시스템의 실시예이다.

도 5는 악성 소프트웨어로부터 전자 장치를 보호하기 위한 펌웨어 기반 솔루션의 실시예에 대한 상세도이다.

도 6은 악성 소프트웨어로부터 펌웨어 기반의 전자 장치 보호를 위한 방법의 실시예이다.

도 7은 악성 소프트웨어로부터 전자 장치를 보호하기 위한 마이크로코드 기반 시스템의 실시예이다.

도 8은 악성 소프트웨어로부터 마이크로코드 기반의 전자 장치 보호를 위한 방법의 실시예이다.

도 9는 전자 장치 상의 보안에 민감한 프로세서 자원들에 대한 소프트웨어 액세스를 규제하는 시스템의 실시예이다.

도 10은 프로세서 자원 제어 구조의 실시예이다.

도 11은 전자 장치 상의 보안에 민감한 프로세서 자원들에 대한 소프트웨어 액세스를 규제하는 방법의 실시예이다.

도 12는 전자 장치 상의 운영체제 하위(below-operating system) 트래핑을 이용하여 메모리를 보호하기 위해 소프트웨어 액세스를 규제하는 시스템의 실시예이다.

도 13은 메모리 맵들의 실시예들을 도시한다.

도 14는 전자 장치 액세스 시도에 대해 운영체제 하위 트래핑을 이용하여 메모리를 보호하는 방법의 실시예이다.

도 15는 전자 장치의 운영체제 커널을 보호하는 시스템의 실시예이다.

도 16은 운영체제에 대한 안전한 액세스들 및 안전한 드라이버 구성요소들의 액세스 맵에 대한 실시예이다.

도 17은 도 16의 액세스 맵을 추가 예시하는 가상 메모리의 실시예이다.

도 18은 운영체제에 대한 안전한 액세스들 및 안전한 드라이버 구성요소들의 액세스 맵을 생성하는 시스템의 실

시예이다.

도 19는 전자 장치의 운영체제 커널을 보호하는 방법의 실시예이다.

도 21은 안전한 운영체제 실행 환경을 제공하기 위한 시스템 내 런칭 모듈의 실시예이다.

도 22는 운영체제를 안전하게 실행하기 위한 운영체제 실행 환경의 실시예이다.

도 23은 안전한 운영체제 실행 환경을 제공하기 위한 시스템이나 방법 안에서 사용할 디스크 매핑 비트맵의 실시예이다.

도 24는 안전한 운영체제 실행 환경을 개시하기 위한 방법의 실시예이다.

도 25는 운영체제를 안전하게 실행하기 위한 운영체제 실행 환경을 제공하는 방법의 실시예이다.

도 26은 무허가 액세스로부터 스토리지 장치를 보호하기 위한 시스템의 실시예이다.

도 27은 무허가 액세스로부터 스토리지 장치를 보호하기 위한 시스템 또는 방법과 함께 사용할 보안 규칙들의 실시예이다.

도 28은 무허가 액세스로부터 스토리지 장치를 보호하기 위한 방법의 실시예이다.

도 29는 애플리케이션과 입/출력 장치 간의 쓰기 액세스를 위한 입/출력 경로를 보호하는 시스템의 실시예이다.

도 30은 애플리케이션과 입/출력 장치 간의 쓰기 액세스를 위한 입/출력 경로를 보호하는 방법의 실시예이다.

도 31은 애플리케이션과 입/출력 장치 간의 읽기 액세스를 위한 입/출력 경로를 보호하는 시스템의 실시예이다.

도 32는 애플리케이션과 입/출력 장치 간의 읽기 액세스를 위한 입/출력 경로를 보호하는 방법의 실시예이다.

도 33은 전자 장치 상에서 숨겨진 프로세스들을 검출하여 복구하기 위한 시스템의 실시예이다.

도 34는 전자 장치 상에서 숨겨진 프로세스들을 검출하여 복구하기 위한 방법의 실시예이다.

도 35는 전자 장치 상에서 숨겨진 프로세스들을 검출하여 복구하기 위한 다른 시스템의 실시예이다.

도 36은 전자 장치 상에서 숨겨진 프로세스들을 검출하여 복구하기 위한 다른 방법의 실시예이다.

도 37은 전자 장치 상에서 숨겨진 프로세스들을 검출하여 복구하기 위한 또 다른 방법의 실시예이다.

도 38은 운영체제의 시스템 호출들에 대한 액세스를 보호하기 위한 시스템의 실시예이다.

도 39는 운영체제의 시스템 호출들에 대한 액세스를 보호하기 위한 시스템이나 방법과 함께 사용할 시스템 호출 테이블의 실시예이다.

도 40은 운영체제의 시스템 호출들에 대한 액세스를 보호하기 위한 방법의 실시예이다.

도 41은 전자 장치 상의 악성 코드 또는 잠재적 악성 코드의 규제 및 통제를 위한 시스템의 실시예이다.

도 42는 전자 장치 상의 자가 변형 코드의 규제 및 통제를 위한 시스템의 실시예이다.

도 43은 전자 장치 상의 악성 코드 수정을 위한 방법의 실시예이다.

도 44는 전자 장치 상의 관련 스레드들에 대한 감시 및 추적 방법의 실시예이다.

도 45는 전자 장치의 메모리 및 스토리지를 보호하는 시스템의 실시예이다.

도 46은 전자 장치의 메모리 및 스토리지를 보호하는 방법의 실시예이다.

도 47은 운영체제의 객체들에 대한 액세스를 보호하기 위한 시스템의 실시예이다.

도 48은 운영체제의 객체들에 대한 액세스를 보호하는 시스템이나 방법과 함께 사용할 동향 상태 맵의 실시예이다.

도 49는 운영체제의 객체들에 대한 액세스를 보호하기 위한 방법의 실시예이다.

도 50은 전자 장치 상의 드라이버들 간 통신을 보호하기 위한 시스템의 실시예이다.

도 51은 드라이버 간 통신의 예시도이다.

도 52는 O/S 하위 보안 에이전트가 보호할 수 있는 전자 장치의 부분들의 예에 대한 추가도면이다.

도 53은 전자 장치에서 드라이버 간 통신에 대한 운영체제 하위 트래핑 및 보안을 위한 방법의 실시예이다.

도 54는 전자 장치 상의 드라이버 필터들의 첨부 및 분리를 보호하기 위한 시스템의 실시예이다.

도 55는 장치 스택 예의 동작에 대한 상세도이다.

도 56은 드라이버 필터들을 첨부하거나 분리하고자 동작하는 악성 소프트웨어에 의해 손상되었을 수 있는 장치 스택들의 예시도이다.

도 57은 전자 장치에서 드라이버 필터 첨부에 대한 운영체제 하위 트래핑을 위한 방법의 실시예이다.

도 58은 전자 장치 상의 드라이버들의 로딩 및 언로딩을 보호하기 위한 시스템의 실시예이다.

도 59a 및 59b는 전자 장치 상의 드라이버들의 로딩 및 언로딩을 보호하기 위한 방법의 실시예이다.

도 60은 메모리로의 코드 로딩에 대해 운영체제 하위 트래핑 및 보호를 위한 시스템의 실시예이다.

도 61은 삽입된 코드가 어떻게 애플리케이션에 의해 수집되어 실행을 위해 메모리 안에 위치하는지에 대한 예시도이다.

도 62a는 애플리케이션의 이미지를 디스크로부터 메모리로 로딩하는 것에 대한 예시도를 도시한다.

도 62b는 애플리케이션의 이미지가 메모리 안에 로딩된 후 수행되는 가능한 액션들에 대한 예시도를 도시한다.

도 63은 코드를 삽입하기 위해 스왑된 콘텐츠에 대해 이루어지는 악의적 공격들의 추가적 예를 도시한다.

도 64는 메모리의 일부가 악성이라고 판단된 뒤의 메모리 맵의 실시예이다.

도 65는 메모리 내 코드 로딩 및 실행에 대한 운영체제 하위 트래핑을 위한 시스템의 실시예이다.

발명을 실시하기 위한 구체적인 내용

[0019] 도 1은 악성 소프트웨어로부터 전자 장치를 보호하기 위한 시스템(100)의 예시적 실시예이다. 시스템(100)은 트리거된(triggered) 이벤트 핸들러(108)와 통신 가능하게 연결되는 운영체제 하위(below-operating system ("O/S") 트래핑 에이전트(104)를 포함할 수 있다. O/S 하위 트래핑 에이전트(104)는 전자 장치(103)의 자원(106)에 대해 시도되는 다양한 액세스들을 트래핑(trap)하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(104)는 트래핑된 액세스 시도와 관련된 트리거된 이벤트를 생성하여 그 트리거된 이벤트를 트리거된 이벤트 핸들러(108)로 보내도록 구성될 수 있다. 트리거된 이벤트 핸들러(108)는 그 트리거된 이벤트를 어떻게 다룰지를 결정하기 위해 하나 이상의 보안 규칙들(114)이나 보호 서버(102)를 조회하도록 구성될 수 있다. 트리거된 이벤트 핸들러(108)는 또한 트리거된 이벤트의 성향이 악성 소프트웨어에 대한 표시인지 아니면 전자 장치(103)의 자원들이나 동작을 전복하기 위한 악의적 시도인지를 평가하도록 구성될 수 있다. 게다가 트리거된 이벤트 핸들러(108)는 트리거된 이벤트가 허용되어야 할지 거부되어야 할지 여부에 대한 판단을 O/S 하위 트래핑 에이전트(104)로 제공하도록 구성되거나, 다른 교정 액션을 산출하도록 구성될 수 있다.

[0020] O/S 하위 트래핑 에이전트(104)는 전자 장치(103) 내 운영체제들보다 낮은 기능 레벨에서 구현될 수 있다. 예를 들어, O/S 하위 트래핑 에이전트(104)는 운영체제(112), 드라이버(111) 또는 애플리케이션(110)에 의한 자원(106)에 대한 액세스 시도를 인터셉트할 수 있다. O/S 하위 트래핑 에이전트(104)는 운영체제 이용 없이 전자 장치(103)의 프로세서 상에서 실행될 수 있다. 일 실시예에서 O/S 하위 트래핑 에이전트(104)는 베어 메탈(bare metal) 환경이나 실행 레벨에서 운영될 수 있다. 또한 O/S 하위 트래핑 에이전트(104)는 전자 장치(103)의 프로세서에 의해 정의될 때, 전자 장치(103)의 모든 운영체제들보다 높은 실행 우선순위로 실행될 수 있다. 예를 들어 보호 링(protection rings)을 이용하는 계층적 보호 도메인 모델과 관련하여 낮은 번호가 높은 우선순위를 나타낼 때, 운영체제(112)는 "Ring0"에서 동작하고 O/S 하위 트래핑 에이전트(104)는 "Ring -1"에서 동작할 수 있다. 드라이버들(111)과 애플리케이션들(110)은 "Ring0"나 "Ring3"에서 동작할 수 있다. 프로세서들의 일부 실시예들에서, "Ring-1"이라는 개념은 "Ring0 특권 모드"라 알려질 수 있고, "Ring0" 개념은 "Ring0 비특권 모드"라 알려질 수 있다. "Ring -1" 또는 "Ring0 특권 모드"에서의 동작은 "Ring0" 또는 "Ring0 비특권 모드"에 비해 추가적 오버헤드 및 비용을 수반할 수 있다. 전자 장치(103)의 운영체제들은

Ring0에서 실행될 수 있다.

[0021] O/S 하위 트래핑 에이전트(104)는 Ring0나 그 이상에서 실행되는 개체들에 대해 투명하게 동작될 수 있다. 그에 따라 O/S 하위 트래핑 에이전트(104)가 존재하는지 존재하지 않는지 여부와 관계없이, 자원(106)에 대해 시도된 액세스가 운영체제(112)나 다른 개체에 의해 동일한 방식으로 요청될 수 있다. O/S 하위 트래핑 에이전트(104)는 수신된 액션을 시행할 때, 상기 요청이 일어나게 하거나, 상기 요청을 거부하거나, 다른 교정 액션을 취할 수 있다. 요청을 거부하려면, O/S 하위 트래핑 에이전트(104)는 간단히 그 요청을 자원(106)이나 프로세서로 넘기지 않거나, 해당 액션이 일어났다는 것을 운영체제(112)에 납득시키기 위해 요청에 대한 스푸핑 또는 더미 응답을 제공할 수 있다.

[0022] 전자 장치(103)의 관련 운영체제들보다 높은 우선순위 또는 전자 장치(103)의 관련 운영체제들보다 아래에 있는 "Ring -1"에서 실행됨으로써, O/S 하위 트래핑 에이전트(104)는 운영체제(112)와 같은 운영체제들을 감염시키는 많은 악성 소프트웨어를 피할 수 있다. 악성 소프트웨어는 운영체제(112) 또는 심지어 "Ring0"에서 실행되는 안티 악성 소프트웨어 소프트웨어까지 속일 수 있는데, 이는 악성 소프트웨어 역시 "Ring0"의 우선순위로 실행될 것이기 때문이다. 그러나 전자 장치(103) 상의 악성 소프트웨어는 악의적 활동을 수행하고자 할 경우 계속 자원(106) 요청을 해야만 한다. 따라서, 민감한 자원들과 연결된 트래핑 동작들이 전자 장치(103) 내 운영체제들의 레벨 아래에서 실행되는 트래핑 에이전트에 의해 보다 잘 수행될 수 있다.

[0023] O/S 하위 트래핑 에이전트(104)는 모든 적절한 방식에 따라 구현될 수 있다. 일 실시예에서 O/S 하위 트래핑 에이전트(104)는 가상 머신 모니터 내에서 구현될 수 있다. 그러한 실시예는 O/S 하위 트래핑 에이전트(104)에 대해 기술된 바와 같이 운영체제들의 레벨 아래에서 동작될 수 있다. 그러한 실시예에 대한 설명은 예컨대 이하의 도 2에서의 가상 머신 모니터(216)에 대한 논의에서 발견될 수 있다. 다른 실시예에서 O/S 하위 트래핑 에이전트(104)는 펌웨어로 구현될 수 있다. 그러한 실시예는 O/S 하위 트래핑 에이전트(104)에 대해 기술된 바와 같이 운영체제들의 레벨 아래에서 동작될 수 있다. 그러한 실시예에 대한 설명은 예컨대 이하의 도 4 및 5에서의 펌웨어 보안 에이전트(440, 516) 또는 펌웨어 보안 에이전트(444)에 대한 논의에서 발견될 수 있다. 또 다른 실시예에서 O/S 하위 트래핑 에이전트(104)는 마이크로코드로 구현될 수 있다. 그러한 구현 예는 O/S 하위 트래핑 에이전트(104)에 대해 기술된 바와 같이 운영체제들의 레벨 아래에서 동작될 수 있다. 그러한 실시예에 대한 설명은 예컨대 이하의 도 7에서의 마이크로코드 보안 에이전트(708)에 대한 논의에서 발견될 수 있다. O/S 하위 트래핑 에이전트(104)는 그러한 실시예들의 조합을 통해 구현될 수 있다.

[0024] 트리거된 이벤트 핸들러(108)가 서로 통신 가능하게 연결된 하나 이상의 이벤트 핸들러들이나 보안 에이전트들에 의해 구현될 수 있다. 트리거된 이벤트 핸들러(108) 및 O/S 하위 트래핑 에이전트(104)는 동일한 보안 에이전트 안에서 구현될 수 있다. 일 실시예에서 트리거된 이벤트 핸들러(108)는 O/S 하위 트래핑 에이전트(104)와 동일한 우선순위 링에서 동작할 수 있다. 다른 실시예에서 트리거된 이벤트 핸들러(108)는 운영체제(112), 드라이버(111), 또는 애플리케이션(110)과 동일한 우선순위로 동작할 수 있다. 또 다른 실시예에서, 트리거된 이벤트 핸들러(108)는 둘 이상의 트리거된 이벤트 핸들러들에 의해 구현될 수 있으며, 이때 적어도 한 트리거된 이벤트 핸들러는 O/S 하위 트래핑 에이전트(104)와 동일한 우선순위 링에서 동작하며 적어도 하나의 트리거된 이벤트 핸들러는 운영체제(112), 드라이버(111) 또는 애플리케이션(110)의 레벨에서 동작한다. O/S 하위 트래핑 에이전트(104)의 레벨에서 동작함으로써, 트리거된 이벤트 핸들러(108)도 마찬가지로, 에이전트 자체를 감염시키는 "Ring0"나 "Ring3" 악성 소프트웨어의 문제를 피할 수 있다. 그러나 운영체제(112), 드라이버(111), 또는 애플리케이션(110)과 함께 "Ring0"나 "Ring3"에서 실행되는 트리거된 이벤트 핸들러(108)는 "Ring -1" 에이전트들의 관점에서 이용 불가능할 수 있는 자원(106)에 대해 시도된 액세스에 관한 정황 정보를 제공할 수 있다.

[0025] 트리거된 이벤트 핸들러(108)는 모든 적절한 방식에 따라 구현될 수 있다. 일 실시예에서 트리거된 이벤트 핸들러(108)는 가상 머신 모니터나 가상 머신 모니터 보안 에이전트에서 구현될 수 있다. 그러한 실시예는 트리거된 이벤트 핸들러(108)에 대해 기술된 바와 같이 운영체제들의 레벨 아래에서 동작될 수 있다. 그러한 실시예에 대한 설명은 예컨대 이하의 도 2에서의 가상 머신 모니터(216)나 보안 가상 머신 모니터(security virtual machine monitor) 보안 에이전트(217)에 대한 논의에서 발견될 수 있다. 다른 실시예에서 트리거된 이벤트 핸들러(108)는 전적으로나 부분적으로 펌웨어를 통해 구현될 수 있다. 그러한 실시예는 트리거된 이벤트 핸들러(108)에 대해 기술된 바와 같이 운영체제들의 레벨 아래에서 동작될 수 있다. 그러한 실시예에 대한 설명은 예컨대 이하의 도 4 및 5에서의 펌웨어 보안 에이전트(440, 516) 또는 펌웨어 보안 에이전트(444)에 대한 논의에서 발견될 수 있다. 트리거된 이벤트 핸들러(108)는 또한, 그 자체가 가상 머신 모니터에서와 같은 방식이나 펌웨어 또는 마이크로코드로 구현될 수 있는 도 4의 O/S 하위 에이전트(450) 안에서 구현될 수 있다. 또 다른

실시예에서 트리거된 이벤트 핸들러(108)는 마이크로코드로 구현될 수 있다. 그러한 구현 예는 트리거된 이벤트 핸들러(108)에 대해 기술된 바와 같이 운영체제들의 레벨 아래에서 동작될 수 있다. 그러한 실시예에 대한 설명은 예컨대 이하의 도 7에서의 마이크로코드 보안 에이전트(708)에 대한 논의에서 발견될 수 있다. 트리거된 이벤트 핸들러(108)는 또한, 그 자체가 가상 머신 모니터에서와 같은 방식이나 펌웨어 또는 마이크로코드로 구현될 수 있는 도 7의 O/S 하위 에이전트(712) 안에서 구현될 수 있다. 트리거된 이벤트 핸들러(108)는 그러한 실시예들의 조합을 통해 구현될 수 있다.

[0026] 일 실시예에서 운영체제(O/S) 하위 트래핑 에이전트(104) 및/또는 트리거된 이벤트 핸들러(108)는 전자 장치(103)의 베어 메탈 계층(bare metal layer)에서 운영될 수 있다. 운영체제 하위 트래핑 에이전트(104) 및/또는 트리거된 이벤트 핸들러(108)는 그들 사이에 운영체제 및 그들이 보호하도록 된 자원(106)을 사용하지 않고 운영될 수 있다. 자원(106)은 프로세서, 프로세서의 특성들, 메모리, 데이터 구조와 같이 메모리에 상주하는 개체들, 또는 함수, 프로세스, 또는 애플리케이션들과 같이 프로세서에 의해 실행되기 위해 메모리에 상주하는 개체들을 포함할 수 있다. 운영체제 하위 트래핑 에이전트(104) 및/또는 트리거된 이벤트 핸들러(108)는 전자 장치(103)의 하드웨어 상에서 바로 운영될 수 있다. 운영체제 하위 트래핑 에이전트(104) 및/또는 트리거된 이벤트 핸들러(108)는 실행할 운영체제(112)와 같은 운영체제 사용을 필요로 하지도, 자원(106)에 대한 풀 액세스를 획득하지도 않을 것이다.

[0027] 운영체제(112), 운영체제 하위 트래핑 에이전트(104) 및 트리거된 이벤트 핸들러(108) 및 자원(106)의 레벨에 있는 개체들 사이의 관계에 참여하지 않는 다른 운영체제들이 전자 장치(103) 상에 존재할 수 있다. 예를 들어 전치 부팅(pre-boot) 시스템은 전자 장치의 일부를 안전하게 시동할 수 있지만 애플리케이션(110), 드라이버(111), 및 운영체제(112)와 관련하여 자원(106)으로 이루어진 전자 장치의 일반 동작에 참여하지는 않을 것이다. 다른 예에서 전자 장치(103)는 마더보드 구성요소들, 플러그인 카드들, 주변기기들, 또는 운영체제(112), 운영체제 하위 트래핑 에이전트(104) 및 트리거된 이벤트 핸들러(108) 및 자원(106)의 레벨에 있는 개체들 사이의 관계 밖의 기능들을 수행할 자체적인 운영체제들 및 프로세서들의 세트들을 포함하는 다른 구성요소들을 포함할 수 있다. 이러한 운영체제들은 내장된 운영체제들일 수 있다. 이러한 운영체제들 중 어느 것도 운영체제 하위 트래핑 에이전트(104) 및 트리거된 이벤트 핸들러(108)의 실행에 사용되지 않을 수도 있다. 또한, 이러한 운영체제들 중 어느 것도 트래핑 에이전트(104) 및 트리거된 이벤트 핸들러(108)에 의해 보호되는 자원(106)을 액세스하지 않을 수도 있다.

[0028] 시스템(100)은 하나 이상의 운영체제 하위 트래핑 에이전트들(104) 및 하나 이상의 트리거된 이벤트 핸들러들(108)의 어떤 조합을 포함할 수 있다. 운영체제 하위 트래핑 에이전트들(104) 및 트리거된 이벤트 핸들러들(108)에 대한 설명은 이어지는 도면들 안의 트래핑 에이전트들, 이벤트 핸들러들 및 보안 에이전트들에서 발견될 수 있다.

[0029] 자원(106)은 전자 장치의 어떤 적절한 자원을 포함할 수 있다. 예를 들어 자원(106)은 레지스터들, 메모리, 제어기들, 또는 I/O 장치들을 포함할 수 있다. 자원(106)의 실시예들에 대한 설명은 예컨대 이하의 도 2의 시스템 자원들(214), 도 4에 도시된 바와 같은 디스플레이(430) 및 스토리지(432)와 같은 구성요소들, 도 7의 시스템 자원들(724)에서 발견될 수 있다.

[0030] 보안 규칙들(114)은 어떤 액션들을 트래핑할 것인지에 대해 O/S 하위 트래핑 에이전트(104)에 정보를 주거나 트래핑된 액션에 기반하여 이벤트를 처리하도록 트리거된 이벤트 핸들러(108)에 정보를 주기 위한 어떤 적절한 규칙들, 로직, 명령들, 명령어들, 플래그들, 또는 기타 메커니즘들을 포함할 수 있다. 트리거된 이벤트 핸들러(108)는 보안 규칙들(114) 중 하나 이상을 O/S 하위 트래핑 에이전트에 제공하도록 구성될 수 있다. 보안 규칙들(114) 중 일부나 전부에 대한 실시예들의 설명이 예컨대 이하의 도 2의 보안 규칙들(222), 도 4의 보안 규칙들(422, 434, 436, 438), 도 5의 보안 규칙들(518), 또는 도 7의 보안 규칙들(707, 723)에 대한 설명에서 보여질 수 있다.

[0031] 시스템(100)의 애플리케이션(110), 드라이버(111) 및 운영체제(112)와 같은 커널 모드 및 사용자 모드 개체들이 어떤 적절한 방식으로 구현될 수 있다. 시스템(100)의 애플리케이션(110), 드라이버(111) 및 운영체제(112)의 실시예들에 대한 설명이 예컨대 이하의 도 2의 애플리케이션(210), 드라이버(211) 및 운영체제(212); 도 4의 애플리케이션(410), 드라이버(411) 및 운영체제(412); 및 도 7의 애플리케이션(709), 드라이버(711) 및 운영체제(713)에 대한 설명에서 보여질 수 있다.

[0032] 전자 장치(103)는 컴퓨터, PDA(personal data assistant), 전화기, 모바일 장치, 서버, 또는 프로그램 명령어들 및/또는 프로세스 데이터를 해석 및/또는 실행하도록 구성될 수 있는 어떤 다른 장치에서와 같이 어떤 적절

한 방식으로 구현될 수 있다. 전자 장치(103)의 실시예들에 대한 설명은 예컨대 도 2의 전자 장치(204), 도 4의 전자 장치(404), 또는 도 7의 전자 장치(701)에 대한 논의들에서 발견될 수 있다.

[0033] 시스템(100)은 전자 장치(103)의 운영체제들 아래의 레벨에서 자원들에 대해 시도된 액세스를 트래핑하기 위한 어떤 적절한 시스템으로 구현될 수 있다. 시스템(100)은 또한 시도된 액세스가 악의적인지 아닌지 여부를 판단하기 위해 보안 규칙들을 조회함으로써, 시도된 액세스를 처리하는 어떤 적절한 수단을 통해 구현될 수 있다. 예를 들어 시스템(100)은 이하의 도 2-8에 기술된 바와 같은 시스템들과 방법들(200, 300, 400, 500, 600, 700 및 800)에 의해 구현될 수 있다.

[0034] 도 2는 악성 소프트웨어로부터 전자 장치를 보호하기 위한 가상 머신 모니터 기반 및 보안 규칙 기반의 실행 가능 보안 솔루션을 위한 시스템(200)의 실시예이다. 시스템(200)은 가상 머신 모니터에서 시스템(100)의 소정 구성요소들을 구현하는 시스템(100)의 실시예일 수 있다. 시스템(200)은 구성 가능한 보안 솔루션에 의해 악성 소프트웨어에 대항하여 보호되어야 할 전자 장치(204)를 포함할 수 있다. 시스템(200)의 구성 가능한 보안 솔루션은 모든 운영체제들 아래에서 실행되는 보안 에이전트, 보안 가상 머신 모니터, 클라우드 기반 보안 에이전트 및 OS 내 동작 보안(in-O/S behavioral security) 에이전트를 포함할 수 있다. O/S 하위 보안 에이전트 및 보안 가상 머신 모니터는 OS 내 동작 보안 에이전트에 의해 사용되는 자원들을 포함하여, 전자 장치(204)의 시스템 자원들에 대한 액세스를 보호하도록 구성될 수 있다. O/S 하위 보안 에이전트는 보안 가상 머신 모니터 안에서 실행될 수 있다. 클라우드 기반 보안 에이전트는 악성 소프트웨어 검출 정보를 O/S 하위 보안 에이전트 및 O/S 내 동작 보안 에이전트로 제공하고, 보안 가상 머신 모니터 및 O/S 내 동작 보안 에이전트로부터 악성 소프트웨어와 관련될 가능성이 있는 의심스러운 동향에 관한 정보를 수신하도록 구성될 수 있다. O/S 내 동작 보안 에이전트는 전자 장치 상에서 동작하는 악성 소프트웨어에 대한 증거를 찾아 전자 장치(204)를 검색하도록 구성될 수 있다. 시스템(200)은 전자 장치(204)의 자원들에 대해 시도된 액세스 이용을 트래핑하고, 그 시도에 해당하는 트리거된 이벤트(유발 이벤트)를 생성하고, 그 트리거된 이벤트에 관한 보안 규칙들을 조회하며 필요시 상기 시도에 관한 교정 액션을 수행하도록 구성된 하나 이상의 O/S 보안 에이전트들을 포함할 수 있다.

[0035] 일 실시예에서, 시스템(200)은 하나 이상의 O/S 내 보안 에이전트(218) 및 보안 가상 머신 모니터("SVMM") 보안 에이전트(217)와 통신 가능하게 연결된 보호 서버(202)를 포함할 수 있다. SVMM 보안 에이전트(217)는 SVMM(216)에 상주할 수 있다. SVMM(216)은 전자 장치(204) 상에 상주하여 동작될 수 있다. O/S 내 보안 에이전트(218) 및 SVMM 보안 에이전트(217)는 통신 가능하게 연결될 수 있다. 보호 서버(202), O/S 내 보안 에이전트(218), SVMM 보안 에이전트(217) 및 SVMM(216)은 전자 장치(204)를 악성 소프트웨어 감염으로부터 보호하도록 구성될 수 있다.

[0036] SVMM 보안 에이전트(217)는 도 1의 트리거된 이벤트 핸들러(108)의 일 실시예일 수 있다. SVMM(216)은 도 1의 O/S 하위 트래핑 에이전트(104)의 일 실시예일 수 있다.

[0037] 전자 장치(204)는 프로세서(206)에 연결된 메모리(208)를 포함할 수 있다. 전자 장치(204)는 어떤 적절한 목적을 위해 전자 장치 상에서 실행되는 하나 이상의 애플리케이션들(210)이나 드라이버들(211)을 포함할 수 있다. 전자 장치(204)는 운영체제(212)를 포함할 수 있다. 운영체제(212)는 전자 장치(204)의 시스템 자원들(214)에 대한 액세스를 애플리케이션들(210)이나 드라이버들(211)로 제공하도록 구성될 수 있다. SVMM(216)은 시스템 자원들(214)의 운영체제(212)의 그러한 호출(call)들을 인터셉트하도록 구성될 수 있다. SVMM(216) 및 SVMM 보안 에이전트(217)는 운영체제(212) 레벨 밑에서 동작할 수 있다. 예를 들어, SVMM(216) 및 SVMM 보안 에이전트(217)는 "Ring -1"과 같은 특권 모드에서 프로세서(260) 상에서 바로 동작될 수 있다.

[0038] 프로세서(206)는 예컨대 마이크로프로세서, 마이크로 컨트롤러, DSP(digital signal processor), ASIC(application specific integrated circuit), 또는 프로그램 명령어들 및/또는 프로세스 데이터를 해석 및/또는 실행하도록 구성된 어떤 다른 디지털 혹은 아날로그 회로를 포함할 수 있다. 일부 실시예들에서 프로세서(206)는 메모리(208)에 저장된 프로그램 명령어들 및/또는 프로세스 데이터를 해석 및/또는 실행할 수 있다. 메모리(208)는 일부 혹은 전체가 애플리케이션 메모리, 시스템 메모리, 또는 둘 모두로서 구성될 수 있다. 메모리(208)는 하나 이상의 메모리 모듈들을 보유 및/또는 하우징하도록 구성된 어떤 시스템, 기기, 또는 장치를 포함할 수 있다; 예를 들어 메모리(208)는 ROM(read-only memory), RAM(random access memory), 고체 상태 메모리, 또는 디스크 기반 메모리를 포함할 수 있다. 각각의 메모리 모듈은 일정 시간 동안 프로그램 명령어들 및/또는 데이터를 보유하도록 구성된 어떤 시스템, 기기 또는 장치(가령, 컴퓨터 판독 가능 비 임시 매체)를 포함할 수 있다.

[0039] 보호 서버(202)는 네트워크(244) 상에서 동작 중일 수 있다. 네트워크(244) 상에서 동작하는 보호 서버(202)는

클라우드 컴퓨팅 방식을 구현할 수 있다. 보호 서버(202)는 악성 소프트웨어 검출 규칙들 및 정보를 업데이트하기 위해 전자 장치(204)의 구성요소들과 통신하도록 구성될 수 있다. 보호 서버(202)는 전자 장치(204)에서 발생된 의심스러운 활동들에 관한 정보를 수신하고 그러한 의심스러운 활동들이 악성 소프트웨어 감염인지 아닌지 여부를 판단하도록 구성될 수 있다. 운영체제(212)는 하나 이상의 O/S 내 보안 에이전트들(218)을 포함할 수 있다. O/S 내 보안 에이전트(218)는 보호 서버(202)로부터 O/S 보안 규칙들(220)과 같은 감시 및 검출 규칙들을 수신하도록 구성될 수 있다. O/S 내 보안 에이전트(218)는 전자 장치(204) 상에서의 의심스러운 활동들을 감시하고 방지하기 위해 보호 서버(202)에 의해 수신된 O/S 내 보안 규칙들(220)을 사용하도록 구성될 수 있다. O/S 내 보안 에이전트(218)는 검출된 의심스러운 활동들을 다시 보호 서버(202)에 보고하도록 구성될 수 있다. O/S 내 보안 에이전트(218)는 악성 소프트웨어 동작들을 금지하고 그런한 금지사항들을 보호 서버(202)에 보고하도록 구성될 수 있다. 하나를 넘는 O/S 내 보안 에이전트(218)가 시스템(200) 안에 존재하는 경우, 각각의 O/S 내 보안 에이전트(218)는 트래핑, 검증(validating) 또는 O/S 내 보안 에이전트(218)와 관련된 다른 작업들의 지정된 일부를 수행하도록 구성될 수 있다. 그러한 일부는 운영체제 하위 보안 에이전트들에 의해 정의될 수 있다. 예를 들어 하나의 O/S 내 보안 에이전트(218)가 MOV 명령어들을 검증하거나 검사할 수 있고, 또 다른 O/S 내 보안 에이전트(218)가 JMP 명령어들을 검증하거나 검사할 수 있다. O/S 내 보안 에이전트(218)는 메모리 내 특정 페이지의 수명을 결정하도록 구성될 수 있다. 예를 들어 O/S 내 보안 에이전트(218)는 메모리의 어떤 페이지를 할당하기 위해 운영체제(212)에 의해 통상적으로 사용되는 프로세스들 및 단계들을 알 수 있다. 마찬가지로, O/S 내 보안 에이전트(218)는 자신의 로더(loader)에 애플리케이션의 이미지를 로딩하기 위해 운영체제(212)에 의해 통상적으로 사용되는 프로세스들 및 단계들을 알 수 있다. 그러한 프로세스들은 정적인 동작 패턴을 따를 수 있다. 따라서 O/S 내 보안 에이전트(218)는 주어진 액션에 대해 표준 절차들이 뒤따랐는지 여부를 판단하기 위해 운영체제(212)의 동작을 추적하도록 구성될 수 있다. O/S 내 보안 에이전트(218)는 SVMM 보안 에이전트(217)에 의해 트래핑된 동작이 O/S 내 보안 에이전트(218)에 의해 관찰된 해당 예상 액션들을 생성했는지 아닌지 여부를 판단하기 위해 SVMM 보안 에이전트(217)와 통신할 수 있다. 불일치는 악성 소프트웨어가 운영체제(212)의 일반 동작 밖에서 시스템 기능을 수행하고자 시도했음을 나타낼 수 있다. 따라서, 예컨대 O/S 내 보안 에이전트(218) 및 SVMM 보안 에이전트(217)는 문제의 페이지가 악성 소프트웨어에 의해 바로 메모리에 로딩되었는지 운영체제 로더에 의해 로딩되었는지 여부를 판단할 수 있다. 그러한 양태는 O/S 내 보안 에이전트(218)나 SVMM 보안 에이전트(217)가 정보를 보호 서버(202)에 보고하게 하거나, 보다 공격적인 트래핑 및 체킹을 이용하게 하거나 어떤 다른 교정 조처를 수행하도록 할 수 있다.

[0040] 일 실시예에서 O/S 내 보안 에이전트(218)는 운영체제(212) 안에 자신을 내장시켜 정황 정보를 제공하도록 구성될 수 있다. 예를 들어, O/S 내 보안 에이전트(218)는 자신이나 하위 구성요소를 드라이버 필터로서 등록하며 드라이버가 보거나 보지 않는 것이 무엇인지를 판단하기 위해 자신을 메인 드라이버에 첨부하도록 구성될 수 있다. 예컨대 NDIS.SYS에 필터로서 첨부됨에 따라, O/S 내 보안 에이전트(219)는 운영체제(212) 드라이버들에 의해 보이는 파일 I/O 동작들을 보고하도록 구성될 수 있다.

[0041] 다른 실시예에서, O/S 내 보안 에이전트(219)는 운영체제(219) 안으로부터 관찰되는 그러한 정보를 운영체제 밑에서 관찰되는 정보와의 비교를 위해 SVMM 보안 에이전트(216)나 다른 O/S 하위 보안 에이전트들에 제공하도록 구성될 수 있다. 두 세트의 정보 간 불일치는 자신을 숨기고자 시도하는 악성 소프트웨어의 존재를 나타낼 수 있다. 예를 들어 O/S 내 보안 에이전트(219)는 NDIS.SYS를 후킹(hook)하거나 필터링 할 수 있고 특정 파일로의 파일 쓰기들에 대해 감시할 수 있다. SVMM 보안 에이전트(216)는 입력 및 출력 명령들을 감시할 수 있다. SVMM 보안 에이전트(216)가 O/S 내 보안 에이전트(219)에 의해 보이는 함수 호출들의 리스트에 기반하여 보여져야 했던 것보다 많은 쓰기들을 판단했다면, 악성 소프트웨어는 운영체제(212)에 의해 제공되는 기능들 밖에서 디스크에 은밀하게 기입하고 있을 수 있다.

[0042] 네트워크(244)는 인터넷, 인트라넷, WAN(wide-area-networks), LAN(local-area-networks), 백홀(back-haul) 네트워크, 피어 투 피어 네트워크, 또는 이들의 어떤 조합과 같이 통신할 어떤 적절한 네트워크 안에서 구현될 수 있다. 보호 서버(202)는 보급 및 명성도 분석 로직을 적용하여 악성 소프트웨어를 추가 검출하기 위해 다양한 전자 장치(204) 상에서 실행되는 다양한 보안 에이전트들(218)로부터 제출되는 보고서들을 이용할 수 있다. 예를 들어 전자 장치(204) 상에서 식별되는 의심스러운 양태는 다른 전자 장치(204)를 적극적으로 보호하기 위해 보호 서버(202)를 위한 규칙 안에 통합될 수 있다. 그러한 규칙은 예를 들어 의심스러운 드라이버가 보고된 회수에 기반하여 결정될 수 있다. 예를 들어, 좁거나 느린 분포 패턴을 가진 미지의 드라이버가 악성 소프트웨어와 관련될 수 있다. 한편, 넓고 빠른 분포도를 가진 미지의 드라이버는 인기 있고 널리 이용 가능한 애플리케이션의 패치와 관련될 수 있다. 다른 예에서, 그러한 검출 드라이버는 호스트 악성 소프트웨어에 알려진 웹 사이트에 액세스 하였을 다른 전자 장치 상에서 실행되는 보안 소프트웨어에 의해 결정되었을 수 있다. 그러한

드라이버는 악성 소프트웨어와 관련된 것으로 결정될 수 있다.

[0043] SVMM(216)은 시스템(200)의 보안 가상 머신 감시 기능들 중 일부나 전부를 구현할 수 있다. SVMM(216)은 레지스터들, 메모리, 또는 I/O 장치들과 같은 시스템 자원들에 대한 액세스를 전자 장치 상에서 실행되는 하나 이상의 운영체제들로 인터셉트하도록 구성될 수 있다. 시스템(200)의 보안 가상 머신 감시 기능들은 SVMM(216)이나 이 개시의 가르침에 따라 전자 장치(204)를 보호하도록 구성된 어떤 다른 가상 머신 모니터를 이용하여 구현될 수 있다. SVMM(216)은 운영체제(212)가 시스템 자원들(214)을 액세스하고자 시도하는 동안 그 자체를 대신하거나 운영체제(212)를 통해 실행하는 애플리케이션들을 대신하여 운영체제(212)에 의해 취해진 액션들을 통제 및 필터링 하도록 구성될 수 있다. SVMM(216)은 전자 장치(204) 상의 운영체제(212) 아래에서 실행할 수 있으며, 운영체제(212) 및 애플리케이션(210)이나 드라이버(211)에서 이용 가능한 일부 혹은 전체 프로세서 자원들에 대해 통제할 수 있다. 애플리케이션(210)은 전자 장치(204) 상에서 실행되기 적합한 어떤 애플리케이션을 포함할 수 있다. 드라이버(211)는 전자 장치(204) 상에서 실행되기 적합한 어떤 드라이버를 포함할 수 있다. SVMM(216)에 의해 통제될 이용 가능한 프로세서 자원들은 가상화를 위해 지정된 자원들을 포함할 수 있다. 일 실시예에서 SVMM(216)은 운영체제(212), 애플리케이션(210) 또는 드라이버(211)에 의해 액세스될 시스템 자원들(214)을 가상화하도록 구성될 수 있다. 다만 예로서, 그러한 시스템 자원들(214)은 입/출력 장치들(226), 시스템 메모리(228) 또는 프로세서 자원들(230)을 포함할 수 있다. 다만 예로서, 프로세서 자원들(230)은 종래의 레지스터들(232), 디버그 레지스터들(234), 메모리 세그먼테이션(236), 메모리 페이징(238), 인터럽트들(240) 또는 플래그들(242)을 포함할 수 있다. I/O 장치들(226)은 키보드, 디스플레이, 마우스 또는 네트워크 카드들과 같이, 그러한 장치들로의 액세스를 포함할 수 있다.

[0044] SVMM(216)은 시스템 자원들(214)을 액세스하기 위해 운영체제(212)로부터 발생된 동작들의 실행을 트래핑하도록 구성될 수 있다. SVMM(216)은 시스템 자원들(214)의 특정 시도 액세스들을 트래핑하도록 구성된 제어 구조를 포함할 수 있다. 어떤 적절한 제어 구조가 사용될 수 있다. 일 실시예에서 그러한 제어 구조는 가상 머신 제어 구조("VMCS")(221)를 포함할 수 있다. SVMM(216)은 VMCS(221) 내부의 플래그들을 조작함으로써 그러한 실행을 트래핑하도록 구성될 수 있다. SVMM(216)은 시스템 자원들(214)에 대한 액세스와 관련된 운영체제(212), 애플리케이션(210) 또는 드라이버(211)의 어떤 적절한 동작을 트래핑하도록 구성될 수 있다. 그렇게 트래핑된 동작들은 예컨대 시스템 메모리(228) 내 메모리의 특정 페이지들의 읽기, 쓰기 및 실행; 프로세서 레지스터(230)로/로부터 어떤 값을 로딩 및 저장; 또는 I/O 장치들(226)로/로부터 읽기 및 쓰기를 포함할 수 있다. 어떤 그러한 동작들은 SVMM(216)에 의해 트래핑될 수 있는 가상 머신 빠져나가기("VM Exit")를 일으킬 수 있다. SVMM(216)은 프로세서(208)에 의해 생성되거나 운영체제(212)의 구성요소들에 의해 개시될 수 있는 인터럽트들(240)의 생성을 트래핑하도록 구성될 수 있다. SVMM(216)은 인(IN) 및 아웃(OUT) 명령어들을 트래핑하여 I/O 장치(226)로/로부터의 시도된 읽기 및 쓰기를 트래핑하도록 구성될 수 있다. SVMM은 예컨대 가상화 기술 지향 I/O("VTd")의 메커니즘들에 대한 액세스를 트래핑하여 그러한 명령어들을 트래핑하도록 구성될 수 있다. VTd는 프로세서(208)에 따라 I/O 장치 가상화를 허용할 수 있다. VTd 설비들을 액세스함으로써, SVMM 보안 에이전트(217)는 VTd에 의해 연결된 장치들을 판단하고, 운영체제(212)로터의 메타 정보, I/O 장치 상의 포트들, 또는 다른 적절한 정보를 판단할 수 있다. SVMM 보안 에이전트(217)는 그러한 가상화된 장치 액세스의 동작을 통제하거나 트래핑하도록 구성될 수 있다. 예를 들어 SVMM 보안 에이전트(217)는 프로그래머블 I/O 포트들에 주어진 I/O 할당치들을 포함하는 I/O 허가 맵들을 결정하도록 구성될 수 있다. SVMM 보안 에이전트(217)는 악성 소프트웨어에 의해 행해질 수 있는 그러한 허가 맵들에 대한 액세스를 트래핑하거나 운영체제(212) 상의 개체들의 관계 및 I/O 장치의 요청을 판단하기 위해 그 허가 맵들을 이용하도록 구성될 수 있다.

[0045] 일 실시예에서 SVMM 보안 에이전트(217)는 SVMM(216) 안에서 동작할 수 있다. 다른 실시예에서 SVMM 보안 에이전트(217)는 SVMM(216)의 외부에서 동작할 수 있지만, SVMM(216)에 통신 가능하게 연결될 수 있다. 그러한 실시예에서, SVMM 보안 에이전트(217)는 운영체제(212)와 같은 전자 장치(204)의 운영체제들의 레벨 아래에서 동작하고 있을 수 있다. SVMM 보안 에이전트(217)는 SVMM(216)과 동일한 레벨 및/또는 동일한 우선순위로 동작하고 있을 수 있다. SVMM 보안 에이전트(217)는 SVMM(216)에 의해 트리거(유발)되거나 트래핑된 이벤트들을 처리하도록 구성될 수 있다. SVMM 보안 에이전트(217)는 커널 레벨 루트킷들의 간섭이 없는 콘텐츠를 검사하기 위해 운영체제(212) 아래의 레벨에서 메모리(228)나 디스크의 콘텐츠를 액세스하도록 구성될 수 있다. 또한, SVMM 보안 에이전트(217)의 일부 동작들은 SVMM(216)에 의해 구현될 수 있고, SVMM(216)의 일부 동작들은 SVMM 보안 에이전트(217)에 의해 구현될 수 있다.

[0046] SVMM 보안 에이전트(217)는 어떤 액션들이 트래핑이나 트리거를 일으킬 것인지와 관련하여 SVMM(216)의 동작을 설정하도록 구성될 수 있다. 일 실시예에서 SVMM(216)은 트래핑된 액션들의 검출을 SVMM 보안 에이전트(217)로

전송하도록 구성될 수 있다. SVMM 보안 에이전트(217)는 트래핑된 액션들이 악성 소프트웨어나 악의적 활동들을 나타내는지 여부를 판단하기 위해 보안 규칙들(222)을 조회하며 보안 규칙들(222)에 기반하여 어떤 후속 액션을 취할지에 대한 지시를 SVMM(216)에 제공할 수 있다. 그러한 후속 액션은 시도된 액션을 허가하거나 시도된 액션을 불허하거나 다른 교정 단계들을 취하는 일을 포함할 수 있다.

[0047] SVMM(216) 및 SVMM 보안 에이전트(217)에 의한 시스템 자원들(214)에 대해 시도된 액세스 및 실행을 트래핑하는 동작이 O/S 내 보안 에이전트(218)에 의해 수집된 정보를 통해 조정될 수 있다. O/S 내 보안 에이전트(218)는 SVMM(216) 및 SVMM 보안 에이전트(217)의 트래핑 및 처리 동작들에 정황을 제공하도록 구성될 수 있다. 예를 들어 특정 운영체제 데이터 구조는 정상적으로 특정 애플리케이션이나 서비스에 의해서만 기입될 수 있다. O/S 내 보안 에이전트(218)는 어떤 애플리케이션들이나 프로세스들이 운영체제(212) 상에서 현재 가시적으로 실행되고 있는지를 판단할 수 있고 그 정보를 SVMM 보안 에이전트(217)로 전송할 수 있다. 특정 애플리케이션이나 서비스가 가시적으로 실행되는 것으로 나열(리스트)되어 있지 않으면, 데이터 구조에 대해 시도된 기입은 미허가 애플리케이션이나 프로세스로부터 나온 것일 수 있다.

[0048] O/S 내 보안 에이전트(218)는 하이퍼콜(hypercall)들을 통해 SVMM(216) 및/또는 SVMM 보안 에이전트(217)와 통신하도록 구성될 수 있다. 하이퍼콜들은 이용될 수 있는 가용 요청들뿐 아니라 관련 입력 및 출력 파라미터들을 정의한 서술자 테이블을 이용해 구현될 수 있다. 그러한 서술자 테이블은 O/S 내 보안 에이전트(218)가 SVMM(216) 및/또는 SVMM 보안 에이전트(217)와 통신하는 것이 가능한 하나 이상의 요청들을 정의할 수 있다. 그러한 서술자 테이블은 또한 그러한 요청에 대한 입력 및 출력 파라미터들이 메모리 안에 위치될 수 있는지 여부를 정의할 수도 있다.

[0049] O/S 내 보안 에이전트(218), SVMM 보안 에이전트(217) 및 보호 서버(202)는 서로를 인증하도록 구성될 수 있다. 보안 에이전트(212), SVMM 보안 에이전트(217) 및 보호 서버(202) 각각은 그 개체들 각각이 인증되지 않으면 서로와 통신을 계속하지 못하도록 구성될 수 있다. SVMM(216)은 O/S 내 보안 에이전트(218) 이미지를 메모리(206) 안에 위치시키고 암호화 서명 알고리즘들을 이용하여 메모리(206) 안의 O/S 내 보안 에이전트(218) 이미지를 검증하도록 구성될 수 있다. 보호 서버(202), O/S 내 보안 에이전트(218) 및 SVMM 보안 에이전트(217) 간의 인증은 암호화 해싱(hashing) 및/또는 서명 알고리즘들을 포함하여 어떤 적절한 방법을 사용할 수 있다. 일 실시예에서 그러한 인증은 사설 비밀 키의 교환을 수반할 수 있다. O/S 내 보안 에이전트(218)는 SVMM 보안 에이전트(217)의 인스턴스를 검증하기 위해 보호 서버(202)로부터 비밀 키를 수신하도록 구성될 수 있다.

[0050] O/S 내 보안 에이전트(218)는 운영체제(212)의 동작에 관한 정황 정보를 가질 수 있다. O/S 내 보안 에이전트(218)는 그러한 정황 정보를 제공하기 위해 SVMM 보안 에이전트(217)와 통신하도록 구성될 수 있다. SVMM 보안 에이전트(217)는 SVMM(216)에 예컨대 메모리의 어떤 페이지들을 어떻게 정의할지 또는 어떤 레지스터들을 트래핑할지를 명령할 수 있다.

[0051] SVMM(216)은 SVMM 보안 에이전트(217)에 의해 정의된 시스템 자원들(214)에 대한 액세스 시도들을 트래핑하도록 구성될 수 있다. 예를 들어 메모리 액세스의 트래핑을 위해, SVMM(216)은 읽기, 쓰기(기입) 또는 실행과 같은 동작들을 트래핑하도록 구성될 수 있다. 프로세서 레지스터들(230)로의 액세스를 트래핑하기 위해, SVMM(216)은 레지스터 값들을 로딩, 저장, 또는 판독하는 것을 포함하는 동작들을 트래핑하도록 명령 받을 수 있다. I/O 동작들을 트래핑하기 위해, I/O 장치들(226), SVMM(216)은 키보드, 마우스 또는 다른 주변기기들로의 입력이나 출력과 같은 동작들을 트래핑하도록 명령 받을 수 있다. 이하의 도면들에서 SVMM 보안 에이전트(217) 및/또는 다른 운영체제 하위 보안 에이전트들은 운영체제 내 보안 에이전트들과 연계하여, I/O 동작, 타깃 I/O 장치(226)의 식별, I/O 장치(226)에 대해 수행될 타깃 동작 및 전송될 데이터에 대해 판단하도록 구성될 수 있다.

[0052] SVMM 보안 에이전트(217)는 운영체제(212)의 어떤 개체가 전자 장치(204)의 자원을 액세스하고자 시도했는지, 또는 운영체제(212)의 어떤 개체에 자원이 속하는지와 같은 정황 정보를 판단하도록 구성될 수 있다. SVMM 보안 에이전트(217)는 어떤 적절한 방법을 통해 그러한 판단을 내리도록 구성될 수 있다. 일 실시예에서 SVMM 보안 에이전트(217)는 운영체제 내 보안 에이전트(218)로부터 그러한 판단들에 대한 정황 정보를 액세스하도록 구성될 수 있다. 다른 실시예에서 SVMM 보안 에이전트(217)는 운영체제(212)의 다양한 프로세스들이나 애플리케이션들의 순서를 판단하기 위해 운영체제(212)의 호출 스택 및/또는 프로세서(208)의 실행 스택을 직접적으로나 간접적으로 액세스하도록 구성될 수 있다. 실행 명령 포인터는 트리거를 일으키는 명령을 가리킬 수 있고 실행 스택 포인터 및 실행 베이스 포인터는 스택 프레임들을 가리킬 수 있다. 실행 베이스 포인터와 스택 사이를 지남으로써, 가까운 동작에 대한 정황을 제공하는 이전의 함수 호출들이 식별될 수 있다. 그러한 스택들은 시도되었던 동작 및 소스 메모리 위치를 가리킬 수 있다. 또 다른 실시예에서, SVMM 보안 에이전트(217)는 시도가

악의적인지 혹은 악성 소프트웨어를 나타내는지 여부를 판단하기 위해 보안 규칙들(222)과 연계하여 메모리 맵을 이용하도록 구성될 수 있다. 그러한 메모리 맵은 예컨대 시도된 액세스의 메모리 위치가 주어질 때, 시도된 자원 액세스를 행했던 개체를 가리킬 수 있다. 그러한 메모리 맵은 예컨대 가상 메모리 페이지 식별자들 및/또는 물리적 메모리 어드레스들 안에 정의될 수 있다. 그러한 메모리 맵은 다른 예에서, 해당 시도의 타깃의 메모리 위치에 해당하는 개체를 가리킬 수 있다. 메모리 맵을 이용하여, SVMM 보안 에이전트(217)는 시도된 액세스의 소스 및 타깃의 아이디들 또는 그들의 개체 소유자들을 판단하도록 구성될 수 있다. 메모리 맵은 시스템의 실행을 감시하는 것을 통해 운영체제 내 보안 에이전트들과 연계하여, 부분적으로 SVMM 보안 에이전트(217) 또는 이하의 도면들에 있는 다른 O/S 하위 보안 에이전트들에 의해 생성될 수 있다. SVMM 보안 에이전트(217) 및/또는 이하 도면의 기타 운영체제 하위 보안 에이전트들은 운영체제 내 보안 에이전트들과 연계하여, 해당 위치가 특정 코드 섹션이나 데이터 섹션에 속하는지 여부; 그것이 어느 모듈, 프로세스, 애플리케이션, 이미지, 또는 기타 개체에 속하는지; 또는 그것이 사용자 모드 또는 커널 모드 개체들과 관련되는지 여부를 판단할 수 있다. SVMM 보안 에이전트(217) 및/또는 이하 도면의 다른 운영체제 하위 보안 에이전트들은 운영체제 내 보안 에이전트들과 연계하여, 가상 메모리 및 물리적 메모리의 매핑을 위해 전자 장치(204) 상에서 실행되는 다양한 개체들의 식별, 위치, 및 허가를 가리키는 메타데이터를 판단할 수 있다. 마찬가지로, SVMM 보안 에이전트(217) 및/또는 이하 도면의 기타 운영체제 하위 보안 에이전트들은 대규모 스토리지 장치 내에서 그러한 개체들의 이미지들의 위치를 판단하기 위해 그 대규모 스토리지 장치 내 섹터들의 매핑을 이용할 수 있다. SVMM 보안 에이전트(217) 및/또는 이하 도면의 기타 운영체제 하위 보안 에이전트들은 운영체제 내 보안 에이전트들과 연계하여, 주어진 개체에 대해 그들이 상주하는 섹터들, 파일들, 디렉토리들 및 볼륨들을 판단할 수 있다.

[0053] SVMM 보안 에이전트(217)는 O/S 내 보안 에이전트(218) SVMM 보안 에이전트(217) 및 SVMM(216)의 동작을 위해 필요 시 시스템 메모리(228)와 같은 메모리를 할당하도록 구성될 수 있다. SVMM 보안 에이전트(217)는 SVMM(216)이 미인가 읽기 및 쓰기 동작들에 대항하여 그렇게 할당된 메모리를 확보할 것을 요청하도록 구성될 수 있다. SVMM(216)은 메모리가 O/S 내 보안 에이전트(218)에 의해 할당될 때의 시점과 SVMM(216)에 의해 프로텍션(보호)이 설정된 시점 사이에 악성 소프트웨어가 악성 코드를 추가할 기회를 제거하기 위해 메모리의 보호가 설정된 후 할당된 메모리를 초기화하도록 구성될 수 있다.

[0054] SVMM 보안 에이전트(217)는 SVMM 보안 규칙들(222)을 안전하게 수신하기 위해 보호 서버(202)와 통신하도록 구성될 수 있다. SVMM 보안 규칙들(222)은 명령어들, 로직, 규칙들, 공유 라이브러리들, 함수들, 모듈들, 또는 어떤 보안 정책들이 활용되는지에 대해 SVMM(216)에 지시하기 위한 어떤 다른 적절한 메커니즘을 포함할 수 있다. SVMM 보안 에이전트(217)는 전자 장치(204)로부터 검출된 악성 소프트웨어 및 의심스러운 활동들에 관해 보호 서버(202)에 정보를 전달하도록 구성될 수 있다.

[0055] O/S 내 보안 에이전트(218)는 O/S 내 보안 규칙들(220)을 수신하기 위해 보호 서버(202)와 통신하도록 구성될 수 있다. O/S 보안 규칙들(220)은 명령어들, 로직, 규칙들, 공유 라이브러리들, 함수들, 모듈들, 또는 O/S 내 보안 에이전트(218)가 전자 장치(204) 상에서 악성 소프트웨어를 검출하게 하는 어떤 다른 적절한 메커니즘을 포함할 수 있다. O/S 내 보안 에이전트(218)는 전자 장치(204) 상에서 검출된 악성 소프트웨어 및 의심스러운 활동들에 관해 보호 서버(202)에 정보를 전송하도록 구성될 수 있다.

[0056] O/S 내 보안 규칙들(220) 및 SVMM 보안 규칙들(222)은 각각 악성 소프트웨어 감염들로부터 전자 장치(204)를 보호하고 악성 소프트웨어를 포함할 수 있는 의심스러운 활동들을 검출하기 위한 보호 규칙들을 포함할 수 있다. O/S 내 보안 에이전트 보안 규칙들은 O/S 내 보안 에이전트(218)에 의해 그 안에서 실행되는 규칙들을 포함할 수 있다. SVMM 보안 규칙들(222)은 SVMM(216) 및/또는 SVMM 보안 에이전트(217)에 의해 그 안에서 실행되는 규칙들을 포함할 수 있다.

[0057] SVMM 보안 규칙들(222)은 전자 장치(204)의 악성 소프트웨어 감염을 어떻게 관찰하고 검출할지에 대한 정의들을 가진 정보를 SVMM 보안 에이전트(217)로 제공하도록 구성될 수 있다. 예를 들어 SVMM 보안 규칙들(222)은 SVMM 보안 에이전트(217)가 악성 소프트웨어 표시를 위해 감시할 수 있는 애플리케이션(210)이나 드라이버(211)와 같은 개체들로부터 어떤 타입의 함수 호출들이나 동향들이 있는지에 대한 분류들을 포함할 수 있다. 다른 예로서, SVMM 보안 규칙들(222)은 어떤 파라미터들을 사용할지, 그렇게 트리거된 함수 호출들로부터 값들을 어떻게 추출할지, 혹은 그러한 호출들의 동작을 어떻게 비준할지를 포함하여, SVMM 보안 에이전트(217)가 그 트리거된 함수 호출들을 어떻게 처리할지에 대한 정의들을 포함할 수 있다. 또한 SVMM 보안 규칙들(222)은 SVMM 내 보안 에이전트(217)가 애플리케이션(210)이나 드라이버(211)와 같은 전자 장치의 개체들의 동향을 어떻게 감시할지에 대한 정보뿐 아니라 그러한 동향 검출 규칙들의 예외들까지 포함할 수 있다. 또 다른 예들에서 SVMM 보안 규칙들(222)은 SVMM 보안 에이전트(217)가 그러한 동향 검출 규칙들에 의해 검출된 악성 행위들을 어떻게 금

지하고 복구할지에 대한 정보를 포함할 수 있다. SVMM 보안 규칙들(222)은 SVMM 보안 에이전트(217)가 감시하고 수집하여 보호 서버(202)로 보내야 할 데이터가 어떤 것인지에 대한 세부사항들을 포함할 수 있다.

[0058] 마찬가지로 O/S 내 보안 규칙들(220)은 전자 장치(204)의 악성 소프트웨어 감염을 어떻게 관찰하고 검출할지에 대한 정의뿐 아니라 그러한 행위들을 SVMM 보안 에이전트(217)와 어떻게 조율할지에 대한 정의들을 가진 정보를 O/S 내 보안 에이전트(218)로 제공하도록 구성될 수 있다.

[0059] SVMM 보안 규칙들(222)은 또한 SVMM(216)이 어떤 액션들을 트래핑할 것인지에 관한 규칙들을 포함할 수 있다. SVMM 보안 에이전트(217)는 그러한 규칙들을 SVMM(216)에 적용하도록 구성될 수 있다. 예를 들어 SVMM 보안 에이전트(217)는 식별 가능한 가상 혹은 물리적 메모리 페이지 안에 트래핑될 함수의 어드레스를 변환하고, SVMM(216)이 그러한 페이지의 실행을 트래핑하라는 요청을 생성하며, 이어서 그 실행을 트래핑한 후 보안 에이전트(217)를 호출(콜)하도록 구성될 수 있다. SVMM 보안 에이전트(217)는 SVMM(216)과의 자신의 인터페이스를 통해 SVMM 보안 규칙들(222)을 수신하도록 구성될 수 있다. 그러한 인터페이스는 하이퍼콜 기반의 인터페이스를 포함할 수 있다. SVMM 보안 에이전트(217)는 어떤 결과적 검출들이나 보고서들을 동일한 하이퍼콜 기반의 인터페이스를 통해 SVMM(216)으로 푸쉬하도록 구성될 수 있다.

[0060] 일 실시예에서 SVMM(216)은 SVMM 보안 에이전트(217)를 조회하지 않고, 트리거된 액션들을 처리하도록 구성될 수 있다. 그러한 실시예에서 SVMM(216)은 SVMM 보안 에이전트(217)로 전달되지 않을 수도 있는, SVMM(216) 내에서 처리되는 추가 트리거들을 설치하도록 구성될 수 있다. 그러한 추가 트리거들이 SVMM 보안 규칙들(222)에 의해 정의될 수 있다. 일 실시예에서 SVMM 보안 규칙들(222)은 SVMM(216)을 위한 메모리 페이지 검색 규칙들을 정의할 수 있다. 그 규칙들은 악성이고 메모리에 상주하는 것이 허용되어서는 안 되는 개체들이나 그 변형들의 리스트를 포함할 수 있다. 그 규칙들은 또한, 시스템 메모리(228) 안에 존재하도록 특정적으로 허용되는 페이지들의 리스트를 포함하도록 구성된 화이트리스트를 포함할 수도 있다. 다른 실시예에서 SVMM 보안 규칙들(222)은 SVMM(216)에 대해 메모리 페이지 액세스 규칙들을 정의할 수 있다. 그 규칙들은 어떤 코드 페이지들이 허용되는지, 혹은 반대로 어떤 코드 페이지들이 주어진 코드나 데이터 페이지를 액세스하는 것이 금지되는지에 대한 정의을 포함할 수 있다. 결론적으로, SVMM 보안 규칙들(222)은 SVMM(216)가 메모리 스캐너로서 기능하도록 하고/하거나 메모리 페이지들에 대한 액세스를 통제할 것을 명령하도록 구성될 수 있다.

[0061] SVMM(216)은 SVMM 보안 에이전트(217), SVMM(216) 및 O/S 내 보안 에이전트(218)를 시스템 자원들(214) 안의 그들 각자의 코드 및 데이터 페이지들에 대한 미허가 읽기 및 쓰기 액세스를 금지하여 보호하도록 구성될 수 있다. 예를 들어 애플리케이션(210)이나 드라이버(211)가 SVMM 보안 에이전트(217), SVMM(216) 및 O/S 내 보안 에이전트(218)의 무결성이나 동작에 영향을 미치는 결과를 가져올 수 있는 시스템 메모리(228), 프로세서 레지스터들(230) 또는 I/O 장치들(226)의 일부에 대한 요청을 행하면, SVMM(216)은 그렇게 시도되는 요청을 인터셉트하고 뒤이어 그 요청을 재라우팅하거나 그것을 거부하거나 다른 적절한 액션을 수행하도록 구성될 수 있다. 다른 예에서 SVMM(216)은 SVMM 보안 에이전트(217) 자체나 다른 상응하거나 관련된 프로그램들과 같은 메모리 보안 소프트웨어 애플리케이션들을 위해 SVMM 보안 에이전트(217), SVMM(216) 및 O/S 내 보안 에이전트(218)에 영향을 미치는 시스템 메모리(228), 프로세서 레지스터들(230) 또는 I/O 장치들(226)의 일부에 대한 읽기 액세스를 허가하도록 구성될 수 있다. 그러한 허가는 SVMM(216)에 대해 시스템 메모리(228)와 같은 시스템 자원들(214)에 대한 액세스를 어떻게 처리할지를 정의할 수 있는 SVMM 보안 규칙들(222) 안에서 정의될 수 있다. 일 실시예에서 SVMM 보안 규칙들(222)은 SVMM 보안 에이전트(217)를 포함할 수 있는 안전한 보안 프로그램들의 화이트리스트를 포함할 수 있다.

[0062] 보안 서버(202)와 통신하기 위해 SVMM(216)은 보안 네트워크 인터페이스(224)를 포함할 수 있다. 보안 네트워크 인터페이스(224)는 SVMM(216)이나 SVMM 보안 에이전트(217)와 같은 전자 장치(204)의 구성요소 및 보호 서버(202)와 같은 네트워크 서버 사이에 안전한 액세스를 제공하도록 구성될 수 있다. SVMM(216)은 보안 네트워크 인터페이스(22)를 구현할 수 있는 논리적 TCP/IP 드라이버 또는 기타 통신 인터페이스를 포함할 수 있다. 보호 서버(202)는 SVMM 보안 규칙들(222)이나 O/S 내 보안 규칙들(220)과 같은 보호 규칙들을 제공할 뿐 아니라, 보안 네트워크 인터페이스(224)를 통해 SVMM(216)이나 SVMM 보안 에이전트(217)가 자신을 업데이트할 것을 지시하라고 통신하도록 구성될 수 있다. 보호 서버(202)는 특정 전자 장치(204) 또는 특정 SVMM(216)을 위해 맞춤화된 규칙들을 전달하도록 구성될 수 있다. 그러한 맞춤화는 전자 장치(204) 상에 보고되었던 악의적 행위들의 타입 및 그와 함께 안티 바이러스 프로그램, 방화벽, 또는 다른 보호 메커니즘과 같은 전자 장치(204) 내부의 기타 보호 메커니즘들을 포함할 수 있다. 일 실시예에서 보호 서버(202)는 예컨대 로컬 네트워크 상에서 전자 장치(204)의 관리자에 의해 운영될 수 있다. 그 경우 관리자는 보호 서버(202)로부터 수신된 규칙들에 의해 구현될 수 있는, 의심스러운 동향들을 처리하기 위한 글로벌 혹은 개인화된 정책들을 세팅할 수 있다. SVMM(21

6)은 SVMM(216)이나 SVMM 보안 에이전트(217)로 보호 서버(202)를 통해 안전하게 전달된 새로운 이미지를 통해 자신을 어떻게 업데이트할지를 알리는 업데이트 엔진을 포함할 수 있다.

[0063] O/S 내 보안 규칙들(220) 및 SVMM 보안 규칙들(222)은 각각 전자 장치(204) 상에서 관찰된 특정 액션들이나 동작들 또는 그들의 클래스들이 보호 서버(202)로 전달될 것을 요청하도록 구성될 수 있다. 여기서 보호 서버는 해당 액션이 전자 장치(204) 상에서 진행되도록 허용되기 전에 그 관찰사항들을 검사 및 검증할 수 있다. 보호 서버(202)는 그러한 액션이 동기적으로 혹은 비동기적으로 검사되게 허용하도록 구성될 수 있다. 일 실시예에서 I/O 내 보안 에이전트(218)는 문제의 여지가 있는 활동들, 코드나 데이터의 세그먼트들, 또는 액션들을 보호 서버(202)에 의한 검증을 위해 SVMM(216)으로 전달하도록 구성될 수 있다. 예를 들어 O/S 내 보안 에이전트(218)는 메모리 안에 로딩된 서명되지 않은 드라이버를 검출함으로써 악성 소프트웨어의 의심스러운 인스턴스를 검출할 수 있다. SVMM(216)은 O/S 내 보안 에이전트(218)로부터 의심스러운 소프트웨어에 대한 정보를 수신할 수 있으며 보호 서버(202)로 그것을 제공할 수 있다.

[0064] SVMM 보안 규칙들(222)은 전자 장치의 어떤 적절한 시스템 자원에 대한 액세스를 허용하거나 거부하도록 구성될 수 있다. 감시되도록 이용 가능한 그러한 자원들은 프로세서(206)에 의해 노출되는 자원들에 종속될 수 있다. 예를 들어 일 실시예에서 SVMM 보안 규칙들(222)은 SVMM(216)dl 시스템 메모리(228), I/O 장치들(226) 및 인터럽트들(140)에 대한 액세스를 제한할 수 있게 하도록 구성될 수 있다. 그러한 제한은 키보드 디스플레이들이나 착탈가능 디스크들과 같은 I/O 장치들로의 미허가 액세스를 금지할 수 있다. 다른 실시예에서, SVMM 보안 규칙들(222)은 SVMM(216)이 인터럽트(240)와 같이 프로세서 레지스터들 안의 개체들을 포함하는 인터럽트 서술자 테이블 엔트리들로의 액세스를 제한할 수 있게 하도록 구성될 수 있다. 또 다른 실시예에서 SVMM 보안 규칙들(222)은 SVMM(216)이 확장 페이지 테이블들("EPT")나 물리적 메모리를 호스팅하기 위해 가상 메모리(게스트 운영체제 관점에서 실제 메모리)의 매핑을 다루는 어떤 다른 메커니즘에 대한 액세스를 제한할 수 있게 하도록 구성될 수 있다.

[0065] 전자 장치(204)가 가상화를 지원하는 프로세서(208) 이외에 하나 이상의 프로세서들을 포함하는 경우, SVMM(216)이나 또 다른 SVMM(216)의 인스턴스는 그러한 다른 프로세서들의 가상화된 자원들을 액세스하고자 하는 시도들을 인터셉트하도록 구성될 수 있다. 전자 장치(204)가 예컨대 프로세서(208)를 포함하는 쿼드 프로세서(quad-processor)를 포함하는 경우, 쿼드 프로세서의 자원들은 SVMM(216)에 의해 보호될 수 있다. 하나 이상의 다른 프로세서들이 가상화를 지원하지 못하는 경우, SVMM(216)은 자신들의 자원들에 대한 액세스를 보호하지 못할지도 모른다. 하나 이상의 다른 프로세서들이 프로세서(208)와 다른 가상화 기술을 지원하는 경우, SVMM(216)은 SVMM(216)이 프로세서(208)와 다른 방식으로 보호되는 경우 자원들이 가상화되는 방식이 상이할 수 있기 때문에 자신들의 자원에 대한 액세스를 보호하도록 구성될 수 있다.

[0066] 동작 시 보호 서버는 네트워크(244) 상에서 실행될 수 있다. O/S 내 보안 에이전트(218)는 전자 장치(204)에서 악성 소프트웨어를 검색하고, 의심스러운 동향에 대해 전자 장치(204) 상의 애플리케이션(210) 및 드라이버(211)와 같은 개체들의 동향을 관찰하고, 발견되었던 어떤 감염들을 복구함으로써 악성 소프트웨어 감염들로부터 전자 장치(204)를 보호하도록 전자 장치(204) 상에서 실행될 수 있다. O/S 내 보안 에이전트(218)는 운영체제(212)와 동일한 우선순위나 레벨에서 실행될 수 있으며 운영체제(212) 안에서 실행될 수 있다. SVMM(216)은 전자 장치(204)의 시스템 자원들에 대해 시도된 액세스를 트래핑함으로써 전자 장치(204)를 악성 소프트웨어 감염으로부터 보호하도록 전자 장치(204) 상에서 동작될 수 있다. SVMM 보안 에이전트(217)는 전자 장치(204)나 다른 적절한 전자 장치 상에서 실행되어, SVMM(216)의 트래핑 동작을 세팅하거나 트래핑된 시스템 자원들에 대해 시도되어 트래핑된 액세스들의 일부나 전부를 처리할 수 있다. SVMM(216) 및 SVMM 보안 에이전트(217)는 "Ring -1"의 우선순위를 가진 운영체제(212) 밑에서 실행될 수 있다. SVMM 보안 에이전트(217)는 SVMM(216) 상에서 실행될 수 있다.

[0067] 보호 서버(202)는 SVMM 보안 규칙들(222) 및 O/S 내 보안 규칙들(220)과 같은 보안 규칙들을 전자 장치(204)로 보낼 수 있다. 그러한 규칙들은 SVMM(216)에 O/S 내 보안 규칙들(220)을 제공할 수 있는 SVMM 보안 에이전트(217)에 의해 수신될 수 있다. 그러한 규칙들은 O/S 내 보안 에이전트(218)에 의해 수신될 수 있다.

[0068] 보호 서버(202), 보안 에이전트(208) 및 SVMM 보안 에이전트(217)는 각각 서로를 인증할 수 있다. SVMM 보안 에이전트(217)는 보안 에이전트(218)의 이미지를 메모리 안에서 찾을 수 있고, 메모리에 상주한 보안 에이전트(218)의 이미지를 검증하기 위해 암호화 서명 알고리즘들을 이용할 수 있다. 보호 서버(202) 및 SVMM 보안 에이전트(217)는 서로를 올바로 식별하기 위해 암호화 해싱 및 서명 알고리즘들을 이용하여 서로를 인증할 수 있다. SVMM 보안 에이전트(217) 및 보호 서버(202)는 또한 서로의 아이디를 인증하기 위해 사설 비밀 키를 교환

할 수 있다. 보안 에이전트(218)는 SVMM 보안 에이전트(217)의 인스턴스를 검증하기 위해 보호 서버(202)로부터 비밀 키를 수신할 수 있다. 보안 에이전트(218), SVMM 보안 에이전트(217) 및 보호 서버(202) 사이의 통신은 그 에이전트들 각각이 서로 인증되지 않으면 완전히 설정되지 못할 수 있다. 마찬가지로 SVMM 보안 에이전트(217) 및 SVMM(216)은 그들이 개별 개체들로서 실행 중이면 서로를 검증 및 인증할 수 있다.

[0069] SVMM(216) 및 SVMM 보안 에이전트(217)는 운영체제(212) 및 전자 장치(204)의 모든 운영체제들 아래에서 실행될 수 있다. SVMM(216)은 운영체제(212), 보안 에이전트(218), 애플리케이션(210) 및 드라이버(211)에 의한 I/O 장치들(226), 시스템 메모리(228) 및 프로세서 레지스터들(230)을 포함하는 시스템 자원들(214)로의 액세스를 감시할 수 있다. SVMM(216)은 운영체제(212), 보안 에이전트(218), 애플리케이션(210), 드라이버(211), 또는 전자 장치(204)의 어떤 다른 개체에 의해 요청되는 주요 동작들의 실행을 트래핑할 수 있다. SVMM(216)은 VMCS(221) 내부의 플래그들을 조작함으로써 그러한 실행을 트래핑할 수 있다. VMCS(221)가 보호되는 자원에 대한 요청을 인터셉트할 때, 추가 동작, 진단 및 복구를 위해 SVMM(216)으로 동작이 핸드오프될 수 있다. 일 실시예에서 동작은 SVMM 보안 에이전트(217)에 의해 후속 처리될 수 있다. 다른 실시예에서 트래핑된 동작의 처리는 SVMM(216) 자체에 의해 수행될 수 있다. SVMM(216)은 악성 소프트웨어에 대한 보호를 제공하기 위해 전자 장치(204)의 어떤 필수 동작을 트래핑할 수 있다. 그러한 동작들은 시스템 메모리(228) 내 특정 코드나 데이터 페이지들의 읽기, 쓰기 및 실행; 시스템 레지스터 및 프로세서 레지스터들(230)로부터의 값을 로딩 및 저장; 또는 I/O 장치들(226)로/로부터 읽기를 포함할 수 있으나 그에 국한되는 것은 아니다. SVMM(216)에 의해 트래핑될 특정 동작들은 SVMM 보안 규칙(222)에 의해 정의될 수 있다.

[0070] 보호 서버(202)는 SVMM 보안 에이전트(217)나 O/S 내 보안 에이전트(218)와 통신하여 각각에 보안 규칙들을 제공할 수 있다. 일 실시예에서 보호 서버(202)는 SVMM 보안 규칙들(222)을 SVMM 보안 에이전트(217)로 전달할 수 있다. 다른 실시예에서 보호 서버(202)는 O/S 내 보안 규칙들(220)을 O/S 내 보안 에이전트(218)로 전달할 수 있다. 또 다른 실시예에서 보호 서버(202)는 O/S 내 보안 규칙들(220)을 SVMM 보안 에이전트(217)로 전달할 수 있고, 그런 다음 SVMM 보안 에이전트가 그 규칙들을 O/S 내 보안 에이전트(218)로 제공할 수 있다.

[0071] 전자 장치(204)를 동작시키는 애플리케이션(210), 드라이버(211) 또는 다른 개체들이 O/S 내 보안 에이전트(218)에 의해 관찰될 수 있다. O/S 내 보안 에이전트(218)는 O/S 내 보안 규칙들(220)을 이용하여 그러한 프로세싱 개체들의 동향을 관찰하고 그들의 동향이 악성 소프트웨어 감염 가능성을 나타내는 의심스러운 동향을 이루는지 여부를 판단할 수 있다. 그러한 의심스러운 활동들이 검출되면, O/S 내 보안 에이전트(218)는 추가 분석 및 지시를 위해 보호 서버(202)로 그 의심 정보를 제공할 수 있다. O/S 내 보안 규칙들(220)은 O/S 내 보안 에이전트(218)로, 그러한 동향들이 의심스럽다는 것을 가리킬 뿐 아니라 교정 액션을 가리킬 수도 있다. 예를 들어 애플리케이션(210)이 악성 소프트웨어를 호스트하는 것으로 알려진 네트워크 목적지와 통신할 수 있다. O/S 내 보안 에이전트(218)는 애플리케이션(210)의 활동을 알아차리고, 뒤이어 그 네트워크 목적지에 대한 애플리케이션(210)의 네트워크 액세스를 차단할 수 있다. O/S 내 보안 에이전트(218)는 또한 전자 장치(204)에서 악성 소프트웨어를 검색할 수 있다. 예를 들어 O/S 내 보안 에이전트(218)는 악성 소프트웨어의 특징에 해당하는 패턴들을 찾아 메모리(206) 또는 시스템 메모리(228)의 콘텐츠를 검사할 수 있다. 그러한 검사는 예컨대 애플리케이션(210)이 알려진 악성 소프트웨어의 세그먼트에 해당하는 코드 블록을 포함한다는 것을 드러낼 수 있다. 그러면 O/S 내 보안 에이전트(218)가 애플리케이션(210)을 복구하거나 애플리케이션(210)을 제거하거나 어떤 다른 적절한 액션을 수행으로써 악성 소프트웨어 감염에 대해 전자 장치(204)를 청소(clean)할 수 있다. O/S 내 보안 에이전트(218)는 어떤 검출된 의심스런 동향들이나 다른 악성 소프트웨어에 대한 표시들에 관해 보호 서버(202)와 통신할 수 있고, 보호 서버(202)로부터 그러한 악성 소프트웨어를 어떻게 처리할지에 대한 지시들을 수신할 수 있다.

[0072] 일 실시예에서 SVMM 보안 에이전트(217)는 시도된 동작을 행했던 개체의 출처에 기반하여 트래핑된 동작을 평가하도록 구성될 수 있다. 예를 들어 어떤 드라이버가 알려지지 않은 도메인으로부터 다운로드 되었거나 알려지지 않은 보증자로부터의 인증서를 가진다면, 후속 동작할 그 드라이버의 능력이 제한될 수 있다. 예를 들어 상태가 알려지지 않은 드라이버는 다른 드라이버에 자신을 첨부하는 기능을 거부당할 수 있다. 드라이버가 악성 소프트웨어를 호스트하는 것으로 알려진 도메인으로부터 다운로드 되었거나 거짓 신용장을 포함하는 경우, 그 드라이버는 심지어 로딩이 허용되지 못할 수 있다. 마찬가지로 어떤 드라이버가 특정 도메인으로부터 나왔거나 특정 저자에 의해 생성된 것으로 알려진 경우, SVMM 보안 에이전트(217)는 그 드라이버를 업데이트하도록 인가된 전자 장치(204) 내 서비스들을 인식하고 그러한 서비스들에 대해 그 드라이버를 기입하거나 액세스하는 기능을 제한하도록 구성될 수 있다. 예를 들어 컴퍼니 X로부터의 커널 드라이버는 전자 장치(204) 상에 상주하는 컴퍼니 X의 업데이트 서비스 소프트웨어로부터만 기입될 수 있다. SVMM 보안 에이전트(217)는 업데이트 서비스

의 동작 및 무결성을 검증하도록 구성될 수 있다. 다른 실시예에서 SVMM 보안 에이전트(217)는 해당 시도의 타깃에 기반하여 트래핑된 동작을 평가하도록 구성될 수 있다. 예를 들어 어떤 서비스로부터 소프트웨어를 업데이트하려는 시도는 커널 드라이버들에 대해 트래핑될 수 있지만 애플리케이션 소프트웨어에 대해서는 트래핑될 수 없다.

[0073] 어떤 개체가 의심스럽다고 판단되었거나 어떤 시도가 악성 소프트웨어를 가리키는 것으로 판단되었으면, 그 시도를 일으킨 프로세스 및 그 프로세스를 포함한 메모리가 연관될 수 있다. 동일 부분의 메모리를 액세스하는 다른 프로세스들도 마찬가지로 악성 소프트웨어라고 판단될 수 있다. 트래핑된 자원 액세스 시도는 저장될 수 있으며, 보호된 자원을 액세스하려는 후속 시도는 그 최초 이벤트에 비추어 평가될 수 있다. 예를 들어 악의적 동작은 코드가 데이터 세그먼트에 기입되고 그런 다음 실행될 것을 요할 수 있다. 따라서 SVMM 보안 에이전트(217)가 그 데이터 세그먼트에 대한 최초 기입 액세스를 트래핑하고, 그 기입을 허용하지만, 그 기입 액세스의 소스를 기록할 수 있다. 이어서 SVMM 보안 에이전트(217)는 데이터 세그먼트를 실행하려는 후속 시도를 트래핑하고, 앞서 트래핑된 동작에 비추어 해당 시도의 악의적 상태, 그것을 시도했던 개체, 또는 다른 적절한 범죄조사 정보를 평가할 수 있다.

[0074] SVMM 보안 에이전트(217)는 VMCS(221)와 같은 제어 구조를 통해 시스템 자원들 중 어느 것이 SVMM(216)이 트래핑해야 하는 것인지에 관하여 SVMM(216)에 지시할 수 있다. 그러면 SVMM(216)은 운영체제(212), 애플리케이션(210) 또는 드라이버(211)와 같은 전자 장치(204)의 개체들로부터 발생된 시스템 자원들(214)에 대한 액세스 요청들을 트래핑할 수 있다. 예를 들어 시스템 메모리(228)의 일부를 판독하거나 기입하거나 실행하라는 요청이 이루어진 경우, SVMM(216)은 VMCS(221) 내 시스템 메모리의 지정된 부분에 대한 플래그 설정을 통해 그러한 요청을 인터셉트할 수 있다. 다른 예에서 입력 또는 출력 동작들과 같이 I/O 장치들(226)에 대해 이루어진 액세스 요청들이 VMCS(221)에 의해 인터셉트될 수 있다. 또 다른 예에서 로딩이나 저장 명령과 같은 프로세스 레지스터들(230)에 대한 요청들이 VMCS(221)에 의해 트래핑될 수 있다. 어떤 그러한 트래핑들은 시도된 액세스에 대한 SVMM(216)의 통지를 가져올 수 있다. SVMM(216)이 시스템 자원들(214)에 대해 시도된 동작을 트래핑하였으면, SVMM(216)은 그 트래핑된 실행을 SVMM 보안 에이전트(217)에 전송할 수 있다.

[0075] O/S 내 보안 에이전트(218) 및 SVMM 보안 에이전트(217)는 운영체제(212) 안에서 행해지는 동작들의 정황을 판단하기 위해 통신할 수 있다. 예를 들어 운영체제(212)로부터 전자 장치(204)의 특정 자원으로의 트래핑된 시스템 호출은 메모리의 특정 부분에서 발생되었을 수 있다. SVMM 보안 에이전트(217)는 O/S 내 보안 에이전트(218)와 통신하여 메모리의 그 특정 부분 안에 어떤 애플리케이션, 프로세스, 또는 다른 개체가 상주하는지를 판단할 수 있다.

[0076] SVMM 보안 규칙들(222) 및 O/S 내 보안 에이전트(218)로부터 트래핑된 동작 및/또는 정황 정보에 기반하여, SVMM 보안 에이전트(217)는 이제 그러한 액세스가 악성 소프트웨어 감염을 나타내는 것들과 같은 의심스러운 액션을 이루는지 여부를 판단할 수 있다. 예를 들어 미인가 애플리케이션에 의해 보호되는 메모리 공간의 시스템 메모리(228)에 대한 변경 시도는 의심스러운 활동일 수 있으며, 그에 따라 SVMM(216)에 의해 검출된 그러한 변경 시도는 SVMM 보안 에이전트(217)에 의해 악성 소프트웨어의 한 동작이라고 해석될 수 있다. 그러한 활동은 추가 명령을 위해 보호 서버(202)로 보고될 수 있고, 아니면 O/S 내 보안 규칙들(220)에 의해 액션이 지시될 수 있다. 그러한 검출의 결과는 시스템 메모리(228)의 변경 시도를 차단하는 것이거나, 변경 시도를 발생했던 전자 장치(204)의 개체 상에 추가 청소(클리닝) 동작들을 유발하는 것일 수 있다.

[0077] SVMM(216)은 SVMM(216), SVMM 보안 에이전트(217) 및/또는 O/S 내 보안 에이전트(218)의 무결성을 보호하기 위해 시스템 자원들(214)에 대한 추가 호출들을 감시할 수 있다. SVMM(216)은 시스템 메모리의 일부가 악성 소프트웨어에 의해 변경되었는지 여부를 판단하기 위해 시스템 메모리(228)의 일부를 검색하도록 SVMM 보안 규칙들(222)에 의해 정의된 검색 동작들을 수행할 수 있다. SVMM(216)은 주어진 메모리 패턴이 안전하지 않다거나 안전하다고 알려진다는 것을 나타내는 서명들, 해시들 또는 기타 규칙들을 활용할 수 있다.

[0078] 예를 들어 SVMM(216)은 시스템 메모리(228) 내 O/S 내 보안 에이전트(218)에 해당하는 코드 및 데이터 페이지들에 대한 인가되지 않은 읽기 및 쓰기 액세스를 금지함으로써 O/S 내 보안 에이전트(218)를 보호할 수 있다. 어떤 악성 소프트웨어는 메모리 변경 또는 시스템 메모리(228)와 관련된 시스템 자원들(214)에 대한 기타 변경을 행함으로써 O/S 내 보안 에이전트(218)를 공격하고자 시도할 수 있다. SVMM(216)은 코드나 데이터 또는 O/S 내 보안 에이전트(218)에 해당하는 다른 시스템 자원들(214)을 변경하도록 허용될 수 있는 전자 장치(204)의 인가된 애플리케이션들 및 기타 개체들의 SVMM 보안 규칙들(222)에 포함된 화이트리스트를 판독할 수 있다. 변경이 화이트리스트 안에 포함되지 않은 개체로부터 발생했다면, SVMM(216)은 그러한 변경이 악성 소프트웨어와 관련

되어 있다고 판단할 수 있다. O/S 내 보안 에이전트(218)에 해당하는 시스템 자원들(214)에 대해 인가되지 않은 액세스는 액세스 차단, 하니팟(honeypot) 프로세스 생성, 보호 서버(202)에 대한 침해 보고, 또는 어떤 다른 적절한 처방을 포함하는 모든 적절한 방식으로 SVMM에 의해 처리될 수 있다.

[0079] SVMM(216)은 전자 장치(204)의 다른 개체들에 속하는 시스템 자원들(214)에 대한 액세스를 트래핑할 수도 있다. 예를 들어 시스템 메모리(228) 내 타깃 메모리 페이지는 운영체제(212)의 커널 동작의 일부에 속하는 샘플 코드나 데이터를 포함할 수 있다. SVMM(216) 및 SVMM 보안 규칙들(222)은 허가된 코드 섹션들에 대해서만 그러한 타깃 페이지로의 액세스를 제한할 수 있다. 결과적으로 시스템 메모리(228) 내 어느 코드 페이지가 타깃 메모리 페이지를 판독하거나 변경하고자 시도하고, 그 코드 페이지가 전자 장치(204)의 허가되지 않은 개체에 속하는 경우, 그러한 액세스는 SVMM(216)에 의해 차단될 수 있다. 따라서 SVMM(216)은 시스템 메모리(228) 내 메모리 페이지들로의 액세스를 통제하도록 작용할 수 있다.

[0080] SVMM 보안 에이전트(217)는 업데이트된 규칙들을 위해 보호 서버(202)를 접촉함으로써 SVMM 보안 규칙들(222) 및 O/S 내 보안 규칙들(220)을 업데이트할 수 있다. 보호 서버(202)는 관찰된 특정 악성 소프트웨어, 관리자 설정사항, 또는 전자 장치(204)의 다른 특징들에 기반하여 SVMM 보안 에이전트(217)로 전달될 규칙들을 구성할 수 있다. SVMM 보안 에이전트(217)는 사용자 요청, 또는 주기적으로, 또는 악성 소프트웨어와 연관될 수 있는 새로운 의심 활동들의 조우와 같은 의미 있는 이벤트의 발생에 따라 전자 장치(204)의 규칙들을 업데이트할 수 있다.

[0081] SVMM 보안 에이전트(217)는 여러 조건들에 해당하는 플래그들을 VMCS 안에서 설정할 수 있다. 그 플래그들은 트래핑될 각종 타입의 자원들을 아우를 수 있다. 예를 들어 VMCS는 메모리의 페이지로의 소정 값의 기입 및 I/O 장치의 버퍼로의 후속 페이지 이동의 조합을 트래핑하도록 구성될 수 있다.

[0082] 시스템(200)은 안티 악성 소프트웨어 시스템 및 소프트웨어의 다른 구현 예들에 비해 하나 이상의 이점들을 포함할 수 있다. 예를 들어 어떤 안티 악성 소프트웨어 솔루션들은 애플리케이션들의 저 레벨 동작들을 트래핑하고 평가하기 위해 운영체제의 다양한 부분들을 후킹할 수 있다. 그러나 이러한 솔루션들 자체는 운영체제 내부나, 두 개의 게스트 운영체제들의 경우 다른 운영체제 안에서 동작할 수 있다. 운영체제의 범위 내, 심지어 커널 레벨 우선순위에서 동작함으로써, 안티 악성 소프트웨어 솔루션은 동일한 운영체제 상에서 역시 실행되는, 아마도 동일한 우선순위로 실행되는 악성 소프트웨어로부터의 악성 소프트웨어 공격들에 민감할 수 있다. 소정 이벤트들에 대한 트래핑이나 트리거링이 운영체제 레벨에서 수행되는 경우, 그러한 트래핑이나 트리거링은 운영체제에 대해 동일하거나 더 낮은 우선순위로 실행되는 악성 소프트웨어에 의해 피싱되거나, 후킹되거나, 리버스 엔지니어링되거나, 훼손되거나 그렇지 않으면 무너질 수 있다. 예를 들어 운영체제에서 악의적 후킹을 검출하고 제거하는 운영체제 상에서 실행되는 안티 악성 소프트웨어 솔루션이 동일한 우선순위로 실행되는 악성 소프트웨어에 의해 관찰될 수 있다. 다른 예에서, 소정 루틴의 동작을 검출하기 위해 필터 드라이버로서 등록한 안티 악성 소프트웨어 솔루션은 드라이버 스택 상에서 안티 악성 소프트웨어 솔루션보다 낮은 악성 필터 드라이버를 등록하는 악성 소프트웨어에 의해 무너질 수 있다. 마찬가지로, 소정 트래핑되거나 트리거된 이벤트들의 처리가 운영체제의 레벨에서 일어나면, 악성 소프트웨어는 그러한 처리에 영향을 미칠 수 있을 것이다. 예를 들어, 악성 소프트웨어는 안티 악성 소프트웨어 솔루션의 교정들을 취소하거나 심지어 안티 악성 소프트웨어 솔루션의 동작을 불능화할 수 있다.

[0083] 다른 예에서, 하이퍼바이저(hypervisor)들이 시스템 메모리(228)와 같은 시스템 자원들로의 액세스를 가상화하도록 작용할 수 있으나, 시스템 자원들로의 액세스를 조건적으로 보호할 수는 없고 그에 따라 보안 하이퍼바이저 역할을 할 수 있다. 그러한 하이퍼바이저들은 시스템 자원들에 대한 악성 활동들, 개체들, 또는 악의적으로 시도된 액세스를 식별하기 위한 보안 규칙들(222) 안의 동향 규칙들과 같은 안티 악성 소프트웨어 규칙들로의 액세스 권한을 가질 수 없다. 그러한 하이퍼바이저들은 운영체제와 동일한 우선순위에서 실행되는 악성 소프트웨어에 대해 취약할 수 있는 운영체제 자체들 안에서 실행될 수 있다. 그러한 하이퍼바이저들은 "Ring0 특권 모드"에서 실행될 수 없는데, 이는 그 모드가 하이퍼바이저에게 시스템 자원들에 대해 시도되는 너무 많은 액세스들을 인터셉트할 것을 요구할 것이기 대문이다. 하이퍼바이저는 게스트 운영체제의 모든 양태들을 가상화하는 일을 맡을 수 있으며, 그러한 가상화 요구들은 악의적 행동을 체크하기 위해 보안 규칙들을 동시에 액세스하기에는 너무 비용이 많이 들 수 있다.

[0084] 도 3은 악성 소프트웨어로부터 가상 머신 모니터 기반의 전자 장치 보호를 위한 방법(300)의 실시예이다. 단계 305에서, O/S 하위 보안 에이전트, O/S 내 보안 에이전트, 보호 서버 및 가상 머신 모니터의 식별 및 보안이 인증될 수 있다. 그러한 인증은 메모리에 위치된 각각의 이미지들, 암호화 해싱 또는 비밀 키들을 찾고 검증하는

것을 포함하는 어떤 적절한 방식을 통해 수행될 수 있다. 단계 305가 완료될 때까지 다른 단계들의 동작은 보류될 수 있다.

[0085] 단계 310에서 보안 규칙들을 판단하기 위해 보호 서버가 액세스될 수 있다. 그러한 보안 규칙들은 단계들 315-380에서 결정을 내리는 데 사용될 수 있다. 단계 315에서 가상 머신 모니터는 시스템 자원들에 대한 액세스를 트래핑하도록 지시될 수 있다. 그러한 액세스는 전자 장치 상에서 실행되는 애플리케이션들, 드라이버들 또는 운영체제들로부터 일어날 수 있다. 가상 머신 모니터는 전자 장치의 어떤 시스템 자원들이 감시되어야 하는지에 대해 지시될 수 있다. 가상 머신 모니터는 감시된 시스템 자원들에 대한 어떤 동작들이 트래핑되어야 하는지에 대해 지시될 수 있다. 예를 들어 시스템 메모리 상의 읽기, 쓰기 또는 실행 동작들이 트래핑될 수 있다. 다른 예에서, 레지스터들 상의 로딩이나 저장 동작들이 트래핑될 수 있다. 또 다른 예에서 I/O 장치들 상의 입력이나 출력 액션들이 트래핑될 수 있다.

[0086] 단계 320에서 트래핑될 그러한 동작들에 해당하는 플래그들이 가상 머신 제어 구조와 같은 제어 구조 안에서 세팅될 수 있다. 그러한 트래핑되는 동작들은 VM 빠져나가기를 생성할 수 있으며, 여기서 플래깅된 자원의 액세스 시 트리거된 이벤트가 생성된다. 단계 325에서 시스템 메모리가 가상 머신 모니터, O/S 내 보안 에이전트 및 O/S 하위 보안 에이전트에 대해 할당될 때, 그러한 메모리는 허가되지 않은 읽기 및 쓰기 동작들에 대해 보호될 수 있다.

[0087] 전자 장치는 단계들 330-340에서 시스템 자원들의 액세스에 대한 트래핑, 단계 345-355에서 악성 소프트웨어가 존재하는지에 대한 메모리 검색, 단계 360-365에서 시도된 메모리 변경에 대한 메모리 검색 중 하나 이상에 의해 동작 및 보호될 수 있다. 시스템 자원들에 대한 액세스 트래핑, 악성 소프트웨어 존재에 대한 메모리 검색 및 시도된 메모리 변경들에 대한 메모리 검색 각각이 나란히 수행될 수 있다. 더 나아가, 이들 각각은 전자 장치의 동작을 보호하는데 필요할 때마다 반복될 수 있다.

[0088] 단계 330에서 시스템 메모리, 레지스터들 또는 I/O 장치들과 같은 시스템 자원에 대한 액세스가 트래핑될 수 있다. 액세스는 VM 빠져나가기를 생성하는 VMCS 플래그를 이용하여 트래핑될 수 있다. 그러한 트래핑은 전자 장치 상에서 실행되는 운영체제들의 레벨 아래에서 수행될 수 있다. 단계 335에서, 요청하는 개체가 요청된 자원을 액세스하기 위한 허가권을 가지는지 여부를 판단하도록 액세스가 분석될 수 있다. 시도된 액세스와 관련된 정황적 정보가 그러한 판단을 내리기 위해 액세스될 수 있다. 그러한 판단을 내리기 위해 보안 규칙들이 액세스될 수 있다. 허가되지 않은 액세스는 의심스러운 것이라 판단될 수 있다. 그러한 처리 및 판단은 전자 장치 상에서 실행되는 운영체제들의 레벨 아래에서 이루어질 수 있다. 해당 액세스가 의심스러운 경우, 단계 340에서 시스템 자원들에 대해 시도된 의심스러운 액세스는 차단될 수 있다. 그러한 시도는 보호 서버에 보고될 수 있다. 해당 액세스가 의심스럽지 않으면, 단계 370에서 해당 액세스가 허용될 수 있다.

[0089] 단계 345에서 악성 소프트웨어 존재에 대해 전자 장치의 메모리 페이지들이 검색될 수 있다. 전자 장치의 메모리를 검색할 때, 전자 장치 상에 상주하는 개체들을 반영한 메모리의 패턴들이 안전한 것으로 알려지는지 여부를 판단하기 위해 화이트리스트가 사용될 수 있다. 안전하다고 알려진 메모리의 패턴이 보이면, 단계 370에서 그 메모리는 전자 장치에 대해 계속 액세스하도록 허용될 수 있고 그대로 유지될 수 있다. 전자 장치의 메모리를 검색할 때, 메모리의 패턴들이 악성 소프트웨어를 포함하거나 관련된 것으로 알려지는지 여부를 판단하기 위해 블랙리스트가 사용될 수 있다. 화이트리스트 및 블랙리스트는 보안 규칙들을 액세스하여 액세스될 수 있다. 단계 350에서, 악성 소프트웨어와 관련된 것으로 알려진 메모리의 패턴이 발견되면, 단계 375에서 그 메모리의 패턴은 복구되거나 제거되거나 중립화됨으로써 전자 장치에 대해 거부되는 액세스일 수 있다.

[0090] 단계 355에서 메모리에 대한 변경이 시도되었었는지 혹은 시도되고 있는지 여부를 판단하기 위해 메모리가 검색될 수 있다. 그러한 검색은 전자 장치 내 운영체제들의 레벨 아래에서 수행될 수 있다. 그러한 메모리는 커널 메모리, 시스템 데이터 구조들 또는 악성 소프트웨어에 의해 변경될 수 있는 전자 장치의 메모리의 어떤 다른 일부를 포함할 수 있다. 예를 들어 전자 장치 상에서 실행되는 액티브 스레드들의 리스트가 악성 프로세스의 존재를 감추도록 변경될 수 있다. 어떤 변경이 검출되면, 단계 365에서 그러한 변경이 허용될 수 있는지 여부가 판단될 수 있다. 그러한 변경이 허용가능한지 여부는 보안 규칙들에 의해 정의될 수 있다. 예를 들어 안티 악성 소프트웨어 프로세스의 코드나 데이터 페이지가 어떤 다른 프로세스에 의한 변경이나 액세스에 대해 보호될 수 있다. 메모리 변경이 허가된 것이라 간주되면, 단계 370에서 그 변경은 허용될 수 있다. 메모리 변경이 허가되지 않은 것이라 허용되지 않는 것으로 판단되면, 단계 375에서 해당 변경이 거부될 수 있다.

[0091] 단계 370에서 액세스나 변경이 허용되면, 그 액세스나 변경이 추후 참조를 위해 저장될 수 있다. 악성 소프트웨어의 어떤 검출들은 지난 액세스 및 현재 검출된 액세스가 모두 자원에 대한 악의적 액세스를 포함하는지 여

부를 판단하기 위해 지난 액세스들이나 변경들을 이용할 수 있다.

[0092] 단계 375에서, 변경, 액세스 또는 다른 동작이 거부되는 경우, 그러한 이벤트는 단계 380에서 보호 서버로 보고될 수 있다. 그러한 보고는 어떤 관련된 악성 소프트웨어나 의심스러운 동향에 관한 정보를 포함할 수 있다.

[0093] 지속적으로, 또는 주기적으로, 또는 요청이 있을 때 전자 장치를 보호하기 위해 필요할 때 방법(300)의 단계들이 반복될 수 있다.

[0094] 도 4는 악성 소프트웨어로부터 전자 장치(404)를 보호하기 위한 펌웨어 기반 및 보안 규칙 기반 시스템(400)의 실시예이다. 시스템(400)은 시스템(100)의 실시예일 수 있으며, 여기서 시스템(100)의 어떤 구성요소들은 펌웨어로 구현될 수 있다. 시스템(400)의 트래핑 동작들은 전자 장치(404)의 운영체제들의 레벨 밑에서 수행될 수 있다. 시스템(400)은 전자 장치(404)의 자원들을 사용하거나 액세스하기 위해 I/O 명령들과 같은 요청들을 트래핑하도록 구성된 하나 이상의 O/S 하위 보안 에이전트들을 포함할 수 있다. 그러한 O/S 하위 보안 에이전트들은 장치들 간에, 혹은 전자 장치(404)의 메인 프로세서와 입력 및 출력 데이터 교환을 관리하도록 구성될 수 있다. 그러한 O/S 하위 보안 에이전트들은 전자 장치(404)의 장치 제어기들과 같은 구성요소들의 펌웨어나 전자 장치(404) 자체의 펌웨어를 통해 실시될 수 있다. 그러한 펌웨어는 비휘발성 메모리 안에 상주할 수 있다. 전자 장치(404)의 그러한 자원들은 도 1의 시스템 자원들(106) 또는 그것의 다양한 가능 실시예들, 또는 시스템(400) 내 장치들에 의해 연결되거나 실시되는 자원들을 포함할 수 있다. 시스템(400)은 전자 장치(404)의 자원들에 대해 시도된 액세스 이용을 트래핑하고, 그 시도에 해당하는 트리거된 이벤트(유발 이벤트)를 생성하고, 그 트리거된 이벤트에 관한 보안 규칙들을 조회하며 필요 시 상기 시도에 관한 교정 액션을 수행하도록 구성된 하나 이상의 O/S 보안 에이전트들을 포함할 수 있다.

[0095] 일 실시예에서 시스템(400)의 O/S 하위 보안 에이전트들은 이하에 기술되고 도 5의 논의들 속에 있는 전자 장치(404)의 구성요소들의 펌웨어를 통해서만 실시될 수 있다. 다른 실시예에서, 시스템(400)의 O/S 하위 보안 에이전트들은 메인 PC 펌웨어(428)와 같은 전자 장치(404) 자체의 펌웨어를 통해 실시될 수 있다. 그러한 실시예에서 메인 PC 펌웨어(428)는 전자 장치(404)의 마더보드 상에서 구현될 수 있다. 또 다른 실시예에서 시스템(400)의 O/S 하위 보안 에이전트는 O/S 하위 에이전트(450)에서도 역시 실시될 수 있다. O/S 하위 에이전트(450)는 운영체제(412)와 같은 전자 장치(404)의 운영체제들의 레벨 아래에서 자원들에 대한 액세스의 트리거링을 제공하거나 그러한 트리거들을 처리하기 위한 어떤 적절한 방식으로 구현될 수 있다. 예를 들어 O/S 하위 에이전트(450)는 도 2의 SVMM(216) 또는 SVMM 보안 에이전트(217)의 실시예일 수 있다. O/S 하위 에이전트(450)는 보안 규칙들(422)을 포함할 수 있다.

[0096] 전자 장치(404)는 전자 장치(404)로부터 입력 및 출력 동작들을 수행하기 위한 하나 이상의 구성요소들을 포함할 수 있다. 전자 장치(404)는 어떤 적절한 수의 그러한 구성요소들 및 구성요소들의 타입들을 포함할 수 있다. 그러한 구성요소들은 그들 자체 프로세서, 메모리 및 펌웨어로 내장된 소프트웨어를 가진 장치들에 의해 구현될 수 있다. 그러한 구성요소의 실시예가 도 5의 I/O 장치(502)일 수 있다.

[0097] 전자 장치(404)는 예컨대 디스플레이(424) 및 스토리지(426)를 포함할 수 있다. 각각의 그러한 구성요소(424, 426)는 펌웨어(430, 432)를 포함할 수 있다. 펌웨어(430, 432)는 각각 도 5의 펌웨어(504)를 구현할 수 있다. 상술한 바와 같이 각각의 그러한 구성요소(424, 426)는 펌웨어 보안 에이전트(440, 442)와 같은 펌웨어 기반의 보안 에이전트를 포함할 수 있다. 펌웨어 보안 에이전트들(440, 442)은 각각 부분적으로나 전체적으로 도 5의 펌웨어 보안 에이전트(516)를 실시할 수 있다. 일 실시예에서 펌웨어 보안 에이전트들(440, 442) 각각은 그들 각자의 펌웨어(430, 432)를 통해 구현될 수 있다. 다른 실시예에서 펌웨어 보안 에이전트들(440, 442) 각각은 그들 각자의 구성요소들(424, 426) 각각 안에 있는 펌웨어(430, 430) 외부에서 구현될 수 있다. 그러한 장치 펌웨어 보안 에이전트들(440, 442) 각각은 각자의 보안 규칙들(434, 436)의 세트와 통신 가능하게 연결될 수 있다. 각각의 그러한 보안 규칙들(434, 436)은 도 5의 보안 규칙들(518)을 실시할 수 있다.

[0098] 전자 장치(404)는 펌웨어를 포함할 수 있다. 일 실시예에서 전자 장치(404)는 메인 PC 펌웨어(428)를 포함할 수 있다. 메인 PC 펌웨어(428)는 베이직 입력/출력 시스템("BIOS")에 의해 실시될 수 있다. 일 실시예에서 메인 PC 펌웨어(428)는 컴퓨터의 BIOS로서 구성될 수 있다. 그러한 경우들에서, 메인 PC 펌웨어(428)는 컴퓨터의 프로세서(406) 동작을 초기화하도록 구성될 수 있다. 메인 PC 펌웨어(428)는 메인 프로세서(406)가 디스플레이(424) 및 스토리지(426)와 같은 I/O 장치들과 통신할 수 있게 구성될 수 있다. 그러한 실시예들에서 컴퓨터는 펌웨어나 BIOS에 의해 프로그램 될 수 있는 프로그래머블 I/O 제어기를 또한 포함할 수 있고 424와 같은 I/O 장치들 및 스토리지(426)의 펌웨어와 통신할 수 있다.

[0099] 메인 PC 펌웨어(428)는 O/S 하위 보안 에이전트를 포함할 수 있다. 일 실시예에서 메인 PC 펌웨어(428)는 PC 펌웨어 보안 에이전트(444)를 포함할 수 있다. PC 펌웨어 보안 에이전트(444)는 시스템 자원들(414)에 대한 요청들을 인터셉트하도록 구성될 수 있다. 그러한 기능을 수행하기 위해, PC 펌웨어 보안 에이전트(444)는 도 2의 SVMM 보안 에이전트(217)나 SVMM(216) 및/또는 도 5의 펌웨어 보안 에이전트(516)의 기능을 전부나 일부 실시할 수 있다. PC 펌웨어 보안 에이전트(444)는 시스템 자원들(414)에 대한 액세스, O/S 내 보안 에이전트(418)와 같은 O/S 내 보안 에이전트들과 O/S 하위 보안 에이전트들의 검증 및 확인, 그리고 보안 규칙들(420, 422)과 같은 보안 규칙들의 배포에 대한 O/S 하위 트리거링 및 핸들링을 수행하기 위해 도 2의 SVMM 보안 에이전트(217)나 SVMM(216)의 기능을 시행할 수 있다. PC 펌웨어 보안 에이전트(444)는 펌웨어를 통한 O/S 하위 트리거링 및 핸들링을 수행하고 보안 규칙들을 업데이트하며 전자 장치(404)의 일부로 보내지는 IN 및 OUT 명령들을 평가하기 위해 도 5의 펌웨어 보안 에이전트(516)의 기능을 실시할 수 있다.

[0100] 전자 장치(404)는 보안 규칙들(438)을 포함할 수 있다. 보안 규칙들(438)은 도 1의 보안 규칙들(114)의 일 실시예일 수 있다. 일 실시예에서 보안 규칙들(438)은 메인 PC 펌웨어(428)에 상주할 수 있다. 다른 실시예에서 보안 규칙들(438)은 메인 PC 펌웨어(428) 외부에 상주할 수 있고 PC 펌웨어 보안 에이전트(444)는 보안 규칙들(438)에 연결될 수 있다.

[0101] 시스템(400)의 보안 에이전트들은 악성 소프트웨어 및 그것의 악의적 동작들을 방지하기 위해 함께 작업하도록 구성될 수 있다. 자원들에 대해 시도되는 액세스가 트래핑될 수 있고, 디스플레이(424)나 스토리지(426)와 같은 장치들 안이나 메인 PC 펌웨어(428) 안의 펌웨어 보안 에이전트들에서의 처리를 위해 후속 이벤트들이 트리거 될 수 있다. 그러한 장치들이나 펌웨어 안의 펌웨어 보안 에이전트들은 트리거된 이벤트들을 처리하거나 그 트리거된 이벤트를 처리를 위해 다른 보안 에이전트로 보내도록 구성될 수 있다. 제한된 실행 및 업데이트 기능들로 인해, 일부 펌웨어 보안 에이전트들은 그들 자신의 트리거된 이벤트들을 처리 시 제한될 수 있고, 그에 따라 그렇게 트리거된 이벤트들을 다른 보안 에이전트들로 전달하는 것이 유리할 수 있다. 펌웨어 보안 에이전트들이 이벤트들을 보낼 보안 에이전트들은 예컨대 O/S 내 보안 에이전트(418)와 같은 O/S 내 보안 에이전트들, O/S 하위 보안 에이전트(450)와 같은 O/S 하위 보안 에이전트, 또는 PC 펌웨어 보안 에이전트(444)와 같은 다른 펌웨어 보안 에이전트를 포함할 수 있다. 이러한 다른 보안 에이전트들은 트리거된 이벤트를 수신하고, 보안 규칙들, 정황적 정보, 또는 허가들을 조회하고 구현될 결과적 액션을 다시 보내도록 구성될 수 있다.

[0102] 그에 따라, 도 4는 펌웨어 기반의 보안 에이전트들에 의해 O/S 하위 트리거링 및 핸들링(처리)을 수행하기 위해 여러 구성요소들의 예들을 도시하고 있지만, 더 많거나 더 적은 구성요소들이 다양한 실시예들에서 사용될 수 있다. 더 많거나 더 적은 구성요소들이 사용될 때, 각각의 구성요소 및 시스템(400)의 기능은 그에 따라 달라질 수 있다. 일 실시예에서 운영체제(412)의 레벨 아래에 있는 시스템(400)의 보안 에이전트들은 하나 이상의 O/S 내 보안 에이전트들(418) 및 펌웨어 보안 에이전트들(440, 442)로 한정될 수 있다. 그러한 예에서, 펌웨어 보안 에이전트들(440, 442)은 보안 규칙들(434, 436)에 대한 업데이트들을 위해 보호 서버(402)에 의존할 수 있다. 펌웨어 보안 에이전트들(440, 442)은 트리거된 이벤트들의 업데이트들이나 핸들링을 위해 O/S 내 보안 에이전트(418)에 의존할 수 있지만, O/S 내 보안 에이전트(418)의 동작은 O/S 하위 보안 에이전트가 O/S 내 보안 에이전트를 비준하지 않으면 덜 안전할 수 있다. 펌웨어 보안 에이전트들(440, 442)은 설치, 제조 또는 구성 시 설정되는 펌웨어 보안 규칙들(434)에 기반하여 트리거링을 제공할 수 있다. 그러한 보안 규칙들은 상대적으로 정적일 수 있다. 그 경우, 펌웨어 보안 에이전트들(440, 442)은 분석이 거의 없이 상대적으로 기본적인 이벤트 트리거링을 제공하도록 구성될 수 있다. 그러한 펌웨어 보안 에이전트들(440, 442)은 그럼에도 불구하고 유용할 수 있는데, 이는 그러한 트리거링이 전자 장치(404)의 운영체제들 이하에서 수행되고 그에 따라 어떤 악의적이거나 의심스러운 동작들을 보다 잘 검출할 수 있다.

[0103] 다른 실시예에서 시스템(400)의 보안 에이전트들은 PC 펌웨어 보안 에이전트(444)나 O/S 하위 에이전트(450)(둘 다는 아님)을 포함할 수 있다. 그 경우, PC 펌웨어 보안 에이전트(444)의 기능은 O/S 하위 에이전트(450)에 의해 구현될 수 있으며, 그 반대의 경우도 성립된다. PC 펌웨어 에이전트(444)나 O/S 하위 에이전트(450)는 보호 서버(402)와 연결될 수 있고, 보안 규칙들(420, 422, 438, 434, 436)과 같은 정보를 획득하고 그러한 정보를 시스템(400) 내 다른 보안 에이전트들과 공유하도록 구성될 수 있다. 그러한 보안 규칙들은 통신, 업데이트, 또는 저장 비용의 목적 상, 각각 각자의 보안 에이전트에 꼭 맞춤 될 수 있다. PC 펌웨어 에이전트(444)나 O/S 하위 에이전트(450)는 펌웨어 보안 에이전트들(440, 442)과 같은 다른 보안 에이전트들로부터 트리거된 이벤트들을 수신하고, 보안 규칙들 및 기타 정보를 적용하며, 결과적 이벤트를 펌웨어 보안 에이전트들(440, 442)로, 혹은 정보를 보호 서버(402)로 보내는 것과 같은 교정적 액션을 수행하도록 구성될 수 있다. PC 펌웨어 에이전트(444)나 O/S 하위 에이전트(450)는 시스템 자원들(414)의 시도된 액세스들을 트래핑하도록 구성될 수 있다.

PC 펌웨어 에이전트(444)나 O/S 하위 에이전트(450)는 트리거된 이벤트들의 정황을 판단하기 위해 O/S 보안 에이전트(418)와 통신하도록 구성될 수 있다. 하나를 넘는 O/S 내 보안 에이전트(418)가 시스템(400) 안에 존재하는 경우, 각각의 O/S 내 보안 에이전트(418)는 트래핑, 검증(validating) 또는 O/S 내 보안 에이전트(418)와 관련된 다른 작업들의 지정된 일부를 수행하도록 구성될 수 있다. 그러한 일부는 운영체제 하위 시스템 보안 에이전트들에 의해 정의될 수 있다. 예를 들어 하나의 O/S 내 보안 에이전트(418)가 MOV 명령어들을 검증하거나 검사할 수 있고, 또 다른 O/S 내 보안 에이전트(418)가 JMP 명령어들을 검증하거나 검사할 수 있다.

[0104] 또 다른 실시예에서 시스템(400)의 보안 에이전트들은 PC 펌웨어 보안 에이전트(444) 및 O/S 하위 에이전트(450) 모두를 포함할 수 있다. 그럼에도 불구하고 그러한 실시예에서, PC 펌웨어 보안 에이전트(444)의 기능의 일부나 전부는 O/S 하위 에이전트(450)에 의해 구현될 수 있으며, 그 반대의 경우도 성립된다. PC 펌웨어 보안 에이전트(444) 및 O/S 하위 에이전트(450) 사이의 작업에 대한 서술은 여러 요인들을 참작할 수 있다. 예를 들어 PC 펌웨어 보안 에이전트(444)와 같은 펌웨어 안에서의 보안 에이전트의 동작은 다른 O/S 하위 에이전트(450)의 동작보다 더 안전할 수 있다. 그러나 보안 규칙들 및 O/S 하위 에이전트(450)의 소프트웨어의 업데이트는 PC 펌웨어 보안 에이전트(444)보다 단순하고 빠를 수 있다.

[0105] 또 다른 실시예에서 하나 이상의 펌웨어 보안 에이전트들이 PC 펌웨어 보안 에이전트(444)나 운영체제 하위 에이전트(422)와 독립적으로 시스템(400) 상에 상주할 수 있다. 그러한 예에서, 펌웨어 보안 에이전트들(440, 442)은 운영체제 내 보안 에이전트(418)의 인스턴스를 비준할 수 있다.

[0106] 펌웨어 보안 에이전트들(440, 442, 444) 각각은 외부 통신을 위해 펌웨어 로직을 감시하고 제어할 수 있기 충분한 펌웨어 로직 안에 상주하도록 구성될 수 있이다. 펌웨어 보안 에이전트들(440, 442, 444)은 그에 따라 특정한 다른 에이전트들과의 특정 정보에 대한 통신을 트래핑하도록 구성될 수 있다. 펌웨어 보안 에이전트들(440, 442, 444)은 수신된 동작 요청뿐 아니라 전송되거나 수신될 데이터를 판단하도록 구성될 수 있다. 또한 펌웨어 보안 에이전트들(440, 442, 444)은 전송되거나 수신될 데이터를 제어하도록 구성될 수 있고, 데이터의 암호화, 압축, 워터마크 삽입 또는 워터마크 디코딩과 같이 데이터에 대한 추가 동작들을 일으키도록 구성될 수 있다. 펌웨어 보안 에이전트들(440, 442, 444)과 통신하는 시스템(400의 다른 보안 에이전트들은 펌웨어 보안 에이전트들(440, 442, 444)에 의해 트래핑될 데이터에 워터마크들을 삽입하거나 펌웨어 보안 에이전트들(440, 442, 44)에 의해 데이터 안에 넣어진 워터마크들을 디코딩하도록 구성될 수 있다.

[0107] 펌웨어 보안 에이전트(440, 442)나 PC 펌웨어 보안 에이전트(444)와의 통신은 예컨대 프로그래머블 입/출력 인터럽트들이나 프로그래머블 입/출력 레지스터들을 통해 수행될 수 있다. 그러한 인터럽트들이나 레지스터들은 펌웨어 보안 에이전트(440, 442, 444)가 상주하는 펌웨어나 장치의 마커에 의해 정의 및 제공될 수 있다.

[0108] 시스템(400)의 O/S 하위 보안 에이전트들 중 하나 이상은 전자 장치(404)의 펌웨어 기반 보안 에이전트들의 안티 악성 소프트웨어 활동들을 조정할 메인 보안 에이전트로서 기능하도록 구성될 수 있다. 일 실시예에서 PC 펌웨어 보안 에이전트(444)는 시스템(400)의 메인 보안 에이전트로서 구성될 수 있다. 다른 실시예에서 O/S 하위 에이전트(450)는 메인 보안 에이전트의 역할을 하도록 구성될 수 있다. 보안 에이전트는 펌웨어 보안 에이전트들(440, 442)로부터 트리거된 이벤트들을 처리하도록 구성될 수 있다. 메인 보안 에이전트는 펌웨어 보안 에이전트들(440, 442) 뿐 아니라 O/S 내 보안 에이전트(418)와 같은 다른 보안 에이전트들의 동작을 비준하도록 구성될 수 있다. 메인 보안 에이전트는 보안 에이전트들이 의심스러운 동향을 인지했거나 악성 소프트웨어를 검출했는지 여부, 시스템(400)이 악성 소프트웨어 공격 상태에 있는지 여부, 혹은 시스템(400)의 관리자가 보안에 영향을 미치는 선호사항이나 설정사항을 변경했는지 여부에 대해 다른 보안 에이전트들에게 알리도록 구성될 수 있다. 메인 보안 에이전트는 공격에 관한 정보를 시스템(400)의 다른 보안 에이전트들과 공유할 수 있다.

[0109] 시스템(400)의 지원들에 대한 액세스를 트래핑하고/하거나 시스템(400)의 운영체제들의 레벨 아래에서 트리거된 결과적 이벤트들을 처리함으로써, 시스템(400)이 악성 소프트웨어에 대항하여 강화된 보안을 제공할 수 있다. 펌웨어 내 보안 에이전트의 동작은 악성 소프트웨어가 보안 에이전트의 동작에 영향을 미칠 기회를 줄일 수 있다. 펌웨어나 장치 레벨의 트래핑 동작들은 자체 동작을 위장하기 위해 시스템(400)의 요소들을 스푸핑(spoof)하거나 피싱하는 악성 소프트웨어의 능력을 줄일 수 있다. 예를 들어 운영체제(412)의 어떤 부분이 악성 소프트웨어에 의해 훼손되는지와 무관하게, 구성요소(424, 426)에 대한 요청이 장치 자체로부터 숨겨지지 못할 수도 있다.

[0110] 도 5는 악성 소프트웨어로부터 전자 장치를 보호하기 위한 펌웨어 기반 솔루션의 실시예에 대한 상세도이다. I/O 장치(502)와 같은 장치가 사용 요청들이나 장치의 자원들에 대한 액세스를 수신 및 트래핑하도록 구성될 수 있다. 일 실시예에서 I/O 장치(502)는 그 요청들이 악성 소프트웨어의 존재를 나타내는지 여부를 판단하기 위

해 그렇게 트래핑된 요청들을 처리하도록 구성될 수 있다. 다른 실시예에서 I/O 장치(502)는 트리거된 이벤트와 같은 트래핑된 요청을 I/O 장치가 상주하는 시스템의 다른 부분으로 전달하도록 구성될 수 있다. 그러한 시스템의 다른 부분은 O/S 하위 보안 에이전트를 포함할 수 있다. I/O 장치(502)는 펌웨어(504) 및 메모리(508)와 연결된 프로세서(506)를 포함할 수 있고, 이때 펌웨어(504)는 프로세서(506)에 의해 실행되기 위해 메모리(508)에 상주하는 명령어들을 포함할 수 있다.

[0111] I/O 장치(502)는 전자 장치의 자원에 대한 액세스를 제어하기 위한 전자 장치의 어떤 적절한 부분을 포함할 수 있다. 일 실시예에서 I/O 장치(502)는 전자 장치의 주변기기의 일부나 전체를 구현할 수 있다. I/O 장치(502)는 예컨대 디스플레이 제어기 카드, 컴퓨터 버스 제어기, 캐시 장치, I/O 제어기 장치, 디스크 제어기, 메모리 장치, 네트워크 제어기, 마더보드, 또는 키보드 제어기에 의해 구현될 수 있다. I/O 장치(502)는 전자 장치 안에 상주할 수 있다. 일 실시예에서 I/O 장치(502)는 물리적 구성요소들과 연결될 수 있다. 그러한 물리적 구성요소들은 다만 예들로서, 디스플레이, 컴퓨터 버스, 메모리, I/O 제어기들, 디스크, 네트워크 카드, 또는 키보드를 포함할 수 있다. 다른 실시예에서 I/O 장치(502)는 연결된 물리적 구성요소들과 개별적으로 존재할 수 있다. 예를 들어 키보드 제어기는 직렬 인터페이스를 통해 키보드와 연결될 수 있다. 그러한 실시예들에서는 I/O 장치(502)는 전자 장치 안에 상주할 수 있지만, 그러한 물리적 구성요소들이 전자 장치와 통신 가능하게 연결되지만 전자 장치 밖에 상주할 수 있다.

[0112] 펌웨어(504)는 I/O 장치(502)의 동작을 제어하도록 구성될 수 있다. 펌웨어(504)는 자원들에 대한 요청들을 트래핑하고 I/O 장치(502)나 I/O 장치(502)가 상주하는 시스템들 안의 운영체제들의 레벨 밑에서 동작하도록 구성된 O/S 하위 보안 에이전트(516)를 포함할 수 있다. O/S 하위 보안 에이전트(516)는 I/O 장치(512)나 I/O 장치(502)가 상주하는 시스템들을 악성 소프트웨어로부터 보호하기 위해, 트래핑된 요청들로부터 비롯된 이벤트들을 처리하여 그 요청을 허용할지 거부할지 그렇지 않으면 처리할지 여부를 판단하도록 구성될 수 있다. 일 실시예에서 펌웨어(504)는 펌웨어 보안 에이전트(516)를 포함할 수 있다. 펌웨어 보안 에이전트(516)는 도 2의 SVMM(216)이나 SVMM 보안 에이전트(217)의 기능의 일부나 전체를 병합할 수 있지만 펌웨어(504) 안에서 구현된다. 그러한 경우, 자원들에 대한 액세스 트래핑 및/또는 트래핑된 요청을 처리하는 것과 같은 SVMM(216)이나 SVMM 보안 에이전트(217)의 기능이 펌웨어 보안 에이전트(516)에 의해 수행될 수 있다. 일 실시예에서 펌웨어 보안 에이전트(516)는 펌웨어(504) 안에서 상주하도록 구성될 수 있다.

[0113] 펌웨어(504)는 I/O 명령들(510), 데이터 전송 엔진(512) 및 프로그래밍 로직(514)을 포함할 수 있다. I/O 명령들(510)은 장치로 명령을 보내거나 수신하기 위한 명령어들을 포함할 수 있다. 그러한 명령들은 IN 혹은 OUT 명령들의 변형들을 포함할 수 있다. I/O 명령들(510)의 실행은 장치의 원하는 액션들을 수행하기 위해 이용될 수 있다. 장치에 의해 수신된 요청들이 I/O 명령들로 변환될 수 있다. 자원들에 대한 특정 요청들에 대한 트래핑이나 트리거링은 관련 I/O 명령들(510)에 따른 트래핑이나 트리거링을 통해 수행될 수 있다. 데이터 전송 엔진(512)은 장치에 대한 요청들 및 뒤이은 응답들에 대한 전송을 처리하도록 구성될 수 있다. 데이터 전송 엔진(512)은 I/O 명령들(510) 및 데이터가 교환되는 I/O 버스를 통해 프로세서(506) 및 프로그래머블 I/O 제어기에 연결될 수 있다. 프로그래머블 로직(514)은 펌웨어(504)가 I/O 명령들(510) 및 데이터 전송 엔진(512)을 동작하게 하는 명령들을 제공하도록 구성될 수 있다. 프로그래밍 로직(514)은 프로세서(506)와 같은 프로세서 안에 로딩될 수 있다.

[0114] 펌웨어 보안 에이전트(516)는 시도된 악의적 동작들을 검출하기 위해 프로그래머블 로직(514)의 동작을 변경하도록 구성될 수 있다. 펌웨어 보안 에이전트(516)는 또한 데이터 전송 엔진(512)을 통한 I/O 장치(502)의 요청들을 인터셉트하기 위해 장치에 대한 요청들의 전송을 감시하고 그러한 요청들이 악의적인지 여부를 판단하도록 구성될 수도 있다. 펌웨어 보안 에이전트(516)는 트래핑되어야 하는 동작들에 상응하는 플래그들이 세팅되는 제어 구조를 포함할 수 있다. 일 실시예에서 플래그들은 트래핑되어야 하는 명령들의 메모리 어드레스에 따라 상기 구조 안에서 세팅될 수 있다. 펌웨어 보안 에이전트(516)는 I/O 장치(502)에 대한 요청들의 인터셉트를 위해 플래그들을 세팅하도록 구성될 수 있다. 그러한 플래그들은 예컨대 I/O 명령들(510)의 특정 명령들, 또는 특정 파라미터들과 조합된 그러한 특정 명령들에 대응할 수 있다. 그러한 플래그들은 특정 요청들이나 요청들의 카테고리들을 인터셉트하도록 구성될 수 있다. I/O 명령(510)의 트래핑된 시도 동작에 대응하는 특정 플래그의 트리거링에 따라, 펌웨어 보안 에이전트(516)는 그 이벤트를 처리하고 결과적 액션을 취하며, 결과 정보를 데이터 전송 엔진(512)을 통해 다른 보안 에이전트로 전달하거나 데이터 전송 엔진(512)을 통해 상기 트리거된 이벤트를 전달하도록 구성될 수 있다.

[0115] I/O 장치(502) 역시 보안 규칙들(518)을 포함할 수 있다. 보안 규칙들(518)은 도 2의 보안 규칙들(222)의 일부 또는 전부를 구현할 수 있다. 보안 규칙들(518)은 메모리(508) 안에 구현될 수 있다. 일 실시예에서 보안 규

칙들(518)은 펌웨어(504)의 외부에 상주할 수 있다. 다른 실시예에서 보안 규칙들(518)은 펌웨어(504)의 내부에 상주할 수 있다. 펌웨어 보안 에이전트(516)는 보안 규칙들(518)과 통신 가능하게 결합되며, 자신의 자원들에 대한 액세스를 위해 I/O 장치(502)에 대해 이루어진 요청들이나 요청들의 카테고리들을 트래핑하려면 어떤 플래그들이 펌웨어(504) 안에서 세팅되어야 하는지를 판단하기 위해 보안 규칙들(518)을 액세스하도록 구성된다. 예를 들어 펌웨어 보안 에이전트(516)는 트리거된 이벤트가 악의적인지 아닌지 여부를 판단하기 위해 보안 규칙들(518)을 액세스하도록 구성될 수 있다. 일 실시예에서 보안 규칙들(518)은 펌웨어 보안 에이전트(518)가 트리거된 이벤트를 처리하게 하는 명령들을 포함할 수 있다. 펌웨어 보안 에이전트(516)는 그 요청을 허용할지 거부할지 혹은 다른 교정 액션을 취할지 여부를 판단하기 위해 그러한 명령들을 이용하도록 구성될 수 있다. 다른 실시예에서 펌웨어 보안 에이전트(516)는 그 요청을 다른 보안 에이전트에 보고할지 여부를 판단하기 위해 그러한 명령들을 이용하도록 구성될 수 있다. 상기 교정 액션들은 그 요청을 허용할지 거부할지 여부에 대한 명령들을 포함할 수 있는 다른 보안 에이전트로부터 응답을 대기하는 동작을 또한 포함할 수 있다.

[0116] 어떤 실시예들에서 펌웨어 보안 에이전트(516)는 펌웨어 보안 에이전트(516) 업데이트를 상대적으로 어렵게 할 수 있는 펌웨어(504) 안에 상주할 수 있다. 또한, 변화하는 특성의 악성 소프트웨어 공격들은 안티 악성 소프트웨어 솔루션들이 융통성이 있을 것을 요할 수 있다. 결과적으로 펌웨어 보안 에이전트(516)는 I/O 장치에 대한 어떤 요청들을 트래핑하고 어떤 후속 액션들을 취할지를 결정하기 위한 정보를 수신하기 위한 어떤 적절한 메커니즘을 이용할 수 있다.

[0117] 그러한 하나의 실시예에서 그러한 메커니즘은 상술한 바와 같은 보안 규칙(518) 액세스를 포함할 수 있다. 펌웨어 보안 에이전트(516)는 다른 보안 에이전트들이나 보호 서버들로부터 새롭고 업데이트된 보안 규칙들(518)을 수신하도록 구성될 수 있다. 융통성을 얻기 위해 펌웨어 보안 에이전트(516)는 예컨대 펌웨어(504) 내 보안 규칙들이 보안 규칙들(518)의 업데이트를 어렵게 할 경우, 펌웨어(504)와 분리된 메모리(508) 안의 보안 규칙들(518)을 저장하도록 구성될 수 있다.

[0118] 그러한 다른 실시예에서 펌웨어 보안 에이전트(516)는 펌웨어 업데이트나 플래시(flash) 시, 보안 규칙들(518)을 업데이트하도록 구성될 수 있다. 그러한 실시예에서 트래핑될 요청들의 융통성은 제한될 수 있다. 결과적으로 보안 규칙들(518)은 매우 특수한 보호 자원들에 대한 것일 수 있다. 예를 들어 디스크 장치의 보안 규칙들(518)은 그 장치의 부트 섹터에 대한 모든 쓰기(기입) 요청들을 트래핑하는 명령들을 포함할 수 있다. 어떤 경우, 다른 보안 에이전트들과의 통신이 저렴한 경우, 보안 규칙들(518)은 광범위한 요청들을 트래핑할 명령들을 포함할 수 있으며, 이때 프로세싱이 다량으로 다른 보안 에이전트들로 떠넘겨질 수 있다.

[0119] 또 다른 그러한 실시예에서 펌웨어 보안 에이전트(516)는 다른 보안 에이전트들로부터 명령들을 수신하도록 구성될 수 있다. 어떤 경우 그러한 명령들은 펌웨어(504)나 펌웨어 보안 에이전트(516)의 함수 호출들에 대한 파라미터들의 형태를 취할 수 있다. 예를 들어 다른 보안 에이전트가 "UpdateRule(*trigger, action*)"이라는 이름의 펌웨어 보안 에이전트(516)의 함수를 호출할 수 있고, 여기서 트래핑하라는 요청은 *trigger*에 상세히 기술되고 취해질 후속 액션은 *action*에 상세히 기술된다. 펌웨어 보안 에이전트(516)는 그에 따라 보안 규칙들의 업데이트에 관한 명령들을 수신하여 보안 규칙들(518)을 업데이트할 수 있다. 다른 경우, 다른 보안 에이전트가 보안 규칙들(518)의 업데이트를 장치(502)의 예비된 메모리 공간에 기입할 수 있고, 그것은 나중에 펌웨어 보안 에이전트(516)에 의해 액세스될 수 있다. 다른 보안 에이전트들로부터 수신될 명령들이 펌웨어 보안 에이전트(516)를 특정 보안 규칙들(518)의 세트를 사용하라고 지시할 수도 있다. 예를 들어 시간 임계적 동작 중에, 펌웨어 보안 에이전트(516)는 최소한의 코어 보안 규칙들(518)의 세트를 이용하라는 명령들에 따라 구성될 수 있다. I/O 장치(502)가 디스크 장치일 때, 그러한 최소한의 핵심 규칙들의 세트가 디스크의 부트 섹터에 대한 액세스를 트래핑하라는 명령들을 포함할 수 있다. 다른 예에서, 시간 임계적 동작들이 현재 수행되고 있지 않으면, 펌웨어 보안 에이전트(516)는 훨씬 더 광범위한 액세스 시도들을 트래핑하고 처리를 위해 대응 이벤트들을 다른 보안 에이전트들로 보내라는 보안 규칙들(518)로부터의 규칙들을 이용하도록 하는 명령들에 의해 구성될 수 있다.

[0120] 펌웨어 보안 에이전트(516)는 I/O 명령들(510)을 제어하고, 수신되거나 전송될 콘텐츠나 데이터를 검색하고, 명령들과 콘텐츠에 대한 액세스 통제를 적용하도록 구성될 수 있다. 펌웨어 보안 에이전트(516)는 기존 장치 펌웨어의 확장으로서 구현될 수 있다.

[0121] 펌웨어 보안 에이전트(516)의 구현은 장치(502) 타입에 따라 달라질 수 있다. 예를 들어 디스플레이 장치들 및 디스크 장치들은 다양한 종류의 콘텐츠나 시도된 명령들에 따라 트리거할 수 있다. 다양한 장치들에서 펌웨어 보안 에이전트들(516)의 생성은 장치와의 특정 종류의 인터페이스에 대해 맞춤화될 수 있다. 예를 들어 장치

(502)가 SATA(Serial Advanced Technology Attachment) 버스를 통해 통신하도록 구성된 경우, 그것에는 SATA 버스들을 통해 통신하는 다른 장치들과 유사한 펌웨어 보안 에이전트들(516)이 장착될 수 있다. 펌웨어 보안 에이전트(516)는 장치(502)의 구조를 지원하거나, 장치(502)의 외부 버스 I/O를 지원하거나, 장치(502)의 다른 인터페이스들을 지원하도록 맞춤화될 수 있다.

[0122] 펌웨어 보안 에이전트(516)는 자원 요청의 일부를 구성할 수 있는 특정 읽기 및 쓰기 명령들을 인터셉트함으로써 장치(502) 내 자원들에 대해 시도된 액세스를 트래핑하도록 구성될 수 있다. 읽기나 쓰기 명령은 보안 규칙들(518) 안에 있는 것과 같은 어떤 규칙에 기반하여 인터셉트되고, 평가되어 차단되거나 허용될 수 있다. 펌웨어 보안 에이전트(516)에 대한 보안 규칙들(518)은 악성 소프트웨어의 증거를 검출하기 위한 어떤 적절한 규칙들을 포함할 수 있다. 그러한 읽기 및 쓰기 명령은 예컨대 드라이버에 대한 함수 호출이나 인터럽트의 결과일 수 있다.

[0123] 예를 들어 보안 규칙들(518)은 펌웨어 보안 에이전트(516)가 장치에 기입될 데이터를 검색하게 하는 규칙들을 포함할 수 있다. 데이터의 콘텐츠 또는 데이터의 해시는 그 데이터가 악성 소프트웨어 데이터나 코드에 해당하는지 여부를 판단하기 위해 평가될 수 있다. 그러한 평가는 해당 콘텐츠를 화이트리스트나 블랙리스트 안의 데이터나 서명과 비교함으로써 이루어질 수 있다. 연속적인 기입은 기입될 콘텐츠나 데이터를 악성 소프트웨어나 악성 소프트웨어가 아닌 것으로서 올바로 식별하기 위해 그 데이터나 콘텐츠의 전체 범위를 적절히 평가하도록 함께 평가되어야 할 수 있다. 예를 들어 어떤 파일이 반복되는 연속 호출들을 통해 장치(502)에 기입될 수 있다. 기입될 데이터는 쓰기 명령의 콘텐츠들에 대한 적절한 검색이 평가될 수 있도록 대기(queue)될 수 있다.

[0124] 다른 예에서 보안 규칙들(518)은 펌웨어 보안 에이전트(516)가 장치 내 기존 데이터를 검색하게 하는 규칙들을 포함할 수 있다. 장치(502)는 네트워크 카드에서와 같이 시스템 외부로부터 수신되는 콘텐츠를 포함할 수 있다. 수신된 정보의 콘텐츠는 그것이 장치(502)에 속하기 때문에, 악성 소프트웨어의 증거를 찾아 검색될 수 있다. 펌웨어 보안 에이전트는 해당 콘텐츠를 화이트리스트나 블랙리스트 안의 데이터나 서명과 비교함으로써 평가를 수행할 수 있다.

[0125] 또 다른 예에서 보안 규칙들(518)은 펌웨어 보안 에이전트(516)가 시간이나 허락에 기반하여 명령을 평가하게 하는 규칙들을 포함할 수 있다. 네트워크 장치나 디스크 장치와 같은 장치(502)는 어떠한 합법적 활동도 수행되지 않을 시기 중의 읽기나 쓰기들로부터 보호될 수 있다. 예를 들어 어떤 악성 소프트웨어가 부팅 중에 디스크 드라이브들을 공격할 수 있다. 그에 따라 펌웨어 보안 에이전트(516)가 디스크가 부팅되고 있는 시간 동안 장치로의 어떠한 쓰기들도 금지할 수 있다. 마찬가지로 장치(502)가 상주하는 시스템의 관리자에 의해 장치들이나 시스템들이 언제 어떻게 사용될 수 있는지에 대한 허가가 설정될 수 있다. 예를 들어 장치(502)가 상주하는 시스템의 관리자는 업무 시간 밖에서는 장치가 사용될 수 없도록 설정할 수 있다. 시스템 상의 네트워크 장치는 업무 시간 밖으로 활동을 옮길 어떠한 합당한 목적도 가지지 않을 것이며, 그에 따라 보안 규칙들(518) 안에서의 허가에 기반하여 네트워크 장치의 읽기 및 쓰기들이 펌웨어 보안 에이전트(516)에 의해 차단될 수 있다. 그러한 이용은 예컨대 장치의 실제 사용자나 도스(denial-of-service) 공격을 행하기 위해 네트워크 장치를 이용하는 악성 소프트웨어에 의한 의도적 활동을 차단할 수 있다.

[0126] 또 다른 예에서 보안 규칙들(518)은 펌웨어 보안 에이전트(516)가 I/O 명령들과 함께 사용되는 파라미터들에 기반하여 명령을 평가하게 하는 규칙들을 포함할 수 있다. 그러한 파라미터들은 예컨대 쓰기 명령이 쓸 어드레스를 포함할 수 있다. 보안 규칙들(518)은 디스크 장치의 특정 부분이 읽기만 가능하다는 것을 가리키는 규칙을 포함할 수 있다. 따라서 펌웨어 보안 에이전트(516)는 데이터를 디스크에 쓰기 위한 OUT 명령과 관련된 파라미터들을 검사하여, 데이터가 기입될 어드레스를 판단하고, 시도된 쓰기가 보안 규칙들(518) 안의 어떤 규칙에 의해 쓰기 보호되는 디스크 부분에 대한 것인 경우 그 명령을 차단할 수 있다. 펌웨어 보안 에이전트(516)는 호출을 발생했던 콘텐츠나 개체와 같은 다른 베이스들과 연계하여 그러한 파라미터를 고려할 수 있다. 예를 들어 기입될 데이터의 콘텐츠를 검색하는 것이 비용이 많이 드는 것일 수 있고, 그에 따라 보안 규칙(518)이 소정 어드레스 범위에 데이터가 기입되어야 하는 경우에만 기입될 데이터를 검색하도록 펌웨어 보안 에이전트(516)를 구성할 수 있다. 다른 예에서 보안 규칙(518)과 같은 보안 규칙들은 어떤 호출하는 개체들에게만 디스크 장치의 소정 부분들로부터의 읽기나 쓰기를 허용할 수 있다. 그에 따라 펌웨어 보안 에이전트(516)는 시도된 쓰기나 읽기를 트래핑할 수 있고, 호출하는 개체의 아이디가 안전하게 판단될 때까지 그 시도를 허용하지 않을 수 있다. 그러한 판단은 장치 함수를 호출하기 위해 사용되는 파라미터들 안의 정보를 평가함으로써 이루어지는데, 이는 일부 그러한 함수들이 호출하는 장치 드라이버나 애플리케이션을 식별할 것이기 때문이다. 그 경우, 펌웨어 보안 에이전트(516)는 호출의 타당성을 판단하기 위한 적절한 단계를 밟을 수 있다. 일 실시예에서 펌웨어 보안 에이전트(516)는 보안 규칙들(518) 안의 화이트리스트나 블랙리스트를 조회하여 호출하는 개체가 그

러한 호출 발신이 허가되는지 여부를 판단할 수 있다. 다른 실시예에서 펌웨어 보안 에이전트(516)는 호출하는 애플리케이션이나 장치 드라이버가 유효한지 여부를 판단하기 위해 장치(502)를 포함하는 시스템 내 다른 보안 에이전트들과 통신할 수 있다. 그러한 다른 보안 에이전트들은 호출하는 애플리케이션이나 장치 드라이버의 동작을 검증했을 수 있고, 아니면 그러한 동작들을 검증했을 수 있는 O/S 내 보안 에이전트들과 통신할 수 있다. 또 다른 예에서, 장치(502)와 같은 장치로의 기존의 드라이버 호출들이 호출하는 개체를 식별하지 못할 수 있다. 그에 따라 어떠한 파라미터들도 이용되지 못할 수 있다. 그러한 예에서 펌웨어 보안 에이전트(516)는 시도된 액세스를 파생했던 호출의 정황을 판단하기 위해, 트리거된 이벤트를 시스템 내 다른 보안 에이전트들로 보내거나 그렇지 않으면 다른 보안 에이전트들을 조회하도록 구성될 수 있다. 그러한 다른 보안 에이전트들은 허가된 개체가 그러한 시도를 했는지 여부를 판단하기 위해 호출의 적절한 정황을 제공할 수 있다.

[0127] 또 다른 예에서, 보안 규칙들(518)은 펌웨어 보안 에이전트(516)가 장치(502)가 상주하는 환경으로부터의 정보에 기반하여 명령을 평가하게 하는 규칙들을 포함할 수 있다. 시스템 내 다른 보안 에이전트들은 제거하기 어려운 악성 소프트웨어 감염을 검출했거나, 청소를 위해 관리자로부터의 직접 개입을 요구할 수 있다. 시스템 내 다른 보안 에이전트들은 의심스러운 동향을 관찰하였을 수 있으나, 그 동향의 성격이 아직 완전히 분석되지 못하였다. 그 경우, 펌웨어 보안 에이전트(516)는 다른 보안 에이전트들로부터 그러한 기존 위협에 대한 통지를 수신할 수 있다. 보안 규칙들(518)은 그에 따라, 펌웨어 보안 에이전트(516)에 대해 감염 타입에 따른 예방적 액션들을 지시할 수 있다. 예를 들어 키보드 장치 안의 펌웨어 보안 에이전트(518)는 키로깅(keylogging)에 대해 알려진 특정 타입의 악성 소프트웨어에 대한 증거가 검출되었지만 아직 제거될 수 없다는 통지를 수신할 수 있다. 보안 규칙들(518)은 그에 따라, 펌웨어 보안 에이전트(516)가 키보드와 통신 중인 정보에 대한 위험 노출을 막기 위해 키보드로부터의 모든 읽기 및 쓰기들을 불허하도록 지시한다.

[0128] 펌웨어 보안 에이전트들(516)은 다양한 방식을 통해 다양한 타입의 장치들의 I/O를 보호할 수 있다. 예를 들어 디스플레이 장치의 펌웨어 보안 에이전트(516)는 악성 소프트웨어 위협에 따라, 디스플레이의 일부를 셧다운할 수 있다. 펌웨어 보안 에이전트(516)는 소정 패턴의 디스플레이를 차단하여 스크린 상에 워터마크가 생성되게 할 수 있다. 펌웨어 보안 에이전트(516)는 시도된 특정 패턴의 디스플레이를 트래핑할 수 있다. 펌웨어 보안 에이전트(516)는 스크린 캡처들을 방지하기 위해 장치로부터 시도된 정보 읽기들을 인터셉트할 수 있다.

[0129] 다른 예에서 키보드 장치를 위한 펌웨어 보안 에이전트(516)는 옵션으로서, 시스템의 나머지와 통신 중에 그 결과들을 인코딩 또는 디코딩할 수 있다. 그러한 암호화는 키로거 같은 악성 소프트웨어 위협이 존재한다는 통지에 따라 펌웨어 보안 에이전트(516)에 의해 설정될 수 있다.

[0130] 또 다른 예에서, 네트워크 장치를 위한 펌웨어 보안 에이전트(516)는 소스 인터넷 프로토콜("IP") 어드레스, 소스 포트 넘버, 송수신될 데이터, 목적지 IP 어드레스나 목적지 포트 넘버에 기반하여 트래핑할 수 있다. 네트워크 장치를 이용하고자 하는 그러한 시도가 트래핑될 때, 펌웨어 보안 에이전트(516)는 악성 소프트웨어의 증거를 위해 송수신될 패킷들의 데이터 페이로드를 검색할 수 있다. 일 실시예에서 그러한 데이터 페이로드들이 다른 보안 에이전트로, 혹은 보안 서버로 보내질 수 있으며, 이때 악성 소프트웨어의 증거를 위해 그 콘텐츠가 검색될 수 있다. 데이터 페이로드의 콘텐츠는 패킷 스니퍼(sniffer)가 그 콘텐츠를 성공적으로 인터셉트하지 못하도록 암호화될 수 있다. 안전하지 않은 네트워크 목적지들과의 통신과 관련된 보안 위험들로 인해 네트워크 장치에 대해 시도된 동작들이 트래핑될 수 있으며, 이때 악의적 목적지와의 네트워크 통신은 장치(502)가 상주하는 시스템의 보안을 위태롭게 할 수 있다. 시도된 동작들은 뱅킹 웹사이트와 같은 특정 데이터 세트의 민감한 특성으로 인해 트래핑될 수 있다. 그 경우, 그러한 웹사이트로부터 데이터 수신 시, 데이터가 다른 보안 에이전트나 호출하는 개체로 보내지기 전에 펌웨어 보안 에이전트(516)에 의해 암호화될 수 있다. 그러한 암호화는 장치(502)의 시스템 내 패킷 스니퍼나 필터가 정보를 성공적으로 인터셉트하지 못하게 할 수 있다.

[0131] 트래핑될 특정 I/O 명령들(510)은 특정 장치 및 그 장치의 동작들에 좌우될 수 있다. 따라서, 장치(502)의 메이커는 특정 장치(502)에 대한 펌웨어 보안 에이전트(516)의 동작을 어떻게 구성할지를 결정할 수 있다. 장치(502)의 메이커는 장치(502)의 기능을 얼마나 많이 다른 보안 에이전트들에 노출할지를 결정할 수 있다. 예를 들어 장치(502)는 트리거된 이벤트들을 그러한 보안 에이전트들로 핸드오프하기 전에 다른 보안 에이전트들을 이용한 검증을 요하도록 구성될 수 있다.

[0132] 동작 시, 하나 이상의 O/S 하위 보안 에이전트들이 시스템(400)이나 시스템(400)의 구성요소들의 펌웨어 안에서 실행될 수 있다. 펌웨어 보안 에이전트(440)는 디스플레이(424) 내에서 동작할 수 있고, 펌웨어 보안 에이전트(442)는 스토리지(426) 안에서 동작할 수 있으며, PC 펌웨어 보안 에이전트(444)는 메인 PC 펌웨어(408) 안에서 동작할 수 있다. O/S 하위 에이전트(450) 및 O/S 내 에이전트(422)가 시스템(400) 안에서 동작할 수 있다. 각

각의 보안 에이전트는 시스템(400) 내 하나 이상의 다른 보안 에이전트들과 통신할 수 있다. 그러한 각각의 보안 에이전트는 통신을 허용하기 전 다른 보안 에이전트의 인스턴스를 검증할 수 있다. 보호 서버(402)는 보안 에이전트를 검증한 후 보안 에이전트들 중 하나 이상과 통신할 수 있다.

[0133] PC 펌웨어 보안 에이전트(444)나 O/S 하위 에이전트가 메인 보안 에이전트로서 지정될 수 있다. 메인 보안 에이전트는 보안 규칙들을 판단하기 위해 보호 서버(402)와 통신할 수 있다. 메인 보안 에이전트는 메인 보안 에이전트 내부적으로 보안 규칙들을 저장할 수 있다. 메인 보안 에이전트는 보안 에이전트들 각각으로 보안 규칙들을 분산시킬 수 있으며, 이때 보안 규칙들이 해당 보안 에이전트 내부에 저장될 수 있다. 보안 규칙들은 대규모 보안 규칙들의 세트에 대한 비용을 줄이기 위해 장치의 타입, 제조자 또는 모델에 대해 맞춤화될 수 있다.

[0134] 규격들(434)과 같은 보안 규칙들의 수신 시, 디스플레이(424)와 같은 장치는 트래핑되어야 하는 장치의 동작들에 해당하는 플래그들을 장치 펌웨어(430) 내부의 제어 구조 안에 세팅할 수 있다. 유사한 작업들이 스토리지(426)에 의해 수행될 수 있다.

[0135] 애플리케이션(410)이나 드라이버(411)는 디스플레이(424)나 스토리지(426)와 같은 장치를 액세스하고자 시도할 수 있다. 애플리케이션이나 드라이버(411)는 운영체제(412)의 커널을 호출함으로써 그러한 시도를 할 수 있고, 그러면 그 커널이 운영체제 장치 드라이버들을 호출할 것이고, 그러면 그 장치 드라이버들이 구성요소(424, 426)로 해당 요청을 보낼 수 있다.

[0136] 상기 요청은 스토리지(426)과 같은 장치에 도달할 수 있다. 장치 상에서 실행되는 펌웨어 보안 에이전트(422)는 제어 구조를 가진 스토리지(426)의 데이터 전송 엔진(412)을 감시함으로써 그러한 요청을 필터링할 수 있다. 상기 요청은 스토리지(426)에 의해 이용 가능한 I/O 명령(510)의 형태를 취할 수 있다. 상기 요청이 펌웨어 보안 에이전트(422)에 의해 세팅되어 있던 어떤 플래그들과 매치하는 경우, 그 요청은 트래핑될 수 있고 그에 따른 이벤트가 유발(트리거)될 수 있다. 펌웨어 보안 에이전트(442)는 트리거된 이벤트를 어떻게 처리할지를 결정하기 위해 보안 규칙들(436)을 조회할 수 있다.

[0137] 일 실시예에서 트리거된 이벤트는 펌웨어 보안 에이전트(442)에 의해 처리될 수 있고, 관련 데이터, 명령, 정황 정보, 시간, 또는 환경 정보와 같이 이용 가능한 정보에 기반하여 교정 액션이 수행될 수 있다. 그러한 교정 액션은 해당 요청을 허용하거나 거부하는 동작을 포함할 수 있고, 악성 코드나 데이터를 제거하거나 전송될 데이터를 암호화할 수 있다. 다른 교정 액션은 트래핑된 이벤트에 관한 정보를 보호 서버(402)로 전달되게 전송하는 동작을 포함할 수 있다. 펌웨어 보안 에이전트(442)는 트래핑된 이벤트의 상태에 대해 다른 보안 에이전트들로 알려서, 그 다른 에이전트들 역시 각자의 보안 규칙들을 조회한 후 교정 액션을 수행하도록 할 수 있다. 예를 들어 펌웨어 보안 에이전트(442)가 알려지지 않은 출처의 악성 소프트웨어 공격을 검출한 경우, 펌웨어 보안 에이전트(440)가 디스플레이(424)에 대한 추가 액세스를 막을 수 있다.

[0138] 다른 실시예에서 트리거된 이벤트는 처리를 위해 O/S 내 보안 에이전트(418), PC 펌웨어 보안 에이전트(444) 또는 O/S 하위 에이전트(450)와 같은 다른 보안 에이전트로 전달될 수 있다. 수신하는 보안 에이전트, 예컨대 PC 펌웨어 보안 에이전트(444)가 보안 규칙들(438)을 조회하여 트리거된 이벤트를 처리할 수 있다. 데이터, 명령, 정황 정보, 시간, 또는 환경 정보와 같이 이용 가능한 정보에 기반하여, 트리거된 이벤트에 의해 표현된 요청이 PC 펌웨어 보안 에이전트(444)에 의해 허용되거나 거부될 수 있다. PC 펌웨어 보안 에이전트(444)는 시도된 자원 액세스에 관한 정황 정보를 판단하기 위해 O/S 내 보안 에이전트(418)와 통신할 수 있다. PC 펌웨어 보안 에이전트(444)는 트리거된 이벤트를 어떻게 처리할지에 대한 추가 정보를 위해 보호 서버(402)와 통신할 수 있다. PC 펌웨어 보안 에이전트(444)는 결과적인 액션에 대한 명령들을 다시 발신 펌웨어 보안 에이전트(442)로 보낼 수 있다. PC 펌웨어 보안 에이전트(444)는 트리거된 이벤트에 관한 정보를 보호 서버(402)로 보내 분석되거나 기록되게 할 수 있다. 그러한 분석이나 기록은 트리거된 이벤트의 악의적 특성이 알려지지 않을 때 수행될 수 있다. PC 펌웨어 보안 에이전트(444)는 시스템(400)의 보안 에이전트들로, 특정 종류의 악성 소프트웨어가 검출되었다거나, 어떤 종류의 의심스러운 활동이 검출되었다거나 시스템(400)이 악성 소프트웨어 공격을 받고 있다는 것을 알릴 수 있다.

[0139] PC 펌웨어 보안 에이전트(444)로부터의 정보 수신 시, 펌웨어 보안 에이전트(440)는 교정 액션을 취할 수 있다. 그러한 액션은 시도된 액세스를 허용하거나 거부하는 동작, 전송될 데이터를 암호화하는 동작, 또는 악성 코드나 데이터를 제거하는 동작을 포함할 수 있다.

[0140] 도 6은 악성 소프트웨어로부터 펌웨어 기반의 설정 가능한 전자 장치 보호를 위한 방법(600)의 실시예이다. 단계 605에서, O/S 하위 보안 에이전트, O/S 내 보안 에이전트, 보호 서버 및 펌웨어 보안 에이전트의 식별 및 보

안이 인증될 수 있다. 그러한 인증은 메모리에 위치된 각각의 이미지들, 암호화 해싱 또는 비밀 키들을 찾고 검증하는 것을 포함하는 어떤 적절한 방식을 통해 수행될 수 있다. 단계 605가 완료될 때까지 다른 단계들의 동작은 보류될 수 있다.

[0141] 단계 610에서 보안 규칙들을 판단하기 위해 보호 서버가 액세스될 수 있다. 그러한 보안 규칙들은 이하의 단계들에서 결정을 내리는 데 사용될 수 있다. 단계 615에서 펌웨어 보안 에이전트는 시스템 자원들에 대한 액세스를 트래핑하도록 지시될 수 있다. 그러한 액세스는 전자 장치 상에서 실행되는 애플리케이션들, 드라이버들 또는 운영체제들로부터 일어날 수 있다. 펌웨어 보안 에이전트는 전자 장치의 어떤 시스템 자원들이 감시되어야 하는지에 대해 지시될 수 있다. 펌웨어 보안 에이전트는 감시된 시스템 자원들에 대한 어떤 동작들이 트래핑되어야 하는지에 대해 지시될 수 있다. 예를 들어 펌웨어 보안 에이전트가 실행되는 장치에 대한 읽기 및 쓰기 명령들이 트래핑되기 위해 식별될 수 있다. 단계 620에서 트래핑될 그러한 동작들에 해당하는 플래그들이 제어 구조 안에서 세팅될 수 있다. 그러한 트래핑되는 동작들은 트리거된 이벤트를 생성할 수 있다.

[0142] 전자 장치는 단계들 630-675에서 시스템 자원들의 액세스에 대한 트래핑, 또는 단계 680-685에서 악성 소프트웨어가 존재하는지에 대한 데이터 검색 중 하나 이상에 의해 동작 및 보호될 수 있다. 시스템 자원들에 대한 액세스 트래핑 및 악성 소프트웨어 존재에 대한 데이터 검색 각각이 나란히 수행될 수 있다. 더 나아가, 이들 각각은 전자 장치의 동작을 보호하는데 필요할 때마다 반복될 수 있다.

[0143] 단계 630에서 시스템 메모리, 레지스터들 또는 I/O 장치들과 같은 시스템 자원에 대한 액세스가 트래핑될 수 있다. 그러한 트래핑은 전자 장치 상에서 실행되는 운영체제들의 레벨 아래에서 수행될 수 있다. 그러한 트래핑은 펌웨어 안에서 수행될 수 있다. 단계 632에서, 트래핑된 시도와 관련하여 결과적인 트리거된 이벤트뿐 아니라 관련 정보가 생성될 수 있다. 단계 635에서 트리거된 이벤트가 현재 처리되어야 하는지 처리되기 위해 다른 보안 에이전트로 전달되어야 하는지 여부가 판단될 수 있다. 그러한 판단은 하나 이상의 보안 규칙들을 액세스함으로써 내려질 수 있다. 트리거된 이벤트가 현재 처리되어야 하면, 단계 640에서 트래핑된 이벤트 및 관련 데이터, 명령, 정황 정보, 시간 또는 환경 정보와 같은 다른 정보에 기반하여 어떤 액션들을 취할지를 판단학 위해 보안 규칙들이 액세스될 수 있다. 예를 들어 기입되거나 판독될 데이터가 민감 혹은 악성 콘텐츠에 대해 검색될 수 있거나; 호출하는 개체가 권리를 가지는지를 보기 위해 그 개체가 식별될 수 있거나; 명령을 호출할 파라미터들이 검사될 수 있거나; 다른 보안 에이전트들로부터 시스템 내 악성 소프트웨어에 대한 경고가 참조될 수 있다.

[0144] 단계 642에서 시도된 액세스가 의심스러운지 아닌지 여부가 판단될 수 있다. 시도된 액세스와 관련된 정보와 함께 보안 룰들을 액세스한 것이 그 시도된 액세스가 의심스럽지 않다는 판단을 가져오면, 단계 645에서 그 시도가 허용될 수 있다. 그러한 시도가 의심스럽다고 판단되면, 단계 647에서 교정 액션이 수행될 수 있다. 그러한 교정 액션은 데이터로부터 악성 콘텐츠를 제거하는 동작, 악의적 시도의 존재에 대해 보호 서버나 다른 보안 에이전트들에 알리는 동작, 시도된 액세스를 불허하는 동작, 또는 전송될 데이터를 암호화하는 동작을 포함할 수 있다. 해당 시도가 의심스럽지 않으면, 단계 650에서 트리거된 이벤트가 허용될 수 있다.

[0145] 단계 655에서 다른 보안 에이전트가 트리거된 이벤트를 처리하는 것으로 판단되면, 트리거된 이벤트는 처리할 다른 보안 에이전트로 보내진다. 단계 670에서 취해질 적절한 액션을 가리키는 응답이 보안 에이전트로부터 수신될 수 있다. 단계 675에서 교정 액션 또는 트리거된 이벤트의 동작을 허용하는 것과 같은 액션이 수행될 수 있다.

[0146] 단계 680에서 악성 소프트웨어 존재에 대해 장치의 메모리가 검색될 수 있다. 그러한 메모리는 다른 네트워크 카드나 앞서 실행된 파일 읽기의 결과들과 같이 다른 개체로부터 도달한 콘텐츠를 포함할 수 있다. 메모리의 콘텐츠가 악의적이거나 의심스럽거나 알려지지 않은 것이라고 알게 된 경우, 단계 685에서 메모리의 그 콘텐츠는 제거될 수 있다.

[0147] 단계 690에서, 시도된 액세스가 거부되었거나 의심스러운 콘텐츠가 발견되었다면, 그러한 이벤트는 다른 보안 에이전트나 보호 서버에 보고될 수 있다. 그러한 보고는 어떤 관련된 악성 소프트웨어나 의심스러운 동향에 관한 정보를 포함할 수 있다.

[0148] 지속적으로, 또는 주기적으로, 또는 요청이 있을 때 전자 장치를 보호하기 위해 필요할 때 방법(600)의 단계들이 반복될 수 있다.

[0149] 도 7은 악성 소프트웨어로부터 전자 장치(204)를 보호하기 위한 마이크로코드 기반 시스템(700)의 실시예이다. 시스템(700)은 마이크로코드를 통해 시스템(100)의 소정 구성요소들을 구현하는 시스템(100)의 실시예일 수 있

다. 시스템(700)의 트래핑 동작들은 전자 장치(701)의 운영체제들의 밑에서 수행될 수 있다. 시스템(700)은 전자 장치(204)의 자원들에 대해 시도된 액세스 이용을 트래핑하고, 그 시도에 해당하는 트리거된 이벤트(유발 이벤트)를 생성하고, 그 트리거된 이벤트에 관한 보안 규칙들을 조회하며 필요 시 상기 시도에 관한 교정 액션을 수행하도록 구성된 하나 이상의 O/S 보안 에이전트들을 포함할 수 있다. 그러한 O/S 하위 보안 에이전트들은 전자 장치(701)의 자원들로부터 생성된 정보를 인터셉트하고, 상기 생성에 대응하는 트리거된 이벤트를 생성하고, 트리거된 이벤트에 관한 보안 규칙들을 조회하며 필요 시 상기 시도에 관한 교정 액션을 수행하도록 구성될 수 있다. 그러한 O/S 보안 에이전트들 중 하나 이상이 시스템(700)의 프로세서 내에서 전적으로 혹은 부분적으로 구현될 수 있다. O/S 하위 보안 에이전트들은 그러한 프로세서의 마이크로코드("μC")를 통해 전적으로 혹은 부분적으로 구현될 수 있다. 시스템(700)에 의해 보호될 수 있는 전자 장치(701)의 시스템 자원들(724)은 예컨대 도 2의 시스템 자원들(224)과 유사한 자원들, 물리적 메모리(714), 프로세서 플래그들(716), 예외들(718), 레지스터들(720) 또는 인터럽트들(722)을 포함할 수 있다.

[0150] 시스템(700)은 마이크로코드 보안 에이전트(708)와 같은 마이크로코드 기반의 O/S 하위 보안 에이전트를 포함할 수 있다. 마이크로코드 보안 에이전트(708)는 프로세서(704)와 같은 프로세서의 마이크로코드(708) 안에 상주할 수 있다. 일 실시예에서, 마이크로코드 보안 에이전트(708)는 애플리케이션(710), 드라이버(711) 또는 운영체제(713)와 같은 시스템(700)의 일부분에 의해 이루어지는 시스템 자원들(724)에 대해 시도되는 액세스를 트래핑하도록 구성될 수 있다. 마이크로코드 보안 에이전트(708)는 시스템 자원들(724)에 대해 시도된 액세스와 같은 것에 기반하여 트리거된 이벤트를 생성하도록 구성될 수 있다. 예를 들어 운영체제(713)는 물리적 메모리(714) 내 어떤 어드레스 안의 코드의 세그먼트를 실행하고자 시도함으로써 프로그램을 시동시키고자 할 수 있다. 다른 예에서 운영체제(713)는 물리적 메모리(714) 내 어느 어드레스를 판독하거나 기입하고자 할 수 있다. 물리적 어드레스(714)가 보여지고 있지만, 마이크로코드 보안 에이전트는 가상 메모리를 액세스하려는 시도를 트래핑하도록 구성될 수 있다. 다른 실시예에서 마이크로코드 보안 에이전트(708)는 마이크로코드 모듈들(710)과 같이 프로세서(702)의 다른 일부분으로부터 시도된 정보 전송을 트래핑하도록 구성될 수 있다. 마이크로코드 모듈들(710)은 명령어들을 실행하기 위해 프로세서(702)의 동작을 수행하도록 구성된 프로세서(702)의 다른 부분을 포함할 수 있다. 그렇게 시도된 정보 전송은 시스템 자원들(724)로부터의 동작들의 결과들을 포함할 수 있다. 예를 들어 코드 처리 중에 0으로 나누기(divide-by-zero) 동작이 마이크로 모듈(710)에 의해 인터셉트될 수 있고 예외(718)를 생성하고 전송하고자 할 수 있다.

[0151] 마이크로코드(706)는 운영체제(713)와 같은 시스템(700)의 구성요소들로부터 수신된 상위 레벨 명령어들을 실행하기 위한 하드웨어 레벨 명령어들을 포함할 수 있다. 마이크로코드(706)는 프로세서(702)에 의해 실행될 회로 레벨 명령어들로 그러한 상위 레벨 명령어들을 변환할 수 있다. 마이크로코드(706)는 프로세서(702)에 의해 구현되는 프로세서의 타입이나 전자 회로에 고유한 것일 수 있다. 마이크로코드(706)는 프로세서(702)의 생성 시 마이크로코드(706)의 특정 콘텐츠를 이용하여 구성될 수 있다. 프로세서(702) 상의 마이크로코드(706)를 업데이트하거나 재프로그래밍하는 기능은 제한될 수 있다. 마이크로코드(706)는 내부 프로세서 메모리(704) 안에 상주할 수 있다. 내부 프로세서 메모리(704)는 메모리(703)와 같이 시스템(700)의 시스템 메모리와 별개인 고속 메모리일 수 있다. 일 실시예에서 내부 프로세서 메모리(704)는 읽기만 가능한 메모리일 수 있다. 다른 실시예에서 마이크로코드(706)는 내부 프로세서 메모리(704)에 포함되는 프로그래머블 로직 어레이 안에 상주할 수 있다. 또 다른 실시예에서 내부 프로세서 메모리(704)는 메모리 저장부나 제어 저장소를 포함하거나 그들에 의해 구현될 수 있다. 그러한 실시예에서 내부 프로세서 메모리(704)는 부분적으로나 전체적으로 고정 램(static- random-access-memory)이나 플래시 메모리로 구현될 수 있다. 그러한 실시예에서 마이크로코드(706)는 프로세서(702)의 초기화의 일부로서 메모리(703)와 같은 어떤 다른 저장 매체로부터 메모리 저장부 안에 로딩되도록 구성될 수 있고, 메모리 저장부에 기입된 데이터를 통해 보안 규칙들이나 기계 명령어들과 같은 새로운 정보로 업데이트되거나, 재설치되거나, 그 새로운 정보를 수신하도록 구성될 수 있다.

[0152] 마이크로코드 보안 에이전트(708)는 어떤 동작, 명령, 전송, 또는 기타 액션들이 트래핑되어야 하는지를 결정하기 위해 보안 규칙들(707)을 액세스하도록 구성될 수 있다. 보안 규칙들(707)은 마이크로코드(706), 또는 다른 적절한 프로세서(702)나 시스템(700)의 일부 안에 상주할 수 있다. 보안 규칙들(707)은 마이크로코드 보안 에이전트(708)에 대해 호출을 발신하고 파라미터들을 통해 정보를 전달하는 다른 보안 에이전트들과 같이 프로세서(702) 외부의 개체들로부터의 기능 호출들에 의해 구현될 수 있다. 마이크로코드 보안 에이전트(708)는 보안 규칙들(707)과 통신 가능하게 연결될 수 있다. 일례에서 보안 규칙(707)은 다음과 같은 로직을 가질 수 있다:

[0153] - 어드레스(x)가 가상 메모리 범위(X1-->X2)나 물리적 메모리 범위(Y1-->Y2) 안의 코드에 의해 실행되면, 처리를 위해 O/S 하위 에이전트에 대해 트리거된 이벤트를 생성;

[0154] - 어드레스(x)가 물리적 메모리 범위(Z1-->Z2) 안의 코드에 의해 실행되면, 명령을 건너뜀;

[0155] - A, B 및 C이면, 메모리 범위(Y1-->Y2)가 메모리 범위(X1-->X2)를 액세스할 수 있음;

[0156] - 메모리 범위들(Y1-->Y2) 및 (T1-->T2)로부터의 코드만이 (Z1-->Z2)에 기입될 수 있음.

[0157] 마이크로코드(706)는 수신된 명령어들의 정황을 이해하기 위한 상태 머신을 포함할 수 있다. 그러한 정보는 예컨대 서로의 정황 안에서 연속적인 동작들을 평가하는 소정 보안 규칙들(707)을 실행하는 데 필요하게 될 수 있다. 그러한 정보는 트리거된 이벤트와 함께 전달될 수 있다.

[0158] 시스템(700)의 O/S 하위 보안 에이전트들 중 하나 이상이 O/S 하위 에이전트(712)에서 역시 실시될 수 있다. O/S 하위 에이전트(712)는 운영체제(713)와 같은 전자 장치(701)의 운영체제들의 레벨 아래에서 자원들에 대한 액세스의 트리거링을 제공하거나 그러한 트리거들을 처리하기 위한 어떤 적절한 방식으로 구현될 수 있다. O/S 하위 에이전트(712)는 도 2의 SVMM(216) 또는 SVMM 보안 에이전트(217), 도 4의 펌웨어 보안 에이전트(440, 442) 또는 PC 펌웨어 보안 에이전트(444), 또는 도 5의 펌웨어 보안 에이전트(516)의 기능 중 일부나 전체를 구현할 수 있다. O/S 하위 에이전트(712)는 보안 규칙들(723)과 통신 가능하게 연결될 수 있다.

[0159] 일 실시예에서, O/S 하위 에이전트(712)와 같이 시스템(700)의 O/S 하위 에이전트들 중 하나 이상이 마이크로코드 보안 에이전트(708)와 같은 마이크로코드 기반 보안 에이전트들에 의해 생성된 트리거된 이벤트들을 처리하도록 구성될 수 있다. O/S 하위 에이전트(712)는 도 1, 2, 4 및 5의 O/S 하위 에이전트들과 비슷한 방식으로 자원들에 대한 액세스를 트래핑하거나 트리거된 이벤트들을 처리하도록 구성될 수 있다. O/S 하위 에이전트(712) 및 마이크로코드 보안 에이전트(708)는 통신 가능하게 연결될 수 있다. 마이크로코드 보안 에이전트(708)는 트리거된 이벤트들을 O/S 하위 에이전트(712)로 보내도록 구성될 수 있다. O/S 하위 에이전트(712)는 O/S 내 보안 에이전트(719)와 같은 다른 보안 에이전트들과 통신 가능하게 연결될 수 있으며 보호 서버(202)에 통신 가능하게 연결될 수 있다. O/S 하위 보안 에이전트(712)는 O/S 내 보안 에이전트(719)와 같은 다른 보안 에이전트들로부터 정황 정보를 수신하도록 구성될 수 있다. 그러한 정보는 시스템 자원들(724)에 대해 시도된 액세스를 생성했던 개체에 대한 정보를 제공할 수 있다. 하나를 넘는 O/S 내 보안 에이전트(719)가 시스템(700) 안에 존재하는 경우, 각각의 O/S 내 보안 에이전트(719)는 트래핑, 검증(validating) 또는 O/S 내 보안 에이전트(719)와 관련된 다른 작업들의 지정된 일부를 수행하도록 구성될 수 있다. 그러한 일부는 운영체제 하위 보안 에이전트들에 의해 정의될 수 있다. 예를 들어 하나의 O/S 내 보안 에이전트(719)가 MOV 명령어들을 검증하거나 검사할 수 있고, 또 다른 O/S 내 보안 에이전트(719)가 JMP 명령어들을 검증하거나 검사할 수 있다.

[0160] O/S 하위 에이전트(712) 역시, 보호 서버(202)로부터 보안 규칙들이나 시기 적절한 정보를 수신하도록 구성될 수 있다. 또한 O/S 하위 에이전트(712)는 마이크로코드 보안 에이전트(708)로부터 수신된 트리거된 이벤트를 어떻게 처리할지를 결정하기 위해 보안 규칙들(723)과 같은 보안 규칙들, O/S 내 보안 에이전트(719)와 같은 다른 보안 에이전트들로부터 수신된 어떤 정황 정보, 또는 보호 서버(202)를 조회하도록 구성될 수 있다.

[0161] 특정 실시예들에서 O/S 하위 에이전트(712)는 시스템(700)에서 조우된 동작의 정황을 이해하기 위해 동향 상태 머신을 포함할 수 있다. O/S 하위 에이전트(712)는 여기서, 정황에 기반하여 마이크로코드 보안 에이전트(708)에 의해 실행될 적절한 액션을 결정하도록 구성될 수 있다. 그러한 액션은 교정 액션, 동작 허용, 동작 거부, 또는 보안 규칙의 필요요건을 촉진하기 위한 다른 단계들의 수행을 포함할 수 있다. 마이크로코드 보안 에이전트(708)는 O/S 하위 에이전트(712)로부터 수신된 것과 같은 액션들을 수행하도록 구성될 수 있다.

[0162] O/S 하위 에이전트(712)는 또한 O/S 내 보안 에이전트(719)와 같은 다른 보안 에이전트에 의해 실행되어야 하는 적절한 액션을 결정하도록 구성될 수 있다. 예를 들어 마이크로코드 보안 에이전트(708)로부터 트리거된 이벤트가 특정 종류의 악성 소프트웨어 위협이나 전자 장치(701)의 커널이나 사용자의 특정 부분에 대한 위협을 나타내면, O/S 하위 에이전트(712)는 교정 액션을 수행하도록 O/S 내 보안 에이전트(719)에 지시하도록 구성될 수 있다. 따라서 O/S 하위 에이전트(712)는 O/S 내 보안 에이전트(719)를 제어할 수 있다.

[0163] O/S 하위 에이전트(712)는 마이크로코드 보안 에이전트(708)의 인스턴스를 검증하도록 구성될 수 있고, 그 반대의 경우도 성립된다. O/S 하위 에이전트(712)는 보안 규칙들(707) 안에 구현되어야 하는 보안 규칙들(723)로부터 나온 것들과 같은 보안 규칙들, 시스템(700)에 관한 상태 정보, 관리자나 환경 설정 및 선호사항, 또는 마이크로코드 보안 에이전트(708)가 동작을 트래핑하고 트리거들을 생성하고 그러한 트리거들을 처리하여 다른 보안 에이전트들로 그들을 전송하게 하는 다른 적절한 정보를 공유하거나 설정하기 위해 마이크로코드 보안 에이전트(708)와 통신하도록 구성될 수 있다.

[0164] O/S 하위 에이전트(712)는 어떤 적절한 메커니즘을 통해 마이크로코드 보안 에이전트(708)로 그러한 정보를 전송하도록 구성될 수 있다. O/S 하위 에이전트(712)는 프로세서(702), 마이크로코드(706) 또는 마이크로코드 보안 에이전트(78)의 함수들을 호출할 수 있고 파라미터들 같은 정보를 그 함수들로 전달할 수 있다. 그러한 함수들은 특히 그러한 변화들을 마이크로코드 보안 에이전트(708)로 전달하기 위해 생성될 수 있다. 예를 들어, 메모리로부터 동작하는 어떤 개체로부터 물리적 메모리의 범위 "B"로부터 물리적 메모리의 다른 범위 "A"의 액세스를 금지하기 위해, "Bar_Memory(*A,B*)" 같은 함수가 사용될 수 있다. 마이크로코드 보안 에이전트(708)는 호출된 이 함수의 결과로서, 마이크로코드(706) 안에서 파라미터들을 세팅하도록 구성될 수 있다. 마이크로코드 보안 에이전트(708)가 O/S 하위 에이전트(712) 대신 그러한 마이크로코드 명령어들을 호출하기 전에 O/S 하위 에이전트(712)를 검증하도록 구성될 수 있게 그러한 마이크로코드 명령어들을 호출하는 것에 특권이 부여될 수 있다. 다른 예에서 O/S 하위 에이전트(712)나 마이크로코드 보안 에이전트(708)는 메모리 저장부, 제어 저장부, 기타 기입 가능한 프로세서(702)의 일부나 마이크로코드(706)에 데이터를 기입함으로써 그러한 정보를 전송할 수 있다.

[0165] 프로세서(702)는 악성 소프트웨어로부터 시스템(700)을 보호하기 위해 모든 필요한 트래핑과 처리를 충분히 구현하기 위해 마이크로코드 보안 에이전트(708)에 대한 제한된 자원들을 가질 수 있다. 일 실시예에서 마이크로코드 보안 에이전트(708)는 프로세서(702)에 의해 수행될 액션의 트래핑만을 구현하도록 구성될 수 있으며, 후속 처리를 위해 시스템(700)의 다른 보안 에이전트들이나 구성요소들로 그러한 트래핑과 관련된 트리거들을 떠넘길 수 있다. 마이크로코드 보안 에이전트(708)는 요청이나 전송의 허용이나 불허와 같은 후속 액션을 취하거나 정보 보고와 같은 다른 액션을 취할 수 있다. 다른 실시예에서 마이크로코드 보안 에이전트(708)는 트리거된 이벤트들의 작은 일부에 대한 처리를 구현하도록 구성될 수 있다. 그러한 처리에 적합한 트리거된 이벤트들은 상당한 정황 정보를 요하지 않는 것들을 포함할 수 있다. 예를 들어 마이크로코드 보안 에이전트(708)는 보안 규칙들(707)을 통해 O/S 하위 에이전트(712)의 인스턴스가 검증되지 않았다면 특정 범위의 메모리 어드레스들이 모든 읽기 및 쓰기들로부터 보호되어야 한다는 정보를 수신할 수 있다. 그러한 보안 규칙이 구현될 수 있는 것은 콘텐츠가 매우 민감하고, O/S 하위 에이전트(712)의 동작 지원 없이 메모리 콘텐츠를 액세스하는 개체의 아이디가 식별될 수 없기 때문이다. 따라서 O/S 하위 에이전트의 인스턴스 및 동작을 검증한 후, 마이크로코드 보안 에이전트(708)는 그러한 검증을 나타내는 비트를 세팅할 수 있다. 메모리에 대해 시도된 액세스가 트리거되고 상기 비트가 아직 세팅되어 있지 않았으면, 마이크로코드 보안 에이전트(708)는 해당 메모리 범위의 콘텐츠 읽기, 쓰기 또는 실행을 불허하도록 구성될 수 있다. 상기 비트가 세팅되어 있으면, 마이크로코드 보안 에이전트(708)는 해당 메모리 범위에 대해 시도된 액세스를 트래핑하고, 호출하는 개체가 해당 메모리 범위를 액세스하도록 허용되었는지 여부를 정황 정보와 기타 설정사항들로부터 평가할 O/S 하위 에이전트(712)로 보내질 트리거된 이벤트를 생성하도록 구성될 수 있다. 그러면 O/S 하위 에이전트(712)는 아마도 해당 액세스를 허용할지 거부할지를 나타내는 결과적 액션을 마이크로코드 보안 에이전트(708)로 다시 보낼 수 있다.

[0166] 트리거된 이벤트는 시도된 액션의 소스, 방법 또는 목적지에 대한 식별에 사용될 수 있는 어떤 적절한 정보를 포함할 수 있다. 트리거된 이벤트는 보안 규칙들을 적용할 마이크로코드 보안 에이전트(708)나 O/S 하위 보안 에이전트(712)에 의해 사용될 수 있다. 트리거된 이벤트는 마이크로코드 보안 에이전트(708)에 의해 생성될 수 있다. 예를 들어, 트리거된 이벤트는 정확히 어떤 자원이 액세스되었는지, 어떤 명령이 호출되었는지, 어떤 명령어 피연산자(operand)들이 사용되었는지, 어떤 메모리 어드레스로부터 해당 시도나 명령어가 나왔는지(즉, 소스 메모리), 동작의 결과가 어떤 메모리 안에 저장되어야 했는지(즉, 타깃 메모리) 또는 어떤 메모리가 감염될 것인지, 또는 시도된 액션의 소스, 방법 또는 목적지의 식별로 이어지는 어떤 다른 정보를 상세화할 수 있다. 마이크로코드 보안 에이전트(708)는 또한 액티브, 슬립, 아이들, 정지, 및 리셋의 프로세서 상태들; 프로세서간 통신; 및 전력 소비와 같은 프로세서(702) 관련 정보를 포함하도록 구성될 수 있다.

[0167] O/S 하위 에이전트(712)와 같은 다른 보안 에이전트는 보안 규칙(722)을 적용할 때 해당 이벤트의 범위를 판단하기 위해 트리거된 이벤트 안에서 그러한 정보를 이용하도록 구성될 수 있다. O/S 하위 에이전트(712)는 운영체제(713)에서 동작하는 개체들에 대한 정보, 보호 서버(202) 내 새로운 정보, 다른 보안 에이전트들, 관리자 설정사항 등에 의해 검출되는 악성 소프트웨어나 기타 위협들과 같은 추가 단서들에 대한 액세스를 할 수 있다. 예를 들어, 물리적 메모리 내 특정 어드레스로부터 발생된 어떤 트래핑된 요청이 있을 때, O/S 하위 에이전트(712)는 특정 어드레스와 관련된 스레드, 프로세스 또는 애플리케이션을 판단할 수 있다. 그런 다음 O/S 하위 에이전트(712)는 그러한 개체가 문제의 액션을 수행하도록 허가되는지 여부를 판단하도록 구성될 수 있다. O/S 하위 에이전트(712)는 상기 개체의 아이디를 판단하도록 구성될 수 있다. O/S 하위 에이전트(712)는 해당 개체를 (가령, 화이트리스트 참조를 통해) 안전하다고 알려진 것, (동향을 관찰하거나 알려진 악성 소프트웨어에 대

한 블랙리스트를 조회하는 등에 따라) 악성이라고 알려진 것으로 분류하도록 구성될 수 있다. O/S 하위 에이전트(712)는 알려지지 않은 악의적 개체들에 대한 정보를 보호 서버(202)에 보고하도록 구성될 수 있다.

[0168] 마이크로코드 보안 에이전트(708)는 다른 보안 에이전트들에서 이용 불가할 수 있는 소정 프로세서(702) 자원들 및 기타 시스템 자원들(724)에 대해 (트래핑 목적으로) 액세스할 수 있다. 일 실시예에서 마이크로코드(706) 안에서의 마이크로코드 보안 에이전트(708)의 구현은 프로세서 외부의 호출하는 개체들에 대한 그러한 자원들의 제한된 노출에 의해 만들어진 한계를 피할 수 있다. 예를 들어 가상 머신 모니터는 가상화 목적 상 프로세서(702)에 의해 노출되었던 자원들에 대한 트래핑 동작들로 국한될 수 있다. 메모리에 대해 시도된 읽기, 쓰기 또는 실행을 트래핑하는 기능의 추가 예를 들 수 있다. 가상 머신 모니터 기반 보안 에이전트는 메모리에만 액세스할 수 있는데, 이는 그것이 가상화되는데 이용가능하기 때문이며, 그 결과, 가상 머신 모니터 기반 보안 에이전트는 메모리 페이지에 대해 시도되는 읽기, 쓰기 또는 실행 시도들만을 추적할 수 있다. 반대로 마이크로코드 보안 에이전트(708)는 특정한 물리적 메모리 어드레스에 대한 읽기, 쓰기 또는 실행 요청을 인터셉트 및 처리할 수 있으며 보안 규칙들(707)에 기반하여 그 요청을 평가할 수 있다. 보다 작은 세분도(granularity)가 시스템(700)에서 보안 솔루션들을 제공함에 있어 보다 큰 융통성을 제공할 수 있다. 특정한 물리적 메모리 어드레스와 관련하여 어떤 명령이 사용되었는지에 대한 명령 레벨의 인식은 시스템(700)에 메모리 페이지가 액세스되었다는 것만이 아니라 어떤 개체가 어떤 자원을 호출했는지에 대해 알린다. 이러한 융통성은 매우 가치 있을 수 있다. 예를 들어 마이크로코드 보안 에이전트(708)는 읽기, 쓰기 또는 실행 시도들에 대해 인접한 두 개의 메모리 어드레스들을 감시할 수 있지만, 보안 규칙들(707)에 의해 두 메모리 어드레스들 중 어느 것이 액세스되었는지에 따라 완전히 다른 액션을 수행하도록 지시될 수 있다. 어떤 시도가 이루어진 메모리 페이지만을 봤을 때, 그러한 규칙들의 구별은 적용되지 못할 수 있다. 다른 예에서 디버그 레지스터들을 감시 및 설정하기 위한 하이퍼바이저들에 의한 다른 방법들은 시스템(700)이 그렇듯이 디버그 레지스터들을 액세스하는 데 사용되었던 명령어들의 정황을 가지지 못하였다. 또한 그러한 디버그 레지스터들을 설정하거나 감시하기 위한 어떤 다른 개체들은 운영체제 레벨 아래에서 실행되지 못하여 그들을 악성 소프트웨어에 보다 취약하게 만든다. 마지막으로, 그러한 디버그 레지스터들을 설정하거나 감시하기 위한 어떤 다른 개체들은 보안이 이루어지지 않으며, 보안 규칙들의 액세스, 그러한 액세스의 평가 및 교정 액션을 취하는 기능이 없다.

[0169] 마이크로코드 보안 에이전트(708)에 의해 취해질 교정 액션들은 보안 규칙들(707)에 의해 결정되거나 O/S 하위 에이전트(712)로부터 수신된 어떤 적절한 액션을 포함할 수 있다. 명령들 또는 명령어들이 허용되거나 거부될 수 있다. 마이크로코드 모듈들(710)로부터 발생된 정보가 허용되거나 억제될 수 있다. 어떤 그러한 명령들, 명령어들, 또는 정보는 변형될 수 있다.

[0170] 마이크로코드 보안 에이전트(708)는 인터럽트 발생을 트래핑하도록 구성될 수 있다. 인터럽트들은 트래핑, 예컨대 "INT" 명령어의 실행에 의해, 이어서 인터럽트와 관련된 정보를 호스팅한다고 알려진 관련 레지스터들을 판독함으로써 트래핑될 수 있다. 예를 들어 범용 레지스터들은 인터럽트의 코드 식별자 및 그것을 호출하는데 사용되는 파라미터들을 학습하기 위해 판독될 수 있다. 예를 들어 인터럽트(13)는 디스크 인터럽트일 수 있고, 알려진 레지스터들의 집합이 그 인터럽트를 읽기나 쓰기 및 데이터의 관련 섹터들과 위치들로서 식별할 수 있다.

[0171] 마이크로코드 보안 에이전트(708)는 프로세서(702)의 입/출력 포트들로 기입되는 값들을 트래핑하도록 구성될 수 있다. 마이크로코드 보안 에이전트(708)는 프로세서(702)에 의해 입/출력 장치들로 기입되는 값들을 트래핑하도록 구성될 수 있다. 마이크로코드 보안 에이전트(708)는 그러한 읽기나 쓰기를 행하기 위한 명령어들에 따라 트래핑하도록 구성될 수 있다.

[0172] 마이크로코드 보안 에이전트(708)는 또한 프로세서(702)의 산술 로직 유닛("ALU")의 소정 동작들을 트래핑하도록 구성될 수 있다. 보호된 해싱 알고리즘의 단계들에 대응하는 프로세서 상의 일련의 동작들이 함수에 대한 허가되지 않은 액세스를 판단하기 위해 트래핑될 수 있다. 어떤 산술 연산들은 악성 소프트웨어에 의해 자신들을 위장하거나 변형하는 데 사용될 수 있다. 소정 산술 명령어들, 비트 명령어들, 또는 MOV 명령어들은 메모리 페이지나 어드레스 범위에서의 콘텐츠 변화를 일으킬 수도 있는 모든 명령어들이다. 그러한 명령어들을 트래핑함으로써, 코드 섹션이나 데이터 섹션에 대한 변경이 기록될 수 있다. 이어지는 분석이 그 코드 섹션이나 데이터 섹션이 자가 변형 악성 소프트웨어의 일부로서 변형되었음을 보여주면, 트래핑되고 기록된 명령어들은 악성 소프트웨어에 의해 사용된 암호화 알고리즘을 추적하는 데 사용될 수 있다. 예를 들어 악성 소프트웨어가 자신을 변형하기 위해 특정 키와 함께 XOR 함수를 이용한다고 판단될 수 있다. 그러한 정보는 자가 변형 악성 소프트웨어를 검출하기 위한 보다 나은 보안 규칙들을 양산할 수 있다. 게다가, 메모리 변경에 대해 계속 추적함으

로써, 명령어들의 적용을 반대로 하여 복구 로직이 수행될 수 있다.

[0173] 또한 마이크로코드 보안 에이전트(708)는 DRM(digital-rights-management) 동작을 수행하도록 구성될 수 있다. 예를 들어 마이크로코드 보안 에이전트(708)는 특정 프로그램을 실행하라는 허가가 요구됨을 나타내는 보안 규칙(707)을 수신하도록 구성될 수 있다. 특정 프로그램은 메모리 내 특정 어드레스에서 찾을 수 있다. 그러한 허가는 예컨대 O/S 하위 보안 에이전트(712)로부터 허가 코드, 키, 또는 바이트를 수신하는 마이크로코드 보안 에이전트(708)의 형태를 취할 수 있다. 그러한 허가는 마이크로코드 보안 에이전트(708)에가 메모리에 대해 시도된 액세스를 트래핑하거나 프로그램 명령어들의 로딩, 및 이후 인가 코드, 키 또는 바이트에 대해 액세스할 수 있는 O/S 하위 보안 에이전트(712)로 트리거된 이벤트를 보냄으로써 수행될 수 있다. O/S 하위 보안 에이전트(712)는 그 결정을 마이크로코드 보안 에이전트(712)로 돌려 보낼 수 있다. 따라서 프로그램의 동작은 인가에 기반하여 허용되거나 불허될 수 있다.

[0174] 또한 마이크로코드 보안 에이전트(708)는 메모리의 첵섬이나 해시에 기반하여 메모리 내 특정 코드의 실행을 중단하도록 구성될 수 있다. 그러한 해시나 첵섬은 보안 규칙(707)에 의해 악의적인 것으로 표시될 수 있다. 코드가 메모리로부터 로딩되기 때문에, 마이크로코드 보안 에이전트(708)는 콘텐츠의 해시나 첵섬을 행하고, 그것을 알려진 악성 코드의 것들과 비교하며, 그런 다음 로딩 시도를 거부하고 공격 코드를 제거하기 위해 복구 함수를 로딩할 수 있다.

[0175] O/S 하위 에이전트(712)는 마이크로코드 보안 에이전트(706)를 포함하는 시스템(700)의 다른 보안 에이전트들로, 시스템(700)이 악성 소프트웨어에 감염되었다거나 의심스러운 동향과 마주하였다거나 그렇지 않으면 위험한 상태에 놓여졌다는 것을 알리도록 구성될 수 있다. 그러한 경우 마이크로코드 보안 에이전트(706)는 프로세서(702)의 일부분의 동작을 불능이 되게 하도록 구성될 수 있다. 마이크로코드 보안 에이전트(706)는 특정 시스템 자원들(724)에 대한 요청들이나 마이크로코드 모듈들(710)로부터 생성된 전송을 트래핑하고 거부함으로써 그러한 동작들이 불능이 되게 하도록 구성될 수 있다. 프로세서(702)의 일부분은 악성 소프트웨어에 민감하거나 악성 소프트웨어에 의해 오용될 가능성이 있기 때문에 불능(디세이블)으로 될 수 있다.

[0176] 마이크로코드 보안 에이전트(706)는 메모리 어드레스들이나 어떤 메모리 어드레스들의 범위를 로딩, 읽기, 쓰기 또는 실행 시도들로부터 보호하도록 구성될 수 있다. 그러한 메모리는 민감한 데이터를 포함하거나, 제한되거나, 민감하거나 보호되는 함수에 대한 초기화 지점일 수 있다. 마이크로코드 보안 에이전트(706)는 액세스하는 소프트웨어가 안전하다거나 중성적이라고 확인되지 않은 경우 그러한 메모리로의 액세스를 금지할 수 있다. 그 경우, O/S 하위 에이전트(712)와 같은 보안 에이전트들은 보호될 특정 메모리 어드레스들을 식별할 수 있는데, 이는 아마도 그러한 메모리 어드레스들이 민감한 정보이거나 보호되는 루틴들에 해당할 수 있기 때문일 것이다. O/S 하위 에이전트(712)는 어떤 어드레스들을 보호해야 하는가에 관한 보안 규칙들(707)과 같은 정보를 마이크로코드 보안 에이전트(708)로 보낼 수 있다. 마이크로코드 보안 에이전트(708)는 그러한 메모리 어드레스들로 시도된 로딩, 실행, 읽기, 또는 쓰기를 트래핑할 수 있고, 트리거된 해당 이벤트를 O/S 하위 에이전트(712)로 보낼 수 있다. O/S 하위 에이전트(712)는 보안 규칙(723), 보호 서버(202)로부터의 정보, 화이트리스트, 또는 어떤 다른 적절한 정보 소스에 따라 호출하는 소프트웨어가 안전한지 중성적인지 여부를 판단할 수 있다. O/S 하위 에이전트(712)는 구현될 액션을 다시 마이크로코드 보안 에이전트(708)로 돌려 보낼 수 있다. 마이크로코드 보안 에이전트(706)는 가상 메모리 내 어떤 페이지나 범위 및/또는 물리적 메몰 내 어드레스나 범위를 보호하도록 구성될 수 있다. 마이크로코드 보안 에이전트(706)는 가상 메모리 페이지들, 위치들, 또는 어드레스들을 물리적 메모리 위치들이나 어드레스들로 변환하도록 구성될 수 있다. 따라서, 트래핑할 가상 메모리 위치나 시도가 일어났던 가상 메모리 위치가 주어질 때, 마이크로코드 보안 에이전트(706)는 대응하는 물리적 메모리 위치들을 판단하도록 구성될 수 있고, 반대의 경우도 성립될 수 있다.

[0177] 또한 마이크로코드 보안 에이전트(708)는 민감한 코드의 액세스를 보호하도록 구성될 수 있다. 일 실시예에서 마이크로코드 보안 에이전트(708)는 특정 어드레스에 대한 액세스를 감시함으로써 상술한 방식으로 민감한 코드에 대한 액세스를 보호하도록 구성될 수 있으며, 여기서 상기 어드레스는 코드가 메모리에 저장될 때 그 코드의 시작부를 나타낸다. 다른 실시예에서 마이크로코드 보안 에이전트(708)는 프로세서(304)의 동작을 민감한 데이터나 코드의 중간으로 이동시킬 수 있는 "JMP"나 그와 유사한 분기 명령들의 실행을 감시하도록 구성될 수 있다. 그러한 경우 마이크로코드 보안 에이전트(708)는 민감한 메모리 범위들과 함께 "JMP" 명령어들의 실행을 트래핑하도록 구성될 수 있다. 마이크로코드 보안 에이전트(708)는 "JMP" 명령어가 어디에서 발생했는지를 분석하도록 구성될 수 있다. 마이크로코드 보안 에이전트(708)는 O/S 하위 에이전트(712)에 의해 처리될 수 있는 트래핑된 "JMP" 실행 시도에 대응하는 트리거된 이벤트를 발생하도록 구성될 수 있다. O/S 하위 에이전트(712)는 "JMP" 명령이 어디에서 발생했고, "JMP" 명령이 발생된 그러한 메모리가 문제의 메모리를 액세스하도록 허

가되는지 여부를 고려하도록 구성될 수 있다.

[0178] 마이크로코드 보안 에이전트(708) 자체나 그 안의 트래핑 기능 역시 시스템(700)의 다른 부분들에 의해 가능 또는 불능이 되도록 구성될 수 있다. 그러한 기능들은 트래핑 및 처리 이벤트들이 비용이 들고 그에 따라 시스템 성능을 해칠 경우 유용할 수 있다. 그러한 가능(인에이블링) 및 불능(디세이블링)은 특정하게 민감한 프로그램들이나 데이터의 사용, 악성 소프트웨어 위협의 검출, 관리자 선호사항 또는 어떤 다른 적절한 이유에 기반할 수 있다. 일 실시예에서 마이크로코드 보안 에이전트(706)는 보안 프로세싱 및 트래핑을 시작하기 위해 O/S 하위 에이전트(712)로부터 MSAOn 신호, VMXOn 신호, 또는 다른 명령어를 수신하도록 구성될 수 있다. 마이크로코드 보안 에이전트(708)는 보안 프로세싱 및 트래핑을 중단하기 위해 MSAOff 신호, "VMWrite VMXOff" 신호 또는 다른 명령어를 수신할 수 있다. 보안 프로세싱 및 트래핑을 시작하거나 중단하기 전에, 마이크로코드 보안 에이전트(708)가 요청을 행한 보안 에이전트의 아이디 및 인스턴스를 확인할 수 있다.

[0179] 또한 마이크로코드 보안 에이전트(708)는 전자 장치(701)의 프로세서(702) 및 다른 프로세서들 간 명령들 및 프로세서간 메시지들을 인터셉트하도록 구성될 수 있다. 그러한 프로세서간 명령들은 적절한 마이크로코드 모듈(710)에 의해 수신되거나 특정 시스템 자원들(724)을 액세스하는 전자 장치(701)의 어떤 개체에 의해 시도될 수 있다. 일 실시예에서 프로세서간 명령들은 운영체제(713)에서 머신 상태 레지스터를 이용하여 프로세서(702)를 액세스하는 소프트웨어로부터 보내질 수 있다. 악성 소프트웨어가 예컨대 프로세서들을 턴 오프하거나 슬립 모드 상태에 있게 하기 위한 메시지들을 보내고자 시도할 수 있다. 마이크로코드 보안 에이전트(708)는 예컨대 프로세서간 명령들에 대응하는 MSR 레지스터로 시도된 쓰기들을 트래핑하도록 구성될 수 있다. 트래핑된 명령에 대해 트리거된 이벤트가 해당 시도의 소스를 확인하도록 처리되기 위해 O/S 하위 에이전트(712)로 보내질 수 있다.

[0180] 마이크로코드 보안 에이전트(708)는 소프트웨어 인터럽트들(722)과 같이 프로세서로부터의 메시지 생성 및 전송을 인터셉트하도록 구성될 수 있다. 마이크로코드 보안 에이전트(708)는 인터럽트의 실행이 인가된 소프트웨어에 의해서만 액세스될 수 있게 그 실행을 제어하도록 구성될 수 있다. 예를 들어 알려진 아이디(메모리 내 드라이버의 해시들, 소스 등에 의해 판단되는 것과 같은)이 없는 드라이버들이나 악성 아이디는 소프트웨어 인터럽트들을 실행하도록 허용될 수 없다. 마이크로코드 보안 에이전트(708)는 인터럽트의 액세스를 트래핑하고 트리거된 이벤트를 처리하기 위해 O/S 하위 에이전트(712)로 보낼 수 있다.

[0181] 다른 예에서 마이크로코드 보안 에이전트(708)는 프로세서(702)에 의한 예외들(718)의 생성을 트래핑하도록 구성될 수 있다. 예외들은 예컨대 0으로 나누기 동작들, 페이지 오류들, 및 디버그 신호들을 포함할 수 있다. 이들을 포함하는 메모리 어드레스들에 대한 읽기 액세스가 마이크로코드 보안 에이전트(708)에 의해 트래핑되고 O/S 하위 에이전트(712)에 의해 처리될 수 있다.

[0182] 마이크로코드 보안 에이전트(708)는 프로세서(702)의 다양한 데이터 구조를 보호하도록 구성될 수 있다. 예를 들어 악성 소프트웨어가 인터럽트 서술자 테이블("IDT")을 공격할 수 있다. 일 실시예에서 마이크로코드 보안 에이전트(708)는 IDT 자체를 포함하는 메모리 위치들에 대한 쓰기 액세스 시도들을 트래핑할 수 있다. 다른 실시예에서, 마이크로코드 보안 에이전트(708)는 "LOAD IDT" 및 "STOE IDT"와 같이 IDT를 바꾸기 위한 함수들이 저장된 메모리 위치들을 보호할 수 있다. 다른 예에서 마이크로코드 보안 에이전트(708)는 인터럽트 핸들러들과 연관된 FELABS 또는 유사 데이터 구조, 또는 플래그들을 보호하도록 구성될 수 있다. 악성 소프트웨어는 허가되지 않은 소스들에 의한 그러한 자원들의 변경을 통해 인터럽트 핸들러들의 동작을 전복하고자 시도할 수 있다.

[0183] 마이크로코드 보안 에이전트(708)가 특정 타입 프로세서의 특정 인스턴스들로 특정될 수 있지만, 다른 회로 구성들이 다른 마이크로코드 명령어들을 필요로 할 때 일련의 보안 규칙들(707)은 주어진 명령어 세트를 이용하여 모든 프로세서들에 대해 유효할 수 있다. 이것은 마이크로코드 보안 에이전트(708)가 관련 자원들이 가변되며 회로에 따라 달라질 수 있는 해당 회로를 빼고는 동일한 명령어 세트를 구현하는 서로 다른 프로세서들 사이를 바꾸지 않을 소정 명령들을 트래핑할 수 있기 때문에 가능할 수 있다. 예를 들어, 메인 데스크탑 중앙 프로세싱 유닛("CPU") 및 내장 시스템 CPU는 둘 모두 동일 메이커로부터 나온 ISA 프로세서들일 수 있고, 따라서 보안 규칙들(707)은 적어도 일부가 두 타입의 프로세서들 사이에서 공유될 수 있다. 반대로, 서로 다른 명령어 세트를 가지는 그래픽 프로세서나 자동차 내장 프로세서 상의 그래픽 프로세싱 유닛은 보안 규칙들(707)을 공유할 수 없을 것이다.

[0184] 동작 시, 마이크로코드 보안 에이전트(708)는 전자 장치(701)의 프로세서(702) 안에서 실행될 수 있으며, O/S 하위 에이전트(712)는 전자 장치(104)의 운영체제의 레벨 밑에서 실행될 수 있다. 마이크로코드 보안 에이전트

(708) 및 O/S 하위 에이전트(712)는 서로를 인증할 수 있다. 마이크로코드 보안 에이전트(708)는 시스템 자원들(724)에 대한 액세스 및 마이크로코드 모듈들(710)에 의해 발생된 출력들이나 전송의 트래핑을 개시할 수 있다. 마이크로코드 보안 에이전트(708)는 O/S 하위 에이전트(712)로부터의 수요나, 보안 규칙(707), 또는 프로세서(702)의 시동에 따라 그렇게 개시될 수 있다. 시스템(700), 관리자, 또는 시스템 설정 시의 발생 때문에, 또는 트리거된 보안 규칙들(723)로 인해, O/S 하위 에이전트(712)는 마이크로코드 보안 에이전트(708)로 보안 가능 요청을 보낼 수 있다. 그러한 요청은 예컨대 특정 프로그램이 실행되어야 하거나 민감한 데이터가 액세스되어야 하거나 악성 소프트웨어 위협이 시스템(700)의 다른 어느 곳에서 검출되었기 때문에 발생될 수 있다. O/S 내 보안 에이전트(719) 및/또는 O/S 하위 시스템 에이전트(712)는 마이크로코드 보안 에이전트(708)로 자신을 인증할 수 있다. 자신을 인증하기 위해, OS 내 보안 에이전트(719) 및/또는 O/S 하위 시스템 에이전트는 인증 프로세스를 개시하기 위해 프로세서(702)에 의해 제공되는 특권을 가진 명령어를 호출할 수 있다. 그 호출이 마이크로코드 보안 에이전트(708)가 예컨대 서명이나 해시를 이용하여 OS 내 보안 에이전트(719) 및/또는 O/S 하위 시스템 에이전트(712)를 평가 및 인증하게 할 수 있다.

[0185] 마이크로코드 보안 에이전트(708)는 O/S 하위 에이전트(712)로부터 보안 규칙들(707)을 수신할 수 있다. 마이크로코드 보안 에이전트(708)는 함수 호출들 또는 메모리 저장과 같은 공유된 메모리로의 기입들을 통해 업데이트될 수 있다. 마이크로코드 보안 에이전트(708)는 보안 규칙들에 기반하여 특정 명령어들, 그러한 명령어들로의 피연산자들, 타깃 어드레스들, 소스 어드레스들, 또는 이들의 어떤 조합을 트래핑하도록 구성된 마이크로코드(706)의 제어 구조로 플래그들을 제공할 수 있다. 마이크로코드 보안 에이전트(708)는 운영체제(713), 애플리케이션(710) 또는 드라이버(711)와 같이 프로세서 위에서 실행되는 개체들에 의해 시도된 시스템 액세스들을 트래핑할 수 있다. 마이크로코드 보안 에이전트(708)의 동작은 그러한 개체들에게 투지시할 수 있다. 마이크로코드 보안 에이전트(708)는 다른 마이크로코드 모듈들(710)의 인스턴스로부터의 출력들과 같은 정보의 생성을 트래핑할 수 있다. 그러한 마이크로코드 모듈들(710)은 프로세서(702)를 위한 다양한 작업들을 수행하도록 구성된 마이크로코드의 다른 부분들을 포함할 수 있다. 예를 들어 마이크로코드 모듈들(710) 중 일부는 프로세서 예외나 인터럽트가 생성되어야 할 때, 입/출력 데이터를 어떻게 라우팅할지를 검출하거나 수학적 연산들을 수행할 수 있다. 마이크로코드 보안 에이전트(708)의 동작은 그러한 모듈들에게 투지시할 수 있다. 마이크로코드 보안 에이전트(708)는 관찰된 이전의 이벤트들에 기초하는 소정 트래핑을 수행하기 위해 상태 머신을 이용할 수 있다.

[0186] 자원에 대한 액세스나 정보 생성을 트래핑할 때, 마이크로코드 보안 에이전트(708)는 트래핑과 관련된 트리거된 이벤트를 생성할 수 있다. 그렇게 트리거된 이벤트는 트래핑된 명령어, 사용된 파라미터들, 발생 메모리 위치들 및 타깃 메모리 위치들과 같은 정황 정보를 포함하여, 트래핑에 대한 정보를 포함할 수 있다.

[0187] 일 실시예에서 마이크로코드 보안 에이전트(708)는 트리거된 이벤트를 처리할 수 있다. 다른 실시예에서 마이크로코드 보안 에이전트(708)는 트리거된 이벤트를 처리할 O/S 하위 에이전트(712)나 다른 보안 에이전트로 보낼 수 있다. 마이크로코드 보안 에이전트(708)는 트리거된 이벤트를 처리할지 여부 및 처리할 방법을 결정하기 위해 보안 규칙들(707)을 조회하거나 트리거된 이벤트를 O/S 하위 에이전트(712)로 보낼 수 있다. 마이크로코드 보안 에이전트(708)는 O/S 하위 에이전트(712)로부터 응답을 기다리거나, 보안 규칙들(707)에 의해 아무 후속처리도 요구되지 않는 경우 트래핑된 액션을 허용할 수 있다. 마이크로코드 보안 에이전트(708)는 보안 규칙들(707)에 기반하여 명령어를 허용하거나 거부하거나 처리될 값이나 파라미터를 교체하는 것과 같은 교정 액션을 취할 수 있다.

[0188] O/S 하위 에이전트(712)는 마이크로코드 보안 에이전트(708)로부터 트리거된 이벤트를 수신할 수 있다. O/S 하위 에이전트(712)는 트리거된 이벤트에 기반하여 취할 적절한 액션을 결정하기 위해 보안 규칙들(723)과 같은 보안 규칙들을 조회할 수 있다. O/S 하위 에이전트(712)는 마이크로코드 보안 에이전트(708)로부터 트리거된 이벤트 정보, O/S 내 보안 에이전트(719)로부터의 정황 정보, 보호 서버(202)로부터의 정보, 다른 보안 에이전트들로부터의 판단들, 관리자 설정사항, 시간, 또는 취해져야 할 적절한 액션을 결정하기 위한 다른 정보를 이용할 수 있다. O/S 하위 에이전트(712)는 O/S 내 보안 에이전트(719) 및/또는 마이크로코드 보안 에이전트(708)로 취할 액션들을 보낼 수 있다. O/S 하위 에이전트(712)는 트리거된 이벤트 및 결과적 액션들에 관한 정보를 보호 서버(202)로 보낼 수 있다.

[0189] 마이크로코드 보안 에이전트(708)는 O/S 하위 에이전트(712)와 같은 다른 보안 에이전트로부터 취해질 액션을 수신할 수 있다. 마이크로코드 보안 에이전트(708)는 명령어를 허용하거나 거부하거나 처리될 값이나 파라미터를 교체하는 것과 같은 수신된 액션을 실행할 수 있다.

[0190] 도 8은 악성 소프트웨어로부터 마이크로코드 기반의 개인화된 설정 가능한 전자 장치 보호를 위한 방법(800)의 실시예이다. 단계 805에서 마이크로코드 보안 에이전트의 인스턴스가 검증될 수 있다. 단계 810에서 다른 보안 에이전트의 인스턴스가 검증될 수 있다. 그러한 보안 에이전트는 O/S 하위 보안 에이전트를 포함할 수 있다. 단계 815에서 프로세서 안에서 마이크로코드 레벨로 트래핑할 하나 이상의 보안 규칙들이 획득되거나 보내지거나 수신될 수 있다. 그러한 보안 규칙들은 예컨대 함수 호출들 또는 공유된 메모리 공간으로의 파라미터들 기입을 통해 전송될 수 있다. 단계 820에서 마이크로코드 레벨에서의 자원들의 보안 트래핑이 개시될 수 있다. 일 실시예에서 그러한 개시는 보안 트래핑을 시작하라는 신호를 수신하는 것으로부터 일어날 수 있다. 그러한 실시예에서 시스템에 대한 악의적 공격이 검출되었거나 민감한 데이터가 시스템 안에 존재할 수 있기 때문에 신호가 수신될 수 있다. 다른 실시예에서 그러한 개시는 보안 규칙의 조회로부터 일어날 수 있다. 또 다른 실시예에서 그러한 개시는 프로세서의 시동으로부터 일어날 수 있다.

[0191] 단계 825에서 트래핑될 동작들에 해당하는 플래그들이 마이크로코드 안에서 세팅될 수 있다. 그러한 플래그들은 특정 명령어들, 그러한 명령어들에 대한 피연산자들, 타깃 어드레스들, 소스 어드레스들, 또는 그들의 어떤 조합에 대응할 수 있다. 그러한 플래그들은 수신되었던 보안 규칙들에 의해 정의될 수 있다. 단계 830에서 실행될 명령어들이 수신되어 트래핑 플래그들과 비교될 수 있다. 단계 835에서 마이크로코드로부터 생성되어 보내질 정보가 수신되어 트래핑 플래그들과 비교될 수 있다. 단계들 830 및 835는 상태 머신을 이용하여 구현될 수 있으며, 이 단계들은 반복될 수 있고 수 차례의 단계 반복들로부터의 결과들이 기억되어 플래그나 보안 규칙에 대해 함께 비교될 수 있다.

[0192] 단계 840에서 명령어나 정보가 트래핑되었는지 여부가 판단될 수 있다. 아무 것도 트래핑되지 않으면, 방법은 단계 830 및 835의 명령어들 및 생성된 명령의 감시로 돌아갈 수 있다. 무엇인가가 트래핑되었으면, 단계 845에서 트래핑과 관련된 트리거된 이벤트가 생성될 수 있다. 그렇게 트리거된 이벤트는 트래핑된 명령어, 사용된 파라미터들, 발생 메모리 위치들 및 타깃 메모리 위치들과 같은 정황 정보를 포함하여, 트래핑에 대한 정보를 포함할 수 있다.

[0193] 단계 850에서 트리거된 이벤트가 마이크로코드 안에서 처리되어야 할지 여부나 마이크로코드 밖의 보안 에이전트가 트리거된 이벤트를 처리해야 할지 여부가 판단될 수 있다. 트리거된 이벤트가 마이크로코드 안에서 처리되어야 할 경우, 단계 855에서 트리거된 이벤트에 대한 적절한 액션이 수행될 수 있다. 그러한 액션은 보안 규칙을 조회함으로써 정의될 수 있다. 그러한 액션은 명령이 실행될 수 있게 하거나 정보가 보내질 수 있게 하는 동작, 명령이나 전송을 거부하는 동작, 메모리나 파라미터들의 값들을 교체하는 동작, 또는 필요한 어떤 다른 교정 동작을 포함할 수 있다. 방법(800)은 이제 단계들(830 및 835)의 보안 감시를 계속할 수 있다.

[0194] 트리거된 이벤트가 마이크로코드 외부에서 처리되어야 하는 경우, 단계 860에서 트리거된 이벤트가 트리거된 이벤트를 처리할 보안 에이전트로 보내질 수 있다. 단계 865에서 트리거된 이벤트와 관련된 추가 정보가 수집될 수 있다. 그러한 정보는 설정사항, 선호사항, 정황 정보, 또는 악성 소프트웨어 상태를 포함할 수 있다. 그러한 정보는 단계 870에서 보안 규칙을 트리거된 이벤트에 적용하는데 이용될 수 있다. 그러한 애플리케이션은 트리거된 이벤트와 관련하여 취해질 액션의 과정을 산출할 수 있다. 단계 875에서 그러한 액션의 과정이 특정될 수 있고 그 특정된 액션을 구현할 수 있는 다양한 보안 에이전트들로 전송될 수 있다. 그러한 액션들은 교정 액션들, 일어날 동작이나 전송을 허용하는 동작, 이벤트를 보호 서버로 보고하는 동작 또는 어떤 다른 적절한 결과를 포함할 수 있다. 단계 880에서 단계 875에서 특정된 액션들이 수행될 수 있다. 방법(800)은 이제 단계들(830 및 835)의 보안 감시를 계속할 수 있다.

[0195] 도 9는 전자 장치(901) 상의 보안에 민감한 프로세서 자원들에 대한 소프트웨어 액세스를 규제하는 시스템(900)의 실시예이다. 시스템(900)은 운영체제(913)와 같은 전자 장치(901)의 운영체제들에서 실행되는 소프트웨어 기반 개체들로부터 프로세서 자원들(924)을 액세스하려는 악의적 시도들을 검출하기 위해 전자 장치(901) 상에서 동작하도록 구성된 O/S 하위 트래핑 에이전트(920) 및 트리거된 이벤트 핸들러(922)를 포함할 수 있다. 또한, O/S 하위 트래핑 에이전트(920) 및 트리거된 이벤트 핸들러(922)는 어떤 시도된 동작들이나 정보의 생성을 트래핑하고 트래핑된 동작이나 정보에 대응하여 생성된 트리거된 이벤트를 어떻게 처리할지를 결정하기 위해 하나 이상의 보안 규칙들(908)을 사용하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(920) 및 트리거된 이벤트 핸들러(922)는 트리거된 이벤트를 허용하거나 거부하거나 다른 교정 액션을 수행하도록 구성될 수 있다.

[0196] 전자 장치(901)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(901)는 메모리(903)에 연결된 하나 이상의 프로세서들(903)을 포함할 수

있다. 프로세서(902)는 도 2의 프로세서(208), 도 4의 프로세서(408), 도 7의 프로세서(702) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(903)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(901)는 하나 이상의 보안 규칙들(921)에 연결된 O/S 내 보안 에이전트(919)를 포함할 수 있는 운영체제(913)를 포함할 수 있다. 운영체제(913)는 도 1의 운영체제(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(919)는 도 1의 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418) 및/또는 도 7의 O/S 내 보안 에이전트(719), 또는 이들의 어떤 적절한 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0197] O/S 하위 트래핑 에이전트(920)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442) 또는 PC 펌웨어 보안 에이전트(444), 도 5의 펌웨어 보안 에이전트(516), 도 7의 마이크로코드 보안 에이전트(708) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 트리거된 이벤트 핸들러(922)는 도 1의 트리거된 이벤트 핸들러(108), 도 2의 SVMM 보안 에이전트(217), 도 4의 O/S 하위 에이전트(450), 도 7의 O/S 하위 에이전트(712) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 다양한 실시예들에서 O/S 하위 트래핑 에이전트(920)의 기능 중 일부는 트리거된 이벤트 핸들러(922)에 의해 수행될 수 있거나, 트리거된 이벤트 핸들러(922)의 기능 중 일부가 O/S 하위 트래핑 에이전트(920)에 의해 수행될 수 있다. 또한 O/S 하위 트래핑 에이전트(920) 및 트리거된 이벤트 핸들러(922)는 동일한 소프트웨어 모듈 안에서 구현될 수 있다.

[0198] 보안 규칙들(908)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(222), 도 4의 보안 규칙들(434, 436, 438), 도 5의 보안 규칙들(518), 도 7의 보안 규칙들(707, 723) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보안 규칙들(921)은 도 2의 보안 규칙들(220), 도 4의 보안 규칙들(420), 도 7의 보안 규칙들(721) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0199] O/S 하위 트래핑 에이전트(920)는 프로세서 자원들(924)과 같은 어떤 적절한 자원에 대한 액세스나 그로부터의 정보를 인터셉트하도록 구성될 수 있다. 예를 들어 프로세서 자원들(924)은 도 1의 자원(16), 도 2의 시스템 자원들(214), 도 4의 디스플레이(424) 및 스토리지(426)와 같은 구성요소들의 일부, 또는 도 7의 시스템 자원들의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 프로세서 자원들(924)은 프로세서가 명령어들을 로딩하고 실행할 수 있게 하기 위해 프로세서(902)와 같은 프로세서가 이용 가능한 자원들을 포함할 수 있다. 그러한 자원들은 예컨대 데이터 레지스터들(928), 제어 레지스터들(930), 캐시들(934), 프로세서 플래그들(936), 프로세서 코어들(938), 프로세서 예외들(940), 또는 프로세서 인터럽트들(942)을 포함할 수 있다. 그러한 자원에 대해 시도된 액세스는 피연산자들과 함께 어셈블리 언어 명령어들과 같은 명령어를 포함할 수 있다. 트래핑이 가능할 수 있는 프로세서 자원들(924)은 프로세서(902)에 의해 노출된 자원들에 좌우될 수 있다. 예를 들어 O/S 하위 트래핑 에이전트(920)가 가상 머신 모니터에서 구현되는 경우, O/S 하위 트래핑 에이전트(920)가 트래핑하기 위해 이용 가능한 프로세서 자원들(924)은 가상화 목적 상 프로세서(902)에 의해 노출된 프로세서 자원들(924)로 한정될 수 있다. 그러한 경우, 프로세서(902)는 프로세서 자원들(924) 중 일부에 대한 가상화 확장을 포함할 수 있다. 다른 예에서 O/S 하위 트래핑 에이전트(920)가 마이크로코드 보안 에이전트 안에서 구현되는 경우, 프로세서(902)는 프로세서(902)의 거의 모든 자원들이 트래핑에 이용 가능하게 할 수 있다.

[0200] O/S 하위 트래핑 에이전트(920)는 프로세서 자원 제어 구조("PRCS")(926)를 포함할 수 있다. PRCS(926)은 레코드, 데이터 구조, 테이블, 또는 어떤 다른 적절한 구조로 구현될 수 있다. PRCS(926)는 어떤 명령어들, 정보, 또는 프로세서 자원들(924)에 대해 시도된 액세스가 트래핑되어야 하는지를 특정하는 정보를 포함할 수 있다. O/S 하위 트래핑 에이전트(920)나 트리거된 이벤트 핸들러(922)는 트래핑되어야 하는 민감한 동작들, 정보, 또는 자원들에 해당하는 플래그들을 PRCS(926) 안에서 세팅하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(920)나 트리거된 이벤트 핸들러(922)는 보안 규칙들(908) 안에 포함된 정보에 따라 PRCS(926) 내에서 그러한 플래그들을 세팅하도록 구성될 수 있다.

[0201] 도 10은 PRCS(1000)의 실시예이다. PRCS(1000)는 도 9의 PRCS(926)의 일 실시예일 수 있다. PRCS(1000)는 트래핑되어야 하는 다양한 프로세서 자원들의 엔트리들로 된 테이블(1014)을 포함할 수 있다. 각각의 엔트리는 트리거된 이벤트를 가져올 수 있는 자원 및 조건들을 식별하는 하나 이상의 필드들(1004, 1006, 1008, 1010,

1012)을 가져올 수 있다. 예를 들어 PRCS(1000)는 트리거 플래그(1002), 자원의 식별자(1004), 자원과 관련된 타입(1006) 트리거 타입(1008), 이벤트를 언제 트리거할 지에 대한 트리거 시기 조건들(1010) 및 이벤트를 트리거할 실행 단계(1012)에 대한 필드들을 가질 수 있다. PRCS(1000)의 구현은 구조(산업 표준 구조 "ISA"와 같은 구조)나 프로세서(902)에 의해 노출된 자원들을 포함하여, 자원들이 식별된 프로세서의 특성에 좌우될 수 있다.

[0202] 트리거 플래그(1002)는 관련 엔트리(1014)에 대한 트래핑 및 트리거링이 턴 온되는지 턴 오프되는지 여부에 대한 표시를 포함할 수 있다. 그러한 플래그는 트래핑 조건이 PRCS(1000)에 엔트리(1014)로서 로딩되게 하지만 여전히 휴지상태일 수 있다. 그에 따라 PRCS(1000)는 이들을 활발히 시행하지 않고 보안 규칙들의 실시예들로 로딩될 수 있다. 트리거 플래그(1002)는 도 9의 O/S 하위 트래핑 에이전트(920)와 같은 개체에 의해 세팅되도록 구성될 수 있다. 그러한 동작은 PRCS(1000)가 특정 자원이나 조건에 대한 트래핑이 가능해야 했거나 불능이 되어야 할 때마다 늘거나 줄도록 요구하는 시스템과 비교하여 훨씬 빠르게 동작할 PRCS(1000)를 사용하여 안티 악성 소프트웨어 시스템을 동작 가능하게 할 수 있다. 개체(1014)를 턴 온하고 턴 오프하는 기능은 안티 악성 소프트웨어 시스템이 선택적으로 소정 동작들을 트래핑하게 할 수 있다. 그러한 선택성은 특정 트래핑 동작이 시간이나 실행의 관점에서 비용이 들 경우 유리할 수 있으며, 그에 따라 엔트리(1014)는 특정 조건들이 검출될 때에만 인에이블 될 것이다. 예를 들어 시스템이 특정 레지스터로 여러 차례 정상적 기입을 하는 경우, 그 레지스터에 대한 액세스의 트래핑은 안티 악성 소프트웨어 시스템의 다른 일부가 악성 소프트웨어 감염 가능성을 가리키는 의심스러운 동향을 검출할 때까지 턴 오프될 수 있다. 그러한 경우 레지스터의 쓰기들에 해당하는 엔트리(1014)의 트리거 플래그(1002)는 자원들을 공격하려는 어떤 추가적인 악의적 시도들을 캐치하기 위해 "ON"으로 세팅될 수 있다.

[0203] 자원 식별자들(1004)은 트래핑되어야 하는 프로세서의 특정 자원의 식별을 포함할 수 있다. 예를 들어 식별자(1004)는 해당 자원이 특정 데이터 레지스터와 같은 레지스터, EAX와 같은 어드레스 레지스터들, 스택 레지스터, 제어 레지스터, 벡터 레지스터, ESP와 같은 스택 포인터들, 명령 레지스터, 프로그램 카운터, 명령 레지스터, 프로그램 상태 워드, 상시 레지스터, 유동 소수점 레지스터, 또는 조건부 레지스터임을 보일 수 있다. 다른 예들로서 식별자(1004)는 자원이 "JMP", "JZ"(조건이 0에 해당할 때 점프), "JNZ"(조건이 0과 같지 않으면 점프), "MOV"(값을 이동), 또는 "SysEnter"(Ring 0 절차에 대한 빠른 호출)이라고 식별할 수 있다. 또 다른 예들로서, 식별자(1004)는 자원이 변환 룩어사이드(lookaside) 버퍼와 같은 캐시; 타임 스탬프 카운터와 같은 카운터; 시스템의 프로세서0, 프로세서1,..,프로세서N과 같은 논리적 코어; 또는 "DIV/0"와 같은 프로세서 예외들 또는 프로세서간 인터럽트나 기타 전역 변수들과 같은 인터럽트들 같은 다른 자원들 중 하나이다. 자원 식별자(1004)는 명령어, 레지스터, 또는 자원 식별자(1004)로 나타내는 다른 자원의 어드레스 표현으로 변환될 수 있다. 자원 타입(1006)은 엔트리(1014)가 포함하는 자원의 클래스나 타입에 대한 식별자를 포함할 수 있다. PRCS(1000)의 일부 엔트리들은 특정 타입의 모든 자원들에 적용될 수 있다.

[0204] 트리거 타입(1008)은 결과적 트리거된 이벤트가 동기되는지 비동기되는지 여부에 대한 식별을 포함할 수 있다. 동기적 트리거들은 트래핑된 자원의 실행이나 전송이 예컨대 해당 시도가 악성 소프트웨어를 나타내는지 여부가 판단될 때까지 정지되게 할 수 있다. 비동기적 트리거들은 트래핑된 자원의 실행이나 전송이 계속되게 할 수 있는 한편 그 트리거는 예컨대 추후의 평가를 위해 기록된다. 일 실시예에서 비동기적으로 트리거된 자원들에 시도된 액세스들은 보다 많은 일련의 액션들에 대한 평가를 구축하는 데 사용될 수 있고, 그러한 일련의 액션들에 대한 적절한 평가는 판단이 이뤄질 수 있기 전에 여러 데이터 포인트들을 필요로 할 수 있다. 예를 들어 명령 포인터 레지스터의 특정 읽기 자체가 악성이 아닐 수 있지만 돌아 온 정보의 후속 사용은 악성일 수 있다. 따라서 상태 머신은 명령 포인터 레지스터의 읽기를 먼저 비동기적으로 트래핑하지만 이어서 다른 명령어 속에서 그 사용을 동기적으로 트래핑하는데 사용될 수 있다.

[0205] 트리거 시기 조건들(1010)은 트리거된 이벤트가 자원의 액세스에 기반하여 생성되게 할 논리적 규칙들이나 조건들을 포함할 수 있다. 예를 들어 트리거된 이벤트들은 레지스터에 대해 그 자원이 기입되거나 판독될 때 생성될 수 있다. 트리거된 이벤트들은 "JMP"와 같은 명령어에 대해 그 명령어가 실행될 때 생성될 수 있다. 트리거된 이벤트들은 변환 룩어사이드 버퍼와 같은 캐시에 대해 그 캐시가 무효화될 때 생성될 수 있다. 트리거된 이벤트들은 프로세서 코어에 대해 그 코어가 아이들이 될 때와 같이 프로세서의 상태에 따라 생성될 수 있다. 프로세서 예외나 프로세서 플래그는 그 플래그나 예외가 세팅되거나 기입될 때 트리거될 수 있다. 트리거 시기 조건들(1010)은 단일 자원에 대한 여러 조건들(값의 범위 같은), 여러 자원에 대한 조건(그에 따라 여러 엔트리들(1014)에 결부됨), 또는 그 둘의 조합과 같은 복합적 논리 조건들을 포함할 수 있다.

[0206] 트리거 시기 조건들(1010)은 트래핑되어야 하는 자원의 타입에 따라 조건들을 포함할 수 있다. 예를 들어 레지

스터는 기입되거나, 어떤 특정 값으로 기입되거나 판독될 때 트리거될 수 있다. 다른 예에서 캐시나 포인터도 마찬가지로 기입되거나, 어떤 특정 값으로 기입되거나 판독될 때 트리거될 수 있다. 또 다른 예에서 프로세싱 코어는 그 코어가 아이들일 때 트리거될 수 있다. 또 다른 예에서 프로세서 코어가 정지하거나 슬립상태가 되거나 활성화되도록 명령하는데 사용되는 것과 같은 프로세서간 인터럽트들은 그 인터럽트가 보내지기 전(인터럽트 테이블의 전역 공간에 대해 시도된 액세스 시)이나 인터럽트가 보내진 후(인터럽트 테이블이 기입된 후) 트리거될 수 있다.

[0207] 트리거할 실행 단계(1012)는 명령어의 실행 중 어느 단계에서 시도된 액세스가 트래핑될지에 대한 표시 및 발생된 트리거된 이벤트를 포함할 수 있다. 트리거할 실행 단계(1012)는 주어진 자원을 트래핑할 추가 요건으로서 트리거 시기 조건들(1010)과 함께 사용될 수 있다. 주어진 엔트리를 트래핑하기 위해, 트리거 시기 조건들(1010)은 관련 명령어가 트리거할 실행 단계(1012)에서 특정된 실행의 단계에 도달할 때 평가될 수 있다. 트리거할 실행 단계(1012)는 예컨대 프로세서에 의한 명령어 실행의 다섯 국면들이나 단계들에 해당하는 엔트리들을 포함할 수 있다. 일 실시예에서 그러한 다섯 개의 명령어 실행 단계들은 1) 명령어 가져오기, 2) 명령어의 디코딩, 3) 실행, 4) 결과들에 대한 메모리 위치 액세스, 및 5) 메모리, 레지스터, 또는 다른 위치로 다시 리턴 값을 기입하는 일을 포함할 수 있다. 그러한 실시예에서 트리거할 실행 단계(1012)는 그 다섯 단계들 중 어느 하나 이전이나 이후에 트리거할 기능을 포함할 수 있다. 이것은 총 여섯 개의 서로 다른 트리거링 옵션들-가져오기 전, 디코딩 후(그리고 실행 전), 실행 후(그리고 메모리 위치 액세스 전), 메모리 위치 액세스 후(그리고 리턴 값 기입 전), 및 리턴 값 기입 후-을 제공한다. 이 실행 단계에 따라 트래핑하는 기능이 다른 안티 악성 소프트웨어 시스템들에서 이용 불가한 상당한 융통성을 제공할 수 있다. 예를 들어 특정 명령어의 실행 결과가 미리 알려질 수 있고, 그에 따라 안티 악성 소프트웨어 시스템은 트리거할 실행 단계(1012)의 값을 그 결과들에 대한 메모리 위치를 액세스한 후, 그러나 명령어에 의해 지시된 대로 다시 레지스터에 리턴 값을 기입하기 전으로 세팅할 수 있다. 이것은 안티 악성 소프트웨어 시스템이 기입되게 하지 않고 안티 악성 소프트웨어 시스템이 동작의 결과들을 평가하게 할 수 있다. 그 결과들이 악성 동작들을 나타내는 경우, 네 번째 실행 단계로부터 리턴되는 값 대신 더미 값이 다시 레지스터에 기입될 수 있다. 시도된 실행에 대한 정보가 그 시도가 악성인지 여부를 판단하는 것을 돕기 위해 시도된 실행에 기반하여 트리거된 이벤트의 핸들러로 제공될 수 있다.

[0208] PRCS(1000)의 각각의 자원(1004)은 다른 것(1004)과 자원(1004)의 액세스의 결합들에 해당하는 여러 엔트리들을 가질 수 있다. 그러한 액세스들의 결합은 트래핑될 두 단계 이상의 프로세스를 포함할 수 있다. 예를 들어 엔트리들(1014)은 a) 제어 레지스터들의 액세스와 결합된 인터럽트 서술자 테이블("IDT")에 해당하는 메모리 위치의 액세스 및 b) 범용 레지스터들의 액세스와 결합하여 인터럽트 서술자 테이블에 해당하는 메모리 위치의 액세스를 위한 각각의 엔트리들을 포함할 수 있다. 또한 도 9에서 그러한 각각의 엔트리들은 시스템(900)의 별개 부분들에 의해 처리될 수 있다. 예를 들어 특정 O/S 내 트래핑 에이전트들(919)은 트래핑된 IDT-일반 레지스터 액세스에 대한 정황 정보 수집을 처리할 수 있고, 다른 O/S 내 트래핑 에이전트(919)가 트래핑된 IDT-제어 레지스터 액세스에 대한 정황 정보 수집을 처리할 수 있다.

[0209] 도 9로 돌아가면, O/S 하위 트래핑 에이전트(920)가 PRCS(926)에서 플래그들을 세팅하거나 엔트리들을 추가하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(920)는 그러한 플래그들이나 엔트리들을 결정하기 위해 보안 규칙들(908)과 같은 하나 이상의 보안 규칙들을 액세스하도록 구성될 수 있다. 일 실시예에서 O/S 하위 트래핑 에이전트(920)는 보안 규칙들(908)이나 보호 서버(202)를 조회한 후 O/S 하위 트래핑 에이전트(920)를 호출할 수 있는 트리거된 이벤트 핸들러(922)로부터 그러한 플래그들이나 엔트리들을 세팅하는 명령어들을 수신하도록 구성될 수 있다. 특정한 특권 루틴들의 세트가 플래그들을 세팅하거나 엔트리들을 추가하기 위한 프로세서(902) 및/또는 O/S 하위 트래핑 에이전트(920)에 의해 PRCS(926)로 제공될 수 있다.

[0210] 전자 장치(901)가 하나를 넘는 프로세서를 포함하면, 그러한 각각의 프로세서는 해당 PRCS(926)를 가질 수 있다. 일 실시예에서 시스템(900)은 그러한 각각의 PRCS(926)를 위한 O/S 하위 트래핑 에이전트(920)를 포함할 수 있다. 다른 실시예에서 O/S 하위 트래핑 에이전트(920)는 그러한 각각의 PRCS(926) 안에 나타낸 자원들을 트래핑하도록 구성될 수 있다.

[0211] 시스템(900)이 가상화를 지원하면 PRCS(926) 자체가 가상화될 수 있다. 가상화된 PRCS(926)의 콘텐츠는 해당 프로세서(902)에 의해 가상화되는 자원들에 국한될 수 있다. 그러한 가상화된 PRCS(926)는 가상 머신 모니터 안에 포함될 수 있다. 그러한 경우, O/S 하위 트래핑 에이전트(920)나 트리거된 이벤트 핸들러(922)는 그러한 가상 머신 모니터 내 PRCS(926)를 제어하도록 구성될 수 있다. 다른 실시예에서 O/S 하위 트래핑 에이전트(920)는 그러한 각각의 PRCS(926) 안에 나타낸 자원들을 트래핑하도록 구성될 수 있다. 또한 PRCS 또는 가상화된 프로세서 별로, 각각의 그러한 가상화된 PRCS(926) 안에 엔트리들(1014)이 생성될 수 있고 트리거 플래그들

(1002)이 세팅될 수 있다.

[0212] O/S 하위 트래핑 에이전트(920)는 트래핑된 시도나 전송으로부터 비롯되는 트리거된 이벤트를 트리거된 이벤트 핸들러(922)로 보내도록 구성될 수 있다. 트리거된 이벤트 핸들러(922)는 트리거된 이벤트의 정보 및 하나 이상의 보안 규칙들(908)에 기반하여 어떤 적절한 후속 액션을 수행하도록 구성될 수 있다. 예를 들어 트리거된 이벤트 핸들러(922)는 시도된 명령의 실행을 허용하지만 실행 후 그 결과들의 통지를 필요로 하도록 구성될 수 있다. 다른 예에서 트리거된 이벤트 핸들러(922)는 명령이나 전송에 대한 실행 전부를 건너뛰도록 구성될 수 있다. 그러한 예는 리턴 값이 요구되지 않는 경우 적용될 수 있다. 또 다른 예에서, 예컨대 실행을 복구 루틴의 어드레스로 보내는 "JMP" 명령어를 사용함으로써 실행이 새로운 위치로 옮겨질 수 있다.

[0213] 동작 시 O/S 하위 트래핑 에이전트(920) 및 트리거된 이벤트 핸들러(922)는 전자 장치(901) 상에서 동작할 수 있다. O/S 하위 트래핑 에이전트(920)는 전자 장치(901)의 운영체제들의 레벨 밑에서 동작될 수 있다. 또한 트리거된 이벤트 핸들러(922) 또한 전자 장치(901)의 운영체제들의 레벨 밑에서 동작될 수 있다. 트리거된 이벤트 핸들러(922)는 보안 규칙들(908)이나 보호 서버(202)를 조회하여 PRCS(926) 안에 어떤 플래그들(1002)이나 엔트리들(1014)을 세팅할지 결정할 수 있다. 트리거된 이벤트 핸들러(922)는 PRCS(926) 안에 어떤 플래그들(1002)이나 엔트리들(1014)을 세팅할지 O/S 하위 트래핑 에이전트(920)에 지시할 수 있다. 사용 중인 애플리케이션(910), 검출된 악성 소프트웨어에 대한 다른 표시들, 앞서 트리거된 이벤트들, 또는 전자 장치(901)에 대한 관리자 설정들과 같이 검출된 다양한 조건들에 따라, O/S 하위 트래핑 에이전트(920) 및 트리거된 이벤트 핸들러(922)가 전자 장치(901)의 동작 중에 동적으로, PRCS(926) 안의 트리거 플래그들(1002)을 변경하거나 새로운 엔트리들(1014)을 추가할 수 있다. 그러한 동적 변경이 기초할 정보는 예컨대 O/S 하위 트래핑 에이전트(920)나 O/S 내 에이전트(919)로부터 나올 수 있다. PRCS(926)의 엔트리들(1014)은 자원(1004)이나 자원 타입(1006)에 따라 식별될 수 있다. 트리거 타입(1008)은 이어지는 트래핑된 이벤트를 동기나 비동기로 설정하도록 세팅될 수 있다. 트리거 시기 조건들(1010)은 트리거할 실행 단계(1012)와 같이, 인터셉트된 요청이 트리거된 이벤트를 어떤 조건 하에서 생성할 것인지를 설정하도록 세팅될 수 있다.

[0214] PRCS(926)의 엔트리들은 시스템(900)이 마주치는 다양한 조건들에 따라 동적으로 가능(인에이블) 또는 불능(디세이블)될 수 있다. 예를 들어 O/S 하위 트래핑 에이전트(920)는 트래핑된 액세스 시도가 많은 긍정 오류(false positives)들고 함께 빈번히 일어나기 때문에, 트리거된 이벤트 핸들러(922)가 전자 장치(901)가 악성 소프트웨어 공격을 받고 있다는 표시를 수신하는 시점까지, 비싼 트래핑 동작을 디세이블할 수 있다. 그런 다음 O/S 하위 트래핑 에이전트(920)는 트래핑 동작을 인에이블할 수 있다. 일 실시예에서 그러한 비싼 조건 하에서, 전자 장치(901)를 더 위험하게 하는 알려지지 않은 악성 소프트웨어 액션들을 방지하기 위해 하나 이상의 프로세서 자원들(924)에 대한 확장된 트래핑이 인에이블될 수 있다. 그러한 확장된 트래핑은 실질적으로, 프로세서, 가상화 프로세서, 스레드, 프로세스 또는 애플리케이션의 전체 실행 환경을 셧다운하는 데까지 확장할 수 있다.

[0215] 프로세서 자원(924)의 요청은 애플리케이션(910), 드라이버(911) 또는 운영체제(913)와 같이 시스템(900) 내 운영체제들의 레벨에 있는 개체로부터 일어날 수 있다. 그 요청은 프로세서 자원들(924)을 지날 수 있지만 O/S 하위 트래핑 에이전트(920)에 의해 인터셉트될 수 있다. 또한 정보나 전송이 다양한 프로세서 자원들(924)을 통해 프로세서로부터 생성될 수 있다. 정보나 전송은 O/S 하위 트래핑 에이전트(920)에 의해 인터셉트될 수 있다.

[0216] O/S 하위 트래핑 에이전트(920)는 그 정보나 전송이 PRCS(926) 내 엔트리들(1014)의 어떤 트리거 시기(1010) 필드들과 매치할 경우 자원의 액세스를 트래핑하기 위해 PRCS(926)를 이용할 수 있고, 이어서 트리거된 이벤트를 생성할 수 있다. "ON"으로 세팅되어 있는 트리거 플래그들(1002)에 의해 인에이블되었던 엔트리들(1014)은 시도된 액세스나 정보나 전송에 매칭될 수 있다. 액세스될 자원은 자원 필드(1004) 및/또는 자원 타입 필드(1006)와 비교될 수 있다. 액세스될 자원이 그러한 필드들과 매치하면, 트리거 시기 조건들(1010)이 평가될 수 있다. 트리거 시기 조건들(1010)이 시스템 정보나 요청에 대한 정보와 매치하면 PRCS(926)는 트리거된 이벤트를 생성할 수 있다. 트리거할 실행 단계(1012)는 트리거된 이벤트를 언제 생성할지를 판단하기 위해 이용될 수 있다. 예를 들어 트리거된 이벤트는 가져오기 명령 전, 가져오기 명령 후, 실행 후, 후속 쓰기를 위해 메모리가 액세스된 후, 또는 기입을 다시 하기 위해 레지스터와 같은 다른 자원이 액세스된 후에 생성될 수 있다. 또한 트리거된 이벤트는 "Interrupt_Sleep" 같은 프로세서간 인터럽트와 같은 정보의 전송 또는 생성 시도에 대해 그 인터럽트가 인터럽트 테이블에 기입되거나 보내지기 전후에 생성될 수 있다. 생성된 트리거된 이벤트는 트리거 타입(1008)에 따라 동기적 혹은 비동기적인 것이 될 수 있다. O/S 하위 트래핑 에이전트(920)는 동기 트리거된 이벤트가 생성된 경우 시도된 자원 액세스나 통신문의 생성 실행을 중지하여, 해당 이벤트의 처리를 기

다릴 수 있다. O/S 하위 트래핑 에이전트(920)는 비동기 트리거된 이벤트가 생성된 경우 시도된 자원 액세스나 통신문의 생성 실행을 허용할 수 있다. O/S 하위 트래핑 에이전트(920)는 시도가 일어났던 메모리 어드레스, 결과들이 어디에 기입되어야 했는지, 또는 어떤 다른 적절한 정보와 같이 시도에 대한 추가 정황 정보를 트리거된 이벤트 안에 추가할 수 있다.

[0217] O/S 하위 트래핑 에이전트(920)는 트리거된 이벤트가 의심스러운지 여부를 결정할 목적의 트리거된 이벤트와 관련된 정보를 포함할 수 있다. 예를 들어 O/S 하위 트래핑 에이전트(920)는 메모리의 어떤 부분으로부터 액세스 시도가 있었는지와 같은 정보를 판단할 수 있다. 그 메모리 부분은 전자 장치(903) 상에서 실행되는 알려진 프로세스들, 애플리케이션들 또는 프로그램들에 대해, 트리거된 이벤트 핸들러(922)에 의해 상관될 수 있다. 시도된 액세스가 알려지지 않거나 허가되지 않은 프로세스, 애플리케이션 또는 프로그램으로부터 일어났으면, 해당 시도는 의심스러운 것일 수 있다. 트리거된 이벤트 핸들러(922)는 그러한 상관관계를 판단하기 위해 O/S 내 보안 에이전트(919)로부터의 정보를 이용할 수 있다. 다른 예에서 O/S 하위 트래핑 에이전트(920)는 상태 머신에 기록된 것들과 같이 앞서 트리거된 이벤트들에 관한 정보를 제공할 수 있다. 현재 트리거된 이벤트들과 관련된 앞서 트리거된 이벤트들은 해당 시도들이 의심스러운지 여부에 대한 정황 정보를 제공할 수 있다.

[0218] O/S 하위 트래핑 에이전트(920)는 보안 규칙들(908)에 따라 O/S 내 에이전트(919)로부터 트리거된 이벤트 안의 정보 및/또는 정황 정보를 평가함으로써 이벤트를 처리할 수 있는 트리거된 이벤트 핸들러(922)로 트리거된 이벤트를 보낼 수 있다. 그에 따른 적합한 액션이 결정될 수 있고, 트래핑된 시도에 적용하기 위해 O/S 하위 트래핑 에이전트(920)로 다시 보내질 수 있다. 그러한 액션은 해당 시도를 허용하는 동작, 명령어 실행을 거부하는 동작, 또는 악성 소프트웨어의 동작을 피하기 위해 다른 데이터나 명령어들을 대체하는 동작을 포함할 수 있다.

[0219] O/S 하위 트래핑 에이전트(920)는 앞으로 시도되는 액세스를 트래핑함에 있어 다음 참조를 위해, 트리거된 이벤트들을 저장할 수 있다. 예를 들어 악성 동작은 여러 개의 명령어들이 프로세서 자원들(924)에 의해 실행될 것을 요할 수 있다. 따라서 그러한 악성 양태에 대한 각각의 단계가 PRCS(926)의 각각의 엔트리(1014) 안에 반영될 수 있다. O/S 하위 트래핑 에이전트(920)는 악성 동작의 최초 단계를 트래핑할 수 있으며, 그 최초 단계 자체는 악성이 아닐 수 있지만 다만 후속 단계들과 결합할 때 악성이 될 수 있다. 그 경우, 그러한 단계에 대한 엔트리(1014)는 비동기적으로 트리거하도록 세팅될 수 있는데, 이는 O/S 하위 트래핑 에이전트(920)나 PRCS(926)가 앞서 처리된 시도들을 인지할 수 있도록 그 조건이 단지 상태 머신 안에 기록되기 때문이다. 악성 동작의 제2단계 트래핑은 트리거 시기 조건(1010)으로서 제1단계의 트래핑을 가질 수 있다.

[0220] 도 11은 전자 장치 상의 보안에 민감한 프로세서 자원들에 대한 소프트웨어 액세스를 규제하는 방법(1100)의 실시예이다. 단계 1105에서 보안 규칙들이 액세스되어, 단계 1110에서 어떤 프로세서 자원들이나 프로세서 전송들이 보호되어야 하는지를 결정하도록 한다. 전자 장치의 운영체제 레벨 하위에서 동작하는 트래핑 에이전트가 어떤 자원들과 전송들이 트래핑되어야 하는지를 결정할 수 있다. 그러한 트래핑 에이전트는 예컨대 가상 머신 모니터, 펌웨어, 또는 프로세서의 마이크로코드 안에서 동작할 수 있다.

[0221] 단계 1115에서, 트래핑될 자원들이나 전송들에 해당하는 엔트리들이 프로세서 자원 제어 구조에 기입될 수 있으며, 프로세서 자원 구조는 특정된 조건 하에서 지정된 자원들이나 전송들의 동작, 액세스 또는 다른 이용을 트래핑하도록 설정될 수 있다. PRCS 내 엔트리들은 자원의 식별자, 자원 타입, 이벤트가 트리거될 조건, 트리거가 비동기인지 동기인지 여부 및 시도된 액세스나 전송이 어느 실행 단계(존재하는 경우)에서 트리거된 이벤트를 도출하는지와 함께 기입될 수 있다. 단계 1120에서 PRCS 내 엔트리들은 또한, 그 엔트리가 트래핑을 위해 활성화되는지 되지 않는지 여부를 나타내는 트리거 또는 인에블먼트(enablement) 플래그와 함께 기입될 수 있다. 트리거 플래그가 세팅되지 않으면, 엔트리는 휴면상태로서 자원들에 대해 시도된 액세스들을 트래핑하는 데 사용되지 않을 수 있다.

[0222] 단계 1125에서 자원들에 대한 액세스나 통신문들의 생성이 감시될 수 있다. 그러한 감시는 PRCS를 통해 일어날 수 있다. 전자 장치 내 개체들은 프로세서 통신문들이나 프로세서 자원을 액세스하려는 시도를 발생하고자 시도할 수 있다. 그러한 자원 액세스 시도들은 전자 장치의 운영체제들의 레벨로부터 발생될 수 있다. 자원을 액세스하고자 하는 명령어, 명령 또는 다른 시도가 PRCS 안의 개체의 자원 식별자와 매칭하되 그 엔트리가 활성화되어 있으면, 해당 시도는 트래핑될 수 있다. 마찬가지로, PRCS 안의 개체의 자원 식별자와 매칭하는 프로세서 전송이 발생되되 그 엔트리가 활성화되어 있으면, 해당 시도는 트래핑될 수 있다. 일 실시예에서 자원을 액세스하거나 전송을 발생하려는 시도는 언제 트리거할지를 특정하는 추가 기준이 만족될 때 트래핑될 수 있다. 예를 들어 제어 레지스터에 시도된 쓰기는 제어 레지스터가 기입될 때 트래핑될 수 있다. 다른 예에서 제어 레

지스터에 시도된 쓰기는 제어 레지스터가 특정 값으로 기입될 때 트래핑될 수 있다.

[0223] 단계 1130에서, 시도된 액세스나 전송이 트래핑되었는지 여부가 판단될 수 있다. 아무 시도도 트래핑되지 않았으면, 단계 1140에서 PRCS 내 엔트리들이 조정되어야 하는지 여부가 판단될 수 있다. 그러한 조정은 그러한 엔트리들의 인에이블이나 디세이블, 새 엔트리들의 추가, 또는 엔트리들의 기준이나 설정사항의 조정을 포함할 수 있다. 방법(1100)은 단계 1125로 돌아갈 수 있다. 그러한 조정은 예컨대 전자 장치에서 검출된 새로운 악성 소프트웨어 위협들, 시간의 경과, 이전에 트래핑된 시도들, 또는 관리자의 설정사항들에 기반할 수 있다.

[0224] 단계 1145에서, 어떤 시도가 트래핑되었으면, 그에 따라 트리거된 이벤트가 동기여야 하는지 비동기여야 하는지 여부가 판단될 수 있다. 트리거 타입이 동기가 아니면, 방법(1100)은 단계(1150)으로의 진행과 나란히 단계(1125)로 돌아갈 수 있다. 트리거 타입이 동기이면, 단계 1150에서 트래핑된 시도에 대한 정보가 저장될 수 있다. 그러한 정보는 예컨대 트래핑된 시도가 트리거된 이벤트를 양산해야 할지 여부에 대한 앞으로의 판단 시 상태 머신에 의해 사용될 수 있다. 단계 1155에서 트리거의 모든 조건들이 만족되는지 여부가 판단될 수 있다. 그러한 조건들은 예컨대 소정 값이 자원에 기입되거나 해당 요청이 메모리 안의 특정 위치들로부터 발생할 것(또는 발생하지 않을 것)을 요할 수 있다. 또한 그러한 조건들은 다른 시도들이 앞서 트래핑되었을 것을 요할 수 있다. 그러한 시도들에 대한 정보가 상태 머신에 의해 액세스되고 저장될 수 있다. 트리거의 모든 조건들이 만족되지 않으면, 방법(1100)은 단계 1125로 돌아갈 수 있다.

[0225] 트리거의 모든 조건들이 만족되면, 단계 1155에서 실행의 어떤 특정 단계(있는 경우)에서 트리거된 이벤트가 발생되어야 하는지가 결정될 수 있다. 그러한 단계들에는 예컨대 시도 시의 명령어가 페치(가져오기)되기 전, 명령어가 페치된 후, 명령어가 실행된 후, 결과를 읽기 위해 메모리가 액세스된 후, 또는 어떤 값이 다시 기입된 후가 포함될 수 있다. 또한 그러한 단계들은 프로세서간 인터럽트가 실행되기 전이나 후를 포함할 수 있다. 지정된 실행 단계가 수행되면, 해당 시도에 대해 트리거된 이벤트가 단계 1165에서 발생될 수 있다. 해당 시도의 소스 또는 목적지 어드레스, 또는 관련된 자원들과 같은 정황 정보가 단계 1170에서 트리거된 이벤트와 함께 포함될 수 있고 단계 1175에서 핸들러로 전달된다.

[0226] 단계 1180에서 보안 규칙들이 조회되어 단계 1185에서 트리거된 이벤트가 의심스러운지, 관리자 설정사항에 의해 허용되지 않는지, 혹은 악성 소프트웨어를 나타내는지 여부가 판단된다. 트리거된 이벤트, 전자 장치의 운영체제 내 다른 이벤트들, 또는 관리자 설정사항들 같은 것의 정황 정보가 트리거된 이벤트로의 보안 규칙들의 적용을 평가하는 데 사용될 수 있다. 트리거된 이벤트가 의심스러운 것이 아니면, 단계 1187에서 트래핑 에이전트로 통지될 수 있고, 방법(1100)은 단계 1125로 돌아갈 수 있다. 트리거된 이벤트가 의심스러운 것이면, 단계 1190에서 그에 따른 교정 액션이 트래핑 에이전트로 보내질 수 있다. 그러한 교정 액션은 자원을 액세스하거나 프로세서 전송을 발생하려는 특정 시도에 따라 좌우될 수 있다. 예를 들어 악성 명령어는 판독 또는 기입될 값이 스푸핑되게(spoofed)할 수 있고, 혹은 점프 명령이 복구 루틴으로 재지향될 수 있다. 단계 1195에서 교정 액션이 적용될 수 있다. 방법(1100)은 단계(1125)로 돌아갈 수 있다.

[0227] 도 12는 전자 장치(1201) 상의 운영체제 하위 트래핑을 이용하여 메모리를 보호하기 위해 소프트웨어 액세스를 규제하는 시스템(1200)의 실시예이다. 시스템(1200)은 운영체제(1213)와 같은 전자 장치(1201)의 운영체제들에서 실행되는 소프트웨어 기반 개체들로부터 메모리를 액세스하려는 악의적 시도들을 검출하기 위해 전자 장치(1201) 상에서 동작하도록 구성된 O/S 하위 보안 에이전트(1220)를 포함할 수 있다. 또한 O/S 하위 보안 에이전트(1220)는 메모리에 시도된 어떤 액세스들을 트래핑하고 트래핑된 동작에 상응하여 생성된 트리거된 이벤트를 어떻게 처리할지를 결정하기 위해 하나 이상의 보안 규칙들(1208) 및 메모리 맵(1206)을 이용하도록 구성될 수 있다. O/S 하위 보안 에이전트(1220)는 트리거된 이벤트를 허용하거나 거부하거나 그에 대해 다른 교정 액션을 수행하도록 구성될 수 있다.

[0228] 전자 장치(1201)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701), 도 9의 전자 장치(901) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(1201)는 물리적 메모리(1203)와 같은 메모리에 연결된 하나 이상의 프로세서들(1202)을 포함할 수 있다. 프로세서(1202)는 도 2의 프로세서(208), 도 4의 프로세서(408), 도 7의 프로세서(702), 도 9의 프로세서(902) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 물리적 메모리(1203)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703), 도 9의 메모리(903) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(1201)는 하나 이상의 보안 규칙들(1221)에 연결된 O/S 내 보안 에이전트(1219)를 포함할 수 있는 운영체제(1213)를 포함할 수 있다.

운영체제(1213)는 도 1의 운영체제(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 9의 운영체제(913) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(1219)는 도 1의 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418) 및/또는 도 7의 O/S 내 보안 에이전트(719), 도 9의 O/S 내 보안 에이전트(919) 또는 이들의 어떤 적절한 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0229] O/S 하위 보안 에이전트(1220)는 도 1의 O/S 하위 트래핑 에이전트(104)나 트리거된 이벤트 핸들러(108), 도 2의 SVMM(216) 또는 SVMM 보안 에이전트(217), 도 4의 펌웨어 보안 에이전트들(440, 442), O/S 하위 에이전트(450), 또는 PC 펌웨어 보안 에이전트(444), 도 5의 펌웨어 보안 에이전트(516), 또는 도 7의 마이크로코드 보안 에이전트(708)나 O/S 하위 에이전트(712), 도 9의 O/S 하위 트래핑 에이전트(920) 또는 트리거된 이벤트 핸들러(922), 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0230] 보안 규칙들(1208)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(222), 도 4의 보안 규칙들(434, 436, 438), 도 5의 보안 규칙들(518), 도 7의 보안 규칙들(707, 723), 도 9의 보안 규칙들(908) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보안 규칙들(1221)은 도 2의 보안 규칙들(220), 도 4의 보안 규칙들(420), 도 7의 보안 규칙들(721), 도 9의 보안 규칙들(921) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0231] O/S 하위 보안(1220)은 전자 장치(1201)의 메모리에 대한 액세스를 인터셉트하도록 구성될 수 있다. 그러한 메모리는 예컨대 물리적 메모리(1203)의 어드레스들에 시도된 액세스나 가상 메모리(1204)의 페이지들에 대해 시도된 액세스를 포함할 수 있다. 상기 시도된 액세스는 운영체제(1213) 또는 애플리케이션(1210)이나 드라이버(1211)와 같이 전자 장치(1201) 상에서 실행할 운영체제(1213)을 이용하는 개체들로부터 발생될 수 있다.

[0232] 일 실시예에서 O/S 하위 보안(1220)에 의해 보호되는 메모리는 가상 메모리(1204)를 포함할 수 있다. 가상 메모리(1204)는 물리적 어드레스 및/또는 스토리지로부터 추상화되어 있는 운영체제(1213), 애플리케이션(1210) 또는 드라이버(1211)와 같은 개체들에서 이용 가능한 메모리를 포함할 수 있다. 가상 메모리(1204)는 실제 사용된 공간들이 본질적으로 물리적 메모리(1203)와 같은 실제적인 물리적 메모리 및/또는 디스크와 같은 스토리지에 퍼져있을 수 있지만, 운영체제(1213), 애플리케이션(1210) 또는 드라이버(1211)와 같은 개체들에게 인접한 메모리 블록으로서 보여질 수 있다. 가상 메모리(1204)는 프로세서(1202)의 확장에 맞춰 가상화될 수 있다. 가상 메모리(1204)의 어드레스 공간은 메모리 페이지들로 분할될 수 있다. 메모리 페이지들은 4 킬로바이트와 같이 각각이 동일한 사이즈로 되어 있을 수 있다. 전자 장치(1201)는 가상 메모리(1204)의 가상 어드레스들을 물리적 메모리(1203)와 같은 메모리의 물리적 어드레스들이나 스토리지의 어드레스들로 변환하기 위해 페이지 테이블들을 이용하도록 구성될 수 있다. 전자 장치(1201)는 가상 메모리(1204)의 가상 어드레스들을 물리적 메모리(1203)와 같은 메모리의 물리적 어드레스들이나 스토리지의 어드레스들로 변환하도록 구성된 메모리 관리 유닛(1214)("MMU")을 포함할 수 있다. 가상 메모리(1204)의 페이지들은 색인화될 수 있다. 가상 메모리(1204) 페이지들에 대해 시도된 액세스는 그 페이지에 대해 시도되는 읽기, 쓰기 또는 실행을 포함할 수 있으며, O/S 하위 보안 에이전트(1220)는 그 시도를 트래핑하도록 구성될 수 있다. 일 실시예에서 가상 메모리(1204)의 페이지는 물리적 메모리 어드레스나 스토리지의 어드레스에 해당할 수 있다. 다른 실시예에서 가상 메모리(1204)의 각각의 페이지는 물리적 메모리 어드레스에 해당할 수 있다. 또 다른 실시예에서, 운영체제(1213)의 특정 부분들과 같은 소정 콘텐츠들을 포함하는 페이지들은 고정되어 전자 장치(1201)의 동작 중에는 변경되지 못할 수 있다.

[0233] 다른 실시예에서 O/S 하위 보안 에이전트(1220)에 의해 보호되는 메모리는 물리적 메모리(1203)를 포함할 수 있다. 물리적 메모리(1203)는 정의된 요소를 포함하는 메모리 범위의 베이스 어드레스일 수 있는 물리적 메모리(1203) 내 특정 어드레스들을 나타내는 마커들 (A), (B), (C), (D), (E), (F), (G), (H), (I), (J), 및 (K)로 보여진 바와 같이, 물리적 메모리의 어드레스들을 통해 액세스될 수 있다. 물리적 메모리(1203)는 특정 메모리 어드레스에 대해 시도된 읽기, 쓰기, 또는 실행을 통해 액세스될 수 있고, O/S 하위 보안 에이전트(1220)는 그러한 시도를 트래핑하도록 구성될 수 있다. 예를 들어 시도된 쓰기는 "MOV Addr1, Value" 명령어의 형식을 취할 수 있으며, 여기서 변수 "Value"로 표현된 값이 "Addr1"으로 표현된 특정 메모리 어드레스에 기입된다. 물리적 메모리(1203) 어드레스로 기입하는 명령어가 사용될 수 있다. 시도된 읽기는 "MOV Value, Addr1" 같은 명령어의 형식을 취할 수 있으며, 여기서 변수 "Value"로 표현된 값이 "Addr1"으로 표현된 특정 메모리 어드레스로부터 판독된다. 물리적 메모리(1203) 어드레스로부터 판독하는 명령어가 사용될 수 있다. 시도되는 실행은 "MOV EIP, Addr1"과 같이 물리적 메모리(1203) 어드레스에 "EIP"와 같은 명령어 포인터 레지스터를 로딩하는 명

령의 형식을 취할 수 있다. 그러한 명령어는 "Addr1"로 표현되는 어드레스에서 시작하는 코드를 실행하도록 구성될 수 있다. 메모리 내 어떤 어드레스를 실행하기 위한 어떤 명령이 사용될 수 있다.

[0234] O/S 하위 보안 에이전트(1220)는 가상 메모리(1204)에 시도된 액세스를 인터셉트하도록 구성될 수 있다. 또한 O/S 하위 보안 에이전트(1220)는 물리적 메모리(1203)에 시도된 액세스를 인터셉트하도록 구성될 수 있다. 일 실시예에서, 가상 메모리(1204)에 대한 요청은 인터셉트될 수 없으나, MMR가 가상 메모리(1204) 페이지를 물리적 메모리(1203) 페이지로 변환한 후 물리적 메모리(1203)에 대한 이후의 해당 액세스 시도에 대해 O/S 하위 보안 에이전트(1220)가 인터셉트하도록 구성될 수 있다. 다른 실시예에서는 물리적 메모리(1203)에 대해 시도된 액세스가 가상 메모리(1204)를 통해 변환되지 않고 바로 이루어질 수 있으며, O/S 하위 보안 에이전트(1220)는 그 시도된 액세스를 인터셉트하도록 구성될 수 있다. 또 다른 실시예에서, 가상 메모리(1204)에 대해 시도된 액세스가 인터셉트될 수 있으나, O/S 하위 보안 에이전트(1220)는 물리적 메모리(1203) 어드레스에 대한 이후의 액세스를 인터셉트하도록 구성되지 않을 수 있다.

[0235] O/S 하위 보안 에이전트(1220)는 O/S 내 보안 에이전트(1219)와 통신 가능하게 연결될 수 있다. O/S 하위 보안 에이전트(1220)는 O/S 내 보안 에이전트(1219)로부터 전자 장치(1201)의 메모리에 대해 시도된 액세스에 대한 정황 정보를 수신하도록 구성될 수 있다. O/S 내 보안 에이전트(1219)에 의해 제공되는 정황 정보는 전자 장치(1201)의 메모리에 대한 특정 액세스를 시도했던 개체들의 아이디를 포함할 수 있다.

[0236] O/S 하위 보안 에이전트(1220)는 메모리 맵(1206)에 통신 가능하게 연결되거나 그것을 포함할 수 있다. 메모리 맵(1206)은 파일, 레코드, 데이터 구조, 또는 다른 적절한 개체 안에서 구현될 수 있다. 메모리 맵(1206)은 전자 장치(1201)의 다양한 개체들의 메모리 내 위치에 관한 정보를 포함할 수 있다. 예를 들어 어떤 프로세스가 실행되기 위해 전자 장치(1201)의 메모리 안에 로딩될 때, 메모리 맵(1206)은 가상 메모리(1204)의 어느 메모리 페이지들 또는 물리적 메모리(1203)의 어느 어드레스 페이지들이 그 프로세스를 포함하는지에 관한 정보를 포함할 수 있다. 전자 장치(1201) 내 메모리의 가상화 구현에 따라, 일부 콘텐츠가 디스크와 같은 스토리지에 로딩될 때, 프로세스의 콘텐츠 전부가 물리적 메모리(1203)에 로딩되거나 로딩되지 않을 수 있다. 그러한 콘텐츠가 액세스되기 위해, 이들은 물리적 메모리(1203)에 로딩될 수 있다. 그 경우 메모리 맵(1206)은 물리적 메모리(1203) 안인지 디스크와 같은 스토리지 안인지, 콘텐츠가 저장된 어드레스들에 대한 정보를 포함할 수 있다. O/S 하위 보안 에이전트(1220)는 가상 메모리(1204) 페이지나 물리적 메모리(1203) 어드레스 내 어떤 주어진 콘텐츠의 아이디나 소유자를 판단하기 위해 메모리 맵(1206)을 이용하도록 구성될 수 있다. O/S 하위 보안 에이전트(1220)는 예컨대 운영체제(1213)의 동작을 프로파일링하고 그런 다음 메모리의 어디에 여러 민감한 구성요소들이 위치되는지를 판단함으로써 메모리 맵(1206)을 구축할 수 있다. 메모리를 액세스하려는 시도들-운영체제(1213) 커널 로딩 또는 커널 모드 명령어들의 실행-이 이루어질 때, O/S 하위 보안 에이전트(1220)는 운영체제(1213)의 어느 부분이 로딩되거나 실행되는지를 판단하기 위한 O/S 내 보안 에이전트(1219)와 통신하도록 구성될 수 있다. 다른 예에서 O/S 하위 보안 에이전트(1220)는 그러한 가상 메모리(1204) 페이지의 메모리 범위의 콘텐츠의 해시나 디지털 서명을 판단하도록 구성될 수 있다. 해시나 디지털 서명은 보안 규칙들(1208)에 포함되거나 보호 서버(202)로부터 획득될 수 있는 알려진 값들과 비교될 수 있다. 알려진 값들은 예컨대 운영체제(1213)의 일부가 식별된 이전의 특징화의 결과일 수 있다. 매핑될 요소들은 보안 규칙들(1208)에 의해 결정될 수 있다. O/S 하위 보안 에이전트(1220)는 요소들이 전자 장치(1201)의 메모리 내 한 위치에서 다른 위치로 복사될 때 메모리 맵(1206) 내 요소들의 이동을 추적하도록 구성될 수 있다.

[0237] 도 13은 메모리 맵들의 실시예들을 도시한다. 일 실시예에서 가상 메모리 맵(1302)은 가상 메모리 내 해당 위치를 통해 추적되는 요소들의 매핑을 포함할 수 있다. 다른 실시예에서 물리적 메모리 맵(1304)은 물리적 메모리 내 해당 위치를 통해 추적되는 요소들의 매핑을 포함할 수 있다. 다양한 실시예들에서 가상 메모리 맵(1302) 및 물리적 메모리 맵(1304)은 어떤 요소가 두 매핑들 안에서 추적될 수 있도록 함께 매핑될 수 있다.

[0238] 가상 메모리 맵(1302)은 서로 다른 가상 메모리 페이지들을 반영할 수 있다. 가상 메모리 맵(1302)은 예컨대 페이지 디렉토리와 같은 커널 운영체제 데이터 구조가 메모리 페이지 1과 메모리 페이지 2에서 발견될 수 있음을 예시할 수 있다. 다른 예에서 "Fn1"이라 불리는 특정 프로세스, 함수, 또는 루틴이 메모리 페이지들 4-6에서 발견될 수 있다. 또 다른 예에서 시스템 서비스 발송 테이블("SSDT")의 허가를 위한 데이터 구조들이 페이지 8에서 보여질 수 있다. 또 다른 예에서 "Fn2"라 불리는 특정 프로세스, 함수, 또는 루틴이 메모리 페이지 8 및 메모리 페이지 9에서 발견될 수 있다.

[0239] 물리적 메모리 맵(1304)은 물리적 메모리와의 요소들의 위치를 반영할 수 있다. 물리적 메모리 내 요소들의 일부가 인접하지 않은 세그먼트들이나 블록들로 메모리 상에 퍼져 있을 수 있다. 또한, 물리적 메모리 내 요소들

의 일부가 임의의 순서로 메모리 상에 퍼져 있을 수 있다. 각각의 세그먼트의 크기는 다양할 수 있다. 세그먼트는 베이스 어드레스로부터 오프셋된 어드레스에서 시작할 수 있다. 도 13에 도시된 베이스 어드레스는 00x000로 어드레스 FFxFFF에서 종료된다. 물리적 메모리의 다양한 세그먼트들의 시작을 나타내는 어드레스들이 (A)-(O)로 표시된다. 물리적 메모리의 여러 세그먼트들 안에 포함되는 요소들에 있어서, 그 요소들의 순서가 언급될 수 있다. 물리적 메모리에서 어떤 요소의 여러 세그먼트들은 요소의 한 세그먼트의 끝이 요소의 다음 세그먼트를 가리킬 수 있는 포인터들에 의해 서로 링크될 수 있다.

[0240] 예를 들어 Fn1은 (A) 및 (B), (J) 및 (K), 그리고 (M) 및 (N) 사이의 세그먼트들로 매핑될 수 있다. 다른 예에서 SSDT 허가들은 (G) 및 (H) 사이의 세그먼트로 매핑될 수 있다. 또 다른 예에서, 페이지 디렉토리 데이터 구조가 (O) 및 FFxFFF, (F) 및 (G), 그리고 (I) 및 (J) 사이의 세그먼트들로 매핑될 수 있다. 또 다른 예에서 Fn2는 (H) 및 (I), 그리고 (B) 및 (C) 사이의 세그먼트들로 매핑될 수 있다.

[0241] 도 12로 돌아가면, O/S 하위 보안 에이전트(1220)는 메모리의 어느 부분을 보호하고 그들을 어떻게 보호할지를 결정하기 위해 보안 규칙들(1208)을 조회하도록 구성될 수 있다. 예를 들어 보안 규칙들(1208)은 페이지 디렉토리 데이터 구조가 전자 장치(1201)의 어떤 특권을 가진 개체들로만 기입될 수 있다는 것을 가리키도록 구성될 수 있다. 따라서, 페이지 디렉토리 데이터 구조로의 기입 시도들이 트래핑될 수 있고, 기입을 시도한 요소들이 검사되어 그들이 안전한지, 알려지지 않은 것인지, 불안전하다고 알려져 있는지 여부를 판단할 수 있다. O/S 하위 보안 에이전트(1220)는 페이지 디렉토리 데이터 구조가 메모리의 어디에 위치하는지를 판단하기 위해 메모리 맵(1206)을 조회하도록 구성될 수 있다. O/S 하위 보안 에이전트(1220)가 예컨대 가상 머신 모니터 내에서 전부나 일부가 구현되는 경우, O/S 하위 보안 에이전트(1220)는 가상 메모리(1204)의 메모리 페이지들 1 및/또는 2로 시도된 어떤 쓰기를 트래핑하기 위해 제어 구조 안의 플래그를 세팅하도록 구성될 수 있다. O/S 하위 보안 에이전트(1220)가 다른 예에서 전체나 일부가 마이크로코드로 구현되는 경우, O/S 하위 보안 에이전트(1220)는 물리적 메모리(1203)의 어드레스들 (O) 및 FFxFFF, (F) 및 (G), 그리고 (I) 및 (J) 사이의 어드레스 범위들 안에서 메모리 어드레스들에 대해 시도된 어떤 쓰기를 트래핑하도록 제어 구조 안의 플래그를 세팅하도록 구성될 수 있다.

[0242] 다른 예에서 보안 규칙들(1208)은 Fn1이 전자 장치의 어떤 특권을 가진 개체들에 의해서만 호출될 수 있다는 것을 가리키도록 구성될 수 있다. 따라서, Fn1을 실행하고자 하는 시도들이 트래핑될 수 있고, Fn1을 호출하는 요소들이 검사되어 그들이 안전한지, 알려지지 않은 것인지, 불안전하다고 알려져 있는지 여부를 판단할 수 있다. O/S 하위 보안 에이전트(1220)는 Fn1이 메모리의 어디에 상주하는지를 판단하기 위해 메모리 맵(1206)을 조회하도록 구성될 수 있다. O/S 하위 보안 에이전트(1220)가 예컨대 가상 머신 모니터 내에서 전부나 일부가 구현되는 경우, O/S 하위 보안 에이전트(1220)는 가상 메모리(1204)의 메모리 페이지들 4, 5 및/또는 6으로 시도된 어떤 실행을 트래핑하기 위해 제어 구조 안의 플래그를 세팅하도록 구성될 수 있다. O/S 하위 보안 에이전트(1220)가 다른 예에서, 마이크로코드를 통해 전부나 일부가 구현되는 경우, O/S 하위 보안 에이전트(1220)는 물리적 메모리(1203)의 메모리 페이지들 (A)에 대해 시도된 어떤 실행을 트래핑하기 위해 제어 구조 안의 플래그를 세팅하도록 구성될 수 있다. Fn1의 여러 부분들이 각기 실행될 수 있는 어떤 경우들에서, O/S 하위 보안 에이전트(1220)는 물리적 메모리(1203)의 (A) 및 (B), (M) 및 (N), 어드레스들 (O) 및 FFxFFF, (F) 및 (G), (J) 및 (K), 또는 (I) 및 (J) 사이의 범위 안에 있는 어떤 메모리 어드레스의 실행 시도를 트래핑하도록 구성될 수 있다.

[0243] 일 실시예에서 O/S 하위 보안 에이전트(1220)는 어떤 개체가 메모리에 기입하라는 호출을 행하였는지를 판단하고 그런 다음 그 개체가 기입을 행하는 것이 허가되는지 아닌지 여부를 판단하는 데 사용되는 O/S 내 보안 에이전트(1219)를 조회하도록 구성될 수 있다. 다른 실시예에서 O/S 하위 보안 에이전트(1220)는 요청이 나왔던 가상 메모리(1204)의 메모리 페이지를 판단하고, 메모리 맵(1206)을 조회하여 그 메모리 페이지가 그 안에 매핑된 어떤 요소들과 관련되는지 여부를 판단하도록 구성될 수 있다. 또 다른 실시예에서 O/S 하위 보안 에이전트(1220)는 요청하는 요소의 메모리 페이지의 해시나 서명을 판단하고 그것을 알려진 개체들의 해시들 및 서명들과 비교하도록 구성될 수 있다.

[0244] O/S 하위 보안 에이전트(1220)가 마이크로코에 의해 전적으로나 부분적으로 구현되는 경우, O/S 하위 보안 에이전트(1220)는 쓰기를 시도했던 명령어의 어드레스를 판단하도록 구성될 수 있다. 일 실시예에서 O/S 하위 보안 에이전트(1220)는 물리적 메모리(1203) 안의 어디에서 그 명령어가 만들어졌는지를 판단하기 위해 명령어 포인터를 검사함으로써 그러한 판단을 내릴 수 있도록 구성될 수 있다. 다른 실시예에서, 메모리 맵(1206)을 액세스함으로써 O/S 하위 보안 에이전트(1220)는 해당 어드레스와 관련된 메모리 맵(1206)으로부터의 요소를 판단하도록 구성될 수 있다. 또 다른 실시예에서 O/S 하위 보안 에이전트(1220)는 요청하는 요소의 해시나 서명을 판

단하고 그것을 알려진 개체들의 해시들 및 서명들과 비교하도록 구성될 수 있다.

[0245] 메모리에 대해 시도된 액세스가 트래핑되었으면, O/S 하위 보안 에이전트(1220)는 보안 규칙들(1208)을 액세스하여 식별된 요청 개체에 기반하여 트래핑된 시도를 어떻게 처리할지를 판단하도록 구성될 수 있다. 보안 규칙들(1208)은 예컨대 운영체제(1213)의 어떤 특정된 커널 부분들만이 Fn1을 호출 및 실행할 수 있다는 것, 혹은 안전하다고 알려지거나 화이트리스트 상에 있는 개체들만이 SSDT의 허가에 따라 기입을 할 수 있다고 정의할 수 있다. O/S 하위 보안 에이전트(1220)는 그런 다음 요청이 진행되도록 허가하거나, 요청을 부정하거나, 응답 또는 기입 값을 스푸핑하거나, 교정 프로세스를 실행하는 것과 같은 어떤 적절한 액션을 수행하도록 구성될 수 있다.

[0246] 동작 시, O/S 하위 보안 에이전트(1220)는 운영체제(1213)와 같은 전자 장치(1201)의 운영체제들의 레벨 아래에서 실행될 수 있다. O/S 하위 보안 에이전트(1220)는 보안 규칙들(1208)을 조회하여 전자 장치(1201)의 어떤 메모리 자원들을 보호할지를 결정할 수 있다. O/S 하위 보안 에이전트(1220)는 메모리 맵(1206)의 콘텐츠를 결정, 전개, 및/또는 상주시킬 수 있다. 그렇게 하기 위해, O/S 하위 보안 에이전트(1220)는 보안 규칙들(1208), 보호 서버(202), 또는 메모리 맵(1206)에 정보를 위치시키기 위한 어떤 다른 적절한 정보 소스를 액세스할 수 있다. O/S 하위 보안 에이전트(1220)은 메모리 맵(1206) 안에 메모리의 소유권 및 콘텐츠를 매핑하기 위해 운영체제(1213), 애플리케이션(1210) 또는 드라이버(1211)와 같은 운영체제 레벨에 있는 개체들로부터 물리적 메모리(1203)나 가상 메모리(1204)에 대한 요청들을 인터셉트할 수 있다. O/S 하위 보안 에이전트(1220)는 메모리 맵(1206)이 상주될 수 있도록 어떤 개체들이 메모리 안에 로딩될지를 판단하기 위해 O/S 내 보안 에이전트(1219)를 액세스할 수 있다. 메모리 맵(1206)은 물리적 메모리(1203), 가상 메모리(1204) 에 대한 메모리 매핑 및/또는 그 둘 사이의 매핑들을 포함할 수 있다.

[0247] O/S 하위 보안 에이전트(1220)는 보안 규칙들(1208)을 조회하여 가상 메모리(1204) 및/또는 물리적 메모리(1203)의 어느 부분들을 보호할지를 결정할 수 있다. 보안 규칙들(1208)은 메모리의 어떤 부분들이 동적으로 보호되어야 한다는 것을 특정할 수 있으며, 여기서 그러한 메모리의 보호는 다양한 고려사항에 따라 O/S 하위 보안 에이전트(1220)에 의해 인에이블되거나 디세이블될 수 있다. 그러한 고려사항들은 예컨대 관리자 설정사항들, 악성 혹은 의심스러운 동향의 검출, 시간, 앞서 검출된 메모리 액세스, 혹은 어떤 다른 적절한 기준을 포함할 수 있다. 전자 장치(1201)의 메모리 보호가 계산 자원 면에서 비용이 많이 드는 경우, 그러한 동적 인에에블 및 디세이블이 O/S 하위 보안 에이전트(1220)가 전자 장치(1201)이 메모리의 중요 부분을 보다 잘 보호할 수 있게 하면서 다른 작업을 수행해야 할 전자 장치(1201)의 기능에 대한 부작용을 줄일 수 있다. 예를 들어 운영체제(1213)의 커널 코드의 콘텐츠를 포함하는 메모리가 O/S 하위 보안 에이전트(1220)에 의해 항상 보호될 수 있는 한편, 제3자 애플리케이션(1210)의 코드의 콘텐츠를 포함하는 메모리는 악성 소프트웨어가 존재하거나 제3자 애플리케이션(1210)에 영향을 미칠 수 있다는 기타 표시가 있을 때에만 보호될 수 있다.

[0248] O/S 하위 보안 에이전트(1220)는 물리적 메모리(1203) 및/또는 가상 메모리(1204)에 대해 시도된 액세스를 트래핑하기 위해 제어 구조 내 어떤 플래그를 세팅할 수 있다. 일 실시예에서 트래핑되도록 지정된 가상 메모리(1204) 내 메모리 페이지에 대해 운영체제(1213) 내 어떤 개체로부터 어떤 요청이 이뤄질 때, O/S 하위 보안 에이전트(1220)는 그 시도된 요청을 인터셉트할 수 있다. 다른 실시예에서, 가상 메모리(1204) 내 어느 메모리 페이지에 대해 어떤 요청이 이루어질 때, O/S 하위 보안 에이전트는 그 요청이 MMU(1214)에 의해 물리적 메모리(1203)의 어드레스에 대한 요청으로 변환될 수 있게 하며, 그에 따라 O/S 하위 보안 에이전트는 그 시도된 요청을 인터셉트할 수 있다. 또 다른 실시예에서, 운영체제(1213) 내 어느 개체로부터 물리적 메모리(1203) 내 어느 어드레스에 대한 요청이 이루어질 수 있을 때, O/S 하위 보안 에이전트(1220)는 그 시도된 요청을 인터셉트할 수 있다.

[0249] 요청이 인터셉트되었으면, O/S 하위 보안 에이전트(1220)는 그 인터셉트된 메모리에 대한 요청을 평가하기 위해 어떤 적절한 메커니즘을 사용할 수 있다. 해당 시도가 의심스러운지 여부를 판단하여 전자 장치(1201)의 지원들을 사용하고자 하는 악성 소프트웨어에 의한 악의적 시도를 가리키기 위해 보안 규칙들(1208)이 사용될 수 있다. 보안 규칙들(1208)은 예컨대 읽기, 쓰기, 또는 실행이 시도되었는지 여부; 어떤 개체가 해당 시도를 했는지; 액세스된 메모리 어드레스나 페이지; 동일한 요청자에 의한 이전의 시도들이나 액션들; 전자 장치(1201)의 사용자에 기반하여 더 많거나 더 적게 제한되는 규칙들과 같은 전자 장치(1201)의 관리자에 의한 보안 설정사항들; 또는 메모리 위치 및/또는 디지털 서명이나 해시 또는 관련 페이지들이나 메모리 어드레스들 상에서 결정된 것과 같은 요청자의 식별자에 대한 고려사항들을 포함할 수 있다.

[0250] 예를 들어 가상 메모리(1204)의 페이지 2 안의 페이지 디렉토리 데이터 구조나 물리적 메모리(1203)의 어드레스

(J)에서 시도된 쓰기가 O/S 하위 보안 에이전트(1220)에 의해 인터셉트될 수 있다. 해당 쓰기가 알려지지 않은 프로세스의 메모리 일부로부터 나왔다면, 그 쓰기는 O/S 하위 보안 에이전트(1220)에 의해 의심스러운 것으로 판단될 수 있다. 그러나, 시도된 쓰기가 운영체제(1213) 커널의 알려지고 검증된 부분으로부터 나왔으면 그 시도는 의심스러운 것이 아니라고 판단될 수 있다. 마찬가지로, 가상 메모리(1204)의 페이지 8이나 물리적 메모리(1203)의 어드레스(H)에서 시도된 Fn2의 실행이 인터셉트될 수 있다. 시도된 실행이 사용자 입력으로부터 이루어졌으면, 그 실행은 의심스러운 것이 아니라고 판단될 수 있다. 시도된 실행이 다른 프로그램의 메모리로부터 이루어졌고 그 프로그램이 승인된 리스트 상에 있는 것이 아니라면, 그 시도는 의심스럽거나 악성이라고 판단될 수 있다.

[0251] 다른 예에서, Fn1이 상호 운용성을 목적으로 보통 자신의 캐시를 다른 애플리케이션에게 노출시키는 웹 브라우저인 경우, O/S 하위 보안 에이전트(1220)는 Fn1의 메모리 페이지들 또는 메모리 어드레스들의 특정된 일부가 다른 애플리케이션들에 의해 판독되게 허용할 수 있다. 그러나 Fn1이 비밀을 유지해야 하는 메타데이터나 다른 정보를 포함하는 경우, O/S 하위 보안 에이전트(1220)는 F1의 메모리 페이지들 또는 메모리 어드레스들의 그러한 부분들이 Fn1 자신 이외의 다른 프로세스로부터 판독되는 것을 보호할 수 있다.

[0252] 프로그램이 의심스럽다거나 악성이라거나 그러지 않으면 악성 소프트웨어를 나타낸다고 판단되었으면, O/S 하위 보안 에이전트(1220)는 어떤 적절한 교정 액션을 취할 수 있다. O/S 하위 보안 에이전트(1220)는 예컨대 가상 메모리(1204)의 메모리 페이지 2나 물리적 메모리(1203)의 어드레스(J)에 대한 쓰기 요청을 거부하지만, 그 값이 기입되었음을 나타내는 결과를 돌려 보낼 수 있다. 그러한 요청을 생성하는 프로세스는 전자 장치(1201)의 자원들을 액세스하려는 추가 시도들에 대해 감시되거나, 중지되거나, 전자 장치(1201)로부터 청소될 수 있다. 다른 예에서, 가상 메모리(1204)의 페이지 8이나 물리적 메모리(1203)의 어드레스(H)에 대해 시도된 실행이 대신 하이퍼팟 프로세스나 클린업 프로세스의 실행으로 유도될 수 있다.

[0253] O/S 하위 보안 에이전트(1220)에 의해 보호되는 메모리의 콘텐츠는 악성 소프트웨어에 의해 공격될 수 있는 데이터, 코드, 또는 기타 유용한 시스템 자원들을 포함할 수 있다. O/S 하위 보안 에이전트(1220)는 예컨대 전자 장치(1201) 상에서 실행되는 프로세스들을 보여주는 메커니즘들을 읽거나 쓰거나 후킹하려고 시도하는 악성 소프트웨어에 대해 메모리의 콘텐츠를 보호하거나, 그것의 코드를 메모리에 로딩된 애플리케이션들의 부분들로 주입하거나, 가상 메모리(1204)를 위한 매핑 테이블들의 허가 및 액세스 플래그들을 변경할 수 있다. 운영체제(1213)의 레벨 아래에서 동작함으로써, O/S 하위 보안 에이전트(1220)는 운영체제(1213) 안에서 커널 모드 레벨로 실행되는 악성 소프트웨어를 피할 수 있다. O/S 하위 보안 에이전트(1220)는 제로 데이(zero-day) 검출을 수행할 수 있는데, 이는 어떤 경우 그것이 요청하는 개체의 아이디가 앞서 악성이라 판단되었다는 지식을 필요로 하지 않을 수 있기 때문이다(그 개체가 알려지지 않았다는 사실이 전자 장치(1201)의 메모리의 어떤 부분들에 대한 액세스를 거부하는데 사용될 수 있다). 운영체제(1213)나 운영체제(1213) 안에서 실행되는 안티바이러스 또는 악성 소프트웨어 조치들이 완전히 훼손되면, 메모리가 운영체제의 레벨에서 실행되는 개체들로부터 완전히 닫힐 수 있다.

[0254] O/S 하위 보안 에이전트(1220)의 한 애플리케이션은 특정 콘텐츠의 읽기, 쓰기 또는 실행이 시도되기도 전에 특정 메모리 페이지의 허가에 대한 변화를 검출하여 가상 메모리(1204)의 콘텐츠에 대해 시도되는 액세스를 검출해야 할 수 있다. MMU(1214)에 의해 사용되는 메모리 테이블들은 메모리, 가상 메모리(1204)의 페이지 자체, 및/또는 물리적 메모리(1203)의 어드레스에 존재할 수 있다. 메모리 테이블의 값들을 변경하려는 시도, 예컨대 프로세스의 코드 섹션의 허가 사항을 "읽기"에서 "쓰기"로 변경하려는 시도 자체가 O/S 하위 보안 에이전트(1220)에 의해 트래핑될 수 있다. 가상 메모리(1204)이 메모리 페이지 또는 물리적 메모리(1203)의 어드레스는 O/S 하위 보안 에이전트(1220)에 의해 보호될 수 있으며, 그러한 위치에서 허가 사항에 새로운 값을 기입하고자 하는 시도가 트래핑될 때 O/S 하위 보안 에이전트(1220)는 그 시도의 요청자가 그러한 변경을 수행하는 것이 허용되는지 여부를 판단할 수 있다. 예를 들어 프로세스의 코드 섹션의 허가 사항을 바꾸라는 요청이 다른 프로세스로부터 일어난 경우, 그 시도된 허가 사항의 변경은 거부될 수 있다.

[0255] 도 14는 전자 장치 액세스 시도에 대해 운영체제 하위 트래핑을 이용하여 메모리를 보호하는 방법(1400)의 실시예이다. 단계 1405에서 전가 기기의 가상 또는 물리적 메모리가 메모리의 콘텐츠 아이디나 소유자를 판단하기 위해 매핑될 수 있다. 메모리를 매핑하기 위해, 예컨대 보호 서버가 액세스될 수 있고; 메모리의 읽기, 쓰기 및 실행이 추적될 수 있고; 그리고/또한 메모리의 콘텐츠가 검색될 수 있고 그 콘텐츠에 대해 서명이 만들어질 수 있다.

[0256] 단계 1410에서 보안 규칙들이 액세스되어, 단계 1415에서 물리적 메모리의 어드레스들이나 가상 메모리의 페이

지들이 보호되도록 할 수 있다. 보호될 메모리는 예컨대 보안 규칙들, 전자 장치의 사용자, 악성 소프트웨어에 대한 표시와 같은 전자 장치에서 관찰된 다른 동향, 보호되는 메모리를 액세스하려는 이전의 시도들, 또는 관리자 설정사항들에 좌우될 수 있다. 보호될 메모리는 전자 장치의 동작 조건들이 변화하면서 동적으로 변화할 수 있다. 보안 규칙들은 보호될 전자 장치의 개체들을 특정할 수 있고, 그 개체들의 물리적 또는 가상 메모리 내 위치가 메모리 맵을 액세스함으로써 결정될 수 있다.

[0257] 단계 1420에서 보안 규칙들의 요건에 따라 메모리에 대해 시도되는 액세스를 트래핑하도록 제어 구조 안에서 플래그들이 세팅될 수 있다. 그러한 플래그들은 가상 메모리의 페이지들 및/또는 물리적 메모리의 어드레스들에 대해 세팅될 수 있다. 플래그들은 보호되어야 하는 메모리 및 플래깅되어야 하는 종류의 액세스 방식(가령, 읽기, 쓰기 또는 실행)에 대한 표시를 포함할 수 있다. 단계 1425에서, 보호되는 메모리에 대한 액세스는 지정된 타입의 액세스 시도가 지정된 어드레스나 페이지에 대해 이루어졌는지 보기 위해 감시될 수 있다. 단계 1430에서 메모리에 대한 액세스 시도가 트래핑되었는지 여부가 판단될 수 있다. 그런 경우가 아니면, 단계 1435에서 보호될 메모리의 플래그들이 변경을 요하는지 여부가 판단될 수 있다. 그런 경우, 방법(1400)은 메모리에 대한 액세스를 보호하기 위한 플래들을 업데이트하기 위해 보안 규칙들을 액세스하도록 단계 1410으로 돌아갈 수 있다. 그렇지 않은 경우 방법(1400)은 보호되는 메모리에 대해 시도된 액세스를 모니터하기 위해 단계 1425로 돌아갈 수 있다.

[0258] 메모리를 액세스하려는 시도가 트래핑되었으면, 단계 1440을 시작하여 트래핑된 시도가 평가될 수 있다. 그러한 시도를 평가하기 위해 메모리 맵이 조회되어 그 요청이 이루어졌던 곳을 판단하고 요청자를 식별하도록 할 수 있다. 그들의 콘텐츠에 대해 기입될 데이터의 값들이 결정되고 평가될 수 있다. 시도(읽기, 쓰기 또는 실행)의 성격이 고려될 수 있다. 이러한 고려사항들의 예들은 단계 1445에서, 시도된 액세스가 악성 소프트웨어를 나타내는지 아닌지 여부를 판단하기 위해 보안 규칙들과 연계하여 이용될 수 있다. 시도된 액세스가 악성 소프트웨어를 가리키는 경우, 단계 1450에서 교정 액션이 수행될 수 있다. 그러한 교정 액션은 요청된 액세스 거부, 스푸핑된 값의 리턴, 또는 하니팟 또는 교정 프로세스 개시를 포함할 수 있다. 시도된 액세스가 악성 소프트웨어를 가리키지 않는 경우, 단계 1455에서 그 요청이 허용될 수 있다. 방법(1400)은 필요 시 전자 장치의 메모리를 계속 보호하기 위해 단계 1425로 돌아갈 수 있다.

[0259] 도 15는 전자 장치(1504)의 운영체제 커널을 보호하는 시스템(1512)의 실시예이다. 시스템(1500)은 운영체제(1512)의 구성요소들(가령, 함수, 데이터, 및/또는 기타 구성요소들) 및 운영체제와 관련된 안전(trusted) 드라이버들을 액세스하려는 악성 소프트웨어의 시도들에 대해 보호하기 위해 전자 장치 상에서 동작하도록 구성된 O/S 하위 보안 에이전트(1516)을 포함할 수 있다. 또한 O/S 하위 보안 에이전트(1516)는 시도된 어떤 동작들을 트래핑하고 그렇게 트래핑된 동작에 어떻게 응답할지를 결정하기 위해 하나 이상의 보안 규칙들(1522)을 이용하도록 구성될 수 있다. O/S 하위 보안 에이전트(1516)는 트래핑된 동작에 대해 허용하거나 거부하거나 그에 대해 다른 교정 액션을 수행하도록 구성될 수 있다.

[0260] 도 15에 도시된 바와 같이, 전자 장치(1504)는 메모리(1508)에 연결된 프로세서(1506), 하나 이상의 애플리케이션들(1510), 하나 이상의 드라이버들(1511), 운영체제(1512), O/S 하위 보안 에이전트(1516) 및 보안 규칙들(1522)을 포함할 수 있다. 전자 장치(1504)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701), 도 9의 전자 장치(901), 도 12의 전자 장치(1201) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 프로세서(1506)는 도 2의 프로세서(208), 도 4의 프로세서(408), 도 7의 프로세서(702), 도 9의 프로세서(902), 도 12의 프로세서(1202) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(1508)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703), 도 9의 메모리(903), 도 12의 물리적 메모리(1203) 또는 가상 메모리 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 애플리케이션(1510)은 도 1의 애플리케이션(110), 도 2의 애플리케이션(210), 도 4의 애플리케이션(410), 도 7의 애플리케이션(709), 도 9의 애플리케이션(910), 도 12의 애플리케이션(1210) 및/또는 이들의 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 드라이버(1511)은 도 1의 드라이버(111), 도 2의 드라이버(211), 도 4의 드라이버(411), 도 7의 드라이버(711), 도 9의 드라이버(911), 도 12의 드라이버(1211) 및/또는 이들의 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 운영체제(1512)는 도 1의 운영체제(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 9의 운영체제(913), 도 12의 운영체제(1213) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 하위 보안

에이전트(1516)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM 보안 에이전트(216) 또는 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442), PC 펌웨어 보안 에이전트(444) 또는 O/S 하위 에이전트(450), 도 5의 펌웨어 보안 에이전트(516), 도 7의 마이크로코드 보안 에이전트(708), 도 9의 O/S 하위 에이전트(920), 도 12의 O/S 하위 보안 에이전트(1220) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0261] 도 15에 도시된 바와 같이, O/S 하위 보안 에이전트(1516)는 보안 규칙들(1522)을 포함할 수 있다. 보안 규칙들(1522)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(222), 도 4의 보안 규칙들(434, 436, 438), 도 5의 보안 규칙들(518), 도 7의 보안 규칙들(707, 723), 도 9의 보안 규칙들(908), 도 12의 보안 규칙들(1208) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보안 규칙들(1522)은 어떤 적절한 방식(가령, 전자 장치(1504)의 사용자에 의해 설정된 정책들, 전자 장치(1504)를 포함하는 회사의 관리자에 의해 설정된 정책들, O/S 하위 보안 에이전트(1516)의 생성자에 의해 설정된 정책들 등)에 따라 설정될 수 있다. 일부 실시예들에서 O/S 하위 보안 에이전트(1516)는 (가령 악성 소프트웨어 정의들에 대한 업데이트 때문에) 네트워크(244)를 통해 보호 서버(202)로부터 보안 규칙들(1522)에 대한 업데이트들이나 수정안들을 요청 및/또는 수신할 수 있다.

[0262] 도 15에 도시된 것과 같이 보안 규칙들(1522)은 액세스 맵(1562) 및 정책들(1564)을 포함할 수 있다. 액세스 맵(1562)은 하나 이상의 안전한 액세스들에 관한 정황 정보와 함께, 운영체제(1512) 및 드라이버(1511)의 다양한 개별 구성요소들(가령, 함수, 데이터 및/또는 다른 구성요소들)에 대한 하나 이상의 안전한 액세스들을 제시한 로그, 리스트, 맵, 또는 다른 데이터 구조를 포함할 수 있다. 도 16은 액세스 맵(1562)의 실시예이다. 소정 실시예들에서 액세스 맵(1562)은 다른 전자 장치(가령, 도 18의 전자 장치(1800)) 상의 실질적으로 악성 소프트웨어가 없는 운영체제(가령, 도 18의 운영체제(1812)) 및 실질적으로 악성 소프트웨어가 없는 그것의 안전한 드라이버들(가령, 안전한 드라이버들(1811))을 시뮬레이션함으로써 생성될 수 있다. 그러한 실시예들에 따른 액세스 맵(1562)의 생성이 도 18 및 19와 관련하여 이하에서 보다 상세히 기술된다. 도 16에 도시된 바와 같이 액세스 맵(1562)은 하나 이상의 함수 액세스 서브맵들(1602), 하나 이상의 데이터 액세스 서브맵들(1604) 및/또는 하나 이상의 스택 액세스 맵들(1606)을 포함할 수 있다.

[0263] 함수 액세스 서브맵(1602)은 운영체제(1512)나 안전한 드라이버(1511)의 특정 함수에 대해, 다른 안전한 함수들에 의한 특정 함수로의 안전한 액세스들을 정의할 수 있다. 함수 액세스 서브맵(1602)은 또한 어떤 함수에 대한 안전한 액세스와 관련된 정황 정보를 포함할 수 있으며, 상기 정황 정보는 어떤 실시예들에서 특정 드라이버 안에서 안전한 액세스의 호출 함수가 위치하는 코드 섹션들(가령, 메모리 위치를 통해 식별됨)을 포함한다.

[0264] 함수 액세스 서브맵(1604)은 운영체제(1512)나 드라이버(1511)의 데이터의 특정 항목에 대해, 안전한 함수들에 의한 특정 데이터로의 안전한 액세스들을 정의할 수 있다. 데이터 액세스 서브맵(1604)은 또한 데이터의 어떤 아이템에 대한 안전한 액세스와 관련된 정황 정보를 포함할 수 있으며, 상기 정황 정보는 일부 실시예들에서, 안전한 함수와 관련된 특정 메모리 위치, 특정 드라이버 안에 그 안전한 함수가 위치되는 코드 섹션들 및/또는 안전한 액세스가 읽기 액세스인지 쓰기 액세스인지 여부를 포함한다.

[0265] 스택 액세스 서브맵(1606)은 여러 함수들 사이에서 허용된 호출 관계들을 기술한 함수 스택을 정의할 수 있다. 스택 액세스 서브맵(1606)에서 스택 내 각각의 특정 함수는 그 함수 스택 내에서 그것 밑에 나타나는 함수를 액세스하는 것이 안전하다. 스택 액세스 서브맵(1606)은 함수 액세스 서브맵(1602)과 유사한 정황 정보를 포함할 수 있다. 스택 액세스 서브맵(1606)은 예컨대 특정 함수 F2가 함수 F3를 호출할 수 있고, 함수 F3가 함수 F4를 호출할 수 있으며, F3를 호출하는 F4와 F2를 호출하는 F3는 안전한 함수 호출 경로가 아니라는 것을 보여줄 수 있다.

[0266] 액세스 맵(1562)에서 제시된 다양한 함수들, 데이터, 코드 섹션들, 드라이버들, 및 기타 개체들의 아이디(아이디)들이 메모리 내 특정 함수, 데이터, 코드 섹션, 드라이버, 또는 개체가 저장되는 메모리 위치(가령, 물리적 메모리 어드레스 또는 가상 메모리 어드레스)에 의해 정의될 수 있다. 도 17은 도 16의 액세스 맵(1562)의 예에서 정의된 함수들 및 데이터 사이의 상호관계들을 추가 예시하는 가상 메모리(1700)의 일 실시예이다. 도 17에 도시된 바와 같이, 메모리(1700)는 메모리 어드레스들 1701, 1706, 1710 및 1714에 각기 위치하는 드라이버들 Y1, Y2, Y3 및 Y4를 포함할 수 있다. 드라이버 Y1은 어드레스 1702에 있는 코드 섹션 X1 내에서 어드레스 1703에 있는 함수 F1을 포함할 수 있다. 드라이버 Y1은 어드레스 1704에 있는 데이터 섹션 D1 내에서 데이터 포인터(1705)를 포함할 수도 있다. 드라이버 Y2는 어드레스 1707에 있는 코드 섹션 X2 내에서 어드레스 1708에 있는 함수 F2를 포함할 수 있다. 드라이버 Y3는 어드레스 1711에 있는 코드 섹션 X3 내에서 어드레스 1712에

있는 함수 F3를 포함할 수 있다. 드라이버 Y4는 어드레스 1715에 있는 코드 섹션 X4 내에서 어드레스 1716에 있는 함수 F4를 포함할 수 있다. 함수 F2의 메모리 어드레스 Z2는 메모리 위치 1709에 존재할 수 있다. 함수 F3의 메모리 어드레스 Z3는 메모리 위치 1713에 존재할 수 있다. 함수 F4의 메모리 어드레스 Z4는 메모리 위치 1717에 존재할 수 있다. 도 17의 여러 화살표들은 도 16의 액세스 맵(1562)에 제시된 함수들과 데이터 세트 사이에서의 안전한 액세스들을 묘사한다. 예를 들어 함수 액세스 서브맵(1602)의 안전한 액세스들은 어드레스 1708에 있는 함수 F2와 어드레스 1703에 있는 함수 F1 사이의 안전한 실행 호출을 나타내는 화살표, 어드레스 1712에 있는 함수 F3와 어드레스 1703에 있는 함수 F1 사이의 안전한 실행 호출을 나타내는 화살표, 어드레스 1716에 있는 함수 F4와 어드레스 1703에 있는 함수 F1 사이의 안전한 실행 호출을 나타내는 화살표에 의해 묘사된다.

[0267] 도 15로 돌아가면, 정책들(1564)은 O/S 하위 보안 에이전트(1516)에 의해 트래핑될 이벤트를 정의하고/하거나 트래핑된 이벤트들을 처리하기 위해 적용될 정책들을 제시하는 로그, 리스트, 또는 다른 데이터 구조를 포함할 수 있다. 특정 실시예에서, 어느 정책은 운영체제(1512)나 안전한 드라이버들(1511)의 구성요소를 저장하는 메모리의 일부에 대해 시도된 액세스(가령, 읽기, 쓰기, 실행, 함수 호출)에 반응하여, O/S 하위 보안 에이전트(1516)가 액세스 맵(1562) 안의 어떤 엔트리가 그러한 드라이버 함수가 그러한 구성요소에 액세스하였다고 나타내는 경우(어떤 경우, 드라이버 함수가 액세스 맵(1562)에 정의된 대로 드라이버의 특정 코드 섹션 안에 존재한다는 표시를 포함) 그 시도된 액세스를 허용할 수 있다고 지시할 수 있다. 동일하거나 대안적인 실시예에서, 어느 정책은 운영체제(1512)나 안전한 드라이버들(1511)의 구성요소를 저장하는 메모리의 일부에 대해 시도된 액세스(가령, 읽기, 쓰기, 실행, 함수 호출)에 반응하여, O/S 하위 보안 에이전트(1516)가 액세스 맵(1562) 안의 어떤 엔트리도 그러한 드라이버 함수가 그러한 구성요소에 액세스하였다고 나타내지 않는 경우(어떤 경우, 드라이버 함수가 액세스 맵(1562)에 정의된 대로 드라이버의 특정 코드 섹션 안에 존재한다는 표시를 포함) 그 시도된 액세스를 거부할 수 있다고 지시할 수 있다. 이들 및 다른 실시예들에서, 어느 정책은 알려지지 않은 드라이버 함수에 의해 운영체제(1512)나 안전한 드라이버들(1511)의 구성요소들에 대해 시도된 액세스들에 대해 시도된 어떤 액세스들은 허용되고 시도된 다른 액세스들은 거부될 수 있고/있거나 그러한 액세스에 관한 정보가 추후 분석을 위한 법의학적 증거로서 보호 서버(202)로 전송될 수 있다고 지시할 수 있다.

[0268] 동작 시, O/S 하위 보안 에이전트(1516)는 이 개시에서 제시된 어떤 트래핑 기법에 따라 운영체제(1512) 및 드라이버들(1511)의 구성요소들에 대해 시도되는 액세스들에 대해 트래핑할 수 있다. 어떤 실시예들에서 O/S 하위 보안 에이전트(1516)는 보안 규칙들(1522)에 따라 이벤트들에 대해 트래핑할 수 있다. 운영체제(1512) 및 드라이버들(1511)의 구성요소에 대해 시도된 액세스를 트래핑함에 따라, O/S 하위 보안 에이전트(1516)는 그 시도된 액세스와 관련된 정황 정보를 액세스 맵(1562)와 비교하여 시도된 액세스가 안전한지를 판단하도록 할 수 있다. 시도된 액세스가 안전하면(가령, 시도된 액세스가 액세스 맵(1562) 안에서 해당 엔트리를 가지는 경우) O/S 하위 보안 에이전트(1516)는 그 액세스를 허용할 수 있다. 시도된 액세스가 안전하지 않으면(가령, 시도된 액세스가 액세스 맵(1562) 안에서 해당 엔트리를 가지지 못하는 경우) O/S 하위 보안 에이전트(1516)는 교정 액션을 개시할 수 있다. 교정 액션은 시도된 액세스의 거부, 해당 액세스를 허용할지 거부할지 여부를 판단하기 위한 정책들(1564) 조회 및/또는 추후 처리를 위해 보호 서버(202)로 그 액세스에 관한 법의학적 데이터(가령, 정황 증거)를 보고하는 것을 포함할 수 있다. 그에 따라 O/S 하위 보안 에이전트(1516)는 액세스 맵(1562) 및 정책들(1564)과 함께, 운영체제(1512) 및 드라이버들(1511)의 구성요소들에 대한 악의적 공격들을 보호할 수 있다.

[0269] 어떤 실시예들에서 정책들(1564)은 안전하지 않은 액세스 시도에 응하여 O/S 하위 보안 에이전트(1516)에 의해 개시되는 교정 액션이 해당 액세스 시도가 잠정적으로 악의적이지 않은 개체에 의한 것인지 잠재적으로 악의적인 개체에 의한 것인지 여부에 따라 좌우될 수 있다고 지시할 수 있다. 잠정적으로 악의적인 개체는 안전하지 않은(가령, 액세스 맵(1562)에 해당 엔트리를 갖지 않는) 액세스 시도를 개시하고, 악성 소프트웨어(가령, 전자 장치(1504)의 민감한 자원들을 액세스하고자 시도하거나, 운영체제(1512)에 의해 제공되는 함수 라우팅을 이용하지 않고 서브함수를 액세스하고자 시도 하는 등)의 잠정적 존재를 가리키는 동향을 보이는 O/S 하위 보안 에이전트(1516)에게 알려지지 않은(화이트리스트나 블랙리스트 중 어느 하나에 나타나지 않는) 애플리케이션, 드라이버 또는 다른 개체일 수 있다. 잠정적으로 악의적이지 않은 개체는 다른 경우 잠정적으로 악의적인 개체라고 발견되지 않는 어떤 개체일 수 있다. 잠정적으로 악의적이지 않은 개체의 경우, 정책들(1564)은 시도된 어떤 액세스들을 허용하면서 다른 것들은 거부할 수 있다. 예를 들어 잠정적으로 악의적이지 않은 개체들에 대해, 네트워크 호출들 및 파일 시스템 호출들은 허용될 수 있고, 반면 내부적 네트워크 발송 루틴 포인터를 변경하거나, 내부적 네트워크 드라이버 인터페이스 사양(NDIS) 포인터들을 변경하거나 커널 코드 섹션, 데이터 섹션 또는 시스템 서비스 발송 테이블(SSDT)로 쓰고자 하는 액세스 시도들은 거부될 수 있다. 한편, 잠정적으로

악의적인 개체들에 대해서는 모든 액세스 시도들이 거부될 수 있다.

[0270] 다른 실시예들에서 정책들(1564)은 알려지지 않은 개체들(가령, 화이트리스트나 블랙리스트에 나타나지 않는 개체들)이 어떤 액세스 시도를 제한적으로 한번 실행하는 것이 허용될 수 있고, 그 이후 해당 액세스에 대한 정보가 보호 서버(202)로 전송되어 어떤 추후 교정 액션을 결정하기 위해 더 평가될 수 있다고 지시할 수 있다.

[0271] 도 18은 액세스 맵(1562)을 생성하기 위한 시스템(1800)의 실시예이다. 시스템(1800)은 전자 장치(1804) 상에서 운영체제(1812) 및 안전한 드라이버들(1811)의 관찰된 동향들에 기반하여 액세스 맵(1562) 안의 개체들을 생성하도록 동작하게 구성된 O/S 하위 보안 에이전트(1816)를 포함할 수 있다. 도 18에 도시된 바와 같이 전자 장치(1804)는 프로세서(1806), 메모리(1808), 드라이버들(1811), 운영체제(1812), 및 O/S 하위 보안 에이전트(1816)를 포함할 수 있다. 전자 장치(1804)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701), 도 9의 전자 장치(901), 도 12의 전자 장치(1201) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0272] 프로세서(1806)는 예컨대 마이크로프로세서, 마이크로 컨트롤러, DSP(digital signal processor), ASIC(application specific integrated circuit), 또는 프로그램 명령어들 및/또는 프로세스 데이터를 해석 및/또는 실행하도록 구성된 어떤 다른 디지털 혹은 아날로그 회로를 포함할 수 있다. 일부 실시예들에서 프로세서(1806)는 메모리(1808)에 저장된 프로그램 명령어들 및/또는 프로세스 데이터를 해석 및/또는 실행할 수 있다. 메모리(1808)는 일부 혹은 전체가 애플리케이션 메모리, 시스템 메모리, 또는 둘 모두로서 구성될 수 있다. 메모리(1808)는 하나 이상의 메모리 모듈들을 보유 및/또는 하우징하도록 구성된 어떤 시스템, 기기, 또는 장치를 포함할 수 있다; 예를 들어 메모리(1808)는 ROM(read-only memory), RAM(random access memory), 고체 상태 메모리, 또는 디스크 기반 메모리를 포함할 수 있다. 각각의 메모리 모듈은 일정 시간 동안 프로그램 명령어들 및/또는 데이터를 보유하도록 구성된 어떤 시스템, 기기 또는 장치(가령, 컴퓨터 판독 가능 비 임시 매체)를 포함할 수 있다.

[0273] O/S 하위 보안 에이전트(1816)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM 보안 에이전트(216) 또는 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442), PC 펌웨어 보안 에이전트(444) 또는 O/S 하위 에이전트(450), 도 5의 펌웨어 보안 에이전트(516), 도 7의 마이크로코드 보안 에이전트(708), 도 9의 O/S 하위 에이전트(920), 도 12의 O/S 하위 보안 에이전트(1220), 도 15의 O/S 하위 보안 에이전트(1516) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0274] 운영체제(1812)는 도 1의 운영체제(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 9의 운영체제(913), 도 12의 운영체제(1213), 도 15의 운영체제(1512) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 안전한 드라이버들(1811)은 도 1의 드라이버(111), 도 2의 드라이버(211), 도 4의 드라이버(411), 도 7의 드라이버(711), 도 9의 드라이버(911), 도 12의 드라이버(1211), 도 15의 드라이버 및/또는 이들의 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 그러나 전자 장치(1804)에서의 사용과 관련하여, 운영체제(1812)는 악성 소프트웨어의 염려가 없을 수 있고 안전한 드라이버들(1811)는 악의적이지 않고 악성 소프트웨어 염려가 없다고 알려진 드라이버들만을 포함할 수 있다. 예를 들어 운영체제(1812) 및 안전한 드라이버들(1811)이 어떤 악성 개체들도 포함하지 않도록 확실히 하기 위해 전자 장치(1804)에 대해 주의를 기울일 수 있다. 특정 예로서, 운영체제(1812) 및 안전한 드라이버들(1811)은 전자 장치(1804)의 비어있거나 새롭게-포맷된 컴퓨터- 판독 가능 매체 상으로 설치될 수 있고, O/S 하위 보안 에이전트(1816)가 아닌 어떤 다른 개체들도 전자 장치(1804) 상에 설치되지 않도록 주의가 필요할 수 있다.

[0275] 동작 시, O/S 하위 보안 에이전트(1816)는 이 개시에서 제시된 어떤 트래핑 기법에 따라 운영체제(1812) 및 안전한 드라이버들(1811)의 구성요소들에 대해 시도되는 액세스들에 대해 트래핑할 수 있다. 시스템(1812) 및 안전한 드라이버들(1811)의 구성요소에 대한 액세스를 트래핑함에 따라, O/S 하위 보안 에이전트(1816)는 그 액세스와 관련된 정황 정보를 판단하고 그 액세스 및 정황 정보를 (가령 함수 액세스 서브맵(1602), 데이터 액세스 서브맵(1604), 함수 스택 액세스 서브맵(1606), 또는 다른 적절한 방식의 일부로서) 저장할 수 있다. 그에 따라, 액세스 맵(1562)의 엔트리들을 생성하기 위해 O/S 하위 보안 에이전트(1816)에 의해 실질적으로 악성 소프트웨어 염려가 없는 전자 장치(1804)의 실행 및 운영체제(1812) 및 그 안전한 드라이버들(1811) 사이의 안전한 종속성들이 관찰될 수 있으며, 각각의 엔트리는 운영체제(1812)나 안전한 드라이버(1811)의 구성요소에 대한 안전한 액세스를 정의한다. 액세스 맵(1562)의 엔트리들이 실질적으로 악성 소프트웨어 염려가 없다고 알려진 개체들의 시뮬레이션된 실행에 기반하여 생성되기 때문에, 액세스 맵(1562)은 추가 개체들 없이 운영체제(1812)

및 그 안전한 드라이버들(1811)의 표준 기대 동향에 대한 표현을 포함할 수 있다. 그에 따라 액세스 맵(1562)은 운영체제(1812) 및 그 안전한 드라이버들(1811)의 구성요소들에 대해 합법적이고 비악의적 액세스들을 가진 엔트리들만을 포함할 수 있다.

[0276] 결과적으로, 전자 장치(1804)의 O/S 하위 보안 에이전트(1816)가 액세스 맵(1562)을 생성하면, 액세스 맵(1562)은 O/S 하위 보안 에이전트(1516)에서 이용가능하게 될 수 있으며(가령, 액세스 맵을 전자 장치(1504)에 다운로드하거나, 컴퓨터 판독 가능 저장 메체를 통해 전자 장치(1504)로 전송하는 등에 의해), 여기서 O/S 하위 보안 에이전트(1516)는 상술한 바와 같이, 시도된 액세스들 중 어느 것이 안전하거나 불안전한지를 판단하고 그 판단에 기반하여 추후 액션을 취하기 위해 운영체제(1512) 및/또는 드라이버들(1511)의 구성요소들에 대해 시도되는 액세스들을 트래핑할 수 있다. 결과적으로, O/S 하위 보안 에이전트는 악의적 액세스들로부터 운영체제(1512) 및 안전한 드라이버들(1511)을 보호할 수 있다.

[0277] 도 19는 전자 장치의 운영체제 커널을 보호하는 방법(1900)의 실시예이다. 방법(1900)에서, 실질적으로 악성 소프트웨어 염려가 없는 운영체제 및 관련된 안전한 드라이버들이 설치되어 있는 제1전자 장치 상에서 실행되는 제1O/S 하위 보안 에이전트가 액세스 맵을 생성하기 위해 사용될 수 있다(가령 단계들 1905-1910 참조). 또한 제2전자 장치 상에서 실행되는 제2O/S 하위 보안 에이전트는 액세스 맵을 참조하여, 제2전자 장치에 설치된 제2 운영체제 및 그 관련 드라이버들의 구성요소들을 보호할 수 있다(가령 단계들 1915-1930 참조).

[0278] 단계 1905에서 실질적으로 악성 소프트웨어 염려가 없는 운영체제 및 관련된 안전한 드라이버들이 설치되어 있는 제1전자 장치 상에서 실행되는 제1O/S 하위 보안 에이전트는 운영체제 및/또는 안전한 드라이버들의 구성요소들(가령, 함수들 및 데이터)에 대한 액세스들을 트래핑할 수 있다. 단계 1910에서, 제1O/S 하위 보안 에이전트는 액세스들과 관련된 정황 정보를 포함하는 액세스들에 관한 정보를 액세스 맵에 기록할 수 있다. 다른 호출 함수에 의한 함수로의 액세스와 관련하여, 그러한 정황 정보는 안전한 액세스의 호출 함수가 특정 드라이버 안에 위치되는 코드 섹션들(가령, 메모리 위치에 의해 식별된 것과 같이) 포함할 수 있다. 호출 함수에 의한 데이터의 한 항목에 대한 액세스와 관련하여, 그러한 정황 정보는 안전한 함수와 관련된 특정 메모리 레지스터, 안전한 함수가 특정 드라이버 안에 위치하는 코드 섹션들 및/또는 안전한 액세스가 읽기 액세스인지 쓰기 액세스인지 여부를 포함할 수 있다.

[0279] 단계 1915에서, 제2전자 장치 상에서 실행되는 제2O/S 하위 보안 에이전트는 제2전자 장치 상에서 실행되는 운영체제 및/또는 드라이버들의 구성요소들에 대해 시도되는 액세스들을 트래핑할 수 있다. 단계 1920에서, 어떤 구성요소에 대해 시도된 액세스를 트래핑함에 따라, 제2O/S 하위 보안 에이전트는 상기 시도된 액세스와 관련된 정황 정보를 액세스 맵과 비교하여, 상기 시도된 액세스가 안전한지를 판단할 수 있다. 시도된 액세스가 액세스 맵 안에 해당하는 엔트리를 가지는 경우, 그 시도된 액세스는 안전할 수 있다. 시도된 액세스가 안전하면, 방법(1900)은 단계(1925)로 진행할 수 있다. 시도된 액세스가 안전하지 않으면, 방법(1900)은 단계(1930)로 진행할 수 있다.

[0280] 단계 1925에서, 시도된 액세스가 안전하다는 판단에 따라, 제2O/S 하위 보안 에이전트는 그 시도된 액세스를 허용할 수 있다. 단계 1925의 완료 후, 방법(1900)은 다시 단계 1915로 진행할 수 있다.

[0281] 단계 1930에서, 시도된 액세스가 안전하지 않다는 판단에 따라, 제2O/S 하위 보안 에이전트는 교정 액션을 개시할 수 있다. 교정 액션은 시도된 액세스의 거부, 해당 액세스를 허용할지 거부할지 여부를 판단하기 위한 정책들 조회 및/또는 추후 처리를 위해 보호 서버로 그 액세스에 관한 법의학적 데이터(가령, 정황 증거)를 보고하는 것을 포함할 수 있다. 단계 1925의 완료 후, 방법(1900)은 다시 단계 1915로 진행할 수 있다.

[0282] 도 20은 전자 장치(2001)를 악성 소프트웨어로부터 보호하도록 구성된 것으로, 운영체제를 안전하게 실행하기 위한 운영체제 실행 환경을 제공하는 시스템(2000)의 일 실시예이다. 도 20의 구성요소들은 도 21 및 도 22에서 그들과 공통적으로 불리는 대응요소들과 동일할 수 있다. 시스템(2000)은 운영체제 실행 환경(2008)("OSEE")의 안전한 시동을 제공하도록 구성된 런칭 모듈(2020)을 포함할 수 있다. 런칭 모듈(2020)은 운영체제("OS") 하위 보안 에이전트(2004), 운영체제(2012), O/S 내 보안 에이전트(2016)와 같은 OSEE(2008)의 구성요소들이 시동 전 악성 소프트웨어에 의해 제약받지 않도록 보장함으로써 OSEE(2008)의 안전한 시동을 제공하도록 구성될 수 있다. 런칭 모듈(2020)이 OSEE(2008)의 안전한 시동을 성공적으로 제공한 후, O/S 하위 보안 에이전트(2004) 및 O/S 내 보안 에이전트(2016)와 같은 OSEE(2008)의 구성요소들은 악성 소프트웨어가 런칭 모듈(2020)과 같은 전자 장치(2001)의 구성요소들을 감염시키는 것을 막도록 협력할 수 있다.

[0283] 전자 장치(2001)는 OSEE(2008)의 안전한 시동을 제공하도록 구성된 런칭 모듈(2020)을 포함할 수 있다.

OSEE(2008)는 하나 이상의 운영체제들(2012)을 실행하기 위한 안전한 환경을 제공하기 위해 O/S 하위 보안 에이전트(2004) 및 O/S 내 보안 에이전트(2016)를 포함할 수 있다. 전자 장치(2001)는 또한 하나 이상의 운영체제들(2012)을 실행하기 위한 안전한 환경을 제공하는 것을 돕도록 하는 보호 서버(2022)에 통신 가능하게 연결될 수 있다. 보호 서버(2022)는 백업 스토리지 장치(2024)를 포함할 수 있다. 전자 장치(2001)는 도 1의 전자 장치(103), 도 2의 전자 장치(104), 도 4의 전자 장치(404), 도 7의 전자 장치(701) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(2001)는 하나 이상의 프로세서들(2002), 메모리(2003), 또는 스토리지 장치(2006)와 같은 자원들(2026)을 포함할 수 있다. 프로세서(2002)는 도 2의 프로세서(208), 도 4의 프로세서(406), 도 7의 프로세서(702) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(2003)는 도 2의 메모리(207), 도 4의 메모리(408), 도 7의 메모리(703) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 운영체제(2012)는 도 1의 운영체제(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(2016)의 실시예들에 대한 설명은 도 22의 O/S 내 보안 에이전트(2206)에 대한 논의들에서 찾아볼 수 있다. O/S 하위 보안 에이전트(2004)의 실시예들에 대한 설명은 도 22의 O/S 하위 보안 에이전트(2208)에 대한 논의들에서 찾아볼 수 있다.

[0284] 스토리지 장치(2006)는 도 1의 자원(106), 도 2의 시스템 자원들(214), 도 4의 스토리지(426), 도 5의 I/O 장치(502) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 스토리지 장치(2006)는 데이터나 다른 정보를 저장하기 위한 어떤 적절한 자원을 포함할 수 있다. 예를 들어 스토리지 장치(2006)는 제한 없이, 직접 액세스 스토리지 장치(가령, 하드 디스크 드라이브나 플로피 디스크), 순차적 액세스 스토리지 장치(가령, 테이프 디스크 드라이브), 컴팩트 디스크, CD-ROM, DVD, 랜덤 액세스 메모리(RAM) 및/또는 플래시 메모리(가령, 플래시 기반 고체 상태 드라이브)를 포함할 수 있다. 스토리지 장치(2006)는 각각이 고정된 데이터 양을 저장할 수 있는 하나 이상의 섹터들로 분할될 수 있다. 예를 들어 스토리지 장치(2006)는 각각 512 바이트의 섹터들로 분할될 수 있지만 어떤 다른 적절한 섹터 사이즈가 사용될 수도 있다. 다양한 실시예들에서, 스토리지 장치(2006)는 보호 서버(2022) 상에서와 같이 전자 장치(2001)로부터 먼 곳에 위치될 수 있다. 다른 실시예들에서, 스토리지 장치(2006)는 전자 장치(2001)의 로컬 자원(2026)일 수 있다.

[0285] 백업 스토리지 장치(2024)는 데이터나 다른 정보를 저장하기 위한 어떤 적절한 자원을 포함할 수 있다. 예를 들어 백업 스토리지 장치(2024)는 스토리지 장치(2006)의 기능에 의해 구현되거나 그러한 기능을 구현하도록 구성될 수 있다. 백업 스토리지 장치(2024)는 스토리지 장치(2006)와 같이 전자 장치(2001)의 로컬 스토리지 장치에 의해 구현될 수 있다. 다른 실시예들에서 백업 스토리지 장치(2024)는 보호 서버(2022) 상에서와 같이 네트워크 상에 위치하는 원격 스토리지 장치에 의해 구현될 수 있다. 백업 스토리지 장치(2024)가 네트워크 상에 위치하는 경우, O/S 하위 보안 에이전트(2004)는 백업 스토리지 장치(2024)를 액세스하기 위해 네트워크 접속을 이용할 수 있다. 네트워크 접속은 악성 소프트웨어에 감염될 수 있는 운영체제 커널의 네트워크 장치 드라이버들을 이용하는 것을 피하기 위해 운영체제(2012) 아래의 우선순위 레벨로 구현될 수 있다. 네트워크 접속은 전자 장치(2001)의 네트워크 카드를 직접 액세스함으로써, HTTPS, iSCSI, NFS, 또는 CIFS 클라이언트의 사용이 백업 스토리지 장치(2024)를 액세스하게 할 수 있는 액티브 관리 기술(AMT)을 사용하여 구현될 수 있다. 그러한 실시예들에서는, 네트워크 접속이 백업 스토리지 장치(2024)를 액세스해야 하지만, 백업 스토리지 장치(2024)가 전자 장치(2001)의 운영체제(2012) 상에서 실행되는 어떤 악성 소프트웨어로부터 분리될 수 있다.

[0286] 보호 서버(2022)는 전자 장치(2001)로부터 멀리 위치될 수 있고, 보안 규칙(2018)을 제공하거나 다른 정보를 송수신하기 위해 런칭 모듈(2020), O/S 하위 보안 에이전트(2004) 및 O/S 내 보안 에이전트(2001)와 같은 전자 장치(2001)의 구성요소들과 통신하도록 구성될 수 있다. 예를 들어 보호 서버(2022)는 자원들(2026)을 액세스하려는 의심스러운 시도들에 관한 정보를 수신할 수 있고, 이 정보를 이후의 분석을 위해 저장할 수 있다. 보호 서버(2022)는 도 1의 보호 서버(102), 도 2의 보호 서버(202) 및/또는 이들의 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0287] 보안 규칙들(2018)은 트래핑을 요하는 이벤트들 및 각각의 이벤트에 대한 적절한 응답을 특정하기 위한 어떤 적절한 규칙들, 로직, 명령들, 명령어들, 플래그들, 또는 다른 메커니즘들을 포함할 수 있다. 보안 규칙들(2018)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(220, 222), 도 4의 보안 규칙들(420, 422, 434, 436, 438), 도 5의 보안 규칙들(518), 도 7의 보안 규칙들(707, 721, 723) 및/또는 이들의 어떤 조합의 기능을 구현

하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0288] 런칭 모듈(2020)은 OS 하위 보안 에이전트(2004), 운영체제(2012), 및 O/S 내 보안 에이전트(2016)와 같은 OSEE(2008)의 구성요소들이 시동 전 악성 소프트웨어에 의해 제약 받지 않도록 보장함으로써 OSEE(2008)의 안전한 시동을 제공하도록 구성될 수 있다. 런칭 모듈(2020)은 O/S 하위 보안 에이전트(2004), 운영체제(2012) 및 O/S 내 보안 에이전트(2016)가 O/S 하위 보안 에이전트(2004), 운영체제(2012) 및 O/S 내 보안 에이전트(2016)와 관련된 하나 이상의 보호 파일들의 무결성을 검증함으로써 악성 소프트웨어에 의해 제약을 받는지 여부를 평가할 수 있다. 런칭 모듈(2020)이 보호 파일들 중 어느 하나에서 악성 소프트웨어를 검출하는 경우, 런칭 모듈(2020)은 백업 사본으로부터 보호 파일들을 복구하도록 구성될 수 있다. 런칭 모듈(2020)이 OSEE(2008)의 구성요소들이 악성 소프트웨어에 의해 제약되지 않는다고 검증하거나 런칭 모듈(2020)이 악성 소프트웨어에 의해 제약 받는 OSEE(2008)의 어떤 구성요소를 성공적으로 복구한 후, 런칭 모듈(2020)이 OSEE(2008)를 시동할 수 있다. OSEE(2008)을 시동할 때, 런칭 모듈(2020)은 운영체제(2012)와 같은 OSEE(2008)의 다른 구성요소들을 시동하기 전에 O/S 하위 보안 에이전트(2004)를 시동할 수 있다.

[0289] 런칭 모듈(2020)이 OSEE(2008)의 안전한 시동을 성공적으로 제공한 후, O/S 하위 보안 에이전트(2004) 및 O/S 내 보안 에이전트(2016)와 같은 OSEE(2008)의 구성요소들은 악성 소프트웨어가 전자 장치(2001)의 자원들(2026)을 감염시키는 것을 막도록 협력할 수 있다. 예를 들어 O/S 하위 보안 에이전트(2004) 및 O/S 내 보안 에이전트(2016)는 보안 규칙들(2018)에 의해 특정된 바와 같이, 스토리지 장치(2026) 상의 다양한 보호 파일들을 액세스하기 위한 시도들을 인터셉트하도록 구성될 수 있다. 보호 파일들은 런칭 모듈(2020), O/S 하위 보안 에이전트(2004), 또는 O/S 내 보안 에이전트(2016)과 관련된 파일들 또는 운영체제(2012)의 코어 파일들을 포함할 수 있다. 이러한 파일들을 악성 소프트웨어로부터 보호하는 것은 이러한 구성요소들에 의해 이용되는 보호장치(safeguard)가 악성 소프트웨어에 의해 전복되지 않게 확실히 하는 것을 도울 수 있다. 예를 들어, 운영체제(2012)가 실행 중에 런칭 모듈(2020)을 악성 소프트웨어로부터 보호함으로써, 런칭 모듈(2020)은 전자 장치(2001)의 다음 시동 시 악성 소프트웨어에 대한 염려가 없을 것이다. 이러한 방식으로 전자 장치(2001)가 부팅될 때, O/S 하위 보안 에이전트(2004), O/S 내 보안 에이전트(2016) 및 운영체제(2012)와 같은 OSEE(2008)의 구성요소들이 런칭 모듈(2020)에 의해 악성 소프트웨어가 있는지 체크될 수 있고, 런칭 모듈(2020)은 운영체제(2012)가 실행 중일 때 OSEE(2008)의 구성요소들에 의해 악성 소프트웨어로부터 보호될 수 있다.

[0290] 도 21은 안전한 운영체제 실행 환경을 제공하기 위한 시스템 내 런칭 모듈(2102)의 실시예이다. 도 21의 구성요소들은 도 20 및 도 22에서 그들과 공통적으로 불리는 대응요소들과 동일할 수 있다. 런칭 모듈(2102)은 예컨대 도 20의 시스템으로부터의 런칭 모듈(2020)이나 도 22의 시스템으로부터의 런칭 모듈(2226)의 기능을 구현하는데 사용될 수 있다. 런칭 모듈(2102)은 O/S 하위 보안 에이전트(2128), 운영체제(2124) 및 O/S 내 보안 에이전트(2126)을 안전하게 시동함으로써 안전한 운영체제 실행 환경(2122)을 제공하도록 구성될 수 있다.

[0291] 런칭 모듈(2102)은 부팅 에이전트(2104), 보안 런칭 에이전트(2110) 및 복구 에이전트(2112)를 포함할 수 있다. 부팅 에이전트(2104)는 전자 장치(2101)가 개시될 때, 운영체제(2124) 및 어떤 다른 소프트웨어(가령, 악성 소프트웨어)에 앞서 보안 런칭 에이전트(2110)가 부팅되게 하도록 구성될 수 있다. 보안 런칭 에이전트(2110)는 OSEE(2122)를 안전하게 시동하도록 구성될 수 있다. OSEE(2122)는 운영체제(2124)를 안전하게 실행하기 위한 실행 환경일 수 있다. 보안 런칭 에이전트(2110)는 O/S 하위 보안 에이전트(2128), 운영체제(2124) 및 O/S 내 보안 에이전트(2126)가 악성 소프트웨어에 감염되었는지 여부를 판단하기 위해 보안 규칙들(2116)을 이용함으로써 OSEE(2122)의 안전한 시동을 제공할 수 있다. 예를 들어, 알려진 패턴의 악성 소프트웨어에 대해 스토리지 장치(2114) 상의 각각의 구성요소에 대한 디스크 이미지를 검색하고, 각각의 구성요소의 디스크 이미지의 암호화된 해시 값들을 비교하고/하거나 악성 소프트웨어를 검출하기 위한 어떤 다른 적절한 방법을 이용함으로써 OSEE(2122)의 구성요소들에 악성 소프트웨어가 있는지 체크할 수 있다. 보안 런칭 에이전트(2110)가 악성 소프트웨어 감염을 검출하면, 복구 에이전트(2112)가 악성 소프트웨어 감염으로부터의 복구를 위해 사용될 수 있다. 보안 런칭 에이전트(2110)에 의해 아무 악성 소프트웨어 감염도 검출되지 않거나 복구 에이전트(2112)에 의해 성공적인 복구가 이루어진 경우, 보안 런칭 에이전트(2110)는 O/S 하위 보안 에이전트(2128), 운영체제(2124) 및 O/S 내 보안 에이전트(2126)를 시동하도록 구성될 수 있다. O/S 하위 보안 에이전트(2128)는 도 22의 O/S 하위 보안 에이전트(2208)의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. O/S 내 보안 에이전트(2126)는 도 22의 O/S 내 보안 에이전트(2206)의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 운영체제(2124)는 도 20의 운영체제(2012)의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 스토리지 장치(2114)는 도 20의 스토리지 장치(2006)의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보안 규칙들(2116)은 도 20의 보안 규칙들(2018)의 기능을 구현

하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0292] 부팅 에이전트(2104)는 마스터 부트 레코드("MBR") 관리자(2106) 및 부트스트랩 로더(2108)를 포함할 수 있고, 전자 장치(2101)가 개시될 때 운영체제(2124) 및 악성 소프트웨어와 같은 어떤 다른 소프트웨어 에 앞서 보안 런칭 에이전트(2110)가 부팅되게 하도록 구성될 수 있다. MBR 관리자(2106)는 스토리지 장치(2114) 상의 기존 MBR(2130)을 부트스트랩 로더(2108)로 대체하도록 구성될 수 있다. MBR(2130)은 스토리지 장치의 제1섹터(즉, 섹터 0) 상에 위치할 수 있으며, 전자 장치(2101)가 개시될 때 운영체제(2124)나 다른 소프트웨어의 부팅을 담당할 수 있다. MBR(2130)을 부트스트랩 로더(2108)로 교체함으로써, 부트스트랩 로더(2108)는 새로운 MBR(2130)이 될 수 있다. 원래의 MBR(2130)은 실행되지 않을 것이며, 그에 따라 MBR(2130)과 관련된 운영체제(2124)나 다른 소프트웨어는 부팅되지 못할 것이다. 대신, 전자 장치(2101)가 개시될 때, 부트스트랩 로더(2108)가 새 MBR(2130)이 되었기 때문에 그것이 실행될 것이다. 부트스트랩 로더(2108)는 OSEE(2122) 시동을 담당하는 보안 런칭 에이전트(2110)를 부팅하도록 구성될 수 있다. 이러한 방식으로, 보안 런칭 에이전트(2110)는 운영체제(2124) 및/또는 어떤 다른 소프트웨어에 앞서 부팅되어, 보안 런칭 에이전트(2110)가 O/S 하위 보안 에이전트(2128), O/S 내 보안 에이전트(2126), 및/또는 운영체제(2124)를 로딩하기 전에 악성 소프트웨어를 체크하게 할 수 있다.

[0293] 보안 런칭 에이전트(2110)는 OSEE(2122)를 시동하도록 구성될 수 있다. OSEE(2122)는 운영체제(2124)를 안전하게 실행하기 위한 실행 환경으로서 구성될 수 있으며, O/S 하위 보안 에이전트(2128), 운영체제(2124) 및 O/S 내 보안 에이전트(2126)를 포함할 수 있다. 보안 런칭 에이전트(2110)는 디스크 I/O 기능, 네트워크 I/O 기능 및 베이직 콘솔 I/O 기능을 제공할 수 있는 슬림 임베디드(slim embedded) 운영체제에 의해 구현될 수 있다. 다른 실시예에서, 보안 런칭 에이전트(2110)는 O/S 하위 보안 에이전트(2128)에 의해 구현될 수 있다. 보안 런칭 에이전트(2110)는 O/S 하위 보안 에이전트(2128), 운영체제(2124) 및 O/S 내 보안 에이전트(2126)가 악성 소프트웨어에 감염되었는지 여부를 검출하도록 구성될 수 있다. 악성 소프트웨어 감염을 검출하기 위해, 보안 런칭 에이전트(2110)는 암호화 해시 알고리즘을 이용하여 O/S 하위 보안 에이전트(2128), 운영체제(2124) 및 O/S 내 보안 에이전트(2126)와 관련된 다양한 보호 파일들(2120)의 무결성을 검증할 수 있다. 보호 파일들은 예컨대 운영체제(2124)의 코어 파일들 및 O/S 하위 보안 에이전트(2128) 및 O/S 내 보안 에이전트(2126)의 실행 가능 이미지들을 포함할 수 있다. 보호 파일(2120)의 무결성을 검증하기 위해, 보안 런칭 에이전트(2110)는 보호 파일(2120)에 대한 해시 값을 계산하기 위한 해시 알고리즘을 이용할 수 있다. 계산된 해시 값은 이제, 보호 파일(2120)에 대해 앞서 생성된 해시 값과 비교될 수 있다. 해시 값들이 서로 상이하면, 보호 파일(2120)은 악성 소프트웨어에 의해 수정되거나 치환되었을 가능성이 있다. 다양한 실시예들에서, 보안 에이전트(2110)는 보호 파일들(2120)의 무결성을 검증하기 위해 디스크 매핑 비트맵("DMB")(2118)를 활용할 수 있다. 디스크 매핑 비트맵(2118)은 스토리지 장치(2114) 상의 각각의 보호 파일(2120)의 위치를 특정할 수 있고, 또한 각각의 보호 파일(2120)에 대해 앞서 생성된 해시 값을 제공할 수 있다. 디스크 매핑 비트맵(2118)의 실시예들에 대한 설명은 도 23의 디스크 매핑 비트맵(2301)에 대한 논의들에서 찾아볼 수 있다. 보안 런칭 에이전트(2110)는 디스크 매핑 비트맵(2118)을 조회하여 스토리지 장치(2114) 상의 보호 파일(2120)의 위치를 식별하고, 보호 파일(2120)의 해시 값을 계산하고, 계산된 해시 값을 디스크 매핑 비트맵(2118)에 의해 제공된 앞서 생성된 해시 값과 비교할 수 있다. 보호 파일(2120)의 해시 값들이 매치하지 않으면, 보호 파일(2120)은 악성 소프트웨어에 의해 수정되었거나 치환되었을 가능성이 있다. 보안 런칭 에이전트(2110)는 잠정 악성 소프트웨어 감염으로부터 회복하기 위해 복구 에이전트(2112)를 시동할 수 있다. 아무 잠정 악성 소프트웨어 감염들도 검출되지 않거나, 복구 에이전트(2112)에 의해 모든 잠정적으로 감염된 파일들이 성공적으로 회복된 경우, 보안 런칭 에이전트(2110)는 O/S 하위 보안 에이전트(2128), 운영체제(2124) 및 O/S 내 보안 에이전트(2126)를 로딩하도록 진행할 수 있다. 보안 런칭 에이전트(2110)는 OSEE(2122)를 시동한 뒤 종료되도록 구성될 수 있다.

[0294] 복구 에이전트(2112)는 O/S 하위 보안 에이전트(2128), 운영체제(2124) 및 O/S 내 보안 에이전트(2126)와 관련된 하나 이상의 보호 파일들(2120)의 악성 소프트웨어 감염으로부터 복구하도록 구성될 수 있다. 악성 소프트웨어 감염으로부터 복구하기 위해, 복구 에이전트(2112)는 백업 스토리지 장치로부터 백업 파일들을 검색하고 감염된 보호 파일들(2120)을 해당 백업 파일들로 교체하도록 구성될 수 있다. 백업 파일들은 전자 장치(2101) 상에서 내부적으로, 예컨대 스토리지 장치(2114) 상에서 저장될 수 있다. 백업 파일들은 전자 장치(2101)로부터 원격 위치 안에도 저장될 수 있다. 예를 들어 백업 파일들은 네트워크를 통해 도 20의 보호 서버(2022)의 백업 스토리지 장치(2024) 같은 것에 저장될 수 있다. 백업 파일들의 메타데이터가 유지될 수 있고, 백업 파일이 생성된 날짜와 시간 및 개정 번호를 포함할 수 있다. 보호 파일들(2120)의 복구를 위해 백업 파일들을 사용하기 전에, 복구 에이전트(2112)는 백업 파일들이 악성 소프트웨어에 감염되지 않았다는 것을 보장하기 위해 백업 파일들의 무결성을 검증하도록 구성될 수 있다. 복구 에이전트(2112)는 보안 런칭 에이전트(2110)가 보호

파일들(2120)의 무결성을 검증하는 것과 비슷한 방식으로 백업 파일들의 무결성을 검증할 수 있다. 예를 들어 복구 에이전트(2112)는 백업 파일에 대한 해시 값을 계산할 수 있고, 계산된 해시 값을 디스크 매핑 비트맵(2118)으로부터의 상응하는 백업 파일의 해시 값과 비교할 수 있다. 해시 값들의 비교가 백업 파일이 악성 소프트웨어에 감염되었을 것이라는 것을 나타내는 경우, 그 백업 파일은 사용되지 않을 것이고/않거나 구 백업 파일이 사용될 수 있다. 복구 에이전트(2112)는 보안 런칭 에이전트(2110)가 O/S 하위 보안 에이전트(2128), 운영체제(2124) 및 O/S 내 보안 에이전트(2126)에서 진행할 수 있도록 보안 런칭 에이전트(2110)에 성공적 복구를 알리도록 구성될 수 있다.

[0295] 도 22는 운영체제를 안전하게 실행하기 위한 운영체제 실행 환경("OSEE")(2002)의 실시예이다. 도 22의 구성요소들은 도 20 및 도 22에서 그들과 공통적으로 불리는 대응요소들과 동일할 수 있다. OSEE(2202)는 예컨대 도 20으로부터의 OSEE(2008)이나 도 21로부터의 OSEE(2122)의 기능을 구현하는데 사용될 수 있다. OSEE(2202)는 운영체제(2204)를 안전하게 실행하기 위한 실행 환경으로서 구성될 수 있으며, 운영체제(2204), O/S 하위 보안 에이전트(2208), O/S 내 보안 에이전트(2206) 및/또는 디스크 보안 에이전트(2214)를 포함할 수 있다. OSEE(2202)는 런칭 모듈(2226)에 의해 안전하게 시동될 수 있다. 그런 다음, O/S 하위 보안 에이전트(2208), O/S 내 보안 에이전트(2206) 및/또는 디스크 보안 에이전트(2214)와 같은 OSEE(2202)의 구성요소들은 악성 소프트웨어가 전자 장치(2201)의 구성요소들을 저해하는 것을 막도록 협력할 수 있다. 예를 들어 OSEE(2202)의 구성요소들은 악성 소프트웨어로부터 런칭 모듈(2226)을 보호하도록 협력할 수 있다. 이러한 방식의 런칭 모듈(2226)의 보호는 전자 장치(2201)의 다음 초기화 시, 런칭 모듈(2226)에 의해 사용된 보호장치들이 악성 소프트웨어 감염된 운영체제(2204), O/S 하위 보안 에이전트(2208) 및/또는 O/S 내 보안 에이전트(2206)의 시동을 허용하도록 전복되지 않게 확실시 하는 것을 도울 수 있다.

[0296] OSEE(2202)는 O/S 하위 보안 에이전트(2208), 운영체제(2204), O/S 내 보안 에이전트(2206) 및/또는 디스크 보안 에이전트(2214)를 포함할 수 있다. OSEE(2202)는 런칭 모듈(2226)에 의해 안전하게 시동될 수 있다. 런칭 모듈(2226)이 OSEE(2202)의 안전한 시동을 성공적으로 제공한 후, O/S 하위 보안 에이전트(2208), O/S 내 보안 에이전트(2206), 및 디스크 보안 에이전트(2214)와 같은 OSEE(2202)의 구성요소들은 악성 소프트웨어가 런칭 모듈(2226)과 같은 전자 장치(2201)의 구성요소들을 저해하는 것을 막도록 협력할 수 있다.

[0297] O/S 하위 보안 에이전트(2208)는 O/S 하위 트래핑 에이전트(2210) 및 트리거된 이벤트 핸들러(2212)를 포함할 수 있다. O/S 하위 트래핑 에이전트(2210)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442) 또는 PC 펌웨어 보안 에이전트(444), 도 5의 펌웨어 보안 에이전트(516), 도 7의 마이크로코드 보안 에이전트(708) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 트리거된 이벤트 핸들러(2010)는 도 1의 트리거된 이벤트 핸들러(108), 도 2의 SVMM 보안 에이전트(217), 도 4의 O/S 하위 에이전트(450), 도 7의 O/S 하위 에이전트(712) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 다양한 실시예들에서 O/S 하위 트래핑 에이전트(2210)의 기능 중 일부는 트리거된 이벤트 핸들러(2212)에 의해 수행될 수 있거나, 트리거된 이벤트 핸들러(2212)의 기능 중 일부가 O/S 하위 트래핑 에이전트(2210)에 의해 수행될 수 있다. 일 실시예에서 트리거된 이벤트 핸들러(2212)는 O/S 하위 보안 에이전트(2208)와 동일한 우선순위 레벨에서 동작할 수 있다. 다른 실시예에서, 트리거된 이벤트 핸들러(2212)는 O/S 내 보안 에이전트(2206)의 일부로서 구현될 수 있으며, 운영체제(2204)의 우선순위 레벨이나 그 위에서 동작할 수 있다. 또 다른 실시예에서, 트리거된 이벤트 핸들러(2212)는 둘 이상의 트리거된 이벤트 핸들러들에 의해 구현될 수 있으며, 이때 적어도 한 트리거된 이벤트 핸들러는 O/S 하위 보안 에이전트(2208)와 동일한 우선순위 레벨에서 동작하며 적어도 하나의 트리거된 이벤트 핸들러는 운영체제(2204)의 우선순위 레벨이나 그 위에서 동작한다.

[0298] O/S 하위 보안 에이전트(2208)는 스토리지 장치(2218)와 같은 전자 장치(2201)의 자원들을 액세스하려는 요청들을 인터셉트하기 위해 O/S 하위 트래핑 에이전트(2210)를 사용하도록 구성될 수 있다. 스토리지 장치(2218)를 액세스하라는 요청을 인터셉트하면, O/S 하위 트래핑 에이전트(2210)는 트래핑된 액세스 시도와 관련된 트리거된 이벤트를 생성하도록 구성될 수 있고, 그 이벤트에 대해 취할 적합한 액션을 결정하기 위해 트리거된 이벤트 핸들러(2212)로 그 트리거된 이벤트를 보내도록 구성될 수 있다. 트리거된 이벤트는 상기 요청과 관련된 스토리지 장치(2218)의 영역(가령, 섹터 및/또는 파일), 요청한 개체 및 요청된 액세스의 타입과 같은 정보를 포함할 수 있다. 요청한 개체는 운영체제(2204), 드라이버(2228) 또는 애플리케이션(2230)과 같은 요청을 개시할 책임이 있는 개체이다. 요청된 액세스 타입은 스토리지 장치(2218)로부터의 코드를 읽기, 쓰기 또는 실행하라는 요청을 포함할 수 있다.

[0299] 트리거된 이벤트 핸들러(2212)는 O/S 하위 트래핑 에이전트(2210)로부터 트리거된 이벤트들을 수신하여 처리하

도록 구성될 수 있다. 트리거된 이벤트들은 O/S 하위 트래핑 에이전트(2210)에 의해 트래핑되었던 스토리지 장치(2218)를 액세스하라는 요청에 대한 정보를 포함할 수 있다. 트리거된 이벤트 핸들러(2212)는 스토리지 장치(2218)의 보호 영역들을 액세스하려는 시도들을 식별하고 적합한 반응을 결정하기 위해, 트리거된 이벤트와 관련된 정황 정보와 함께 하나 이상의 보안 규칙들(2216)을 활용하도록 구성될 수 있다. 보호 섹터 및/또는 파일과 같은 보호 영역을 액세스하려는 시도를 식별한 후, 트리거된 이벤트 핸들러(2212)는 보호 영역을 액세스하려는 시도가 허가된 것인지 여부를 판단하기 위해 보안 규칙들(2216)을 조회하도록 구성될 수 있다. 트리거된 이벤트 핸들러(2212)는 또한 O/S 하위 보안 에이전트(2208)로 적합한 액션에 대한 결정을 제공하도록 구성될 수 있다. 예를 들어 트리거된 이벤트 핸들러(2212)는 트리거된 이벤트가 허용되어야 하는지 거부되어야 하는지 여부, 또는 다른 교정 액션이 취해져야 하는지를 O/S 하위 보안 에이전트(2208)로 알릴 수 있다.

[0300] O/S 내 보안 에이전트(2206)는 도 1의 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418), 도 7의 O/S 내 보안 에이전트(719), 또는 이들의 어떤 적절한 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(2206)는 운영체제(2204)의 우선순위 레벨이나 그 위에서 실행될 수 있고, 악성 소프트웨어로부터 전자 장치(2201)를 보호하기 위해 하나 이상의 보안 규칙들(2216)을 조회하도록 구성될 수 있다. 예를 들어 보안 규칙들(2216)은 스토리지 장치(2218) 상의 어떤 보호 파일들(2222)을 액세스하려는 시도들을 인터셉트할 것을 O/S 내 보안 에이전트에 요구할 수 있다. 보안 규칙들(2216)은 보호 파일(2222)을 액세스하려는 특정 시도가 허가된 것인지 여부를 더 특정할 수 있다. 그러나, O/S 내 보안 에이전트(2206)가 운영체제(2204)의 우선순위 레벨이나 그 위에서 실행되기 때문에, O/S 내 보안 에이전트(2206) 자체가 운영체제(2204) 상에서 실행되는 악성 소프트웨어에 감염될 수 있고 O/S 내 보안(2206)의 보호장치들이 우회될 수 있다. 이러한 가능성을 막도록 돕기 위해, O/S 하위 보안 에이전트(2208)는 악성 소프트웨어로부터 O/S 내 보안 에이전트(2206)를 보호하도록 구성될 수 있다.

[0301] 디스크 보안 에이전트(2214)는 DMB 발생기(2232) 및 디스크 보호기(2234)를 포함할 수 있으며, 악성 소프트웨어로부터 런칭 모듈(2226) 및 OSEE(2202)의 구성요소들과 같은 전자 장치(2201)의 구성요소들을 보호하는 데 사용될 수 있다. 디스크 보안 에이전트(2214)는 모든 적절한 방식에 따라 구현될 수 있다. 일 실시예에서 디스크 보안 에이전트(2214)는 O/S 하위 보안 에이전트(2208)의 일부로서 구현될 수 있고/있거나 O/S 하위 보안 에이전트(2208)와 동일한 우선순위 레벨에서 실행될 수 있다. 다른 실시예에서, 디스크 보안 에이전트(2214)는 O/S 내 보안 에이전트(2206)의 일부로서 구현될 수 있으며, 운영체제(2204)의 우선순위 레벨이나 그 위에서 동작할 수 있다. 또 다른 실시예에서, 디스크 보안 에이전트(2214)는 둘 이상의 디스크 보안 에이전트들에 의해 구현될 수 있으며, 이때 적어도 한 디스크 보안 에이전트는 O/S 하위 보안 에이전트(2208)와 동일한 우선순위 레벨에서 동작하며 적어도 하나의 디스크 보안 에이전트는 운영체제(2204)의 우선순위 레벨이나 그 위에서 동작한다.

[0302] 디스크 보호기(2234)는 런칭 모듈(2226) 및 OSEE(2202)의 구성요소들을 그러한 구성요소들과 관련된 다양한 보호 파일들(2222)을 액세스하려는 허가되지 않은 시도들을 인터셉트함으로써 악성 소프트웨어로부터 보호하도록 구성될 수 있다. 보호 파일들(2222)은 코어 운영체제 파일들(가령, 운영체제 커널 파일들), 코어 보안 에이전트 파일들(가령, O/S 하위 보안 에이전트(2208) 및 O/S 내 보안 에이전트(2206)의 실행 가능 이미지들), 및/또는 이러한 파일들의 백업 사본들을 포함할 수 있다. 디스크 보호기(2234)는 보호 파일들(2222)이 저장되는 스토리지 장치(2218)의 섹터들을 액세스하려는 허가되지 않은 시도들을 인터셉트함으로써 보호 파일들(2222)에 대한 허가되지 않은 액세스를 막을 수 있다. 어떤 실시예들에서 디스크 보호기(2234)는 보호 파일(2222) 및 보호 파일들(2222)이 저장된 스토리지 장치(2218) 상의 섹터들을 식별하기 위해 디스크 매핑 비트맵을 이용할 수 있다. 디스크 매핑 비트맵(2220)의 실시예들에 대한 설명은 도 23의 디스크 매핑 비트맵(2301)에 대한 논의들에서 찾아볼 수 있다. 디스크 매핑 비트맵(2220)은 예컨대 각각의 보호 파일이 저장된 스토리지 장치의 섹터 또는 섹터들을 포함하여, 다양한 보호 파일들과 관련된 정보를 포함할 수 있다. 디스크 보호기(2234)는 보호 파일들(2222)이 저장된 스토리지 장치(2218)의 섹터들을 식별하기 위해 디스크 매핑 비트맵(2220)을 조회할 수 있다. 디스크 보호기(2234)는 이때 보호 파일들(2222)과 관련된 섹터들을 액세스하려는 시도들을 인터셉트할 수 있고, 시도가 허가된 것인지 여부를 판단하기 위해 보안 규칙들(2216)을 조회할 수 있다. 예를 들어 보안 규칙들(2216)은 코어 운영체제 파일들로의 쓰기 요청이 운영체제(2204)로부터 나온 것이 아니라면 거부되어야 한다고 특정할 수 있다.

[0303] 어떤 실시예들에서 디스크 보호기(2234)의 기능은 O/S 하위 보안 에이전트(2208)의 구성요소들에 의해 구현될 수 있다. 디스크 보호기(2234)를 O/S 하위 보안 에이전트(2208)의 구성요소로서 구현함으로써, 디스크 보호기(2234)는 운영체제(2204) 밑의 레벨에서 실행할 수 있으며 운영체제(2204)를 괴롭히는 많은 악성 소프트웨어를

피할 수 있다. 디스크 보호기(2234)의 기능은 예컨대 O/S 하위 트래핑 에이전트(2210) 및 트리거된 이벤트 핸들러(2212)에 의해 구현될 수 있다. O/S 하위 트래핑 에이전트(2210)는 보호를 요하는 스토리지 장치(2218)의 섹터들을 식별하기 위해 디스크 매핑 비트맵(2220)을 조회하도록 구성될 수 있다. O/S 하위 트래핑 에이전트는 스토리지 장치(2218)의 식별된 섹터들을 액세스하려는 시도들을 트래핑하도록 더 구성될 수 있으며, 시도가 허가된 것인지 여부를 판단하기 위해 보안 규칙들(2216)을 활용할 수 있다. 이러한 방식으로, 디스크 매핑 비트맵(2220)에 의해 식별된 보호 파일들(2222)이 허가되지 않은 액세스로부터 보호될 수 있다.

[0304] 다른 실시예들에서 디스크 보호기(2234)의 기능은 O/S 내 보안 에이전트(2206)의 구성요소들로서 구현될 수 있다. 예를 들어 O/S 내 보안 에이전트(2206)는 디스크 보호기(1133)의 기능을 구현하기 위해 디스크 필터 드라이버를 포함할 수 있다. 필터 드라이버는 운영체제(2204)의 특정 장치를 위한 기존의 드라이버 스택 안에 삽입될 수 있고 전에 존재한 드라이버들의 기능을 보충하는 데 사용될 수 있는 드라이버(2228)일 수 있다. 예를 들어 디스크 필터 드라이버는 디스크(가령, 스토리지 장치(2218))의 기존 드라이버 스택 안에 삽입될 수 있으며 이전에 존재한 디스크 드라이버들의 기능을 보충할 수 있다. 디스크 필터 드라이버는 디스크 매핑 비트맵(2220)을 조회하여 보호를 요하는 스토리지 장치(2218)의 섹터들을 식별함으로써 디스크 보호기(1133)의 기능을 구현할 수 있다. 디스크 필터 드라이버는 이때 스토리지 장치(2218)의 보호 섹터들을 액세스하려는 시도들을 인터셉트할 수 있고, 시도가 허가된 것인지 여부를 판단하기 위해 보안 규칙들을 이용할 수 있다. 이러한 방식으로, 디스크 매핑 비트맵(2220)에 의해 식별된 보호 파일들(2222)이 허가되지 않은 액세스로부터 보호될 것이다. 그러나, 디스크 필터 드라이버가 운영체제(2204)의 우선순위 레벨이나 그 위에서 실행되기 때문에, 디스크 필터 드라이버 자체가 운영체제(2204) 상에서 실행되는 악성 소프트웨어에 감염될 수 있고 디스크 필터 드라이버의 보호장치들이 우회될 수 있다. 그에 따라, 어떤 실시예들에서 디스크 보호기(2234)의 기능은 O/S 하위 보안 에이전트(2206) 및 O/S 내 보안 에이전트(2206) 둘 모두에 의해 구현될 수 있다. 예를 들어 O/S 내 보안 에이전트(2206)는 상술한 바와 같이, 스토리지 장치(2218)를 액세스하려는 허가되지 않은 시도들을 인터셉트하기 위해 디스크 필터 드라이버를 사용하도록 구성될 수 있고, O/S 하위 보안 에이전트(2208)는 메모리나 스토리지 장치(2218) 내 디스크 필터 드라이버 이미지를 변경하려는 허가되지 않은 시도들을 막기 위해 구현됨으로써, 운영체제(2204)와 동일한 우선순위 레벨에서 실행되는 악성 소프트웨어에 의해 디스크 필터 드라이버가 전복되는 것을 막을 수 있다.

[0305] 디스크 보호기(2234)는 전자 장치(2201)의 셧다운에 앞서, MBR의 무결성을 검증하도록 더 구성될 수 있다. 예를 들어 전자 장치(2201)의 셧다운이 개시될 때, 디스크 보호기(2234)는 MBR(2224)의 해시 값을 계산하도록 구성될 수 있다. 디스크 보호기(2234)는 이때 MBR(2224)의 앞서 생성된 해시 값을 얻기 위해 디스크 매핑 비트맵(2220)을 조회할 수 있고, 계산된 해시 값을 앞서 생성된 해시 값과 비교할 수 있다. 해시 값들이 서로 상이하면, MBR(2224)은 악성 소프트웨어에 의해 변경되었을 가능성이 있고, 디스크 보호기(2234)는 MBR(2224)를 백업 사본으로 대체하도록 구성될 수 있다. 이러한 방식으로, 전자 장치(2201)의 다음 시동 시, 악성 소프트웨어 감염된 MBR(2224)가 부팅되지 않을 것이다.

[0306] DMB 발생기(2232)는 디스크 매핑 비트맵(2220)을 생성하고 업데이트하도록 구성될 수 있다. 예를 들어 DMB 발생기(2232)는 각각의 보호 파일(2222)이 저장된 스토리지 장치(2218)의 섹터들을 판단하도록 구성될 수 있고, 각각의 보호 파일(2222)의 해시 값을 생성하도록 더 구성될 수 있다. DMB 발생기(2232)는 보호 파일(2222)에 대한 해당 섹터들 및 해시 값을 디스크 매핑 비트맵(2220) 안에 저장할 수 있다. DMB 발생기(2232)는 모든 적절한 방식에 따라 구현될 수 있다. 예를 들어 DMB 발생기(2220)의 기능이 O/S 하위 보안 에이전트(2208)이나 O/S 내 보안 에이전트(2206)의 일부로서 구현될 수 있고, 혹은 DMB 발생기(2220)의 기능이 O/S 하위 보안 에이전트(2208) 및 O/S 내 보안 에이전트(2206)에 의해 구현될 수 있다.

[0307] 일 실시예에서 DMB 발생기(2232)는 보호 파일들(2222)을 액세스하라는 요청들을 인터셉트함으로써 디스크 매핑 비트맵(2220)을 생성할 수 있다. 예를 들어 O/S 내 보안 에이전트(2206)는 보호 파일들(2222)을 액세스하라는 요청들을 인터셉트하도록 구성된 파일 시스템 필터 드라이버를 포함할 수 있다. 파일 시스템 필터 드라이버는 파일 시스템이나 다른 파일 시스템 필터 드라이버를 타깃으로 하는 요청들을 인터셉트한다. 해당 요청이 의도한 타깃에 도달하기 전에 그 요청을 인터셉트함으로써, 필터 드라이버는 원래의 요청 타깃에 의해 제공되는 기능을 확장하거나 대체할 수 있다. O/S 내 보안 에이전트(2206)로부터의 파일 시스템 필터 드라이버가 보호 파일(2222)로 향하는 파일 I/O 요청들을 인터셉트할 수 있다. 필터 드라이버는 이때 보호 파일(2222)의 콘텐츠가 저장되는 스토리지 장치(2218) 상의 섹터들을 획득하기 위해 파일 시스템에 문의할 수 있다. 필터 드라이버는 이때 보호 파일(2222)의 디스크 섹터 레이아웃을 판단하기 위해 파일 시스템의 마스터 포맷 테이블(MFT)을 액세스할 수 있다. 디스크 매핑 비트맵(2220)은 보호 파일(2222)이 저장된 식별된 섹터들을 특정하기 위해 업데이

트될 수 있다. 보호 파일(2222)에 대해 아무 해시 값도 생성되어 있지 않으면, 해시 값이 생성될 수 있고 디스크 매핑 비트맵(2220)이 그 새로운 해시 값을 포함하도록 업데이트될 수 있다. 보호 파일(2222)이 업데이트될 경우 새로운 해시 값 또한 생성되어 디스크 매핑 비트맵(2220)에 저장될 수 있다. 예를 들어 파일 시스템 필터 드라이버가 보호 파일(2222)로의 쓰기 요청을 인터셉트하는 경우, 보호 파일(2222)의 수정된 콘텐츠를 이용하여 새로운 해시 값이 생성되어야 할 것이다.

[0308] 도 23은 안전한 운영체제 실행 환경을 제공하기 위한 시스템이나 방법 안에서 사용할 디스크 매핑 비트맵(2301)의 실시예이다. 디스크 매핑 배트맵(2301)은 예컨대 도 1의 디스크 매핑 비트맵(2118), 도 22의 디스크 매핑 비트맵(2220), 또는 도 26의 디스크 매핑 비트맵(2628)의 기능을 구현하는 데 사용될 수 있다. 디스크 매핑 비트맵(2301)은 파일일 수 있으며 다양한 보호 파일(2302)과 관련된 정보를 포함할 수 있다. 예를 들어 디스크 매핑 비트맵(2301)은 각각의 보호 파일(2302)이 저장되는 스토리지 장치의 섹터들(2304)을 식별할 수 있고 각각의 보호 파일(2302)에 대한 해시 값(2306)을 포함할 수 있다. 디스크 매핑 비트맵(2301)은 다양한 보호 파일들(2302)의 무결성을 검증하는 데 사용될 수 있다. 예를 들어 도 21의 보안 런칭 에이전트(2110) 및/또는 복구 에이전트(2112)가 보호 파일들(232)의 무결성을 검증하기 위해 디스크 매핑 비트맵(2301)으로부터의 정보를 이용할 수 있다. 디스크 매핑 비트맵(2301)은 예컨대 도 22의 DMB 발생기(2232)에 의해 생성될 수 있다.

[0309] 디스크 매핑 비트맵(2301)은 스토리지 장치 상의 지정된 섹터들 안에 저장될 수 있다. 지정된 섹터들은 운영체제의 파일 시스템을 구현하는 데 사용되는 스토리지 장치의 동일한 부분 상에 상주할 수 있다. 지정된 섹터들은 그 섹터들이 운영체제에 의해 사용되는 것을 막기 위해 점유되었다고 마크될 수 있다. 스토리지 장치는 또한 디스크 매핑 비트맵(2301)이 운영체제와 다른 파티션의 지정 섹터들 상에 저장되게 하도록 구획될 수 있다. 디스크 매핑 비트맵(2301)은 또한 네트워크 상에 위치한 원격 스토리지 장치 상에 저장될 수 있다. 예를 들어 디스크 매핑 비트맵(2301)은 도 20으로부터의 보호 서버(2022)나 도 26으로부터의 보호 서버(2602)와 같은 보호 서버 상에 저장될 수 있다.

[0310] 디스크 매핑 비트맵(2301)은 각각의 보호 파일(2302), 보호 파일(2302)이 저장된 스토리지 장치의 섹터나 섹터들(2304), 및 보호 파일(2302)에 대한 해시 값(2306)을 식별할 수 있다. 디스크 매핑 비트맵(2301)에 의해 식별된 보호 파일들(2302)은 코어 보안 에이전트 파일들(2308), 코어 운영체제 파일들(2310), 및 백업 파일들(2312)을 포함할 수 있다. 코어 보안 에이전트 파일들(2308)은 MBR 및 O/S 하위 보안 에이전트와 O/S내 보안 에이전트 실행자들을 포함할 수 있다. 코어 운영체제 파일들(2310)은 운영체제 커널 파일들 및 다른 운영체제 파일들을 포함할 수 있다. 예를 들어 운영체제가 마이크로소프트 윈도우즈의 어떤 변형인 경우, 코어 운영체제 파일들(2310)은 ntoskrnl.exe, hal.sys, win32k.sys, ntfs.sys, disk.sys, 및/또는 tcpip.sys를 포함할 수 있다. 코어 운영체제 파일들(2310)은 특정 운영체제에 따라 가변될 수 있다. 백업 파일들(2312)은 각각의 코어 보안 에이전트 파일(2308) 및 각각의 코어 운영체제 파일(2310)의 백업 사본을 포함할 수 있다. 다양한 실시예들에서, 백업 파일들(2312)은 코어 보안 에이전트 파일들(2308) 및 코어 운영체제 파일들(2310)과 동일한 스토리지 장치 상에 저장되지 않을 것이다. 그러한 실시예들에서 디스크 매핑 비트맵(2301)은 백업 파일들(2312)이 저장되는 특정 스토리지 장치를 식별할 수도 있다. 대안적으로, 섹터들(2304) 및 해시 값들(2306)과 같이 백업 파일들(2312)과 연관된 정보를 저장하기 위해 별개의 디스크 매핑 비트맵(2301)이 사용될 수 있다.

[0311] 각각의 보호 파일(2302)에 대해 디스크 매핑 비트맵(2301)은 암호화 해시 알고리즘을 이용하여 생성된 해시 값(2306)을 저장할 수 있다. 해시 알고리즘은 데이터의 블록을 입력으로서 수신할 수 있고 비트 스트링이나 해시 값을 출력으로서 생성할 수 있는 알고리즘을 포함할 수 있다. 서로 다른 데이터 세트들에 대한 해시 값들은 보통 서로 별개일 수 있다. 각각의 보호 파일(2302)에 대한 해시 값(2306)이 각각의 보호 파일(2302)의 콘텐츠를 이용하여 해시 알고리즘의 입력으로서 생성된다. 예컨대 시큐어 해시 알고리즘 2(Secure Hash Algorithm 2("SHA-2"))나 메시지 다이제스트 알고리즘 5(Message-Digest Algorithm 5("MD5"))을 포함하여, 어떤 적절한 암호화 해시 알고리즘이 사용될 수 있다.

[0312] 디스크 매핑 비트맵(2301)은 예컨대 도 21의 보안 런칭 에이전트(2110) 및/또는 복구 에이전트(2112), 또는 도 22로부터의 O/S 하위 보안 에이전트(2208), O/S 내 보안 에이전트(2206) 및/또는 디스크 보안 에이전트(2214), 도 26의 O/S 하위 보안(2616) 및/또는 O/S 내 보안 에이전트(2618)에 의해 보호 파일들(2301)의 잠정 악성 소프트웨어 감염을 검출하는 데 사용될 수 있다. 보호 파일(2302)의 잠정 악성 소프트웨어 감염을 검출하기 위해, 해시 알고리즘이 보호 파일(2302)의 무결성을 검증하는 데 사용될 수 있다. 디스크 매핑 비트맵(2304)은 보호 파일(2302)이 저장된 스토리지 장치 상의 섹터들(2304)를 식별하기 위해 조회될 수 있고, 그런 다음 보호 파일의 콘텐츠가 스토리지 장치의 적절한 섹터들(2304)로부터 검색될 수 있다. 이때 SHA-2나 MD5와 같이 선택된 해시 알고리즘이 보호 파일(2302)의 콘텐츠를 이용해 해시 값을 생성하는데 사용될 수 있고, 생성된 해시 값은 디

스크 매핑 비트맵(2301)으로부터의 대응 해시 값(2306)과 비교될 수 있다. 해시 값들이 서로 상이하면, 보호 파일(2302)은 악성 소프트웨어에 의해 수정되거나 치환되었을 가능성이 있다.

[0313] 디스크 매핑 비트맵(2301)는 모든 적절한 방식에 따라 생성될 수 있다. 일 실시예에서 디스크 매핑 비트맵(2301)은 보호 파일들(2302)을 액세스하라는 요청들을 인터셉트하고, 보호 파일들(2302)과 관련된 정보를 획득하고, 보호 파일들에 대한 정보로 디스크 매핑 비트맵(2301)을 업데이트함으로써 생성될 수 있다. 일부 실시예들에서, 요청은 예컨대 도 26으로부터의 O/S 하위 보안 에이전트(2616)와 같이 운영체제보다 낮은 우선순위 링에서 실행되는 소프트웨어에 의해 인터셉트될 수 있다. 다른 실시예들에서, 요청은 예컨대 도 26으로부터의 O/S 내 보안 에이전트(2618)와 같이 운영체제와 동일한 우선순위 링에서 실행되는 소프트웨어에 의해 인터셉트될 수 있다. 예를 들어 도 26으로부터의 O/S 내 보안 에이전트(2618)는 파일 시스템 필터 드라이버를 포함할 수 있다. 파일 시스템 필터 드라이버는 보호 파일(2302)로 향하는 파일 I/O 요청들을 인터셉트할 수 있다. 필터 드라이버는 이때 보호 파일(2302)의 콘텐츠가 저장되는 스토리지 장치 상의 섹터들(2304)을 획득하기 위해 파일 시스템에 문의할 수 있다. 필터 드라이버는 이때 보호 파일(2302)의 디스크 섹터 레이아웃을 판단하기 위해 파일 시스템의 마스터 파일 테이블(MFT)을 액세스할 수 있다. 디스크 매핑 비트맵(2301)은 보호 파일(2302)이 저장된 식별된 섹터들(2304)을 특정하기 위해 업데이트될 수 있다. 파일 시스템 필터 드라이버가 보호 파일(2302)로의 쓰기 요청을 인터셉트한 경우, 업데이트된 보호 파일(2302)의 콘텐츠를 이용하여 새로운 해시 값이 생성될 수 있고 디스크 매핑 비트맵(2301)이 새로운 해시 값을 저장하도록 업데이트될 수 있다

[0314] 도 24는 안전한 운영체제 실행 환경을 개시하기 위한 방법의 실시예이다. 단계 2410에서, 스토리지 장치의 기존 MBR은 보안 런칭 환경을 부팅하도록 구성된 다른 MBR로 대체될 수 있다. MBR은 스토리지 장치의 제1섹터(즉, 섹터 0)에 위치할 수 있으며 전자 장치의 시동 시 실행될 수 있다. 이러한 방식으로, 전자 장치가 개시될 때 원래의 MBR이 실행되지 않을 수 있고, 그에 따라 원래의 MBR과 관련된 운영체제나 다른 소프트웨어가 로딩되지 않을 수 있다. 대신, 다른 MBR이 실행되어 보안 런칭 환경을 로딩할 수 있다. 단계 2420에서 전자 장치가 개시될 수 있고 그에 따라 단계 2410에서의 다른 MBR이 실행될 수 있다. 다른 MBR은 보안 런칭 환경 로딩을 진행할 수 있다.

[0315] 단계 2430에서 보안 규칙들이 얻어질 수 있다. 보안 규칙들은 스토리지 장치 상에 내부적으로 저장되거나, 예컨대 보호 서버 상에 원격으로 저장될 수 있다. 그러한 보안 규칙들은 단계들 2440-2480에서 결정을 내리는 데 사용될 수 있다. 단계 2440에서 다양한 보호 파일들의 백업 사본들이 생성되었는지 여부가 판단될 수 있다. 백업을 요하는 보호 파일들은 보안 규칙들 안에서 특정될 수 있다. 백업 파일들은 예컨대 보안 런칭 환경과 관련된 다른 MBR 파일들, 하나 이상의 보안 에이전트들과 관련된 파일들, 및 코어 운영체제 파일들을 포함할 수 있다. 백업 사본들이 생성되었으면, 단계 2450에서 백업 사본들이 생성된다. 백업 사본들은 스토리지 장치 상에 내부적으로 저장되거나, 예컨대 보호 서버 상에 원격으로 저장될 수 있다.

[0316] 단계 2460에서, 보안 에이전트들 또는 운영체제가 악성 소프트웨어에 감염되었는지 여부가 판단될 수 있다. 보안 에이전트들은 O/S 하위 보안 에이전트 및/또는 O/S 내 보안 에이전트를 포함할 수 있다. 일 실시예에서 보안 에이전트들 및 운영체제는 그 보안 에이전트들 및 운영체제와 관련된 다양한 보호 파일들의 무결성을 검증함으로써 악성 소프트웨어에 대해 체크될 수 있다. 해싱 알고리즘이 보호 파일들의 무결성을 검증하는 데 사용될 수 있다. 예를 들어 보호 파일의 콘텐츠를 이용하여 각 보호 파일에 대한 해시 값이 계산될 수 있고, 계산된 해시 값은 보호 파일에 대해 앞서 생성된 해시 값과 비교될 수 있다. 보호 파일의 해시 값들이 서로 상이하면, 보호 파일은 악성 소프트웨어에 의해 수정되었을 가능성이 있다. 어떤 실시예들에서 디스크 매핑 비트맵은 각각의 보호 파일이 스토리지 장치 상에 저장되는 섹터들을 식별할 수 있고, 각각의 보호 파일에 대해 앞서 생성된 해시 값을 포함할 수도 있다. 그러한 실시예들에서 디스크 매핑 비트맵은 보호 파일의 콘텐츠가 저장된 섹터들을 결정하기 위해 조회될 수 있고, 해시 값은 보호 파일의 콘텐츠를 이용해 계산될 수 있다. 디스크 매핑 비트맵은 또한, 앞서 생성된 해시 값이 계산된 해시 값과 비교될 수 있도록, 보호 파일의 앞서 생성된 해시 값을 검색하도록 조회될 수 있다. 보호 파일의 해시 값들이 서로 상이하면, 악성 소프트웨어 감염이 추정될 수 있으며, 단계 2470에서 보호 파일들은 그 잠정적 악성 소프트웨어 감염으로부터 복구될 수 있다. 보호 파일들에 대한 해시 값들이 매치하면, 보호 파일들이 변경되지 않았을 것이고 따라서 악성 소프트웨어에 감염되지 않았을 수 있다. 그 경우, 방법은 보안 에이전트들 및 운영체제가 로딩될 수 있는 단계 2480으로 진행할 수 있다.

[0317] 단계 2470에서 잠정 악성 소프트웨어 감염에 대한 복구가 수행될 수 있다. 복구는 감염되었을 수 있는 각각의 보호 파일의 백업 사본들을 검색하고 그 잠정 감염된 보호 파일들을 해당 백업 사본으로 대체함으로써 수행될 수 있다. 백업 사본들은 로컬 스토리지 장치 상에 위치되거나, 보호 서버 상과 같이 멀리 위치될 수 있다. 잠

정 감염된 보호 파일들을 대체하기 위해 백업 사본들을 사용하기 전에, 백업 파일들 자체가 악성 소프트웨어에 감염되지 않았음을 확실히 하기 위해 백업 파일들의 무결성 역시 검증될 수 있다.

[0318] 해당 백업 사본들을 이용하여 보호 파일들이 복구되었으면, 단계 2480에서 보안 에이전트들 및 운영체제가 로딩될 수 있다. 보안 에이전트들은 O/S 하위 보안 에이전트 및/또는 O/S 내 보안 에이전트를 포함할 수 있다. O/S 하위 보안 에이전트는 운영체제 아래의 우선순위 레벨에서 실행될 수 있고, O/S 내 보안 에이전트는 운영체제와 같거나 그 상위의 우선순위 레벨에서 실행될 수 있다. O/S 하위 보안 에이전트 및 O/S 내 보안 에이전트는 악성 소프트웨어로부터 전자 장치를 보호하기 위해 협력할 수 있다. 예를 들어, O/S 하위 보안 에이전트 및 O/S 내 보안 에이전트는 허가되지 않은 액세스로부터 스토리지 장치와 같은 전자 장치의 자원들을 보호할 수 있다. 어떤 실시예들에서 보호는 O/S 하위 보안 에이전트, O/S 내 보안 에이전트 및/또는 운영체제의 안전한 시동을 제공하는 것을 담당할 수 있는 전자 장치의 구성요소들에 대해 제공될 수 있다. 예를 들어 O/S 하위 보안 에이전트 및 O/S 내 보안 에이전트는 단계들 2410-2470을 수행할 책임이 있는 구성요소들을 보호할 수 있다. 이런 식으로, 전자 장치가 다음에 개시될 때, 단계 2420에서 로딩되는 보안 런칭 환경이 악성 소프트웨어에 의해 구속되지 않을 수 있다.

[0319] 도 24로부터의 방법의 단계들은 필요 시, 지속적으로나 주기적으로나 요청에 따르거나 악성 소프트웨어 및/또는 다른 의심스러운 동향의 검출을 포함하는 어떤 이벤트의 트리거 시, 스토리지 장치를 보호하기 위해 반복될 수 있다.

[0320] 도 25는 운영체제를 안전하게 실행하기 위한 운영체제 실행 환경을 제공하는 방법(2500)의 실시예이다. 단계 2505에서, O/S 하위 보안 에이전트, O/S 내 보안 에이전트 및 보호 서버의 식별 및 보안이 인증될 수 있다. 그러한 인증은 암호화 해싱 및/또는 비밀 키들을 이용하여 각각의 구성요소에 대한 메모리 내 이미지들을 찾아 검증하는 것을 포함하는 어떤 적절한 방법을 이용하여 수행될 수 있다. 단계 2505가 완료될 때까지 다른 단계들의 동작은 보류될 수 있다. 단계 2510에서 보안 규칙들이 얻어질 수 있다. 보안 규칙들은 O/S 하위 보안 에이전트 및/또는 O/S 내 보안 에이전트에 의해 스토리지 장치 상에 내부적으로 저장되거나, 예컨대 보호 서버 상에 원격으로 저장될 수 있다. 그러한 보안 규칙들은 단계들 2515-1475에서 결정을 내리는 데 사용될 수 있다.

[0321] 단계 2515에서, 보호 파일을 액세스하려는 시도가 인터셉트될 수 있다. 인터셉트된 시도는 O/S 내 보안 에이전트와 같이 운영체제 레벨이나 그 위에서 발생하거나, O/S 하위 보안 에이전트와 같이 운영체제 하위의 레벨에서 발생할 수 있다. 보호 파일들은 MBR, 하나 이상의 보안 에이전트들과 관련된 파일들, 하나 이상의 보안 에이전트들(가령, 도 21에서의 로딩 모듈(2102))을 시동하는데 사용되는 파일들, 및 코어 운영체제 파일들을 포함할 수 있다. 보호 파일들은 보안 규칙들에 의해 특정될 수 있다. 단계 2520에서, 보호 파일에 대한 엔트리가 디스크 매핑 비트맵에 추가되어야 하는지 여부가 판단될 수 있다. 디스크 매핑 비트맵은 파일 또는 다른 데이터 구조로서 구현될 수 있으며, 각각의 보호 파일이 위치되는 스토리지 장치 상의 섹터들 및 각각의 보호 파일과 관련된 해시 값과 같이 보호 파일들에 대한 어떤 정보를 저장할 수 있다. 단계 2515에서 디스크 매핑 비트맵이 액세스되고 있는 보호 파일에 대한 이러한 정보를 포함하고 있지 않으면, 보호 파일의 엔트리가 디스크 매핑 비트맵에 추가될 수 있다. 예를 들어, 디스크 매핑 비트맵은 보호 파일이 저장되는 섹터들을 특정하지 않거나 보호 파일의 해시 값을 특정하지 않을 수 있다. 이러한 정보가 디스크 매핑 비트맵으로부터 누락되어 있으면, 단계 2525에서 디스크 매핑 비트맵이 이 정보를 포함하도록 업데이트될 수 있다. 디스크 매핑 비트맵을 업데이트하기 위해, 보호 파일의 콘텐츠를 저장하는 섹터들이 식별될 수 있고 보호 파일의 콘텐츠를 이용하여 해시 값이 생성될 수 있다. 보호 파일이 저장된 스토리지 장치 상의 섹터들을 판단하는 것은 파일 시스템에 문의하고 보호 파일의 섹터 레이아웃을 판단하기 위해 마스터 포맷 테이블(MFT)을 액세스하는 일을 수반할 수 있다. 보호 파일의 콘텐츠는 스토리지 장치의 적절한 섹터들로부터 검색될 수 있고, 그 보호 파일의 콘텐츠를 암호화 해싱 알고리즘의 입력으로서 사용하여 해시 값이 계산될 수 있다. 보호 파일의 해당 섹터들 및 계산된 해시 값은 이제 디스크 매핑 비트맵 안에 저장될 수 있다.

[0322] 단계 2530에서, 보호 파일에 대한 액세스가 허가되어 있는지 여부가 판단될 수 있다. 이러한 판단은 O/S 내 보안 에이전트와 같이 운영체제 레벨이나 그 위에서 일어나거나, O/S 하위 보안 에이전트와 같이 운영체제 하위의 레벨에서 일어날 수 있다. 요청한 개체가 보호 파일을 액세스하도록 허가될 수 있는지 여부를 판단하기 위해, 보호 파일을 액세스하고자 시도된 요청과 관련된 정황 정보가 보안 규칙들과 연계하여 분석될 수 있다. 예를 들어 보안 규칙들은 운영체제, 특정 애플리케이션 및/또는 특정 장치 드라이버가 보호 파일을 액세스하는 것이 허용될 수 있는지 없는지를 특정할 수 있다. 보안 규칙들은 또한, 보호 파일을 액세스하도록 허용될 수 있는 요청 개체에 대해 읽기, 쓰기 또는 실행과 같은 액세스 허가들을 특정할 수도 있다. 보호 파일에 대한 액세스가 허가되지 않으면, 단계 2555에서 액세스는 거부될 수 있다. 보호 파일에 대한 액세스가 허가되면, 단계

2535에서 그 보호 파일이 업데이트되고 있는지가 판단될 수 있다. 보호 파일이 업데이트되고 있으면, 단계 2540에서 디스크 매핑 비트맵 또한 업데이트될 수 있다. 예를 들어 보호 파일에 대한 업데이트가 파일을 저장하는 데 사용되는 스토리지 상의 섹터들의 변화를 가져오는 경우, 디스크 매핑 비트맵이 보호 파일을 저장하는 데 사용되는 적합한 섹터들을 식별하기 위해 업데이트될 수 있다. 또한, 보호 파일에 대한 새로운 해시 파일이 생성되어 디스크 매핑 비트맵 안에 저장될 수 있다. 단계 2545에서, 보호 파일의 백업 사본 역시 보호 파일의 최근 업데이트를 반영하도록 업데이트될 수 있다.

[0323] 보호 파일에 대한 액세스가 허가되면, 단계 2550에서 보호 파일에 대한 액세스가 허용될 수 있다. 보호 파일에 대한 액세스가 허가되지 않으면, 단계 2555에서 액세스가 거부될 수 있고, 단계 2560에서 해당 액세스 시도에 관한 의심스러운 정보가 보호 서버로 보고될 수 있다.

[0324] 단계 2565에서 전자 장치의 셧다운이 검출되는지가 판단될 수 있다. 셧다운이 검출되지 않으면, 단계 2515에서 보호 파일들을 액세스하고자 하는 시도들을 계속 인터셉트하도록 방법이 재개될 수 있다. 셧다운이 검출되면, 전자 장치의 다음 시동 시 악성 소프트웨어 감염된 MBR이 부팅되지 않게 보장하기 위해 MBR의 무결성이 단계 2570에서 검증될 수 있다. MBR의 무결성은 MBR의 콘텐츠를 사용해 해시 값을 계산하고, 계산된 해시 값을 디스크 매핑 비트맵으로부터의 이전에 생성된 해시 값과 비교함으로써 검증될 수 있다. 해시들이 서로 상이하면, MBR은 변경되었을 수 있으므로 백업 사본으로 대체될 수 있다. MBR의 무결성이 검증된 후, 단계 2575에서 전자 장치가 셧다운될 수 있다.

[0325] 도 25로부터의 방법의 단계들은 필요 시, 지속적으로나 주기적으로나 요청에 따르거나 어떤 이벤트의 트리거 시, 스토리지 장치를 보호하기 위해 반복될 수 있다.

[0326] 도 26은 무허가 액세스로부터 스토리지 장치(2606)를 보호하기 위한 시스템(900)의 실시예이다. 시스템(900)은 트리거된(triggered) 이벤트 핸들러(2608)와 통신 가능하게 연결되는 운영체제 하위(below-operating system ("O/S") 보안 에이전트(2616)를 포함할 수 있다. O/S 하위 보안 에이전트(2616)는 전자 장치(2601)의 스토리지 장치(2606)를 액세스하려는 시도들을 트래핑하도록 구성된 O/S 하위 트래핑 에이전트(2604)를 포함할 수 있다. O/S 하위 트래핑 에이전트(2604)는 트래핑된 액세스 요청과 관련된 트리거된 이벤트를 생성하여 그 트리거된 이벤트를 트리거된 이벤트 핸들러(2608)로 보내도록 구성될 수 있다. 트리거된 이벤트 핸들러(2608)는 그 트리거된 이벤트를 어떻게 다룰지를 결정하기 위해 하나 이상의 보안 규칙들(2614)이나 보호 서버(2602)를 조회하도록 구성될 수 있다. 트리거된 이벤트 핸들러(2608)는 또한, 트리거된 이벤트의 성향이 악성 소프트웨어에 대한 표시인지 아니면 스토리지 장치(2606)를 전복하기 위한 악의적 시도인지를 평가하도록 구성될 수 있다. 게다가 트리거된 이벤트 핸들러(2608)는 트리거된 이벤트가 허용되어야 할지 거부되어야 할지 여부에 대한 판단을 O/S 하위 트래핑 에이전트(2604)로 제공하도록 구성되거나, 다른 교정 액션을 산출하도록 구성될 수 있다. O/S 하위 보안 에이전트(2616)는 운영체제(1219)에서 실행되는 O/S 내 보안 에이전트(2612)와 통신 가능하게 연결될 수 있다. 시스템(900)은 백업 스토리지 장치(2620)를 이용하여 스토리지 장치(2606) 상의 데이터를 복구하도록 구성될 수 있다.

[0327] 전자 장치(2601)는 도 1의 전자 장치(103), 도 2의 전자 장치(104), 도 4의 전자 장치(404), 및/또는 도 7의 전자 장치(701) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(2601)는 메모리(2603)에 연결된 하나 이상의 프로세서들(2602)을 포함할 수 있다. 프로세서(2602)는 도 2의 프로세서(208), 도 4의 프로세서(406), 및/또는 도 7의 프로세서(702), 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(2603)는 도 2의 메모리(207), 도 4의 메모리(408) 및/또는 도 7의 메모리(703) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(2601)는 O/S 내 보안 에이전트(2618)를 포함할 수 있는 운영체제(2612)를 포함할 수 있다. 운영체제(2612)는 도 1의 운영체제(112), 도 2의 운영체제(212), 도 4의 운영체제(412) 및/또는 도 7의 운영체제(713), 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(2618)는 도 1의 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418) 및/또는 도 7의 O/S 내 보안 에이전트(719), 또는 이들의 어떤 적절한 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0328] 스토리지 장치(2606)는 도 1의 자원(106), 도 2의 시스템 자원들(214), 도 4의 스토리지(426) 또는 도 5의 I/O 장치(502)의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 스토리지 장치(2606)는 데이터나 다른 정보를 저장하기 위한 어떤 적절한 자원을 포함할 수 있다. 예를 들어 스토리지 장치(2606)는 제

한 없이, 직접 액세스 스토리지 장치(가령, 하드 디스크 드라이브나 플로피 디스크), 순차적 액세스 스토리지 장치(가령, 테이프 디스크 드라이브), 컴팩트 디스크, CD-ROM, DVD, 랜덤 액세스 메모리(RAM) 디스크 및/또는 플래시 메모리(가령, 플래시 기반 고체 상태 드라이브)를 포함할 수 있다. 스토리지 장치(2606)는 대규모 스토리지 장치를 포함할 수 있다. 스토리지 장치(2606)는 PCI, 직렬 ATA, USB 또는 파이어와이어(firewire)를 포함할 수 있으나 그에 국한되지 않는 시스템 버스를 이용한 접속 타입이나 인터페이스 방법과 무관하게 전자 장치(2601)에 연결되는 스토리지 장치를 포함할 수 있다. 스토리지 장치(2606)는 영구 블록 장치를 포함할 수 있다. 스토리지 장치(2606)는 각각이 고정된 데이터 양을 저장할 수 있는 하나 이상의 섹터들(924)로 분할될 수 있다. 예를 들어 스토리지 장치(2606)는 각각 512 바이트의 섹터들로 분할될 수 있지만 어떤 다른 적절한 섹터 사이즈가 사용될 수도 있다. 스토리지 장치(2606) 상의 섹터들(924)은 정적이거나 동적일 수 있다. 정적 섹터의 위치는 고정적인 반면 동적 섹터는 고정되지 않는다. 예를 들어 마스터 부트 레코드(2626)(MBR)은 정적이고, 스토리지 장치(2606) 상의 최초 섹터인 섹터 0에 위치한다. 보호를 요할 수 있는 동적 섹터들은 마스터 파일 테이블(즉, 파일 시스템 상에 저장된 모든 파일 관련 메타 데이터를 포함하는 파일), 운영체제 커널 파일들, 장치 드라이버들, 및 O/S 하위 보안 에이전트(2616)이나 O/S 내 보안 에이전트(2618)과 같은 안티 악성 소프트웨어 애플리케이션을 저장하는 섹터들을 포함할 수 있다. 동적 섹터들은 고정되어 있지 않으므로, 동적 섹터들 상에 저장된 파일들은 파일 시스템 상의 자신들의 정황적 존재로부터 파일들로부터의 데이터가 상주하는 스토리지 장치(2606) 상의 물리적 섹터들로 매핑되어야 한다.

[0329] O/S 하위 보안 에이전트(2616)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442) 또는 PC 펌웨어 보안 에이전트(444), 도 5의 펌웨어 보안 에이전트(516) 또는 도 7의 마이크로코드 보안 에이전트(708)의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 도 4의 펌웨어 보안 에이전트들(440 또는 442) 또는 도 5의 펌웨어 보안 에이전트(516)의 기능과 함께 O/S 하위 보안 에이전트(2618)를 구현하는 실시예들에서, O/S 하위 보안 에이전트(2616)은 스토리지 장치(2606)의 펌웨어에서 구현될 수 있다. 트리거된 이벤트 핸들러(2608)는 도 1의 트리거된 이벤트 핸들러(108), 도 2의 SVMM 보안 에이전트(217), 도 4의 O/S 하위 에이전트(450) 또는 도 7의 O/S 하위 에이전트(712)의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 도 4의 펌웨어 보안 에이전트들(440 또는 442) 또는 도 5의 펌웨어 보안 에이전트(516)의 기능과 함께 트리거된 이벤트 핸들러(2608)를 구현하는 실시예들에서, 트리거된 이벤트 핸들러(2608)는 스토리지 장치(2606)의 펌웨어에서 구현될 수 있다. 다양한 실시예들에서 O/S 하위 보안 에이전트(2616)의 기능 중 일부는 트리거된 이벤트 핸들러(2608)에 의해 수행될 수 있거나, 트리거된 이벤트 핸들러(2608)의 기능 중 일부가 O/S 하위 보안 에이전트(2616)에 의해 수행될 수 있다. 또한 O/S 하위 보안 에이전트(2616) 및 트리거된 이벤트 핸들러(2608)는 동일한 소프트웨어 모듈 안에서 구현될 수 있다.

[0330] O/S 하위 보안 에이전트(2616)는 전자 장치(2601)의 운영체제들(2612)보다 낮은 기능 레벨에서 구현될 수 있다. 예를 들어 O/S 하위 보안 에이전트(2616)는 운영체제(2612), 드라이버(2611) 또는 애플리케이션(2610)에 의한 스토리지 장치(2606)에 대한 액세스 시도를 인터셉트할 수 있다. O/S 하위 보안 에이전트(2616)는 운영체제 이용 없이 전자 장치(2601)의 프로세서 상에서 실행될 수 있다. 일 실시예에서 O/S 하위 보안 에이전트(2616)는 베어 메탈 환경이나 실행 레벨에서 운영될 수 있다. 또한 O/S 하위 보안 에이전트(2616)는 전자 장치(2601)의 프로세서에 의해 정의될 때, 전자 장치(2601)의 모든 운영체제들(2612)보다 높은 우선순위 링에서 실행될 수 있다. 예를 들어 보호 링(protection rings)을 이용하는 계층적 보호 도메인 모델과 관련하여 낮은 번호가 높은 우선순위를 나타낼 때, 운영체제(2612)는 "Ring0"에서 동작하고 O/S 하위 보안 에이전트(2616)는 "Ring -1"에서 동작할 수 있다. 드라이버들(2611)과 애플리케이션들(2610)은 "Ring0"나 "Ring3"에서 동작할 수 있다. 전자 장치(2601)의 운영체제들은 Ring0에서 실행될 수 있다.

[0331] "Ring -1"에서 실행됨으로써, O/S 하위 보안 에이전트(2616)는 운영체제(2612)와 같은 운영체제들을 괴롭히는 많은 악성 소프트웨어를 피할 수 있다. O/S 하위 보안 에이전트(2616)는 Ring0나 그 이상에서 실행되는 개체들에 대해 투명하게 동작될 수 있다. 그에 따라 O/S 하위 보안 에이전트(2616)가 존재하는지 여부와 관계없이, 스토리지 장치(2606)를 액세스하려는 시도가 운영체제(2612)나 다른 개체에 의해 동일한 방식으로 요청될 수 있다. O/S 하위 보안 에이전트(2616)는 스토리지 장치(2606)를 액세스하라는 요청을 시행할 때, 그 요청을 허용하거나, 그 요청을 거부하거나, 스토리지 장치(2606) 상의 데이터를 파괴하거나, 스토리지 장치(2606)의 매체 표면을 파괴하거나, 스토리지 장치(2606) 상의 데이터를 암호화하거나 다른 교정 액션을 취할 수 있다. 요청을 거부하기 위해, O/S 하위 보안 에이전트(2616)는 간단히 그 요청이 스토리지 장치(2606)이나 프로세서(2602)에 도달되는 것을 막거나, 해당 액션이 일어났다는 것을 운영체제(2612)에 납득시키기 위해 요청에 대한 스푸핑 또는 더미 응답을 제공할 수 있다. 그 요청을 허용하기 위해, O/S 하위 보안 에이전트(2616)는 단순히 그 요청을 스토리지 장치(2606)이나 프로세서(2602)로 넘길 수 있다. 데이터를 파괴하기 위해, O/S 하위 보안 에이전트

(2616)는 스토리지 장치(2606) 상의 데이터에 덮어 쓰거나 그렇지 않으면 그 데이터를 제거하도록 구성될 수 있다. 스토리지 장치(2606)의 매체 표면을 파괴하기 위해, O/S 하위 보안 에이전트(2616)는 스토리지 장치(2606)가 읽기나 쓰기 데이터에 대해 운용되지 않는 상태에 놓여지게 하는 액션을 수행할 수 있다. 스토리지 장치(2606) 상의 데이터를 암호화하기 위해, O/S 하위 보안 에이전트(2616)는 스토리지 장치(2606) 상의 데이터를 암호화하기 위한 어떤 적절한 암호화 알고리즘을 사용하고 스토리지 장치(2606) 상의 암호화되지 않은 데이터를 그 암호화된 데이터로 대체할 수 있다.

[0332] O/S 하위 보안 에이전트(2616)는 스토리지 장치(2606)를 액세스하려는 요청들을 트래핑하도록 구성된 O/S 하위 트래핑 에이전트(2604)를 포함할 수 있다. 스토리지 장치(2606)를 액세스하려는 요청들은 운영체제(2612), 드라이버(2611) 또는 애플리케이션(2610)에 의해 개시될 수 있다. O/S 하위 트래핑 에이전트(2604)는 해당 요청을 개시할 책임이 있는 요청한 개체를 식별하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(2604)는 트래핑된 액세스 시도와 관련된 트리거된 이벤트를 생성하고, 그 이벤트에 대해 취할 적합한 액션을 결정하기 위해 트리거된 이벤트 핸들러(2608)로 그 트리거된 이벤트를 보내도록 더 구성될 수 있다. 트리거된 이벤트는 상기 요청과 관련된 스토리지 장치(2606)의 영역(가령, 섹터 및/또는 파일), 요청한 개체 및 요청된 액세스의 타입과 같은 정보를 포함할 수 있다. 해당 요청과 관련된 스토리지 장치(2606)의 영역은 스토리지 장치(2606)의 하나 이상의 섹터들이거나 스토리지 장치(2606) 상에 저장된 파일일 수 있다. 요청하는 개체는 운영체제(2612), 드라이버(2611) 또는 애플리케이션(2610)일 수 있다. 예를 들어 애플리케이션(2610)이나 드라이버(2611)가 스토리지 장치(2606)에 대한 액세스를 요청한 경우, 트리거된 이벤트는 액세스를 요청한 특정 애플리케이션(2610)이나 드라이버(2611)를 가리킬 수 있다. 요청이 특정 애플리케이션(2610)이나 드라이버(2611)가 아닌 운영체제(2612)로부터 나온 것이면, 트리거된 이벤트는 요청이 그 운영체제(2612)로부터 나온 것이라고 가리킬 수 있다. 요청된 액세스 타입은 스토리지 장치(2606)로부터 판독하라거나, 스토리지 장치(2606)로 기입하라거나 스토리지 장치(2606) 상의 코드를 실행하라는 요청을 포함할 수 있다.

[0333] 일 실시예에서 O/S 하위 트래핑 에이전트(2604)는 시스템에 대한 공격, 악성 소프트웨어 감염, 또는 어떤 다른 잠정적 보안 위협과 같은 이벤트를 검출한 후에만 스토리지 장치(2606)를 액세스하라는 요청들을 트래핑하도록 구성될 수 있다. 그러한 실시예에서, 시스템(100)의 자원들은 잠정적 보안 위협이 검출될 때까지 보존된다. 다른 실시예에서, O/S 하위 트래핑 에이전트(2604)는 잠정 보안 위협이 검출되었는지 여부와 무관하게 항상, 스토리지 장치(2606)를 액세스하라는 요청을 트래핑하도록 구성될 수 있다.

[0334] 다른 실시예에서 O/S 하위 보안 에이전트(2616)는 파일 입/출력에 대해 드라이버들이나 시스템 함수들에 대한 호출의 실행을 트래핑함으로써 스토리지 장치(2606)에 대해 시도된 액세스들을 트래핑하도록 구성될 수 있다. 그러한 호출들의 트래핑은 가상 메모리 페이지 레벨에서 수행될 수 있고, 이때 그러한 드라이버들이나 시스템 함수들을 포함하는 메모리 페이지들이 O/S 하위 보안 에이전트(2616)에 의해 식별되어 보호될 수 있다. 그 경우, O/S 하위 보안 에이전트(2616)는 부분적으로나 전체적으로, 예컨대 가상 머신 모니터나 마이크로코드를 통해 구현될 수 있다. 그러한 호출들의 트래핑은 물리적 메모리 어드레스 레벨에서 수행될 수 있고, 이때 그러한 드라이버들이나 시스템 함수들의 코드 섹션들의 메모리 어드레스들이 O/S 하위 보안 에이전트(2616)에 의해 식별되어 보호될 수 있다. 그 경우, O/S 하위 보안 에이전트(2616)는 부분적으로나 전체적으로, 예컨대 마이크로코드를 통해 구현될 수 있다. 악성 소프트웨어가 그러한 함수들을 직접 호출할 수 있는데, 이 경우 O/S 하위 보안 에이전트(2616)는 그 호출자가 스토리지 장치(2606)의 특정 부분을 액세스하기 위한 허가권을 가지는지 여부를 식별하기 위해 그러한 함수의 호출자를 판단할 수 있다. 악성 소프트웨어는 예컨대 파일 함수들의 문서화되지 않은 서브함수들을 호출하거나 해당 함수를 전혀 호출하지 않고 함수의 코드 섹션 안으로 바로 분기함으로써, 간접적으로 그러한 함수들을 호출할 수 있다. 그러한 시도들은 호출자의 아이디를 감추거나 그렇지 않으면 악성 소프트웨어에 의해 파일 I/O의 사용을 모호하게 하는 데 사용될 수 있다. 그러한 경우, O/S 하위 보안 에이전트(2616)는 서브함수들의 실행을 트래핑하거나 파일 I/O 함수들의 코드 섹션으로 이끄는 JMP나 분기 명령어를 트래핑함으로써, 시도된 파일 I/O를 트래핑할 수 있다. 그러한 동향 자체가 의심스러우므로, 호출자가 알려져 있지 않더라도 O/S 하위 보안 에이전트(2616)는 그러한 액세스 시도의 호스트가 의심스럽고 그 시도가 악성 소프트웨어를 나타낸다고 판단하도록 구성될 수 있다.

[0335] 또 다른 실시예에서 O/S 하위 보안 에이전트(2616)는 디스크들을 액세스하기 위해 생성된 인터럽트들을 트래핑함으로써 스토리지 장치(2606)의 액세스 시도를 트래핑하도록 구성될 수 있다. 그러한 인터럽트들은 파일 I/O를 위해 정상적 함수에 의해 호출되거나, 그 함수들의 사용을 피해 스토리지 장치(2606)로 직접 쓰기를 시도하는 악성 소프트웨어에 의해 생성될 수 있다. O/S 하위 보안 에이전트(2616)는 인터럽트의 소스를 판단하고, 인터럽트의 성격을 식별하고, 어떤 정황 정보나 파라미터들을 식별하고, 인터럽트의 타깃을 식별하고, 해당 시도

가 의심스러운지 아닌지 여부를 판단하도록 구성될 수 있다. 해당 시도가 의심스러운지 여부에 대한 판단은 예컨대 호출자의 아이디, 또는 해당 액션 자체가 의심스러웠는지 여부를 포함할 수 있다. 예를 들어 악성 소프트웨어는 일련의 명령어들을 실행할 수 있으며, 이때 기입될 섹터들의 카운트(가령. "MOV al, count"), 기입될 트랙의 식별(가령 "MOV ch, track"), 기입될 섹터의 식별(가령, "MOV cl, sector"), 기입될 헤드의 식별(가령, "MOV dh, head"), 기입될 볼륨의 식별(가령, "MOV dl, drive"), 수행될 파일 I/O의 타입에 대한 식별(가령, "MOV ah, 03h") 및 파일로 기입될 데이터의 메모리 위치(가령, "MOV bx, buf")가 범용 레지스터들 안으로 옮겨질 수 있다. 특정 범용 레지스터들로의 그러한 정보의 할당은 이어지는 파일 I/O 인터럽트의 정보를 로딩하기 위한 어떤 알려진 방법일 수 있다. 이러한 할당들은 "MOV" 명령어를 사용하여 이루어질 수 있다. 이어서, "INT 13h"와 같이 인터럽트 13을 생성하기 위한 명령어가 실행될 수 있다. O/S 하위 보안 에이전트(2616)는 명령을 트래핑하고, 시도된 파일 I/O의 성격 및 스토리지 장치(2606)의 타깃 부분을 판단하기 위해 관련 레지스터들의 콘텐츠를 검사하도록 구성될 수 있다. O/S 하위 보안 에이전트(2616)는 그러한 동작의 호출자가 스토리지 장치(2606)의 특정 부분에 대한 기입에 대해 허가권을 가지는지 여부를 결정하기 위해 보안 규칙들을 조회하도록 구성될 수 있다. O/S 하위 보안 에이전트(2616)는 그러한 명령들의 시퀀스가 허가된 파일 I/O 드라이버로부터 일어났는지 여부나 그들이 알려지지 않거나 악의적 프로세스에 의해 직접 실행되었는지 여부를 판단하기 위해 실행 이력을 검사하도록 구성될 수 있다. 그러한 경우, 호출자는 그의 상태가 이전에 악성이라고 알려져 있지 않다고 하더라도, 그러한 동향에 기반하여 악성이라고 판단될 수 있다. 마지막으로, 인터럽트를 실행하기 위해 표준 파일 I/O 드라이버 호출이 이루어졌더라도, 드라이버의 호출자가 식별될 수 있고 O/S 하위 보안 에이전트(2616)는 호출자가 문제의 스토리지 장치(2606)의 부분을 액세스하기 위해 허가권을 가지는지 여부를 판단하도록 구성될 수 있다.

[0336] O/S 하위 에이전트(2616)는 매핑 에이전트(2622)를 포함할 수 있다. 매핑 에이전트(2622)는 어떤 파일을 파일 시스템 상의 그것의 정황적 존재로부터 파일이 저장되는 스토리지 장치(2606) 상의 섹터들(924)로 매핑하도록 구성될 수 있다. 일 실시예에서 매핑 에이전트(2622)는 O/S 하위 보안 에이전트(2616)와 동일한 우선순위 링에서 동작할 수 있다. 다른 실시예에서 매핑 에이전트(2622)는 O/S 내 보안 에이전트(2618)의 일부로서 구현될 수 있고, 운영체제(2612), 드라이버(2611) 또는 애플리케이션(2610)과 동일한 우선순위 링에서 동작할 수 있다. 또 다른 실시예에서, 매핑 에이전트(2622)는 둘 이상의 트리거된 매핑 에이전트들에 의해 구현될 수 있으며, 이때 적어도 한 매핑 에이전트는 O/S 하위 보안 에이전트(2616)와 동일한 우선순위 링에서 동작하며 적어도 하나의 매핑 에이전트는 운영체제(2612), 드라이버(2611) 또는 애플리케이션(2610)의 우선순위 링에서 동작한다. 매핑 에이전트(2622)는 O/S 하위 트래핑 에이전트(2604)나 트리거된 이벤트 핸들러(2608)로부터 파일을 매핑하라는 요청을 수신할 수 있고, 파일이 저장되는 스토리지 장치(2606) 상의 섹터들을 제공하여 응답할 수 있다. 그러한 실시예는 O/S 하위 트래핑 에이전트(2604) 및/또는 트리거된 이벤트 핸들러(2608)이 스토리지 장치(2606)의 동일한 섹터들 상에 항상 저장되는 것은 아닐 수 있는 동적으로 위치하는 파일들이나 데이터를 액세스하라는 요청들을 식별하게 할 수 있다. 예를 들어 마스터 파일 테이블, 운영체제 커널 파일들, 장치 드라이버들 및 안티 악성 소프트웨어 소프트웨어의 위치가 스토리지 장치(2606)의 동일한 섹터들(924) 상에 항상 있는 것은 아닐 수 있으며, 매핑 에이전트(2622)는 이러한 파일들이 저장되는 섹터들을 식별하는 데 사용될 수 있다. 일부 실시예들에서, 매핑 에이전트(2622)는 보호 파일이 저장되는 섹터들을 판단하기 위해 파일 시스템에 문의할 수 있다. 매핑 에이전트(2622)는 또한, 보호 파일이 저장되는 스토리지 장치(2606) 상의 섹터들(924)을 식별하기 위해 디스크 매핑 비트맵(2628)을 사용할 수 있다. 디스크 매핑 비트맵(2628)은 도 23의 디스크 매핑 비트맵(2301)의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 디스크 매핑 비트맵(2628)은 예컨대 각각의 보호 파일이 저장된 스토리지 장치의 섹터 또는 섹터들을 포함하여, 다양한 보호 파일들과 관련된 정보를 포함할 수 있다. 보호 파일이 업데이트되는 경우, 디스크 매핑 비트맵(2628)으로부터의 정보 역시 업데이트될 수 있다. 이러한 방식으로, 매핑 에이전트(2622)가 보호 파일을 파일 시스템 상의 그것의 정황적 존재로부터 해당 파일이 상주하는 스토리지 장치(2606)의 섹터들(924)로 매핑하라는 요청을 수신할 때, 매핑 에이전트(2622)는 보호 파일에 대응하는 섹터들(924)을 식별하기 위해 디스크 매핑 비트맵(2628)을 조회할 수 있다.

[0337] 트리거된 이벤트 핸들러(2608)가 서로 통신 가능하게 연결된 하나 이상의 이벤트 핸들러들이나 보안 에이전트들에 의해 구현될 수 있다. 트리거된 이벤트 핸들러(2608) 및 O/S 하위 트래핑 에이전트(2604)는 동일한 보안 에이전트 안에서 구현될 수 있다. 일 실시예에서 트리거된 이벤트 핸들러(2608)는 O/S 하위 트래핑 에이전트(2604)와 동일한 우선순위 링에서 동작할 수 있다. 다른 실시예에서 트리거된 이벤트 핸들러(2608)는 O/S 내 보안 에이전트(2618)의 일부로서 구현될 수 있고, 운영체제(2612), 드라이버(2611) 또는 애플리케이션(2610)과 동일한 우선순위 링에서 동작할 수 있다. 또 다른 실시예에서, 트리거된 이벤트 핸들러(2608)는 둘 이상의 트

리거된 이벤트 핸들러들에 의해 구현될 수 있으며, 이때 적어도 한 트리거된 이벤트 핸들러는 O/S 하위 보안 에이전트(2616)와 동일한 우선순위 링에서 동작하며 적어도 하나의 트리거된 이벤트 핸들러는 운영체제(2612), 드라이버(2611) 또는 애플리케이션(2610)의 우선순위 링에서 동작한다. O/S 하위 트래핑 에이전트(2604)의 우선순위 링에서 동작함으로써, 트리거된 이벤트 핸들러(2608)도 마찬가지로, 에이전트 자체를 감염시키는 "Ring0" 나 "Ring3" 악성 소프트웨어의 문제를 피할 수 있다. 그러나 운영체제(2612), 드라이버(2611), 또는 애플리케이션(2610)과 함께 "Ring0"나 "Ring3"에서 실행되는 트리거된 이벤트 핸들러(2608)는 "Ring -1" 에이전트들의 관점에서 이용 불가능할 수 있는 스토리지 장치(2606)에 대해 시도된 액세스에 관한 정황 정보를 제공할 수 있다.

[0338] 트리거된 이벤트 핸들러(2608)는 O/S 하위 트래핑 에이전트(2604)로부터 트리거된 이벤트들을 수신하여 처리하도록 구성될 수 있다. 트리거된 이벤트 핸들러(2608)는 또한, O/S 하위 보안 에이전트(2616) 및/또는 O/S 하위 트래핑 에이전트(2604)로 보안 규칙들(2614)을 제공하도록 구성될 수 있다. 트리거된 이벤트들은 O/S 하위 트래핑 에이전트(2604)에 의해 트래핑되었던 스토리지 장치(2606)를 액세스하라는 요청에 대한 정보를 포함할 수 있다. 트리거된 이벤트 핸들러(2608)는 스토리지 장치(2606)의 보호 영역들을 액세스하려는 시도들을 식별하고 적합한 반응을 결정하기 위해, 트리거된 이벤트와 관련된 정황 정보와 함께 하나 이상의 보안 규칙들(2614) 또는 보호 서버(2602)를 활용하도록 구성될 수 있다. 예를 들어 트리거된 이벤트 핸들러(2608)는 보호 섹터들 및/또는 파일들과 같이 스토리지 장치(2606)의 보호 영역들을 액세스하고자 하는 시도들을 식별하기 위해 보안 규칙들(2614)을 이용할 수 있다. 트리거된 이벤트 핸들러(2608)는 보호 파일들을 액세스하라는 요청들을 식별하는 것을 돕는 매핑 에이전트(2622)를 이용할 수 있다. 예를 들어 트리거된 이벤트 핸들러(2608)는 매핑 에이전트(2622)로 보호 파일을 스토리지 장치(2606) 상의 대응 섹터들로 매핑하라는 요청을 보낼 수 있다. 매핑 에이전트(2622)는 보호 파일에 대응하는 섹터들로 응답할 수 있다. 트리거된 이벤트 핸들러(2608)는 파일에 대응하는 섹터들을 액세스하려는 시도들을 식별함으로써 보호 파일을 액세스하려는 시도들을 식별할 수 있다. 보호 섹터 및/또는 파일과 같은 보호 영역을 액세스하려는 시도를 식별한 후, 트리거된 이벤트 핸들러(2608)는 보호 영역을 액세스하려는 시도가 허가된 것인지 여부를 판단하기 위해 보안 규칙들(2614)을 조회하도록 구성될 수 있다. 트리거된 이벤트 핸들러(2608)는 또한 O/S 하위 보안 에이전트(2616)로 적합한 액션에 대한 결정을 제공하도록 구성될 수 있다. 예를 들어 트리거된 이벤트 핸들러(2608)는 트리거된 이벤트가 허용되어야 하는지 거부되어야 하는지 여부, 특정 데이터나 매체 표면이 파괴되어야 하는지 여부, 또는 데이터가 암호화되어야 하는지 여부를 O/S 하위 보안 에이전트(2616)로 알릴 수 있다.

[0339] O/S 하위 보안 에이전트(2616)은 단독으로, 혹은 트리거된 이벤트 핸들러(2608)이나 O/S 내 보안 에이전트(2618)와 같은 구성요소들과 함께, 스토리지 장치(2606)를 액세스하는 통상적인 안전한 방법들을 결정하도록 구성될 수 있다. 예를 들어 스토리지 장치(2620)의 섹터들에 대한 쓰기나 읽기들은 파일 I/O 드라이버들에 의한 호출을 통해 보통 이루어질 수 있다. 따라서 보호 섹터로의 트랩된 쓰기 시도가 O/S 하위 보안 에이전트(2616)가 해당 시도를 하기 위해 사용된 절차들이나 함수들을 검사함으로써 평가될 수 있다. 액세스한 섹터들에서 예상 동향으로부터의 일탈이 관찰되어 악성 소프트웨어에 대한 표시라고 평가될 수 있다. 만일, 예컨대 O/S 하위 보안 에이전트(2616)가 보호 섹터에 대한 어떤 쓰기 시도가 정상적 파일 I/O 함수들이나 드라이버들을 사용하지 않고 인터럽트 13에 대한 직접적인 호출을 통해 이루어졌다면 그러한 쓰기 시도는 의심스러운 것일 수 있다.

[0340] 백업 스토리지 장치(2620)가 스토리지 장치(2606) 상의 데이터를 백업하고 복구하는 데 사용될 수 있다. 예를 들어 O/S 하위 보안 에이전트(2616) 및 O/S 내 보안 에이전트(2618)는 스토리지 장치(2606)로부터의 데이터를 백업하고 다양한 상황에서 그 데이터를 복구하도록 구성될 수 있다. 보안 규칙들(2614)은 백업이 허가된 스토리지 장치(2606)의 특정 섹터들(924)을 특정할 수 있다. 스토리지 장치(2606)로부터의 데이터가 복구를 요할 때, 스토리지 장치(2606)의 적합한 섹터들이 백업 스토리지 장치(2620)의 대응 섹터들로부터의 데이터를 이용하여 기입될 수 있다. 필요 시, 복구 프로세스 중에 스토리지 장치(2606)로의 복수의 기입이 이용될 수 있다. 일부 실시예들에서 스토리지 장치(2606)로부터의 데이터가 오염되어 있거나 악성 소프트웨어에 감염되어 있다고 판단되면 그 데이터가 복구될 수 있다. 그러한 판단은 악성 소프트웨어의 존재를 검출하기 위해 스토리지 장치(2606)의 섹터들을 검색함으로써 수행될 수 있다. 스토리지 장치(2606)의 섹터들을 검색할 때, 악성 소프트웨어를 포함하거나 그와 관련된 것으로 알려진 데이터의 패턴들을 식별하기 위해 블랙리스트가 사용될 수 있다. 블랙리스트는 보안 규칙들(2614)에 의해 정의될 수 있다. 악성 소프트웨어와 관련된 것으로 알려진 데이터의 패턴이 발견되면, 감염된 섹터들이 백업 스토리지 장치(2620)로부터 복구될 수 있다. 일부 실시예들에서, 디스크 매핑 비트맵(2628)이 사용되어 다양한 보호 파일들이 악성 소프트웨어에 감염될 수 있는지 여부를 판단하도록 할 수 있다. 디스크 매핑 비트맵(2628)의 실시예들에 대한 설명은 예컨대 도 23의 디스크 매핑 비트맵

(2301)에 대한 논의들에서 찾아볼 수 있다. 디스크 매핑 비트맵(2628)은 스토리지 장치(2606) 상의 보호 파일의 위치를 특정할 수 있고, 또한 보호 파일에 대해 앞서 생성된 해시 값을 제공할 수 있다. 디스크 매핑 비트맵(2628)은 보호 파일의 위치를 식별하기 위해 조회될 수 있고, 해시는 보호 파일의 콘텐츠를 사용하여 계산될 수 있으며, 계산된 해시는 디스크 매핑 비트맵(2628)으로부터 앞서 생성된 해시 값과 비교될 수 있다. 해시 값들이 매칭하지 않으면, 보호 파일이 악성 소프트웨어에 의해 변경되었을 가능성이 있으므로 그 파일은 백업 스토리지 장치(2620)로부터 복구된다. 어떤 실시예들에서 백업 스토리지 장치(2620)는 또한 스토리지 장치(2606) 상의 데이터를 복구하는데 사용되기 전에 악성 소프트웨어가 있는지 체크된다. 백업 스토리지 장치(2620)가 감염되어 있으면, 백업 스토리지 장치(2620)로부터의 백업 데이터는 사용되지 않을 수 있고/있거나 더 오래된 백업이 사용되거나 스토리지 장치(2606)를 액세스하라는 요청이 거부될 수 있다.

[0341] 백업 스토리지 장치(2620)로부터의 데이터는 악성 소프트웨어에 감염될 수 있는 운영체제(2612)의 파일 시스템 메커니즘을 이용하는 것을 피하기 위해 O/S 하위 보안 에이전트(2616)에 의해 스토리지 장치(2606)로 기입될 수 있다. 그러나, 백업 스토리지 장치(2620)로부터의 데이터로 스토리지 장치(2606)에 대한 데이터를 복구하기 위해 어떤 다른 보안 프로세스가 사용될 수 있다. 각각의 백업을 위한 메타데이터가 보유될 수 있으며, 개정 넘버, 백업이 생성된 날짜와 시간, 및 백업과 관련된 애플리케이션(2610)이나 다른 개체를 포함할 수 있다. 백업 스토리지 장치(2620)는 네트워크 상에서와 같이 스토리지 장치(2606)로부터 먼 곳에 위치될 수 있다. 예를 들어 백업 스토리지 장치(2620)는 보호 서버(2602)와 관련될 수 있다. 백업 스토리지 장치(2620)가 네트워크 상에 위치하는 경우, O/S 하위 보안 에이전트(2616)는 악성 소프트웨어에 감염될 수 있는 운영체제 커널 네트워크 장치 드라이버들을 이용하는 것을 피하기 위해 백업 스토리지 장치(2620)를 액세스하기 하는 데 대역 밖 네트워크 접속을 이용할 수 있다. 대역 밖 네트워크 접속은 일 실시예에서, 전자 장치(2601)의 네트워크 카드를 직접 액세스함으로써, HTTPS, iSCSI, NFS, 또는 CIFS 클라이언트의 사용이 백업 스토리지 장치(2620)를 액세스하게 할 수 있는 액티브 관리 기술(AMT)을 사용하여 구현될 수 있다.

[0342] 보호 서버(2602)는 네트워크 상에서 동작할 수 있으며 클라우드 컴퓨팅 방식을 구현할 수 있다. 보호 서버(2602)는 보안 규칙들(2614)을 저장하고, 보안 규칙들(2614) 및 다른 정보를 제공하기 위해 O/S 하위 보안 에이전트(2616), O/S 내 보안 에이전트(2618) 및/또는 트리거된 이벤트 핸들러(2608)과 같은 시스템(900)의 요소들과 통신하도록 구성될 수 있다. 보호 서버(2602)는 백업 스토리지 장치(2620)를 포함할 수 있다. 백업 스토리지 장치(2620)는 보안 규칙들(2614)를 저장하고/거나 스토리지 장치(2606)로부터의 데이터를 백업하기 위해 사용될 수 있다.

[0343] 보안 규칙들(2614)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(220, 222), 도 4의 보안 규칙들(420, 422, 434, 436, 438), 도 5의 보안 규칙들(518) 또는 도 7의 보안 규칙들(707, 721, 723)의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보안 규칙들(2614)의 실시예에 대한 설명은 이하의 도 27의 논의에서 발견될 수 있다.

[0344] 도 27은 무허가 액세스로부터 스토리지 장치를 보호하기 위한 시스템 또는 방법과 함께 사용할 보안 규칙들의 실시예이다. 보안 규칙들(2700)은 트래핑을 요하는 이벤트들 및 각각의 이벤트에 대한 적절한 응답을 특정하기 위한 어떤 적절한 규칙들, 로직, 명령들, 명령어들, 플래그들, 또는 다른 메커니즘들을 포함할 수 있다. 예를 들어 보안 규칙들(2700)은 트래핑을 요하는 이벤트들을 식별하고 각각의 이벤트에 대한 적합한 응답을 결정하기 위해, 도 26으로부터의 O/S 하위 보안 에이전트(2616), O/S 내 보안 에이전트(2618) 및/또는 트리거된 이벤트 핸들러(2608)에 의해 이용될 수 있다. 보안 규칙들(2700)은 스토리지 장치를 액세스하라는 모든 요청들이 트래핑될 것을 요구할 수 있고, 읽기, 쓰기 및/또는 실행 요청들과 같이 요청의 특정 타입들에 대한 트래핑만을 요구할 수 있다. 보안 규칙들(2700)은 보호를 요하는 스토리지 장치의 특정 섹터들이나 파일들과 같이, 스토리지 장치의 보호 영역들(2702)을 특정하는 규칙들을 더 포함할 수 있다. 각각의 보호 영역(2702)에 대해, 보안 규칙들(2700)은 각각의 보호 영역(2702)을 액세스하는 것이 허가되거나 허가되지 않을 수 있는 운영체제, 애플리케이션들 또는 드라이버들과 같은 요청하는 개체들(2704)을 특정할 수 있다. 보안 규칙들(2700)은 또한 보호 영역(2702)을 액세스하도록 허가된 각각의 개체(2704)에 대해, 보호 영역(2702)에 대한 읽기(2706a), 쓰기(2706b) 또는 실행(2706c)과 같은 액세스 허가 사항들(2706)을 특정할 수 있다.

[0345] 어떤 보안 규칙들(2700)은 애플리케이션과 무관하거나(application agnostic) 애플리케이션에 특정한 것일 수 있다. 애플리케이션에 무관한 규칙들은 스토리지 장치의 보호 영역(2702)에 대한 액세스를 요청하는 애플리케이션과 무관하게 적용된다. 애플리케이션 특정 규칙들은 요청을 개시한 애플리케이션에 따라 보호 영역(2702)에 대한 액세스를 허가하거나 금지할 수 있다. 규칙(2710a)은 어떤 개체에 의한 마스터 부트 레코드로의 기입 요청이 거부되어야 한다고 특정하는 애플리케이션에 무관한 규칙의 예이다. 규칙(2710b)은 보안 에이전트가 스

토리지 장치 상의 자신의 이미지에 대해 기입하는 것을 허용하면서 어떤 다른 개체가 스토리지 장치 상의 그 보안 에이전트의 이미지에 대해 기입하는 것은 금지하는 애플리케이션 특정 규칙의 예이다. 규칙 2710c 및 2710d 역시 애플리케이션 특정 규칙들의 예들이다. 규칙 2710c는 인터넷 익스플로러 애플리케이션의 코드 페이지들이 어느 개체에 의해서도 기입될 수 없다는 것을 특정한다. 규칙 2710d는 인터넷 익스플로러 애플리케이션의 데이터 페이지들에 대한 쓰기 요청이 그 인터넷 익스플로러 애플리케이션으로부터 나온 경우에는 그 요청이 허용될 수 있지만, 인터넷 익스플로러 어플리케이션의 데이터 페이지들에 대해 어떤 다른 개체로부터의 쓰기 요청은 거부될 것임을 특정한다.

[0346] 보안 규칙들(2700)은 애플리케이션들이나 운영체제에 의해, 그 애플리케이션들 및 운영체제가 데이터나 코드 페이지들과 같이 그들 각자의 정보에 대한 필수적인 보호를 특정할 수 있다고 정의될 수 있다. 보안 규칙들(2700)은 또한 관리자에 의해 설정되고 도 26에서의 보호 서버(2602)와 같이 원격으로 저장될 수 있다. 보안 규칙들(2700)은 원격 위치로부터 검색 및/또는 업데이트될 수 있다.

[0347] 일부 실시예들에서, 보안 규칙들(2700)은 스토리지 장치에 대한 액세스를 허용하기 전에, 도 26에서의 보호 서버(2602)와 같은 보호 서버로의 네트워크 접속을 요할 수 있다. 보호 서버로의 접속이 이용 불가한 경우, 보안 규칙들(2700)은 스토리지 장치에 대한 액세스를 금지할 수 있고, 스토리지 장치의 데이터나 매체 표면을 파괴할 수 있다. 예를 들어 보안 규칙들(2700)은 도 26에서의 O/S 하위 보안 에이전트(2616)와 같은 보안 에이전트가 특정된 수의 날들 동안 보호 서버에 연결될 수 없으면, 그 보안 에이전트는 스토리지 장치의 보안이 위험한 상태에 있다고 추정할 수 있다. 그러한 실시예들에서, 스토리지 장치 상의 데이터가 물리적으로 훼손되었다고 하더라도 그 데이터는 보호된다.

[0348] 도 28은 무허가 액세스로부터 전자 장치의 스토리지 장치를 보호하기 위한 방법의 실시예이다. 단계 2805에서, O/S 하위 보안 에이전트, O/S 내 보안 에이전트, 트리거된 이벤트 핸들러 및 보호 서버의 식별 및 보안이 인증될 수 있다. 그러한 인증은 암호화 해싱이나 비밀 키들을 이용하여 각각의 구성요소에 대한 메모리 내 이미지들을 찾아 검증하는 것을 포함하는 어떤 적절한 방법을 이용하여 수행될 수 있다. 단계 2805가 완료될 때까지 소정 실시예들에서 다른 단계들의 동작은 보류될 수 있다.

[0349] 단계 2810에서 보안 규칙들이 얻어진다. 보안 규칙들은 O/S 하위 보안 에이전트, O/S 내 보안 에이전트 또는 트리거된 이벤트 핸들러에 의해 내부적으로 저장되거나, 예컨대 보호 서버 상에 원격으로 저장될 수 있다. 그러한 보안 규칙들은 단계들 2815-2860에서 결정을 내리는 데 사용될 수 있다. 단계 2815에서 보안 위협이 검출되었는지가 판단된다. 예를 들어 O/S 하위 보안 에이전트 및/또는 O/S 내 보안 에이전트가 전자 장치 상의 악성 소프트웨어를 식별하거나 전자 장치를 감염시키기 위한 악성 소프트웨어의 시도를 식별할 수 있다. 보안 위협이 검출되지 않았으면, 아무 액션도 취해지지 않을 수 있다. 보안 위협이 검출되었으면, 단계 2820에서 O/S 하위 보안 에이전트가 스토리지 장치에 대한 액세스를 트래핑하라고 지시 받는다. 어떤 실시예들에서, O/S 하위 보안 에이전트는 보안 위협이 검출되었는지 여부와 무관하게, 인증 프로세스 후 스토리지 장치에 대한 액세스를 트래핑하라고 지시 받는다.

[0350] 단계 2825에서, 스토리지 장치를 액세스하라는 요청이 트래핑된다. 그러한 트래핑은 전자 장치에서 실행되는 운영체제보다 낮은 우선순위 링에서 실행되는 소프트웨어에 의해 수행될 수 있다. 예를 들어 O/S 하위 보안 에이전트가 트래핑 기능을 수행할 수 있다. 단계 2830에서, 트래핑된 요청과 관련된 스토리지 장치의 섹터들이 악성 소프트웨어에 감염되었는지 여부가 판단된다. 그러한 판단은 악성 소프트웨어의 존재를 검출하기 위해 스토리지 장치의 섹터들을 검색함으로써 수행될 수 있다. 스토리지 장치의 섹터들을 검색할 때, 악성 소프트웨어를 포함하거나 그와 관련된 것으로 알려진 데이터의 패턴들을 식별하기 위해 블랙리스트가 사용될 수 있다. 악성 소프트웨어와 관련된 것으로 알려진 데이터의 패턴이 발견되면, 단계 2835에서 감염된 섹터들이 백업 스토리지 장치로부터 복구된다. 어떤 실시예들에서 백업 스토리지 장치는 또한 스토리지 장치를 복구하는데 사용되기 전에 악성 소프트웨어가 있는지 검색된다. 백업 스토리지 장치가 감염되어 있으면, 백업이 사용되지 않을 수 있고/있거나 더 오래된 백업이 사용되거나 스토리지 장치를 액세스하라는 요청이 거부될 수 있다.

[0351] 단계 2840에서 스토리지 장치의 보호 섹터에 대한 액세스가 요청되었는지 여부가 판단된다. 보호 섹터들은 보안 규칙들에 의해 정의된다. 보안 규칙들은 특정 섹터들이 보호될 것을 요구하거나, 동적으로 위치하는 특정 파일들 및/또는 데이터가 보호될 것을 요구할 수 있다. 예를 들어 보안 규칙들은 고정적이고 스토리지 장치의 최초 섹터(섹터 0)에 위치하는 마스터 부트 레코드의 보호를 요할 수 있다. 다른 예로서, 보안 규칙들은 또한 마스터 파일 테이블, 운영체제 커널 파일들, 장치 드라이버들, 또는 안티 악성 소프트웨어 소프트웨어의 보호를 요할 수 있다. 이러한 파일들은 동적 위치들을 가질 수 있어 항상 같은 섹터들 상에 저장되는 것은 아니다.

동적으로 저장된 파일이나 데이터가 보호를 요하면, 그 파일이나 데이터는 파일 시스템 상에서의 자신의 정황적 존재로부터 그 파일이나 데이터가 상주하는 스토리지 장치 상의 실제 섹터들로 매핑된다. 스토리지 장치를 액세스하라는 요청이 보호 섹터를 포함하지 않으면, 단계 2850에서 스토리지 장치를 액세스하라는 요청은 허용된다. 스토리지 장치를 액세스하라는 요청이 보호 섹터를 포함하면, 단계 2845에서 보호 섹터에 대한 액세스가 허가되는지 여부가 판단된다. 요청한 개체가 보호 파일을 액세스하도록 허가되는지 여부를 판단하기 위해, 스토리지 장치를 액세스하고자 시도된 요청과 관련된 정황 정보가 보안 규칙들과 연계하여 분석될 수 있다. 예를 들어 보안 규칙들은 운영체제, 특정 애플리케이션 또는 특정 장치 드라이버가 보호 섹터를 액세스하는 것이 허용될 수 있는지 없는지를 특정할 수 있다. 보안 규칙들은 또한, 보호 파일을 액세스하도록 허용되는 요청 개체에 대해 읽기, 쓰기 또는 실행과 같은 액세스 허가 사항들을 특정할 수도 있다.

[0352] 보호 섹터에 대한 액세스가 허가되면, 단계 2850에서 스토리지 장치를 액세스하라는 요청이 허용된다. 보호 섹터에 대한 액세스가 허가되지 않으면, 단계 2855에서 스토리지 장치를 액세스하라는 요청이 거부된다. 어떤 실시예들에서, 다른 교정 액션이 수행될 수 있다. 예를 들어 스토리지 장치 상의 데이터가 파괴되거나 암호화되거나, 스토리지 장치의 매체 표면이 파괴될 수 있다. 스토리지 장치를 액세스하라는 요청이 허가되지 않으면, 단계 2860에서 스토리지 장치를 액세스하려는 시도가 보호 서버에 보고된다. 그러한 보고는 어떤 관련된 악성 소프트웨어나 의심스러운 동향에 관한 정보를 포함할 수 있다.

[0353] 도 28로부터의 방법의 단계들은 필요 시, 지속적으로나 주기적으로나 요청에 따르거나 어떤 이벤트의 트리거 시, 스토리지 장치를 보호하기 위해 반복될 수 있다.

[0354] 도 29는 애플리케이션 및 입/출력 장치 간의 쓰기 액세스를 위한 입/출력 경로를 보호하는 시스템(2900)의 실시예이다. 시스템(2900)은 전자 장치(2904)의 애플리케이션 입력/출력(I/O) 경로들로의 악성 소프트웨어 공격들에 대해 보호되어야 하는 전자 장치(2904)를 포함할 수 있다. 전자 장치(2904)는 운영체제 하위 보안 에이전트(2916) I/O 장치(2926), 애플리케이션(2910), 운영체제(2912) 및 드라이버(2911)를 포함할 수 있다. 전자 장치(2904)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(401), 도 7의 전자 장치(701) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0355] O/S 하위 보안 에이전트(2916)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM 보안 에이전트(216) 또는 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442), PC 펌웨어 보안 에이전트(444) 또는 O/S 하위 에이전트(450), 도 5의 펌웨어 보안 에이전트(516), 도 7의 마이크로코드 보안 에이전트(708) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 하위 보안 에이전트(2916)는 악성 소프트웨어로부터 전자 장치(2904)의 애플리케이션 I/O 경로들을 보호하도록 구성될 수 있다. I/O 장치(2926)는 도 2의 장치(226), 도 4의 디스플레이(424) 또는 스토리지(426), 도 5의 입/출력 장치(502) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 애플리케이션(2910)은 도 1의 애플리케이션(110), 도 2의 애플리케이션(210), 도 4의 애플리케이션(410), 도 7의 애플리케이션 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 드라이버(2911)는 도 1의 드라이버(111), 도 2의 드라이버(211), 도 4의 드라이버(411), 도 7의 드라이버 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 운영체제(2912)는 도 1의 운영체제(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0356] 도 29의 화살표로 도시되고 이하의 도 30과 관련하여 기술되는 바와 같이, O/S 하위 보안 에이전트(2916)는 애플리케이션 I/O 경로를 통한 데이터의 전송에 대해 트래핑할 수 있다. 트래핑 시, O/S 하위 보안 에이전트(2916)는 I/O 쓰기 액세스와 관련하여 애플리케이션(2910)으로부터 I/O 장치(2926)로 전달될 콘텐츠를 인터셉트할 수 있다. O/S 하위 보안 에이전트(2916)는 인터셉트된 I/O 콘텐츠를 변경하고, 그 변경된 콘텐츠를 정상적 I/O 경로를 통해(가령, 운영체제(3112) 및 드라이버(3111)를 통해) 전송할 수 있다. 그 변형된 콘텐츠는 I/O 경로 데이터를 인터셉트할 수 있는 어떤 악성 소프트웨어가 실제 사용자 데이터 대신 더미 데이터를 인터셉트하게 하도록, "스푸핑" 혹은 "더미" 콘텐츠를 포함할 수 있다. O/S 하위 보안 에이전트(2916)는 변형된 I/O 콘텐츠가 I/O 장치(2926)를 위한 장치 드라이버에 도달할 때 그 변형된 I/O 콘텐츠를 인터셉트할 수 있고, 그것을 오리지널 콘텐츠로 대체함으로써, 악성 소프트웨어 공격으로부터 I/O 전송을 보호할 수 있다. 또한 O/S 하위 보안 에이전트(2916)는 규칙들(가령, 보안 규칙들(114, 220, 222, 438, 434, 436, 518, 707, 721 및/또는 723))에 기반하여, 정상적 I/O 경로를 통해 전송된 변형된 콘텐츠가 악성 소프트웨어 유사 양태(가령, 변형된

콘텐츠가 스니핑, 후킹 및/또는 공격 당했다는 것을 나타내는 양태)에 의해 영향을 받았는지 여부를 검출하고 악성 소프트웨어 유사 양태가 검출된 경우 교정 액션을 취할 수 있다. 일부 실시예들에서 O/S 하위 보안 에이전트(2916)는 보호 서버(202)로 악성 소프트웨어 유사 양태의 발생에 관한 정보를 전송할 수 있다. 예를 들어 O/S 하위 보안 에이전트(2916)는 보호 서버(202)가 악성 소프트웨어 유사 양태를 일으키는 악성 소프트웨어를 식별하게 도울 수 있고/있거나 전자 장치(2904) 및/또는 다른 전자 장치들에 대한 추가 감염을 막을 수 있는 법의학적 정보를 보호 서버(202)로 전송할 수 있다. 그러한 법의학적 정보는 제한 없이, 해당 양태가 발생했던 전자 장치의 아이디, 악성 소프트웨어 유사 양태를 식별하는 O/S 하위 보안 에이전트, 악성 소프트웨어 유사 양태가 일어난 장치(2926) 및/또는 애플리케이션 I/O 경로, O/S 하위 보안 에이전트에 의해 I/O 경로 안으로 놓여지는 수정된 콘텐츠 및/또는 (잠정 악성 소프트웨어에 의해 이루어진 변형된 데이터에 대한 변형들을 가리키는) 인터셉트된 변형 데이터를 포함할 수 있다.

[0357] 애플리케이션(2910) 및 입/출력 장치(2926) 사이의 경로가 소정 개수의 요소들로 보여지고 있으나, 그 경로는 애플리케이션(2910) 및 입/출력 장치(2926) 사이의 입력 또는 출력 경로를 구현하는 데 필요한 만큼 많은 구성요소들을 포함할 수 있다. 예를 들어 운영체제(2912) 및 드라이버(111)는 애플리케이션(2910) 및 입/출력 장치(2926) 사이에 정보를 전달하기 위한 여러 하위 구성요소들을 포함할 수 있다. 운영체제(2912) 및 드라이버(111)와 그들의 하위 구성요소들은 시스템이나 드라이버 정의 함수들을 이용하여 서로를 호출하도록 구성될 수 있다. O/S 하위 보안 에이전트(2916)는 애플리케이션(2910) 및 입/출력 장치(2926) 사이의 입/출력 경로를 따라 어떤 전송들이나 동작들을 트래핑하도록 구성될 수 있다. 예를 들어 윈도우즈 환경에서 장치(2926) 안에 이미지를 두기 위해, 애플리케이션(2910)은 BitBlt 함수를 이용하여 gdi32.dll을 호출하도록 구성될 수 있고, 그것은 NtGDIBitBlt 함수를 이용하여 ndtll.dll을 호출하도록 구성될 수 있고, 그것은 NtGDIBitBlt 함수를 이용하여 win32k.sys를 호출하도록 구성될 수 있으며, 그것은 장치(2926)에 의해 구현되는 디스플레이로의 입/출력을 처리할 수 있는 그래픽 I/O 드라이버를 호출하도록 구성될 수 있다. O/S 하위 보안 에이전트(2916)는 예컨대 그러한 함수들의 코드 섹션들을 포함하는 메모리 위치들의 실행을 트래핑함으로써 그러한 함수 호출들의 실행을 트래핑하도록 구성될 수 있다. 메모리 위치들은 예컨대 가상 메모리 페이지나 물리적 메모리의 어드레스 범위를 포함할 수 있다.

[0358] O/S 하위 보안 에이전트(2916)는 애플리케이션(2910) 및 장치(2926) 사이의 경로를 따라 명령들이나 정보의 전송을 위해 그러한 함수들의 호출자들을 결정하고, 그들이 허가된 개체에 의해 실행되었는지 여부를 판단하도록 구성될 수 있다. 예를 들어 드라이버(2911)의 함수는 2911의 함수를 액세스하기 위해 시스템에 의해 제공되는 방법들(운영체제(2912) 안의 함수들 같은 것)을 이용하는 대신, 악성 프로세스에 의해 직접 호출될 수 있다. O/S 하위 보안 에이전트(2916)는 드라이버(2911)의 함수 실행을 트래핑하고, 액세스가 발생했던 메모리 어드레스에 기반하여 예컨대 애플리케이션(2910)이 드라이버(2911)를 직접 호출했고 그 호출이 운영체제(2912) 안의 허가된 개체로부터 발생하지 않았다고 판단하도록 구성될 수 있다. 그러한 액세스는 운영체제(2912) 내의 보안 대처들을 피하여 이루어졌을 수 있다. O/S 하위 보안 에이전트(2916)는 그러한 액세스가 악성 소프트웨어를 가리킨다고 판단하여, 시도된 액세스를 거부하도록 구성될 수 있다.

[0359] 또한 O/S 하위 보안 에이전트(2916)는 입/출력 버퍼들에 대응하는 메모리 위치들에 대해 시도되는 정보의 읽기 또는 쓰기를 트래핑함으로써, 애플리케이션(2910) 및 장치(2926) 사이의 정보 전송을 트래핑하도록 구성될 수 있다. 예를 들어 운영체제(2912)는 I/O 버퍼에 정보를 기입하고, 그 버퍼 안에서 장치(2926)로 보낼 정보를 찾으러 가도록 드라이버(2911)의 함수를 호출할 수 있다. 정보의 크기 때문에 정보를 직접 전달하는 대신 그러한 버퍼들이 파라미터들로서 사용될 수 있다. 그에 따라 O/S 하위 보안 에이전트(2916)는 예컨대 I/O 버퍼의 가상 메모리 페이지나 물리적 어드레스로의 읽기나 쓰기 액세스를 트래핑하도록 구성될 수 있다. O/S 하위 보안 에이전트(2916)는 개체가 I/O 버퍼를 읽거나 쓰도록 허가되는지 여부를 판단하기 위해 I/O 버퍼를 액세스하는 개체의 아이디를 판단하도록 구성될 수 있다. 예를 들어 O/S 하위 보안 에이전트(2916)는 키보드 데이터에 대해 버퍼로 시도된 액세스를 트래핑하도록 구성될 수 있다. 애플리케이션(2910)이 버퍼로부터 정보를 바로(즉, 운영체제(2912)를 통한 정상적 연쇄 호출의 범위 밖에서) 읽고자 시도하는 경우, O/S 하위 보안 에이전트(2916)는 버퍼 내 키보드 데이터에 대해 시도되는 직접 액세스가 키로거(keylogger)와 같은 악성 소프트웨어를 가리키기 때문에 그 액세스를 거부하도록 구성될 수 있다. 다른 예에서, 디스플레이 데이터를 위한 버퍼가 스크린 캡처 악성 소프트웨어를 막기 위해 액세스로부터 보호될 수 있다. 또 다른 예에서, 네트워크 출력을 위한 버퍼가 도스(DOS) 공격 발생이나 패킷 변경을 막기 위해 액세스로부터 보호될 수 있다.

[0360] 따라서 일부 실시예들에서, O/S 하위 보안 에이전트(2916)는 버퍼를 읽거나 쓰기가 허가된 것으로 알려지지 않은 I/O 버퍼들을 액세스하는 개체들에 대한 모든 액세스를 차단하도록 구성될 수 있다. 그러한 실시예들에서,

악성 소프트웨어 상태가 알려지지 않은 애플리케이션(2910)이나 개체는 그 개체가 앞서 악성 소프트웨어라고 식별되지 않았더라도 차단될 수 있고, 예컨대 악성 서명이 있는지 개체를 검색할 수 있다. 다른 실시예들에서 O/S 하위 보안 에이전트(2916)는 알려진 연쇄 호출 안에서 해당 버퍼 바로 아래나 위에 있는 드라이버들, 인터페이스들, 애플리케이션들 또는 다른 개체들에 대해 버퍼들의 액세스만을 허용하도록 구성될 수 있다. 마찬가지로, O/S 하위 보안 에이전트(2916)는 알려진 연쇄 호출 안에서 해당 버퍼 바로 아래나 위에 있는 드라이버들, 인터페이스들, 애플리케이션들 또는 다른 개체들에 대해 드라이버(2911)나 운영체제(2912)의 함수들에 대한 액세스만을 허용하도록 구성될 수 있다. 그렇게 알려진 연쇄 호출은 어떤 개체들이 애플리케이션(2910) 및 장치(2926) 사이의 경로를 따라 서로를 호출하는지를 이해하고, 프로파일링하며 벤치마킹하기 위해 알려진 안전 시스템의 통상적 동작을 관찰함으로써 정의될 수 있다. 그러한 알려진 안전한 동작들의 특성화가 O/S 하위 보안 에이전트(2916)에서 액세스 가능한 보안 규칙들 안에서 실시될 수 있다. 그러한 알려진 연쇄관계 밖에서 드라이버(2911)나 운영체제(2912)와 같은 드라이버들의 구성요소들에 대한 어떤 호출들이나 I/O 버퍼들에 대한 호출들은 거부될 수 있다.

[0361] O/S 하위 보안 에이전트(2916)는 애플리케이션(2910) 및 장치(2926) 사이의 경로 내에서의 호출을 트래핑하고, 전달될 데이터를 읽고, 데이터를 암호화하고, 데이터를 경로 안에 재삽입하고, 동작이 진행될 수 있게 할 수 있다. 일 실시예에서 장치(2926)는 그러한 데이터를 해독하도록 구성된 펌웨어 보안 에이전트를 포함할 수 있다. 그러한 펌웨어 보안 에이전트 및 O/S 하위 보안 에이전트(2916)는 그러한 암호화를 조율하도록 통신 가능하게 연결될 수 있고/있거나 각각이 그러한 암호화를 조율할 유사한 보안 규칙들을 가질 수 있다. 반대로, O/S 하위 보안 에이전트(2916)는 경로 안의 호출을 트래핑하고, 장치로부터 나오는 데이터를 해독하고, 그 데이터를 경로에 재삽입하고 동작이 진행될 수 있게 하도록 구성될 수 있다. 다른 실시예에서, O/S 하위 보안 에이전트(2916)는 경로의 한층 아래나 위의 호출을 트래핑하고, 전달될 데이터를 판독하고, 데이터를 해독하고, 그 데이터를 경로에 재삽입하고 동작이 진행될 수 있게 하도록 구성될 수 있다.

[0362] 또한 O/S 하위 보안 에이전트(2916)는 애플리케이션(2910) 및 장치(2926) 사이의 경로를 따라 전달될 데이터를 검사하고 악성 소프트웨어 표시가 있는지 데이터를 검색하도록 구성될 수 있다. O/S 하위 보안 에이전트(2916)는 경로 안에서 개체들 사이에서 전달되는 데이터를 제어하거나 전달된 데이터(더미 데이터 같은 것)를 파라미터들로서 대체하도록 구성될 수 있다.

[0363] 도 30은 애플리케이션 및 입/출력 장치 간의 쓰기 액세스를 위한 입/출력 경로를 보호하는 방법(3000)의 실시예이다. 단계 3002에서 O/S 하위 보안 에이전트는 애플리케이션 I/O 경로가 악성 소프트웨어 공격에 취약한지를 판단할 수 있다. 여기 개시된 애플리케이션 I/O 경로의 보호를 위한 시스템들과 방법들은 상당한 프로세서, 메모리 및/또는 다른 자원들을 소비하기 때문에, 애플리케이션 I/O 경로가 악성 소프트웨어 공격들을 당하기가 특히 쉬울 때에만 그러한 시스템들과 방법들을 이용하는 것이 바람직할 수 있다. 애플리케이션 I/O 경로는 애플리케이션이나 운영체제가 잠정적으로 민감한 정보가 전송될 수 있는 I/O 동작을 수행하고 있을 때 악성 소프트웨어 공격에 취약할 수 있다. 예를 들어 O/S 하위 보안 에이전트는 애플리케이션이 애플리케이션 I/O 경로 상에서 금융 데이터, 기업 인사 데이터, 계좌 번호, 유저네임, 패스워드, 사회보장번호(주민등록번호) 및/또는 전자 장치 사용자의 다른 식별 데이터와 같은 민감한 정보를 노출할 수 있는 은행이나 기타 금융 웹사이트를 액세스하고 있는 경우에, 애플리케이션 I/O 경로가 악성 소프트웨어 경로에 취약하다고 판단할 수 있다.

[0364] 단계 3003에서 애플리케이션 I/O 경로가 취약하다고 판단된 경우, 방법(3000)은 단계 3005로 진행할 수 있다. 그렇지 않으면 방법(3000)은 단계 3002로 돌아갈 수 있고, 애플리케이션 I/O 경로 보호는 애플리케이션 I/O 경로가 취약하다고 판단될 때까지 사용되지 않을 수 있다.

[0365] 단계 3005에서 O/S 하위 보안 에이전트는 애플리케이션으로부터 장치(가령, 디스플레이, 디스크 드라이브, 키보드 등)으로의 I/O 쓰기 액세스에 대해 트래핑할 수 있다. 예를 들어 I/O 쓰기 액세스가 윈도우즈 운영체제 안에서 애플리케이션으로부터 디스플레이 장치로의 데이터 전송을 포함하는 경우, O/S 하위 보안 에이전트는 비트-블록 전송 동작(가령, BitBit)에 대한 애플리케이션의 호출이나 디스플레이 I/O 함수들(가령, gdi32.dll, ntdll.dll 등)의 라이브러리에 대한 호출의 실행에 대해 트래핑할 수 있다. I/O 쓰기 또는 읽기 액세스는 최종 장치로 도달하기 위해 드라이버들 및 드라이버들의 함수들에 대한, 그리고 그들 사이의 일련의 호출들 혹은 연쇄 호출들을 포함할 수 있다. 예를 들어 윈도우즈에서, 애플리케이션은 BitBit 함수를 이용하여 gdi32를 호출할 수 있고, 그것은 NtGDIBitBlt 함수를 이용하여 ntdll.dl를 호출할 수 있고, 그것은 NtGDIBitBlt를 이용하여 win32k.sys를 호출할 수 있고, 그것이 디스플레이 장치를 액세스할 수 있는 그래픽 I/O 드라이버를 호출할 수 있다.

[0366] 단계 3010에서 O/S 하위 보안 에이전트는 I/O 동작의 콘텐츠(가령, 디스플레이 장치 상에 디스플레이될 이미지, 디스크 드라이브에 기입될 데이터 등)를 인터셉트할 수 있다.

[0367] 단계 3015에서, O/S 하위 보안 에이전트는 I/O 콘텐츠를 변경할 수 있다. 예를 들어 콘텐츠는 애플리케이션 I/O 경로 공격을 시도하는 악성 소프트웨어가 원래의 콘텐츠를 이루는 민감한 정보 대신 변경된 콘텐츠만을 액세스할 수 있도록, "스푸핑" 또는 "더미" 콘텐츠로 변경될 수 있다. O/S 하위 보안 에이전트는 어떤 적절한 방식으로 I/O 콘텐츠를 변경할 수 있다. 예를 들어 디스플레이 장치에 디스플레이될 이미지를 대체하기 위해, 변경된 콘텐츠는 오리지널 콘텐츠 대신 어떤 파라미터로서 비트-블록 전송 동작에 전달될 수 있다. 특정 예에서, O/S 하위 보안 에이전트는 민감한 파일이나 이메일의 텍스트 콘텐츠를 미리 결정된 더미 콘텐츠로 대체할 수 있다.

[0368] 단계 3020에서 O/S 하위 보안 에이전트는 애플리케이션 I/O 경로를 통해, 애플리케이션이 실행되는 운영체제 및 운영체제 및 장치 사이의 드라이버들에 의한 동작을 포함하는 보통의 동작에 대해 변경된 콘텐츠를 전송할 수 있다. 이 단계 중에, 애플리케이션 I/O 경로에 영향을 미치는 악성 소프트웨어가 I/O 콘텐츠를 도용하고자 시도할 수 있다. 그러나, 도용된 데이터는 O/S 하위 보안 에이전트에 의해 삽입된 변형된 더미 콘텐츠일 수 있으므로, 오리지널 콘텐츠를 도용으로부터 보호할 수 있다.

[0369] 단계 3025에서 O/S 하위 보안 에이전트는 변형된 콘텐츠가 I/O 장치에 도달할 때(가령, I/O 장치를 가진 전자 장치나 장치의 통신 포트들에서) 그 변형된 콘텐츠를 인터셉트할 수 있다. 단계 3030에서, O/S 하위 보안 에이전트는 그 변형된 콘텐츠를 오리지널 콘텐츠로 대체할 수 있다. 예를 들어, I/O 쓰기 액세스가 윈도우즈 시스템 내에서 어떤 애플리케이션으로부터 디스플레이 장치로의 데이터 전송을 포함하는 경우, 이미지 대체는 디스플레이 장치를 가진 전자 장치의 I/O 포트들의 후킹, 그래픽 I/O 드라이버의 메모리 후킹, 또는 그래픽 I/O 드라이버로부터의 디스플레이 명령의 실행에 대한 후킹이나 트리거링에 의해 구현될 수 있다. 그에 따라 오리지널 콘텐츠는 애플리케이션과 장치 사이에서 대역 밖 전송되어, 전통적 애플리케이션 I/O 경로의 콘텐츠를 도용하고자 시도하는 악성 소프트웨어로부터 계속 보호될 수 있다.

[0370] 단계 3035에서, O/S 하위 보안 에이전트는 변형된 콘텐츠가 악성 소프트웨어 유사 양태에 의해 영향을 받았는지 여부를 판단할 수 있다. 예를 들어 규칙들(가령, 보안 규칙들(114, 220, 222, 438, 434, 436, 518, 707, 721 및/또는 723)에 기반하여, O/S 하위 보안 에이전트는 인터셉트된 변형된 콘텐츠가 악성 소프트웨어에 의해 영향을 받았음(가령, 변형된 데이터가 애플리케이션 I/O 경로에서 자체 변형된 것처럼 애플리케이션 I/O 경로를 지났을 경우)을 나타내는 특징을 가지는지 여부를 판단할 수 있다. 또한, O/S 하위 보안 에이전트가 변형된 콘텐츠가 악성 소프트웨어 유사 양태에 의해 영향을 받았다고 판단하는 경우, O/S 하위 보안 에이전트는 교정 액션(가령, 제거 액션, 격리 및/또는 악성 소프트웨어 중립화)을 취할 수 있다. 또한 일부 실시예들에서, O/S 하위 보안 에이전트는 악성 소프트웨어 유사 양태 발생에 관한 정보(가령, 법의학적 정보)를 보호 서버로 전송할 수 있다.

[0371] 도 31은 애플리케이션 및 입/출력 장치 간의 읽기 액세스를 위한 입/출력 경로를 보호하는 시스템(3100)의 실시예이다. 시스템(3100)은 전자 장치(3104)의 애플리케이션 입력/출력(I/O) 경로들로의 악성 소프트웨어 공격들에 대해 보호되어야 하는 전자 장치(3104)를 포함할 수 있다. 전자 장치(3104)는 운영체제 하위 보안 에이전트(3116) I/O 장치(3126), 애플리케이션(3110), 운영체제(3112) 및 드라이버(3111)를 포함할 수 있다. 시스템(3100)은 전자 장치(3104)의 애플리케이션 입력/출력(I/O) 경로들로의 악성 소프트웨어 공격들에 대해 보호되어야 하는 전자 장치(3104)를 포함할 수 있다. 전자 장치(3104)는 운영체제 하위 보안 에이전트(3116) I/O 장치(3126), 애플리케이션(3110), 운영체제(3112) 및 드라이버(3111)를 포함할 수 있다. 전자 장치(3104)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(401), 도 7의 전자 장치(701), 도 29의 전자 장치(2904) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0372] O/S 하위 보안 에이전트(3116)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM 보안 에이전트(216) 또는 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442), PC 펌웨어 보안 에이전트(444) 또는 O/S 하위 에이전트(450), 도 5의 펌웨어 보안 에이전트(516), 도 7의 마이크로코드 보안 에이전트(708), 도 29의 O/S 하위 보안 에이전트(2916) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 하위 보안 에이전트(3116)는 악성 소프트웨어로부터 전자 장치(3104)의 애플리케이션 I/O 경로들을 보호하도록 구성될 수 있다. I/O 장치(3126)는 도 2의 장치(226), 도 4의 디스플레이(424) 또는 스토리지(426), 도 5의 입/출력 장치(502), 도 29의 I/O 장치(2926) 및/또는 이들의 어떤 조합의

기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 애플리케이션(3110)은 도 1의 애플리케이션(110), 도 2의 애플리케이션(210), 도 4의 애플리케이션(410), 도 7의 애플리케이션(709), 도 29의 애플리케이션(2910) 및/또는 이들의 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 드라이버(3111)은 도 1의 드라이버(111), 도 2의 드라이버(211), 도 4의 드라이버(411), 도 7의 드라이버(711), 도 29의 드라이버(2911) 및/또는 이들의 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 운영체제(3112)는 도 1의 운영체제(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 29의 운영체제(2912) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0373] O/S 하위 보안 에이전트(3116)는 악성 소프트웨어로부터 전자 장치(3104)의 애플리케이션 I/O 경로들을 보호하도록 구성될 수 있다. 도 31의 화살표로 도시되고 이하의 도 32와 관련하여 기술되는 바와 같이, O/S 하위 보안 에이전트(3116)는 애플리케이션 I/O 경로를 통한 데이터의 전송에 대해 트래핑할 수 있다. 트래핑 시, O/S 하위 보안 에이전트(3116)는 I/O 읽기 액세스와 관련하여 I/O 장치(3126)으로부터 애플리케이션(3110)으로 전달될 콘텐츠를 인터셉트할 수 있다. O/S 하위 보안 에이전트(3116)는 인터셉트된 I/O 콘텐츠를 변경하고, 그 변경된 콘텐츠를 정상적 I/O 경로를 통해(가령, 운영체제(3112) 및 드라이버(3111)를 통해) 전송할 수 있다. 그 변형된 콘텐츠는 I/O 경로 데이터를 인터셉트할 수 있는 어떤 악성 소프트웨어가 실제 사용자 데이터 대신 더미 데이터를 인터셉트하게 하도록, "스푸핑" 혹은 "더미" 콘텐츠를 포함할 수 있다. O/S 하위 보안 에이전트(3116)는 변형된 I/O 콘텐츠가 I/O 장치(3126)를 위한 장치 드라이버에 도달할 때 그 변형된 I/O 콘텐츠를 인터셉트할 수 있고, 그것을 오리지널 콘텐츠로 대체함으로써, 악성 소프트웨어 공격으로부터 I/O 전송을 보호할 수 있다. 또한 O/S 하위 보안 에이전트(3116)는 규칙들(가령, 보안 규칙들(114, 220, 222, 438, 434, 436, 518, 707, 721 및/또는 723))에 기반하여, 정상적 I/O 경로를 통해 전송된 변형된 콘텐츠가 악성 소프트웨어 유사 양태(가령, 양태가 변형된 콘텐츠가 스니핑, 후킹 및/또는 공격 당했다는 것을 나타냈음)에 의해 영향을 받았는지 여부를 검출하고 악성 소프트웨어 유사 양태가 검출된 경우 교정 액션을 취할 수 있다. 일부 실시예들에서 O/S 하위 보안 에이전트(3116)는 보호 서버(202)로 악성 소프트웨어 유사 양태의 발생에 관한 정보를 전송할 수 있다. 예를 들어 O/S 하위 보안 에이전트(3116)는 보호 서버(202)가 악성 소프트웨어 유사 양태를 일으키는 악성 소프트웨어를 식별하게 도울 수 있고/있거나 전자 장치(3104) 및/또는 다른 전자 장치들에 대한 추가 감염을 막을 수 있는 법의학적 정보를 보호 서버(202)로 전송할 수 있다. 그러한 법의학적 정보는 제한 없이, 해당 양태가 발생했던 전자 장치의 아이디, 악성 소프트웨어 유사 양태를 식별하는 O/S 하위 보안 에이전트, 악성 소프트웨어 유사 양태가 일어난 장치(3126) 및/또는 애플리케이션 I/O 경로, O/S 하위 보안 에이전트에 의해 I/O 경로 안으로 놓여지는 수정된 콘텐츠 및/또는 (잠정 악성 소프트웨어에 의해 이루어진 변형된 데이터에 대한 변형들을 가리키는) 인터셉트된 변형 데이터를 포함할 수 있다.

[0374] 도 32는 애플리케이션 및 입/출력 장치 간의 읽기 액세스를 위한 입/출력 경로를 보호하는 방법(3200)의 실시예이다. 단계 3202에서 O/S 하위 보안 에이전트는 애플리케이션 I/O 경로가 악성 소프트웨어 공격에 취약한지를 판단할 수 있다. 단계(3202)는 방법(3000)의 단계(3002)와 유사할 수 있다. 단계 3203에서 애플리케이션 I/O 경로가 취약하다고 판단된 경우, 방법(3200)은 단계 3205로 진행할 수 있다. 그렇지 않으면 방법(3200)은 단계 3202로 돌아갈 수 있고, 애플리케이션 I/O 경로 보호는 애플리케이션 I/O 경로가 취약하다고 판단될 때까지 사용되지 않을 수 있다. 단계(3203)는 방법(3000)의 단계(3003)와 유사할 수 있다.

[0375] 단계 3205에서 O/S 하위 보안 에이전트는 장치로부터 애플리케이션(가령, 디스플레이, 디스크 드라이브, 키보드 등)으로의 I/O 읽기 액세스에 대해 트래핑할 수 있다. 단계 3210에서 O/S 하위 보안 에이전트는 I/O 동작의 콘텐츠(가령, 키보드로부터 수신된 키 스트로크들, 디스크 드라이브로부터 판독될 데이터 등)를 인터셉트할 수 있다.

[0376] 단계 3215에서, O/S 하위 보안 에이전트는 I/O 콘텐츠를 변경할 수 있다. 예를 들어 콘텐츠는 애플리케이션 I/O 경로 공격을 시도하는 악성 소프트웨어가 원래의 콘텐츠를 이루는 민감한 정보 대신 변경된 콘텐츠만을 액세스할 수 있도록, "스푸핑" 또는 "더미" 콘텐츠로 변경될 수 있다. O/S 하위 보안 에이전트는 어떤 적절한 방식으로 I/O 콘텐츠를 변경할 수 있다.

[0377] 단계 3220에서 O/S 하위 보안 에이전트는 애플리케이션 I/O 경로를 통해, 애플리케이션이 실행되는 운영체제 및 운영체제 및 장치 사이의 드라이버들에 의한 동작을 포함하는 보통의 동작에 대해 변경된 콘텐츠를 전송할 수 있다. 이 단계 중에, 애플리케이션 I/O 경로에 영향을 미치는 악성 소프트웨어가 I/O 콘텐츠를 도용하고자 시도할 수 있다. 그러나, 도용된 데이터는 O/S 하위 보안 에이전트에 의해 삽입된 변형된 더미 콘텐츠일 수 있으

므로, 오리지널 콘텐츠를 도용으로부터 보호할 수 있다.

[0378] 단계 3225에서, O/S 하위 보안 에이전트는 변형된 콘텐츠가 애플리케이션에 도달할 때 그 변형된 콘텐츠를 인터셉트할 수 있다. 단계 3230에서, O/S 하위 보안 에이전트는 그 변형된 콘텐츠를 오리지널 콘텐츠로 대체할 수 있다. 그에 따라 오리지널 콘텐츠는 애플리케이션과 장치 사이에서 대역 밖 전송되어, 전통적 애플리케이션 I/O 경로의 콘텐츠를 도용하고자 시도하는 악성 소프트웨어로부터 계속 보호될 수 있다.

[0379] 단계 3235에서, O/S 하위 보안 에이전트는 변형된 콘텐츠가 악성 소프트웨어 유사 양태에 의해 영향을 받았는지 여부(가령, 변형된 데이터가 애플리케이션 I/O 경로에서 자체 변형된 상태로 애플리케이션 I/O 경로를 통과했는지)를 판단할 수 있다. 예를 들어 규칙들(가령, 보안 규칙들(114, 220, 222, 438, 434, 436, 518, 707, 721 및/또는 723)에 기반하여, O/S 하위 보안 에이전트는 인터셉트된 변형된 콘텐츠가 악성 소프트웨어에 의해 영향을 받았음을 나타내는 특징을 가지는지 여부를 판단할 수 있다. 또한, O/S 하위 보안 에이전트가 변형된 콘텐츠가 악성 소프트웨어 유사 양태에 의해 영향을 받았다고 판단하는 경우, O/S 하위 보안 에이전트는 교정 액션(가령, 제거 액션, 격리 및/또는 악성 소프트웨어 중립화)을 취할 수 있다. 또한 일부 실시예들에서, O/S 하위 보안 에이전트는 악성 소프트웨어 유사 양태 발생에 관한 정보(가령, 법의학적 정보)를 보호 서버로 전송할 수 있다.

[0380] 또한 일부 실시예들에서, (가령 방법(3000)의 단계들 3015 및 3020 및/또는 방법(3200)의 단계들(3215 및 3220에서) 애플리케이션 I/O 경로를 통해 전송되는 더미 데이터는 전자 장치(2904) 및/또는 전자 장치(3104) 상의 악성 소프트웨어 존재를 추적하는데 사용될 수 있다. 예를 들어, 제1장치에서의 애플리케이션이 네트워크(가령, 제2전자 장치에 의해 호스팅된 뱅킹 또는 다른 금융 웹사이트)를 통해 제2전자 장치로 민감한 정보를 전송할 때, O/S 하위 보안 에이전트는 제2전자 장치에 대한 액세스를 스푸핑할 수 있는 더미 정보(가령, "가짜" 유저네임과 패스워드를 은행 웹사이트로 제공할 수 있는 더미 정보)를 I/O 경로 안에 삽입할 수 있다. 제2전자 장치는 제2전자 장치가 이런 방식으로 스푸핑될 때, 제2전자 장치의 보안 에이전트가 악성 소프트웨어 유사 양태가 발생했는지를 판단하기 위해 액세스 중에 취해지는 액션들(가령, 프로파일 정보의 변경이나 다른 액션들과 같이 스푸핑된 액세스 중에 은행 웹사이트에서 취해지는 액션들)을 추적하도록 구성될 수 있게 그 자신의 보안 에이전트를 포함할 수 있다. 제2전자 장치에서의 보안 에이전트가 악성 소프트웨어 유사 양태가 발생하였다고 판단하면, 제2전자 장치가 교정 액션을 취할 수 있다. 예를 들어 제2전자 장치에서의 보안 에이전트는 악성 소프트웨어 유사 양태의 존재를 가리키기 위해 (가령, 제2전자 장치와 통신 가능하게 연결된 보호 서버(102)로) 적합한 메시지를 전송할 수 있다. 그러한 메시지는 예컨대 악성 소프트웨어 유사 양태에 대한 설명 및/또는 제1전자 장치의 아이디(가령, 인터넷 프로토콜 어드레스나 다른 식별 정보)을 포함하는 법의학적 증거를 포함할 수 있다.

[0381] 도 33은 전자 장치(3304) 상에서 숨겨진 프로세스들을 검출하여 복구하기 위한 시스템(3300)의 실시예이다. O/S 내 보안 에이전트(3318) 및 O/S 하위 보안 에이전트(3316)는 전자 장치(3304) 상에서 실행중인 프로세스의 동작을 감추도록 구성된 악성 소프트웨어와 같은 악성 감염들을 검출하고 복구하기 위해 전자 장치(3304) 상에서 동작할 수 있다. 전자 장치(3304)는 메모리(3308)에 연결된 프로세서(3306), 운영체제(3312) 및 하나 이상의 프로세스들(3373)을 포함할 수 있다. 전자 장치(3304)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 프로세서(3306)는 도 2의 프로세서(208), 도 4의 프로세서(408), 도 7의 프로세서(702) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(3308)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 운영체제(3312)는 도 1의 운영체제(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(3318)는 도 2의 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418), 도 7의 O/S 내 보안 에이전트(719), 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 하위 보안 에이전트(3316)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM 보안 에이전트(216) 또는 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442), PC 펌웨어 보안 에이전트(444) 또는 O/S 하위 에이전트(450), 도 5의 펌웨어 보안 에이전트(516), 도 7의 마이크로코드 보안 에이전트(708) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0382] 프로세스들(3373)은 전자 장치(3304) 상에서 동작하도록 구성될 수 있다. 전자 장치(3304) 상에서 동작하는 하

나 이상의 프로세스들(3373)은 악성 소프트웨어와 관련된 악성 프로세스들일 수 있다. 전자 장치(3304) 상의 악성 소프트웨어는 안티 악성 소프트웨어 소프트웨어에 의한 검출을 피하기 위해 프로세스들(3373) 중 하나 이상의 악성 프로세스들의 존재를 위장하도록 동작할 수 있다. 예를 들어 운영체제(3312)는 운영체제 커널 메모리(3380)를 포함할 수 있다. 운영체제 커널 메모리(3380)는 전자 장치(3304) 상의 프로세스들의 실행을 추적하기 위한 하나 이상의 메커니즘들을 포함할 수 있다. 일례에서 그러한 메커니즘은 액티브 프로세스 리스트(3384)를 포함할 수 있다. 액티브 프로세스 리스트(3384)는 전자 장치(3304) 상에서 동작하는 프로세스들을 추적하기 위해 데이터 구조, 레코드, 파일 또는 어떤 다른 적절한 방법으로 구현될 수 있다. 예컨대 프로세스(3373b)가 악성 소프트웨어와 관련된 악성 프로세스들인 경우, 전자 장치(3304) 상의 악성 소프트웨어는 프로세스(3373b)에 대한 참조(reference)를 제거하기 위해 액티브 프로세스 리스트(3384)를 변경할 수 있다. 그에 따라, 전자 장치(3304) 상에서 실행되는 보안 소프트웨어는 어떤 프로세스들이 전자 장치(3304) 상에서 능동적으로 실행되고 악성 소프트웨어에 대해 검사되어야 하는지를 결정할 때, 프로세스(3373b)를 검사를 위한 액티브 프로세스로서 인식하지 않을 것이다.

[0383] 운영체제(3312)는 준비 큐(ready queue)들(3322)을 포함할 수 있다. 준비 큐들(3322)은 전자 장치(3304) 상에서 동작하는 액티브 스레드들을 제시하는 하나 이상의 적절한 데이터 구조들(가령, 어레이들, 테이블들, 리스트들 등)을 포함할 수 있다. 액티브 프로세스(3373)에는 하나 이상의 개별 스레드들이 구비될 수 있다. 스레드는 전자 장치(3304)에 의한 실행을 위해 액티브 프로세스(3373)의 나머지 스레드들과 별개로 독립적으로 스케줄링될 수 있는 액티브 프로세스(3373) 안의 프로세싱 단위(가령, 하나 이상의 명령어들)라고 간주될 수 있다. 윈도우즈 운영체제의 준비 큐들(3322)의 예로서, 준비 큐들(3322)은 KiDispatcherReadyListHead라고 알려진 변수에 의해 구현될 수 있다. 준비 큐들(3322)은 또한 액티브 스레드들에 관한 다양한 메타데이터, 예컨대 스레드를 포함하는 프로세스의 식별자, 그러한 프로세스의 이미지 네임, 시작 어드레스, 사용자 모드 어드레스, 장치 객체 및/또는 다른 적절한 정보를 포함할 수 있다. 윈도우즈 운영체제에서, 그러한 프로세스 정보는 스레드와 관련된 실행 스레드("ETHREAD") 데이터 구조 안에 포함될 수 있다.

[0384] 시스템(3300)의 동작 중에, O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 운영체제(3312) 상에서 실행할 보안 장치 드라이버(3370)를 양산할 수 있다. 보안 장치 드라이버(3370)는 커널 모드 장치 드라이버 서비스들을 제공하기 위한 드라이버, 모듈, 실행자, DLL 또는 어떤 다른 적절한 메커니즘에서 구현될 수 있다. 보안 장치 드라이버(3370)는 전자 장치(3304) 상에서 실행되는 프로세스들을 열거하기 위해 운영체제(3312)의 다양한 부분들을 호출하도록 구성될 수 있다. 예를 들어 보안 장치 드라이버(3370)는 커널 메모리(3380)나 액티브 프로세스 리스트(3384)를 검사하도록 구성될 수 있다. 보안 장치 드라이버(3370)는 보안 장치 드라이버(3370)가 검출할 수 있는 액티브 프로세스들(3373)의 제1리스트(가령, 액티브 프로세스 리스트(3384))를 전송하도록 구성될 수 있다. 보안 장치 드라이버(3370)는 O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)로 액티브 프로세스 리스트(3384)를 전송하도록 구성될 수 있다. 일 실시예에서, 보안 장치 드라이버(3370)는 하이퍼콜을 통해, 주어진 검출 프로세스와 관련된 실행 프로세스("EPROCESS") 구조를 O/S 하위 보안 에이전트(3316)로 전달하도록 구성될 수 있다. 보안 장치 드라이버(3370)가 운영체제와 같거나 그보다 낮은 실행 특권 링에서 실행되기 때문에, 보안 장치 드라이버(3370)에 의해 열거된 액티브 프로세스들은 액티브 프로세스 리스트(3384) 상에 나타나는 액티브 프로세스들로 국한될 수 있는 바, 이는 악성 프로세스들 자체로의 참조를 제거할 변형된 액티브 프로세스 리스트(3384)를 가지는 악성 프로세스들이 보안 장치 드라이버(3370)에 의해 열거되지 않을 것임을 의미한다. 윈도우즈 운영체제에서, 보안 장치 드라이버(3370)는 운영체제로부터 프로세스 리스트를 요청하여 결정될 시스템 프로세스 정보를 식별하는 함수 ZwQuerySystemInformation를 이용하도록 구성될 수 있다. O/S 하위 보안 에이전트(3316) 역시 그러한 함수를 이용하도록 구성될 수 있고, 그러한 액션을 수행함에 있어 보다 안전해질 수 있다. 보안 장치 드라이버(3370)는 열거된 프로세스들을 액티브 프로세스들의 제1리스트(3385) 안에 놓을 수 있다. 소정 실시예들에서, 제1리스트(3385)는 실질적으로 액티브 프로세스 리스트(3384)에 해당할 수 있다. 다른 실시예들에서 별도의 제1리스트(3385)는 생성되지 않을 것이며, 보안 장치 드라이버(3370)는 그러한 제1리스트(3385) 대신 액티브 프로세스 리스트(3384)를 사용할 수 있다.

[0385] 반대로, O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 운영체제(3312)와 같거나 더 높은 실행 특권 링에서 실행될 수 있으며, 그에 따라 전자 장치(3304) 상에서 실행되는 개별 스레드들을 나열할 수 있다. 적어도 그렇게 열거된 스레드들에 기반하여, O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 액티브 프로세스 리스트(3384)로부터 자신들에 대한 참조를 제거했을 수 있는 악성 프로세스들을 포함하여, 전자 장치(334) 상에서 실행되는 모든 액티브 프로세스들(3373)을 판단할 수 있다. 예를 들어, 소정 실시예들에서 O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 준비 큐들(3322)을 검

색하고 준비 큐(3322) 안에 모든 스레들을을 나열하여 그 스레드들을 리스트 안에 위치시킬 수 있다. 매 스레드마다, O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 그 스레드를 소유한 프로세스 및 그러한 프로세스에 관한 정보를 위치시킬 수 있고(가령, 해당 스레드와 관련된 ETHREAD 정보와 같은 메타데이터에 대한 참보를 통해), 그에 따라 O/S 내 보안 에이전트(3318)이 액티브 프로세스 리스트(3384)로부터 그들 자체에 대한 참조들을 제거했을 수 있는 악성 프로세스들을 포함하여 액티브 프로세스들(3373)의 제2리스트(3386)를 열거하게 할 수 있다.

[0386] 특정 실시예를 더 예시하자면, 어떤 스레드와 관련된 ETHREAD 데이터 구조는 ThreadsProcess 필드, StartAddress 필드, DeviceTo Verify 필드, a Win32StartAddress 필드, 및 ThreadListEntry 필드를 포함하는 여러 메타데이터 필드들을 포함할 수 있다. ThreadsProcess 필드를 분석함으로써 O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 해당 프로세스에 대한 프로세스 식별자 및 이름이 판단될 수 있게 하는 스레드를 소유한 프로세스를 식별할 수 있다. StartAddress 및 Win32StartAddress로부터, O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 메모리 내 어느 코드가 프로세스를 실행하고 있는지를 식별할 수 있고, 그에 따라 해당 스레드를 소유한 프로세스가 의심스러운 것으로 파악된 경우 의심스러운 드라이버, 애플리케이션 및/또는 다른 프로그램의 추가 식별을 가능하게 할 수 있다. DeviceToVerify 필드에 기반하여, O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 장치 객체가 스레드와 관련된 것인지 여부를 판단할 수 있고, 그에 따라 그 스레드를 소유한 프로세스가 의심스럽다고 파악된 경우 드라이버 객체 및 악성 소프트웨어 드라이버를 식별할 수 있다. ThredListEntry는 동일한 프로세스 안에서 스레드들의 열거를 지원할 수 있다.

[0387] O/S 내 보안 에이전트(3318)는 액티브 프로세스들(3373)의 제1리스트(3385) 및 액티브 프로세스들(3373)의 제2리스트(3386)을 비교할 수 있으며, 제2리스트(3386)에는 나타나지만 제1리스트(3385)에는 나타나지 않는 액티브 프로세스들(3373)을 의심스러운 프로세스들로서 식별할 수 있다. 그러한 의심스러운 프로세스의 증거가 전자 장치(3304) 상에서 실행되는 안티 바이러스 또는 안티 악성 소프트웨어 소프트웨어 및 운영체제(3312)로부터 숨겨진 악성 소프트웨어의 증거일 수 있다.

[0388] 다른 실시예들에서 O/S 하위 보안 에이전트(3316)는 스레드들을 열거하기 위해 준비 큐들(3322)을 검색하고, 적어도 식별된 스레드들에 기반하여 액티브 프로세스들(3373)의 제2리스트(3386)를 열거할 수 있다(가령, 해당 스레드들과 연관된 ETHREAD 정보와 같은 메타데이터에 대한 참조를 통해). 그러한 실시예들에서 O/S 하위 보안 에이전트(3316)는 O/S 내 보안 에이전트(3318)로부터 보안 장치 드라이버(3370)에 의해 생성된 액티브 프로세스들의 제1리스트(3385)를 수신하거나, 메모리에서 바로 판독하여 액티브 프로세스들(3373)의 제1리스트(3385)를 액세스할 수 있다. O/S 하위 보안 에이전트(3316)는 액티브 프로세스들(3373)의 제1리스트(3385) 및 액티브 프로세스들(3373)의 제2리스트(3386)을 비교할 수 있으며, 제2리스트(3386)에는 나타나지만 제1리스트(3385)에는 나타나지 않는 액티브 프로세스들(3373)을 의심스러운 프로세스들로서 식별할 수 있다. 그러한 의심스러운 프로세스는 전자 장치(3304) 상에서 실행되는 안티 바이러스 또는 안티 악성 소프트웨어 소프트웨어 및 운영체제(3312)로부터 감춰질 수 있다. 그러한 의심스러운 프로세스의 증거가 전자 장치(3304) 상에서 실행되는 안티 바이러스 또는 안티 악성 소프트웨어 소프트웨어 및 운영체제(3312)로부터 숨겨진 악성 소프트웨어의 증거일 수 있다.

[0389] O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)가 전자 장치(3304) 상에서 실행되는 숨겨진 프로세스의 증거가 존재한다고 판단한 경우, O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 운영체제(3312), 운영체제 커널 메모리(3380) 또는 전자 장치(3304)의 다른 요소들을 검색하여 그러한 프로세스와 관련하여 어떠한 변형들이 이루어졌는지를 판단하도록 구성될 수 있다. 예를 들어 O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 악성 소프트웨어에 의해 수행되는 것으로 알려진 어떤 메모리 변형들에 대해 검색하도록 구성될 수 있다. 어떤 실시예들에서 O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 운영체제 코드 섹션(3382) 및 액티브 프로세스 리스트(3384)를 검색하도록 구성될 수 있다. 이러한 것들 및 기타 실시예들에서, O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 의심스러운 프로세스들과 관련된 스레드들의 스레드 메타데이터(가령, ETHREAD 정보)를 이용하여, 변형들에 대해 검색할 전자 장치(3304)의 요소들 및/또는 그 일부를 결정할 수 있다.

[0390] 악의적 변형이 발견된 경우, O/S 내 보안 에이전트(3318) 또는 O/S 하위 보안 에이전트(3316)는 교정 액션을 취할 수 있다. 예를 들어 O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 운영체제 커널 메모리(3380)에서 발견된 어떤 악의적 변형들을 복구하도록 구성될 수 있다. 다른 예로서, O/S 내 보안 에이전트(3318)이나 O/S 하위 보안 에이전트(3316)는 운영체제 커널 메모리(380) 내 메모리 변형들에 대한 검사를 통

해 판단된 어떤 검출된 루트킷 감염들을 제거하도록 구성될 수 있다. 또 다른 예로서, O/S 내 보안 에이전트(3318) 또는 O/S 하위 보안 에이전트(3316)은 어떤 내부 데이터 구조나 코드 섹션들에 대한 어떤 감염들을 복구하도록 구성될 수 있다. 이러한 것들 및 기타 실시예들에서, O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316)는 의심스러운 프로세스들과 관련된 스레드들의 스레드 메타데이터(가령, ETHREAD 정보)를 이용하여, 취해질 교정 액션을 결정할 수 있다(가령, 메타데이터가 의심스러운 동향에 대해 책임이 있는 드라이버들, 악성 소프트웨어 프로세스들의 특정 메모리 위치 등을 식별할 수 있다). O/S 내 보안 에이전트(3318) 또는 O/S 하위 보안 에이전트(3316)는 보안 장치 드라이버(3370)에 의해 결정된 프로세스들의 제1리스트(3385) 및 준비 큐들(3322)에 존재하는 스레드 메타데이터의 분석으로부터 결정된 프로세스들의 제2리스트(3386) 사이에서 발견된 모든 차이에 대해 숨겨진 프로세스에 의한 변형들을 찾기 위한 검색 프로세스를 반복하도록 구성될 수 있다.

[0391] 도 34는 전자 장치 상에서 숨겨진 프로세스들을 검출하여 복구하기 위한 방법의 실시예이다. 단계 3405에서 하나 이상의 보안 에이전트들(가령, O/S 내 보안 에이전트(3318) 및/또는 O/S 하위 보안 에이전트(3316))이 전자 장치의 운영체제 상에서 실행될 보안 장치 드라이버를 양산할 수 있다. 단계 3410에서 보안 장치 드라이버는 전자 장치의 운영체제에 대한 표준 시스템 호출들을 통해 전자 장치 상에서 실행되는 액티브 프로세스들을 나열할 수 있다. 보안 장치 드라이버는 열거된 프로세스들을 액티브 프로세스들의 제1리스트 안에 놓을 수 있다. 보안 장치 드라이버가 운영체제와 같거나 그보다 낮은 실행 특권 링에서 실행될 수 있기 때문에, 보안 장치 드라이버에 의해 열거된 액티브 프로세스들은 운영체제의 액티브 프로세스 리스트 상에 나타나는 액티브 프로세스들로 국한될 수 있는 바, 이는 악성 프로세스들 자체로의 참조를 제거하도록 액티브 프로세스 리스트를 변형했던 그 악성 프로세스들이 보안 장치 드라이버에 의해 열거되지 않을 것임을 의미한다.

[0392] 단계 3415에서 하나 이상의 보안 에이전트들이 스레드 준비 큐들을 검색할 수 있으며, 적어도 그러한 검색에 기반하여 전자 장치 상에서 실행되는 개별 스레드들을 나열하고 그들을 스레드 리스트 안에 놓을 수 있다. 단계 3420에서 적어도 스레드들과 관련된 메타데이터(가령, 해당 스레드들과 관련된 ETHREAD 정보나 해당 스레드들을 소유한 프로세스들을 제시하는 다른 메타데이터)에 기반하여, 하나 이상의 보안 에이전트들이 그 스레드들을 소유한 프로세스들을 찾아서 액티브 프로세스들의 제2리스트를 생성할 수 있다. 제2리스트는 액티브 프로세스 리스트로부터 그들(악성 프로세스들) 자신들로의 참조들을 제거했을 수 있는 악성 프로세스들을 포함할 수 있다.

[0393] 단계 3425에서 하나 이상의 보안 에이전트들은 액티브 프로세스들의 제1리스트 및 액티브 프로세스들의 제2리스트를 비교할 수 있다. 단계 3430에서, 하나 이상의 보안 에이전트들이 제2리스트에는 나타나고 제1리스트에는 나타나지 않는 액티브 프로세스들을 의심스러운 프로세스들로서 식별할 수 있다. 그러한 의심스러운 프로세스의 증거가 전자 장치 상에서 실행되는 안티 바이러스 또는 안티 악성 소프트웨어 소프트웨어 및 전자 장치 상에서 실행되는 운영체제로부터 숨겨진 악성 소프트웨어의 증거일 수 있다.

[0394] 단계 3435에서, 하나 이상의 보안 에이전트들이 전자 장치 상에 실행되는 숨겨진 프로세스에 대한 증거가 존재한다고 판단하는 경우, 그 하나 이상의 보안 에이전트들은 의심스러운 프로세스에 의해 전자 장치의 일부에 대해 변형이 이루어졌는지를 판단할 수 있다. 변형이 이루어졌는지를 판단하기 위해, 하나 이상의 보안 에이전트들은 그러한 프로세스와 관련된 어떤 변형들이 이루어졌는지 여부를 판단하도록 운영체제 및/또는 운영체제 커널 메모리를 검색할 수 있다. 예를 들어 하나 이상의 보안 에이전트들은 악성 소프트웨어에 의해 수행되는 것으로 알려진 어떤 메모리 변형들에 대해 검색할 수 있고/있거나, 운영체제 커널 메모리의 운영체제 코드 섹션 및/또는 액티브 프로세스 리스트를 검색할 수 있다.

[0395] 단계 3440에서 하나 이상의 보안 에이전트들은 변형이 발견되었을 경우 교정 액션을 취할 수 있다. 예를 들어 하나 이상의 보안 에이전트들은 운영체제 커널 메모리에서 발견된 어떤 악의적 변형들을 복구할 수 있다. 다른 예로서, 하나 이상의 보안 에이전트들은 운영체제 커널 메모리 내 메모리 변형들에 대한 검사를 통해 판단된 어떤 검출된 루트킷 감염들을 제거할 수 있다. 추가 예로서, 하나 이상의 보안 에이전트들은 어떤 내부 데이터 구조나 코드 섹션들에 대한 어떤 감염들을 복구할 수 있다. 방법(3400)의 부분들이 각각의 식별된 의심스러운 프로세스마다 반복될 수 있다. 따라서, 하나 이상의 보안 에이전트들은 보안 장치 드라이버에 의해 결정된 프로세스들의 제1리스트 및 준비 큐들에 존재하는 스레드 메타데이터의 분석으로부터 결정된 프로세스들의 제2리스트 사이에서 발견된 모든 차이에 대해 숨겨진 프로세스에 의한 변형들을 찾기 위한 검색 프로세스를 반복할 수 있다.

[0396] 바람직하게도 상술한 방법들 및 시스템들은 운영체제의 어떤 함수의 후킹이나 트래핑에 대한 요건 없이, 루트킷들 및/또는 다른 악성 소프트웨어의 식별을 제공할 수 있다.

[0397] 도 35는 전자 장치(3504) 상에서 숨겨진 프로세스들을 검출하여 복구하기 위한 시스템(3500)의 실시예이다. O/S 내 보안 에이전트(3516), 보안 장치 드라이버(3570) 및 보안 동적 링크 라이브러리(DLL)(3572)는 전자 장치(3504) 상에서 실행중인 프로세스의 동작을 감추도록 구성된 악성 소프트웨어와 같은 악성 감염들을 검출하고 복구하기 위해 전자 장치(3504) 상에서 동작할 수 있다. 전자 장치(3504)는 메모리(3508)에 연결된 프로세서(3506), 운영체제(3512), 보안 DLL(3572), O/S 하위 보안 에이전트(3516), 가상 머신 제어 구조(3552)("VMCS"), 하나 이상의 프로세스들(3573)(가령, 프로세스(3573a, 3573b, 및 3573c)), 그러한 프로세스들과 관련된 어드레스 공간들(3587)(가령, 어드레스 공간들(3587a, 3587b 및 3587c), 및 CR3 제어 레지스터(3560) 같은 제어 레지스터와 같은 하나 이상의 시스템 자원들을 포함할 수 있다. 프로세서 레지스터들(3530)은 예컨대 CR3 레지스터(3560) 같은 레지스터들이나 어떤 다른 레지스터들(3568)을 포함할 수 있다. CR3가 프로세서 레지스터(3530)의 일례로서 주어지지만, 모든 적절한 제어 레지스터가 사용될 수 있다. CR3 레지스터(3560)은 전자 장치(3504) 상의 CPU의 일반적 동향을 제어하거나 바꾸도록 구성된 프로세서 레지스터일 수 있다. CR3 레지스터(3560)은 전자 장치(3504) 상에서 실행되는 프로세서(3506)와 같은 프로세서가 가상 메모리 어드레스를 물리적 메모리 어드레스로 변환하게 할 수 있도록 구성될 수 있다. CR3 레지스터(3560)는 스택에 상주하는 것과 같은 현재 요청된 태스크에 대해, 그리고 O/S 스케줄러에 의한 동작에 대해 선택된 페이지 디렉토리 및 페이지 테이블들을 찾도록 구성될 수 있다. CR3 레지스터(3560)는 어떤 적절한 가상 어드레싱 제어 레지스터 안에 구현될 수 있다. 전자 장치(3504)의 특정 디자인이나 구현 예에 따라 다른 레지스터들(268)이 프로세서 레지스터들(3530) 안에 존재할 수 있다. 프로세서 레지스터들(3530)은 프로세서(3506) 또는 전자 장치(3504)의 다른 프로세서와 관련될 수 있다.

[0398] 전자 장치(3504)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701), 도 33의 전자 장치(3304) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 프로세서(3506)는 도 2의 프로세서(208), 도 4의 프로세서(408), 도 7의 프로세서(702), 도 33의 프로세서(3306) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(3508)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703), 도 33의 메모리(3308) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 운영체제(3512)는 도 1의 운영체제(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 33의 운영체제(3312) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 하위 보안 에이전트(3516)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM 보안 에이전트(216) 또는 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442), PC 펌웨어 보안 에이전트(444) 또는 O/S 하위 에이전트(450), 도 5의 펌웨어 보안 에이전트(516), 도 7의 마이크로코드 보안 에이전트(708), 도 33의 O/S 하위 보안 에이전트(3316) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0399] 전자 장치(3504)는 가상 머신 제어 구조(1152)를 포함할 수 있다. 일 실시예에서 가상 머신 제어 구조(1152)는 O/S 하위 보안 에이전트(3516) 안에 상주할 수 있다. 다른 실시예에서 가상 머신 제어 구조(1152)는 O/S 하위 보안 에이전트(3516)와 통신 가능하게 연결될 수 있다. 그러한 실시예에서, 가상 머신 제어 구조(1152)의 기능의 일부나 전부는 O/S 하위 에이전트(3516)에 의해 수행될 수 있다. 또한 그러한 실시예에서, O/S 하위 보안 에이전트(3516)의 기능 중 일부나 전부가 가상 머신 제어 구조(1152)에 의해 수행될 수 있다. 가상 머신 제어 구조(1152)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 VMCS(216), 도 4의 펌웨어 보안 에이전트들(440, 442) 또는 PC 펌웨어 보안 에이전트(444), 도 5의 펌웨어 보안 에이전트(516) 또는 도 7의 마이크로코드 보안 에이전트(706)에 의해 전적으로나 일부가 구현될 수 있다. 가상 머신 제어 구조(1152)는 프로세서 레지스터들(3530)과 같은 자원들이나 다른 자원들에 대한 요청들을 트래핑하기 위한 데이터 구조, 레코드, 파일, 모듈 또는 어떤 다른 적절한 개체에서 구현될 수 있다. 시스템(3500)이 도 2의 시스템(200)에 의해 전체나 일부가 구현될 수 있는 경우와 같은 일 실시예에서, 가상 머신 제어 구조(1152) 및 O/S 하위 보안 에이전트(3516)는 프로세서 레지스터들(3530)과 같은 전자 장치(3504)의 시스템 자원들이나 어떤 다른 적절한 시스템 자원들로의 액세스를 가상화하도록 구성될 수 있다.

[0400] 가상 머신 제어 구조(1152)는 프로세서 레지스터들(3530)과 같은 시스템 자원들의 운영체제(3512)에 의해 요청되는 동작들을 트래핑하기 위해 하나 이상의 플래그들(1154)을 포함할 수 있다. 플래그들(1154)은 트래핑을 위한 플래그들, 예컨대 프로세스 정황 전환 플래그(3556) 및/또는 읽기 함수 플래그(3558)를 포함할 수 있다. 플래그들(1154)은 프로세서 레지스터들(3530)과 같은 시스템 자원들에 대한 액세스를 트래핑하기 적합한 어떤 플

래그를 포함할 수 있다. O/S 하위 보안 에이전트(3516)는 가상 머신 제어 구조(1152)의 어느 플래그들(1154)이 시스템 자원들에 대한 액세스를 트래핑하는 데 사용될 것인지를 세팅하도록 구성될 수 있다. 가상 머신 제어 구조(1152) 및 O/S 하위 보안 에이전트(3516)에 의해 트래핑되고/거나 보호될 수 있는 시스템 자원들은 프로세서 레지스터들(3530)을 포함할 수 있으나 그에 국한되지 않는다.

[0401] 프로세스들(3573)은 전자 장치(3504) 상에서 동작하도록 구성될 수 있다. 전자 장치(3504) 상에서 동작하는 하나 이상의 프로세스들(3573)은 악성 소프트웨어와 관련된 악성 프로세스들일 수 있다. 전자 장치(3504) 상의 악성 소프트웨어는 안티 악성 소프트웨어 소프트웨어에 의한 검출을 피하기 위해 프로세스들(3573) 중 하나 이상의 악성 프로세스들의 존재를 위장하도록 동작할 수 있다. 예를 들어 운영체제(3512)는 운영체제 커널 메모리(3580)를 포함할 수 있다. 운영체제 커널 메모리(3580)는 전자 장치(3504) 상의 프로세스들의 실행을 추적하기 위한 하나 이상의 메커니즘들을 포함할 수 있다. 일례에서 그러한 메커니즘은 액티브 프로세스 리스트(3584)를 포함할 수 있다. 액티브 프로세스 리스트(3584)는 전자 장치(3504) 상에서 동작하는 프로세스들을 추적하기 위해 데이터 구조, 레코드, 파일 또는 어떤 다른 적절한 방법으로 구현될 수 있다. 예컨대 프로세스(3573b)가 악성 소프트웨어와 관련된 악성 프로세스들인 경우, 전자 장치(3504) 상의 악성 소프트웨어는 프로세스(3573b)에 대한 참조(reference)를 제거하기 위해 액티브 프로세스 리스트(3584)를 변경할 수 있다. 그에 따라, 전자 장치(3504) 상에서 실행되는 보안 소프트웨어는 어떤 프로세스들이 전자 장치(3504) 상에서 능동적으로 실행되고 악성 소프트웨어에 대해 검사되어야 하는지를 결정할 때, 프로세스(3573b)를 검사를 위한 액티브 프로세스로서 인식하지 않을 것이다.

[0402] 전자 장치(3504) 상에서 동작한 프로세스들(3573) 또는 다른 개체들은 가상 메모리를 이용할 때 정상 동작들의 일부로서 프로세스들(3573) 중 하나와 관련된 프로세스 정황 전환의 사용을 요할 수 있다. 가상 메모리의 사용을 촉진하기 위해 운영체제(3512)는 프로세스 정황 전환, 읽기 또는 주어진 프로세스로의 첨부를 수행하도록 구성될 수 있다. 그러한 액션들은 운영체제(3512)에게 CR3 레지스터(3560)와 같은 제어 레지스터를 포함하는 시스템 자원들을 액세스하고자 시도하도록 요구할 수 있다. 운영체제(3512)는 CR3 레지스터(3560)의 읽기를 "move value, CR3"의 명령 형식으로 생성할 수 있다. 운영체제(3512)는 "move CR3, value"의 명령 형식으로 CR3 레지스터(3560)의 값 변경을 시도하도록 구성될 수 있다.

[0403] 가상 머신 제어 구조(1152)는 레지스터들(3530)을 포함해 전자 장치(3504)의 시스템 자원들을 액세스하려는 운영체제(3512)에 의한 시도들을 인터셉트하도록 구성될 수 있다. 가상 머신 제어 구조(1152)는 전자 장치(3504)의 시스템 자원들을 액세스하려는 운영체제(3512)에 의해 시도된 어떤 명령들을 트래핑하고자 하도록 구성될 수 있다. 가상 머신 제어 구조(1152)는 운영체제(3512)의 명령들을 인터셉트하기 위해 플래그들을 사용하도록 구성될 수 있다. 일 실시예에서, 가상 머신 제어 구조(1152)는 CR3 레지스터(3560) 상의 프로세스 정황 전환 및 읽기 명령들을 인터셉트 하기 위해 플래그들(3556-3558)을 포함할 수 있다. O/S 하위 보안 에이전트(3516)는 가상 머신 제어 구조(1152) 안에서 그러한 플래그들(3556-3558)을 세팅하도록 구성될 수 있다. 가상 머신 제어 구조(1152)는 CR3 레지스터(3560)와 관련된 읽기나 프로세서 정황 전환과 같이 플래깅된 동작의 인터셉트 시 생성되는 이벤트인 VM exit(빠져 나가기)를 생성하도록 구성될 수 있다. 일 실시예에서 가상 머신 제어 구조(1152)는 가상 메모리와 관련된 제어 레지스터의 어떤 시도된 액세스에 대해 VM exit을 생성하도록 구성될 수 있다. 전자 장치(3504) 상에서 실행되는 프로세스들(3573) 중 하나가 프로세스 정황 전환을 수행하거나 프로세스와 관련된 프로세스 공간을 판독하고자 시도할 때마다, 가상 머신 제어 구조(1152)는 VM exit을 생성하고 시도된 명령에 대한 정보를 O/S 하위 보안 에이전트(3516)으로 전송하도록 구성될 수 있다. 예시하자면, O/S 하위 보안 에이전트(3516)는 CR3 레지스터(3560)(또는 다른 레지스터(3568))에 대한 모든 그러한 액션들을 레지스터 변화사항(3576) 안에 기록하도록 구성될 수 있다. 레지스터 변화사항(3576)은 파일, 구조, 데이터 구조, 레코드 또는 CR3 레지스터(3560)이나 다른 레지스터(3568)에 대한 변화의 이력을 저장하기 위한 어떤 다른 적절한 메커니즘으로 구현될 수 있다. O/S 하위 보안 에이전트(3516)는 CR3 레지스터(3560)의 모든 액세스들을 기록함에 따라, 전자 장치(3504)에서 프로세스 정황 전환을 시도했던 모든 프로세스들(3573)의 레코드를 가질 수 있다. 레지스터 변화사항(3576)으로서의 그러한 변경들의 레코드는 전자 장치(3504) 상에서 실행되는 프로세스들의 레코드로서의 역할을 하도록 구성될 수 있다. O/S 하위 보안 에이전트(3516)는 레지스터 변경사항(3576)으로부터 전자 장치(3504)에서 실행되고 있는 모든 프로세스들을 포함하는 실행중인 프로세스들(3586)의 리스트를 결정하도록 구성될 수 있다.

[0404] CR3 레지스터에 대한 액세스들을 인터셉트함으로써, O/S 하위 보안 에이전트(3516)는 실행 안팎에서 스와핑될 프로세스들의 기능을 거부함으로써 실행을 동결(freezing)할 수 있다. 실행을 동결함으로써, O/S 하위 보안 에이전트(3516)는 교정 또는 검출 조처들을 피하거나 전복하기 위해 동작할 수 있는 악성 소프트웨어와의 경합 조

건들 속에 들어가지 않고 실행중인 프로세스들의 리스트들을 모으도록 구성될 수 있다.

[0405] 대안적으로, O/S 하위 보안 에이전트(3516)는 정황 전환을 위한 운영체제 함수들(가령, 윈도우즈 함수 SwapContext)을 감시함으로써 실행중인 프로세스들(3586)의 리스트를 결정하도록 구성될 수 있다. 운영체제 정황 전환 함수들의 감시는 함수 호출에 대한 코드 섹션들과 관련된 메모리의 프로세서 레벨 감시를 통하거나 그러한 코드 섹션들을 가리키는 실행 명령어 포인터("EIP")의 관찰을 통해 수행될 수 있다. 예를 들어, SwapContext가 상주하는 물리적이거나 가상적인 메모리 위치가 보안 규칙 안에 매핑되어 기술될 수 있다. SwapContext가 상주하는 물리적 어드레스 또는 가상 메모리 어드레스에 대한 플래그가 세팅되어, 그 메모리에 대해 시도된 어떤 실행이 트래핑될 수 있도록 할 수 있다. 예를 들어 O/S 하위 보안 에이전트(3516)가 도 2의 SVMM 보안 에이전트(217)에 의해 전적으로나 부분적으로 구현되면, O/S 하위 보안 에이전트(3516)는 SwapContext가 상주하는 가상 메모리 페이지에 대해 시도된 어떤 실행을 트래핑하기 위해 VMCS(1152)를 세팅할 수 있다. 또 다른 예로서, O/S 하위 보안 에이전트(3516)가 도 7의 마이크로코드 보안 에이전트(708)에 의해 전적으로나 부분적으로 구현되면, O/S 하위 보안 에이전트(3516)는 SwapContext 함수의 코드 섹션을 시작하는 물리적 메모리 어드레스에 대해 시도된 실행을 트래핑하기 위해 VMCS(1152)를 세팅할 수 있다.

[0406] 운영체제(3512)는 보안 장치 드라이버(3570)를 포함할 수 있다. 보안 장치 드라이버(3570)는 커널 모드 장치 드라이버 서비스들을 제공하기 위한 드라이버, 모듈, 실행자, DLL 또는 어떤 다른 적절한 메커니즘에서 구현될 수 있다. 보안 장치 드라이버(3570)는 전자 장치(3504) 상에서 실행되는 실행중인 프로세스들을 열거하기 위해 운영체제(3512)의 다양한 부분들을 호출하도록 구성될 수 있다. 예를 들어 보안 장치 드라이버(3570)는 커널 메모리(3580)나 액티브 프로세스 리스트(3584)를 검사하도록 구성될 수 있다. 보안 장치 드라이버(3570)는 보안 장치 드라이버(3570)가 검출할 수 있는 실행중인 프로세스 리스트(3580)를 전송하도록 구성될 수 있다. 보안 장치 드라이버(3570)는 O/S 하위 보안 에이전트(3516)로 실행중인 프로세스 리스트(3580)를 전송하도록 구성될 수 있다. 일 실시예에서, 보안 장치 드라이버(3570)는 하이퍼콜을 통해, 주어진 검출 프로세스와 관련된 EPROCESS 구조를 O/S 하위 보안 에이전트(3516)로 전달하도록 구성될 수 있다. 보안 장치 드라이버(3570)에 의해 검출된 액티브 프로세스들의 EPROCESS 구조를 수신 시, O/S 하위 보안 에이전트(3516)는 보안 장치 드라이버(3570)로부터 수신된 그러한 각각의 EPROCESS 구조와 관련된 CR3 레지스터(3560) 값들(또는 다른 레지스터(3568) 값들)을 산출하도록 구성될 수 있다. O/S 하위 보안 에이전트(3516)는 보아 서비스 드라이버(3570)로부터 수신된 실행중인 프로세스들의 리스트(3580)을 O/S 하위 보안 에이전트(3516)가 레지스터 변화사항(3576)으로부터 결정했던 실행중인 프로세스들의 리스트(3586)와 비교하도록 구성될 수 있다. O/S 하위 보안 에이전트(3516)는 O/S 하위 보안 에이전트(3516)이 컴파일 했던 레지스터 변화사항(3576)과 비교하여 보안 장치 드라이버(3570)로부터의 실행 프로세스 리스트(3580)로부터 파생된 어떤 CR3 값들이 누락되어 있는지 여부를 판단하도록 구성될 수 있다. 그러한 차이는 숨겨진 프로세스를 가리킬 수 있다. 그러한 프로세스는 보안 장치 드라이버(3570), 전자 장치(3504) 상에서 실행되는 어떤 안티 바이러스 또는 안티 악성 소프트웨어 소프트웨어 및 운영체제(3512)로부터 감춰질 수 있다. 그러나 그러한 프로세스의 증거는 O/S 하위 보안 에이전트(3516)에는 보여졌을 수도 있는데, 이는 그러한 감춰진 프로세스가 예컨대 프로세스 어드레스 공간의 읽기나 프로세스 정황 전환을 시도하였기 때문이다. 그러한 감춰진 프로세스의 증거는 커널 모드 보안 장치 드라이버(3570)가 그 감춰진 프로세스를 검출할 수 없기 때문에 커널 루트 장치 드라이버를 통한 커널 모드 감염의 증거가 될 수 있다.

[0407] O/S 하위 보안 에이전트(3516)는 숨겨진 프로세스는 아니지만 대신, 리스트들이 컴파일되었을 때의 시기들 사이에서 정상적 실행 과정에서 삭제되었던 누락된 프로세스를 판단할 수 있다. 그러한 스레드들을 숨겨진 프로세스들이라 잘못 식별하는 것을 막게 돕기 위해, O/S 하위 보안 에이전트(3516)는 프로세스들을 생성 및 삭제하는 함수들의 실행을 감시하도록 구성될 수 있다. 그러한 함수들은 예컨대 pspProcessCreate 또는 pspTerminateProcess를 포함할 수 있다. O/S 하위 보안 에이전트(3516)는 그 나열 액션들 및 관찰한 생성 또는 삭제 함수들에 대한 타임 스탬프 레코드들을 만들어, 만일 어느 프로세스가 누락된 경우 그 프로세스가 누락되었다고 식별되기 전에 삭제 함수를 통해 삭제되었는지 여부를 판단할 수 있도록 구성될 수 있다.

[0408] O/S 하위 보안 에이전트(3516)가 전자 장치(3504) 상에서 실행되는 숨겨진 프로세스의 증거가 존재한다고 판단한 경우, O/S 하위 보안 에이전트(3516)는 운영체제(3512) 및 운영체제 커널 메모리(3380)를 검색하여 그러한 프로세스와 관련하여 어떠한 변형들이 이루어졌는지를 판단하도록 구성될 수 있다. O/S 하위 보안 에이전트(3516)는 악성 소프트웨어에 의해 수행되는 것으로 알려진 어떤 메모리 변형들에 대해 검색하도록 구성될 수 있다. 어떤 실시예들에서 O/S 하위 보안 에이전트(3516)는 운영체제 코드 섹션(3582) 및 액티브 프로세스 리스트(3584)를 검색하도록 구성될 수 있다. O/S 하위 보안 에이전트(3516)는 운영체제 커널 메모리(3580)에서 발견된 어떤 악의적 변형들을 복구하도록 구성될 수 있다. O/S 하위 보안 에이전트(3516)는 운영체제 커널 메모리

(3580) 내 메모리 변형들에 대한 검사를 통해 판단된 어떤 검출된 루트킷 감염들을 제거하도록 구성될 수 있다. O/S 하위 보안 에이전트(3516)는 어떤 내부 데이터 구조나 코드 섹션들에 대한 어떤 감염들을 복구하도록 구성될 수 있다. O/S 하위 보안 에이전트(3516)는 O/S 하위 보안 에이전트(3516) 및 보안 장치 드라이버(3570)에 의해 결정된 프로세스들 사이에서 발견된 모든 차이에 대해 숨겨진 프로세스에 대한 메모리 변형들에 대해 검색하는 프로세스를 반복하도록 구성될 수 있다. 보안 장치 드라이버(3570)는 실행중인 프로세스 리스트(3586)와 같은 O/S 하위 보안 에이전트(3516)로부터 최종 프로세스 리스트를 수신하도록 구성될 수 있다. 보안 장치 드라이버(3570)는 O/S 하위 보안 에이전트(3516)로의 하이퍼콜을 통해 실행중인 프로세스 리스트(3586)를 액세스하도록 구성될 수 있다.

[0409] 보안 DLL(3572)은 전자 장치(3504) 상에서 동작하도록 구성될 수 있다. 보안 DLL(3572)은 동적 링크 라이브러리(DLL), 공유 라이브러리, 실행자, 또는 이하에 기술되는 바와 같이 자신의 함수들을 수행하기 위한 어떤 다른 적절한 메커니즘으로 구현될 수 있다. 보안 장치 드라이버(3570)는 보안 DLL(3572)나 보안 DLL(3572)에 대한 참조를 프로세스 어드레스 공간(3587)과 같은 전자 장치(35040 상에서 실행중인 각각의 프로세스의 어드레스 공간 안에 주입하도록 구성될 수 있다. 프로세스 어드레스 공간들(3587) 각각은 그에 따라 보안 DLL(3572)에 대한 포인터를 포함할 수 있다. 보안 DLL(3572)은 전자 장치(3504) 상에서 실행되는 모든 사용자 모드 프로세스들을 나열하도록 구성될 수 있다. 보안 DLL(3572)은 사용자 모드 프로세스들을 나열하기 위해 어떤 적절한 기법을 이용하도록 구성될 수 있다. 예를 들어 보안 DLL(3572)은 윈도우즈 운영체제에서 NtQuerySystemlnformation 함수를 사용하고, ProcessFirst 및 ProcessNext 함수들을 이용하여 프로세스들을 탐색하도록 구성될 수 있다. 그러한 함수들은 O/S 하위 보안 에이전트에 의해 수행될 수도 있다. 보안 DLL(3572)은 그 결과들을 실행중인 프로세스 리스트(3588)로 컴파일하도록 구성될 수 있다. 보안 DLL(3572)은 실행중인 프로세스 리스트(3588)를 보안 장치 드라이버(3570)로 전송하도록 구성될 수 있다. 보안 DLL(3572)은 그러한 실행중인 프로세스 리스트(3588)를 보안 입/출력 호출들을 통해 전송하도록 구성될 수 있다. 보안 장치 드라이버(3570)는 수신된 실행중인 프로세스 리스트(3588)를 O/S 하위 보안 에이전트(3516)으로부터 수신된 실행중인 프로세스 리스트(3586)와 비교하도록 구성될 수 있다. 보안 장치 드라이버(3570)는 그러한 프로세스 리스트들의 차이가 사용자 모드 루트킷 공격을 포함할 수 있다고 판단하도록 구성될 수 있다. 보안 장치 드라이버(3570)나 보안 DLL(3572)는 프로세스 어드레스 공간들(3587)과 관련된 메모리 안의 프로세스 코드 및 데이터 섹션들을 검사하여 어떤 메모리 변경이 이루어졌는지 여부를 판단하도록 구성될 수 있다. 보안 장치 드라이버(3570)는 프로세스 어드레스 공간들(357)에서 이루어진 어떤 메모리 변경들을 복구하도록 구성될 수 있다. 보안 장치 드라이버(3570)는 보안 DLL(3572)로부터 실행중인 프로세스 리스트(3588) 및 O/S 하위 보안 에이전트(3516)로부터의 실행중인 프로세스 리스트(3588) 사이에 발견된 차이마다, 프로세스 어드레스 공간들(3587)의 검색, 메모리 변경 검출, 및 그러한 메모리 변경들의 복구 프로세스를 반복하도록 구성될 수 있다.

[0410] 동작 시 프로세스들(3573)은 전자 장치(3504) 상에서 동작할 수 있다. 프로세스들(3573) 중 하나 이상이 숨겨질 수 있다. 예를 들어 프로세스(3573b)가 악성 소프트웨어와 관련될 수 있으며, 전자 장치(3504) 상에서 실행되는 안티 바이러스나 안티 악성 소프트웨어 소프트웨어로부터 자신의 악의적 동작들을 위장하기 위해 숨겨질 수 있다. 프로세스들(3573)은 운영체제(3512)를 통해 전자 장치(3504)의 시스템 자원들을 액세스할 수 있다. 프로세스들(3573)은 메모리의 상이한 부분들을 액세스하기 위해, 또는 프로세서(3508)에 의해 실행되기 위해, 전자 장치(3504)의 제어 레지스터 액세스를 요구할 수 있다. 그러한 액세스는 프로세스 정황 전환 야기나 프로세스 어드레스 공간의 판독을 포함할 수 있다. 그러한 요건들이 운영체제(3512)에 의해 처리될 수 있으며, 이때 운영체제(3512)는 CR3 레지스터(3560)과 같은 레지스터를 액세스한다. 가상 머신 제어 구조(1152)는 그러한 요청들을 인터셉트하고 그 요청에 대해 VM exit을 생성할 수 있다. 가상 머신 제어 구조(1152)는 O/S 하위 보안 에이전트(3516)로 그러한 시도들과 관련된 정보를 제공할 수 있다. O/S 하위 보안 에이전트(3516)은 가상 머신 제어 구조(1152) 상에서 "move CR3, value"(3556)이나 "move value, CR3"(3558) 명령어들을 트래핑하기 위한 것들과 같은 플래그들을 세팅할 수 있다. O/S 하위 보안 에이전트(3516)는 CR3 레지스터(3560) 및 레지스터 변화사항(3576)에 대해 시도된 모든 읽기들이나 변경들을 기록할 수 있다.

[0411] 전자 장치(3504) 상에서 실행되는 하나 이상의 프로세스들(3573)이 숨겨졌는지 여부를 판단하기 위해, 보안 장치 드라이버(3570)는 어떤 커널 모드 프로세스들이 운영체제(3512) 상에서 실행되고 있는지 운영체제(3512)로부터 판단할 수 있다. 보안 장치 드라이버(3570)는 액티브 프로세스 리스트(3584)와 같은 운영체제 커널 메모리(3580)의 일부를 검색함으로써 그러한 프로세스들을 판단할 수 있다. 보안 장치 드라이버(3570)는 그에 따라, 전자 장치(3504)의 커널 모드 중의 동작을 검출할 수 있는 실행중인 프로세스 리스트(3580)를 가질 수 있다. 보안 장치 드라이버(3570)는 O/S 하위 보안 에이전트(3516)로 실행중인 프로세스 리스트(3580)를 전송할 수 있다. 보안 장치 드라이버(3570)은 실행중인 프로세스 리스트(3580)에서 검출된 각각의 프로세스의 EPROCESS 구

조를 하이퍼콜을 통해 O/S 하위 보안 에이전트(3516)로 전달함으로써, 실행중인 프로세스 리스트(3580)를 O/S 하위 보안 에이전트(3516)로 보낼 수 있다. O/S 하위 보안 에이전트(3516)는 실행중인 프로세스 리스트(3580) 안에 포함된 그러한 각각의 EPROCESS의 CR3 값들을 산출할 수 있다. O/S 하위 보안 에이전트(3516)는 그런 다음, 실행중인 프로세스 리스트(3580)에서 보안 장치 드라이버(3570)로부터 발생한 CR3 값들을, 전자 장치(3504)의 동작 중에 컴파일 했던 레지스터 변화사항(3576)과 비교할 수 있다. 실행중인 프로세스 리스트(3580)과 레지스터 변화사항(3576) 사이의 어떤 차이는 프로세스들(3573) 중 하나 이상이 전자 장치(3504) 상에 숨겨진 결과일 수 있다.

[0412] O/S 하위 보안 에이전트(3516)가 전자 장치(3504) 상에서 실행되는 숨겨진 프로세스의 증거가 존재한다고 판단한 경우, O/S 하위 보안 에이전트(3516)는 운영체제(3512) 및 운영체제 커널 메모리(3380)를 검색하여 그러한 프로세스와 관련하여 어떠한 변형들이 이루어졌는지를 판단할 수 있다. O/S 하위 보안 에이전트(3516)는 악성 소프트웨어에 의해 수행되는 것으로 알려진 어떤 메모리 변형들을 검색할 수 있다. 일 실시예들에서 O/S 하위 보안 에이전트(3516)는 메모리 변형에 대해, 운영체제 코드 섹션(3582) 및 액티브 프로세스 리스트(3584)를 검색할 수 있다. O/S 하위 보안 에이전트(3516)는 운영체제 커널 메모리(3580)에서 발견된 어떤 악의적 변형들을 복구할 수 있다. O/S 하위 보안 에이전트(3516)는 운영체제 커널 메모리(3580)에서의 메모리 변형들에 대한 검사를 통해 판단된 어떤 검출된 루트킷 감염을 제거하거나, 어떤 내부 데이터 구조나 코드 섹션들에 대한 어떤 감염들을 복구할 수 있다. O/S 하위 보안 에이전트(3516)는 O/S 하위 보안 에이전트(3516) 및 보안 장치 드라이버(3570)에 의해 결정된 프로세스들 사이에서 발견된 각각의 차이에 대해 숨겨진 프로세스에 대한 메모리 변형들에 대해 검색하는 프로세스를 반복하도록 구성될 수 있다. O/S 하위 보안 에이전트(3516)는 실행중인 프로세스 리스트(3586)와 같은 최종 프로세스 리스트를 생성하고 보안 장치 드라이버(3570)에 그 리스트를 보낼 수 있다.

[0413] 보안 장치 드라이버(3570)는 보안 DLL(3572)나 보안 DLL(3572)에 대한 참조를 실행중인 프로세스 리스트(3586)와 같은 전자 장치(3504) 상에서 실행중인 각각의 프로세스의 어드레스 공간 안에 주입할 수 있다. 프로세스 어드레스 공간들(3587) 각각은 그에 따라 보안 DLL(3572)에 대한 포인터를 포함할 수 있다. 보안 DLL(3572)은 전자 장치(3504) 상에서 실행되는 모든 사용자 모드 프로세스들을 나열할 수 있다. 보안 DLL(3572)은 그 결과들을 실행중인 프로세스 리스트(3588)에 컴파일하고, 실행중인 프로세스 리스트(3588)를 보안 장치 드라이버(3570)에 보낼 수 있다.

[0414] 보안 장치 드라이버(3570)는 수신된 실행중인 프로세스 리스트(3588)를 O/S 하위 보안 에이전트(3516)으로부터 수신된 실행중인 프로세스 리스트(3586)와 비교할 수 있다. 보안 장치 드라이버(3570)는 그러한 프로세스 리스트들의 어떤 차이가 사용자 모드 루트킷 공격과 같은 악성 소프트웨어 감염을 가리킬 수 있다고 판단할 수 있다. 보안 장치 드라이버(3570)는 프로세스 어드레스 공간들(3587)과 관련된 메모리 안의 프로세스 코드 및 데이터 섹션들을 검사하여 어떤 메모리 변경이 이루어졌는지 여부를 판단하여 필요 시 복구를 행할 수 있다. 보안 장치 드라이버(3570)는 보안 DLL(3572)로부터 실행중인 프로세스 리스트(3588) 및 O/S 하위 보안 에이전트(3516)로부터의 실행중인 프로세스 리스트(3588) 사이에 발견된 차이마다, 프로세스 어드레스 공간들(3587)의 검색, 메모리 변경 검출, 및 그러한 메모리 변경들의 복구 프로세스를 반복할 수 있다.

[0415] 도 36은 전자 장치 상에서 숨겨진 프로세스들을 검출하여 복구하기 위한 방법(3600)의 실시예이다.

[0416] 단계 3605에서 제어 레지스터에 대해 시도된 액세스가 인터셉트되고 기록될 수 있다. 그러한 제어 레지스터는 CR3 제어 레지스터일 수 있다. 시도된 그러한 액세스는 VM exit을 트래핑함으로써 인터셉트될 수 있다. 상기 인터셉트는 가상 머신 제어 구조에서 플래그를 세팅하여 구현될 수 있다. 주어진 시간 동안 제어 레지스터를 액세스하는 모든 프로세스들의 레코드들을 포함할 수 있는 O/S 하위 레벨 프로세스 리스트를 생성하기 위해(단계 3610) 단계 3605가 반복될 수 있다. 가상 메모리를 이용한 시스템에서, 그러한 가상 메모리를 전환 및 액세스하기 위해 제어 레지스터가 액세스될 수 있다. 단계들 3605-3610은 O/S 하위 보안 에이전트의 지원을 통해 구현될 수 있다. 단계들 3605-3610은 방법(3600)에서 이루어진 다양한 비교들을 위한 업데이트된 기준치를 필요 시 제공하기 위해, 방법(3600)의 동작 중에 주기적으로나 필요할 때마다 반복될 수 있다.

[0417] 단계 3615에서 전자 장치의 운영체제 상에서 실행되는 프로세스들이 운영체제의 커널 모드 관점에서 판단될 수 있다. 그러한 프로세스들은 운영체제의 커널 모드의 나열 함수들을 이용해 판단될 수 있다. 예를 들어, 실행중인 프로세스들을 판단하기 위해 운영체제의 액티브 프로세스 리스트가 액세스될 수 있다. 단계 3620에서 그런 프로세스들은 O/S 레벨 프로세스 리스트를 생성하는데 이용될 수 있다. 단계 3625에서 O/S 레벨 프로세스 리스트 내 각각의 리스트의 EPROCESS 구조로부터 제어 레지스터 값들이 산출될 수 있다. 그러한 레지스터 값들

은 해당 프로세스가 O/S 레벨 프로세스 리스트 상에서 상호 참조되는 것을 가능하게 할 수 있다.

[0418] 단계 3630에서 O/S 하위 레벨 및 O/S 레벨 프로세스 리스트들이 비교되어 어떤 차이가 존재하는지 여부를 판단할 수 있다. O/S 레벨 프로세스 리스트에서 누락된 어떤 프로세스들이 O/S 하위 레벨 프로세스 리스트 안에 존재하는 경우, 단계 3635에서 그 프로세스들은 감춰져 있을 수 있고 그에 따라 악성일 수 있다고 판단될 수 있다.

[0419] 단계 3640에서, 운영체제 및 시스템 메모리가 그 숨겨진 프로세스와 관련된 메모리 변경에 대해 검색될 수 있다. 일 실시예에서 그러한 자원들은 프로세스 나열 관련 변경에 대해 검색될 수 있다. 예를 들어 운영체제 코드 섹션들 및/또는 운영체제 액티브 프로세스 리스트가 검색될 수 있다. 단계 3645에서 어떤 검출된 메모리 변경들이 복구될 수 있다. 단계 3650에서, O/S 하위 레벨 프로세스 리스트 및 O/S 레벨 프로세스 리스트의 요소들 사이에 아무 차이가 존재하지 않을 때까지 숨겨진 모든 프로세스들에 대해 단계들 3605-3645이 반복될 수 있다.

[0420] 도 37은 전자 장치 상에서 숨겨진 프로세스들을 검출하여 복구하기 위한 방법(3700)의 실시예이다. 방법(3700)은 방법(3600)이 커널 모드 및 사용자 모드 프로세스들을 둘 다 포함하는 리스트들의 생성과 비교를 수반하는데 반해 방법(3700)은 커널 모드 프로세스 리스트들 및 사용자 모드 프로세스 리스트들에 대해 별개의 생성 및 비교를 수반한다는 점에서 방법(3600)과 다르다. 한 리스트에는 있고 다른 리스트에는 없는 프로세스들의 존재를 비교함으로써, 악성 소프트웨어가 사용자 모드 루트킷인지 커널 모드 루트킷인지와 같은 악성 소프트웨어 프로세스의 질이 판단될 수 있다. 또한 루트킷은 감염된 하나 혹은 여러 개의 프로세스들을 가질 수 있다.

[0421] 단계 3705에서 제어 레지스터에 대해 시도된 액세스가 인터셉트되고 기록될 수 있다. 그러한 제어 레지스터는 CR3 제어 레지스터일 수 있다. 시도된 그러한 액세스는 VM exit을 트래핑함으로써 인터셉트될 수 있다. 상기 인터셉트는 가상 머신 제어 구조에서 플래그를 세팅하여 구현될 수 있다. 주어진 시간 동안 제어 레지스터를 액세스하는 모든 프로세스들의 레코드들을 포함할 수 있는 O/S 하위 레벨 프로세스 리스트를 생성하기 위해(단계 3710) 단계 3705가 반복될 수 있다. 가상 메모리를 이용한 시스템에서, 그러한 가상 메모리를 전환 및 액세스하기 위해 제어 레지스터가 액세스될 수 있다. 단계들 3705-3710은 O/S 하위 보안 에이전트의 지원을 통해 구현될 수 있다. 단계들 3705-3710은 방법(3700)에서 이루어진 다양한 비교들을 위한 업데이트된 기준치를 필요 시 제공하기 위해, 방법(3700)의 동작 중에 주기적으로나 필요할 때마다 반복될 수 있다.

[0422] 단계 3715에서 전자 장치의 운영체제의 커널 모드에서 실행되는 프로세스들이 운영체제의 커널 모드 관점에서 판단될 수 있다. 그러한 프로세스들은 운영체제의 커널 모드의 나열 함수들을 이용해 판단될 수 있다. 예를 들어, 커널 모드에서 실행중인 프로세스들을 판단하기 위해 운영체제의 액티브 프로세스 리스트가 액세스될 수 있다. 단계 3720에서 그런 프로세스들은 O/S 레벨 프로세스 리스트를 생성하는데 이용될 수 있다. 단계 3725에서 O/S 레벨 프로세스 리스트 내 각각의 리스트의 EPROCESS 구조로부터 제어 레지스터 값들이 산출될 수 있다. 그러한 레지스터 값들은 해당 프로세스가 O/S 레벨 프로세스 리스트 상에서 상호 참조되는 것을 가능하게 할 수 있다.

[0423] 단계 3730에서 O/S 하위 레벨 및 O/S 레벨 프로세스 리스트들이 비교되어 어떤 차이가 존재하는지 여부를 판단할 수 있다. O/S 레벨 프로세스 리스트에서 누락된 어떤 프로세스들이 O/S 하위 레벨 프로세스 리스트 안에 존재하는 경우, 단계 3735에서 그 프로세스들은 감춰져 있을 수 있고 그에 따라 악성일 수 있으며 어쩌면 커널 모드 루트킷의 형태를 취할 수 있다고 판단될 수 있다. 일 실시예에서, 이와 달리, O/S 레벨 프로세스 리스트로부터 누락된 프로세스들은 실제로 사용자 모드 프로세스들이라고 판단될 수 있다. 그러한 실시예에서 단계 3715는 전자 장치의 사용자 모드 프로세스들을 나열하지 않았을 수 있다.

[0424] 단계 3740에서, 운영체제 및 시스템 메모리가 그 숨겨진 프로세스와 관련된 메모리 변경에 대해 검색될 수 있다. 일 실시예에서 그러한 자원들은 프로세스 나열 관련 변경에 대해 검색될 수 있다. 예를 들어 운영체제 코드 섹션들 및/또는 운영체제 액티브 프로세스 리스트가 검색될 수 있다. 단계 3745에서 어떤 검출된 메모리 변경들이 복구될 수 있다. 단계 3750에서, O/S 하위 레벨 프로세스 리스트 및 O/S 레벨 프로세스 리스트의 커널 모드 요소들 사이에 아무 차이가 존재하지 않을 때까지 숨겨진 모든 커널 모드 프로세스들에 대해 단계들 3735-3745이 반복될 수 있다.

[0425] 단계 3755에서 전자 장치의 사용자 모드 프로세스들이 결정되어 나열될 수 있다. 단계 3755는 공유 라이브러리를 각각의 실행중인 프로세스의 어드레스 공간 안에 주입함으로써 구현될 수 있다. 공유 라이브러리는 운영체제의 사용자 모드 프로세스 나열 함수들을 호출할 수 있다. 단계 3760에서 사용자 레벨 프로세스 리스트는 단

계 3755의 결과들을 이용하여 생성될 수 있다. 단계 3765에서 O/S 하위 레벨 및 사용자 레벨 프로세스 사이의 차이들이 판단될 수 있다. 사용자 레벨 프로세스 리스트 안에서 발견되지 않고 이전에 O/S 레벨 프로세스 리스트 안에서도 발견되지 않은 어떤 프로세스들은 숨겨진 사용자 모드 프로세스들일 수 있고 그에 따라 악성 소프트웨어와 관련된 것일 수 있다. 단계 3770에서 사용자 모드의 애플리케이션 및 프로세스 공간들이 메모리 변경에 대해 검색될 수 있다. 공유 라이브러리가 주입되었던 어드레스 공간들의 프로세스 코드가 그러한 메모리 변경에 대해 검색될 수 있다. 사용자 모드 프로세스 코드를 호스팅하는 메모리 부분들 및 데이터 섹션들이 단계 3775에서 검색될 수 있다. 단계 3780에서 어떤 검출된 메모리 변경들이 복구될 수 있다. 단계 3785에서, O/S 하위 레벨 프로세스 리스트 및 사용자 레벨 프로세스 리스트의 사용자 모드 요소들 사이에 아무 차이가 존재하지 않을 때까지 단계들 3755-3780이 반복될 수 있다.

[0426] 도 38은 전자 장치(3801) 상에 실행되는 운영체제(3813)의 시스템 호출들에 대한 액세스를 보호하기 위한 시스템(3800)의 실시예이다. 시스템(3800)은 운영체제(3813)와 같은 전자 장치(3801)의 운영체제들에서 실행되는 소프트웨어 기반 개체들로부터 시스템 호출들 및/또는 시스템 호출 테이블을 액세스하려는 악의적 시도들을 검출하기 위해 전자 장치(3801) 상에서 동작하도록 구성된 O/S 하위 트래핑 에이전트(3820) 및 트리거된 이벤트 핸들러(3822)를 포함할 수 있다. 또한, O/S 하위 트래핑 에이전트(3820) 및 트리거된 이벤트 핸들러(3822)는 시스템 호출들 및/또는 시스템 호출 테이블(3804)에 대한 액세스를 언제 트래핑할지와 트래핑된 동작과 관련된 트리거된 이벤트를 어떻게 처리할지를 결정하기 위해 하나 이상의 보안 규칙들(3808)을 사용하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(3820) 및 트리거된 이벤트 핸들러(3822)는 트리거된 이벤트를 허용하거나 거부하거나 다른 교정 액션을 수행하도록 구성될 수 있다.

[0427] 전자 장치(3801)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701), 도 9의 전자 장치(901), 도 12의 전자 장치(12) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(3801)는 메모리(3803)에 연결된 하나 이상의 프로세서들(3802)을 포함할 수 있다. 프로세서(3802)는 도 2의 프로세서(208), 도 4의 프로세서(408), 도 7의 프로세서(702), 도 9의 프로세서(902), 도 12의 프로세서(1202) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(3803)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703), 도 9의 메모리(903), 도 12의 메모리(1203) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(3801)는 시스템 호출 테이블(3804), 가상 메모리 페이지 테이블(3806) 및 O/S 내 보안 에이전트(3819)를 포함할 수 있는 운영체제(3813)를 포함할 수 있다. 운영체제(3813)는 도 1의 운영체제들(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 9의 운영체제(913), 도 12의 운영체제(1213) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(3819)는 도 1의 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418), 도 7의 O/S 내 보안 에이전트(719), 도 9의 O/S 내 보안 에이전트(919), 도 12의 O/S 내 보안 에이전트(1219) 및/또는 이들의 어떤 적절한 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 보안 규칙들(3808)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(220, 222), 도 4의 보안 규칙들(420, 434, 436, 438), 도 5의 보안 규칙들(518), 도 7의 보안 규칙들(707, 721, 723), 도 9의 보안 규칙들(908, 921), 도 12의 보안 규칙들(1208, 1221) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보호 서버(3818)는 도 1의 보호 서버(102), 도 2의 보호 서버(202) 및/또는 이들의 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0428] O/S 하위 트래핑 에이전트(3820)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442) 및/또는 PC 펌웨어 보안 에이전트(444), 도 5의 펌웨어 보안 에이전트(516) 및/또는 도 7의 마이크로코드 보안 에이전트(708), 도 9의 O/S 하위 에이전트(920), 도 12의 O/S 하위 보안 에이전트(1220) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 트리거된 이벤트 핸들러(3822)는 도 1의 트리거된 이벤트 핸들러(108), 도 2의 SVMM 보안 에이전트(217), 도 4의 O/S 하위 에이전트(450), 도 7의 O/S 하위 에이전트(712), 도 9의 트리거된 이벤트 핸들러(922) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 다양한 실시예들에서 O/S 하위 트래핑 에이전트(3820)의 기능 중 일부는 트리거된 이벤트 핸들러(3822)에 의해 수행될 수 있고/있거나, 트리거된 이벤트 핸들러(3822)의 기능 중 일부가 O/S 하위 트래핑 에이전트(3820)에 의해 수행될 수 있다. 또한 O/S 하위 트래핑 에이전트(3820) 및 트리거된 이벤트 핸들러(3822)는 동일한 소프트웨어 모듈 안에서 구현될

수 있다.

[0429] 페이지 테이블(3806)이 데이터 구조로서 구현될 수 있고, 운영체제(3813)의 가상 메모리 시스템을 구현하기 위해 사용될 수 있다. 가상 메모리 시스템은 전자 장치(3801)의 메모리(3803)에 대한 액세스를 가상화하는 메모리 관리 시스템이다. 가상 메모리 시스템에서, 운영체제(3813) 상에서 실행되는 소프트웨어 프로세스들이 프로세스가 메모리의 인접 블록으로서 취급할 수 있는 가상 어드레스 공간과 함께 제공된다. 실제로, 가상 어드레스 공간은 물리적 메모리의 다양한 영역들에 걸쳐 분산될 수 있다. 프로세스가 메모리에 대한 액세스를 요청할 때, 운영체제(3813)는 해당 프로세스의 가상 어드레스를 데이터가 실제로 저장된 메모리(3803) 내 물리적 어드레스로 매핑하는 일을 담당할 수 있다. 가상 액세스 공간은 가상 메모리 페이지들이라 불리는 인접하는 가상 메모리 어드레스들의 고정 크기 블록들로 분할될 수 있다. 페이지 테이블(3806)은 가상 메모리 페이지로부터 그 가상 메모리 페이지가 저장된 메모리(3803) 내 대응하는 물리적 어드레스로의 매핑을 저장하는 데 사용될 수 있다. 페이지 테이블(3806)은 주어진 가상 메모리 페이지에 대해 허가된 액세스 타입을 특정하기 위해, 읽기, 쓰기 및/또는 실행과 같은 다양한 액세스 허가 사항들을 포함할 수 있다. 어떤 실시예들에서 O/S 하위 트래핑 에이전트(3820) 및/또는 트리거된 이벤트 핸들러(3822)는 어떤 발생된 예외들이나 시도된 읽기, 쓰기 또는 실행 동작들을 캐치하고, 메모리(3803)를 액세스하라는 허가되지 않은 요청이 악성 소프트웨어를 나타내는지를 판단하기 위해 보안 규칙들(3808)을 이용하도록 구성될 수 있다.

[0430] 시스템 호출 테이블(3804)은 시스템 호출들을 구현하기 위해 운영체제(3813)에 의해 사용되는 데이터 구조일 수 있다. 시스템 호출은 운영체제(3813)에 의해 제공되는 루틴 및/또는 시스템 서비스일 수 있다. 시스템 호출 테이블(3804)은 애플리케이션(3810) 및 운영체제(3813) 사이의 인터페이스를 제공하여, 애플리케이션(3810)이 애플리케이션(3810)이 수행하는 것이 허가되지 않을 동작을 수행하도록 운영체제(3813)에 요청하는 것을 허용할 수 있다. 각각의 시스템 호출은 특정 시스템 호출에 대한 엔트리가 저장될 수 있는 시스템 호출 테이블(3804)로의 인덱스를 이용하여 식별될 수 있다. 시스템 호출 테이블(3804)의 각각의 엔트리는 특정 시스템 호출에 대응하는 코드가 저장될 수 있는 메모리(3803) 내 어드레스를 저장할 수 있다. 그런 엔트리들은 포인터들로서 구현될 수 있다. 시스템 호출은 운영체제(3813)로 적합한 인덱스를 알리고 이어서 운영체제(3813)로 제어를 넘김으로써 실행될 수 있다. 운영체제(3813)는 이때 시스템 호출 테이블(3804)을 조회하여 특정 시스템 호출에 대응하는 코드가 저장될 수 있는 메모리(3803) 내 위치를 식별할 수 있다. 운영체제(3813)는 여기서 상기 코드를 실행하고 시스템 호출을 요청하는 일을 담당하는 소프트웨어 구성요소로 제어를 돌려보낼 수 있다. 시스템 호출 테이블(3804)의 실시예들에 대한 설명은 이하의 도 39의 시스템 호출 테이블(3901)에 대한 논의들에서 찾아볼 수 있다.

[0431] O/S 하위 트래핑 에이전트(3820)는 메모리(3803) 및/또는 프로세서(3802)와 같이 시스템 호출들과 관련된 어떤 적절한 자원(3816)으로의 액세스나 그로부터 나온 정보를 인터셉트하도록 구성될 수 있다. 예를 들어 자원들(3816)는 도 1의 자원(106), 도 2의 시스템 자원들(214), 도 7의 시스템 자원들, 도 9의 프로세서 자원들(924), 도 12의 가상 메모리(1204) 및/또는 물리적 메모리(1203) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 자원들(3816)은 프로세서가 명령어들을 로딩하고 실행할 수 있게 하기 위해 프로세서(3802)와 같은 프로세서가 이용 가능한 자원들을 포함할 수 있다. 그러한 자원들은 예컨대 데이터 레지스터들, 제어 레지스터들, 캐시들, 프로세서 플래그들, 프로세서 코어들, 프로세서 예외들, 및/또는 프로세서 인터럽트들을 포함할 수 있다. 자원들(3816)은 또한 가상 및/또한 물리적 메모리(3803)를 포함할 수 있다. 그러한 자원에 대해 시도되는 액세스는 피연산자들을 사용하는 어셈블리 언어 명령어와 같은 명령어를 포함할 수 있고, 시도되는 그러한 액세스는 해당 명령어의 실행을 트래핑함으로써 트래핑될 수 있다.

[0432] O/S 하위 트래핑 에이전트(3820)는 메모리(3803) 및/또는 프로세서(3802)의 자원들과 같은 어떤 적절한 자원에 대한 액세스나 그로부터의 정보를 인터셉트하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(3820)는 시스템 호출들 및/또는 시스템 호출 테이블(3804)을 액세스하는 시도들을 트래핑하는 데 사용될 수 있는 시스템 호출 트래퍼(3814)를 포함할 수 있다. 시스템 호출 트래퍼(3814) 및/또는 트리거된 이벤트 핸들러(3822)는 여기서 보안 규칙들(3808)과 공동으로 트래핑된 시도와 연관된 정황 정보를 이용하여, 그 시도를 허용, 거부, 및/또는 하나 이상의 가입자들(3812)에게 보고할지 여부를 판단할 수 있다. 정황 정보는 트래핑된 액세스 시도의 요청 개체, 문제의 특정 시스템 호출 및/또는 시도된 액세스의 특정 타입(가령, 시스템 호출을 실행하는 시도 또는 시스템 호출 테이블의 어느 엔트리에 대한 읽기/쓰기 시도)을 포함할 수 있다.

[0433] 시스템 호출 트래퍼(3814)는 O/S 하위 트래핑 에이전트(3820)의 모듈 및/또는 구성요소일 수 있으며, 어떤 적절한 방식에 따라 시스템 호출들에 대한 액세스를 트래핑하도록 구성될 수 있다. 예를 들어 시스템 호출 트래퍼(3814)는 시스템 호출의 실행을 위해 제어를 운영체제(3813)로 넘기는 명령어와 같이, 시스템 호출들을 구현하

는 데 사용되는 어셈블리 언어 명령어의 실행을 트래핑하도록 구성될 수 있다. 트래핑할 특정 명령어는 전자 장치(3801)의 특정 프로세서(3802) 및/또는 운영체제(3813)에 좌우될 수 있다. 일례로서, x86 명령어 세트 구조("ISA")를 지원하는 프로세서(3802) 상에서 실행되는 마이크로소프트 윈도우즈의 어떤 변형을 이용할 때, 시스템 호출 트래퍼(3814)는 'SysEnter' 및/또는 'KiFastSysCall' 명령어들을 실행하려는 시도들을 트래핑할 수 있다. 이러한 명령어들은 시스템 호출을 실행하기 위해 제어를 운영체제(3813)에게 넘기도록 기능한다. 'SysEnter' 명령어를 실행하려는 시도들의 트래핑은 "Ring3" 우선순위에서 실행되는 소프트웨어로부터의 시도들만을 트래핑하는 한편, 'KiFastSysCall' 명령어를 실행하려는 시도들의 트래핑은 "Ring0"나 "Ring3" 우선순위에서 실행되는 소프트웨어로부터의 시도들을 트래핑할 수 있다. 어떤 실시예들에서 'SysEnter' 및/또는 'KiFastSysCall' 명령어들을 실행하려는 시도들은 그 특정 명령어들이 저장될 수 있는 물리적 메모리 위치에 대응하는 가상 메모리 페이지를 실행하고자 하는 시도를 트래핑함으로써 트래핑될 수 있다.

[0434] 다른 실시예에서 시스템 호출 트래퍼(3814)는 시스템 호출 테이블(3804)을 액세스하려는 시도들을 트래핑하도록 구성될 수 있다. 시스템 호출 테이블(3804)를 액세스하려는 시도를 트래핑하는 데 사용되는 특정 방법은 전자 장치(3801)의 특정 프로세서(3802) 및/또는 운영체제(3813)에 좌우될 수 있다. x86 ISA를 지원하는 프로세서(3802)를 사용할 때, 시스템 호출 트래퍼(3814)는 MOV 명령어를 실행하려는 어떤 시도들을 트래핑함으로써 시스템 호출 테이블(3804)로의 쓰기나 읽기 시도들을 트래핑할 수 있다. 예를 들어 시스템 호출 트래퍼(3814)는 "MOV syscall_table_address, EAX" 명령어를 트래핑함으로써 시스템 호출 테이블(3804)에 대한 쓰기 시도들을 트래핑할 수 있다. 이 명령어는 EAX 레지스터로부터의 값을 syscall_table_address로 특정된 메모리 어드레스에 있는 시스템 호출 테이블 안에 기입하는 동작을 할 수 있다. 시스템 호출 테이블(3804)로의 쓰기 시도들을 트래핑함으로써, 시스템 호출 트래퍼(3814)는 악성 소프트웨어가 악성 코드를 포함하는 메모리 어드레스를 가지고 시스템 호출 테이블(3804) 안의 엔트리를 덮어쓰는 것을 막을 수 있다. 마찬가지로, 시스템 호출 트래퍼(3814)는 "MOV EAX, syscall_table_address" 명령어를 트래핑함으로써 시스템 호출 테이블(3804)로부터의 읽기 시도들을 트래핑할 수 있다. 이 명령어는 syscall_table_address에 의해 특정된 메모리 어드레스에 있는 시스템 호출 테이블의 어떤 엔트리로부터의 값을 판독하는 동작을 할 수 있다. 운영체제(3813)가 시스템 호출에 해당하는 코드의 메모리 내 위치를 식별할 수 있게 하려면 시스템 호출 테이블(3804)이 읽혀져야 하므로, 시스템 호출 테이블(3804) 안의 엔트리를 읽으려는 시도들의 트래핑은 시스템 호출 테이블(3804) 안의 엔트리와 관련된 시스템 호출을 실행하려는 모든 시도들을 효과적으로 트래핑할 것이다. 또한 시스템 호출 테이블(3804) 내 엔트리를 읽으려는 시도들의 트래핑은 악성 소프트웨어와 같이 시스템 호출 테이블(3804)을 읽으려는 어떤 직접적 시도를 트래핑할 것이다. 어떤 실시예들에서 시스템 호출 테이블(3804)를 읽으려는 모든 시도들이 트래핑된다. 어떤 실시예들에서 시스템 호출 테이블(3804)을 액세스하려는 시도는 시스템 호출 테이블(3804)이 저장될 수 있는 물리적 메모리 위치에 대응하는 가상 메모리 페이지를 액세스하려는 시도를 트래핑함으로써 트래핑될 수 있다.

[0435] 또 다른 실시예에서 시스템 호출 트래퍼(3814)는 시스템 호출을 위한 코드가 상주하는 메모리 위치에 있는 코드를 실행하려는 시도를 트래핑함으로써 시스템 호출을 실행하려는 시도를 트래핑하도록 구성될 수 있다. 시스템 호출 테이블(3804)은 특정 시스템 호출을 위한 코드가 상주하는 메모리 위치를 식별하기 위해 조회될 수 있다. 시스템 호출을 실행하려는 시도를 트래핑하는데 사용되는 특정 방법은 전자 장치(3801) 내 프로세서(3802)의 타입에 따라 달라질 수 있다. 일 실시예에서, 시스템 호출을 실행하려는 시도는 예컨대 명령어 포인터(IP) 레지스터의 값에 기반하는 트리거를 이용하여 트래핑될 수 있다. 어떤 실시예들에서 IP 레지스터는 프로그램 카운터(PC) 레지스터로서 알려질 수 있다. IP 레지스터는 특정 프로세서에 따라, 현재 실행되고 있는 명령어의 어드레스나 실행될 다음 명령어의 어드레스를 저장하는데 사용될 수 있다. x86 ISA를 지원하는 프로세서(3802)를 사용할 때, 시스템 호출 트래퍼(3814)는 IP 레지스터의 값을 감시하고 IP 레지스터의 값이 시스템 호출의 어드레스를 포함할 때 실행을 트래핑함으로써, 특정 시스템을 실행하려는 시도들을 트래핑할 수 있다. 다른 실시예들에서 시스템 호출을 실행하려는 시도는 그 시스템 호출의 코드가 저장될 수 있는 물리적 메모리 위치에 대응하는 가상 메모리 페이지를 실행하고자 하는 시도를 트래핑함으로써 트래핑될 수 있다. 시스템 호출 테이블(3804)은 트래핑될 특정 시스템 호출에 해당하는 인덱스로 조회되어 그 시스템 호출에 대응하는 코드의 메모리 내 위치를 식별하도록 할 수 있다. 일 실시예에서, 시스템 호출 테이블(3804)에 안에서 링크된 루틴들 및 함수들을 포함하는 메모리 위치들의 실행은 시스템 호출 테이블(3804) 안에 있는 것으로 알려진 위치를 향하는 "JMP" 루틴과 같이 제어를 옮기는 명령어를 트래핑함으로써 트래핑될 수 있다.

[0436] 시스템 호출들 및 시스템 호출 테이블(3804)에 대한 액세스를 트래핑하기 위한 상기 방법들의 특정 구현은 O/S 하위 트래핑 에이전트(3820) 및/또는 시스템 호출 트래퍼(3814)의 특정 구현에 따라 달라질 수 있다. 예를 들어 O/S 하위 트래핑 에이전트(3820) 및/또는 시스템 호출 트래퍼(3814)가 가상 머신 모니터에서 구현되는 경우,

특정 메모리 어드레스에 기반하는 어떤 트래핑(가령, 시스템 호출 테이블 안의 어느 엔트리로의 쓰기/읽기 시도에 기반하는 트래핑 및/또는 시스템 호출의 코드를 포함하는 메모리 위치에서의 실행에 기반하는 트래핑)은 가상 메모리 어드레스에 기반할 수 있는데, 이는 메모리 어드레스가 가상 메모리 어드레스에서 물리적 메모리 어드레스로 변환되지 않았을 것이기 때문이다. 다른 예로서, O/S 하위 트래핑 에이전트(3820) 및/또는 시스템 호출 트래퍼(3814)이 마이크로코드 보안 에이전트에서 구현되면, 특정 메모리 어드레스에 기반하는 어떤 트래핑은 물리적 메모리 어드레스에 기반할 수 있는데, 이는 가상 어드레스로부터 물리적 어드레스로의 변환이 이미 마이크로코드 레벨에서 수행되었을 것이기 때문이다.

[0437] 시스템 호출 테이블(3804)을 액세스하려고 시도하는 명령어나 시스템 호출 테이블(3804)과 관련된 함수들의 메모리 위치는 그러한 시도의 트래핑 중에 검출될 수 있다. 메모리 위치는 해당 시도를 한 개체를 판단하기 위해 분석될 수 있다.

[0438] 특정 시도를 트래핑한 후, 시스템 호출 트래퍼(3814)는 해당 시도와 관련된 트리거된 이벤트를 생성하여 트리거된 이벤트 핸들러(3822)로 보낼 수 있다. 트리거된 이벤트 핸들러는 여기서 보안 규칙들(3808)과 공동으로 트래핑된 이벤트와 연관된 정황 정보를 이용하여, 그 이벤트를 허용, 거부, 및/또는 하나 이상의 가입자들(3812)에게 보고할지 여부를 판단할 수 있다. 정황 정보는 트래핑된 이벤트의 요청 개체, 문제의 특정 시스템 호출 및/또는 요청된 액션(가령, 시스템 호출을 실행하는 시도 또는 시스템 호출 테이블(3804)의 어느 엔트리에 대한 읽기/쓰기 시도)을 포함할 수 있다. 예를 들어 안전하고 악성 소프트웨어 염려가 없다고 알려진 어떤 개체들만이 시스템 호출 테이블(3804)로의 쓰기가 허용될 수 있다. 악성 소프트웨어 상태가 앞서 알려지지 않은 개체는 시스템 호출 테이블(3804)로의 허가되지 않은 쓰기 시도에 기반하여 악성 소프트웨어라고 판단될 수 있다. 다른 예에서, 시스템 호출 테이블(3804) 안에 링크된 함수들의 실행이 트래핑될 수 있고, 함수에 대해 호출을 수행한 개체가 시스템 호출 테이블(3804)을 통해 그러한 시도를 수행하였을 경우에만 실행이 허용될 수 있다. 악성 소프트웨어 상태가 앞서 알려지지 않았던 개체에 의한 시도는 그러한 함수들을 실행하는 것에 대한 직접 액세스가 거부될 수 있다. 또한 블랙리스트나 다른 판단을 통해 악성 소프트웨어라고 판단된 개체로서, 시스템 호출 테이블(3804)이나 그 관련 함수들을 액세스하려고 시도하는 개체는 액세스가 거부될 수 있고 다른 적절한 교정 액션이 수행될 수 있다.

[0439] 가입자들(3812)은 시스템 호출 테이블(3804)을 액세스하려는 트래핑된 시도 및/또는 시스템 호출을 실행하려는 트래핑된 시도와 관련된 정보를 위해 항상 사용하는 어떤 개체들을 포함할 수 있다. 가입자들(3812)은 전자 장치(3801) 상의 애플리케이션들(3810) 및/또는 보안 에이전트들을 포함할 수 있고/있거나 제3자 애플리케이션들이나 다른 소프트웨어를 포함할 수 있다. 예를 들어 가입자(3812)는 O/S 하위 트래핑 에이전트(3820), 트리거된 이벤트 핸들러(3822) 및/또는 O/S 내 보안 에이전트(3819)와 같이 악성 소프트웨어 검출을 위해 트래핑된 시도와 관련된 정황 정보를 이용할 수 있는 전자 장치(3801) 상에서 실행되는 보안 소프트웨어를 포함할 수 있다. 일부 실시예들에서 각각의 가입자(3812)는 예컨대 펌웨어 보안 에이전트와 같이 O/S 하위 트래핑 에이전트(3820)와 동일한 우선순위 레벨에서 동작하는 그 자신의 보안 에이전트를 제공할 수 있다. 가입자(3812)는 또한 보호 서버(3818)와 같이 원격으로 실행되는 보안 소프트웨어를 포함할 수 있다. 다른 예로서, 가입자들(3812)은 전자 장치(3801)에 의해 사용되는 I/O 장치와 같은 특정 자원의 제조자를 포함할 수 있다. 제조자는 시스템 호출 테이블에 대한 액세스 및/또는 자원과 관련된 시스템 호출과 같은 시스템 호출의 실행을 통해 자원을 훼손시키려는 어떤 의심스러운 시도들에 관심이 있을 수 있다. 다른 예로서, 가입자들(3812)은 DRM(digital rights management) 시스템의 관리자를 포함할 수 있다. DRM 시스템은 디지털 콘텐츠의 사용을 제한 및/또는 통제할 수 있고, 일반적으로 비디오 및/또는 음악 콘텐츠와 같이 저작권으로 보호되는 디지털 콘텐츠를 보호하는 데 사용된다. DRM 시스템의 관리자는 다양한 디지털 보호 파일들이 언제 어떻게 액세스되는지를 아는 데 관심이 있을 수 있으며, 이것은 보호 파일들을 액세스하는 데 사용될 수 있는 다양한 시스템 호출들을 추적함으로써 만족될 수 있다. 애플리케이션 프로그래밍 인터페이스("API")가 가입자들(3812)에게 제공되어, 가입자들(3812)이 시스템 호출을 실행하려는 트래핑된 시도 및/또는 시스템 호출 테이블(3804)을 액세스하려는 트래핑된 시도와 관련된 정보를 액세스하는 것을 가능하게 할 수 있다.

[0440] 도 39는 운영체제의 시스템 호출들에 대한 액세스를 보호하기 위한 시스템 및/또는 방법과 함께 사용할 시스템 호출 테이블(3901)의 실시예이다. 시스템 호출 테이블(3901)은 각각의 시스템 호출(3904)을 위한 코드가 상주하는 메모리(3908) 내 어드레스들(3906)을 저장하기 위해 운영체제에 의해 사용될 수 있다. 시스템 호출 테이블(3901)은 예컨대 도 38의 시스템 호출 테이블(3804)의 기능을 구현하는데 사용될 수 있다. 시스템 호출 테이블(3901)은 테이블, 레코드 및/또는 다른 적절한 데이터 구조에 의해 구현될 수 있다. 마이크로소프트 윈도우즈 운영체제의 어떤 변형을 이용한 실시예들에서, 시스템 호출 테이블(3901)은 시스템 서비스 서술자 테이블

("SSDT")에 의해 구현될 수 있다. 시스템 호출(3904)은 운영체제에 의해 제공되는 루틴 및/또는 시스템 서비스일 수 있다. 통상적 시스템 호출들(3904)은 예컨대 파일들을 조작 및/또는 실행하기 위한 열기, 읽기, 쓰기, 닫기 및/또는 실행, 새로운 프로세스를 생성하기 위한 ntCreateProcess 및/또는 새로운 드라이버를 로딩하기 위한 ntLoadDriver 및 ZwLoadDriver를 포함할 수 있다.

[0441] 시스템 호출(3904)은 애플리케이션 및 운영체제 사이의 인터페이스를 제공하여, 애플리케이션으로 하여금 애플리케이션이 수행하는 것이 허가되지 않을 동작을 수행하도록 운영체제에 요청하는 것을 허용할 수 있다. 예를 들어 통상적으로 "Ring3" 우선순위를 실행하는 애플리케이션은 디스크 상의 파일을 액세스해야 하지만 디스크 I/O 동작을 수행할 허가권을 가지지 않을 수 있다. 상기 애플리케이션은 운영체제가 그 애플리케이션으로부터의 요청을 만족시킬 수 있게 하기 위해 운영체제로 제어를 넘기기 위해 읽기나 쓰기 파일 시스템 호출과 같은 시스템 호출(3904)을 이용할 수 있다. "Ring0" 우선순위에서 실행될 수 있는 운영체제는 특정 시스템 호출(3904)과 관련된 서비스를 제공할 수 있고, 그런 다음 다시 애플리케이션으로 제어를 넘길 수 있다. 예를 들어 운영체제는 시스템 호출(3904)에 대응하는 코드가 위치된 메모리 어드레스(3906)를 식별하기 위해 시스템 호출 테이블(3901)을 액세스할 수 있다. 운영체제는 그런 다음, 메모리(3908) 안의 특정 어드레스(3906)에 있는 코드를 실행할 수 있고 그런 다음 제어를 다시 애플리케이션으로 넘길 수 있다. 이런 방식으로 애플리케이션은 통상적으로 운영체제와 같이 "Ring0" 우선순위에서 실행되는 소프트웨어에서만 이용 가능한 소정 서비스들을 이용할 수 있다.

[0442] 각각의 시스템 호출(3904)은 시스템 호출(3904)에 대한 엔트리가 저장되는 시스템 호출 테이블(3901)로의 인덱스(3902)를 이용하여 조회될 수 있다. 예를 들어 시스템 호출 테이블(3901)은 총 N 개의 엔트리들을 가지며, 각각의 엔트리는 0에서 N-1까지의 범위를 가진 인덱스(3902)를 이용하여 조회될 수 있다. 시스템 호출(3904)은 운영체제로 적합한 인덱스(3902)를 알리고 운영체제(3813)로 제어를 넘김으로써 실행될 수 있다. 일부 실시예들에서 소프트웨어 구성요소는 적합한 인덱스(3902)를 프로세서의 레지스터 안에 놓음으로써 특정할 수 있고, 그런 다음 시스템 호출(3904)의 실행을 위해 운영체제로 제어를 넘기는 명령어를 실행할 수 있다. 예를 들어, x86 명령어 세트 구조("ISA")를 사용하는 일 실시예에서, 아래의 명령어들이 애플리케이션에 대한 시스템 호출들을 구현하는 데 사용될 수 있다:

[0443] "MOV EAX, index"

[0444] "SysEnter"

[0445] 첫 번째 명령은 'index'를 프로세서의 EAX 레지스터 안으로 이동시키며, 여기서 'index'는 특정 시스템 호출(3904)의 엔트리가 상주하는 시스템 호출 테이블(3901) 안의 인덱스(3902)에 해당하는 정수이다. 'SysEnter' 명령어가 이제 제어를 운영체제로 넘기고, 운영체제는 EAX 레지스터에 특정된 인덱스(3902)로 시스템 호출 테이블(3901)을 액세스할 수 있다. 시스템 호출 테이블(3901)의 특정 인덱스(3902)에 있는 엔트리는 특정 시스템 호출(3904)에 대한 코드가 상주하는 메모리(3908) 내 위치를 가리키는 메모리 어드레스(3906)를 특정할 수 있다. 그러면 프로세서가 메모리(3908) 내 특정된 어드레스(3906)에 위치한 코드를 실행할 수 있다. 시스템 호출들(3904)은 애플리케이션들, 운영체제들 및/또는 드라이버들을 포함하는 어떤 소프트웨어 구성요소에 의해 실행될 수 있다. x86 ISA 상의 한 예로서, 운영체제 및/또는 드라이버가 'KiFastSysCall' 명령어를 이용하는 것을 제외하면 애플리케이션과 비슷한 방식으로 시스템 호출들(3904)을 실행할 수 있다.

[0446] 시스템 호출들(3904)이 시스템 호출 테이블(3901)에 추가 및/또는 그로부터 제거될 수 있다. 예를 들어 새로운 장치가 전자 장치에 추가되는 경우, 새로운 장치의 장치 드라이버가 운영체제에 의해 로딩되어야 할 수 있고, 애플리케이션들이 새로운 장치의 기능을 활용할 수 있게 하기 위해 시스템 호출(3904)이 시스템 호출 테이블(3901)로 추가되어야 할 수 있다. 새로운 시스템 호출을 위한 코드가 메모리(3908) 안에 로딩될 수 있고, 새로운 시스템 호출(3904)을 위한 엔트리가 시스템 호출 테이블(3901) 끝에 추가되어 시스템 콜을 위한 코드가 상주하는 메모리(3908) 내 어드레스(3906)를 특정할 수 있다.

[0447] 시스템 호출들(3904)을 구현하기 위한 상술한 실시예들은 많은 가능한 실시예들 중 단지 일부이다. 시스템 호출들(3904) 및/또는 시스템 호출 테이블(3901)은 어떤 적절한 방식에 따라 구현될 수 있다. 시스템 호출들(3904) 및/또는 시스템 호출 테이블(3901)의 특정한 구현은 전자 장치의 특정 프로세서 및/또는 운영체제에 좌우될 수 있다.

[0448] 도 40은 전자 장치 상에 실행되는 운영체제의 시스템 호출들에 대한 액세스를 보호하기 위한 방법(4000)의 실시예이다. 단계 4005에서, O/S 하위 보안 에이전트, O/S 내 보안 에이전트, 트리거된 이벤트 핸들러 및 보호 서

버의 식별 및 보안이 인증될 수 있다. 그러한 인증은 암호화 해싱 및/또는 비밀 키들을 이용하여 각각의 구성 요소에 대한 메모리 내 이미지들을 찾아 검증하는 것을 포함하는 어떤 적절한 방법을 이용하여 수행될 수 있다. 단계 4005가 완료될 때까지 다른 단계들의 동작은 보류될 수 있다. 단계 4010에서 보안 규칙들이 얻어진다. 보안 규칙들은 O/S 하위 보안 에이전트, O/S 내 보안 에이전트 및/또는 트리거된 이벤트 핸들러에 의해 내부적으로 저장되고/거나, 예컨대 보호 서버 상에 원격으로 저장될 수 있다. 그러한 보안 규칙들은 단계들 4015-4040에서 결정을 내리는 데 사용될 수 있다.

[0449] 단계 4015에서 시스템 호출을 실행하려는 시도 및/또는 시스템 호출 테이블을 액세스하려는 시도가 인터셉트될 수 있다. 일부 실시예들에서, 시스템 호출을 실행하려는 시도는 시스템 호출들을 구현하는 데 사용되는 제어 이동 명령어를 실행하려는 시도를 트래핑함으로써 인터셉트될 수 있다. 예를 들어 어떤 프로세스들 및/또는 운영체제들이 SysEnter 및/또는 KiFastSysCall 명령어와 같은 제어 이동 명령어를 이용하여 시스템 호출들을 구현할 수 있고, 시스템 호출을 실행하려는 시도는 적합한 제어 이동 명령의 실행을 트래핑함으로써 인터셉트될 수 있다. 시스템 호출을 실행하려는 시도 역시, 특정 제어 이동 명령어들이 저장될 수 있는 물리적 메모리 위치에 대응하는 가상 메모리 페이지를 실행하고자 하는 시도를 트래핑함으로써 인터셉트될 수 있다. 일부 실시예들에서, 시스템 호출을 실행하려는 시도는 시스템 호출의 코드를 포함하는 메모리 위치에 있는 코드를 실행하려는 시도를 트래핑함으로써 인터셉트될 수 있다. 그러한 실시예들에서 트래핑은 IP 레지스터의 값에 기반할 수 있다. 예를 들어 시스템 호출을 위한 코드를 포함하는 메모리 위치를 식별하기 위해 시스템 호출 테이블이나 메모리 맵이 조회될 수 있고, IP 레지스터가 특정 시스템 호출에 대한 메모리 위치의 어드레스를 포함할 때 트래핑이 일어날 수 있다. 다른 실시예들에서 시스템 호출을 실행하려는 시도는 특정 시스템 호출의 코드가 저장될 수 있는 물리적 메모리 위치에 대응하는 가상 메모리 페이지를 실행하고자 하는 시도를 트래핑함으로써 인터셉트될 수 있다. 일부 실시예들에서 시스템 호출 테이블을 읽거나 쓰고자 하는 시도 역시 인터셉트될 수 있다. 그러한 실시예들에서 시스템 호출 테이블 내 메모리 위치에 쓰거나 읽는데 사용되는 명령어의 실행을 트래핑함으로써 해당 시도가 인터셉트될 수 있다. 예를 들어 x86 명령어 세트 구조 상에서, MOV 명령어가 시스템 호출 테이블 내 위치로 쓰거나 읽는 데 사용될 때 그 명령어가 트래핑될 수 있다. 시스템 호출 테이블에 쓰거나 읽으려는 시도는 일부 실시예들에서, 시스템 호출 테이블이 저장될 수 있는 물리적 메모리 위치에 대응하는 가상 메모리 페이지를 액세스하려는 시도를 트래핑함으로써 인터셉트될 수 있다.

[0450] 단계 4020에서 시도된 액세스의 소스가 식별된다. 예를 들어 시도된 액세스가 애플리케이션, 드라이버, O/S 내 보안 에이전트, 운영체제 및/또는 다른 소프트웨어 개체로부터 나올 수 있다. 단계 4025에서 해당 시도가 허가되는지 여부가 판단된다. 특정 시도가 허가될 수 있는지 없는지 여부를 판단하기 위해 그 시도와 관련된 정황 정보와 함께 보안 규칙들이 사용될 수 있다. 정황 정보는 시도된 액세스의 소스 및/또는 특정 액세스 타입을 포함할 수 있다. 예를 들어 보안 규칙들은 운영체제만이 시스템 호출 테이블에 기입할 수 있다고 특정할 수 있다. 다른 예로서, 보안 규칙들은 어떤 엔트리와 관련된 서명된 드라이버나 다른 소프트웨어 구성요소가 그 자신의 엔트리에 기입할 수 있다는 것을 특정할 수 있다. 해당 시도가 허가되면, 단계 4030에서 액세스가 허용된다. 해당 시도가 허가되지 않으면, 단계 4035에서 액세스가 거부된다. 마지막으로 단계 4040에서, 해당 시도가 하나 이상의 가입자들에게 보고되어야 하는지 여부가 판단된다. 해당 시도가 보고되어야 하는지 여부는 문제의 특정 시스템 호출 및 시도된 액세스와 관련된 정황 정보에 달려있을 수 있다. 보안 규칙들은 시스템 호출을 실행 및/또는 시스템 호출 테이블을 액세스하려는 시도가 하나 이상의 가입자들에게 언제 보고되어야 하는지를 특정할 수 있다.

[0451] 도 40으로부터의 방법의 단계들은 필요 시, 지속적으로나 주기적으로나 요청에 따르거나 어떤 이벤트의 트리거 시, 전자 장치를 보호하기 위해 반복될 수 있다.

[0452] 도 41은 전자 장치(4104) 상의 악성 코드 또는 잠재적 악성 코드의 규제 및 통제를 위한 시스템(4100)의 실시예이다. 예를 들어 시스템(4100)은 전자 장치(4104) 상의 자가 변형 코드의 규제 및 통제를 위해 사용될 수 있다. 시스템(4100)은 검출을 피하기 위해 스스로를 자가 변형하려는 악성 소프트웨어의 시도들에 대해 보호하기 위해 전자 장치(4104) 상에서 동작하도록 구성된 O/S 하위 보안 에이전트(4116)를 포함할 수 있다. 다른 예로서 시스템(4100)은 전자 장치(4104) 상의 악성 코드를 변경하기 위해 사용될 수 있다. 시스템(4100)은 검출된 악성 소프트웨어를 중립화하기 위해 악성 코드를 변경하도록 전자 장치(4104) 상에서 동작하도록 구성된 O/S 하위 보안 에이전트(4116)를 포함할 수 있다. 또 다른 예로서, 시스템(4100)은 잠정적 악성 코드를 포함할 수 있는 스레드 패밀리들을 식별하기 위해 스레드들을 감시하고 추적하기 위해 사용될 수 있다. 시스템(4100)은 스레드들 사이의 관계를 감시하고 추적하도록 전자 장치(4104) 상에서 동작하도록 구성된 O/S 하위 보안 에이전트(4116)를 포함할 수 있다.

[0453] 또한 O/S 하위 보안 에이전트(4116)는 시도된 어떤 동작들을 트래핑하고 그렇게 트래핑된 동작에 어떻게 응답할지를 결정하기 위해 하나 이상의 보안 규칙들(4122)을 이용하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(4116)는 트래핑된 동작에 대해 허용하거나 거부하거나 그에 대해 다른 교정 액션을 수행하도록 구성될 수 있다.

[0454] 도 41에 도시된 바와 같이, 전자 장치(4104)는 메모리(41088)에 연결된 프로세서(4106), 운영체제(4112), O/S 하위 보안 에이전트(4116) 및 보안 규칙들(4122)을 포함할 수 있다. 전자 장치(4104)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701), 도 9의 전자 장치(901), 도 12의 전자 장치(1201) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 프로세서(4106)는 도 2의 프로세서(208), 도 4의 프로세서(408), 도 7의 프로세서(702), 도 9의 프로세서(902), 도 12의 프로세서(1202) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(4108)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703), 도 9의 메모리(903), 도 12의 물리적 메모리(1203) 또는 가상 메모리 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 운영체제(4112)는 도 1의 운영체제(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 9의 운영체제(913), 도 12의 운영체제(1213) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 하위 보안 에이전트(4116)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM 보안 에이전트(217) 또는 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442), PC 펌웨어 보안 에이전트(444) 또는 O/S 하위 에이전트(450), 도 5의 펌웨어 보안 에이전트(516), 도 7의 마이크로코드 보안 에이전트(708), 도 9의 O/S 하위 에이전트(920), 도 12의 O/S 하위 보안 에이전트(1220) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 보안 규칙들(4122)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(222), 도 4의 보안 규칙들(434, 436, 438), 도 5의 보안 규칙들(518), 도 7의 보안 규칙들(707, 723), 도 9의 보안 규칙들(908), 도 12의 보안 규칙들(1208) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보안 규칙들(4122)은 어떤 적절한 방식(가령, 전자 장치(4104)의 사용자에 의해 설정된 정책들, 전자 장치(4104)를 포함하는 회사의 관리자에 의해 설정된 정책들, O/S 하위 보안 에이전트(4116)의 생성자에 의해 설정된 정책들 등)에 따라 설정될 수 있다. 일부 실시예들에서 O/S 하위 보안 에이전트(4116)는 (가령 악성 소프트웨어 정의들에 대한 업데이트 때문에) 네트워크(244)를 통해 보호 서버(202)로부터 보안 규칙들(4122)에 대한 업데이트들이나 수정안들을 요청 및/또는 수신할 수 있다.

[0455] 운영체제(4112)는 O/S 내 보안 에이전트(4118)를 포함할 수 있다. O/S 내 보안 에이전트(4118)는 도 2의 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418), 도 7의 O/S 내 보안 에이전트(718), 도 9의 O/S 내 보안 에이전트(919), 도 12의 O/S 내 보안 에이전트(1219) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0456] 도 41에 도시된 것과 같이 메모리(4108)는 허가 플래그들(4136) 및 이력(4140)를 포함할 수 있다. 허가 플래그들(4136)은 메모리(4108)에 저장된 콘텐츠와 관련된 허가 사항들을 설정하는 플래그들, 변수들 또는 다른 데이터를 보유할 수 있다. 예를 들어 허가 플래그들(4136)은 메모리(4108)의 특정 위치(가령, 페이지나 어드레스)에 대해, 전자 장치(4104) 상에서 실행되는 개체들이 특정 위치에 저장된 콘텐츠를 읽기, 쓰기 및/또는 실행할 수 있는지 여부를 나타낼 수 있다. 일부 실시예들에서, 허가 플래그들(4136)은 메모리(4108)의 페이지 테이블 엔트리들(PTE들) 및/또는 페이지 디렉토리 엔트리들(PDE들) 안에서 구현될 수 있다. 허가 플래그들(4136)은 특정 메모리 위치(가령, 페이지나 어드레스 범위)에 저장될 수 있다.

[0457] 이력(4140)은 트래핑된 액세스 시도들 및 트래핑된 액세스 시도들과 관련된 정보(가령, 시도된 액세스의 타입, 트래핑된 액세스 시도와 관련된 메모리 위치 등)을 기록하기 위한 로그, 리스트, 캐시 및/또는 다른 적절한 데이터 구조를 포함할 수 있다. 이력(4140)에 대해 시도되는 악의적 액세스를 통해 O/S 하위 보안 에이전트(4116)의 영향을 피하려는 악성 소프트웨어에 의한 시도를 막기 위해, 이력(4140)의 콘텐츠는 메모리를 보호하기 위해 여기 개시된 방법들 중 하나 이상에 따라 액세스 시도들로부터 보호될 수 있다. 예를 들어 O/S 하위 보안 에이전트(4116)는 이력(4140)이 존재하는 메모리 페이지나 메모리 어드레스 범위로의 O/S 하위 보안 에이전트(4116) 이외의 개체들로부터의 액세스 시도들을 트래핑할 수 있으며 그런 트래핑된 액세스 시도들을 거부할 수 있다.

[0458] 상술한 바와 같이 O/S 하위 보안 에이전트(4116)는 보안 규칙들(4122)에 기반하여 악성 코드의 존재를 검출할 수 있다. O/S 하위 보안 에이전트(4116)는 위에서 논의된 어떤 방법 및/또는 어떤 다른 적절한 방식으로 악성

코드의 존재를 검출할 수 있다. 예를 들어 O/S 하위 보안 에이전트(4116)는 전자 장치(4104)의 메모리(4108)나 다른 자원들에 대한 액세스를 트래핑함으로써 악성 코드의 존재를 검출할 수 있다. 다른 예로서, O/S 하위 보안 에이전트(4116)는 악성 코드를 찾아 메모리(4108) 및/또는 스토리지(4126)의 페이지들을 검색함으로써 악성 코드의 존재를 검출할 수 있다. 추가 예로서, O/S 하위 보안 에이전트(4116)는 O/S 내 보안 에이전트(4118)로부터 O/S 보안 에이전트(4118)가 악성 코드의 존재를 검출하였다는 통신문을 수신함으로서 메모리 내 악성 코드의 존재를 검출할 수 있다.

[0459] 특히, O/S 하위 보안 에이전트(4116)는 보안 규칙들(4112)에 기반하여, 개별적으로나 집합적으로 자가 변형 악성 소프트웨어의 존재를 가리킬 수 있는 메모리에 대해 시도된 하나 이상의 액세스들을 트래핑할 수 있다. 예로서, 허가 플래그들(4136)에서 제시된 바와 같이 메모리의 어떤 위치의 허가 사항에 대한 변경(가령, 읽기에서 읽기/쓰기로, 또는 읽기/쓰기로부터 읽기/쓰기/실행으로)은 악성 소프트웨어의 존재를 (가령, 개별적으로나 시도된 다른 메모리 액세스들과 집합적으로) 가리킬 수 있다. 따라서 O/S 하위 보안 에이전트(4116)는 허가 플래그들(4136)에 대해 검출된 변경에 대해 트래핑할 수 있다. 예를 들어 일부 실시예들에서, O/S 하위 보안 에이전트(4116)는 보안 규칙들(4122)에 따라, 허가 플래그들(4136)을 포함하는 메모리(4106)의 위치들(가령, 페이지들이나 어드레스들)에 대해 시도된 액세스들에 대해 트래핑할 수 있다. 동일하거나 대안적 실시예들에서, O/S 하위 보안 에이전트(4116)는 보안 규칙들(4122)에 따라, 메모리 허가 사항들을 변경하기 위해 운영체제(4112)의 호출들 및/또는 함수들(가령, MiSetProtectionOnSection, AllocateVirtualMemory(), MiProtectVirtualMemory() 및/또는 윈도우즈의 다른 적합한 함수 호출들)을 포함하는 메모리(4108)의 위치들(가령, 페이지들 또는 어드레스들) 및/또는 운영체제(4112)의 해당 허가 플래그들(가령, NTProtectVirtualMemory, ZwProtectVirtualMemory, ProtectVirtualMemory 및/또는 윈도우즈의 다른 적절한 플래그들)을 포함하는 메모리(4108)의 위치들(가령, 페이지들이나 어드레스들)에 대해 시도된 액세스들을 트래핑할 수 있다.

[0460] 다른 예로서, 메모리(4106)의 한 위치로부터 다른 곳으로의 콘텐츠 복사는 악성 소프트웨어의 존재를 가리킬 수 있다(가령, 개별적으로나 다른 메모리 액세스 시도들과 함께 집합적으로). 따라서 O/S 하위 보안 에이전트(4116)는 메모리 위치들 사이에서 콘텐츠의 복사와 관련된 액세스 시도들에 대해 트래핑할 수 있다. 예를 들어 일부 실시예들에서, O/S 하위 보안 에이전트(4116)는 보안 규칙들(4122)에 따라, 메모리의 한 위치에서 다른 위치로 콘텐츠를 복사하기 위한 프로세서 함수들에 대해 트래핑할 수 있다. 동일하거나 다른 대안적 실시예들에서, O/S 하위 보안 에이전트(4116)는 보안 규칙들(4122)에 따라, 윈도우즈의 MemUICopy 함수와 같이 데이터를 복사하는 운영체제(4112)의 호출들 및/또는 함수들을 포함하는 메모리(4108)의 위치들(가령, 페이지들이나 어드레스들)로의 액세스 시도들에 대해 트래핑할 수 있다.

[0461] 다른 예로서, 메모리(4106)에 저장된 콘텐츠의 변경이나 "제자리에 쓰기(writing-in-place)"는 악성 소프트웨어의 존재를 가리킬 수 있다(가령, 개별적으로나 다른 메모리 액세스 시도들과 함께 집합적으로). 따라서 O/S 하위 보안 에이전트(4116)는 메모리(4108) 내 콘텐츠의 제자리에 쓰기와 관련된 액세스 시도들에 대해 트래핑할 수 있다. 예를 들어 일부 실시예들에서, O/S 하위 보안 에이전트(4116)는 보안 규칙들(4122)에 따라, 메모리(4108) 내 제자리에서 콘텐츠를 변경하는 프로세서 함수들에 대해 트래핑할 수 있다. 동일하거나 다른 대안적 실시예들에서, O/S 하위 보안 에이전트(4116)는 보안 규칙들(4122)에 따라, 콘텐츠를 제자리에서 변경하는 운영체제(4112)의 호출들 및/또는 함수들을 포함하는 메모리(4108)의 위치들(가령, 페이지들이나 어드레스들)로의 액세스 시도들에 대해 트래핑할 수 있다.

[0462] 다른 예로서, 메모리에 저장된 콘텐츠의 복사나 변경의 실행은 악성 소프트웨어의 존재를 가리킬 수 있다(가령, 개별적으로나 다른 메모리 액세스 시도들과 함께 집합적으로). 따라서 O/S 하위 보안 에이전트(4116)는 메모리(4108) 내 콘텐츠의 실행과 관련된 액세스 시도들에 대해 트래핑할 수 있다. 예를 들어 일부 실시예들에서, O/S 하위 보안 에이전트(4116)는 보안 규칙들(4122)에 따라, 메모리(4108) 내 콘텐츠를 실행하는 프로세서 함수들에 대해 트래핑할 수 있다. 동일하거나 다른 대안적 실시예들에서, O/S 하위 보안 에이전트(4116)는 보안 규칙들(4122)에 따라, 콘텐츠를 실행하는 운영체제(4112)의 호출들 및/또는 함수들을 포함하는 메모리(4108)의 위치들(가령, 페이지들이나 어드레스들)로의 액세스 시도들에 대해 트래핑할 수 있다.

[0463] 다른 예로서, 메모리 안으로의 콘텐츠 로딩은 악성 소프트웨어의 존재를 가리킬 수 있다(가령, 개별적으로나 다른 메모리 액세스 시도들과 함께 집합적으로). 따라서 O/S 하위 보안 에이전트(4116)는 메모리(4108) 안으로의 코드 로딩과 관련된 액세스 시도들에 대해 트래핑할 수 있다. 예를 들어 일부 실시예들에서, O/S 하위 보안 에이전트(4116)는 보안 규칙들(4122)에 따라, 메모리(4108) 안으로의 코드 로딩을 위한 프로세서 함수들이나 시스템 함수들에 대해 트래핑할 수 있다. O/S 하위 보안 에이전트(4116)는 운영체제 로더의 사용과 같이, 메모리(4108) 안에 코드를 로딩하기 위한 안전하거나 규범적인 방법들을 결정하도록 구성될 수 있다. 운영체제(411

2)에 의해 취해지는 로직이나 단계들이 알려질 수 있도록 그러한 안전하거나 규범적인 방법들이 벤치마킹되거나 매핑될 수 있다. 코드를 메모리(4108) 안에 로딩하려는 시도를 트래핑할 때, O/S 하위 보안 에이전트(4116)는 그러한 시도가 코드 로딩을 위한 알려진 방법들과 매치하는지 여부를 판단할 수 있다. 예를 들어, 이미 할당된 메모리 부분 안으로의 코드 로딩과 관련된 시도 및 메모리로의 직접 쓰기로 운영체제 로더를 우회함으로써 그렇게 하려는 시도의 경우, 해당 시도는 악의적인 것이라고 판단될 수 있다.

[0464] 코드를 포함하는 페이지나 메모리 범위가 변경되었다면, O/S 하위 보안 에이전트(4116)는 변경을 계속 추적하도록 구성될 수 있다. 변경된 코드에 대한 후속 동작들은 실행 진행이 허가되는 경우, 추적되어 기록될 수 있다. 그러나 O/S 하위 보안 에이전트(4116)는 예컨대 다른 커널 모드 개체들이나 운영체제를 포함하는 메모리의 특권화된 위치들을 액세스하려는 그러한 변형 코드에 의한 시도들을 트래핑한 후 거부함으로써, 그러한 코드에 보다 적은 특권을 부여할 수 있다. 변형된 코드의 악성 소프트웨어 상태는 알려지지 않을 수 있고, 그것이 결론적으로 안전하다고 판단될 때까지 O/S 하위 보안 에이전트(4116)는 커널 함수들이나 루틴들로의 변형 코드 액세스를 거부할 수 있다.

[0465] O/S 하위 보안 에이전트(4116)는 하나 이상의 트래핑된 액세스 시도들에 관한 정보를 이력(4140) 안에 기록할 수 있다. 때때로 O/S 하위 보안 에이전트(4116)는 특정 메모리 위치와 관련하여 의심스러운 양태가 발생했는지를 판단하기 위해 이력(4140)을 분석할 수 있다. 그 분석 중에, O/S 하위 보안 에이전트(4116)는 이력(4140)에 구현된 바와 같이 특정 메모리 위치에 대한 양태가 자가 변형 악성 소프트웨어 코드의 잠정 존재를 증거할 수 있는 의심스러운 양태를 나타내는지를 판단하도록 규칙들(4122)을 조회할 수 있다. 예를 들어 이력(4140)의 분석이 제1메모리 위치에서의 콘텐츠가 제2위치로 복사되었고, 제2위치에서 변형되었고, 그런 다음 제2위치의 콘텐츠에 대한 실행 시도가 발생하였다는 것을 나타낸 경우, 그러한 나타냄은 자가 변형 악성 소프트웨어 코드의 잠정적 존재에 대한 증거가 될 수 있다. 다른 예로서, 이력(4140)의 분석이 각각 제3위치에 공통적 원형(ancestor)을 가지는 제1메모리 위치 및 제2메모리 위치에서의 콘텐츠가 각각 시도된 실행의 타깃이었다는 것을 나타내는 경우, 그러한 나타냄은 자가 변형 악성 소프트웨어 코드의 잠정 존재에 대한 증거일 수 있다. 또 다른 예로서, 이력(4140)의 분석이 특정 메모리 위치에서의 콘텐츠가 복수의 다른 메모리 위치들에서 원형들을 가진다는 것을 나타내는 경우, 그러한 나타냄은 자가 변형 악성 소프트웨어 코드의 잠정적 존재에 대한 증거일 수 있다. 또한 이력(4140)은 계층구조 내 레벨들과 개체들 사이에 이루어진 변형들을 기록할 수 있다.

[0466] 여기 사용된 바와 같이, 다른 메모리 위치에 있는 콘텐츠가 특정 메모리 위치에 있는 콘텐츠의 복사되거나 변형된 버전인 경우 상기 특정 메모리 위치의 콘텐츠는 그 다른 메모리 위치에 있는 콘텐츠의 "원형"이며, 다른 메모리 위치에 있는 콘텐츠가 특정 메모리 어드레스에 있는 콘텐츠 이외에 하나 이상의 중간 원형들의 파생자인 경우를 포함한다.

[0467] 모든 메모리 위치에 적용 시 그러한 이력(4140)의 기록이 전자 장치(4104)의 프로세싱 자원들의 상당 부분을 소비할 수 있기 때문에, O/S 하위 보안 에이전트(4116)는 특정 메모리 위치가 악성 소프트웨어에 취약하다는 것을 나타낼 수 있는 어떤 액세스 시도의 발생 시 그 특정 메모리 위치에 대해서만 이력(4140)을 기록할 수 있다. 예를 들어 O/S 하위 보안 에이전트(4116)는 (허가 플래그들(4136)에 구현된 바와 같이) 특정 메모리 위치에 대한 허가 사항들에 대한 변경을 트래핑할 때, 그 특정 메모리 위치에 대해 이력(4140) 기록을 시작할 수 있다.

[0468] 또한, 의심스러운 양태가 발생했는지를 판단하기 위해 이력(4140)을 분석하는 일은 모든 메모리 위치 및/또는 시도된 액세스에 대해 적용 시, 전자 장치의 프로세싱 자원들의 상당한 부분을 소비할 수 있기 때문에, O/S 하위 보안 에이전트(4116)는 특정 메모리 위치와 관련된 특정 트래핑된 액세스 시도의 발생 시에만 특정 메모리 위치에 대해 이력을 분석할 수 있다. 예를 들어 일부 실시예들에서, O/S 하위 보안 에이전트(4116)는 특정 메모리 위치에서의 콘텐츠에 대해 시도된 액세스를 트래핑할 때, 그 특정 메모리 위치와 관련된 이력(4140)의 분석을 개시할 수 있다.

[0469] 소정 실시예들에서, 하나의 트래핑된 액세스 시도 발생은 이력(4140)의 분석 없이도 의심스러운 동향을 나타낼 수 있다. 예를 들어, 허가 플래그들(4136)에서 제시된 바와 같이 소정 메모리 위치들의 허가 사항에 대한 변경(가령, 읽기에서 읽기/쓰기로, 또는 읽기/쓰기로부터 읽기/쓰기/실행으로)은 악성 소프트웨어의 존재를 가리킬 수 있다. 예를 들어 운영체제 커널이나 보안 애플리케이션을 저장하는 메모리의 위치들에 대한 허가 사항들의 변경은 잠정적 악성 소프트웨어의 증거를 보이는 의심스러운 양태를 가리킬 수 있다.

[0470] O/S 하위 보안 에이전트(4116)가 악성 소프트웨어의 잠정적 존재를 증거하는 의심스러운 양태를 검출한 경우(가령, 이력(4140)의 분석이나 하나의 트래핑된 액세스 시도에 기반함), O/S 하위 보안 에이전트(4116)는 (가령, 보안 규칙들(4122)에 따라) 교정 액션을 개시할 수 있다. 예를 들어 일부 실시예들에서, O/S 하위 보안 에이전

트(4116)는 의심스러운 양태를 검출 시, 콘텐츠가 악성인지 여부를 판단하기 위해, 검출된 의심스러운 양태와 관련된 특정 메모리 위치에 저장된 콘텐츠를 알려진 악성 소프트웨어 및/또는 알려진 신뢰/인증 프로세스들과 비교할 수 있다. 그러한 비교는 콘텐츠의 해시, 지문 또는 다른 서명을 알려진 프로세스들의 해시들, 지문들 또는 다른 서명들과 비교함으로써 이루어질 수 있다.

[0471] 다른 대안으로서, 혹은 추가적으로 O/S 하위 보안 에이전트(4116)가 악성 소프트웨어의 잠정적 존재를 증거하는 의심스러운 양태를 검출한 경우(가령, 이력(4140)의 분석이나 하나의 트래핑된 액세스 시도에 기반하여), O/S 하위 보안 에이전트(4116)는 의심스러운 양태와 관련된 법의학적 증거(가령, 메모리 위치의 콘텐츠, 메모리 위치와 관련된 이력(4140) 등)를 추후 분석을 위해 보호 서버(202)로 보고할 수 있다. 일부 실시예들에서 보호 서버(202)는 이때, 콘텐츠와 관련된 서명(가령, 해시나 지문)을 생성하고, 그 서명과 관련된 정책이나 블랙리스트 엔트리를 생성하며, 다른 전자 장치들 상에서 실행되는 보안 에이전트들로 그러한 정보를 전송할 수 있다. 동일하거나 대안적인 실시예들에서, 보호 서버(202)는 의심스러운 양태가 실제로 악성 소프트웨어를 나타내는지를 판단하기 위해 (가령, 다른 전자 장치들로부터 수신된 법의학적 증거와 함께) 의심스러운 동향을 더 분석할 수 있고, 만일 그런 경우 유사 양태가 악성 소프트웨어 존재의 증거인지 아닌지 여부에 관해 전자 장치로 (가령 보안 규칙들(4122)의 형식으로) 명령들을 전송할 수 있다.

[0472] O/S 하위 보안 에이전트(4116)가 의심스러운 동향과 관련된 메모리 위치의 콘텐츠가 악성이라고 판단하면(가령, 보호 서버(202), 보안 규칙들(4122)에 대한 참조 및/또는 다른 판단으로부터 수신된 알려진 프로세스들, 정보와의 콘텐츠 비교에 의해), O/S 하위 보안 에이전트(4116)는 (가령 보안 규칙들(4122)에 따라) 추가 교정 액션을 취할 수 있다. 그러한 교정 액션은 콘텐츠에 대한 실행 불허, 콘텐츠에 대한 변경 취소(가령, 이력(4140)에 제시된 대로 콘텐츠의 변경 및 복사), 콘텐츠 복구, 무해한 콘텐츠로 콘텐츠를 대체 및/또는 콘텐츠와 관련된 프로세스를 디세이블하는 것을 포함할 수 있으나, 그에 국한되는 것은 아니다.

[0473] 상술한 다양한 실시예들에서, 메모리(4108)의 특정 부분에 대해 적용되는 보안 규칙들(4122) 및 보호는 콘텐츠가 메모리(4108)의 다른 부분들 사이에서 이동될 때 과도기적으로 적용될 수 있다. 따라서, 예컨대 메모리(4508)의 특정 부분에 있는 콘텐츠를 메모리(4108)의 다른 부분으로 이동할 때 보안 규칙들(4122)의 특정 세트가 그 콘텐츠에 적용되는 경우, O/S 하위 보안 에이전트(4116)는 메모리(4108)의 목적지 부분에 적용할 보안 규칙들(4122)을 업데이트할 수 있다.

[0474] 상술한 바와 같이 O/S 하위 보안 에이전트(4116)는 보안 규칙들(4122)에 기반하여 악성 코드의 존재를 검출할 수 있다. O/S 하위 보안 에이전트(4116)는 위에서 논의된 어떤 방법 및/또는 어떤 다른 적절한 방식으로 악성 코드의 존재를 검출할 수 있다. 예를 들어 O/S 하위 보안 에이전트(4116)는 전자 장치(4104)의 메모리(4108)나 다른 자원들에 대한 액세스를 트래핑함으로써 악성 코드의 존재를 검출할 수 있다. 다른 예로서, O/S 하위 보안 에이전트(4116)는 악성 코드를 찾아 메모리(4108) 및/또는 스토리지(4126)의 페이지들을 검색함으로써 악성 코드의 존재를 검출할 수 있다. 추가 예로서, O/S 하위 보안 에이전트(4116)는 O/S 내 보안 에이전트(4118)로부터 O/S 보안 에이전트(4118)가 악성 코드의 존재를 검출하였다는 통신문을 수신함으로서 메모리 내 악성 코드의 존재를 검출할 수 있다.

[0475] 전자 장치(4104) 상의 악성 코드에 대한 검출에 반응하여(그러한 코드가 자가 변경 코드인지 다른 악성 코드인지 여부에 따라), O/S 하위 보안 에이전트(4116)는 악성 코드를 변경하는 것을 포함하는 교정 액션을 취할 수 있다. 여기에 사용된 바와 같이, 악성 코드의 "변경"은 제한 없이, 메모리(4108)에 구현되는 것과 같은 악성 코드의 변경, 스토리지(4126)에 구현되는 것과 같은 악성 코드의 변경 및/또는 전자 장치(4104)의 메모리(4108) 및 다른 자원들에 대한 악성 코드의 액세스에 대한 변경을 포함할 수 있다. 악성 코드의 변경은 악성 코드를 포함하는 메모리(4108)의 부분(가령, 페이지)이 악성 코드나 심지어 감염을 인지 못한 프로그램에 속할 수 있기 때문에 유리할 수 있다. 예를 들어 그러한 악성 코드가 워드 프로세싱 문서, 운영체제 커널의 일부 또는 악성 소프트웨어 자체에 내장될 수 있다.

[0476] 메모리(4108)에 내장된 것과 같은 악성 코드를 변경 시, O/S 하위 보안 에이전트(4116)는 악성 코드를 포함하는 프로그램이 스스로 종료되고/거나 악성 코드를 중립화할 수 있는(가령, 악성 코드 세그먼트와 관련된 스레들이나 프로세스들과 관련된 모든 코드 및 데이터를 삭제함으로써) 안전한 코드로 실행을 넘길 수 있도록 악성 코드를 수정할 수 있다. 예를 들어 O/S 하위 보안 에이전트(4116)는 운영체제(4112)의 "exit" 함수에 대한 호출을 메모리(4108) 내 악성 코드 안에 삽입함으로써, 악성 코드의 실행이 궁극적으로 종료되게 할 수 있다. 다른 예로서, O/S 하위 보안 에이전트(4116)는 메모리(4108)의 악성 코드 안에, 악성 코드를 중립화할 수 있는(가령, 악성 코드 세그먼트와 관련된 스레들이나 프로세스들과 관련된 모든 코드 및 데이터를 삭제함으로써) 코드의 알

려진 안전한 부분을 저장하고 있는 메모리(4108)의 다른 부분으로 악성 코드의 실행을 재지정할 수 있는 명령어(가령, "JUMP" 명령어)를 삽입할 수 있다. 또 다른 예로서, 악성 코드가 현재 실행 중이면, O/S 하위 보안 에이전트(4116)는 악성 코드를 중립화할 수 있는(가령, 악성 코드 세그먼트와 관련된 스레들이나 프로세스들과 관련된 모든 코드 및 데이터를 삭제함으로써) 코드의 알려진 안전한 부분으로 실행 제어가 전달되게 하도록 메모리(4108) 내 명령어 포인터 값들을 변경시킬 수 있다.

[0477] 어떤 예들에서는 악성 코드에 의해 구현되는 악성 프로세스를 단순히 종료하는 것은 바람직하지 않을 수 있다. 예를 들어 종료와 삭제는 삭제나 종료가 바람직하지 않은 부수 효과를 가질 수 있는 운영체제나 기타 다른 안전 애플리케이션들의 감염된 부분에 부적절할 수 있다. 따라서 O/S 하위 보안 에이전트(4116)는 악성 코드가 복구되도록 악성 코드를 변경하여, 감염된 애플리케이션이 아무 감염도 일어나지 않았던 것처럼 효과적으로 실행될 수 있게 한다. 예를 들어 O/S 하위 보안 에이전트(4116)은 알려지거나 안전한 코드로 악성 코드를 대체할 수 있다. 특정 예로서, 운영체제의 알려진 부분이 특정 메모리 페이지에서 감염된 경우, O/S 하위 보안 에이전트(4116)는 운영체제의 그러한 부분에 대해 알려진 코드를 가지고 그 특정 페이지를 덮어쓸 수 있다. 일 실시예에서 O/S 하위 보안 에이전트(4116)는 보호 서버로부터 대체 페이지를 획득할 수 있다. 그러한 대체 페이지는 요청 시 생성될 수 있고, 혹은 운영체제 구성요소, 드라이버 또는 다른 모듈의 알려진 부분을 대체하도록 구성될 수 있다. 다른 대안으로서 O/S 하위 보안 에이전트(4116)는 다른 메모리 위치에서 감염된 코드의 복구된 버전으로 실행이 계속되도록 메모리(4108)의 명령어 포인터를 변경할 수 있다.

[0478] 스토리지(4126)에 구현된 것과 같은 악성 코드를 변경할 때 O/S 하위 보안 에이전트(4116)는 악성 코드를 치환하거나 삭제할 수 있다. 예를 들어, 메모리(4108)와 스토리지(4126) 사이의 콘텐츠 전달을 트래핑함으로써, O/S 하위 보안 에이전트(4116)는 로그, 리스트, 캐시 또는 다른 데이터 구조 안에 스토리지(4126)에 저장된 대응 콘텐츠에 대한 메모리(4108)에 저장된 콘텐츠의 관계에 관한 정보를 모아서 저장할 수 있다. 그에 따라 O/S 하위 보안 에이전트(4116)가 메모리(4108) 내 악성 코드를 식별하면, 그것은 스토리지(4126)에 저장된 대응 콘텐츠에 대한 메모리(4108)에 저장된 콘텐츠의 관계에 대해 수집된 정보를 참조할 수 있고, 악성 코드를 가진 메모리(4108)의 위치들에 대응하는 스토리지(4126)의 위치들에 있는 콘텐츠를 변경할 수 있다. 그러한 변경은 제한 없이, 스토리지(4126) 내 대응 콘텐츠의 삭제 또는 스토리지(4126) 및/또는 메모리(4108) 내 악성 코드의 자가 중단 또는 삭제를 일으키도록 콘텐츠를 변경하는 것을 포함할 수 있다.

[0479] 메모리(4108) 및 전자 장치(4104)의 다른 자원들에 대한 악성 코드의 액세스를 변경할 때, O/S 하위 보안 에이전트(4116)는 악성 코드 세그먼트의 메모리(4108)나 전자 장치(4104)의 다른 자원들에 대한 모든 액세스를 거부할 수 있다. 메모리(4108) 및 다른 자원들에 대한 그러한 거부는 악성 코드를 포함하는 프로세스가 실패하거나 그렇지 않으면 무효한 상태에 놓이게 할 수 있다. 예를 들어 악성 코드가 식별되었으면, O/S 하위 보안 에이전트(4116)는 악성 코드를 포함하는 프로세스에 의한 메모리(4108)나 전자 장치(4104)의 자원들에 대해 시도된 액세스들을 트래핑하여 그러한 액세스를 거부할 수 있다.

[0480] 상술한 변경 기법들의 일부에 따라, O/S 하위 보안 에이전트(4116)는 악성 코드를 중립화하면서 악성 코드를 그대로 둘 수 있다. 그런 상황에서 O/S 하위 보안 에이전트는 악성 코드를 격리하여 법의학적 증거로서 추가 분석을 위해 보호 서버(202)로 전달할 수 있다. 보호 서버(202)는 이때 그 악성 코드와 관련된 서명(가령, 해시나 지문)을 생성하고, 그 서명과 관련된 정책이나 블랙리스트 엔트리를 생성하며, 다른 전자 장치들 상에서 실행되는 보안 에이전트들로 그러한 정보를 전송할 수 있다.

[0481] 일부 실시예들에서, 메모리(4108)의 특정 부분에 존재하는 악성 코드에 대한 식별은 O/S 하위 보안 에이전트(4116)가 악성 코드를 가진 메모리(4108)의 다른 부분들을 식별하게 할 수 있다. 예를 들어 악성 소프트웨어 유사 양태를 보이는 스레드를 검출할 때, O/S 하위 보안 에이전트(4116)는 그 스레드의 실행 어드레스 및/또는 악성 코드의 메모리 페이지 내 위치를 판단할 수 있다. 가상 메모리 구성에 있어서, 애플리케이션 코드가 근접 나열될 수 있는 반면, 물리적 메모리에서는 애플리케이션 코드가 실질적으로 근접하지 않을 수 있다. 따라서, 메모리(4108) 내 물리적 메모리 어드레스들에서 스토리지(4126) 내 가상 메모리 어드레스 간 운영체제에 의해 관리되는 매핑의 이점을 수행으로써, O/S 하위 보안 에이전트(4116)는 역시 악성 코드를 포함할 수 있는 식별된 악성 코드와 인접한 가상 메모리의 부분들을 식별할 수 있고, 그러한 가상 메모리 부분들을 다시, 감염될 수 있는 물리적 메모리 어드레스들에 매핑할 수 있다. 그에 따라 그러한 물리적 메모리 어드레스들에 있는 코드의 실행이 악성 코드의 존재에 대해 O/S 하위 보안 에이전트에 의해 더 감시될 수 있다.

[0482] 또한 O/S 하위 보안 에이전트(4116)는 스레드들에 의한 자원들의 실행 및/또는 사용과 관련된 활동을 감시하고, 그러한 감시에 기반하여 다양한 스레드들 사이의 관계를 판단하도록 구성될 수도 있다. 따라서, 특정 스레드가

악성이라고 식별될 때, O/S 하위 보안 에이전트(4116)는 악성 스레드(가령, 원형 스레드들, 파생 스레드들, 형제 스레드들 등)와 관련된 스레드들을 판단할 수 있고 악성 스레드외에 관련 스레드들에 관해 교정 액션을 취할 수 있다.

[0483] 그러한 감시 및 추적을 수행하기 위해 O/S 하위 보안 에이전트(4116)는 메모리(4108), 스토리지(4126), 네트워크(244) 및/또는 전자 장치(4104)의 다른 자원들에 대한 액세스를 감시하고; 스레드 실행 및/또는 스레드들에 의한 자원의 사용에 관한 운영체제 서비스, 호출 및/또는 함수들을 감시하고/하거나; 의심스러운 양태를 검출하기 위해 여기에 기술된 기법들 중 하나 이상을 이용할 수 있다. 예를 들어 O/S 하위 보안 에이전트(4116)는 메모리(4108), 스토리지(4126) 및/또는 전자 장치(4104)의 다른 구성요소들 상의 코드 읽기, 쓰기 및/또는 실행을 위해 시도된 액세스들, 허가 플래그들(4136)에 대해 시도된 변경들 및/또는 (가령, 개별적으로나 다른 메모리 액세스들과 함께 집합적으로) 의심스러운 양태를 나타낼 수 있는 다른 액세스 시도들을 (가령, 보안 규칙들(4122)에 기반하여) 트래핑하고 그러한 액세스 시도들에 관한 정보를 이력(4140)에 기록하는 도 12의 O/S 하위 보안 에이전트(1216)의 기능을 구현할 수 있다.

[0484] 또 다른 예로서, O/S 하위 보안 에이전트(4116)는 의심스러운 양태일 수 있는 스레드들에 의해 시도되는 스레드 실행 및/또는 자원 사용에 관한 운영체제 서비스, 호출 및/또는 함수들을 (가령, 보안 규칙들에 기반하여) 트래핑하고 그렇게 시도된 액세스들에 관한 정보를 이력(4140)에 기록하는 O/S 하위 보안 에이전트(712), 마이크로코드 보안 에이전트(708) 및/또는 O/S 하위 보안 에이전트(920)의 기능을 구현할 수 있다. 또한, 일부 실시예들에서 O/S 내 보안 에이전트(4118)는 의심스러운 양태를 증거할 수 있는 스레드들의 실행 및/또는 스레드들의 자원 사용에 대한 운영체제(4112)의 사용자 모드 또는 커널 모드 기능들에 대해 트래핑하고, 시도된 그러한 액세스들에 관한 정보를 이력(4140)에 기록하고/하거나 그러한 정보를 O/S 하위 보안 에이전트(4116)로 전송하도록 구성될 수 있다.

[0485] 스레드들 간의 관계를 판단하기 위해 O/S 하위 보안 에이전트(4116)는 메모리 관점에서 운영체제의 스레드 동기 객체들에 대해 시도되는 액세스를 감시할 수 있다. 예시하자면, 초기 스레드가 이차 스레드를 양산할 수 있고, 그러면 그 이차 스레드가 연산을 시작하(고 프로세스의 메인 스레드가 되면서) 초기 스레드는 스스로 종료된다. 다른 예로서, 스레드들은 프로세스간 통신(IPC) 호출들을 통해 서로를 생성, 종료 또는 보류하도록 동작할 수 있다. 따라서 스레드들은 여러 프로세스들에 걸쳐 이어질 수 있고, 한 프로세스의 스레드가 다른 프로세스들의 스레드들에 대해 생성, 종료 또는 중단을 위한 IPC 호출을 행할 수 있다. O/S 내 보안 에이전트(4118)는 그러한 IPC 호출들을 개시하기 위해 운영체제 호출(가령, 윈도우즈 실시예에서 NTCreateThread, NTSuspendThread 또는 NTTerminateThread와 같은 호출들)에 대해 트래핑함으로써 IPC 호출을 추적할 수 있다.

[0486] 그러나 O/S 내 보안 에이전트를 이용한 그러한 IPC 호출들에 대한 트래핑은 악성 소프트웨어에 의해 훼손되거나 회피될 수 있다. 따라서 O/S 하위 보안 에이전트(4116)가 IPC 호출들의 개시와 관련된 메모리나 프로세서 자원들로 시도되는 액세스들에 대해 트래핑함으로써 그러한 액세스 시도들을 감시할 수 있다. 예를 들어 O/S 하위 보안 에이전트(4116)는 IPC 호출 개시와 관련된 프로세서 자원들로 시도되는 액세스들을 트래핑하기 위해 도 9의 O/S 하위 트래핑 자원(920)의 기능을 구현할 수 있다. 다른 예로서, O/S 하위 보안 에이전트(4116)가 그러한 IPC 호출들을 위한 실행 코드가 저장된 메모리 위치들(가령, 페이지들이나 어드레스들)로 시도되는 액세스들을 트래핑하도록 도 12의 O/S 하위 보안 에이전트(1220)의 기능을 구현할 수 있다. IPC 호출과 관련된 이벤트에 대한 트래핑 시, O/S 하위 보안 에이전트(4116)는 그 이벤트들에 관한 정보(가령, 스레드 식별자들)를 이력(940)에 기록할 수 있다.

[0487] IPC 호출과 관련된 스레드들을 식별하기 위해, O/S 하위 보안 에이전트(4116)는 (가령 도 7에서 시스템 자원들(724)이라 식별된 것들과 같은) 하나 이상의 프로세서 자원들을 액세스하여 특정 스레드에 관한 정보를 획득할 수 있다. 예를 들어 윈도우즈 운영체제에서 실행되는 스레드들에 있어서, 프로세서 레지스터(가령, FS 레지스터)가 프로세서 제어 블록(PCB)이라 불리는 각각의 프로세서에 대한 메모리 안의 어떤 구조를 가리킬 수 있다. PCB는 프로세서 상에서 현재 실행되는 스레드의 ETHREAD 데이터 구조 및 예정된 스레드들에 대한 ETHREAD 리스트들을 포함하는, 프로세서 상의 스레드들을 관리하기 위한 스레드 스케줄러에 의해 사용되는 정보를 포함한다. 스레드 관련 ETHREAD 데이터 구조는 그 스레드에 대한 식별자를 포함해, 다수의 메타데이터 필드들을 포함할 수 있다. 그에 따라 O/S 하위 보안 에이전트(4116)는 보안을 윈도우즈에 적용할 때, 프로세서의 PCB의 메모리 위치를 판단하기 위해 프로세서 자원들 내 정보를 액세스하고, 그런 다음 특정 스레드에 대한 ETHREAD 정보를 얻기 위해 PCB를 액세스할 수 있다.

[0488] IPC 호출들에 관한 이력(4140)에 저장된 정보에 기반하여, O/S 하위 보안 에이전트(4116)는 다양한 스레드들 사

이의 관계를 판단하기 위해 이력(4140)을 분석할 수 있다. 그 분석 중에, O/S 하위 보안 에이전트(4116)는 이력(4140)에 제시된 스레드 양태가 둘 이상의 스레드들 간 관계를 나타내는지를 판단하기 위해 규칙들(4122)을 조회할 수 있다. 결과적으로 특정 스레드나 그것의 호스트 애플리케이션이 악성이라고 판단되면, O/S 하위 보안 에이전트(4116)는 그 특정 스레드와 관련된 하나 이상의 스레드들을 판단하고 그 관련 스레드들에 대해 교정 액션에 착수할 수 있다. 예를 들어 교정 액션은 O/S 하위 보안 에이전트(4116)가 그러한 관련 스레드들이 악성 코드를 포함하는지를 판단하기 위해 (가령, 이 개시의 어딘가에 기술되는 하나 이상의 기법들을 사용하여) 그러한 스레드들을 검사, 검색 및/또는 분석하는 것을 포함할 수 있다. 다른 예로서, 교정 액션은 그러한 스레드들이 악성이라고 판단될 경우 O/S 하위 보안 에이전트(4116)가 (가령, 이 개시의 어딘가에 기술되는 하나 이상의 기법들을 이용하여) 그러한 하나 이상의 관련 스레드들을 종료, 삭제, 변경 또는 중립화하는 것을 포함할 수 있다. 추가적 예로서, 교정 액션은 O/S 하위 보안 에이전트(4116)가 추가 분석을 위해 보호 서버(202)로 특정 스레드 및 그 관련 스레드들과 연관된 법의학적 증거를 전송하는 것을 포함할 수 있다. 보호 서버(202)는 정보를 분석하고, 취해질 어떤 추가 교정 액션에 관한 명령어들을(가령, 보안 규칙들(4122)의 형식으로) 전자 장치(4104)로 전송할 수 있다. 또 다른 예로서, O/S 하위 보안 에이전트(4116)는 악성 스레드들을 포함하는 메모리의 부분들(가령, 페이지들, 메모리 어드레스들 등)을 복구하고자 시도할 수 있다.

[0489] 그러한 복구를 수행하기 위해, O/S 하위 보안 에이전트(4116)는 때때로 메모리(4106)나 그 특정 부분들(가령, 운영체제, 보안 애플리케이션, 또는 중요 드라이버를 저장하는 메모리의 부분들)에 대한 스냅샷들을 생성하고, 그 스냅샷들을 (가령, 이력(4140)안에) 저장할 수 있다. 스냅샷들은 스냅샷의 날짜 및 시간과 같은 정황 정보, 스냅샷과 관련된 개체(가령, 운영체제, 애플리케이션 또는 드라이버), 메모리 페이지와 관련된 스레드 식별자, 가상 메모리 내 메모리의 어드레스 위치 등와 함께 저장될 수 있다. 악성 스레드나 스레드 패밀리가 위치하는 경우, 그러한 스레드들을 포함하는 메모리의 부분들은 악성 스레드를 가진 메모리의 부분을 알맞은 스냅샷으로, 그 스냅샷과 관련된 정황 정보의 적어도 일부에 기반하여, 대체함으로써 복구될 수 있다. 일부 실시예들에서 O/S 하위 보안 에이전트(4116)는 또한 스냅샷 생성에 뒤따라, 메모리 위치에 대해 안전한 개체들에 의해 이루어진 변경들을 (가령 이력(4140에) 기록함으로써, 악성 소프트웨어 검출에 따른 스냅샷으로의 복귀가 합법적 변경을 취소하지 않게 할 수 있다.

[0490] 모든 스레드들에 적용되는 경우 관련 스레드들 및 스레드들에 의한 잠정적 악성 양태에 대한 감시가 전자 장치의 프로세싱 자원들의 상당 부분을 소비할 수 있기 때문에, O/S 하위 보안 에이전트(4116)는 특정 메모리 위치가 악성 소프트웨어에 취약하고/거나 특정 메모리가 중요하거나 민감한 코드나 데이터(가령, 운영체제나 보안 애플리케이션)를 저장하고 있음을 나타낼 수 있는 액세스 시도의 발생 시에만 그러한 감시를 수행할 수 있다. 예를 들어 O/S 하위 보안 에이전트(4116)는 (허가 플래그들(4136)에 구현된 것과 같이) 특정 메모리 위치를 위한 허가 사항에 대한 변경을 트래핑할 때, 특정 메모리 위치에 대한 스레드 동향 및 스레드 관계들의 감시를 시작할 수 있다.

[0491] 도 42는 전자 장치 상의 자가 변경 코드의 규제 및 통제를 위한 방법(4200)의 실시예이다. 단계 4205에서 O/S 하위 보안 에이전트는 메모리에 대해 시도되는 액세스들에 대해 트래핑할 수 있으며, 그러한 액세스 시도들 각각은 개별적으로나 집합적으로 자가 변경 악성 소프트웨어의 존재를 가리킬 수 있다. 트래핑되는 액세스 시도들은 보안 규칙들에 의해 판단될 수 있다. 잠정적으로 악성 소프트웨어를 가리키는 액세스 시도들은 제한없이, 메모리 허가 사항들에 대한 변경, 한 메모리 위치의 콘텐츠를 다른 메모리 위치로 복사, 메모리 위치의 콘텐츠 변경 및 메모리 위치의 실행을 포함할 수 있다.

[0492] 단계 4210에서 O/S 하위 보안 에이전트는 트래핑된 액세스 시도들에 관한 정보(가령, 시도된 액세스 타입, 트래핑된 액세스 시도와 관련된 메모리 위치 등)를 이력 안에 기록할 수 있다. 그러한 이력의 기록은 모든 메모리 위치에 적용 시 전자 장치의 프로세싱 자원들의 상당 부분을 소비할 수 있기 때문에, O/S 하위 보안 에이전트는 특정 메모리 위치가 악성 소프트웨어에 취약하다는 것을 가리킬 수 있는 메모리 액세스 시도의 발생 시에 특정 메모리 위치에 대한 이력 상의 기록을 개시할 수 있다(가령, 특정 메모리 어드레스에 대해 이력의 기록을 개시하기 위한 트리거링 이벤트들을 설정하는 보안 규칙들에 기반하여). 예를 들어 O/S 하위 보안 에이전트는 (가령, 메모리 위치에 대한 허가 플래그들에서 구현되는 것과 같이) 특정 메모리 위치에 대한 허가 사항들에 대한 변경을 트래핑할 때 특정 메모리 위치에 대한 이력 기록을 시작할 수 있다.

[0493] 단계 4215에서 O/S 하위 보안 에이전트는 (가령 보안 규칙들에 따라) 특정 메모리 위치에 대한 이력의 분석 개시를 일으킬 수 있는 액세스 시도에 대해 감시할 수 있다. 의심스러운 양태가 일어났는지를 판단하기 위해 이력을 분석하는 것은 모든 메모리 위치 및/또는 액세스 시도에 적용 시, 전자 장치의 프로세싱 자원들의 상당 부분을 소비할 것이기 때문에, O/S 하위 보안 에이전트는 특정 메모리 위치와 관련하여 트래핑된 특정 액세스 시

도의 발생 시 (가령, 아래의 단계 4220에서) 특정 메모리 위치에 대해 이력의 분석을 개시할 수 있다. 예를 들어 일부 실시예들에서 O/S 하위 보안 에이전트는 특정 메모리 위치에 있는 콘텐츠에 대해 시도된 액세스를 트래핑 시, 특정 메모리 위치와 관련된 이력의 분석 개시를 일으킬 수 있다.

[0494] 단계 4220에서 O/S 하위 보안 에이전트(4116)는 특정 메모리 위치와 관련하여 의심스러운 양태가 일어났는지를 판단하기 위해 이력을 분석할 수 있다. 그러한 분석 중에, O/S 하위 보안 에이전트는 보안 규칙들을 조회하여, 이력에 구현된 것과 같은 특정 메모리 위치에 대한 양태가 자가 변형 악성 소프트웨어 코드의 잠정적 존재를 증거할 수 있는 의심스러운 양태를 나타내는지를 판단할 수 있다. 예를 들어, 이력의 분석이 제1메모리 위치에 있는 콘텐츠가 제2위치로 복사되었고, 제2위치에서 변형되었으며 그런 다음 제2위치의 콘텐츠의 실행이 일어났음을 나타내는 경우, 그러한 나타냄은 자가 변형 악성 소프트웨어 코드의 잠정적 존재의 증거일 수 있다. 다른 예로서, 이력의 분석이 각각 제3위치에 공통적인 원형을 가지는 제1메모리 위치 및 제2메모리 위치의 콘텐츠가 시도된 실행의 타깃이었음을 나타내는 경우, 그러한 나타냄은 자가 변형 악성 소프트웨어 코드의 잠정적 존재에 대한 증거일 수 있다. 또 다른 예로서, 이력의 분석이 특정 메모리 위치에서의 콘텐츠가 복수의 다른 메모리 위치들에서 원형들을 가진다는 것을 나타내는 경우, 그러한 나타냄은 자가 변형 악성 소프트웨어 코드의 잠정적 존재에 대한 증거일 수 있다.

[0495] 단계 4225에서 O/S 하위 보안 에이전트는 악성 소프트웨어의 잠정적 존재를 증거하는 의심스러운 동향이 (가령, 하나의 트래핑된 액세스 시도나 이력의 분석에 기반하여) 검출되었는지를 판단할 수 있다. 의심스러운 동향이 검출되었으면, 방법(4200)은 단계(4230)로 진행할 수 있다. 그렇지 않으면, 방법(4200)은 다시 단계(4205)로 진행할 수 있다. 단계 4230에서, (가령, 하나의 트래핑된 액세스 시도나 이력의 분석에 기반하여) 악성 소프트웨어의 잠정적 존재를 증거하는 의심스러운 동향의 검출에 따라, O/S 하위 보안 에이전트는 (가령, 보안 규칙들(4122)에 따라) 교정 액션을 개시할 수 있다. 예를 들어 일부 실시예들에서, O/S 하위 보안 에이전트는 의심스러운 동향의 검출 시, 검출된 의심스러운 동향과 관련된 특정 메모리 위치에 저장된 콘텐츠를 알려진 악성 소프트웨어 및/또는 알려진 안전/인증 프로세스들과 비교하여 그 콘텐츠가 악성인지 여부를 판단할 수 있다. 그러한 비교는 콘텐츠의 해시, 지문 또는 다른 서명을 알려진 프로세스들의 해시들, 지문들 또는 다른 서명들과 비교함으로써 이루어질 수 있다. 다른 예로서, O/S 하위 보안 에이전트는 의심스러운 동향을 검출 시, 의심스러운 동향과 관련된 법의학적 증거(가령, 메모리 위치의 콘텐츠, 메모리 위치와 관련된 이력 등)를 추가 분석을 위해 보호 서버에 보고할 수 있다.

[0496] 단계 4235에서 O/S 하위 보안 에이전트는 (가령, 알려진 프로세스들과의 비교, 보호 서버로부터 수신된 정보, 보안 규칙들에 대한 참조 및/또는 다른 판단에 의해) 악성 동향과 관련된 메모리 위치의 콘텐츠가 악성인지를 판단할 수 있다. 콘텐츠가 악성이면, 방법(4200)은 단계 4240로 진행할 수 있다. 그렇지 않으면, 방법(4200)은 다시 단계 4205로 진행할 수 있다. 단계 4240에서,콘텐츠가 악성이라고 판단함에 따라, O/S 하위 보안 에이전트는 (가령 보안 규칙들에 따라) 추가 교정 액션을 취할 수 있다. 그러한 교정 액션은 콘텐츠 실행 불허, 콘텐츠 변경(가령, 이력에 제시된 것과 같은 콘텐츠의 변경 및 복사) 취소, 콘텐츠 복구, 콘텐츠를 무해한 콘텐츠로 대체 및/또는 콘텐츠와 관련된 프로세스 디세이블을 포함할 수 있으나, 그에 국한되는 것은 아니다. 단계 4240의 완료 후, 방법(4200)은 다시 단계 4205로 진행할 수 있다.

[0497] 도 43은 전자 장치 상에서 악성 코드를 수정하는 방법(4305)의 실시예이다. 단계 4305에서 O/S 하위 보안 에이전트는 전자 장치 상의 악성 코드의 존재를 검출할 수 있다. 예를 들어 O/S 하위 보안 에이전트는 전자 장치의 메모리나 전자 장치의 다른 자원들로의 액세스를 트래핑함으로써 악성 코드의 존재를 검출할 수 있다. 다른 예로서, O/S 하위 보안 에이전트는 전자 장치의 메모리 및/또는 스토리지의 페이지들을 악성 코드가 있는지 검색함으로써 악성 코드의 존재를 검출할 수 있다. 또 다른 예로서, O/S 하위 보안 에이전트는 악성 코드의 존재를 검출한 O/S 내 보안 에이전트로부터 통신문을 수신함으로써 메모리 내 악성 코드의 존재를 검출할 수 있다.

[0498] 단계 4310-4320에서, 전자 장치 상의 악성 코드 검출에 따라, O/S 하위 보안 에이전트는 악성 코드를 변경하는 것을 포함하는 교정 액션을 취할 수 있다. 예를 들어 단계 4310에서, O/S 하위 보안 에이전트는 악성 코드를 포함하는 프로그램이 자가 종료하고/거나 악성 코드를 중립화할 수 있는(가령, 악성 코드 세그먼트와 관련된 스레드들이나 프로세스들과 관련된 모든 코드 및 데이틀 삭제함으로써) 안전 코드로 실행을 넘기도록 악성 코드를 변경할 수 있다. 예를 들어 O/S 하위 보안 에이전트는 전자 장치의 메모리 내 악성 코드 안으로 운영체제의 "exit" 함수에 대한 호출을 삽입함으로써, 악성 코드의 실행이 궁극적으로 종료되게 할 수 있다. 다른 예로서, O/S 하위 보안 에이전트는 (가령, 악성 코드 세그먼트와 관련된 스레드들이나 프로세스들과 관련된 모든 코드나 데이터를 삭제함으로써) 악성 코드를 중립화할 수 있는 알려진 안전한 코드 부분이 저장된 메모리의 다른 부분으로 악성 코드의 실행을 재지정할 수 있는 명령어(가령 "JUMP" 명령어)를 전자 장치의 메모리 내 악성 코드 안

에 삽입할 수 있다. 또 다른 예로서, 악성 코드가 현재 실행 중이면, O/S 하위 보안 에이전트는 (가령, 악성 코드 세그먼트와 관련된 스레드들이나 프로세스들과 관련된 모든 코드나 데이터를 삭제함으로써) 악성 코드를 중립화할 수 있는 코드의 알려진 안전한 부분으로 실행 제어가 전달되게 하기 위해 메모리 내 명령어 포인터 값들을 변경할 수 있다. 대안적으로 O/S 하위 보안 에이전트는 다른 메모리 위치에서 감염된 코드의 복구 번전으로 실행이 계속되도록 악성 코드를 알려진 안전한 코드로 대체하거나 메모리 내 명령어 포인터를 변경함으로써, 악성 코드가 복구되도록 악성 코드를 수정하여, 감염된 애플리케이션이 마치 아무 감염도 일어나지 않은 것처럼 효과적으로 실행될 수 있게 할 수 있다.

[0499] 단계 4315에서 O/S 하위 보안 에이전트는 스토리지 안에 구현된 것과 같은 악성 코드를 변경할 수 있다. 예를 들어, 전자 장치의 메모리와 전자 장치 사이 또는 그 반대의 경우의 콘텐츠 이동을 트래핑함으로써, O/S 하위 보안 에이전트는 로그, 리스트, 캐시 또는 다른 데이터 구조 안에, 스토리지에 저장된 대응 콘텐츠에 대한 메모리에 저장된 콘텐츠의 관계에 관한 정보를 수집하여 저장할 수 있다. 그에 따라 O/S 하위 보안 에이전트가 메모리에서 악성 코드를 식별한 경우, 그것은 스토리지에 저장된 대응 콘텐츠에 대한 메모리에 저장된 콘텐츠의 관계에 관해 수집된 정보를 참조하고, 악성 코드를 가진 메모리의 위치들에 상응하는 스토리지의 위치들에 있는 콘텐츠를 변경할 수 있다. 그러한 변경은 제한 없이, 스토리지 및/또는 메모리 내 악성 코드의 자가 제거나 삭제를 일으키도록 스토리지 내 대응 콘텐츠의 삭제나 콘텐츠의 변경을 포함할 수 있다.

[0500] 단계 4320에서 O/S 하위 보안 에이전트는 예컨대 전자 장치의 메모리나 다른 자원들에 대한 어떤 악성 코드 세그먼트의 액세스를 거부하기 위해, 전자 장치의 메모리 및 다른 자원들로의 악성 코드의 액세스를 변경할 수 있다. 그러한 메모리 및 다른 자원들에 대한 거부는 악성 코드를 포함하는 프로세스가 실패하거나 무효한 상태가 되게 할 수 있다. 예를 들어 악성 코드가 식별되었으면, O/S 하위 보안 에이전트는 전자 장치의 메모리나 자원들로 악성 코드를 포함한 프로세스에 의해 시도된 액세스들을 트래핑하고 그러한 액세스를 거부할 수 있다.

[0501] 단계 4325에서 O/S 하위 보안 에이전트는 검출된 악성 코드의 물리적 메모리 어드레스에 기반하여 잠정적으로 악성 코드를 가진 메모리의 다른 부분들을 식별할 수 있다. 예를 들어 악성 소프트웨어 유사 양태를 보이는 스레드를 검출 시, O/S 하위 보안 에이전트는 그 스레드에 대한 실행 어드레스 및/또는 악성 코드의 메모리 페이지 안에서의 위치를 판단할 수 있다. 가상 메모리 구성에 있어서, 애플리케이션 코드가 근접 나열될 수 있는 반면, 물리적 메모리에서는 애플리케이션 코드가 실질적으로 근접하지 않을 수 있다. 따라서, 운영체제가 메모리 내 물리적 메모리 어드레스를 스토리지 내 가상 메모리 어드레스들로 매핑함으로써 관리되는 매핑의 이점을 수행으로써, O/S 하위 보안 에이전트는 역시 악성 코드를 포함할 수 있는 식별된 악성 코드에 대응하는 가상 메모리의 부분들을 식별하고, 그 가상 메모리 부분들을 다시, 감염될 수 있는 물리적 메모리 어드레스들로 매핑할 수 있다. 그에 따라, 그러한 물리적 메모리 어드레스들에서의 코드의 실행이 O/S 하위 보안 에이전트에 의해 악성 코드가 존재하는지 더 감시될 수 있다.

[0502] 단계 4330에서 O/S 하위 보안 에이전트는 악성 코드를 격리하고 추가 분석을 위해 법의학적 증거로서 보호 서버에 전달할 수 있다.

[0503] 도 44는 전자 장치 상에서 관련 스레드들을 감시 및 추적하는 방법(4400)의 실시예이다. 단계 4405에서 O/S 하위 보안 에이전트는 다른 스레드에 의한 한 스레드의 생성, 보류 또는 중단과 관련된 스레드 동기 객체들의 함수 호출들과 관련된 메모리나 프로세서 자원들로 시도된 액세스들을 트래핑할 수 있다. 예를 들어 O/S 하위 보안 에이전트는 프로세스간 통신(IPC) 호출들과 관련된 메모리나 프로세서 자원들로 시도된 액세스들을 트래핑할 수 있다. 단계 4410에서 O/S 하위 보안 에이전트는 그렇게 트래핑된 액세스 시도들과 관련된 정보(가령, 스레드 식별자들)를 이력으로 저장할 수 있다.

[0504] 단계 4415에서 O/S 하위 보안 에이전트는 메모리나 프로세서 자원들에 대해 시도된 액세스들을 트래핑할 수 있고, 여기서 그러한 액세스 시도들 각각은 개별적으로나 집합적으로 악성 소프트웨어의 존재를 나타낼 수 있다. 트래핑되는 액세스 시도들은 보안 규칙들에 의해 판단될 수 있다. 잠정적으로 악성 소프트웨어를 가리키는 액세스 시도들은 제한 없이, 메모리 허가 사항들에 대한 변경, 한 메모리 위치의 콘텐츠를 다른 메모리 위치로 복사, 메모리 위치의 콘텐츠 변경 및 메모리 위치의 실행을 포함할 수 있다. 단계 4420에서 O/S 하위 보안 에이전트는 그 트래핑된 액세스 시도들과 관련된 정보(가령, 스레드 식별자들)을 이력에 저장할 수 있다. 단계 4410 및 4420에서, O/S 하위 보안 에이전트는 스레드 메타데이터에 대한 메모리 위치를 판단하기 위해 프로세서 자원들 안의 정보를 액세스하고, 그 스레드 메타데이터에 기반하여 이력 내 정보의 일부로서 저장할 특정 스레드들에 대한 스레드 식별자들을 얻을 수 있다.

[0505] 단계 4425에서 O/S 하위 보안 에이전트는 특정 스레드와 관련하여 악성 소프트웨어 감염과 일치하는 양태가 발

생했는지를 판단하기 위해 (가령 보안 규칙들에 따라) 이력을 분석할 수 있다. 단계 4430에서, 만일 악성 소프트웨어 감염과 일치하는 양태가 일어났으면, O/S 하위 보안 에이전트는 이력을 분석하여 악성 소프트웨어 활동이 식별되었던 특정 스레드와 관련된 하나 이상의 스레드들을 판단할 수 있다.

[0506] 단계 4435에서 O/S 하위 보안 에이전트는 특정 스레드 및 하나 이상의 관련 스레드들에 대해 교정 액션을 취할 수 있다. 예를 들어 교정 액션은 O/S 하위 보안 에이전트가 그러한 관련 스레드들이 악성 코드를 포함하는지를 판단하기 위해 그 스레드들을 검사, 검색 및/또는 분석하는 것을 포함할 수 있다. 다른 예로서 교정 액션은 그러한 스레드들이 악성이라고 판단된 경우 O/S 하위 보안 에이전트가 그러한 하나 이상의 관련 스레드들을 종료, 삭제, 변경 또는 중립화하는 것을 포함할 수 있다. 추가 예로서, 교정 액션은 O/S 하위 보안 에이전트가 특정 스레드 및 그것의 관련 스레드들과 연관된 법의학적 증거를 추가 분석을 위해 보호 서버로 전송하는 것을 포함할 수 있다.

[0507] 도 45는 전자 장치(4504)의 메모리 및 스토리지를 보호하는 시스템(4500)의 일 실시예이다. 시스템(4500)은 전자 장치(4504)의 메모리(4508) 및 스토리지(4526)를 액세스하려는 악성 시도들에 대해 보호하기 위해 전자 장치(4501) 상에서 동작하도록 구성되는 O/S 하위 보안 에이전트(4516)을 포함할 수 있다. 또한 O/S 하위 보안 에이전트(4516)는 어떤 시도 동작들을 트래핑하고 그 트래핑된 동작에 어떻게 반응할지를 판단하기 위해 하나 이상의 보안 규칙들(4522)을 이용하도록 구성될 수 있다. O/S 하위 보안 에이전트는 트래핑된 동작을 허용, 거부, 또는 그에 대한 다른 교정 액션을 수행하도록 구성될 수 있다.

[0508] 도 45에 도시된 바와 같이 전자 장치(4504)는 메모리(4508)에 연결된 프로세서(4506), 애플리케이션(4510), 드라이버(4511), 운영체제(4512), 운영체제 하위 보안 에이전트(4516), 스토리지(4526) 및 애플리케이션 자산들(4548)을 포함할 수 있다. 전자 장치(4504)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701), 도 9의 전자 장치(901), 도 12의 전자 장치(1201) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 프로세서(4506)는 도 2의 프로세서(208), 도 4의 프로세서(408), 도 7의 프로세서(702), 도 9의 프로세서(902), 도 12의 프로세서(1202) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(4508)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703), 도 9의 메모리(903), 도 12의 물리적 메모리(1203) 또는 가상 메모리 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 애플리케이션(4510)은 도 1의 애플리케이션(110), 도 2의 애플리케이션(210), 도 4의 애플리케이션(410), 도 7의 애플리케이션(709), 도 9의 애플리케이션(910), 도 12의 애플리케이션(1210) 및/또는 이들의 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 드라이버(4511)은 도 1의 드라이버(111), 도 2의 드라이버(211), 도 4의 드라이버(411), 도 7의 드라이버(711), 도 9의 드라이버(911), 도 12의 드라이버(1211) 및/또는 이들의 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 운영체제(4512)는 도 1의 운영체제들(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 9의 운영체제(913), 도 12의 운영체제(1213) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 하위 보안 에이전트(4516)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM 보안 에이전트(217) 또는 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442), PC 펌웨어 보안 에이전트(444) 또는 O/S 하위 에이전트(450), 도 5의 펌웨어 보안 에이전트(516), 도 7의 마이크로코드 보안 에이전트(708), 도 9의 O/S 하위 에이전트(920), 도 12의 O/S 하위 보안 에이전트(1220) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(4518)는 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418), 도 7의 O/S 내 보안 에이전트(718), 도 9의 O/S 내 보안 에이전트(919), O/S 내 보안 에이전트(1219) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 스토리지 장치(4526)는 도 4의 스토리지(426)의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0509] 보안 규칙들(4522)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(222), 도 4의 보안 규칙들(434, 436, 438), 도 5의 보안 규칙들(518), 도 7의 보안 규칙들(707, 723), 도 9의 보안 규칙들(908), 도 12의 보안 규칙들(1208) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보안 규칙들(4522)은 어떤 적절한 방식(가령, 전자 장치(4504)의 사용자에 의해 설정된 정책들, 전자 장치(4504)를 포함하는 회사의 관리자에 의해 설정된 정책들, O/S 하위 보안 에이전트(4516)의 크리에이터에 의해 설정된 정책들 등)에 따라 설정될 수 있다. 일부 실시예들에서 O/S 하위 보안 에이전트(4516)는 (가령 악성 소

프트웨어 정의들에 대한 업데이트 때문에) 네트워크(244)를 통해 보호 서버(202)로부터 보안 규칙들(4522)에 대한 업데이트들이나 수정안들을 요청 및/또는 수신할 수 있다.

[0510] O/S 하위 보안 에이전트(4516)는 메모리 추적 기능부(4542), 스토리지 추적 기능부(4544) 및 메모리/스토리지 보안 계층(4546)을 포함할 수 있다. 메모리 추적 기능부(4542)은 메모리(4508)에 대한 액세스를 감시하기 위해 메모리(4508)와 인터페이스할 수 있다. 예를 들어 메모리 추적 기능부(4542)는 (가령, 페이지 테이블 플래그들 및/또는 비트 플래그들에 의해 지시된 바와 같이) 메모리(4508) 내 특정 페이지 읽기, 쓰기 또는 실행을 위한 애플리케이션(4510), 드라이버(4511) 및/또는 운영체제(4512)의 액세스 시도를 트래핑하거나 트리거하기 위해, 도 1의 O/S 하위 보안 에이전트(104), 도 2의 SVMM 보안 에이전트(217) 또는 SVMM(216), 도 4의 펌웨어 보안 에이전트(442), 도 7의 마이크로코드 보안 에이전트(708), 도 9의 O/S 하위 보안 에이전트(920) 및/또는 도 12의 O/S 하위 보안 에이전트(1220)의 기능을 구현하기 위해 구성되거나 그 기능에 의해 전체나 일부가 구현될 수 있다. 다른 예로서, 메모리 추적 기능부(4542)는 메모리(4508) 내 특정 어드레스를 읽거나 쓰거나 실행하려는 애플리케이션(4510), 드라이버(4511) 및/또는 운영체제(4512)에 의해 시도된 액세스를 트래핑하거나 트리거하기 위해, O/S 하위 보안 에이전트(712) 및/또는 마이크로코드 보안 에이전트(708)의 기능을 구현하도록 구성되거나 그러한 기능에 의해 전체나 일부가 구현될 수 있다. 그에 따라 메모리 추적 기능부(4542)는 메모리(4508) 내 한 위치에서 다른 위치로의 콘텐츠 이동(가령, 한 페이지에서 다른 페이지로, 혹은 한 어드레스에서 다른 어드레스로) 또는 메모리(4508) 및 스토리지(4526) 사이의 이동(가령, 스토리지(4526)으부터 실행 코드의 가상 메모리 정황의 스왑이나 로딩과 관련)을 추적할 수 있다. 또한 메모리 추적 기능부(4542)는 추적된 이동에 관한 정보를 메모리 추적 기능부(4542) 및/또는 메모리/스토리지 보안 계층(4546)에 의해 액세스될 수 있는 로그, 리스트, 캐시 또는 다른 적합한 데이터 구조에 저장할 수 있다.

[0511] 스토리지 추적 기능부(4544)는 스토리지(4526) 내 한 위치에서 다른 위치로의 콘텐츠 이동 또는 메모리(4508) 및 스토리지(4526) 사이의 이동을 감시하기 위해 스토리지(4526)와 인터페이스할 수 있다. 예를 들어 스토리지 추적 기능부(4544)는 스토리지(4526) 내 특정 섹터를 읽거나 쓰거나 실행하려는 애플리케이션(4510), 드라이버(4511) 및/또는 운영체제(4512)의 액세스 시도를 트래핑하거나 트리거하기 위해, 도 1의 O/S 하위 보안 에이전트(104), 도 2의 SVMM 보안 에이전트(217) 또는 SVMM(216), 도 4의 펌웨어 보안 에이전트(442), 도 7의 마이크로코드 보안 에이전트(708), 도 9의 O/S 하위 보안 에이전트(920) 및/또는 도 12의 O/S 하위 보안 에이전트(1220)의 기능을 구현하기 위해 구성되거나 그 기능에 의해 전체나 일부가 구현될 수 있다. 다른 예로서, 스토리지 추적 기능부(4544)는 스토리지(4526) 내 특정 어드레스를 읽거나 쓰거나 실행하려는 애플리케이션(4510), 드라이버(4511) 및/또는 운영체제(4512)에 의해 시도된 액세스를 트래핑하거나 트리거하기 위해, O/S 하위 보안 에이전트(712) 및/또는 마이크로코드 보안 에이전트(708)의 기능을 구현하도록 구성되거나 그러한 기능에 의해 전체나 일부가 구현될 수 있다. 그에 따라 스토리지 추적 기능부(4544)는 스토리지(4526) 내 한 위치에서 다른 위치로의 콘텐츠 이동(가령, 한 섹터에서 다른 섹터로, 혹은 한 어드레스에서 다른 어드레스로) 또는 메모리(4508) 및 스토리지(4526) 사이의 이동(가령, 스토리지(4526)으부터 실행 코드의 가상 메모리 정황의 스왑이나 로딩과 관련)을 추적할 수 있다.

[0512] 동작 시 메모리/스토리지 보안 계층(4546)은 보안 규칙들(4522)를 수신하고, 보안 규칙들(4522)을 메모리 추적 기능부(4542) 및 스토리지 추적 기능부(4544)로 전송할 수 있다. 그에 따라, 메모리 추적 기능부(4542) 및 스토리지 추적 기능부(4544)는 감시가 인에이블되는지 여부를 나타내고/거나 메모리(4508) 및/또는 스토리지(4526)의 어느 부분들이 감시되어야 하는지를 식별할 수 있는 보안 규칙들(4522)에 기반할 수 있다.

[0513] 메모리 추적 기능부(4542) 및 스토리지 추적 기능부(4544)는 메모리/스토리지 보안 계층(4546)으로 메모리(4508) 및/또는 스토리지(4526)에 시도된 액세스들(가령, 메모리(4508)나 스토리지(4526) 안이나 메모리 및 스토리지(4526) 사이에서의 콘텐츠 이동 시도)을 통지할 수 있다. 메모리/스토리지 보안 계층(4546)은 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM 보안 에이전트(216) 또는 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442), PC 펌웨어 보안 에이전트(444) 또는 O/S 하위 에이전트(450), 도 5의 펌웨어 보안 에이전트(516), 도 7의 마이크로코드 보안 에이전트(708), 도 9의 O/S 하위 에이전트(920), 도 12의 O/S 하위 보안 에이전트 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리/스토리지 보안 계층(4546)은 아래에 상세히 기술된 바와 같이, 그렇게 시도된 액세스들이 악성 소프트웨어 공격을 나타내는지를 판단하기 위해 보안 규칙들(4522) 및/또는 애플리케이션 자산들(4548)에 따라 메모리 추적 기능부(4542) 및/또는 스토리지 추적 기능부(4544)에 의해 보고된 메모리(4508) 및/또는 스토리지(4526)의 액세스 시도들을 분석할 수 있다. 어떤 실시예들에서 메모리/스토리지 보안 계층(4546)은 메모리(4508) 및/또는 스토리지(4526)에, 도 45의 활동들(4532) 및 활동들(4534)에 의해 나타낸 바와 같이

메모리 추적 기능부(4542) 및 스토리지 추적 기능부(4544)에 의해 보고된 엑세스들의 로그, 리스트 또는 다른 표시를 저장할 수 있다. 그에 따라, 메모리(4508) 및 스토리지(4526)에 대해 시도된 개별 액세스들을 분석하는 것 외에, 메모리/저장 보안 계층(4546)는 액세스들의 동향 이력이 악성 소프트웨어의 존재를 나타내는지를 판단하기 위해 보안 규칙들(4522)에 비추어 활동들(4532) 및/또는 활동들(4534)에 구현된 활동들의 이력을 분석할 수 있다.

[0514] 특정 실시예에서, 메모리/스토리지 보안 계층(4546)은 도 2의 SVMM(216)의 기능을 구현하도록 구성되거나 그 기능에 의해 전체나 일부가 구현될 수 있고, 메모리 추적 기능부(4542)는 도 7의 마이크로코드 보안 에이전트(708)의 기능을 구현하도록 구성되거나 그 기능에 의해 전체나 일부가 구현될 수 있으며, 스토리지 추적 기능부(4544)는 도 4의 펌웨어 보안 에이전트(442)의 기능을 구현하도록 구성되거나 그 기능에 의해 전체나 일부가 구현될 수 있다. 그러한 실시예들에서, 메모리 추적 기능부(4542)는 특정 메모리 액세스들에 대해 트래핑할 수 있고, 스토리지 추적 기능부(4544)는 특정 스토리지 액세스들에 대해 트래핑할 수 있으며, 각각은 그러한 트래핑된 이벤트들에 대해 메모리/스토리지 보안 계층(4546)로 통지할 수 있다. 메모리/저장 보안 계층(4546)는 메모리/스토리지에 대해 시도된 개별 액세스들을 분석할 수 있고/있거나 액세스들의 동향 이력이 악성 소프트웨어의 존재를 나타내는지를 판단하기 위해 보안 규칙들(4522)에 비추어 활동들의 이력을 분석할 수 있다.

[0515] 다른 특정 실시예에서, 메모리/스토리지 보안 계층(4546), 메모리 추적 기능부(4542) 및 스토리지 추적 기능부(4544) 각각은 도 2의 단일 SVMM(216)의 기능을 구현하도록 구성되거나 그 기능에 의해 전체나 부분이 모두 구현될 수 있다. 그러한 실시예에서, SVMM(216)은 특정 메모리 액세스들에 대해 트래핑하고, 특정 스토리지 액세스들에 대해 트래핑하고, 메모리 및/또는 스토리지에 대해 시도된 개별 액세스들을 분석하고/거나 액세스들의 동향 이력이 악성 소프트웨어의 존재를 나타내는지를 판단하기 위해 보안 규칙들(4522)에 비추어 활동들의 이력을 분석할 수 있다.

[0516] 애플리케이션(4510), 드라이버(4511), 운영체제(4512) 및/또는 다른 개체에 대한 애플리케이션 자산들(4548)은 그러한 개체 및 그 구성요소들이 메모리(4508) 및/또는 스토리지(4526) 안에 어떻게 상주하는지를 나타내는 맵, 테이블, 리스트 및/또는 다른 데이터 구조를 나타낼 수 있다. 애플리케이션 자산들(4548)은 개체가 저장될 수 있는 메모리(4508) 및/또는 스토리지(4526)의 부분들(가령 메모리 페이지, 메모리 어드레스 범위, 디스크 섹터, 디스크 어드레스 등)을 식별할 수 있다. 상술한 바와 같이, 애플리케이션 자산들(4548) 및/또는 보안 규칙들(4522)에 기반하여, 메모리/스토리지 보안 에이전트(4516)는 메모리 추적 기능부(4542) 및/또는 스토리지 추정 기능부(4544)로부터의 메모리(4508) 및/또는 스토리지(4526)에 대해 시도된 액세스들에 관한 통지들이 악성 소프트웨어 공격을 나타내는지를 판단할 수 있다. 예를 들어 애플리케이션(4510)이 포터블 실행 파일인 실시예들에서, 애플리케이션 자산들(4548)은 애플리케이션(4510)의 실행 코드를 포함하는 메모리(4508) 및/또는 스토리지(4526)에 저장된 애플리케이션(4510)의 부분들을 식별하고/거나 애플리케이션(4510)의 구성요소들이 저장된 메모리(4508) 및/또는 스토리지(4526)의 위치들을 포함하여 애플리케이션(4510)의 데이터를 포함하는 메모리(4508) 및/또는 스토리지(4526)에 저장된 애플리케이션(4510)의 부분들을 식별할 수 있다. 보안 규칙들(4522)은 그러한 애플리케이션(4510)의 예에 대해, 애플리케이션(4510)의 실행 코드를 포함하는 메모리(4508) 및/또는 스토리지(4526)의 부분들에 대한 애플리케이션(4510) 이외의 프로그램들로부터 발생된 쓰기 액세스들이 악성 소프트웨어 공격을 나타낸다는 것을 정의할 수 있다. 추가적으로나 대안적으로, 보안 규칙들(4522)은 그러한 애플리케이션(4510)의 예에 대해, 애플리케이션(4510)의 데이터를 포함하는 메모리(4508) 및/또는 스토리지(4526)의 부분들에 대한 애플리케이션(4510) 이외의 프로그램들로부터 발생된 읽기나 쓰기 액세스들이 악성 소프트웨어 공격을 나타낸다는 것을 정의할 수 있다.

[0517] 다른 예로서, 애플리케이션(4510)이 워드 프로세싱 프로그램인 실시예들에서, 애플리케이션 자산들(4548)은 애플리케이션(4510)의 실행 코드를 포함하는 메모리(4508) 및/또는 스토리지(4526)에 저장된 애플리케이션(4510)의 부분들을 식별하고, 애플리케이션(4510)의 구성요소들이 저장된 메모리(4508) 및/또는 스토리지(4526)의 위치들을 포함하여 애플리케이션(4510)의 스크립트들, 이미지들, 포맷된 텍스트, 메모들 및 기타 데이터를 포함하는 메모리(4508) 및/또는 스토리지(4526)에 저장된 애플리케이션(4510)의 부분들을 식별할 수 있다. 보안 규칙들(4522)은 어떤 애플리케이션(4510) 예에 대해, 애플리케이션(4510)의 데이터를 포함하는 메모리(4508) 및/또는 스토리지(4526)의 부분들에 대한 애플리케이션(4510)이 아닌 다른 프로그램들의 특정 세트로부터 발생된 읽기나 쓰기 액세스들(가령, 운영체제, 안티 악성 소프트웨어 애플리케이션 등으로부터 발생된 액세스들)이 허용될 수 있고, 그러한 특정 프로그램들의 세트가 아닌 프로그램들에 의한 액세스는 악성 소프트웨어 공격을 나타낼 수 있다고 정의할 수 있다.

[0518] 애플리케이션 자산들(4548)은 애플리케이션(4501), 드라이버(4511), 운영체제(4512) 및/또는 다른 프로그램(가

령, 애플리케이션 판매자, 프로그래머 또는 크리에이터), 전자 장치(4504)의 사용자에 의해, 전자 장치(4504)를 포함하는 기업의 관리자에 의해, O/S 하위 보안 에이전트(4516)의 크리에이터에 의해 그리고/또한 다른 적절한 개인에 의해 생성되거나 정의될 수 있다. 어떤 실시예들에서, 애플리케이션 자산들(4548)은 어떤 프로그램에 대해, 그 프로그램의 온 스토리지(on-storage) 구조 및 그 프로그램의 인 메모리(in-memory) 구조 사이의 관계들(가령, 메모리(4508) 및 스토리지(4526) 내 프로그램의 구성요소들 사이의 매핑)을 포함할 수 있다.

[0519] 애플리케이션 자산들(4548)을 모으기 위해, O/S 내 보안 에이전트(4518) 및/또는 O/S 하위 보안 에이전트(4516)가 임의 개의 적절한 기법들을 이용할 수 있다. 예를 들어 O/S 내 보안 에이전트(4518) 및/또는 O/S 하위 보안 에이전트(4516)는 가상 메모리 동작들과 관련하여 운영체제(4512)에 의해 생성될 수 있는 가상 메모리 페이지 스왑들과 관련된 정보를 수집할 수 있다. 예를 들어 윈도우즈에서, O/S 내 보안 에이전트(4518)는 프로토타입 페이지 테이블 엔트리(PTE)를 액세스하여 그 정보를 O/S 하위 보안 에이전트(4516)로 전송할 수 있다. 다른 실시예들에서, O/S 하위 보안 에이전트(4516)가 액세스가 수행될 때마다 메모리(4508) 내 페이지들 및/또는 디스크 상의 섹터들(4526)의 해시, 지문 또는 다른 고유 식별자를 생성할 수 있고, 그 식별자들의 캐시(가령, 메모리(4508) 및/또는 스토리지(4526) 상에 저장될 캐시)를 관리할 수 있다. 그러한 상황에서 O/S 하위 보안 에이전트(4516)는 스토리지(4526)의 어느 섹터가 메모리(4508)의 어느 페이지에 로딩되는지 및 그 반대의 경우를 판단하기 위해 간단한 비교를 적용할 수 있다. 그러한 매핑은 보안 에이전트(4516 및/또는 4518)가 메모리(4508) 및/또는 스토리지(4526) 내 개체들의 특정 정보의 위치들을 추적하게 할 수 있다.

[0520] 메모리(4508) 및 스토리지(4526)의 모든 액세스들에 적용 시 메모리/저장 보안 계층(4546), 메모리 추적 기능부(4542) 및/또는 스토리지 추적 기능부(4544)에 의해 수행되는 감시 및 분석이 전자 장치(4504)의 프로세싱 자원들의 상당 부분을 소비할 수 있기 때문에, 메모리(4508) 및 스토리지(4526)의 감시 및 분석은 특정하게 정의된 상황들에만 이행될 수 있다. 예를 들어 어떤 실시예들에서, 보안 규칙들(4522)은 메모리 추적 기능부(4542) 및/또는 스토리지 추적 기능부(4544)가 메모리(4508) 및/또는 스토리지(4526)의 특정 부분들(가령, 운영체제들이나 중요 드라이버들이나 애플리케이션들을 포함하는 부분들)만을 감시하라고 정의할 수 있다. 다른 예로서, 동일하거나 다른 실시예들에서, 보안 규칙들(4522)은 메모리 추적 기능부(4542) 및/또는 스토리지 추적 기능부(4544)가 메모리(4508) 및/또는 스토리지(4526)의 특정 부분 내 프로그램이 의심스럽다는 것을 다른 표시들이 보이고/거나 표시들이 악성 소프트웨어 공격이 일어났을 것임을 보일 경우 그 프로그램을 감시한다고 정의할 수 있다. 추가 예로서, 메모리 추적 기능부(4512) 및/또는 스토리지 추적 기능부(4544)는 스토리지(4526)로부터의 실행 코드의 가상 메모리 정황 스왑이나 로딩의 경우에서와 같이, 콘텐츠가 메모리(4508)로부터 스토리지(4526)으로 로딩될 때나 그 반대의 경우일 때를 제외하고 메모리 트래핑 및 스토리지 트래핑을 포기할 수 있다.

[0521] 동작 시, 위에서 언급한 바와 같이 메모리/스토리지 보안 계층(4546)은 보안 규칙들(4522) 및/또는 애플리케이션 자산들(4548)에 비추어 메모리(4508) 및/또는 스토리지(4526)으로 보고되는 액세스들을 분석함으로써 메모리(4508) 및/또는 스토리지(4526)를 보호할 수 있다. 메모리(4508) 및/또는 스토리지(4526)에 대해 시도된 액세스에 대한 통지를 수신한 후, 메모리/스토리지 보안 계층(4546)은 그 시도된 액세스를 요청한 개체(가령, 운영체제(4512), 드라이버(4511) 또는 애플리케이션(4510))의 아이디를 판단할 수 있다. 예를 들어, O/S 내 보안 에이전트(4518)는 운영체제(4512)로부터 특정 메모리(4508) 및/또는 스토리지(4526)의 요청 개체에 대한 정황 정보를 모을 수 있고, 그 정보를 메모리/스토리지 보안 계층(4546)으로 전송할 수 있다. 또한 메모리/스토리지 보안 계층(4546)은 요청한 개체의 아이디를 검증하고 그 개체가 악성 소프트웨어에 의해 손상되었는지 여부를 판단할 수 있다(가령, 메모리에 저장된 개체의 맵이나 해시를 그 개체에 대해 알려지고 예상된 맵이나 해시와 비교하거나 메모리에 저장된 개체를 악성 소프트웨어 존재에 대해 검색함으로써). 또한 메모리/스토리지 보안 계층(4546)은 그 개체가 해당 요청을 하는 것이 허가되는지 여부를 판단할 수 있다(가령, 보안 규칙들(4522) 및/또는 애플리케이션 자산들(4548)에 기반하여 그 개체가 메모리(4508)나 스토리지(4526)의 특정 부분을 액세스하도록 허가되는지 여부를 판단). 또한 메모리/스토리지 보안 계층(4546)은 시도된 액세스와 관련된 콘텐츠(가령, 판독, 기입 또는 실행되는 데이터나 실행 코드)가 악성 소프트웨어를 포함하는지 여부를 판단하기 위해 그 콘텐츠를 검색할 수 있다. 또한 메모리/스토리지 보안 계층(4546)은 (가령, 활동들(4532) 및/또는 활동들(4534)에 저장된 것과 같은) 액세스들의 이력에 대한 동향 분석이 악성 소프트웨어의 존재(가령, 허가되지 않은 개체들에 의해 O/S(4512)의 보호 부분들의 액세스 시도들)를 나타내는지를 판단할 수 있다. 메모리/스토리지 보안 계층(4546)은 시도된 액세스가 악성 소프트웨어와 관련되어 있다고 판단된 경우 교정 액션을 취할 수 있다. 교정 액션은 시도된 액세스 금지, 요청한 개체 종료, 요청한 개체 복구, 악성 소프트웨어 관련 이벤트의 발생을 보호 서버(202)로 전송 및/또는 어떤 다른 적절한 액션을 포함할 수 있다.

[0522] 특정 예로서, (스토리지 추적 기능부(4544)로부터의 통지에 의해 지시된 것과 같이) 스토리지(4526)의 특정 섹

터에 대한 요청에 따라, 메모리/스토리지 보안 계층(4546)은 적어도 보안 규칙들(4522)에 기반하여 특정 섹터가 액세스되어야 하는지 여부를 판단할 수 있다. 또한 메모리/스토리지 보안 계층(4546)은 시도된 액세스와 관련된 콘텐츠(가령, 시도된 액세스 관련 판독, 기입 또는 실행되는 데이터나 실행 코드)가 잠정적 악성 소프트웨어 감염의 염려가 없는지 여부를 판단하기 위해 그 콘텐츠를 검색할 수 있다. 또한 메모리/스토리지 보안 계층(4546)은 적어도 보안 규칙들(4522)에 기반하여, 시도된 액세스를 요청한 개체가 특정 섹터를 액세스하는 데 허가된 것인지 여부를 판단할 수 있다. 그러한 판단이 시도된 액세스가 악성 소프트웨어와 관련되지 않았다는 것을 나타내는 경우, 메모리/스토리지 보안 계층(4546)은 시도된 액세스를 허용할 수 있다.

[0523] 다른 특정 예에서, (메모리 추적 기능부(4542)로부터의 통지에 의해 나타낸 바와 같이) 메모리의 특정 페이지에 대한 요청에 따라, 메모리/스토리지 보안 계층(4546)은 그 시도된 액세스와 관련된 콘텐츠(가령, 시도된 액세스와 관련하여 읽혀지거나 기입되거나 실행될 데이터나 실행 코드)가 잠정적 악성 소프트웨어 감염의 염려가 없는지 여부를 판단하기 위해 그 콘텐츠를 검색할 수 있다. 또한 메모리/스토리지 보안 계층(4546)은 적어도 보안 규칙들(4522)에 기반하여, 시도된 액세스를 요청한 개체가 특정 페이지를 액세스하는 데 허가된 것인지 여부를 판단할 수 있다. 또한, 그 시도된 액세스가 스토리지(4526)로부터 메모리(4508)로의 이동이면, 메모리/스토리지 보안 계층(4546)은 적어도 보안 규칙들(4522)에 기반하여, 콘텐츠가 이동되기 위해 나오는 스토리지(4526)의 특정 부분이 안전한 소스인지를 판단할 수 있다. 그러한 판단이 시도된 액세스가 악성 소프트웨어와 관련되지 않았다는 것을 나타내는 경우, 메모리/스토리지 보안 계층(4546)은 시도된 액세스를 허용할 수 있다.

[0524] 또한 메모리(4508)나 스토리지(4526)의 특정 부분에 대해 적용되는 보안 규칙들(4522) 및 보호는 콘텐츠가 메모리(4508) 및 스토리지(4526) 사이에서, 또는 메모리(4508)의 서로 다른 부분들 사이에서, 또는 스토리지(4526)의 서로 다른 부분들 사이에서 이동할 때 과도기적으로 적용될 수 있다. 따라서, 예컨대 메모리(4508)의 특정 부분에 있는 콘텐츠를 메모리(4508)나 스토리지(4526)의 다른 부분으로 이동할 때 보안 규칙들(4522)의 특정 세트가 그 콘텐츠에 적용되는 경우, 메모리/스토리지 보안 계층(4546)은 메모리(4508)나 스토리지(4526)의 목적지 부분에 적용할 보안 규칙들(4522)을 업데이트할 수 있다.

[0525] 도 46은 전자 장치의 메모리 및 스토리지를 보호하는 방법(4600)의 실시예이다. 단계 4605에서 메모리/스토리지 보안 계층은 메모리 추적 기능부 및 스토리지 추적 기능부에 보안 규칙들을 전송할 수 있다. 여기 개시된 메모리 및 스토리지의 악성 소프트웨어로부터의 보호를 위한 시스템들과 방법들은 상당한 프로세서, 메모리 및/또는 다른 자원들을 소비하기 때문에, 메모리 또는 스토리지의 특정 위치가 악성 소프트웨어 공격들을 당하기가 특히 쉬울 때에만 그러한 시스템들과 방법들을 이용하는 것이 바람직할 수 있다. 메모리나 스토리지의 부분은 예컨대 그것이 운영체제나 보안 애플리케이션의 부분을 포함하는 경우나, 전자 장치 상에서 이전의 공격에 대한 표시가 보여졌거나 검출되었던 경우, 악성 소프트웨어 공격에 취약할 수 있다.

[0526] 단계 4610에서 메모리 추적 기능부 및 스토리지 추적 기능부는 보안 규칙들에 따라 액세스들을 감시할 수 있다. 감시를 위해 메모리 추적 기능부 및 스토리지 추적 기능부는 메모리/스토리지 보안 계층으로부터 수신된 보안 규칙들에 의해 식별되는 메모리나 스토리지의 특정 부분들에 대해 시도된 액세스들(가령, 시도된 읽기, 쓰기 또는 실행들)에 대해 트래핑 또는 트리거할 수 있다.

[0527] 단계 4615에서, 메모리 추적 기능부 및/또는 스토리지 추적 기능부는 메모리/스토리지 보안 계층에 메모리 및/또는 스토리지에 대해 시도되는 액세스들에 대한 통지를 전송할 수 있다.

[0528] 단계 4620에서, 메모리/스토리지 보안 계층은 메모리/스토리지 보안 계층으로의 통지에서 식별된 시도된 액세스를 요청한 개체(가령, 운영체제, 드라이버 또는 애플리케이션)의 아이디를 판단할 수 있다. 예를 들어, O/S 내 보안 에이전트는 메모리/스토리지 보안 계층과 통신하여, 운영체제로부터 특정 메모리 및/또는 스토리지의 요청 개체에 대한 정황 정보를 모을 수 있고, 그 정보를 메모리/스토리지 보안 계층으로 전송할 수 있다.

[0529] 단계 4625에서, 메모리/스토리지 보안 계층은 요청하는 개체의 아이디를 검증하고 그 개체가 악성 소프트웨어에 의해 훼손되었는지 여부를 판단할 수 있다. 예를 들어 메모리/스토리지 보안 계층은 메모리에 저장된 개체의 맵이나 해시를 개체에 대해 알려진 예상된 맵이나 해시와 비교할 수 있다. 다른 예에서 메모리/스토리지 보안 계층은 메모리에 저장된 것과 같은 개체를 악성 소프트웨어 존재에 대해 검색할 수 있다.

[0530] 단계 4630에서 메모리/스토리지 보안 계층은 해당 개체가 해당 요청을 하는 것이 허가되는지 여부를 판단할 수 있다. 예를 들어 메모리/스토리지 보안 계층은 보안 규칙들 및/또는 애플리케이션 자산들을 조회하여, 해당 개체가 메모리(4508)나 스토리지(4526)의 특정 부분을 액세스하도록 허가되는지 여부를 판단할 수 있다. 단계 4635에서 메모리/스토리지 보안 계층은 시도된 액세스와 관련된 콘텐츠(가령, 읽혀지거나 기입되거나 실행되는

데이터나 실행 코드)를 분석할 수 있다. 예를 들어 메모리/스토리지 보안 계층은 콘텐츠가 악성 소프트웨어를 포함하는지 여부를 판단하기 위해, 시도된 액세스와 관련된 콘텐츠를 검색할 수 있다.

[0531] 단계 4640에서 메모리/스토리지 보안 계층은 메모리 및/또는 스토리지에 대한 액세스들의 이력을 분석할 수 있다. 그 이력은 전자 장치의 메모리 및/또는 스토리지에 로그나 리스트로서 저장될 수 있다. 그러한 분석은 이력이 악성 소프트웨어의 존재를 나타내는지를 판단하기 위해 메모리 및/또는 스토리지에 대한 액세스들의 히스트리에 대한 동향 분석을 포함할 수 있다.

[0532] 단계 4645에서, 메모리/스토리지 보안 계층은 (가령, 단계 4620-4640 중 하나 이상의 분석 및 판단들에 기반하여) 메모리 추적 기능부 및/또는 스토리지 추적 기능부에 의해 보고된 메모리 및/또는 스토리지에 대해 시도된 액세스가 악성 소프트웨어에 의해 영향을 받았다는 것을 나타내는지 여부를 판단할 수 있다. 또한, 메모리/스토리지 보안 계층이 변형된 콘텐츠가 악성 소프트웨어 유사 양태에 의해 영향을 받았다고 판단하는 경우, 메모리/스토리지 보안 계층은 교정 액션(가령, 제거 액션, 격리 및/또는 악성 소프트웨어 중성화)을 취할 수 있다. 또한 일부 실시예들에서, 메모리/스토리지 보안 계층은 악성 소프트웨어 유사 양태 발생에 관한 정보(가령, 법의학적 정보)를 보호 서버로 전송할 수 있다.

[0533] 단계 4650에서, 메모리/스토리지 보안 계층은 액세스에 대한 통지를 전자 장치의 메모리 및/또는 스토리지 상에 저장된 액세스들의 로그나 리스트에 추가할 수 있다. 액세스들의 이력에 대한 동향 분석을 수행하기 위해, 메모리/스토리지 보안 계층은 그 저장된 로그나 리스트를 추후 액세스할 수 있다. 단계 4650의 완료 후, 방법(4600)은 다시 단계 4605로 돌아갈 수 있다.

[0534] 도 47은 전자 장치(4701) 상에 실행되는 운영체제(4713)의 객체들에 대한 액세스를 보호하기 위한 시스템(4700)의 실시예이다. 시스템(4700)은 운영체제(4713) 상에서 실행되는 소프트웨어 기반 개체들에 의한 객체들(4706) 및/또는 객체 관리자(4704)를 액세스하려는 악의적 시도들을 검출하기 위해 전자 장치(4701) 상에서 동작하도록 구성된 O/S 하위 트래핑 에이전트(4720) 및 트리거된 이벤트 핸들러(4722)를 포함할 수 있다. 또한, O/S 하위 트래핑 에이전트(4720) 및 트리거된 이벤트 핸들러(4722)는 객체들(4706) 및/또는 객체 관리자(4704)에 대한 액세스를 언제 트래핑할지와 트래핑된 동작과 관련된 트리거된 이벤트를 어떻게 처리할지를 결정하기 위해 하나 이상의 보안 규칙들(4708)을 사용하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(4720) 및 트리거된 이벤트 핸들러(4722)는 트리거된 이벤트를 허용하거나 거부하거나 다른 교정 액션을 수행하도록 구성될 수 있다.

[0535] 전자 장치(4701)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701), 도 9의 전자 장치(901), 도 12의 전자 장치(12) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(4701)는 메모리(4703)에 연결된 하나 이상의 프로세서들(4702)을 포함할 수 있다. 프로세서(4702)는 도 2의 프로세서(208), 도 4의 프로세서(408), 도 7의 프로세서(702), 도 9의 프로세서(902), 도 12의 프로세서(1202) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(4701)는 O/S 내 보안 에이전트(4719) 및 객체들(4706)을 관리할 객체 관리자(4704)를 포함할 수 있는 운영체제(4713)를 포함할 수 있다. 운영체제(4713)는 도 1의 운영체제들(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 9의 운영체제(913), 도 12의 운영체제(1213) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(4719)는 도 1의 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418), 도 7의 O/S 내 보안 에이전트(719), 도 9의 O/S 내 보안 에이전트(919), 도 12의 O/S 내 보안 에이전트(1219) 및/또는 이들의 어떤 적절한 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 보안 규칙들(4708)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(220, 222), 도 4의 보안 규칙들(420, 434, 436, 438), 도 5의 보안 규칙들(518), 도 7의 보안 규칙들(707, 721, 723), 도 9의 보안 규칙들(908, 921), 도 12의 보안 규칙들(1208, 1221) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보호 서버(4714)는 도 1의 보호 서버(102), 도 2의 보호 서버(202) 및/또는 이들의 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0536] 메모리(4703)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703), 도 9의 메모리(903), 도 12의 메모리(1203) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(4703)는 메모리(4703)에 대한 액세스를 가상화하도록 구성된 가상 메모리 시스템을 사용하여 구현될 수 있다. 가상 메모리 시스템에서, 운영체제(4713) 상에서 실행되는 소프트웨어 프

로세스들이 프로세스가 메모리의 인접 블록으로서 취급할 수 있는 가상 어드레스 공간과 함께 제공될 수 있다. 실제로, 가상 어드레스 공간은 물리적 메모리의 다양한 영역들에 걸쳐 분산될 수 있다. 프로세스가 메모리에 대한 액세스를 요청할 때, 운영체제(4713)는 해당 프로세스의 가상 메모리 어드레스를 데이터가 실제로 저장된 메모리(4703) 내 물리적 어드레스로 매핑하는 일을 담당할 수 있다. 가상 어드레스 공간은 가상 메모리 페이지들이라 불리는 인접하는 가상 메모리 어드레스들의 고정 크기 블록들로 분할될 수 있다. 페이지 테이블은 가상 메모리 페이지로부터 그 가상 메모리 페이지가 저장된 메모리(4703) 내 대응하는 물리적 어드레스로의 매핑을 저장하는 데 사용될 수 있다. 페이지 테이블은 읽기, 쓰기 및/또는 실행과 같은 다양한 액세스 허가 사항들을 포함하여 주어진 가상 메모리 페이지에 대해 허가될 수 있는 액세스 타입을 특정할 수 있다. 어떤 프로세스가 관련 가상 메모리 페이지의 액세스 허가 사항들에 의해 허가되지 않는 방식으로 가상 메모리 어드레스를 액세스하고자 시도할 때, 그 시도는 거부될 수 있다.

[0537] O/S 하위 트래핑 에이전트(4720)는 도 1의 O/S 하위 트래핑 에이전트(104), 도 2의 SVMM(216), 도 4의 펌웨어 보안 에이전트들(440, 442) 및/또는 PC 펌웨어 보안 에이전트(444), 도 5의 펌웨어 보안 에이전트(516) 및/또는 도 7의 마이크로코드 보안 에이전트(708), 도 9의 O/S 하위 에이전트(920), 도 12의 O/S 하위 보안 에이전트(1220) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 트리거된 이벤트 핸들러(4722)는 도 1의 트리거된 이벤트 핸들러(108), 도 2의 SVMM 보안 에이전트(217), 도 4의 O/S 하위 에이전트(450), 도 7의 O/S 하위 에이전트(712), 도 9의 트리거된 이벤트 핸들러(922) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 다양한 실시예들에서 O/S 하위 트래핑 에이전트(4720)의 기능 중 일부는 트리거된 이벤트 핸들러(4722)에 의해 수행될 수 있고/있거나, 트리거된 이벤트 핸들러(4722)의 기능 중 일부가 O/S 하위 트래핑 에이전트(4720)에 의해 수행될 수 있다. 또한 O/S 하위 트래핑 에이전트(4720) 및 트리거된 이벤트 핸들러(4722)는 동일한 소프트웨어 모듈 안에서 구현될 수 있다.

[0538] 운영체제(4713)는 운영체제(4713)와 관련된 각각의 자원을 나타내기 위해 객체들(4706)을 이용하는 객체지향 환경으로서 구현될 수 있다. 예를 들어 운영체제(4713)는 드라이버들(4711), 애플리케이션들(4710), 프로세스들, 메모리(4703), 파일들 및/또는 폴더들, 물리적 장치들 및/또는 운영체제(4713)에 의해 사용되는 다른 자원을 나타내는 객체들을 가질 수 있다. 또한, 운영체제(4713) 상에서 실행되는 각각의 애플리케이션(4710) 및/또는 다른 소프트웨어 프로세스 역시 특정 애플리케이션(4710)이나 소프트웨어 프로세스에 의해 사용되는 자원들을 표현하기 위한 객체들(4706)을 사용할 수 있다. 객체들(4706)은 객체(4706)의 특정 타입에 고유할 수 있고 특정 객체(4706)의 데이터를 조작하는데 사용될 수 있는 객체 함수들(4730)을 포함할 수 있다. 객체들(4706)은 헤더 및 바디를 포함할 수 있는 데이터 구조에 의해 표현될 수 있다. 객체(4706)의 헤더는 모든 객체들에 공통적인 관리 필드들을 포함할 수 있다. 이 필드들은 객체들(4706)을 관리하기 위해 객체 관리자(4704)에 의해 사용될 수 있다. 객체(4706)의 헤더는 예컨대 객체(4706)를 식별하기 위한 객체 이름 및/또는 객체(4706)와 관련된 액세스 허가 사항을 특정하는 보안 서술자들을 포함할 수 있다. 객체(4706)의 바디는 객체(4706)의 특정 타입에 고유한 객체 고유 데이터 필드들을 포함할 수 있다.

[0539] 객체 관리자(4704)는 운영체제(4713)의 객체들(4706)을 관리하기 위해 운영체제(4713) 안에서 실행되도록 구성될 수 있다. 객체 관리자(4704)는 객체들(4706)을 관리하는 데 사용될 수 있는 다양한 객체 관리자 함수들(4726)을 이용하여 구현될 수 있다. 예를 들어 객체 관리자 함수들(4726)은 생성, 삭제, 데이터 수정 및/또는 객체들(4706)의 설정사항 변경에 사용되는 함수들을 포함할 수 있다. 객체 관리자 함수들(4726)은 하나 이상의 서브 함수들(4728)을 이용하여 구현될 수 있다. 마이크로소프트 윈도우즈 운영체제의 객체 관리자(4704)에 의해 사용되는 객체 관리자 함수들(4726)의 예들이 표 1에서 보여질 수 있다.

ObAssignObjectSecurityDescriptor	ObFindHandleForObject	ObQueryObjectAuditingByHandle
ObAssignSecurity	ObFreeObjectCreateInfoBuffer	ObQuerySecurityDescriptorInfo
ObAuditInheritedHandleProcedure	ObGetHandleInformation	ObQueryTypeInfo
ObCheckCreateObjectAccess	ObGetHandleInformationEx	ObQueryTypeName
ObCheckObjectAccess	ObGetObjectInformation	ObReferenceFileObjectForWrite
ObClearProcessHandleTable	ObGetObjectSecurity	ObReferenceObjectByHandle
ObCloseHandle	ObGetProcessHandleCount	ObReferenceObjectByName
ObCreateObject	ObGetSecurityMode	ObReferenceObjectByPointer
ObCreateObjectType	ObInheritDeviceMap	ObReferenceObjectEx
ObDeassignSecurity	ObInitProcess	ObReferenceObjectSafe
ObDeleteCapturedInsertInfo	ObInitSystem	ObReferenceProcessHandleTable
ObDereferenceDeviceMap	ObInitializeFastReference	ObReferenceSecurityDescriptor
ObDereferenceObject	ObInsertObject	ObReleaseObjectSecurity
ObDereferenceObjectDeferDelete	ObIsLUIDDeviceMapsEnabled	ObSetDeviceMap
ObDereferenceObjectEx	ObIsObjectDeletionInline	ObSetDirectoryDeviceMap
ObDereferenceProcessHandleTable	ObKillProcess	ObSetHandleAttributes
ObDereferenceSecurityDescriptor	ObLogSecurityDescriptor	ObSetSecurityDescriptorInfo
ObDupHandleProcedure	ObMakeTemporaryObject	ObSetSecurityObjectByPointer
ObDuplicateObject	ObOpenObjectByName	ObShutdownSystem
ObEnumerateObjectsByType	ObOpenObjectByPointer	ObSwapObjectNames
ObFastDereferenceObject	ObPerfDumpHandleEntry	ObValidateSecurityQuota
ObFastReferenceObject	ObPerfHandleTableWalk	ObWaitForSingleObject
ObFastReferenceObjectLocked	ObQueryDeviceMapInformation	
ObFastReplace	ObQueryNameString	

[0540] 표 1: 마이크로소프트 윈도우즈 객체 관리자 함수들의 예들

[0541] 메모리 맵(4718)은 도 12의 메모리 맵(1206)의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 메모리 맵(4718)은 파일, 레코드, 데이터 구조, 또는 어떤 다른 적절한 개체 안에서 구현될 수 있다. 메모리 맵(4718)은 O/S 하위 트래핑 에이전트(4720)의 일부로서 포함될 수 있거나, O/S 하위 트래핑 에이전트(4720)와 통신 가능하게 연결될 수 있다. 메모리 맵(4718)은 다양한 객체 자원들(4734)의 메모리(4703) 내 위치에 관한 정보를 포함할 수 있다. 객체 자원들(4734)은 예컨대 객체 관리자(4704), 객체 관리자 함수들(4726) 및/또는 서브 함수들(4728), 객체들(4706) 및/또는 객체 함수들(4730)을 포함할 수 있다. 메모리 맵(4718)은 가상 메모리 내 메모리 페이지들, 물리적 메모리 내 어드레스 범위들 및/또는 특정 객체 자원(4734)이 저장될 수 있는 디스크 상의 위치에 관한 정보를 포함할 수 있다. O/S 하위 트래핑 에이전트(4720)는 가상 메모리 페이지나 물리적 메모리 어드레스 내 어떤 주어진 콘텐츠의 아이디나 소유자를 판단하기 위해 메모리 맵(4718)을 이용하도록 구성될 수 있다.

[0542] O/S 하위 트래핑 에이전트(4720)는 메모리 맵(4718)의 콘텐츠를 결정, 전개, 및/또는 상주시킬 수 있다. 그렇게 하기 위해, O/S 하위 트래핑 에이전트(4720)는 보안 규칙들(4708), 보호 서버(4714), 또는 메모리 맵(4718)에 정보를 위치시키기 위한 어떤 다른 적절한 정보 소스를 액세스할 수 있다. O/S 하위 트래핑 에이전트(4720)는 예컨대 운영체제(4713)의 동작을 프로파일링하고 그런 다음 메모리의 어디에 다양한 객체 자원들(4734)이 위치되는지를 판단함으로써 메모리 맵(4718)을 구축할 수 있다. O/S 하위 트래핑 에이전트(4720)은 O/S 내 보안 에이전트(4719)와 연계하여, 개별 함수들의 실행 스택들을 가로질러 객체 함수들(4730), 객체 관리자 함수들(4726) 및/또는 객체 관리자 서브함수들(4728)의 메모리 내 위치들을 식별할 수 있다. O/S 하위 트래핑 에이전트(4720)는 메모리 맵(4718) 안에 메모리의 소유권 및 콘텐츠를 매핑하기 위해 운영체제(4713), 애플리케이션(4710) 또는 드라이버(4711)와 같은 운영체제 레벨에 있는 개체들로부터 물리적 메모리나 가상 메모리에 대한 요청들을 인터셉트할 수 있다. 예를 들어, 객체 자원들(4734)을 액세스하려는 시도들이 트래핑될 때, O/S 하위 트래핑 에이전트(4720)는 어떤 객체 자원들(4734)이 액세스되고 있고/있거나 어떤 개체가 특정 객체 자원(4734)의 액세스를 담당하는지를 판단하기 위해 O/S 내 보안 에이전트(4719)와 통신하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(4720)는 메모리 맵(4718)이 상주될 수 있도록 어떤 개체들이 메모리 안에 로딩될지를 판단하기 위해 O/S 내 보안 에이전트(4719)와 통신할 수 있다. 메모리 맵(4718)은 물리적 메모리, 가상 메모리에 대한 메모리 매핑 및/또는 그 둘 사이의 매핑들을 포함할 수 있다.

[0543] 객체지향 운영체제(4713)의 실질적으로 모든 자원들이 객체들(4706)에 의해 표현될 수 있기 때문에, 악성 소프트웨어는 객체들(4706)을 공격하여 운영체제(4713)의 보안을 훼손시키고자 시도할 수 있다. O/S 하위 트래핑 에이전트(4720) 및/또는 트리거된 이벤트 핸들러(4722)가 악성 소프트웨어로부터 객체들(4706)을 보호하도록 구성될 수 있다. 악성 소프트웨어에 의한 객체들(4706)에 대한 공격은 객체(4706) 및/또는 객체 관리자(4704)를 조작하려는 허가되지 않은 시도와 같이, 부적절한 객체 자원들(4734)로의 시도를 포함할 수 있다. 예를 들어 운영체제(4713)는 운영체제(4713) 상에서 실행되는 각각의 소프트웨어 프로세스를 나타내는 객체들(4706)을 포함할 수 있으며, 악성 소프트웨어가 운영체제(4713) 상에서 실행 중일 수 있는 보안 애플리케이션과 관련된 특정 프로세스 객체(4706)를 삭제할 수 있다. 이런 방식으로, 보안 애플리케이션의 실행이 중단되어, 악성 소프트웨어가 보안 소프트웨어의 보호장치들을 전복시키고 더 나아가 악성 활동들을 수행하게 할 수 있다. 다른 예

로서, 악성 소프트웨어는 악성 소프트웨어 스캐너들로부터 자신들을 위장하기 위해 객체 이름들과 같은 악성 소프트웨어 자신의 객체들(4706)의 필드들을 편집할 수 있다. 악성 소프트웨어는 또한 객체(4706)의 보안 설정사항들을 변경하고자 시도할 수 있다. 예를 들어 악성 소프트웨어는 어떤 개체에 의해 파일이 액세스될 수 있게, 코어 운영체제(4713)를 나타내는 객체(4706)의 액세스 허가를 변경하고자 시도할 수 있다. 악성 소프트웨어는 객체 관리자 함수들(4726), 객체 관리자 서브 함수들(4728) 및/또는 객체 함수들(4730)을 호출함으로써 상술한 공격들을 간접 수행할 수 있다. 악성 소프트웨어는 특정 함수의 호출자로서 스스로를 숨기기 위해, 함수를 호출하기에 앞서 객체의 보안 설정사항을 바꿀 수 있다. 객체 관리자 함수(4726)는 허가되지 않은 실행으로부터 보호되지만 하나 이상의 서브 함수들(4728)은 보호되지 않는 경우, 악성 소프트웨어는 객체 관리자 함수(4730)가 아니라 객체 관리자 서브 함수들(4728)을 호출할 수 있다. 이런 방식으로 악성 소프트웨어는 객체 관리자 함수(4726)의 하나 이상의 비보호 서브 함수들(4728)을 호출함으로써 객체 관리자 함수(4726)의 보호를 피해갈 수 있다. 악성 소프트웨어는 객체들(4706) 및/또는 객체 관리자(4704)가 저장되는 메모리(4703) 안의 위치들을 액세스함으로써 객체들(4706)을 직접 공격할 수도 있다.

[0544] O/S 하위 트래핑 에이전트(4720) 및/또는 트리거된 이벤트 핸들러(4722)는 객체 자원들(4734)을 액세스하려는 시도들을 보호함으로써 객체(4706)에 대한 악성 소프트웨어 공격들을 방지하도록 구성될 수 있다. 예를 들어 O/S 하위 트래핑 에이전트(4720)는 객체 자원들(4734)을 읽거나 쓰거나 실행하려는 시도들을 트래핑하도록 구성될 수 있으며, 트리거된 이벤트 핸들러(4722)는 트래핑된 시도들이 악성 소프트웨어를 나타내는지 여부를 판단하도록 구성될 수 있다. 객체 자원들(4734)을 액세스하려는 트래핑된 시도들은 객체 관리자(4704)의 함수들(4726) 및/또는 객체 서브함수들(4728), 객체들(4706)의 함수들(4730)을 실행하려는 시도들 및/또는 객체 자원들(4734)이 저장된 메모리(4703) 내 위치들을 직접 액세스하려는 시도들을 포함할 수 있다. O/S 하위 트래핑 에이전트(4720)의 메모리 트래핑 기능은 도 12의 O/S 하위 보안 에이전트(1220)의 기능을 구현하도록 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0545] O/S 하위 트래핑 에이전트(4720)는 어떤 적절한 방식으로 객체 함수들(4730), 객체 관리자 함수들(4726) 및/또는 객체 관리자 서브 함수들(4728)의 코드를 실행하려는 시도들을 트래핑하도록 구성될 수 있다. 예를 들어 O/S 하위 트래핑 에이전트(4720)는 특정 함수의 코드가 저장될 수 있는 메모리 위치들에서 코드를 실행하려는 시도들을 트래핑하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(4720)는 트래핑을 요하는 함수들의 메모리(4703) 내 위치들을 식별하기 위해 메모리 맵(4718)을 조회하도록 구성될 수 있다. 트래핑된 코드 실행 시도들은 가상 메모리 레벨 또는 물리적 메모리 레벨에서 트래핑될 수 있다. 예를 들어 O/S 하위 트래핑 에이전트(4720)는 특정 함수와 관련된 가상 메모리 페이지를 실행하려는 시도를 트래핑하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(4720)는 또한, 특정 함수의 코드가 저장될 수 있는 물리적 메모리 어드레스에 대응하는 가상 메모리 어드레스에서 코드를 실행하려는 시도를 트래핑하도록 구성될 수 있다. 그러한 트래핑은 가상 메모리 어드레스로부터 물리적 메모리 어드레스로의 변환에 앞서 일어날 수 있다. 또 다른 실시예에서, O/S 하위 트래핑 에이전트(4720)는 특정 함수의 코드가 저장될 수 있는 물리적 메모리 어드레스에서 코드를 실행하려는 시도를 트래핑하도록 구성될 수 있다. 그러한 트래핑은 가상 메모리 어드레스로부터 물리적 메모리 어드레스로의 변환 후에 일어나거나, 가상 메모리를 통해 일차 변환되기 전에 물리적 메모리 어드레스의 코드를 실행하려는 직접적 시도 후에 일어날 수도 있다. 객체 함수(4730), 객체 관리자 함수(4726) 및/또는 객체 관리자 서브 함수(4728)를 실행하려는 시도를 트래핑한 후, O/S 하위 트래핑 에이전트(4720)는 트래핑된 시도와 관련된 트리거된 이벤트를 생성하여 그 트래핑된 시도를 처리하기 위해 트리거된 이벤트 핸들러(4722)로 보낼 수 있다.

[0546] O/S 하위 트래핑 에이전트(4720)는 메모리(4703) 내 객체 자원들(4734)을 액세스 하기 위한 시도들을 트래핑하도록 구성될 수 있다. 예를 들어 O/S 하위 트래핑 에이전트(4720)는 객체들(4706) 및/또는 객체 관리자(4704)를 저장하는데 사용되는 메모리 위치들을 액세스하려는 시도들을 트래핑하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(4720)는 또한, 객체 함수들(4730), 객체 관리자 함수들(4726) 및/또는 객체 관리자 서브 함수들(4728)의 코드를 저장하는 데 사용되는 메모리 위치들로의 쓰기 시도들을 트래핑하도록 구성될 수 있다. 그러한 트래핑은 악성 소프트웨어가 객체 함수들(4730), 객체 관리자 함수들(4726) 및/또는 객체 관리자 서브 함수들(4728)의 코드를 악성 코드로 덮어쓰는 것을 방지할 것이다. 어떤 실시예들에서 O/S 하위 트래핑 에이전트(4720)는 객체 자원들(4734)의 메모리(4703) 내 위치들을 식별하기 위해 메모리 맵(4718)을 이용할 수 있다. 일 실시예에서 O/S 하위 트래핑 에이전트(4720)는 객체 자원들(4734)의 가상 메모리 어드레스들에 대응하는 가상 메모리 페이지들을 액세스하려는 시도들을 트래핑하도록 구성될 수 있다. 다른 실시예에서 O/S 하위 트래핑 에이전트(4720)는 객체 자원들(4734)이 저장될 수 있는 물리적 메모리 어드레스들에 대응하는 가상 메모리 어드레스들을 액세스하려는 시도들을 트래핑하도록 구성될 수 있다. 그러한 트래핑은 가상 메모리 어드레스로부터 물리적 메모리 어드레스로의 변환에 앞서 일어날 수 있다. 또 다른 실시예에서 O/S 하위 트래핑 에이전트

(4720)는 객체 자원들(4734)이 저장될 수 있는 물리적 메모리 어드레스들을 액세스하려는 시도들을 트래핑하도록 구성될 수 있다. 그러한 트래핑은 가상 메모리 어드레스로부터 물리적 메모리 어드레스로의 변환 후에 일어나거나, 가상 메모리를 통해 일차 변환되기 전에 물리적 메모리 어드레스를 액세스하려는 직접적 시도 후에 일어날 수도 있다.

[0547] 일 실시예에서 O/S 하위 트래핑 에이전트(4720)는 특정 객체 자원(4734)를 액세스하기 위한 요청을 담당하는 요청 개체를 판단하기 위해 O/S 내 보안 에이전트(4719)를 조회하도록 구성될 수 있다. 다른 실시예에서 O/S 하위 트래핑 에이전트(4720)는 요청이 나왔던 가상 메모리 페이지를 판단하고, 메모리 맵(4718)을 조회하여 그 메모리 페이지가 그 안에 매핑된 어떤 요소들과 관련되는지 여부를 판단하도록 구성될 수 있다. 또 다른 실시예에서 O/S 하위 트래핑 에이전트(4720)는 요청하는 요소의 가상 메모리 페이지의 해시나 서명을 판단하고 그것을 알려진 개체들의 해시들 및 서명들과 비교하도록 구성될 수 있다. 객체 자원들(4734)을 액세스하려는 시도를 트래핑하고 요청한 개체를 식별한 후, O/S 하위 트래핑 에이전트(4720)는 요청된 특정 객체 자원(4734), 액세스 타입 및 요청한 개체를 포함하여, 트래핑된 시도와 관련된 정보를 포함하는 트리거된 이벤트를 생성할 수 있다. O/S 하위 트래핑 에이전트(4720)는 트리거된 이벤트를 트래핑된 시도를 처리하는 트리거된 이벤트 핸들러(4722)로 보낼 수 있다.

[0548] 트리거된 이벤트 핸들러(4722)는 O/S 하위 트래핑 에이전트(4720)로부터 트래핑된 시도와 관련된 트리거된 이벤트를 수신하도록 구성될 수 있다. 트리거된 이벤트 핸들러(4722)는 트리거된 이벤트와 관련해 취할 적합한 액션을 결정하기 위해, 보안 규칙들(4708)과 함께 트리거된 이벤트와 관련된 정황 정보를 이용할 수 있다. 일부 실시예들에서, 트리거된 이벤트 핸들러(4722)는 트리거된 이벤트와 관련된 정황 정보를 식별하기 위해 O/S 내 보안 에이전트(4719)와 협력할 수 있다. 정황 정보는 트래핑된 시도의 요청 개체, 트래핑된 시도와 관련된 특정 객체(4706) 및/또는 특정 객체(4706)와 관련해 요청된 액세스의 타입을 포함할 수 있다. 보안 규칙들(4708)은 예컨대 보안 애플리케이션과 관련된 프로세스 객체가 보안 애플리케이션 자체에 의해서만 삭제될 수 있다고 특정할 수 있다. 다른 예로서 보안 규칙들(4708)은 운영체제(4713)로부터 새로운 객체들(4706)을 생성하려는 시도들을 허가할 수 있고, O/S 하위 트래핑 에이전트(4720)에게 새로 생성된 객체들(4706)을 액세스하려는 추후 시도들을 트래핑하라고 요청할 수 있다.

[0549] O/S 하위 트래핑 에이전트(4720)는 운영체제(4713)의 동향을 나타내는 동향 상태 맵(4732)를 생성하기 위해 객체 자원들(4734)로의 액세스를 감시하도록 구성될 수 있다. 예를 들어 O/S 하위 트래핑 에이전트(4720)는 객체 자원들(4734)를 액세스하려는 시도들을 트래핑하고 각각의 트래핑된 동작을 나타내기 위해 동향 상태 맵(4732)을 업데이트함으로써, 동향 상태 맵(4732)을 생성할 수 있다. 동향 상태 맵(4732)은 알려지지 않은 제로 데이(zero-day) 악성 소프트웨어에 의한 공격들을 예방적으로 검출하여 금지하기 위해 동향 분석 시스템(4716)을 구현하는 데 사용될 수 있다. 동향 상태 맵(4732) 및 동향 분석 시스템(4716)의 실시예들에 대한 설명은 이하의 도 48의 동향 상태 맵(4802) 및 동향 분석 시스템(4804)에 대한 논의에서 찾아 볼 수 있다.

[0550] 도 48은 운영체제의 객체들에 대한 액세스를 보호하는 시스템이나 방법과 함께 사용할 동향 상태 맵(4802)의 실시예이다. 예를 들어 동향 상태 맵(4802)은 도 47의 동향 상태 맵(4732)으로서 구현될 수 있으며, 도 47의 동향 분석 시스템(4716), O/S 하위 트래핑 에이전트(4720) 및/또는 트리거된 이벤트 핸들러(4722)에 의해 생성 및/또는 사용될 수 있다. 객체지향 환경에서, 운영체제 및 파일들, 애플리케이션들, 프로세서들, 드라이버들 및/또는 장치들을 포함해 운영체제의 자원들 전부가 객체들로서 구현될 수 있다. 동향 상태 맵(4802)은 운영체제의 객체들 간 동작들 및/또는 상호동작들에 기반하여 객체지향 운영체제의 동향에 대한 표현을 제공할 수 있다. 동향 상태 맵(4802)은 전형적으로 악성 소프트웨어와 관련된 객체 상호동작들의 패턴들을 식별하는 데 사용될 수 있다.

[0551] 동향 상태 맵(4802)은 그래프 및/또는 맵을 포함하는 어떤 적절한 데이터 구조를 이용해 구현될 수 있다. 그래프를 이용하는 일 실시예에서, 각각의 노드는 운영체제의 어느 객체를 나타낼 수 있으며, 각각의 노드 사이의 에지들은 객체들 간 동작들 및/또는 상호동작들을 나타낼 수 있다. 예를 들어, 운영체제에 의한 프로세스의 실행은 운영체제 객체를 나타낼 노드 및 프로세스 객체를 나타낼 노드를 이용하여 동향 상태 맵에 의해 나타내질 수 있다. 동향 상태 맵은 프로세스가 운영체제에 의해 실행되었음을 나타내는 운영체제로부터 프로세스 객체까지의 에지를 포함할 수 있다. 그런 다음 그 프로세스가 파일을 오픈하면, 동향 상태 맵은 특정 파일 객체를 나타내는 노드를 포함하도록 업데이트될 수 있고 그 특정 파일이 해당 프로세스에 의해 오픈되었다는 것을 나타내는 프로세스 객체로부터 파일 객체로의 에지를 포함할 수 있다. 동향 상태 맵은 객체들 사이에서 수행되는 각각의 동작에 대해 이러한 방식으로 계속해서 업데이트될 수 있다. 어떤 실시예들에서, 동향 상태 맵은 전체 운영체제의 동향을 나타내도록 구현되거나, 운영체제 상에서 실행되는 특정 애플리케이션, 드라이버 및/또는 프로

세스와 같이 운영체제의 특정 구성요소의 동향만을 나타내도록 구현될 수 있다.

[0552] 동향 상태 맵(4802)은 악성 소프트웨어에 감염된 운영체제와 관련된 동향 상태 맵의 실시예이다. 동향 상태 맵(4802)는 운영체제(4806)의 객체들, 객체 관리자(4816), 보안 애플리케이션(4808) 및 악성 소프트웨어(4810)를 나타내는 노드들을 포함한다. 운영체제(4806)로부터 보안 애플리케이션(4808)까지의 에지는 운영체제(4806)에 의한 보안 애플리케이션(4808)의 실행을 나타내며, 운영체제(4806)로부터 악성 소프트웨어(4810)까지의 에지는 운영체제(4806)에 의한 악성 소프트웨어(4810)의 실행을 나타낸다. 운영체제(4806)로부터 객체 관리자(4816)까지의 에지는 운영체제(4806)에 의한 객체 관리자(4816)의 생성을 나타낸다. 악성 소프트웨어(4810)는 악성 소프트웨어(4810)에 의해 수행된 악성 활동들을 나타나는 여러 에지들과 관련된다. 악성 소프트웨어(4806)로부터 운영체제 파일들(4814)까지의 두 에지들은 악성 소프트웨어가 운영체제 파일들(4814)를 오픈하는 것과 운영체제 파일들(4814)로 기입하는 것을 나타낸다. 일례로서, 운영체제 파일들(4814)은 운영체제(4806)가 초기화될 때 실행될 수 있는 애플리케이션들을 특정하는 데 사용될 수 있으며, 악성 소프트웨어(4810)는 스스로를 이러한 애플리케이션들 중 하나로서 포함하도록 그 파일들에 기입할 수 있다. 악성 소프트웨어(4810)로부터 보안 애플리케이션(4808)까지의 에지는 악성 소프트웨어(4810)가 보안 애플리케이션(4808)을 종료시키려고 시도하는 것을 나타낸다. 악성 소프트웨어(4810)로부터 시스템 호출 테이블(4812)까지의 에지는 악성 소프트웨어(4810)가 시스템 호출 테이블(4812)로 기입하는 것을 나타낸다. 악성 소프트웨어(4810)는 예컨대 특정 시스템 호출을 위한 엔트리를 변경하기 위해 시스템 호출 테이블(4812)에 기입을 할 수 있다. 이러한 방식으로 시스템 호출이 수행될 때마다, 악성 소프트웨어(4810)의 악성 코드가 의도한 시스템 호출 대신에 실행될 수 있다. 악성 소프트웨어(4810)로부터 객체 관리자(4816)까지의 에지는 악성 소프트웨어가 객체 관리자(4810)의 특정 함수를 호출하려고 시도하는 것을 나타낸다. 예를 들어 악성 소프트웨어(4810)는 객체 관리자(4816)의 객체 삭제 함수를 호출함으로써 운영체제(4716)의 객체를 삭제하고자 시도할 수 있다. 동향 상태 맵(4802)은 동향 상태 맵의 가능한 하나의 실시예만을 나타낸다. 동향 상태 맵(4802)은 운영체제의 객체들의 동작들 및/또는 상호동작들을 나타내는 데 적합한 어떤 방식으로 구현될 수 있다.

[0553] 도 47로 돌아가면, 일부 실시예들에서 동향 상태 맵(4732)이 O/S 하위 트래핑 에이전트(4720)에 의해 생성될 수 있다. 다른 실시예들에서 동향 상태 맵(4732)은 이전에 생성되었을 수 있으며, 알려지지 않은 제로 데이 악성 소프트웨어에 의한 공격들을 예방적으로 검출하여 방지하기 위해 사용될 수 있다.

[0554] 동향 상태 맵(4732)은 운영체제(4713)의 객체들(4706) 사이의 상호동작들 및/또는 동작들을 감시함으로써 생성될 수 있다. 예를 들어 O/S 하위 트래핑 에이전트(4720)는 객체 자원들(4734)을 액세스하려는 시도들을 트래핑할 수 있고 각각의 트래핑된 동작을 반영하기 위해 동향 상태 맵(4732)을 업데이트할 수 있다. 어떤 실시예들에서 동향 상태 맵(4732)은 악성 소프트웨어에 감염된 운영체제(4713)를 이용하여 생성될 수 있다. 다른 실시예들에서 동향 상태 맵(4732)은 악성 소프트웨어 염려가 없는 운영체제(4713)를 이용하여 생성될 수 있다. 일부 실시예들에서, 동향 상태 맵(4732)이 생성된 후, 악성 소프트웨어와 관련된 동향의 패턴들을 격리하고/거나 안전한 동향의 패턴들을 격리하기 위해 그것이 분석될 수 있다. 그러한 실시예들에서 동향 상태 맵(4732)은 그 격리된 동향만을 나타내기 위해 업데이트될 수 있고, 혹은 새로운 동향 상태 맵이 그 격리된 동향만을 나타내도록 생성될 수 있다. 이러한 방식으로, 동향 상태 맵(4732)은 악성 소프트웨어와 관련되었다고 알려진 객체 동향의 모델 및/또는 안전하다고 알려진 객체 동향의 모델을 제공할 수 있다. 예를 들어, 동향 상태 맵(4732)이 악성 소프트웨어에 감염된 운영체제(4713) 상에서 생성되는 경우, 동향 상태 맵(4732)은 그 악성 동향을 격리하기 위해 분석될 수 있다. 악성 소프트웨어에 의해 전형적으로 수행되는 악성 동향은 다른 무엇보다, 코어 운영체제 파일들의 변경, 시스템 호출 테이블 액세스 및/또는 보안 애플리케이션들과 관련된 프로세스들 죽이기를 포함한다. 악성 소프트웨어에 감염된 운영체제(4713)의 동향 상태 맵(4732)을 분석함으로써, 악성 동향이 객체 레벨에서 분석될 수 있다. 객체 레벨에서 악성 동향을 분석하는 것은 특정한 악성 활동들이 악성 활동 수행을 담당하는 객체들 사이의 동작들의 패턴들과 상관되게 할 수 있다. 마찬가지로, 동향 상태 맵(4732)이 악성 소프트웨어 염려가 없는 운영체제(4713) 상에서 생성되면, 동향 상태 맵(4732)은 알려진 안전한 동향을 객체 동작들의 패턴들과 상관시키기 위해 객체 레벨에서 안전한 동향을 분석하는 데 사용될 수 있다.

[0555] 어떤 실시예들에서 동향 상태 맵(4732)은 알려지지 않은 제로 데이 악성 소프트웨어에 의한 공격들을 예방적으로 검출하여 방지하기 위해 사용될 수 있다. 그러한 실시예들에서 동향 상태 맵(4732)은 이전에 생성되었을 수 있으며, 전형적으로 악성 소프트웨어와 관련된 동향의 모델 및/또는 안전하다고 알려진 동향의 모델을 제공할 수 있다. 그러한 실시예들에서 동향 상태 맵(4732)은 전형적으로 악성 소프트웨어와 관련된 운영체제(4713)의 동향을 식별하기 위해 동향 분석 시스템(4716)에 의해 사용될 수 있다. 동향 분석 시스템(4716)은 O/S 하위 트래핑 에이전트(4720)에 의해 구현될 수 있거나, 트리거된 이벤트 핸들러(4722)에 의해 구현될 수 있고, 혹은 다

른 실시예들에서 동향 분석 시스템(4716)의 기능이 O/S 하위 트래핑 에이전트(472)에 의해 일부가, 그리고 트리거된 이벤트 핸들러(4722)에 의해 일부 구현될 수 있다. O/S 하위 트래핑 에이전트(4720)는 객체 자원들(4734)을 액세스하려는 시도들을 트래핑하도록 구성될 수 있으며, 동향 분석 시스템(4716)은 트래핑된 시도가 악성 소프트웨어를 나타내는지를 판단하는데 사용될 수 있다. 동향 분석 시스템(4716)은 시도된 액세스를 동향 상태 맵(4732)과 비교할 수 있다. 동향 상태 맵(4732)이 알려진 안전한 동향을 나타내는 실시예들에서, 동향 분석 시스템(4716)은 트래핑된 시도가 동향 상태 맵으로부터의 어떤 안전한 동향과 매칭하는지 여부를 판단할 수 있다. 매칭이 발견되면, 동향 분석 시스템은 트래핑된 시도가 안전하다고 판단하고 그 시도를 허용하기로 결정할 수 있다. 동향 상태 맵(4732)이 악성 소프트웨어와 관련된 동향을 나타내는 실시예들에서, 동향 분석 시스템(4716)은 트래핑된 시도가 동향 상태 맵으로부터의 어떤 악성 소프트웨어 관련 동향과 매칭하는지 여부를 판단할 수 있다. 매칭이 발견되면, 동향 분석 시스템(4716)은 트래핑된 시도가 안전하지 않다고 판단하고 그 시도를 거부하기로 결정할 수 있다.

[0556] 어떤 실시예들에서는 여러 동향 상태 맵들(4732)이 사용될 수 있다. 예를 들어 O/S 하위 트래핑 에이전트(4720)는 현재의 동향 상태 맵 및 모델 동향 상태 맵을 포함할 수 있다. 현재의 동향 상태 맵은 운영체제(4713)의 현재의 동향을 나타낼 수 있다. 모델 동향 상태 맵(4732)은 전형적으로 악성 소프트웨어와 관련된 모델 동향 및/또는 안전하다고 알려진 모델 동향을 제공하는 이전에 생성된 상태 맵일 수 있다. O/S 하위 트래핑 에이전트(4720)는 객체 자원들(4734)에 대해 시도된 액세스를 트래핑할 수 있으며 시도된 액세스를 반영하도록 현재의 동향 상태 맵을 업데이트할 수 있다. 그러면 동향 분석 시스템(4716)이 모델 동향 상태 맵을 현재의 동향 상태 맵과 비교할 수 있다. 이런 방식으로 동향 분석 시스템(4716)은 트래핑된 시도가 악성 소프트웨어와 관련된 것인지를 판단하기 위해, 현재의 동향 상태 맵으로부터의 이전 동향과 연관지어, 트래핑된 시도를 분석할 수 있다. 이것은 동향 분석 시스템(4716)이 트래핑된 시도를 보다 효과적으로 평가할 수 있게 한다.

[0557] 도 49는 운영체제의 객체들에 대한 액세스를 보호하기 위한 방법(4900)의 실시예이다. 단계 4905에서, O/S 하위 보안 에이전트, O/S 내 보안 에이전트, 트리거된 이벤트 핸들러 및 보호 서버의 식별 및 보안이 인증될 수 있다. 그러한 인증은 암호화 해싱 및/또는 비밀 키들을 이용하여 각각의 구성요소에 대한 메모리 내 이미지들을 찾아 검증하는 것을 포함하는 어떤 적절한 방법을 이용하여 수행될 수 있다. 단계 4905가 완료될 때까지 다른 단계들의 동작은 보류될 수 있다. 단계 4910에서 보안 규칙들이 얻어진다. 보안 규칙들은 O/S 하위 보안 에이전트, O/S 내 보안 에이전트 및/또는 트리거된 이벤트 핸들러에 의해 내부적으로 저장되고/거나, 예컨대 보호 서버 상에 원격으로 저장될 수 있다. 그러한 보안 규칙들은 단계들 4915-4945에서 결정을 내리는 데 사용될 수 있다.

[0558] 단계 4915에서, 운영체제의 객체들과 관련된 자원을 액세스하려는 시도가 인터셉트될 수 있다. 운영체제의 객체들과 연관된 자원들은 예컨대 객체 관리자, 객체 관리자 함수들 및/또는 서브 함수들, 객체 자체들 및/또는 객체들의 함수들을 포함할 수 있다. 인터셉트된 시도들은 객체 함수들, 객체 관리자 함수들 및/또는 객체 관리자 함수들의 서브 함수들을 저장하는 메모리 내 위치들에서 코드를 실행하고자 하는 시도들을 포함할 수 있다. 인테셉트된 시도들은 또한 객체들 및/또는 객체 관리자가 저장되는 메모리 내 위치들을 액세스하고자 하는 시도들을 포함할 수 있다. 어떤 실시예들에서, 시도들은 가상 메모리 어드레스에서 물리적 메모리 어드레스로의 변환 전에 가상 메모리 레벨에서 인터셉트될 수 있다. 다른 실시예들에서 시도들은 물리적 어드레스 레벨에서 인터셉트될 수 있다. 어떤 실시예들에서 메모리 맵이 보호될 객체 자원들의 메모리 내 위치들을 특정하는 데 사용될 수 있다.

[0559] 단계 4920에서 인터셉트된 시도의 요청 개체가 식별된다. 예를 들어 인터셉트된 시도가 애플리케이션, 드라이버, O/S 내 보안 에이전트, 운영체제 및/또는 다른 소프트웨어 개체로부터 나올 수 있다. 어떤 실시예들에서 요청 개체는 운영체제 상에서 실행되는 개체들의 어드레스들을 포함하는 메모리 맵을 조회함으로써 식별될 수 있다.

[0560] 단계 4925에서, 운영체제의 현재의 동향 상태 맵이 업데이트될 수 있다. 현재의 동향 상태 맵은 운영체제의 객체들 간 상호동작들 및/또는 동작들에 기반하여 운영체제의 동향을 표시하는 데이터 구조일 수 있다. 객체 자원을 액세스하려는 인터셉트된 시도 각각에 대해, 현재의 동향 상태 맵은 그 인터셉트된 시도에 대응하는 동작을 반영하도록 업데이트될 수 있다. 단계 4930에서 현재의 동향 상태 맵이 모델 동향 상태 맵과 비교될 수 있다. 모델 동향 상태 맵은 통상적으로 악성 소프트웨어와 관련된 동향 및/또는 통상적으로 안전하다고 알려진 동향을 나타낼 수 있다. 그러한 비교는 악성 소프트웨어와 관련된 객체 동작들의 패턴들에 대한 식별을 가능하게 할 수 있거나, 안전하다고 알려진 객체 동작들의 패턴들에 대한 식별을 가능하게 할 수 있다.

[0561] 단계 4935에서 인터셉트된 시도가 허가되는지 여부가 판단된다. 모델 상태 맵에 대한 현재의 동향 상태 맵의 단계 4930으로부터의 비교에 기반하여 악성 소프트웨어가 식별되었다면, 그 시도는 허가되지 않을 것이다. 단계 4930의 비교로부터 어떤 악성 소프트웨어도 식별되지 않았다면, 특정 시도가 허가되는지 여부를 판단하기 위해, 인터셉트된 시도와 관련된 정황 정보와 함께 보안 규칙들이 사용될 수 있다. 정황 정보는 인터셉트된 시도의 요청 개체, 인터셉트된 시도와 관련된 특정 객체 및/또는 요청된 액세스의 타입을 포함할 수 있다. 예를 들어 보안 규칙은 보안 애플리케이션과 관련된 프로세스 객체가 보안 애플리케이션 자체에 의해서만 삭제될 수 있다고 특정할 수 있다. 그러한 시도가 허가된다고 판단되면, 단계 4940에서 액세스가 허용될 수 있다. 해당 시도가 허가되지 않으면, 단계 4945에서 액세스가 거부될 수 있다.

[0562] 도 49로부터의 방법의 단계들은 필요 시, 지속적으로나 주기적으로나 요청에 따르거나 어떤 이벤트의 트리거 시, 전자 장치를 보호하기 위해 반복될 수 있다.

[0563] 도 50은 전자 장치(5001) 상의 드라이버들 간 통신을 보호하기 위한 시스템(5000)의 실시예이다. 시스템(5000)은 전자 장치(5001)의 운영체제(5012)와 같은 운영체제의 드라이버들 간의 통신을 인터셉트하거나 전복시키려는 악의적 시도들을 검출하기 위해 전자 장치(5001) 상에서 동작하도록 구성된 O/S 하위 보안 에이전트(5020)를 포함할 수 있다. 또한 O/S 하위 보안 에이전트(5020)는 예컨대 시도된 어떤 드라이버 간 통신을 트래핑해야 하는지, 드라이버 간 통신 기능부들에 대해 시도된 어떤 액세스들을 트래핑해야 하는지, 또는 그 시도들이 그 시도 및 관련된 개체들에 기반하여 허가되는지 여부를 판단하기 위해 하나 이상의 보안 규칙들(5008)을 이용하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(5020)는 트래핑된 시도에 대해 허용하거나 거부하거나 그에 대해 다른 교정 액션을 수행하도록 구성될 수 있다.

[0564] 전자 장치(5001)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701), 도 9의 전자 장치(901), 도 12의 전자 장치(1201) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(5001)는 메모리(5003)와 같은 메모리에 연결된 하나 이상의 프로세서들(5002)을 포함할 수 있다. 프로세서(5002)는 도 2의 프로세서(208), 도 4의 프로세서(408), 도 7의 프로세서(702), 도 9의 프로세서(1202), 도 12의 프로세서(1202) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(5003)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703), 도 9의 메모리(903), 도 12의 물리적 메모리(1203) 또는 가상 메모리(1204) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(5001)는 하나 이상의 보안 규칙들(5021)에 연결된 O/S 내 보안 에이전트(5019)를 포함할 수 있는 운영체제(5012)를 포함할 수 있다. 운영체제(5012)는 도 1의 운영체제(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 9의 운영체제(913), 도 12의 운영체제(1213) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(5019)는 도 1의 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418) 및/또는 도 7의 O/S 내 보안 에이전트(719), 도 9의 O/S 내 보안 에이전트(919), 도 12의 O/S 내 보안 에이전트(1219) 또는 이들의 어떤 적절한 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0565] O/S 하위 보안 에이전트(5020)는 도 1의 O/S 하위 트래핑 에이전트(104)나 트리거된 이벤트 핸들러(108), 도 2의 SVMM(216) 또는 SVMM 보안 에이전트(217), 도 4의 펌웨어 보안 에이전트들(440, 442), O/S 하위 에이전트(450), 또는 PC 펌웨어 보안 에이전트(444), 도 5의 펌웨어 보안 에이전트(516), 또는 도 7의 마이크로코드 보안 에이전트(708)나 O/S 하위 에이전트(712), 도 9의 O/S 하위 트래핑 에이전트(920) 또는 트리거된 이벤트 핸들러(922), 도 12의 O/S 하위 보안 에이전트(1220) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0566] 보안 규칙들(5008)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(222), 도 4의 보안 규칙들(434, 436, 438), 도 5의 보안 규칙들(518), 도 7의 보안 규칙들(707, 723), 도 9의 보안 규칙들(908), 도 12의 보안 규칙들(1208) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보안 규칙들(5021)은 도 2의 보안 규칙들(220), 도 4의 보안 규칙들(420), 도 7의 보안 규칙들(721), 도 9의 보안 규칙들(921), 도 12의 보안 규칙들(1221) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0567] 전자 장치(5001)는 드라이버(5029)와 같은 드라이버 또는 드라이버(5029)의 드라이버간 통신 기능부와 통신하거나 그를 사용하기 위해 전자 장치(5001)의 자원을 액세스하고자 시도할 수 있는 하나 이상의 애플리케이션들,

드라이버들 또는 다른 개체들(예컨대 애플리케이션(5026)이나 드라이버2(5028))을 포함할 수 있다. 애플리케이션(5026) 또는 드라이버2(5028)는 어떤 프로세스, 애플리케이션, 프로그램 또는 드라이버를 포함할 수 있다. 애플리케이션(5026) 또는 드라이버2(5028)는 드라이버(5029)를 호출하고자 시도할 수 있으며, 그에 따라 드라이버(5029) 및 다른 드라이버 사이의 드라이버간 통신을 일으키는 호출들의 시퀀스를 개시할 수 있다. 애플리케이션(5026) 또는 드라이버2(5028)는 직접적으로나 다른 루틴들에 대한 호출들을 통해, 드라이버간 통신의 기능부들을 액세스하도록 시도할 수 있다. 애플리케이션(5026)이나 드라이버2(5028)는 드라이버 서브 함수(5030)를 호출하고자 시도할 수 있다. 드라이버간 통신 기능부들에 대한 그러한 모든 호출들 및 액세스는 프로세서(5002)와 연계하여 메모리(5003)에 대한 동작들에 대해 시도되는 읽기, 쓰기 또는 실행을 통해 시도될 수 있다. 운영체제 하위 보안 에이전트(5020)는 드라이버간 통신의 기능부들에 대해 시도되는 그러한 호출들이나 액세스를 인터셉트하고, 그 시도가 악성 소프트웨어를 나타내는지 여부를 판단하기 위해 O/S 내 보안 에이전트(5019)로부터의 정황 정보 및/또는 보안 규칙들(5008)을 조회하며, 어떤 알맞은 교정 액션을 취할 수 있도록 구성될 수 있다. 운영체제 하위 보안 에이전트(5020)는 메모리(5003)에 대한 액세스 및/또는 프로세서(5002)의 사용을 트래핑하는 것을 통해 그러한 인터셉트를 행하도록 구성될 수 있다. 운영체제 하위 보안 에이전트(5020)는 보안 규칙들(5008)을 액세스하고 메모리(5003) 액세스 및/또는 프로세서 사용(5002)에 대한 어떤 액세스 시도가 트래핑될 것인지를 판단하도록 구성될 수 있다. 운영체제 하위 보안 에이전트(5020)는 트래핑되어야 하는 액션들에 대응하는 제어 구조의 플래그들을 세팅하도록 구성될 수 있다.

[0568] 일 실시예에서, 애플리케이션(5026)이나 드라이버2(5028)는 메모리 페이지를 통한 드라이버간 통신과 연관된 메모리(5003)의 부분들을 액세스하도록 시도할 수 있으며, 이때 메모리(5003)는 운영체제(5012)에 의해 가상화되어 있다. 그러한 실시예에서 O/S 하위 보안 에이전트(5020)는 메모리 페이지 기반으로 메모리(5003)의 액세스나 실행 시도를 트래핑하도록 구성될 수 있다. 다른 실시예에서 애플리케이션(5026)이나 드라이버2(5028)는 드라이버간 통신과 관련된 메모리(5003)의 물리적 부분들을 액세스하도록 시도할 수 있다. 그러한 실시예에서 O/S 하위 보안 에이전트(5020)는 메모리 어드레스 기반으로 메모리(5003)의 액세스나 실행 시도를 트래핑하도록 구성될 수 있다.

[0569] 전자 장치(5001)의 운영체제(5012) 및 드라이버들은 드라이버간 통신을 위한 기능부들을 제공할 수 있다. 예를 들어 NTFS.SYS(5031)와 같은 드라이버가 발송 루틴 포인터들(5032), 엑스포트(export) 어드레스 테이블(5034), 임포트(import) 어드레스 테이블들(5036) 또는 고속 I/O 루틴 포인터들(5038)을 포함할 수 있다. 발송 루틴 포인터들(5032)은 CodeSectionl, CodeSection2 또는 Malware Code Section(5046)과 같은 코드 섹션들 안에 내장된 함수들과 같은 드라이버의 함수들에 대한 포인터들을 포함할 수 있다. 엑스포트 어드레스 테이블(5034)는 함수들과 관련된 코드 섹션들에 대한 포인터들을 포함할 수 있으며, 상기 포인터들은 다른 드라이버들에 의해 드라이버의 함수들을 호출하는데 사용될 수 있다. 임포트 어드레스 테이블들(5036)은 호출할 드라이버에 대해, 하나 이상의 다른 드라이버들의 함수들에 대한 포인터들의 리스트들을 포함할 수 있다. 그러한 임포트 어드레스 테이블들(5036)은 다른 드라이버의 엑스포트 어드레스 테이블을 들여온(임포트) 결과가 될 수 있다. 그러한 드라이버간 통신의 기능부들이 드라이버의 데이터에 대한 액세스를 위해 제공될 수 있다. 그러한 드라이버의 데이터는 드라이버의 특성과 메이커에 고유할 수 있다. 예를 들어 NTFS.SYS(5031)는 모든 오픈 파일들에 대한 포인터들을 포함할 수 있는 오픈 파일 처리 리스트(5040) 또는 운영체제(5012) 안에 장착된 각각의 스토리지 볼륨에 대한 포인터들을 포함할 수 있는 장착된 볼륨들의 리스트(5042)에 대한 데이터 섹션들이나 구조들을 포함할 수 있다. 드라이버간 통신 기능부들은 악성 소프트웨어에 의한 공격을 받기 쉬울 수 있으며, 그에 따라 O/S 하위 보안 에이전트(5020)가 그러한 기능부들, 그들 아래에 내재한 메커니즘들 또는 드라이버 데이터와 같이 그러한 기능부들의 대상의 사용에 대해 시도되는 액세스를 트래핑할 수 있다.

[0570] 도 51은 드라이버 간 통신의 예시도이다. 애플리케이션(5102)은 네트워크 인터페이스("NIC") 카드(5116)에 대한 네트워크 요청이나 디스크(5128)에 대한 파일 요청과 같은 요청을 하도록 시도할 수 있다. 그러한 장치들에 도달하기 위해, 해당 요청은 운영체제(5104)를 통해 다뤄질 수 있다. 운영체제(5104)의 입/출력 요청들은 운영체제 입/출력 관리자(5106)에 의해 다뤄질 수 있다.

[0571] 운영체제 입/출력 관리자(5106)가 일련의 드라이버들에서 이용 가능한 함수들을 호출함으로써 네트워크 요청을 보낼 수 있다. 운영체제 입/출력 관리자(5106)는 SOCKET DRIVER AFS.SYS(5108)를 호출할 수 있고, 그것은 타입 디맨드 인터페이스(Type Demand Interface("TDI")) 프로토콜 드라이버(5110)를 호출할 수 있고, 그것이 네트워크 드라이버 인터페이스 사양(Network Driver Interface Specification("NDIS")) 드라이버(5112)를 호출할 것이며, 그것이 이어서 NIC 카드(5116)의 하드웨어에 고유할 수 있는 NDIS.SYS를 호출할 수 있다. 응답은 애플리케이션(5102)에 대한 동일한 연쇄 드라이버들 따를 수 있다.

[0572] 마찬가지로, 운영체제 입/출력 관리자(5106)가 일련의 드라이버들에서 이용 가능한 함수들을 호출함으로써 파일 요청을 보낼 수 있다. 운영체제 입/출력 관리자(5106)는 첨부된 파일 시스템 필터 드라이버(5118)를 가진 파일 시스템 드라이버(5120)를 호출할 수 있고, 그것은 다시 디스크 필터 드라이버(5122)가 첨부된 디스크 드라이버(5124)를 호출할 것이며, 그것이 다시 디스크(5128)로의 물리적 입/출력을 다룰 수 있는 DISK.SYS와 같은 디스크(5128)에 고유한 디스크 드라이버를 호출할 수 있다. 응답은 애플리케이션(5102)에 대한 동일한 연쇄 드라이버들을 따를 수 있다.

[0573] 도 51 안에서 드라이버들 및 커널 모듈들의 호출들 각각은 드라이버들 자체에 의해 특정된 시스템 전역에 걸치는 호출들을 이용하여 수행될 수 있다. 악성 소프트웨어는 도 51에 도시된 요소들 각각 간의 호출을 후킹, 전복, 하이재킹, 스푸핑 또는 공격할 수 있다. 따라서 이들은 통신이 일어날 때와 그러한 통신을 가능하게 하는 메커니즘들을 보호하는 두 경우 모두에서, 도 50의 O/S 하위 보안 에이전트(5020)가 보호하도록 구성될 수 있는 통신들의 예를 나타낸 것이다.

[0574] 도 52는 도 50의 O/S 하위 보안 에이전트(5020)와 같은 O/S 하위 보안 에이전트가 드라이버간 통신과 연계하여 보호하도록 구성될 수 있는 전자 장치의 일부에 대한 추가 예시도이다. 드라이버간 통신은 애플리케이션(5202)과 같이 사용자 모드에서 발생한 요청이나 드라이버2(5204)와 같이 다른 드라이버로부터의 요청에 의해 일어날 수 있다. 애플리케이션(5202)은 전자 장치의 어느 부분에 대해 장치 요청 명령(5208)을 행하도록 구성될 수 있다. 장치 요청 명령(5208)은 시스템 서비스 발송 테이블("SSTD")(5210)에 의해 장치 함수(5212)로 변환될 수 있다. 장치 함수(5212)는 해당 장치와 연관된 드라이버로 애플리케이션(5202)에 의해 이루어진 요청에 대응하는 I/O 요청 패킷("IRP")을 전송하도록 구성될 수 있다. 도 52에서 그러한 드라이버는 디렉토리 제어를 위해 I/O 요청을 처리할 수 있는 드라이버1(5206)일 수 있다. 드라이버간 통신은 드라이버2(5204)와 같은 다른 커널 모드 드라이버로부터 발생하는 요청에 의해 일어날 수 있다.

[0575] 도 50의 O/S 하위 보안 에이전트(5020)는 어떤 적절한 방식으로 드라이버간 통신을 공격할 수 있는 악성 소프트웨어에 대해 저자 기기(5001)를 보호하도록 구성될 수 있다. 예를 들어 악성 소프트웨어가 IoCallDriver 명령(5220)과 같이 IRP 명령들을 송신하거나 수신하기 위한 함수들을 후킹할 수 있다. 그러한 악성 소프트웨어는 해당 함수가 지정된 타깃(가령, 드라이버가 실행할 수 있는 시스템 정의 함수들을 포함하는 발송 루틴들(5209))에 대해 의도된 IRP를 잘못 향하게 할 수 있다. 대신 악성 소프트웨어 후크(hook)(5224)가 IoCallDriver(5220) 위에 인스톨되어 해당 요청을 인터셉트할 수 있다. 악성 소프트웨어 후크(5224) 뒤에 있는 악성 소프트웨어가 그 요청을 원래의 목적지인 IRP MJ READ(5226)로 보내기 전이나 악성 코드(5228)를 대신 실행하기 전에에 그 요청을 검사할 수 있다. 그에 따라 O/S 하위 보안 에이전트는 IoCallDriver(5220)에 대한 명령어들을 포함하는 메모리를 보호하거나, 그 드라이버나 재호출(callback) 루틴으로의 악성 호출들에 대해 IoCallDriver(5220)의 실행을 검사하도록 구성될 수 있다. O/S 하위 보안 에이전트(5020)는 내보내는 함수들을 트래핑하거나 루틴들을 발송하도록 구성될 수 있다. O/S 하위 보안 에이전트(5020)는 운영체제 구조들 안에서 그러한 함수들이나 루틴들에 대한 포인터들뿐 아니라 그 함수들 자체의 메모리 위치들의 실행을 트래핑하도록 구성될 수 있다. 예를 들어 악성 소프트웨어는 엑스포트 어드레스 테이블(이하에 더 상세히 기술되는 "EAT") 안의 어느 포인터에 대한 메모리 위치 내 값을 변경하려고 시도하거나 함수의 코드 섹션의 콘텐츠 자체를 변경(가령, 악성 코드에 "JMP" 삽입)하고자 시도할 수 있다. 그러한 포인터나 함수에 대한 액세스를 트래핑함으로써, 해당 함수의 호출자를 판단하기 위해, 트래핑된 시도가 디코딩될 수 있다.

[0576] 다른 예에서 드라이버1(5206)이 드라이버들과 같은 다른 개체들에 의해 호출될 수 있는 그 자신에 고유한 드라이버1(5206)에 의해 제공되는 함수들의 EAT(5211)를 관리할 수 있다. EAT(5211)는 지정된 함수를 실행하기 위한 코드 섹션들의 위치를 가리키는 함수 포인터들의 리스트나 어레이를 포함할 수 있다. 악성 소프트웨어는 EAT의 엔트리들이 더 이상 올바른 코드 섹션들을 가리키지 못하도록 그 포인터들의 값들을 바꿀 수 있다. 그 포인터들은 대신 코드의 잠정적 악성 섹션들을 가리키도록 되어, EAT(5211) 안의 해당 포인터를 참조해 다른 드라이버에 의해 해당 드라이버 함수가 호출될 때 악성 코드가 실행되도록 할 수 있다. 예를 들어 EAT(5211)는 보통, Driver1Fn1 코드 섹션(5214)을 가리킬 수 있는 함수 Driver1Fn1 및 Driver1Fn2 코드 섹션을 가리킬 수 있는 함수 Driver1Fn2에 대한 포인터들을 포함할 수 있다. 그러나 Driver1Fn2가 이제 악성 소프트웨어 코드 섹션(5218)을 가리키게 하도록 악성 소프트웨어가 두 번째 포인터를 변경하였을 수 있다. 그에 따라 O/S 하위 보안 에이전트는 쓰기 요청들을 인터셉트하고 그 쓰기가 검증되지 않은 경우 EAT(5211)로 기입하려는 그러한 트래핑된 시도들을 거부하여 EAT(5211)가 상주하는 메모리 공간을 보호하도록 구성될 수 있다. 상기 검증은 예컨대 드라이버1(5206) 자체가 자신의 함수들을 업데이트하는 것을 포함할 수 있다. O/S 하위 보안 에이전트는 또한 EAT(5211)를 기입, 변경 또는 세팅하기 위한 어떤 시도된 함수의 실행을 트래핑하도록 구성될 수 있다. O/S 하

위 보안 에이전트는 해당 시도의 호출자가 그러한 함수를 수행하는 것이 허가된다는 것과, 그 호출자가 EAT(5211)을 변경하기 위해 문서화되지 않은 서브루틴을 호출하는 등과 같이 표준 절차를 전복시키지 않았다는 것을 검증할 수 있다.

[0577] 또 다른 예에서 드라이버2(5204)와 같은 다른 드라이버가 드라이버1(5206)의 EAT(5211)를 받아들여서(임포트) 그 테이블을 드라이버1(5206)의 함수들과 관련된 자체 임포트 어드레스 테이블("IAT")(5222)로서 호스트할 수 있다. IAT(5222)을 사용하여, 드라이버2(5204)는 드라이버1(5206)의 함수들을 호출하도록 구성될 수 있다. IAT(5222)는 운영체제 로더에 의해 채워질 수 있다. 악성 소프트웨어가 여러 방식으로 IAT를 감염시킬 수 있다. Driver2Fn2와 같은 함수가 현재 악성 소프트웨어 코드 섹션(5218)과 같이 악성인 코드 섹션을 가리키도록 IAT(5222) 안의 값들이 변경될 수 있다. 그에 따라 O/S 하위 보안 에이전트는 쓰기 요청들을 인터셉트하고 그 쓰기가 검증되지 않은 경우 IAT(5222)로 기입하려는 그러한 트래핑된 시도들을 거부하여 IAT(5222)가 상주하는 메모리 공간을 보호하도록 구성될 수 있다. 상기 검증은 예컨대 운영체제 로더가 IAT(5222)를 로딩하는 것을 포함할 수 있다. O/S 하위 보안 에이전트는 또한 IAT(5222)를 기입, 변경 또는 세팅하기 위한 어떤 시도된 함수의 실행을 트래핑하도록 구성될 수 있다. O/S 하위 보안 에이전트는 해당 시도의 호출자가 그러한 함수를 수행하는 것이 허가된다는 것과, 그 호출자가 IAT(5222)을 변경하기 위해 문서화되지 않은 서브루틴을 호출하는 등과 같이 표준 절차를 전복시키지 않았다는 것을 검증할 수 있다.

[0578] 또 다른 예에서 Driver1Fn1과 같은 드라이버 함수가 호출되었으면, Driver1Fn1 코드 섹션(5212)에서와 같은 코드가 실행 시작될 수 있다. 악성 소프트웨어는 그러한 코드 섹션의 일부를 재기입하거나 주입하여, 해당 루틴이 호출될 때 악성 코드가 실행되도록 할 수 있다. 그에 따라 O/S 하위 보안 에이전트는 쓰기 요청들을 트래핑하고 그 쓰기가 검증되지 않은 경우 드라이버의 코드 섹션에 기입하려는 그러한 트래핑된 시도들을 거부하여 드라이버 함수들의 코드가 상주하는 메모리 공간을 보호하도록 구성될 수 있다. 상기 검증은 예컨대 그 쓰기가 패치를 사용하여 스스로를 업데이트하는 드라이버로부터 발생한 것이라 판단하는 것을 포함할 수 있다. O/S 하위 보안 에이전트는 또한 드라이버 함수들의 코드 섹션들을 기입, 변경 또는 세팅하기 위한 어떤 시도된 함수의 실행을 트래핑하도록 구성될 수 있다. O/S 하위 보안 에이전트는 해당 시도의 호출자가 그러한 함수를 수행하는 것이 허가된다는 것과, 그 호출자가 드라이버 함수들의 코드 섹션들을 변경하기 위해 문서화되지 않은 서브루틴을 호출하는 등과 같이 표준 절차를 전복시키지 않았다는 것을 검증할 수 있다.

[0579] 또 다른 예에서, 드라이버 함수들의 코드가 허가된 임포트 또는 엑스포트 어드레스 테이블을 액세스함으로써가 아닌 악성 소프트웨어에 의해 직접 호출될 수 있다. 그에 따라 O/S 하위 보안 에이전트는 악성 코드(5228)에 의한 직접 실행으로부터 Driver1Fn2 코드 섹션(5216)과 같은 드라이버의 함수들의 실행을 보호하도록 구성될 수 있다. 그러한 O/S 하위 보안 에이전트는 함수의 실행을 트래핑할 수 있다. O/S 하위 보안 에이전트는 정황 정보로부터, 각자의 IAT들(5222)에서 어떤 드라이버들이 운영체제에 의해 그렇게 업데이트되었는지를 판단함으로써 어떤 드라이버들이 드라이버1(5206)에 대한 함수 실행에 대해 허가를 받았는지를 판단할 수 있다. O/S 하위 보안 에이전트는 호출이 어디에서 이루어졌는지를 판단할 수 있고, 그 위치가 알려진 허가된 드라이버들에 해당하지 않으면 그 시도가 거부될 수 있다. 일 실시예에서 도 50의 O/S 내 보안 에이전트(5019)가 정황 정보를 제공하기 위해 운영체제에 드라이버나 드라이버 필터로서 등록할 수 있다. 예를 들어 루트킷 드라이버는 파일 I/O를 위한 NTFS.SYS 호출을 막을 수 있다. O/S 내 보안 에이전트(5019)는 NTFS.SYS로 또는 그로부터 이루어진 모든 호출들을 보기 위해 NTFS.SYS 상에 필터로서 등록할 수 있고, 그런 다음 혹시 있다면 어떤 함수 호출들이 루트킷에 의해 파일 I/O에 사용되었는지에 대해 O/S 하위 보안 에이전트들로 정보를 줄 수 있다.

[0580] 도 50으로 돌아가면, 동작 시 드라이버(5029) 및 드라이버2(5028)과 같은 드라이버들은 어떤 적절한 방법을 통해 통신할 수 있다. O/S 하위 보안 에이전트(5020)는 그러한 통신이나 그러한 통신을 가능하게 하는 메커니즘들에 대해 시도된 변경을 트래핑할 수 있다. 일 실시예에서 O/S 하위 보안 에이전트(5020)는 도 52에 기술된 드라이버간 통신에 대한 악의적 간섭의 예들 중 어느 하나를 트래핑하고 평가할 수 있다.

[0581] 일례에서 그러한 통신은 IRP를 통해 보내지는 I/O 제어 코드를 포함할 수 있다. O/S 하위 보안 에이전트(5020)는 IRP를 통해 I/O 제어 코드를 보내기 위한 함수 호출에 해당하는 코드의 실행을 트래핑하고, 발송자가 허가되었는지 여부를 평가하며, 필요 시 어떤 교정 액션을 취할 수 있다.

[0582] 다른 예에서 그러한 통신은 코드 섹션 1과 같이 드라이버의 함수의 코드 섹션에 대한 호출을 포함할 수 있다. O/S 하위 보안 에이전트(5020)는 코드 섹션 1의 실행 시도를 트래핑할 수 있다. O/S 하위 보안 에이전트(5020)은 상기 시도된 실행이 함수를 액세스하는 합법적 수단을 이용하는 합법적 소스로부터 발생한 것인지 여부를 판단할 수 있다. O/S 하위 보안 에이전트(5020)은 호출자를 식별하고, 호출자가 알려져 있는지 여부와 판단된

호출자의 아이디에 기반하여 어떤 규칙들이 함수의 실행을 금지하는지 여부를 판단할 수 있다. 예를 들어 코드 섹션 2의 실행이 알려져 있고 디지털 인증서를 가지는 드라이버들로 국한될 수 있다. O/S 하위 보안 에이전트(5020)는 그러한 액세스를 발생했을 수 있는 드라이버2(5028)가 화이트리스트에 따라 안전하다고 알려져 있고 디지털 인증서를 가지는지 여부를 판단할 수 있다. 다른 예에서 O/S 하위 보안 에이전트(5020)는 호출이 드라이버(5029)를 통해 이루어졌는지 여부나 그 호출이 드라이버(5029) 액세스 (및 마찬가지로 이 명세서에서 이용되는 보안 조처들) 없이 문서화되지 않은 드라이버(5030)의 서브 함수를 통해 이루어졌는지를 판단할 수 있다. 관련 예가 예컨대 특정된 드라이버 함수 메커니즘들 중 어느 하나를 사용하지 않고 코드 섹션 1으로 바로 점프하거나 분기하도록 하는 애플리케이션(5026)에 의한 어떤 시도를 트래핑하는 것일 수 있다. 검색이나 그 서명을 통해 애플리케이션(5026)이 악성이라고 알려진 것이 아니라도, 그러한 동향은 매우 의심스럽고 악성 소프트웨어를 가리키므로, O/S 하위 보안 에이전트(5020)는 해당 액세스가 악성 소프트웨어를 가리킨다고 판단할 수 있다.

[0583] 또 다른 예에서, 재호출(callback) 루틴(5044)이 드라이버의 데이터 공간 안과 같이 드라이버 안에 등록될 수 있다. 재호출 루틴은 드라이버나 특정 드라이버 함수의 빠져나오기(exit)에 대해 실행하도록 트리거될 수 있다. 그러한 재호출 루틴(5044)는 악성일 수 있다. 이와 같이 O/S 하위 보안 에이전트(5020)는 메모리 안의 드라이버(5031)의 코드 섹션이나 데이터 섹션에 대해 시도되는 쓰기를 검출함으로써 어떤 재호출 루틴의 생성 시도를 트래핑할 수 있다. 시도한 기입자가 악성이라고 알려진 경우 해당 시도가 거부될 수 있다. 시도된 기입자가 알려져 있지 않으면, 해당 쓰기는 허용될 수 있지만 수행될 액션들이 악성인지 여부를 판단하기 위해 재호출 루틴의 후속 실행 자체가 트래핑될 수 있다. 예를 들어 어떤 로거가 복제 네트워크 패킷들이 악성 서버로 전송되는 재호출 루틴(5044)을 인스톨할 수 있다. 그 호출 루틴의 후속 동향이 관찰되고 악성 소프트웨어에 대한 추가 표시에 대해 평가될 수 있다.

[0584] 또 다른 예에서, 애플리케이션(5026)이 EAT(5034)로부터 어떤 어드레스를 읽고자 시도하, 그런 다음 해당 함수를 바로 실행할 수 있다. O/S 하위 보안 에이전트(5020)는 EAT(5034)에 대해 시도된 읽기를 트래핑하고, 판독자가 그러한 시도 및 코드 섹션 1과 같은 함수의 후속 실행을 하는 것이 허가되는지 여부를 판단할 수 있다. 그러한 시도는 악성 소프트웨어가 종속 드라이버로서 등록하고 그 자신의 임포트 어드레스 테이블을 통해 함수 포인터 리스트를 수신하는 것과 같이, 운영체제(5012)에 의해 제공되는 표준 방법들을 이용하지 않고 바로 EAT(5034)를 읽고자 시도했다는 것을 나타낼 수 있다.

[0585] 추가 예에서, 드라이버 2(5028)가 NTFS.SYS(5031)과 같은 드라이버의 데이터 섹션을 바로 조작하고자 시도할 수 있다. O/S 하위 보안 에이전트(5020)는 드라이버간 통신에 대한 악의적 공격을 막기 위해 드라이버의 데이터 섹션에 대해 시도되는 어떠한 조작들이라도 트래핑할 수 있다. 예를 들어 O/S 하위 보안 에이전트(5020)는 고속 I/O 라우팅 포인터들(5038)로 시도된 쓰기를 트래핑할 수 있고, 그 시도가 NTFS.SYS(5031) 자체나 운영체제(5012)로부터 발생했는지 여부를 평가할 수 있다. 그렇지 않으면 O/S 하위 보안 에이전트(5020)는 드라이버2와 같은 다른 드라이버로부터 발생한 것으로 판단되는 트래핑된 시도를 거부할 수 있다. 마찬가지로, 어떤 그러한 중요 데이터가 커널 운영체제(5012)에 의해 보유되는 경우, O/S 하위 보안 에이전트(5020)는 그 데이터를 포함하는 메모리에 대해 시도되는 액세스를 트래핑하도록 구성될 수 있다.

[0586] 또 다른 추가 예에서, 드라이버2(5028)는 임포트 어드레스 테이블(5036)에 대해 시도된 읽기를 통해 드라이버의 정보로부터 다른 제3자들에 대한 정보를 획득하고자 할 수 있다. O/S 하위 보안 에이전트(5020)는 임포트 어드레스 테이블(5036)에 대해 시도된 읽기를 트래핑할 수 있고, NTFS.SYS와 같은 드라이버 자체, 그 어드레스 테이블이 임포트되게 하였던 제3자 또는 운영체제(5012)로부터 일어난 것이 아닌 어떠한 시도들도 거부할 수 있다.

[0587] 또 다른 추가 예에서, 드라이버의 일부를 액세스하기 위한 함수 호출이 후킹되어, 악성 소프트웨어가 전자 장치(5001)의 다양한 부분들로의 액세스를 취할 수 있게 할 수 있다. O/S 하위 보안 에이전트(5020)는 악성 후크들을 시스템 함수들에 추가하고자 시도된 쓰기들을 트래핑하여 그러한 함수 호출들이 상주하는 메모리를 보호함으로써 그러한 공격들에 대해 방어할 수 있다. 마찬가지로 O/S 하위 보안 에이전트(5020)는 악성 코드를 주입하기 위해 코드 섹션을 바로 액세스할 수 있는 악성 소프트웨어에 대해 함수의 코드 섹션을 보호할 수 있다. 예를 들어 O/S 하위 보안 에이전트(5020)는 주입 코드의 추가를 막기 위해 코드 섹션 2 안에 들어 있는 함수의 코드에 대해 시도되는 쓰기들을 트래핑할 수 있다.

[0588] 드라이버간 호출들과 관련된 다양한 자원들을 트래핑하는 것이 비용이 많이 들 수 있기 때문에, O/S 하위 보안 에이전트(5020)는 필요 시 그러한 자원들의 트래핑을 인에이블(가능) 또는 디세이블(불능)할 수 있다. 각각의 트래핑된 시도에 대해, O/S 하위 보안 에이전트(5020)는 그 작용 드라이버나 모듈을 식별하고, 타깃 드라이버를

식별하며, 액세스 타입을 식별할 수 있다. 그러한 타입은 읽기, 쓰기 또는 실행 타입을 포함할 수 있다. O/S 하위 보안 에이전트(5020)는 전자 장치(5001)의 자원들을 액세스하려는 시도가 악성인지 아닌지 여부를 평가하기 위해 이러한 요소들을 어떤 다른 적절한 기준과 함께 고려할 수 있다.

[0589] 도 53은 전자 장치에서 드라이버 간 통신에 대한 운영체제 하위 트래핑 방법(5300)의 실시예이다. 단계 5305에서 보호될 드라이버간 통신과 관련된 자원들을 판단하기 위해 보안 규칙들이 액세스될 수 있다. 그러한 보안 규칙들은 자원들, 및 자원에 대해 시도된 액세스가 트래핑되고 평가되게 할 기준을 식별할 수 있다.

[0590] 단계 5310에서 전자 장치 안의 운영체제들의 레벨 아래의 제어 구조 안에서 플래그들이 세팅될 수 있다. 플래그들은 예컨대 드라이버간 통신 함수들의 실행, 메모리에 로딩된 드라이버들의 데이터나 코드 섹션들에 대한 읽기나 쓰기, 드라이버간 통신 서브 함수들의 로드의 실행 및/또는 드라이버간 통신을 위한 드라이버의 코드 섹션들로의 점프, 분기, 또는 기타 직접적 실행 시도를 트래핑하기 위해 세팅될 수 있다. 플래그들은 상술한 시도들에 대응하는 메모리 페이지들을 통한 가상 메모리 액세스 및/또는 메모리 어드레스들을 통한 물리적 메모리 액세스에 대해 세팅될 수 있다.

[0591] 단계 5315에서 전자 장치가 드라이버간 통신과 관련된 자원들을 액세스하려는 시도들의 트래핑을 위해 감시될 수 있다. 단계 5320에서 어떤 시도들도 트래핑되지 않았으면, 프로세스(5300)는 트래핑된 시도들에 대해 계속 감시하도록 단계 5315로 진행할 수 있다. 시도가 트래핑되었으면, 그 시도는 단계 5325에서 시작하여 처리될 수 있다. 그러한 처리는 전자 장치의 운영체제들의 레벨 아래에서 수행될 수 있다. 단계 5325에서, 그 시도가 악성인지 여부를 분석하기 위해 유용한 정보가 수집될 수 있다. 예를 들어 해당 시도를 한 프로세스, 애플리케이션, 드라이버 또는 루틴이 결정될 수 있다. 전자 장치의 운영체제 안으로부터의 정황 정보가 O/S 내 보안 에이전트로부터 획득될 수 있다.

[0592] 단계 5330에서, 드라이버 간 통신과 관련된 어떤 드라이버의 데이터 섹션에 대해 시도된 액세스가 허가되지 않은 것인지 여부가 판단될 수 있다. 그러한 데이터 섹션의 콘텐츠는 EAT, IAT 또는 어떤 다른 적절한 정보를 포함할 수 있다. 허가되지 않은 경우, 단계 5360에서 해당 시도가 의심스럽고/거나 악성이라고 판단할 수 있으며, 해당 시도가 거부될 수 있다.

[0593] 허가된 것인 경우, 단계 5335에서, 허가된 함수를 사용하지 않고 드라이버간 통신을 위한 함수의 콘텐츠가 바로 액세스되었는지 여부가 판단될 수 있다. 일 실시예에서, 호출 프로세스나 루틴이 드라이버의 그러한 부분들을 액세스함에 있어 허가되지 않은 것인지 여부가 판단될 수 있다. 허가되지 않은 경우, 단계 5360에서 해당 시도가 의심스럽고/거나 악성이라고 판단할 수 있고, 해당 시도가 거부될 수 있다. 어떤 적절한 교정 액션이 수행될 수 있다. 허가된 경우, 단계 5345에서 그러한 액세스를 위해 지정된 함수를 사용하지 않고 드라이버간 통신의 서브 함수가 직접 실행되었는지 여부가 판단될 수 있다. 일 실시예에서, 호출 프로세스나 루틴이 그러한 시도를 함에 있어 허가되지 않은 것인지 여부가 판단될 수 있다. 허가되지 않은 경우, 단계 5360에서 해당 시도가 의심스럽고/거나 악성이라고 판단할 수 있고, 해당 시도가 거부될 수 있다. 허가된 경우, 단계 5350에서, 드라이버간 통신이 허가된 개체에 의해 호출되었는지 여부 또는 시도된 분기, 점프 또는 다른 직접 실행이 허가된 개체에 의해 호출되었는지 여부가 판단될 수 있다. 그렇지 못한 경우, 단계 5360에서 해당 시도가 의심스럽고/거나 악성이라고 판단할 수 있고, 해당 시도가 거부될 수 있다. 그에 해당하는 경우, 단계 5355에서 해당 시도가 허용될 수 있다.

[0594] 해당 시도가 허용되고 재호출 함수를 등록하기 위한 것이면, 단계 5365에서 새로 추가된 재호출 함수에 할당된 메모리가 추가 트래핑을 위해 마킹될 수 있다. 그러한 단계는 특히, 재호출 함수를 등록하고자 시도하는 개체가 알려지지 않았을 때의 경우나, 재호출을 등록했던 개체의 악성 소프트웨어 상태가 결과적으로 결정될 수 없을 경우에 수행될 수 있다. 따라서 재호출 함수의 코드에 의한 후속 읽기, 쓰기 또는 실행들이 트래핑되고 평가될 수 있다. 그렇지 않으면 단계 5385에서 드라이버가 실행 허용될 수 있다. 방법(5300)은 옵션으로서, 드라이버간 통신을 위해 전자 장치의 자원들에 대해 시도된 액세스들의 감시를 계속하기 위해 단계 5315로 돌아갈 수 있다.

[0595] 도 54는 전자 장치(5401) 상의 드라이버 필터들의 첨부 및 분리를 보호하기 위한 시스템(5400)의 실시예이다. 시스템(5400)은 운영체제(5412)와 같은 전자 장치(5401)의 운영체제들의 드라이버 필터들을 첨부하거나 분리하려는 악의적 시도들을 검출하기 위해 전자 장치(5401) 상에서 동작하도록 구성된 O/S 하위 보안 에이전트(5420)를 포함할 수 있다. 또한 O/S 하위 보안 에이전트(5420)는 시도된 어떤 드라이버 필터들의 첨부나 분리들이 트래핑된 동작에 대응할 수 있고, 그 시도 및 그 시도를 수행한 개체에 기반하여 그 시도들이 허가되는지 여부를 판단하기 위해 하나 이상의 보안 규칙들(5408)을 이용하도록 구성될 수 있다. O/S 하위 트래핑 에이전트

(5420)는 트래핑된 이벤트에 대해 허용하거나 거부하거나 그에 대해 다른 교정 액션을 수행하도록 구성될 수 있다.

[0596] 전자 장치(5401)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701), 도 9의 전자 장치(901), 도 12의 전자 장치(1201) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(1201)는 물리적 메모리(1203)와 같은 메모리에 연결된 하나 이상의 프로세서들(1202)을 포함할 수 있다. 프로세서(5402)는 도 2의 프로세서(208), 도 4의 프로세서(408), 도 7의 프로세서(702), 도 9의 프로세서(1202), 도 12의 프로세서(1202) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(5403)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703), 도 9의 메모리(903), 도 12의 물리적 메모리(1203) 또는 가상 메모리(1204) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(5401)는 하나 이상의 보안 규칙들(5421)에 연결된 O/S 내 보안 에이전트(5419)를 포함할 수 있는 운영체제(5412)를 포함할 수 있다. 운영체제(5412)는 도 1의 운영체제들(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 9의 운영체제(913), 도 12의 운영체제(1213) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(5419)는 도 1의 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418) 및/또는 도 7의 O/S 내 보안 에이전트(719), 도 9의 O/S 내 보안 에이전트(919), 도 12의 O/S 내 보안 에이전트(1219) 또는 이들의 어떤 적절한 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0597] O/S 하위 보안 에이전트(5420)는 도 1의 O/S 하위 트래핑 에이전트(104)나 트리거된 이벤트 핸들러(108), 도 2의 SVMM(216) 또는 SVMM 보안 에이전트(217), 도 4의 펌웨어 보안 에이전트들(440, 442), O/S 하위 에이전트(450), 또는 PC 펌웨어 보안 에이전트(444), 도 5의 펌웨어 보안 에이전트(516), 또는 도 7의 마이크로코드 보안 에이전트(708)나 O/S 하위 에이전트(712), 도 9의 O/S 하위 트래핑 에이전트(920) 또는 트리거된 이벤트 핸들러(922), 도 12의 O/S 하위 보안 에이전트(1220) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0598] 보안 규칙들(5408)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(222), 도 4의 보안 규칙들(434, 436, 438), 도 5의 보안 규칙들(518), 도 7의 보안 규칙들(707, 723), 도 9의 보안 규칙들(908), 도 12의 보안 규칙들(1208) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보안 규칙들(5421)은 도 2의 보안 규칙들(220), 도 4의 보안 규칙들(420), 도 7의 보안 규칙들(721), 도 9의 보안 규칙들(921), 도 12의 보안 규칙들(1221) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0599] 전자 장치(5401)는 하나 이상의 애플리케이션들, 드라이버들 또는 드라이버를 이용하여 전자 장치(5401)의 자원들을 액세스하고자 할 수 있는 다른 개체들(예를 들어 "애플리케이션1")을 포함할 수 있다. 일 실시예에서 그러한 자원은 I/O 장치(5430)일 수 있다. I/O 장치(5430)는 예컨대 스토리지 장치, 디스플레이 장치, 주변기기, 키보드 또는 입/출력을 위해 사용할 전자 장치(5401)의 어떤 다른 장치나 구성요소를 포함할 수 있다. 일 실시예에서 I/O 장치(5430)는 가상 장치일 수 있다. 운영체제(5412)는 자원에 대한 요청을 처리할 수 있다. 일 실시예에서 운영체제(5412)는 I/O 관리자(5422)와 같은 요청들을 위한 핸들러를 포함할 수 있다. I/O 관리자(5422)는 자원에 대한 요청을 파싱하여 처리하고, 요청의 추가 처리를 위해 해당 요청을 적합한 드라이버로 보내도록 구성될 수 있다. 예를 들어 I/O 관리자(5422)는 그러한 I/O 요청을 애플리케이션1에서 I/O 드라이버(5428)로 보낼 수 있다. 운영체제(5412)는 비한정적으로 디스플레이, 키보드, 디스크 스토리지, 시리얼, USB(Universal Serial Bus), 파이어와이어, IEEE-488, 플러그인 보드, 프린터, 컴퓨터 버스 또는 네트워킹을 포함하는 자원들에 대한 요청들을 관리 및 변환할 어떤 적절한 수의 다양한 드라이버들을 포함할 수 있다. I/O 드라이버(5428)와 같은 드라이버들은 예컨대 I/O 장치들로의 직접 어드레싱을 수행하도록 구성될 수 있다. 일 실시예에서 운영체제(5412)는 하드웨어 장치를 모방할 수 있는 가상 장치 드라이버들을 포함할 수 있다.

[0600] I/O 드라이버(5428)는 장치 스택(5424)과 같은 구조의 사용을 통해 액세스될 수 있다. 장치 스택(5424)은 드라이버 및 어떤 추가 드라이버 필터들을 포함하는 구조일 수 있다. 예를 들어 장치 스택(5424)은 I/O 드라이버(5428) 위에 상주하는 하나 이상의 I/O 필터들을 포함할 수 있다. I/O 요청과 같은 요청이 드라이버 스택(5424)를 통해 드라이버(5428)로 보내질 수 있으나 I/O 필터들(5426)에 의해 인터셉트될 수 있다. 운영체제(5412)는 드라이버나 자원에 대한 특수 동작들을 수행하기 위한 어떤 적절한 수의 다양한 드라이버 필터들을 포함할 수 있다. 예를 들어 I/O 드라이버 필터들(5426)과 같은 드라이버 필터들은 요청을 조건화하거나 포맷하거

나, 최적화를 제공하거나, 결과들을 캐싱하거나 어떤 다른 적절한 함수를 수행할 수 있다. I/O 드라이버 필터들(5426)과 같은 드라이버 필터들의 특정 구현들은 드라이버 자체의 성격 및/또는 아이디에 따라 달라질 수 있다. 예를 들어 어떤 드라이버 필터들은 디스플레이, 키보드 또는 파일 스토리지와 같은 특정 종류의 모든 드라이버들에 대해 적용될 수 있고, 어떤 드라이버 필터들은 특정 드라이버의 특정 브랜드나 모델에 적용 가능할 수 있다. I/O 요청과 같은 요청을 수신한 후, I/O 필터들(5426)과 같은 필터들은 요청에 따르거나 요청에 대신하여 동작들을 수행하고, 그런 다음 I/O 드라이버(5428)와 같은 드라이버로 필터링된 요청을 전달할 수 있다. I/O 드라이버(5428)는 I/O 장치(5430)와 같은 장치와 통신하고 그에 따라 미가공 결과를 수신할 수 있다. 미가공(raw) 결과는 I/O 필터들(5426)과 같은 동일한 필터들을 통해 다시 보내진다. I/O 필터들(5426)은 포맷팅, 콘텐츠, 프레젠테이션 또는 어떤 다른 적절한 목적을 위해 결과들을 필터링할 수 있다. 필터링된 결과는 I/O 관리자(5422)나 궁극적으로 애플리케이션 1과 같은 장치 스택을 호출했던 개체로 다시 보내질 수 있다.

[0601] 도 55는 장치 스택(5500) 예의 동작에 대한 상세도이다. 예시의 목적 상, 장치 스택(5500)은 애플리케이션들을 스토리지 디스크 상의 파일들과 인터페이스하기 위해 파일 I/O 드라이버 스택으로서 구성될 수 있다. 장치 스택(5500)은 파일 I/O 드라이버(5506)로/로부터의 요청들을 필터링하도록 구성된 "필터1"(5502) 및 안티 악성 소프트웨어 파일 I/O 필터(5504)를 포함할 수 있다. 장치 스택(5500)의 베이스는 파일 I/O 드라이버(5506)일 수 있고, 요청들이 장치 스택(5500)으로 들어와서 드라이버로 보내질 수 있고, 결과들이 위로 돌려보내져서 장치 스택(5500)의 최상위에서 액세스될 수 있다. 예를 들어 요청들은 필터1(5502)에 의해 수신되고, 안티 악성 소프트웨어 파일 I/O 필터(5510)로 보내지며, 그런 다음 파일 I/O 드라이버(5506)로 보내질 수 있다. 각각의 필터는 스택 위나 아래로 그 요청을 전달하기 전에 개별 필터링 동작들을 수행하도록 구성될 수 있다. 파일 I/O 드라이버(5506)는 필터링된 I/O 요청을 수행하고 그 결과들을 자체 필터링된 결과들을 필터1(5502)으로 돌려보내도록 구성될 수 있는 안티 악성 소프트웨어 파일 I/O 필터(5504)로(존재할 경우) 돌려보내도록 구성될 수 있다. 필터1 및 안티 악성 소프트웨어 파일 I/O 필터(5504)는 각각 상기 결과들에 대해 필터링 동작들을 수행하도록 구성될 수 있다. 장치 스택(5500)의 필터들은 읽기, 쓰기 또는 실행과 같은 어떤 적절한 요청을 필터링하도록 구성될 수 있다.

[0602] 장치 스택(5500)은 스택을 구성하고 장치 스택(5500) 안의 필터들 및 드라이버들 사이의 통신을 돕기 위한 어떤 적절한 메커니즘을 포함할 수 있다. 예를 들어 장치 스택(5500)은 장치 스택(5500)의 베이스를 식별하고 필터들의 차수를 식별하기 위한 데이터 구조들을 포함할 수 있다. 장치 스택(5500)을 정렬하는 데이터 구조의 한 예는 포인터들(5508, 5510, 5512 또는 5514)을 포함할 수 있다. 각각의 포인터는 스택 안에서 상향 또는 하향에 위치한 다음 개체의 어드레스를 포함할 수 있다. 예를 들어 안티 악성 소프트웨어 파일 I/O 필터(5510)는 파일 I/O 드라이버(5514)일 수 있는 스택 아래의 다음 개체를 가리키는 포인터(5512), 및 필터1(5502)일 수 있는 스택 위의 다음 개체에 대한 포인터(5510)를 포함할 수 있다. 그러한 데이터 구조의 예가 장치 객체에 의해 구현될 수 있다.

[0603] 필터1(5502)은 파일 "Kernel.DLL"의 컨텐츠가 다른 파일 "Malware.DLL"로 덮어쓰여지도록 명령하는 명령인 "Write Malware.DLL to Kernel.DLL"과 같은 파일 I/O 요청을 수신하도록 구성될 수 있다. 필터1(5502)은 요청을 수신하고, 그 요청에 대한 자신의 동작들을 수행하며, 그 요청을 안티 악성 소프트웨어 I/O 필터(5504)일 수 있는 스택 아래의 다음 개체로 보낼 수 있다. 안티 악성 소프트웨어 파일 I/O 필터(5504)는 허가되지 않은 프로그램들에 의한 부당변경(tampering)으로부터 시스템의 코어 파일들을 보호하도록 구성될 수 있다. 예를 들어 악성 소프트웨어는 커널 운영체제 콘텐츠, 마스터 부트 레코드들 또는 안티 악성 소프트웨어 소프트웨어 파일들과 같은 어떤 시스템 파일들을 변경하거나 삭제하도록 시도할 수 있다. 도 55의 예에서, 요청은 "Kernel.DLL"과 같은 보호 파일 위에 알려지지 않은 잠정적 악성 파일 "Malware.DLL"을 쓰도록 하는 시도일 수 있고/있거나 요청이 운영체제의 커널 프로세스들이 아닌 프로세스로부터 발생될 수 있다. 그에 따라 안티 악성 소프트웨어 파일 I/O 필터(5504)는 예컨대 그러한 요청을 필터링하고 그 요청이 파일 I/O 드라이버(5506)에 도달하는 것을 차단하도록 구성될 수 있다. 안티 악성 소프트웨어 파일 I/O 필터(5504)는 추가 분석을 위해 시스템 상에서 실행되는 안티 악성 소프트웨어 에이전트(5516)로 상기 차단된 요청을 보내도록 구성될 수 있다. 일 실시예에서 안티 악성 소프트웨어 에이전트는 도 54의 O/S 내 보안 에이전트(5419)나 어떤 적절한 안티 악성 소프트웨어 모듈, 소프트웨어, 시스템 또는 프로세스에 의해 전체나 일부가 구현될 수 있다. 안티 악성 소프트웨어 파일 I/O 필터(5504)는 스택 위로 스푸핑된 응답을 돌려보내어, 해당 요청을 시도한 개체가 마치 쓰기가 성공했던 것처럼 진행할 수 있도록 구성될 수 있다. 그렇지 않고, 쓰기 시도가 의심스럽거나 악성인 것으로 간주되지 않으면, 안티 악성 소프트웨어 파일 I/O 필터(5504)는 그 요청을 파일 I/O 드라이버(5506)로 전달하고 그 결과를 필터1(5502)로 돌려보내도록 구성될 수 있다.

[0604] 도 56은 드라이버 필터들을 첨부하거나 분리하도록 동작하는 악성 소프트웨어에 의해 손상되었을 수 있는 장치 스택들의 예시도이다. 파일 I/O 장치 스택(5602)은 도 54의 장치 스택(5424)나 도 55의 장치 스택에 의해 기술된 것과 같은 장치 스택의 구현예일 수 있다. 파일 I/O 장치 스택(5602)은 파일 I/O 드라이버(5610)에 대한 액세스를 제공하고, "필터1"(5606) 및 안티 악성 소프트웨어 파일 I/O 필터(5608)과 같은 필터를 포함할 수 있다. 안티 악성 소프트웨어 파일 I/O 필터(5608)는 도 55의 안티 악성 소프트웨어 파일 I/O 필터(5504)의 구현예일 수 있다. 안티 악성 소프트웨어 파일 I/O 필터(5608)는 드라이버 스택(5602)으로부터 분리되었을 수 있다. 그러한 분리는 드라이버 필터를 분리하거나 제거하기 위한 시스템 함수 호출이나 파일 I/O 장치 스택(5602)의 데이터 구조들에 대한 직접적인 조작에 의해 수행되었을 수 있다. 필터1(5606)은 더 이상 요청을 안티 악성 소프트웨어 파일 I/O 필터(5608)를 통해 전달하지 않고, 대신 그것을 우회하여 그 요청을 파일 I/O 드라이버(5610)와 같은 다음 개체로 바로 보낼 수 있다. 포인터(5612)는 대신 안티 악성 소프트웨어 파일 I/O 필터(5608)를 지나 다음 개체를 가리키도록 변경되었을 수 있다. 파일 I/O 드라이버(5610)는 더 이상, 안티 악성 소프트웨어 파일 I/O 필터(5608)에 의해 필터링된 요청을 수신하지 않을 것이다. 수신된 요청에 응하여, 파일 I/O 드라이버(5610)는 필터1으로 응답을 다시 보낼 수 있는 데, 이는 안티 악성 소프트웨어 파일 I/O 필터(5608)가 더 이상 필터들의 베이스에 없도록 파일 I/O 드라이버(5610)의 포인터(5614)가 수정되었을 수 있기 때문이다. 따라서 안티 악성 소프트웨어 파일 I/O 필터(5608)가 파일 I/O 장치 스택(5602)으로부터 효과적으로 제거될 수 있다.

[0605] 안티 악성 소프트웨어 파일 I/O 필터(5608)의 제거는 허가되었거나, 악성 소프트웨어 공격의 결과였을 수 있다. Ring0와 같이 안티 악성 소프트웨어 파일 I/O 필터(5608)와 동일한 실행 우선순위에서 동작하는 악성 소프트웨어는 검출되지 않고 필터를 분리시키는데 있어 성공적일 수 있다. 안티 악성 소프트웨어 파일 I/O 필터(5608)가 일례로서 보여지고 있지만, 다른 드라이버 필터들도 마찬가지로 공격받을 수 있다.

[0606] 키보드 I/O 장치 스택(5604)은 도 54의 장치 스택(5424)나 도 55의 장치 스택에 의해 기술된 것과 같은 장치 스택의 전체 혹은 부분 구현예일 수 있다. 키보드 I/O 장치 스택(5604)은 시스템의 키보드 장치에 대한 액세스를 제공하도록 구성될 수 있다. 일 실시예에서, 키보드 I/O 장치 스택(5604)은 원래, 키보드 드라이버(5620) 위에 필터1(5616)와 같은 필터를 포함했을 수 있다. 그러나 드라이버 필터 첨부 동작이 악성 키로거 필터(5618)를 키보드 I/O 장치 스택(5604) 안에 첨부했을 수 있다. 필터1(5616)의 포인터(5622)나 키보드 드라이버(5620)의 포인터(5628)와 같은 스택의 데이터 구조는 악성 키로거 필터(5618)가 필터1(5616)과 키보드 드라이버(5620) 사이에 삽입되게 하도록 변경되었을 수 있다. 그러한 동작은 드라이버 필터를 첨부하기 위한 시스템 함수 호출을 통하거나 키보드 I/O 장치 스택(5604)의 데이터 구조들에 대한 직접적인 조작에 의해 수행되었을 수 있다. 악성 키로거 필터(5618)는 사용자 키스트로크들을 캡처하고 이들을 파일이나 원격 서버에 저장하도록 구성될 수 있다.

[0607] 악성 키로거 필터(5618)나 다른 잠정적 악성 필터들은 안티 악성 소프트웨어 소프트웨어로부터의 검출을 피하는 스택의 어느 위치에 인스톨될 수 있다. 예를 들어 잠정적 악성 필터는 안티 악성 소프트웨어 필터에 의해 수행될 수 있는 어떠한 교정 액션이라도 악성 필터에 의해 취소될 수 있도록, 스택에서 안티 악성 소프트웨어 필터보다 아래의 위치에 인스톨될 수 있다. 또한, 악성 필터는 악성 필터의 동작이 위장되도록 안전 필터 대신 스택 안에 삽입될 수 있다.

[0608] 도 54로 돌아가면, 운영체제(5412)의 레벨에서 실행되는 안티 악성 소프트웨어 소프트웨어는 드라이버 필터들의 악성 첨부 및 분리에 전적으로 대처할 수 없는데, 이는 그러한 활동들을 행하는 악성 소프트웨어 역시 동일한 우선순위 레벨에서 실행되고 있을 수 있기 때문이다.

[0609] O/S 하위 보안 에이전트(5420)는 하위 레벨 운영체제(5412)에 드라이버 필터를 첨부 또는 분리하는 시도를 인터셉트하도록 구성될 수 있다. O/S 하위 보안 에이전트(5420)는 운영체제(5412) 아래의 레벨 하위 보안 에이전트(5420)는 드라이버 필터들의 첨부 및 분리와 관련된 자원들을 판단하고 그러한 자원들에 대해 시도된 액세스를 트래핑하기 위해 보안 규칙들(5408)을 조회하도록 구성될 수 있다. 그러한 자원들은 예컨대 메모리(5403)의 부분들을 포함할 수 있다. 그러한 액세스 시도들을 트래핑한 후, O/S 하위 보안 에이전트(5420)는 보안 규칙들(5408)에 기반하여 해당 액세스를 시도하는 개체가 액션을 수행하도록 허가되는지 여부를 판단하도록 구성될 수 있다. O/S 하위 보안 에이전트(5420)는 요청을 허용 또는 거부하거나 다른 알맞은 액션을 수행하도록 구성될 수 있다.

[0610] 일 실시예에서 메모리(5403)는 가상 메모리를 포함할 수 있다. 그러한 실시예에서, 메모리(5403)는 첨부 함수들(5436), 첨부 서브 함수들(5438), 분리 함수들(5440) 및/또는 분리 서브 함수들(5442)의 코드; 드라이버 데이

터 구조들의 허가 사항들(5444)로부터의 데이터; 및/또는 드라이버 데이터 구조들(5446) 자체를 포함하는 메모리 페이지들을 포함할 수 있다. 첨부 함수들(5436) 및 분리 함수들(5440)이 전자 장치(5401) 내 개체들의 운영체제(5412)에 의해 첨부 또는 분리 드라이버들로 제공될 수 있다. 그러한 개체들은 정상적으로 첨부 함수들(5436)이나 분리 함수들(5440)을 호출할 수 있다. 첨부 함수들(5436) 및 분리 함수들(5440)을 제공함으로써, 운영체제(5412)는 드라이버 필터들을 추가하거나 제거하기 위해 전자 장치(5401)의 개체들로 통제되고 안전하며 효율적인 메커니즘들을 제공할 수 있다. 그러나 첨부 서브 함수들(5438) 및 분리 서브 함수들(5442)은 운영체제에 의해 문서화되지 않거나, 사용되지 못할 수 있다. 첨부 서브 함수들(5438) 및 분리 서브 함수들(5442)은 관련된 첨부 함수들(5436) 및 분리 함수들(5440)에 의해서만 사용되게 되어 있을 수 있다. 악성 소프트웨어는 첨부 서브 함수들(5438) 및 분리 서브 함수들(5442)의 개별 인스턴스들을 호출함으로써 첨부 함수들(5436) 및 분리 함수들(5440)의 보안 및 제어 메커니즘들을 우회할 수 있다.

[0611] 다른 실시예에서 메모리(5403)는 물리적 메모리를 포함할 수 있다. 그러한 실시예에서, 메모리(5403)는 시스템의 첨부 함수들(5436), 첨부 서브 함수들(5438), 분리 함수들(5440), 분리 서브 함수들(5442)의 코드; 드라이버 데이터 구조들의 허가 사항들(5444)에 관한 데이터; 및/또는 드라이버 데이터 구조들(5446) 자체를 포함하는 메모리 어드레스들을 포함할 수 있다.

[0612] 첨부 함수들(5436)의 코드는 애플리케이션이나 드라이버가 드라이버 필터를 인에이블하기 위해 시스템(5400)이나 운영체제(5412)에 의해 지정되는 어떤 함수들의 어떤 코드를 포함할 수 있다. 그러한 함수들은 I/O 필터들(5426) 중 하나를 장치 스택(5424)에 첨부하는 것과 같이 어떤 드라이버 필터를 드라이버 스택에 첨부하기 위한 함수들을 포함할 수 있다. 이러한 함수들이 다시, 드라이버 필터를 인에이블함에 있어 특정 작업들을 실행하기 위한 서브루틴들이나 다른 함수들을 호출할 수 있다. 분리 함수들(5440)의 코드를 포함하는 메모리는 애플리케이션이나 드라이버가 드라이버 필터를 디세이블하기 위해 시스템(5400)이나 운영체제(5412)에 의해 지정되는 어떤 함수들의 어떤 코드를 포함할 수 있다. 그러한 함수들은 I/O 필터들(5426) 중 하나를 장치 스택(5424)으로부터 분리하는 것과 같이 어떤 드라이버 필터를 드라이버 스택으로부터 분리하기 위한 함수들을 포함할 수 있다. 이러한 함수들이 다시, 드라이버 필터를 디세이블함에 있어 특정 작업들을 실행하기 위한 서브루틴들이나 다른 함수들을 호출할 수 있다. 일례에서 윈도우즈 첨부 함수들(5436)은 다음과 같은 것을 포함할 수 있으나 그에 국한되지 않는다: IoAttachDevice(), IoAttachDeviceByPointer(), IoAttachDeviceToDeviceStack(), and IoAttachDeviceToDeviceStackSafe(). 다른 예에서 윈도우즈 분리 함수들(5440)은 다음과 같은 것을 포함할 수 있으나 그에 국한되지 않는다: IoDeleteDevice() and IoDetachDevice().

[0613] 그러한 서브루틴들이나 다른 함수들의 코드는 첨부 서브 함수들(5438)이나 분리 서브 함수들(5442)의 코드를 포함하는 메모리 안에 포함될 수 있다. 악성 소프트웨어는 운영체제(5412)에 의한 검출을 피하기 위해 직접 서브 함수들을 호출할 수 있다. 그에 따라, 시스템(5400)의 개체가 첨부 함수들(5436)이나 분리 함수들(5440)과 같은 표준 함수들의 메모리 내 코드를 사용하지 않고 그러한 서브 함수들을 바로 호출하였다면, 그렇게 시도된 액세스는 의심스러운 것이라 판단될 수 있다. 일례에서 윈도우즈 첨부 서브 함수들(5436)은 IopAttachDevicetoDeviceStackSafe()을 포함할 수 있지만 그에 국한되지 않는다. 그러한 서브 함수는 윈도우즈 첨부 함수들(5436)의 인스턴스들 각각에 의해 호출될 수 있다.

[0614] 드라이버 데이터 구조들의 허가 사항들(5444)은 장치 스택(5424)과 같은 장치 스택과 관련된 데이터 구조들을 읽거나 쓰거나 실행하는 기능을 세팅하기 위한 테이블, 플래그들 또는 어떤 다른 적절한 데이터 구조나 표시를 포함할 수 있다. 그러한 허가 사항들(5444)은 예컨대 도 55의 포인터들(5508, 5510, 5512 또는 5514)나 도 56의 포인터들(5612, 5614, 5622, 5624, 5626 또는 5628)과 같은 장치 스택 내 포인터들을 쓰거나 읽는 기능을 제어할 수 있다. 메모리(5403) 내 허가 사항들(5444)을 변경하려는 허가되지 않은 시도는 I/O 필터들(5426) 중 하나와 같은 드라이버 필터를 악의적으로 첨부하거나 분리하려는 악성 소프트웨어의 첫 단계를 가리킬 수 있다.

[0615] 드라이버 데이터 구조들(5446)은 I/O 드라이버(5428)나 장치 스택(5424)과 같은 드라이버나 장치 스택을 구성하기 위한 어떤 적절한 데이터 구조를 포함할 수 있다. 예를 들어 드라이버 데이터 구조들(5446)은 도 55의 포인터들(5508, 5510, 5512 또는 5514)나 도 56의 포인터들(5612, 5614, 5622, 5624, 5626 또는 5628)을 포함할 수 있다. 악성 소프트웨어가 자신의 호출 루틴들을 철저하게 위장했더라도, 드라이버 필터를 첨부하거나 분리하는 것은 드라이버 데이터 구조들(5446) 내 값들의 변경을 요할 수 있다. 따라서 드라이버 데이터 구조들(5446)의 값들에 대한 허가되지 않은 시도들은 악성 소프트웨어를 가리킬 수 있다.

[0616] O/S 하위 보안 에이전트(5420)은 물리적 메모리 및/또는 가상 메모리 기반으로 메모리(5403)의 드라이버 필터 관련 콘텐츠를 보호하도록 구성될 수 있다. 예를 들어 O/S 하위 보안 에이전트(5420)는 첨부 함수들(5436), 첨

부 서브 함수들(5438), 분리 함수들(5440), 분리 서브 함수들(5442)의 코드; 드라이버 데이터 구조들의 허가 사항들(5444)로부터의 데이터; 및/또는 드라이버 데이터 구조들(5446) 자체를 포함하는 메모리 페이지들을 읽거나 쓰거나 실행하려고 시도하는 요청들을 인터셉트하도록 구성될 수 있다. 그 경우, O/S 하위 보안 에이전트(5420)는 부분적으로나 전체적으로 가상 머신 모니터 안에서 구현될 수 있다. 다른 예에서 O/S 하위 보안 에이전트(5420)는 첨부 함수들(5436), 첨부 서브 함수들(5438), 분리 함수들(5440) 및/또는 분리 서브 함수들(5442)의 코드; 드라이버 데이터 구조들의 허가 사항들(5444)로부터의 데이터; 및/또는 드라이버 데이터 구조들(5446) 자체를 포함하는 메모리 어드레스들을 읽거나 쓰거나 실행하려고 시도하는 요청들을 인터셉트하도록 구성될 수 있다.

[0617] O/S 하위 보안 에이전트(5420)는 메모리(5403)의 드라이버 필터 관련 콘텐츠에 대한 그러한 요청을 인터셉트하고 정황 정보에 비추어 그 요청을 평가하도록 구성될 수 있다. 그러한 정황 정보는 그 요청을 했던 개체들, 요청의 성격(가령, 읽기, 쓰기 또는 실행), 시도된 쓰기 값들, 개체가 해당 요청을 했던 방식, 메모리(5403)의 드라이버 필터 관련 콘텐츠을 요청하려는 이전의 시도들 및/또는 O/S 내 보안 에이전트(5419)로부터 메모리(5403)를 액세스하고자 시도했을 수 있는 운영체제(5412)의 레벨에 있는 개체들의 동작에 관한 정보를 포함할 수 있다.

[0618] 요청에 대한 평가에 기반하여 O/S 하위 보안 에이전트(5420)는 요청을 허용하거나, 요청을 거부하거나, 호출한 개체로 다시 스푸핑된 응답을 보내거나, 어떤 다른 적절한 교정 액션을 수행하도록 구성될 수 있다.

[0619] 동작 시 O/S 하위 보안 에이전트(5420)는 드라이버 필터들의 첨부 및/또는 분리를 보호하기 위해 전자 장치(5401) 상에서 동작할 수 있다. 애플리케이션, 드라이버 또는 "애플리케이션2"와 같은 다른 개체가 필터 첨부나 분리 시도를 일으킬 수 있다. 애플리케이션2는 예컨대 사용자 모드, 커널 모드에서 운영체제(5412)와 동일한 레벨이나 운영체제(5412)보다 높은 레벨에서 동작할 수 있다. O/S 하위 보안 에이전트(5420)는 전자 장치(5401) 상의 드라이버 필터들의 첨부 및 분리를 어떻게 보호할지를 결정하기 위해 보안 규칙들(5408)을 액세스할 수 있다. O/S 하위 보안 에이전트(5420)는 예컨대 첨부 함수(5436), 첨부 서브함수(5438), 분리 함수(5440) 및/또는 분리 서브함수(5442)의 코드에 대한 메모리 페이지들이나 어드레스들의 실행 시도; 드라이버 데이터 구조들에 대한 허가 사항들(5444)의 메모리 페이지들이나 어드레스들에 대한 쓰기 시도; 및/또는 드라이버 데이터 구조들(5446)에 대한 읽기나 쓰기 시도들을 캡처하도록 제어 구조 플래그들을 세팅할 수 있다. 애플리케이션2는 I/O 필터들(5426)과 같은 드라이버 필터들과 관련된 메모리(5403)의 콘텐츠를 액세스하는 것을 포함하는 다양한 메커니즘들을 통해 그러한 드라이버 필터들을 액세스하도록 시도할 수 있다.

[0620] 일 실시예에서 애플리케이션2는 예컨대 드라이버 데이터 구조들의 허가 사항들(5444)에 대해 쓰기를 시도하는 것과 같이 메모리(5403) 내 값들의 직접적 조작을 통해, 필터 첨부나 분리를 시도할 수 있다. 그렇게 시도된 쓰기는 나중에 데이터 구조들의 값들이 드라이버 필터를 첨부하거나 분리하도록 재기입되도록 읽기/쓰기할 드라이버 데이터 구조에 대해 읽기만 가능한 허가 사항들을 변경하고자 하는 시도일 수 있다. 그러한 시도는 I/O 필터들(5426)과 같은 드라이버 필터들을 액세스하기 위한 표준 보안 메커니즘들을 우회할 수 있다. 그러한 메커니즘들을 우회함으로써, 상기 시도는 운영체제(5412)의 보안 조처들로부터 위장되거나 숨겨지거나 그러한 보안 조처들을 저지시킬 수 있다.

[0621] 다른 실시예에서 애플리케이션2는 그러한 동작들을 위해 운영체제(5412)에 의해 제공되는 첨부나 분리 함수(5432)를 호출 및 실행함으로써 필터 첨부나 분리를 시도할 수 있다. 그러한 첨부나 분리 함수(5432)는 다시 첨부 또는 분리 서브 함수(5434a)의 인스턴스를 호출하여 실행할 수 있다. 첨부 또는 분리 서브 함수(5434a)는 드라이버 데이터 구조들(5446)에 대한 액세스나 그러한 드라이버 데이터 구조들에 대한 허가 사항들(5444)로의 액세스 시도를 일으키는 특정 호출들을 수행할 수 있다. 첨부 또는 분리 함수들(5432)은 운영체제(5412)에 의해 I/O 필터들(5426)과 같은 드라이버 필터들을 액세스하는 표준 보호 메커니즘으로서 제공될 수 있다. 첨부나 분리 함수(5432)는 운영체제(5412)의 소정 프로세스들만이 드라이버 필터들을 액세스하는 함수를 사용할 수 있도록 보호될 수 있다.

[0622] 또 다른 실시예에서 애플리케이션2는 드라이버 데이터 구조들(5446)에 대한 표준 보호 액세스를 위해 운영체제(5412)에 의해 제공되는 첨부 또는 분리 함수(5432)와 같은 함수들을 이용하지 않고, 첨부 또는 분리 서브 함수(5434b)에 대한 인스턴스를 바로 호출하여 실행함으로써 필터 첨부를 시도할 수 있다. 운영체제(5412)가 첨부 또는 분리 함수(5432)에 대해 그럴 수 있는 것처럼 첨부 또는 분리 서브 함수(5434b)의 사용을 보호하고 허가하기 위한 메커니즘들을 포함하지 않는 경우, 악성 소프트웨어 자신을 숨기거나 위장하거나 운영체제(5412)의 보안 조처들을 저지하기 위해 서브 함수(5434b)가 악성 소프트웨어에 의해 직접 사용될 수 있다.

[0623] O/S 하위 보안 에이전트(5420)는 메모리(5403)의 드라이버 필터 관련 콘텐츠에 대해 시도된 액세스를 트래핑할 수 있다. O/S 하위 보안 에이전트(5420)는 인터셉트된 액세스를 어떻게 처리할지를 결정하기 위한 제어 구조를 포함할 수 있다. O/S 하위 보안 에이전트(5420)는 시도된 그러한 액세스를 어떻게 다룰지를 결정하기 위해 보안 규칙들(5408)이나 보호 서버를 액세스할 수 있다.

[0624] 예를 들어 애플리케이션2에 의한 첨부 함수(5436) 또는 분리 함수(5440) 실행 시도가 트래핑될 수 있다. I/O 필터들(5426을 액세스하기 위한 표준 보호 방법과 같이 운영체제에 의해 제공될 수 있는 그러한 함수들의 사용은 예컨대 디지털 서명된 드라이버들로만 제한될 수 있다. 따라서 일 실시예에서 O/S 하위 보안 에이전트(5420)는 드라이버가 디지털 서명될 것을 요하는 규칙을 판단하고, 호출하는 애플리케이션이나 드라이버를 판단하고 드라이버가 서명되어 있는지 아닌지 여부를 판단하도록 보안 규칙들(5408)을 액세스할 수 있다. O/S 하위 보안 에이전트(5420)는 애플리케이션2의 서명 상태를 판단하기 위해 운영체제(5412)를 액세스할 수 있는 O/S 내 보안 에이전트(5419)를 액세스할 수 있다. 그러한 액세스는 운영체제(5412)의 호출 스택을 검사함으로써 이루어질 수 있다. 다른 실시예에서 O/S 하위 보안 에이전트(5420)는 예컨대 애플리케이션2의 해시에 기반하여, 애플리케이션2가 블랙리스트 상에 있는지 화이트리스트 상에 있는지 아니면 악성 상태에 대해 알려져 있지 않은지 여부를 판단할 수 있다. O/S 하위 보안 에이전트(5420)는 애플리케이션2가 알려져 있지 않으면 예방조치로서 애플리케이션2가 차단되거나 애플리케이션2에 관한 가능한 정보가 보호 서버로 보고될 수 있다고 판단할 수 있다. 또한 애플리케이션2가 알려져 있는 경우, O/S 하위 보안 에이전트(5420)는 애플리케이션2의 많은 동작들을 트래핑함으로써 애플리케이션2의 동작을 보다 철저히 모니터링할 수 있다. O/S 하위 보안 에이전트(5420)는 애플리케이션2가 악성 소프트웨어를 포함한다고 판단하거나, 애플리케이션2를 차단하거나, 애플리케이션2에 대해 전자 장치(5401)를 청소하거나 다른 교정 액션을 취할 수 있다.

[0625] 다른 예에서 애플리케이션2에 의한 첨부 서브함수(5438) 또는 분리 서브함수(5442) 실행 시도가 트래핑될 수 있다. 그러한 함수들의 사용은 정상적으로는 첨부 함수(5436)나 분리 함수(5438)와 같은 표준 보안 메커니즘의 사용을 통해 수행되어야만 할 것이다. 따라서 일 실시예에서 O/S 하위 보안 에이전트(5420)는 첨부 서브함수(5438)나 분리 서브함수(5442)를 호출한 루틴이 알려지고 나열된 표준 또는 보안 메커니즘들 중 하나가 아니라면 그 루틴에 기반하여 트래핑할 수 있다. 다른 실시예에서 O/S 하위 보안 에이전트(5420)는 첨부 서브루틴(5438)이나 분리 서브루틴(5442)에 대해 시도된 액세스들 전부를 트래핑하고, 이어서 호출 루틴을 판단하고, 호출 루틴이 표준 또는 보안 메커니즘들 중 하나가 아닌 경우 그 요청을 거부할 수 있다. 호출 루틴은 예컨대 장치 스택(5424), 드라이버 데이터 구조들(5446) 안의 정보를 통하거나 메모리(403)내 어느 메모리 페이지나 메모리 어드레스로부터 서브 함수를 실행하라는 명령이 만들어졌는지를 판단하여 그 페이지나 어드레스를 메모리 맵과 상관시킴으로써 결정될 수 있다. 호출 루틴이 첨부 함수(5436)나 분리 함수(5442)의 인스턴스라고 판단되는 경우, 그러한 함수들을 호출한 루틴이 앞서 기술한 바와 같이 검증될 수 있다. O/S 하위 보안 에이전트(5420)는 허가된 첨부 또는 분리 함수의 호출로부터 야기된 것이 아닌 첨부 서브 함수(5438)나 분리 서브 함수(5442)에 대한 어떤 호출도 거부할 수 있다.

[0626] 또 다른 예에서 애플리케이션2에 의해 드라이버 데이터 구조들에 대한 허가 사항들을 쓰거나 드라이버 데이터 구조들(5446)을 읽거나 쓰고자 하는 시도들이 O/S 하위 보안 에이전트(5420)에 의해 트래핑될 수 있다. 그러한 모든 시도들을 트래핑하는 것은 첨부나 분리 함수들의 실행으로부터 발생된 그러한 시도들의 트래핑을 포함할 수 있다. 따라서 그러한 시도들의 트래핑 시, O/S 하위 보안 에이전트(5420)는 메모리의 어느 부분 또는 어떤 개체로부터 그러한 시도가 이루어졌는지를 판단할 수 있다. 그러한 시도가 허가된 함수로부터 나왔으면, 그 시도는 허용될 수 있다. 허가된 함수 자체의 호출자가 앞서 기술한 바와 같이 검증될 수 있다. 그러한 시도가 허가된 함수로부터 이루어지지 않았다면, 그 시도는 애플리케이션2에 의해 장치 스택(5424)을 직접 조작하려는 악의적 시도를 가리킬 수 있으므로 그 시도는 차단될 수 있다.

[0627] O/S 하위 보안 에이전트(5420)는 어떤 시도가 악의적인지 아닌지 여부를 판단할 때 장치의 타입을 고려할 수 있다. 예를 들어 가상 디스크 볼륨이 특히 필터들에 의해 남용되기 쉬울 수 있다. 따라서 O/S 하위 보안 에이전트(5420)는 장치의 타입을 판단하기 위해 객체와 같은 드라이버 데이터 구조들(5446)을 액세스할 수 있고 그 타입이 "FILE VIRTUAL VOLUME"인 경우 요청자가 디지털 서명되어 있을 것을 요할 수 있다. 그러한 요건은 운영체제(5412)에 의한 요건들과 독립적인 것일 수 있다. 어떤 시도를 트래핑해야 할지 여부를 판단하거나 그 시도가 악성인지 여부를 판단할 때 고려되어야 할 장치 객체들의 다른 타입들로는 비한정적인 것으로서, 한번 쓰기 가능한 매체, 가상 볼륨들, 탈착가능 매체, 원격 장치들, 플로피 디스켓, 읽기만 가능한 장치들, 장착된 장치들, 플러그 앤 플레이 장치들 또는 자동으로 생성된 이름들을 가진 장치들이 포함될 수 있다. 보안 규칙들(5408)은 장치 객체들의 그러한 타입에 대한 고려사항을 포함할 수 있다. 예를 들어 모뎀 타입의 장치는 호출한 드라이

버가 알려지지 않은 경우 모든 첨부에 대해 보호될 수 있다. 이것은 O/S 하위 보안 에이전트(5420)가 악성 드라이버들이 팩스 및 모뎀 동작들에 대해 스니핑하는 것을 방지하게 할 수 있다. 다른 예에서, 장치 스캐너의 드라이버들에 대한 어떤 알려진 합법적 필터 사용도 존재하지 않는 경우, 장치 스캐너의 드라이버는 모든 첨부에 대해 보호될 수 있다.

[0628] 상술한 바와 같이 O/S 하위 보안 에이전트(5420)는 I/O 필터들(5426)을 액세스하고자 시도한 개체에 기반하여 시도를 일으키거나 처리할 수 있다. 또한 O/S 하위 보안 에이전트(5420)는 액세스되어야 하는 장치의 소유 드라이버를 결정하고, 시도가 악성인지 아닌지 여부를 판단할 때 그 정보를 고려할 수 있다. 드라이버 데이터 구조들(5446)이나 장치 데이터 구조들은 각각 드라이버 및 장치를 링크하는 정보를 포함할 수 있다. 그러한 데이터 구조들은 드라이버와 장치 사이의 관계를 판단하기 위해 액세스될 수 있다. 만일, 예컨대 I/O 필터들(5426)을 액세스하려는 시도가 애플리케이션2에 의해 있었지만 애플리케이션1이 I/O 장치(5340)를 소유한다고 판단되면, 그 요청은 거부될 수 있다.

[0629] 도 57은 전자 장치에서 드라이버 필터 첨부에 대한 운영체제 하위 트래핑을 위한 방법(5700)의 실시예이다. 단계 5705에서 보호될 드라이버 필터 첨부와 관련된 자원들을 판단하기 위해 보안 규칙들이 액세스될 수 있다. 그러한 보안 규칙들은 자원들, 및 자원에 대해 시도된 액세스가 트래핑되고 평가되게 할 기준을 식별할 수 있다.

[0630] 단계 5710에서 전자 장치 안의 운영체제들의 레벨 아래의 제어 구조 안에서 플래그들이 세팅될 수 있다. 플래그들은 예컨대 첨부 함수들이나 분리 함수들의 실행, 첨부 서브 함수들이나 분리 서브 함수들의 실행, 드라이버 및 장치 데이터 구조들의 읽기/쓰기/실행 허가 사항들에 대한 쓰기, 및/또는 데이터 구조들 자체에 대한 읽기나 쓰기들의 시도들을 트래핑하기 위해 세팅될 수 있다. 플래그들은 상술한 시도들에 대응하는 메모리 페이지들을 통한 가상 메모리 액세스 및/또는 메모리 어드레스들을 통한 물리적 메모리 액세스에 대해 세팅될 수 있다.

[0631] 단계 5715에서 전자 장치가 드라이버 필터들의 첨부와 관련된 자원들을 액세스하려는 시도들의 트래핑을 위해 감시될 수 있다. 단계 5720에서 어떤 시도들도 트래핑되지 않았으면, 프로세스(5700)는 트래핑된 시도들에 대해 계속 감시하도록 단계 5715로 진행할 수 있다. 시도가 트래핑되었으면, 그 시도는 단계 5725에서 시작하여 처리될 수 있다. 그러한 처리는 전자 장치의 운영체제들의 레벨 아래에서 수행될 수 있다. 단계 5725에서, 그 시도가 악성인지 여부를 분석하기 위해 유용한 정보가 수집될 수 있다. 예를 들어 해당 시도를 한 프로세스, 애플리케이션, 드라이버 또는 루틴이 결정될 수 있다. 전자 장치의 운영체제 안으로부터의 정황 정보가 O/S 내 보안 에이전트로부터 획득될 수 있다. 해당 시도와 관련된 장치의 장치 타입이 그 장치의 보유 드라이버의 판단과 같이 판단될 수 있다.

[0632] 단계 5735에서, 허가된 함수를 사용하지 않고 장치 객체나 드라이버 스택의 데이터 구조들이 바로 액세스되었는지 여부가 판단될 수 있다. 일 실시예에서, 호출 프로세스나 루틴이 그러한 데이터 구조들을 액세스하는 것이 허가되지 않는지 여부가 판단될 수 있다. 데이터 구조가 직접 액세스되었다면, 단계 5760에서 해당 시도가 의심스럽고/거나 악성이라고 판단할 수 있고, 해당 시도가 거부될 수 있다. 어떤 적절한 교정 액션이 수행될 수 있다. 데이터 구조들이 직접 액세스되지 않았다면, 단계 5740에서 장치 객체나 장치 스택의 데이터 구조들에 대한 허가 사항들이 바로 기입되도록 시도되었는지 여부가 판단될 수 있다. 일 실시예에서, 호출 프로세스나 루틴이 그러한 쓰기 시도를 함에 있어 허가되지 않은 것인지 여부가 판단될 수 있다. 그러한 데이터 구조가 직접 기입되었다면, 단계 5760에서 해당 시도가 의심스럽고/거나 악성이라고 판단할 수 있고, 해당 시도가 거부될 수 있다. 그러한 데이터 구조들이 바로 기입되지 않았다면, 단계 5745에서 그러한 액세스를 위해 지정된 함수를 사용하지 않고 첨부 또는 분리 서브 함수가 바로 실행되었는지 여부가 판단될 수 있다. 일 실시예에서, 호출 프로세스나 루틴이 그러한 시도를 함에 있어 허가되지 않은 것인지 여부가 판단될 수 있다. 그러한 서브 함수들이 바로 실행되었다면, 단계 5760에서 해당 시도가 의심스럽고/거나 악성이라고 판단할 수 있고, 해당 시도가 거부될 수 있다. 그러한 서브 함수들이 바로 실행되지 않았다면, 단계 5750에서 첨부 함수나 분리 함수가 허가된 개체에 의해 호출되었는지 여부가 판단될 수 있다. 함수가 허가된 개체에 의해 호출되지 않았다면, 단계 5760에서 해당 시도가 의심스럽고/거나 악성이라고 판단할 수 있고, 해당 시도가 거부될 수 있다. 함수가 허가된 개체에 의해 호출되었으면, 단계 5755에서 해당 시도가 허용될 수 있다.

[0633] 단계들 5755나 5760의 실행 후, 방법(5700)은 옵션으로서, 드라이버 필터 첨부를 위해 전자 장치의 자원들에 대해 시도된 액세스들의 감시를 계속하기 위해 단계 5715로 돌아갈 수 있다.

[0634] 도 58은 전자 장치(5801) 상의 드라이버들의 로딩 및 언로딩을 보호하기 위한 시스템(5800)의 실시예이다. 시스템(5800)은 운영체제(5812)와 같은 전자 장치(5801)의 운영체제들의 드라이버들을 로딩하거나 언로딩하려는

악의적 시도들을 검출하기 위해 전자 장치(5801) 상에서 동작하도록 구성된 O/S 하위 보안 에이전트(5820)를 포함할 수 있다. 또한 O/S 하위 보안 에이전트(5820)는 시도된 어떤 드라이버 로딩이나 언로딩이 트래핑된 동작에 대응할 수 있고, 그 시도 및 그 시도를 수행한 개체에 기반하여 그 시도들이 허가되는지 여부를 판단하기 위해 하나 이상의 보안 규칙들(5808)을 이용하도록 구성될 수 있다. O/S 하위 트래핑 에이전트(5820)는 트래핑된 이벤트에 대해 허용하거나 거부하거나 그에 대해 다른 교정 액션을 수행하도록 구성될 수 있다.

[0635] 전자 장치(5801)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701), 도 9의 전자 장치(901), 도 12의 전자 장치(1201) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(5801)는 메모리(5803)와 같은 메모리에 연결된 하나 이상의 프로세서들(5802)을 포함할 수 있다. 프로세서(5802)는 도 2의 프로세서(208), 도 4의 프로세서(408), 도 7의 프로세서(702), 도 9의 프로세서(902), 도 12의 프로세서(1202) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(5803)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703), 도 9의 메모리(903), 도 12의 물리적 메모리(1203) 또는 가상 메모리(1204) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(5801)는 하나 이상의 보안 규칙들(5821)에 연결된 O/S 내 보안 에이전트(5819)를 포함할 수 있는 운영체제(5812)를 포함할 수 있다. 운영체제(5812)는 도 1의 운영체제들(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 9의 운영체제(913), 도 12의 운영체제(1213) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(5819)는 도 1의 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418) 및/또는 도 7의 O/S 내 보안 에이전트(719), 도 9의 O/S 내 보안 에이전트(919), 도 12의 O/S 내 보안 에이전트(1219) 또는 이들의 어떤 적절한 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0636] O/S 하위 보안 에이전트(5820)는 도 1의 O/S 하위 트래핑 에이전트(104)나 트리거된 이벤트 핸들러(108), 도 2의 SVMM(216) 또는 SVMM 보안 에이전트(217), 도 4의 펌웨어 보안 에이전트들(440, 442), O/S 하위 에이전트(450), 또는 PC 펌웨어 보안 에이전트(444), 도 5의 펌웨어 보안 에이전트(516), 또는 도 7의 마이크로코드 보안 에이전트(708)나 O/S 하위 에이전트(712), 도 9의 O/S 하위 트래핑 에이전트(920) 또는 트리거된 이벤트 핸들러(922), 도 12의 O/S 하위 보안 에이전트(1220) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0637] 보안 규칙들(5808)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(222), 도 4의 보안 규칙들(434, 436, 438), 도 5의 보안 규칙들(518), 도 7의 보안 규칙들(707, 723), 도 9의 보안 규칙들(908), 도 12의 보안 규칙들(1208) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보안 규칙들(5821)은 도 2의 보안 규칙들(220), 도 4의 보안 규칙들(420), 도 7의 보안 규칙들(721), 도 9의 보안 규칙들(921), 도 12의 보안 규칙들(1221) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0638] 전자 장치(5801)는 드라이버를 로딩하거나 언로딩하기 위해, 하나 이상의 애플리케이션들, 드라이버들 또는 드라이버를 이용하여 전자 장치(5801)의 자원들을 액세스하고자 할 수 있는 다른 개체들(예를 들어 애플리케이션(5826))을 포함할 수 있다. 애플리케이션(5826)은 어떤 프로세스, 애플리케이션, 프로그램 또는 드라이버를 포함할 수 있다. 애플리케이션(5826)은 메모리(5803)와 같은 자원을 액세스하고자 시도할 수 있다. 일 실시예에서, 애플리케이션(5826)은 메모리 페이지를 통해 메모리(5803)를 액세스하고자 시도할 수 있으며, 이때 메모리(5803)는 운영체제(5812)에 의해 가상화되어 있다. 다른 실시예에서 애플리케이션(5826)은 물리적 메모리의 어드레스에 대한 액세스를 통해 메모리(5803)를 액세스하고자 시도할 수 있다. 애플리케이션(5826)은 메모리(5803) 내 명령어들을 실행하기 위해 프로세서(5802)를 이용하고자 시도할 수 있다.

[0639] 운영체제(5812)는 전자 장치(5801) 내 애플리케이션(5826)과 같은 개체들에게 드라이버들을 로딩 및 언로딩하기 위한 함수들을 제공할 수 있다. 그러한 개체들이 보통 로딩 및 언로딩 함수들을 호출할 수 있다. 그러한 함수들을 제공함으로써, 운영체제(5812)는 드라이버들을 로딩하거나 언로딩하기 위해 전자 장치(5801)의 개체들로 통제되고 안전하며 효율적인 메커니즘들을 제공할 수 있다. 그러한 함수들은 다시 로딩 서브 함수들 및 언로딩 서브 함수들의 조합에 의존할 수 있다. 그러한 로딩 및 언로딩 서브 함수들은 운영체제(5812)에 의해 문서화되지 않거나, 사용되지 못할 수 있다. 로딩 및 언로딩 서브 함수들은 관련된 로딩 및 언로딩 함수들에 의해서만 사용되도록 의도될 수 있다. 악성 소프트웨어는 로딩 서브함수들 및 언로딩 서브함수들의 개별 인스턴스들을 호출함으로써 로딩 및 언로딩 함수들의 보안 및 제어 메커니즘들을 우회할 수 있다. 또한 악성 소프트웨어는

그러함 함수들이나 서브 함수들의 코드 섹션들 안으로 바로 점프함으로써 로딩 및 언로딩 함수들의 보안 및 제어 메커니즘들을 우회할 수 있다. 또한 악성 소프트웨어는 드라이버가 실행될 메모리 안으로 로딩될 때 드라이버를 변경하고자 시도할 수 있으며, 이때 디스크 상의 드라이버의 이미지는 악성 소프트웨어로부터 깨끗하지만 멀웨에 의해 주입된 코드가 드라이버 안에 로딩될 때 그 코드가 훼손된 드라이버를 만들어 낸다.

[0640] 운영체제(5812)에 의한 어떤 적절한 로딩 또는 언로딩 함수 제공은 로딩 또는 언로딩 드라이버들에 대해 사용될 수 있다. 예를 들어 윈도우즈 운영체제에 의해 구현되는 운영체제(5812)는 드라이버들을 로딩하기 위해 ZwLoadDriverQ 함수를 이용하고 드라이버를 언로딩하기 위해 ZwUnloadDriverQ 함수를 이용할 수 있다. 그러한 로딩 또는 언로딩 함수들은 어떤 적절한 수나 종류의 서브 함수들을 호출할 수 있다. 예를 들어, ZwLoadDriver()가 NtLoadDriver()를 호출할 수 있고, 그것은 이어서 IopLoadUnloadDriver()를 호출할 수 있다. IopLoadUnloadDriver()은 이어서 드라이버들을 로딩하기 위해 IopUnloadDriver()를 호출하거나 드라이버들을 언로딩하기 위해 IopUnloadDriver를 호출할 수 있다. 따라서 드라이버들을 로딩하거나 언로딩하기 위해 함수 호출의 계층 구조가 사용될 수 있다. 일 실시예에서 드라이버들이 다른 동작들의 부수 효과로서 로딩될 수 있다. 예를 들어 윈도우즈 운영체제에 의해 구현되는 운영체제(5812)는 전체 시스템 이미지를 로딩하기 위해 MmLoadSystemImage() 함수를 사용할 수 있고, 그 와중에 드라이버들이 결과적으로 로딩될 수 있다. 이러한 함수에 의해 로딩된 이미지는 드라이버들의 상대 어드레스들을 포함할 수 있고, 그러한 상대 어드레스들은 재배치(relocation) 테이블에 기반하여 재배치되어야 한다. 그러한 작업을 수행하기 위해 MmLoadSystemImage()에 대한 호출이 서브 함수 LdrRelocateImage를 이용할 수 있다. 다른 실시예에서 MiMapViewOfDataSection(), MiMapViewOfImageSection() 또는 MiMapViewOfPhysicalSection()와 같은 어떤 보조 함수들이 드라이버들을 로딩하기 위해 사용될 수 있다.

[0641] 드라이버들의 로딩 및 언로딩을 보호하기 위해, O/S 하위 보안 에이전트(5820)는 전자 장치(5801) 자원들의 어떤 부분들이 트래핑되어야 하고 시도와 관련된 어떤 정황 정보가 결정되어야 하는지를 판단하기 위해 보안 규칙들(5808)을 액세스하도록 구성될 수 있다. O/S 하위 보안 에이전트(5820)는 어떤 적절한 함수나 서브 함수에 대해 시도된 실행을 트래핑하도록 구성될 수 있다. O/S 하위 보안 에이전트(5820)는 이러한 함수들이나 서브 함수들에 대해 시도된 실행을 어떤 적절한 방식으로 트래핑하도록 구성될 수 있다.

[0642] 일 실시예에서 O/S 하위 보안 에이전트(5820)는 메모리(5803)에서 그러한 함수들의 실행을 위해 제어 구조 상의 플래그들을 세팅하도록 구성될 수 있다. 예를 들어 O/S 하위 보안 에이전트(5820)는 메모리(5803) 내 어드레스(D)에서 ZwLoadDriver()라는 함수 엔트리 포인트에 대해 시도된 실행을 트래핑하도록 구성될 수 있다. 예를 들어 O/S 하위 보안 에이전트(5820)는 메모리(5803) 내 어드레스 (E)에서NtLoadDriver()라는 함수 엔트리 포인트에 대해 시도된 실행을 트래핑하도록 구성될 수 있다.

[0643] 다른 실시예에서 O/S 하위 보안 에이전트(5820)는 악성 소프트웨어에 대한 강력한 표시일 수 있는 그러한 함수들의 코드 섹션들의 직접 실행들을 트래핑하도록 구성될 수 있다. 예를 들어 O/S 하위 보안 에이전트(5820)는 메모리(5803)의 어드레스 (A)에서 NtLoadDriver()의 코드 섹션에 대한 직접 액세스를 낳는 "JMP" 명령어나 그와 유사한 명령어의 어떤 시도를 트래핑하도록 구성될 수 있다.

[0644] 어떤 드라이버 로딩의 시도 시, O/S 하위 보안 에이전트(5820)는 로딩이 허가되기 전에 드라이버 로딩의 잠정 효과들을 판단하도록 구성될 수 있다. O/S 하위 보안 에이전트(5820)는 디스크(5824)와 같은 스토리지 안의 드라이버의 이미지를 검사하도록 구성될 수 있다. 예를 들어, 새 드라이버(5830)의 로딩 시도 트래핑 시, O/S 하위 보안 에이전트(5820)는 디스크(5824) 상의 새 드라이버(5830)의 이미지의 콘텐츠를 검색하도록 구성될 수 있다. O/S 하위 보안 에이전트(5820)는 디스크(5324) 상의 새 드라이브(5830)의 이미지의 콘텐츠에 대한 해시나 디지털 서명 및/또는 그러한 콘텐츠의 일부(가령, 코드 섹션)에 대한 해시나 서명을 판단하도록 구성될 수 있다. O/S 하위 보안 에이전트(5820)는 보안 규칙들(5808)과 같은 정보나 화이트리스트 또는 블랙리스트에 따라, 새 드라이버(5830)가 안전하다고 알려져 있는지, 악성이나 알려지지 않은 것이라고 알려져 있는지 여부를 판단하도록 구성될 수 있다. O/S 하위 보안 에이전트(5820)는 디스크(5824) 상에 상주할 때 새 드라이버(5830)의 레이아웃을 평가하여 새 드라이버(5830)의 아이디를 확인하도록 구성될 수 있다. O/S 하위 보안 에이전트(5820)는 동일 서명자, 서명 정보 또는 서명 날짜와 같이 새 드라이버(5830)의 생성자에 의해 제공되는 디지털 서명을 평가하도록 구성될 수 있다. O/S 하위 보안 에이전트(5820)는 디스크(5824) 안에 상주할 때 새 드라이버(5830)의 이미지의 파일들의 이름들을 평가하도록 구성될 수 있다.

[0645] 일 실시예에서 O/S 하위 보안 에이전트(5820)는 로딩 동작을 허가하지만 추가 동향 감시를 인에블시킴으로써 드라이버에 대해 시도된 로딩을 조건적으로 허용하도록 구성될 수 있다. 예를 들어 애플리케이션(5826)에 의한

새 드라이버(5830)의 로딩 시도 트래핑 시, O/S 하위 보안 에이전트(5820)는 애플리케이션(5826)이나 새 드라이버(5830)의 이미지 어느 것도 알려져 있지 않지만 로딩될 드라이버가 초기에 운영체제(5812)의 중요 부분을 간섭한다는 어떤 사인도 보이지 않는다고 판단할 수 있다. 액세스의 잘못된 포지티브 블로킹(false positive blocking)을 막기 위해 O/S 하위 보안 에이전트(5820)는 새 드라이버(5830)의 로딩을 허용하도록 구성될 수 있다. 그러나 드라이버의 성격이 아직 확실히 판단되지 않으면, O/S 하위 보안 에이전트(5820)는 메모리(5803) 내 어드레스 (B)에서 새 드라이버(5830)에 할당된 메모리 공간에 추가 플래그들이나 트리거들을 할당하도록 구성될 수 있다. 따라서, 새 드라이버(5830)의 코드가 실행되고 새 드라이버가 다양한 액션들을 취하고자 시도할 때, O/S 하위 보안 에이전트(5820)는 전자 장치(5801)에서 악성 액션을 취하지 못하도록 새 드라이버(5830)의 액션들을 감시하도록 구성될 수 있다.

[0646] 다른 실시예에서 O/S 하위 보안 에이전트(5820)는 로딩 동작을 허가하지만 메모리(5803)에 상주할 때 드라이버의 추가 보안 검증들을 수행함으로써 드라이버에 대해 시도된 로딩을 조건적으로 허용하도록 구성될 수 있다. O/S 하위 보안 에이전트(5820)는 그러한 검사가 완료될 때까지 드라이버의 실행을 중지하도록 구성될 수 있다. O/S 하위 보안 에이전트(5820)에 의해 수행되는 메모리(5803) 내 드라이버의 이미지에 대한 보안 검증들은 디스크(5824)와 같은 스토리지에 상주할 때 드라이버의 이미지에 대해 수행되는 보안 검증과 유사할 수 있다. 예를 들어 O/S 하위 보안 에이전트(5820)는 어드레스 범위 (B)에서 메모리 내 새 드라이버(5830)의 이미지의 콘텐츠를 검색하고, 이미지의 콘텐츠나 이미지 서브 섹션의 해시나 디지털 서명을 판단하고, 그 이미지를 화이트리스트나 블랙리스트와 비교하며, 메모리(5803)에 상주할 때 새 드라이버(5830)의 이미지의 파일 이름들이나 파일 레이아웃을 평가하도록 구성될 수 있다. 또한 O/S 하위 보안 에이전트(5820)는 디스크(5824)와 같은 스토리지 안의 드라이버의 이미지를 분석한 결과들과 메모리(5803) 내 드라이버의 이미지를 비교하도록 구성될 수 있다. 어떤 차이들은 로딩되었을 때 드라이버 안에 코드가 주입되었다는 것을 나타낼 수 있다.

[0647] 드라이버의 언로딩 시도 시, O/S 하위 보안 에이전트(5820)는 그러한 드라이버가 중요한지 여부를 판단하기 위해 언로딩될 드라이버의 아이디를 판단하도록 구성될 수 있다. 일 실시예에서 O/S 하위 보안 에이전트(5820)는 관련 드라이버들 및 개체들과 함께 메모리(5803) 내 메모리 범위 (B) 및 어드레스 (C) 사이에 있는 것들과 같은 값들의 상관을 나타내는 메모리 맵을 조회하도록 구성될 수 있다. 도 58의 예에서 그러한 범위는 구(old) 드라이버(5828)에 의해 사용된 메모리 공간에 대응할 수 있다. 언로딩될 드라이버의 아이디에 따라, O/S 하위 보안 에이전트(5820)는 언로딩될 드라이버의 아이디에 고유한 보안 규칙들(5808)을 적용하도록 구성될 수 있다. 예를 들어 악의적 언로딩 동작들의 잘못된 포지티브 식별(false positive identification)을 최소화하기 위해, 보안 규칙들(5808)은 전자 장치(5801)의 보안이나 동작에 중요하다고 식별되었거나 악성 소프트웨어에 취약한 드라이버들의 제거만을 한정할 수 있다.

[0648] O/S 하위 보안 에이전트(5820)는 드라이버 정보에 대해 시도된 직접적 조작을 트래핑하도록 구성될 수 있다. 그러한 직접적 조작은 악성 소프트웨어가 드라이버를 언로딩하는 시스템 함수들의 사용을 피함으로써 시도될 수 있다. 일 실시예에서 O/S 하위 보안 에이전트(5820)는 어드레스 (C) 및 메모리 범위 (B) 사이의 메모리(5830) 내 구 드라이버(5828)에 대한 것들과 같이, 특정 어드레스 범위 안에서 로딩이나 언로딩과 관련된 드라이버의 데이터 구조에 대해 시도되는 액세스를 트래핑하도록 구성될 수 있다. O/S 하위 보안 에이전트(5820)는 상기 시도가 승인된 함수 호출을 통해 수행되었는지 여부를 판단하도록 구성될 수 있다. 그런 경우가 아니면, O/S 하위 보안 에이전트(5820)는 상기 시도가 의심스럽다고 판단하도록 구성될 수 있다.

[0649] O/S 하위 보안 에이전트(5820)는 어떤 시도된 드라이버 로딩 또는 언로딩 동작의 소스를 판단하도록 구성될 수 있다. O/S 하위 보안 에이전트(5820)는 해당 시도의 소스를 고려하면서 로딩하거나 언로딩할 자원들에 대해 시도된 액세스를 평가하도록 구성될 수 있다. 예를 들어 그러한 시도가 신뢰되는 소스로부터 나온 것이고, 이때 그 동작을 수행하는 애플리케이션이나 드라이버가 디지털 서명된 것이고 그 코드 섹션의 해시가 화이트리스트 상에 있으면, O/S 하위 보안 에이전트(5820)는 그러한 소스가 악성 소프트웨어 상태가 알려지지 않은 드라이버를 로딩하는 것을 허용하도록 구성될 수 있다. 다른 예에서 시도가 악성으로 판단된 소스로부터 나온 것이면, O/S 하위 보안 에이전트(5820)는 그것이 어떠한 드라이버들이라도 로딩하거나 언로딩하는 것을 허용하지 않도록 구성될 수 있다. 또 다른 예에서, 로딩이나 언로딩의 서브 함수가 액세스되었고, 호출한 프로세스, 애플리케이션, 드라이버 또는 함수가 그 서브 함수를 호출하도록 지정된 시스템 제공 함수가 아니라면, O/S 하위 보안 에이전트(5820)는 해당 액세스를 허용하지 않도록 구성될 수 있다. 또 다른 예에서, 함수나 서브 함수의 코드 섹션이 "JMP"나 그와 유사한 명령어를 통해 바로 액세스되면, O/S 하위 보안 에이전트(5820)는 어떤 메모리 위치로부터 해당 명령어 시도가 이루어졌는지를 판단하고, 그 시도가 드라이버들을 로딩하거나 언로딩하는 데 허가된 함수나 서브 함수 안으로부터 나온 것이 아니면 그 액세스를 거부하도록 구성될 수 있다. 그러한 모든 예들

에서 O/S 하위 보안 에이전트(5820)는 전자 장치(5801)의 자원들에 대해 시도되는 액세스를 일으키는 연쇄 호출들을 판단하기 위해 호출 스택을 지나도록 구성될 수 있다. 연쇄 호출의 각각의 단계에서, O/S 하위 보안 에이전트(5820)는 호출한 개체를 평가하도록 구성될 수 있다.

[0650] 또한 O/S 하위 보안 에이전트(5820)는 관리자의 설정사항에 기반하여, 시도된 로딩 또는 언로딩이 악성인지 여부를 판단하도록 구성될 수 있다. 예를 들어 전자 장치(5801)가 동작되는 기업의 관리자가 전자 장치(5801)가 관리자로서 로깅되지 않는 경우 어떠한 드라이버들도 로딩되거나 언로딩되지 않도록 지정할 수 있다.

[0651] 또한 O/S 하위 보안 에이전트(5820)는 전자 장치(5801)의 자원들을 액세스하려는 앞서 트래핑된 시도들에 기반하여, 시도된 로딩이나 언로딩 동작이 악성인지 여부를 판단하도록 구성될 수 있다. 로딩이나 언로딩 시도가 전자 장치(5801) 상에서 검출된 다른 의심가는 액션과 관련된다고 설정하는 적절한 기준이 사용될 수 있다. 예를 들어 애플리케이션(5826)이 앞서 구 드라이버(5828)와 같은 민감한 드라이버를 언로딩하고자 하는 시도 시 거부되었었다면, O/S 하위 보안 에이전트(5820)는 애플리케이션(5826)이 의심스러운 활동들을 수행했기 때문에 애플리케이션(5826)에 의한 후속 로딩 또는 언로딩 시도들을 거부하도록 구성될 수 있다.

[0652] 일 실시예에서 O/S 하위 보안 에이전트(5820)는 프로세서(5802)에 의한 명령어들의 실행 및 메모리 페이지 별 메모리(5803) 액세스를 트래핑하도록 구성될 수 있다. 그러한 실시예에서 운영체제(5812)는 전자 장치(5801) 상에서 실행되기 위해 프로세서(5802) 및 메모리(5803)에 의존하는 개체들에게 프로세서(5802) 및 메모리(5803)에 대한 액세스를 가상화하도록 구성될 수 있다. 다른 실시예에서 O/S 하위 보안 에이전트(5820)는 프로세서(5802)에 의한 명령어들의 실행 및 물리적 메모리 어드레스 단위로 메모리(5803) 액세스를 트래핑하도록 구성될 수 있다. 그러한 실시예에서, 메모리(5803)의 콘텐츠가 인접한 것으로 보여지고 있지만, 그 콘텐츠는 물리적 메모리의 상이한 섹션들 사이에 흩어져 있을 수 있다.

[0653] 드라이버를 로딩하거나 언로딩하려는 시도가 검출되었으면, O/S 하위 보안 에이전트(5820)는 어떤 교정 액션을 수행하도록 구성될 수 있다. 예를 들어 해당 시도를 한 개체가 검역을 받거나 전자 장치(5801)로부터 제거될 수 있다. 다른 예에서, 로딩되었던 드라이버가 검역을 받거나 전자 장치(5801)로부터 제거될 수 있다.

[0654] 동작 시 O/S 하위 보안 에이전트(5820)는 드라이버들의 로딩 및 언로딩을 보호하기 위해 전자 장치(5801) 상에서 실행될 수 있다. O/S 하위 보안 에이전트(5820)는 5801의 어떤 자원들이 드라이버 로딩 및 언로딩에 대해 보호될지를 결정하기 위해 보안 규칙들(5808)을 조회할 수 있다. O/S 하위 보안 에이전트(5820)는 이때 그러한 자원들에 대해 시도된 액세스를 트래핑하기 위해 하나 이상의 제어 구조들 안에서 플래그들을 세팅할 수 있다. 예를 들어 O/S 하위 보안 에이전트(5820)는 메모리(5803) 내 어드레스 (D)에서 SwLoadDriver()의 실행, 어드레스 (A)에서 NtLoadDriver()의 액세스, 어드레스 (E)에서 NtLoadDriver()의 실행, (B)의 메모리 범위에서 새 드라이버에 할당된 공간에 대한 액세스, (C) 및 (B) 사이의 범위 안에서 구 드라이버의 공간에 대한 액세스, 또는 디스크(5824) 상의 새 드라이버의 이미지 읽기에 대한 플래그들을 세팅할 수 있다.

[0655] 애플리케이션(5826)은 새 드라이버(5830)와 같은 드라이버를 로딩하고/하거나 구 드라이버(5828)와 같은 드라이버를 언로딩하도록 시도하기 위해 전자 장치(5801)의 하나 이상의 자원들을 액세스할 수 있다. 예를 들어 애플리케이션(5826)은 새 드라이버(5830)를 로딩하기 위해 ZwLoadDriver()의 인스턴스를 호출할 수 있다. O/S 하위 보안 에이전트(5820)는 어드레스 (D)에서 함수의 실행 시도를 트래핑하고 호출한 개체, 애플리케이션(5826)의 아이디를 판단할 수 있다. O/S 하위 보안 에이전트(5820)는 애플리케이션(5826)을 검색하고, 그것의 콘텐츠의 디지털 해시나 서명을 계산하고, 애플리케이션(5826)의 아이디를 판단하기 위해 화이트리스트, 블랙리스트, 보안 규칙들(5808) 및/또는 보호 서버를 검사할 수 있다. O/S 하위 보안 에이전트(5820)는 애플리케이션(5826)이 안전하다고 알려져 있는지, 악성이라고 알려져 있는지, 알려지지 않은 것인지 여부를 판단할 수 있다. 호출된 함수, 애플리케이션(5826)의 아이디, 및 O/S 내 보안 에이전트(5819)로부터 수집된 어떤 정황 정보에 기반하여, O/S 하위 보안 에이전트(5820)는 애플리케이션(5826)이 ZwLoadDriver()(5832)와 같은 로딩 또는 언로딩 함수를 호출하는 것이 허용되는지 여부를 판단할 수 있다.

[0656] 다른 예에서 애플리케이션(5826)은 새 드라이버(5830)를 로딩하기 위해 NtLoadDriver()(5834)의 인스턴스를 호출할 수 있다. NtLoadDriver()(5834)에 대한 호출은 NtLoadDriver()(5834b)로의 직접 호출을 통하거나, 예컨대 ZwLoadDriver()(5832)(NtLoadDriver()(5834a)의 인스턴스를 다시 호출함)에 대한 호출에 의해 직접 수행될 수 있다. O/S 하위 보안 에이전트(5820)는 어드레스 (E)에서의 함수의 실행 시도를 트래핑할 수 있고, 전자 장치(5801) 안에서 어떤 개체로부터 그 호출이 이루어졌는지를 판단할 수 있다. O/S 하위 보안 에이전트(5820)는 그러한 판단을 내리기 위해 함수나 실행 스택을 반복적으로 밟아 나갈 수 있다. 연쇄 실행 안에서 발견되는 각각의 개체마다, O/S 하위 보안 에이전트(5820)는 ZwLoadDriver()에 대해 위의 예에서 내려졌던 것과 비슷한 판

단을 내릴 수 있다. 특히, O/S 하위 보안 에이전트(5820)는 그러한 목적을 위해 운영체제(5812)에 의해 제공된 함수를 통해 서브 함수가 적절히 액세스되었는지 여부를 판단할 수 있다. 따라서, O/S 하위 보안 에이전트(5820)가 호출이 ZwLoadDriver()로부터 이루어졌다고 판단하면, 그 호출은 허가될 수 있다. 그러나 호출이 애플리케이션(5826)로부터 바로 이루어졌고, 그에 따라 O/S 하위 보안 에이전트(5820)가 해당 호출이 허가된 채널을 통해 이루어지지 않았다고 판단하는 경우, 그 호출은 악성이라고 판단될 수 있다.

[0657] 또 다른 예에서, O/S 하위 보안 에이전트(5820)는 어드레스 (A)에서의 NtLoadDriver()와 같이 드라이버를 로딩하거나 언로딩하기 위한 함수나 서브 함수의 코드 섹션으로 시도된 점프, 분기 또는 다른 실행을 트래핑할 수 있다. O/S 하위 보안 에이전트(5820)는 시도된 점프, 분기 또는 실행이 NtLoadDriver()안으로부터 이루어졌는지, 다른 허가된 개체 안으로부터 이루어졌는지 여부를 판단할 수 있다. 그러한 허가된 개체 안에서부터 이루어진 것이 아닌 경우, O/S 하위 보안 에이전트(5820)는 시도된 점프, 분기 또는 실행이 악성이라고 판단할 수 있다. 그렇게 시도된 점프, 분기 또는 실행은 악성 소프트웨어가 함수나 서브 함수 호출을 피하고 드라이버를 직접 로딩하거나 언로딩하려고 시도한 결과일 수 있다.

[0658] 또 다른 예에서 O/S 하위 보안 에이전트(5820)는 새 드라이버(5830)의 이미지를 디스크(5824)로부터 메모리(5803)의 범위 (B) 안에 새 드라이버(5830)에 할당된 공간으로 로딩하는 것과 같이, 드라이버의 이미지를 스토리지로부터 메모리 안으로 로딩하고자 하는 시도를 트래핑할 수 있다. 로딩을 허용하기 전에, O/S 하위 보안 에이전트(5820)는 디스크(5824) 상의 새 드라이버(5830)의 이미지의 콘텐츠를 검사할 수 있다. 이미지가 악성 소프트웨어를 가리키지 않은 경우나, 개체가 알려지지 않은 것이거나 안전한 것으로 알려지면, O/S 하위 보안 에이전트(5820)는 드라이버가 메모리 안에 로딩되는 것을 허용할 수 있다. O/S 하위 보안 에이전트(5820)는 새로 로딩된 새 드라이버(5830)에 의한 악성 활동들에 대비해 추가 보호를 제공하기 위해 어드레스 범위 (B)의 실행 또는 액세스에 대해 추가 플래그들을 놓을 수 있다. O/S 하위 보안 에이전트(5820)는 메모리(5803)에 상주할 때의 새 드라이버(5830)의 이미지를 디스크(5824) 상에서 관찰되는 이미지와 비교하고, 어떤 변경들이 로딩 프로세스 중에 주입되었을 수 있는 코드를 나타내는지 여부를 판단할 수 있다. 그러한 어떤 코드 주입들이 악성 소프트웨어의 표시일 수 있다.

[0659] 추가 예에서 O/S 하위 보안 에이전트(5820)는 메모리(5803) 내 메모리 범위 (B) 및 위치 (C) 사이에 구 드라이버(5828)에 대해 할당된 공간과 같이 메모리에 로딩된 드라이버들의 메모리 공간으로의 쓰기 명령들과 같이 시도된 액세스들을 트래핑할 수 있다. O/S 하위 보안 에이전트(5820)는 그러한 액세스를 트래핑하고, 호출 개체를 판단하고, 그 개체가 보안 규칙들(5808)에 따라 그러한 변화를 만드는 데 있어 권리를 가지는지 여부를 판단할 수 있다. 그렇게 시도된 변경들이 전자 장치(5801)로부터 어떤 드라이버를 수동적으로 제거하려는 악성 소프트웨어 시도의 일부일 수 있다.

[0660] 드라이버 로딩 및 언로딩을 위해 자원들을 액세스하고자 시도하는 동안 O/S 하위 보안 에이전트(5820)에 의해 트래핑된 액션 시도들이 허용되거나 거부될 수 있다. O/S 하위 보안 에이전트(5820)는 추후 평가 시 이용하기 위해 해당 시도 기록, 시도한 개체를 검역 또는 제거, 보호 서버에 그 시도를 보고 또는 어떤 다른 적절한 액션과 같이 추가 교정 조처들을 취할 수 있다.

[0661] 운영체제(5812)의 레벨에서 실행되는 안티 악성 소프트웨어 소프트웨어는 드라이버들의 악성 로딩 및 언로딩에 전적으로 대처할 수 없는데, 이는 그러한 활동들을 행하는 악성 소프트웨어 역시 동일한 우선순위 레벨에서 실행되고 있을 수 있기 때문이다.

[0662] 도 59a 및 59b는 전자 장치 상의 드라이버들의 로딩 및 언로딩을 보호하기 위한 방법(5900)의 일 실시예를 도시한다. 단계 5905에서 보호될 드라이버 로딩 및 언로딩과 관련된 자원들을 판단하기 위해 보안 규칙들이 액세스될 수 있다. 그러한 보안 규칙들은 자원들, 및 자원에 대해 시도된 액세스가 트래핑되고 평가되게 할 기준을 식별할 수 있다.

[0663] 단계 5910에서 전자 장치 안의 운영체제들의 레벨 아래의 제어 구조 안에서 플래그들이 세팅될 수 있다. 플래그들은 예컨대 시도된 로딩 및 언로딩 함수들의 실행, 로딩 서브함수들이나 언로딩 서브함수들의 실행, 메모리에 로딩된 드라이버들의 이미지에 대한 쓰기, 로딩 실행 시 스토리지 내 드라이버들의 이미지들로부터의 읽기 및/또는 로딩 및 언로딩 함수들 및 서브 함수들의 코드 섹션들로의 점프, 분기 또는 다른 직접적 실행을 트래핑하기 위해 세팅될 수 있다. 플래그들은 상술한 시도들에 대응하는 메모리 페이지들을 통한 가상 메모리 액세스 및/또는 메모리 어드레스들을 통한 물리적 메모리 액세스에 대해 세팅될 수 있다.

[0664] 단계 5915에서 전자 장치가 드라이버들의 로딩 및 언로딩과 관련된 자원들을 액세스하려는 시도들의 트래핑을

위해 감시될 수 있다. 단계 5920에서 어떤 시도들도 트래핑되지 않았으면, 프로세스(5900)는 트래핑된 시도들에 대해 계속 감시하도록 단계 5915로 진행할 수 있다. 시도가 트래핑되었으면, 그 시도는 단계 5925에서 시작하여 처리될 수 있다. 그러한 처리는 전자 장치의 운영체제들의 레벨 아래에서 수행될 수 있다. 단계 5925에서, 그 시도가 악성인지 여부를 분석하기 위해 유용한 정보가 수집될 수 있다. 예를 들어 해당 시도를 한 프로세스, 애플리케이션, 드라이버 또는 루틴이 결정될 수 있다. 전자 장치의 운영체제 안으로부터의 정황 정보가 O/S 내 보안 에이전트로부터 획득될 수 있다. 드라이버를 로딩하고자 하는 시도가 이루어졌으면, 디스크 상의 드라이버의 이미지가 평가될 수 있다.

[0665] 단계 5935에서, 허가된 함수를 사용하지 않고 드라이버를 로딩하거나 언로딩하기 위해 드라이버의 콘텐츠가 바로 액세스되었는지 여부가 판단될 수 있다. 일 실시예에서, 호출 프로세스나 루틴이 드라이버의 그러한 부분들을 액세스함에 있어 허가되지 않은 것인지 여부가 판단될 수 있다. 콘텐츠가 직접 액세스되었다면, 단계 5960에서 해당 시도가 의심스럽고/거나 악성이라고 판단할 수 있고, 해당 시도가 거부될 수 있다. 어떤 적절한 교정 액션이 수행될 수 있다. 콘텐츠가 허가되지 않은 함수를 이용하거나 허가되지 않은 메모리 위치로부터 바로 액세스되지 않았으면, 단계 5940에서 시도된 로딩 동작과 관련된 스토리지 내 드라이버 이미지의 콘텐츠가 평가될 수 있다. 이미지가 악성 콘텐츠가 있는지 검색되거나, 드라이버 레이아웃이 관찰되어 기록되거나, 드라이버의 해시가 산출되거나, 디지털 증명서의 생성자가 평가되거나, 어떤 다른 적절한 조사 액션이 수행될 수 있다. 단계 5943에서 스토리지 내 디스크의 이미지에 대한 정보가 콘텐츠가 의심스럽고/거나 악성이라는 것을 나타내는지 여부가 판단될 수 있다. 콘텐츠가 의심스럽고/거나 악성인 경우, 단계 5960에서 해당 요청은 거부될 수 있다. 그렇지 않은 경우, 단계 5945에서 그러한 액세스를 위해 지정된 함수를 사용하지 않고 로딩 또는 언로딩 서브 함수가 바로 실행되었는지 여부가 판단될 수 있다. 일 실시예에서, 호출 프로세스나 루틴이 그러한 시도를 함에 있어 허가되지 않은 것인지 여부가 판단될 수 있다. 호출 프로세스가 허가되지 않은 것이면, 단계 5960에서 해당 시도가 의심스럽고/거나 악성이라고 판단할 수 있고, 해당 시도가 거부될 수 있다. 허가된 경우, 단계 5950에서, 로딩 함수나 언로딩 함수가 허가된 개체에 의해 호출되었는지 여부 또는 시도된 분기, 점프 또는 다른 직접 실행이 허가된 개체에 의해 호출되었는지 여부가 판단될 수 있다. 함수가 허가된 함수에 의해 호출되지 않았다면, 단계 5960에서 해당 시도가 의심스럽고/거나 악성이라고 판단할 수 있고, 해당 시도는 거부될 수 있다. 함수가 허가된 함수에 의해 호출되었으면, 단계 5955에서 해당 시도가 허용될 수 있다.

[0666] 해당 시도가 허용되고 그 시도가 드라이버를 로딩하기 위한 것이면, 단계 5965에서 새로 로딩된 드라이버에 할당된 메모리가 추가 트래핑을 위해 마킹될 수 있다. 그러한 단계는 특히, 드라이버를 로딩하고자 시도하는 개체가 알려지지 않았을 때의 경우나, 로딩되었던 드라이버의 악성 소프트웨어 상태가 결과적으로 결정될 수 없을 경우에 수행될 수 있다. 따라서 이어지는 드라이버 메모리의 읽기, 쓰기 또는 실행이 트래핑될 수 있다. 단계 5970에서, 메모리 안에 로딩되어 있을 때의 드라이버 이미지의 콘텐츠가 평가되고/거나, 로딩되기 전에 스토리지 안에 상주했던 드라이버의 이미지를 평가한 결과들에 대해 비교될 수 있다. 단계 5975에서, 결과들이 상이하고 악성 소프트웨어 주입을 가리키거나, 새로 로딩된 드라이버의 평가가 악성 소프트웨어를 나타내면, 단계 5980에서 드라이버가 제거되거나 검역되거나 그에 대비해 다른 적절한 교정 액션이 수행될 수 있다. 그렇지 않으면 단계 5985에서 드라이버가 실행되도록 허용될 수 있다.

[0667] 단계들 5960, 5980 또는 5985의 실행 후, 방법(5900)은 옵션으로서, 드라이버 로딩 및 언로딩을 위해 전자 장치의 자원들에 대해 시도된 액세스들의 감시를 계속하기 위해 단계 5915로 돌아갈 수 있다.

[0668] 도 60은 메모리로의 코드 로딩에 대해 운영체제 하위 트래핑 및 보호를 위한 시스템(6000)의 실시예이다. 시스템(6000)은 메모리(6003)와 같은 메모리 안으로 코드를 로딩하려는 시도들을 트래핑하기 위해 전자 장치(4501) 상에서 동작하도록 구성되는 O/S 하위 보안 에이전트(6020)를 포함할 수 있다. 일 실시예에서 O/S 하위 보안 에이전트(6020)는 커널 모드 코드를 메모리(6003) 안에 로딩하려는 시도들을 트래핑하도록 구성될 수 있다. 또한 O/S 하위 보안 에이전트(6020)는 메모리(6003) 안으로 커널 모드 코드를 로딩하려는 시도들을 트래핑하기 위해 어떤 전자 장치(6001)의 자원들을 보호해야 하고, 해당 시도 및 해당 시도를 행함에 있어 관련되는 개체들에 기반하여 그러한 시도들이 허가되는지 여부를 판단하기 위해 하나 이상의 보안 규칙들(6008)을 이용하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 시도를 허용 또는 거부하거나 다른 교정 액션을 수행하도록 구성될 수 있다.

[0669] 전자 장치(6001)는 도 1의 전자 장치(103), 도 2의 전자 장치(204), 도 4의 전자 장치(404), 도 7의 전자 장치(701), 도 9의 전자 장치(901), 도 12의 전자 장치(1201) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(6001)는 메모리(6003)와 같은 메모리에 연결된 하나 이상의 프로세서들(6002)을 포함할 수 있다. 프로세서(6002)는 도 2의 프로세서

(208), 도 4의 프로세서(408), 도 7의 프로세서(702), 도 9의 프로세서(902), 도 12의 프로세서(1202) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 메모리(6003)는 도 2의 메모리(206), 도 4의 메모리(406), 도 7의 메모리(703), 도 9의 메모리(903), 도 12의 물리적 메모리(1203) 또는 가상 메모리(1204) 또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. 전자 장치(6001)는 하나 이상의 보안 규칙들(6021)에 연결된 O/S 내 보안 에이전트(6019)를 포함하거나 그에 통신 가능하게 연결될 수 있는 운영체제(6012)를 포함할 수 있다. O/S 내 보안 에이전트(6019)는 O/S 하위 보안 에이전트(6020)와 통신 가능하게 연결될 수 있다. 운영체제(6012)는 도 1의 운영체제들(112), 도 2의 운영체제(212), 도 4의 운영체제(412), 도 7의 운영체제(713), 도 9의 운영체제(913), 도 12의 운영체제(1213) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다. O/S 내 보안 에이전트(6019)는 도 1의 O/S 내 보안 에이전트(218), 도 4의 O/S 내 보안 에이전트(418) 및/또는 도 7의 O/S 내 보안 에이전트(719), 도 9의 O/S 내 보안 에이전트(919), 도 12의 O/S 내 보안 에이전트(1219) 또는 이들의 어떤 적절한 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 전체적으로나 부분적으로 구현될 수 있다.

[0670] O/S 하위 보안 에이전트(6020)는 도 1의 O/S 하위 트래핑 에이전트(104)나 트리거된 이벤트 핸들러(108), 도 2의 SVMM(216) 또는 SVMM 보안 에이전트(217), 도 4의 펌웨어 보안 에이전트들(440, 442), O/S 하위 에이전트(450), 또는 PC 펌웨어 보안 에이전트(444), 도 5의 펌웨어 보안 에이전트(516), 또는 도 7의 마이크로코드 보안 에이전트(708)나 O/S 하위 에이전트(712), 도 9의 O/S 하위 트래핑 에이전트(920) 또는 트리거된 이벤트 핸들러(922), 도 12의 O/S 하위 보안 에이전트(1220) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0671] 보안 규칙들(6008)은 도 1의 보안 규칙들(114), 도 2의 보안 규칙들(222), 도 4의 보안 규칙들(434, 436, 438), 도 5의 보안 규칙들(518), 도 7의 보안 규칙들(707, 723), 도 9의 보안 규칙들(908), 도 12의 보안 규칙들(1208) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다. 보안 규칙들(6021)은 도 2의 보안 규칙들(220), 도 4의 보안 규칙들(420), 도 7의 보안 규칙들(721), 도 9의 보안 규칙들(921), 도 12의 보안 규칙들(1221) 및/또는 이들의 어떤 조합의 기능을 구현하기 위해 구성되거나 그러한 기능에 의해 구현될 수 있다.

[0672] O/S 하위 보안 에이전트(6020) 및/또는 O/S 내 보안 에이전트(6019)는 메모리 맵(6010)에 통신 가능하게 연결될 수 있다. 메모리 맵(6010)은 메모리(6003) 내 운영체제(6012)의 다양한 개체들의 페이지들이나 어드레스들의 위치들의 매핑을 포함할 수 있다. O/S 하위 보안 에이전트(6020) 및/또는 O/S 내 보안 에이전트(6019)는 메모리(6003) 내 어떤 주어진 메모리 위치나 페이지에 대해, 어떤 프로세스, 동적 링크 라이브러리("DLL"), 애플리케이션, 모듈 혹은 전자 장치(6020)의 다른 개체가 그 위치나 페이지와 관련되는지를 판단하기 위해 메모리 맵(6010)을 액세스하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020) 및/또는 O/S 내 보안 에이전트(6019)는 예를 들어, 메모리(6003)의 어떤 부분들을 트래핑할지, 혹은 메모리(6003)에 대해 트래핑된 액세스 시나 어떤 함수의 트래핑된 실행 시, 전자 장치(6001)의 어떤 개체가 해당 시도를 발생했는지를 판단하기 위해 그러한 정보를 이용하도록 구성될 수 있다. 또한 O/S 하위 보안 에이전트(6020)는 메모리(6003)의 어떤 영역에 대해 추가 트래핑된 시도들을 악성이라고 연관시키기 위해 메모리 맵(6010)을 이용할 수 있으며, 이때 메모리(6003) 안으로 코드를 로딩하려는 이전의 시도가 악성이라고 판단되었고 메모리(6003)의 상기 영역과 연관되었다.

[0673] 스토리지(6044)는 전자 장치(6001)에 통신 가능하게 연결되거나 전자 장치(6001) 안에 상주할 수 있다. 스토리지(6044)는 하드 디스크, 플래시 드라이브, 램(random access memory (RAM)) 디스크, 컴팩트 디스크, DVD 매체 드라이브, 또는 어떤 다른 적절한 메모리와 같이 대규모 저장을 위한 어떤 적절한 매체를 포함할 수 있다. 스토리지(6044)는 PCI(peripheral component interconnect) SATA(serial advanced technology attachment), USB(universal serial bus) 또는 파이어와이어와 같은 어떤 적절한 인터페이스를 통해 전자 장치(6001)에 통신 가능하게 연결될 수 있다.

[0674] 전자 장치(6001)는 코드를 메모리(6003) 안에 로딩하기 위해 전자 장치(6001)의 자원을 액세스하고자 할 수 있는, 하나 이상의 애플리케이션들, 드라이버들 또는 다른 개체들(예를 들어 애플리케이션(6026)이나 드라이버(6028))을 포함할 수 있다. 애플리케이션(6026) 또는 드라이버(6028)는 어떤 프로세스, 애플리케이션, 프로그램 또는 드라이버를 포함할 수 있다. 애플리케이션(6026)이나 드라이버(6028)은 직접으로, 혹은 다른 루틴들에 대한 호출들을 통해, 예컨대 읽기, 쓰기 또는 실행 명령어들을 사용하여 메모리(6003)나 스토리지(6044)의 일부를 액세스함으로써 코드를 메모리(6003) 안에 로딩하고자 시도할 수 있다. O/S 하위 보안 에이전트(6020)는 그러한 호출들이나 액세스를 인터셉트하고, 그 시도가 악성 소프트웨어를 나타내는지 여부를 판단하기 위해 O/S

내 보안 에이전트(6019)로부터의 정황 정보 및/또는 보안 규칙들(6008)을 조회하며, 어떤 알맞은 교정 액션을 취할 수 있도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 메모리(6003)나 스토리지(6044)에 대한 액세스 및/또는 프로세서(6002)의 사용을 트래핑하는 것을 통해 그러한 인터셉트를 행하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 보안 규칙들(6008) 및/또는 메모리 맵(6010)을 액세스하고 메모리(6003)나 스토리지(6044) 액세스 및/또는 프로세서 사용(6002)에 대한 어떤 시도가 트래핑될 것인지를 판단하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 트래핑되어야 하는 액션들에 대응하는 제어 구조의 플래그들을 세팅하도록 구성될 수 있다.

[0675] 일 실시예에서, 애플리케이션(6026)이나 드라이버2(6028)와 같은 개체들은 메모리 페이지를 통한 메모리(6003) 안으로의 코드 로딩과 연관된 메모리(6003)의 부분들을 액세스하도록 시도할 수 있으며, 이때 메모리(6003)는 운영체제(6012)에 의해 가상화되어 있다. 그러한 실시예에서 O/S 하위 보안 에이전트(6020)는 메모리 페이지 기반으로 메모리(6003)의 액세스나 실행 시도를 트래핑하도록 구성될 수 있다. 다른 실시예에서, 애플리케이션(6026)이나 드라이버(6028)는 그러한 실시예에서 메모리에 코드를 로딩하는 것과 관련된 메모리(6008)의 물리적 부분들에 대한 액세스를 시도할 수 있고, O/S 하위 보안 에이전트(6020)는 메모리 어드레스 기반으로 메모리(6003)의 액세스나 실행 시도를 트래핑하도록 구성될 수 있다.

[0676] 메모리(6003)는 메모리(6003) 안으로의 코드 로딩을 위한 액션들과 관련된 하나 이상의 콘텐츠나 위치들을 포함할 수 있다. O/S 하위 보안 에이전트(6020)는 어떤 그러한 콘텐츠나 위치들에 대한 액세스를 트래핑하도록 구성될 수 있다. 예시적 예들로서 다음과 같은 것들이 주어진다. 메모리(6003)는 메모리(6003) 내 다른 페이지들이나 어드레스 범위들에 대한 허가 사항들(6032)을 포함할 수 있는 페이지 테이블 디렉토리(6030)를 호스팅하기 위한 어드레스들 범위들이나 페이지들을 포함할 수 있다. 예를 들어 허가 사항들(6032)은 메모리 위치들 (A), (B), (C), (D) 및 (E)에서 콘텐츠를 읽기, 쓰기 및/또는 실행하기 위한 허가 사항들의 조합에 대한 설정을 포함할 수 있다. 메모리(6003)는 예컨대 애플리케이션(6026)이나 드라이버(6028)에 의해 할당될 수 있는 위치 (A)에 빈 공간(6034)을 포함할 수 있다. 메모리(6003)은 메모리 위치 (E)에서 드라이버 코드 섹션(6036)과 같은 전자 장치(6001)의 개체들이나 함수들의 코드 섹션들을 포함할 수 있다. 메모리(6003)는 메모리 위치 (B)의 비할당 공간(6038)과 같이, 운영체제(6012)에 관한 한 어떤 개체에도 할당되지 않은 메모리의 섹션들이나 범위들을 포함할 수 있다. 메모리(6003)는 메모리 위치 (C)에 있는 비신뢰 드라이버(6040)에 대한 공간과 같이, 안전하거나 악성인 악성 소프트웨어 상태가 알려져 있지 않은 드라이버들을 위한 메모리의 섹션들 또는 범위들을 포함할 수 있다. 메모리(6003)는 비존재 콘텐츠(6042)가 있는 메모리의 섹션들이나 범위들을 포함할 수 있다. 메모리(6003) 내 그러한 비존재 콘텐츠(6042)는 예컨대 디스크로 스왑되었었고 예컨대 스토리지(6044) 상의 하나 이상의 스왑된 파일들(6046) 내 스왑된 콘텐츠(6048) 안에 상주할 수 있는 콘텐츠를 가진 메모리 페이지들을 포함할 수 있다.

[0677] 스토리지(6044)는 메모리(6003) 안으로의 코드 로딩을 위한 액션들과 관련된 하나 이상의 콘텐츠나 위치들을 포함할 수 있다. O/S 하위 보안 에이전트(6020)는 어떤 그러한 콘텐츠나 위치들에 대한 액세스를 트래핑하도록 구성될 수 있다. 예시적 예들로서 다음과 같은 것들이 주어진다. 스토리지(6044)는 스왑된 콘텐츠(6048)가 저장될 수 있는 스왑 파일들(6046)을 포함할 수 있다. 스왑된 콘텐츠(6048)는 사실은 스토리지(6044)에 저장되지만 메모리(6003) 안에 존재하는 것으로 전자 장치(6001)의 개체들에게 보여질 수 있다. 스토리지(6044)는 애플리케이션 이미지(6050)를 저장하는 섹션들이나 어드레스들을 포함할 수 있다. 애플리케이션 이미지(6050)는 애플리케이션, 드라이버, DLL 또는 애플리케이션(6026)이나 드라이버(6028)와 같이 전자 장치(6001) 상에서 실행될 다른 개체를 포함할 수 있다.

[0678] O/S 하위 보안 에이전트(6020) 및/또는 O/S 내 보안 에이전트(6019)는 메모리(6003) 및/또는 스토리지(6004)의 부분들을 악성 소프트웨어가 있는지 검색하도록 구성될 수 있다. 그러한 검색은 메모리(6003) 및/또는 스토리지(6004) 내 콘텐츠가 보안 규칙들(6008, 6021)에 의해 판단 시 악성이라고 판단된 콘텐츠와 동일한지 여부를 판단하기 위해 콘텐츠의 디지털 서명, 체섬, 또는 해시들을 계산하는 동작을 포함할 수 있다. 그러나 어떤 악성 소프트웨어는 실행될 메모리(6003) 안에 스스로나 다른 악성 코드를 삽입함으로써 전자 장치(6001)를 공격하고자 시도할 수 있다. 코드를 메모리(6003) 안에 로딩하려는 시도들을 트래핑함으로써, O/S 하위 보안 에이전트(6020)는 초기에 예컨대 드라이버 파일, 메모리 내 이미지, 또는 코드를 저장하는 다른 규범적 방법 안에 놓이지 않은 코드를 트래핑하거나 검색하거나 확보하도록 구성될 수 있다. 그러한 코드를 초기에 드라이버 파일, 메모리 내 이미지 또는 코드를 저장하는 다른 규범적 방법 안에 위치시키지 않음으로써, 악성 소프트웨어는 O/S 하위 보안 에이전트(6020) 및/또는 O/S 내 보안 에이전트(6019)가 메모리(6003) 및/또는 스토리지(6004)의 부분들을 악성 소프트웨어가 있는지 검색하고자 하는 앞서 기술된 시도들을 피하고자 할 수 있다. 또한 메모리

(6003) 안에 코드를 주입하는 비정상적 방법들은 주입된 콘텐츠나 주입을 시도한 개체가 이미 악성이라고 알려져 있든지 그렇지 않든지 여부와 상관 없이, 그러한 주입을 시도하는 개체가 악성이라는 표시일 수 있다.

[0679] O/S 하위 보안 에이전트(6020)는 운영체제 로더의 사용과 같이, 메모리(6003) 안에 코드를 로딩하기 위한 안전하거나 규범적인 방법들을 결정하도록 구성될 수 있다. 운영체제(6012)에 의해 취해지는 로직이나 단계들이 알려질 수 있도록 그러한 안전하거나 규범적인 방법들이 벤치마킹되거나 매핑될 수 있다. 코드를 메모리(6003) 안에 로딩하려는 시도를 트래핑할 때, O/S 하위 보안 에이전트(6020)는 그러한 시도가 코드 로딩을 위한 알려진 방법들과 매치하는지 여부를 판단할 수 있다. 예를 들어, 이미 할당된 메모리 부분 안으로의 코드 로딩과 관련된 시도 및 메모리로의 직접 쓰기로 운영체제 로더를 우회함으로써 그렇게 하려는 시도의 경우, 해당 시도는 악의적인 것이라고 판단될 수 있다.

[0680] 악성 소프트웨어가 메모리(6003) 안에 프로세서(6002)에 의해 실행 할 코드를 주입하는 것을 막기 위해, O/S 하위 보안 에이전트(6020)는 실행 할 코드를 메모리(6003) 안에 로딩하는 정상적 방법들을 피하는, 코드 주입을 요하는 하나 이상의 시도 단계들을 트래핑하도록 구성될 수 있다. 실행 할 코드를 메모리(6003) 안에 로딩하는 그러한 정상적 방법들은 예컨대 스토리지(6044)로부터 실행될 애플리케이션, 드라이버 또는 다른 개체의 이미지를 읽고, 그러한 이미지로부터 메모리(6003) 안으로 코드를 위치시키고, 그런 다음 그 로딩된 코드를 실행하기 위해 운영체제(6012)의 운영체제 로더를 이용하는 동작을 포함할 수 있다. O/S 하위 보안 에이전트(6020)는 예컨대 운영체제 로더의 사용을 피하고/거나 그러한 단계들 중 어느 하나를 비정상적 방식으로 수행하는 실행 코드를 로드하고자 하는 시도들을 트래핑하도록 구성될 수 있다. 그러한 비정상적 방법들은 보안 규칙들(6008)에 의해 정의될 수 있다. 그러한 비정상적 방법들은 허가된 애플리케이션 이미지(6050) 내 파일로부터 애초에 읽혀지거나 메모리(6003) 내 등가적 애플리케이션 이미지 안에 위치되지 않은 코드에 대한 어떤 로딩 시도를 트래핑하는 것을 포함할 수 있다.

[0681] 일 실시예에서 O/S 하위 보안 에이전트(6020)는 메모리(6003) 안으로 코드 쓰기 시도 및 이어지는 코드의 실행 시도를 트래핑하도록 구성될 수 있다. 예를 들어 O/S 하위 보안 에이전트(6020)는 새로 할당된 공간(6034)일 수 있는 어드레스 (A)에서의 쓰기 시도를 트래핑하도록 구성될 수 있다. 그러한 쓰기 시도 자체가 악성 소프트웨어를 나타내는 것이 아니므로, 해당 시도가 기록되어 추후 참조를 위해 사용될 수 있다. 기입된 콘텐츠가 검색, 기록 또는 평가될 수 있다. O/S 하위 보안 에이전트(6020)는 어드레스 (A)에 기입된 코드에 대한 추후 실행 시도를 트래핑하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 콘텐츠의 실행 시도가 앞서 트래핑된 콘텐츠의 로딩에 뒤따른 것이기 때문에, 콘텐츠의 로딩이 코드를 메모리(6003) 안에 로딩하는 것을 포함한다고 판단하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 시도들의 소스, 시도들의 타깃, 로딩된 콘텐츠, O/S 내 보안 에이전트(6019)로부터의 정보 또는 어떤 다른 적절한 정보와 같은 정황 정보로부터, 코드의 로딩이 악성 소프트웨어를 나타내는지 아닌지 여부를 판단하도록 구성될 수 있다. 예를 들어 코드가 메모리에 로딩되었고 실행이 시도되었다는 것을 검출할 때, O/S 하위 보안 에이전트(6020)는 로딩되었던 콘텐츠를 검색하고 그 코드가 악성 소프트웨어라고 알려져 있는지 여부를 판단하도록 구성될 수 있다. 다른 예에서, 코드의 로딩 및 실행 시도의 검출 시, O/S 하위 보안 에이전트(6020)는 그 코드가 운영체제 로더와 같은 운영체제(6012)의 정상 함수들을 이용하여 로딩되었는지를 판단하도록 구성될 수 있다. 코드가 알려지지 않은 드라이버로부터의 직접 쓰기와 같은 다른 비표준적 방법들을 이용하여 로딩되었으면, O/S 하위 보안 에이전트(6020)는 코드에 대해 시도된 로딩이 의심스럽다고 판단하여 그 코드의 실행을 차단하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)가 메모리(6103) 내 코드의 로딩 및 실행 시도를 트래핑할 수 있게 하고, O/S 하위 보안 에이전트(6020)가 상기 시도들이 의심스럽거나 악성이라고 판단할 수 있게 하는 방법들의 다른 예들이 이하에 기술된다.

[0682] 일 실시예에서 O/S 하위 보안 에이전트(6020)는 스토리지(6044)로부터 애플리케이션 이미지(6050)의 읽기 시도 및 부분(6034)과 같은 메모리(6003)의 새 섹션에 대해 이어지는 콘텐츠 쓰기 시도, 및 이어지는 실행 시도를 트래핑하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 디스크 상에 상주할 때의 이미지와 메모리에 상주할 때의 이미지를 비교하도록 구성될 수 있다. 코드 섹션의 서명들이나 파일 레이아웃과 같은 두 이미지들이 서로 다르면, 그 차이들은 메모리(6003)에 상주할 때 코드가 이미지 안에 주입되었다는 것을 나타낼 수 있다. 그러한 주입은 악성이라 판단될 수 있다. 그러나 O/S 하위 보안 에이전트(6020)가 디스크로부터 애플리케이션 이미지(6050)의 읽기 시도를 트래핑하지 않을 수 있고, 그에 따라 O/S 하위 보안 에이전트(6020)는 그 이미지를 메모리(6003)에 로딩된 것과 비교하지 못할 수 있다.

[0683] 도 61은 주입된 코드가 어떻게 애플리케이션에 의해 수집되어 실행을 위해 메모리 안에 위치하는지에 대한 예시도이다. 애플리케이션(6102)은 도 60의 애플리케이션(6026)이나 드라이버(6028)에 의해 구현될 수 있다. O/S 하위 보안 에이전트(6120)는 도 60의 O/S 하위 보안 에이전트(6020)에 의해 구현될 수 있다. 메모리(6103)는

도 60의 메모리(6003)에 의해 구현될 수 있다. 메모리(6103)는 애플리케이션(6130) 및 드라이버(6132)를 위한 메모리 공간들을 포함할 수 있다. 각각의 메모리 공간(6130, 6132)은 그들 각자의 개체들을 위한 코드 섹션들(6138, 6140)의 공간을 포함할 수 있다. 애플리케이션(6102)은 악성 소프트웨어 서버(6104)로부터 코드를 다운로드할 수 있다. 악성 소프트웨어 서버(6104)는 웹 사이트, 네트워크 상의 다른 전자 장치, 또는 애플리케이션(6102)에 통신 가능하게 연결된 어떤 다른 개체일 수 있다. 애플리케이션(6102)은 그러한 코드를 애플리케이션(6130) 자체의 코드 섹션들(6138, 6140) 안에 기입하거나 드라이버(6132)와 같은 다른 개체 안에 기입할 수 있다. O/S 하위 보안 에이전트(6120)는 이 명세서에서 주어진 다양한 가르침에 따라 그러한 시도를 인터셉트하고 평가하도록 구성될 수 있다. 예를 들어 O/S 하위 보안 에이전트(6120)는 시도된 쓰기 및 실행을 트래핑하고, 쓰기 시도된 콘텐츠를 평가하고, 애플리케이션(6102)의 아이디를 평가하며, 타깃 메모리 위치들을 평가하도록 구성될 수 있다.

[0684] 도 62a 및 62b는 주입된 코드가 메모리(6003) 안에 위치되기 위해 애플리케이션에 의해 어떻게 수집될 수 있는지를 예시한다. 애플리케이션1(6202) 및 애플리케이션2(6204)는 도 60의 애플리케이션(6026)이나 드라이버, 또는 다른 유사 개체들에 의해 구현될 수 있다. O/S 하위 보안 에이전트(6220)는 도 60의 O/S 하위 보안 에이전트(6020)에 의해 구현될 수 있다. 메모리(6203)는 도 60의 메모리(6003)에 의해 구현될 수 있다. 디스크(6244)는 도 60의 스토리지(6044)에 의해 구현될 수 있다.

[0685] 도 62a는 애플리케이션1과 같은 애플리케이션의 이미지(6252)를 디스크(6244)로부터 메모리(6203)로 로딩하는 것에 대한 예시도를 도시한다. 운영체제 로더는 애플리케이션1(6250)의 디스크 이미지(6250)를 읽고, 그것을 메모리(6203) 내 애플리케이션1의 이미지(6234)에 기입하도록 구성될 수 있다. 애플리케이션1의 디스크 이미지(6250)는 복사될 수 있는 암호화된 콘텐츠(6252)를 포함하며, 메모리(6203) 내 애플리케이션1의 이미지(6234) 안에 남을 수 있다. O/S 하위 보안 에이전트(6220)는 운영체제 로더에 의해 호출된 읽기 및 쓰기 시도들을 트래핑할 수 있다. 그러나 일 실시예에서, O/S 하위 보안 에이전트(6220)는 두 이미지들이 다르지 않을 것이고, 운영체제의 정상적 메커니즘들에 의해 액션이 수행되었기 때문에 어떤 악성 활동들이 이미지 로딩 시 수반된다고 의심할 이유가 없을 수 있다.

[0686] 도 62b는 애플리케이션의 이미지가 메모리 안에 로딩된 후 수행되는 가능한 액션들에 대한 예시도를 도시한다. 애플리케이션2(6204)는 전자 장치 상에서 동작할 수 있다. 애플리케이션1(6202)은 전자 장치 상에서 동작할 수 있고, 암호화된 콘텐츠(6236)를 해독하라는 명령어를 보낼 수 있다. 그러한 명령은 암호화된 콘텐츠(6236)가 해독되고 해독된 콘텐츠(6238a)가 애플리케이션1의 이미지(6234)의 메모리 공간 안에 기입되고/거나 애플리케이션2의 이미지(6240)의 메모리 공간 안에 해독된 콘텐츠(6238b)로서 기입되는 결과를 가져올 수 있다. 따라서 애플리케이션1(6202)는 그 자신의 코드 섹션이나 다른 개체의 코드 섹션 안에 코드를 주입하도록 시도할 수 있다. 애플리케이션1(6202)은 해독된 콘텐츠(6238a)를 제자리에서 이전 암호화 콘텐츠(6236) 위에 기입하거나 애플리케이션1의 이미지(6234)의 다른 메모리 부분 안에 기입할 수 있다. 해독된 콘텐츠(6238)로부터의 코드는 악성일 수 있다. 그에 따라 O/S 하위 보안 에이전트(6220)는 이 명세서에서 주어진 다양한 가르침에 따라 해독된 콘텐츠(6238)와 같은 코드에 대해 시도되는 쓰기 및 실행을 트래핑하고 평가하도록 구성될 수 있다. 예를 들어 O/S 하위 보안 에이전트(6220)는 시도된 쓰기 및 실행을 인터셉트하고, 쓰기 시도된 콘텐츠를 평가하고, 애플리케이션(6102)의 아이디를 평가하며, 타깃 메모리 위치들을 평가하도록 구성될 수 있다.

[0687] 도 60으로 돌아가면, O/S 하위 보안 에이전트(6020)는 전자 장치(6001)의 알려진 개체들의 위치를 포함하여, 메모리(6003)의 부분들의 레이아웃을 판단할 수 있다. 일 실시예에서, 그러한 레이아웃은 운영체제(6012)의 커널 이미지들 및 그 구성요소들을 포함할 때뿐 아니라 안전 드라이버들(및 그들의 코드와 데이터 섹션들) 및 다른 드라이버들(및 그들의 코드와 데이터 섹션들)을 포함할 때 커널 가상 메모리의 레이아웃을 포함할 수 있다. 그러한 레이아웃은 메모리 맵(6010)에 저장될 수 있다. O/S 하위 보안 에이전트(6020)는 운영체제(6012)의 시동 및 동작을 프로파일링하거나, 운영체제(6012) 및 안전 드라이버들의 구성요소들을 디지털 인증서 검색이나 검증을 통해 검증하거나, 어떤 다른 적절한 방식들에 의해, 그러한 레이아웃을 보안 규칙들(6008)이나 보호 서버로부터 판단할 수 있다. O/S 하위 보안 에이전트(6020)는 또한 메모리(6003)의 어느 부분들이 전자 장치(6001)의 개체들 중 어느 하나에 할당되지 않는지를 판단할 수도 있다. 커널 메모리 내 어떤 주어진 페이지에 대한 관련 물리적 메모리 어드레스들을 기술하는 물리적 메모리로의 커널 메모리 매핑이 메모리 맵(6010)에 포함될 수 있다. 메모리 맵(6010)은 또한 메모리의 어떤 부분들이 할당되었거나 할당되지 않았는지, 혹은 어느 부분들이 비존재 콘텐츠나 스왑 파일들과 연관되는지에 대한 설명들을 포함할 수 있다. 운영체제(6012)는 페이지 프레임 넘버 데이터베이스("Pfn")와 같은 데이터베이스 안에 운영체제(6012)에 의해 관리되는 가상 페이지들을 기술하는 정보를 가질 수 있다. 그러한 데이터베이스의 순회가 할당되고, 매핑되고 비존재 콘텐츠를 포함하는 등의

메모리 페이지들을 판단하는 데 사용될 수 있다.

[0688] 메모리(6003)의 레이아웃이 알려지면, O/S 하위 보안 에이전트(6020)는 비할당 영역들에서의 메모리 할당 및 실행을 감시하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 그렇게 시도된 동작들을 트래핑하고 기입될 콘텐츠를 검색하도록 구성될 수 있다. 그러한 동작들의 단계들에는 개체가 허가 사항들이 인에블된 상태로 메모리(6003)의 부분들을 할당하고, 새로운 콘텐츠를 메모리(6003)의 부분들에 기입하고, 실행을 가능하게 하기 위해 메모리(6003)의 부분에 대한 허가 사항들을 변경하고, 실행을 위해 메모리(6003)의 부분을 호출하는 것이 포함될 수 있다. O/S 하위 보안 에이전트(6020)는 그 단계들이 악성 소프트웨어를 나타내는지 여부를 판단하기 전에 그러한 단계들을 트래핑하여 실행 시도를 중지하도록 구성될 수 있다. 예를 들어 O/S 하위 보안 에이전트(6020)는 위치 (A)에서 메모리(6003)의 비할당 부분(6034)의 할당 시도, 그 부분(6034)으로의 콘텐츠 쓰기 시도, 위치 (A)의 허가 사항을 "쓰기(기입)"에서 "쓰기/실행"으로 변경하도록 페이지 테이블 디렉토리(6030)의 허가 사항들(6032)에 쓰기, 및 위치 (A)에서 허가 사항의 실행 시도를 트래핑하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 아래의 윈도우즈 함수들과 같이 메모리를 할당하기 위한 어떤 적절한 함수를 트래핑하도록 구성될 수 있다:

[0689] - MmAllocateContiguousMemory

[0690] - MmAllocateContiguousMemorySpecifyCache

[0691] - MmAllocateSpecialPool

[0692] - MmAllocateContiguousMemorySpecifyCacheNode

[0693] - MmAllocatelndependentPages

[0694] - MmAllocateMappingAddress

[0695] - MmAllocateNonCachedMemory

[0696] - MmAllocatePagesForMdl 또는

[0697] - MmAllocatePagesForMdlEx

[0698] O/S 하위 보안 에이전트(6020)는 메모리의 할당 시도를 트래핑하기 위해 상기 함수들에 해당하는 메모리(6003) 내 페이지들의 실행이나 메모리(6003) 내 물리적 어드레스들의 실행을 트래핑하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 함수들의 호출자를 판단하고 알려진 악성 소프트웨어에 대해 그 호출자의 콘텐츠를 검색한다. 만일, 예컨대 할당 시도가 알려진 안전한 애플리케이션 함수의 서브 함수를 통해 이뤄졌으면, 그 시도는 악성 소프트웨어가 해당 함수와 관련된 보안을 피하고자 시도한다는 것을 나타낼 수 있다. O/S 하위 보안 에이전트(6020)는 추후 참조하기 위해 할당된 메모리 부분을 메모리 맵(6010)에 추가하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 호출자와 같은 정황 정보와 함께, 시도된 메모리 할당을 기록하도록 구성될 수 있다.

[0699] O/S 하위 보안 에이전트(6020)는 코드를 메모리에 쓰기 위한 어떤 적절한 함수나 방법을 트래핑하도록 구성될 수 있다. 애플리케이션(6026)이나 드라이버(6028)과 같은 개체들은 코드를 메모리(6003)에 입력하기 위해 운영체제(6012)에 의해 제공되는 함수들이나 방법들을 사용할 수 있다. 그러나 이 개체들이 검출을 피하려는 시도로 메모리에 직접 기입할 수 있다. 따라서 O/S 하위 보안 에이전트(6020)는 메모리(6003)에 코드를 입력하려는 시도를 트래핑하며, 이때 그 시도가 그러한 지정된 함수들을 사용하지 않고 이루어졌으면 O/S 하위 보안 에이전트(6020)는 코드를 메모리(6003)에 로딩하려는 시도가 의심스럽다고 판단하도록 구성될 수 있다.

[0700] O/S 하위 보안 에이전트(6020)는 메모리 위치에 대한 허가 사항을 변경하기 위한 어떤 적절한 함수나 방법을 트래핑하도록 구성될 수 있다. 예를 들어 메모리 페이지 허가 사항을 변경하기 위해, 이하의 윈도우즈 함수들 중 하나가 호출될 수 있다.

[0701] - MmProtectMdlSystemAddress

[0702] - NtProtectVirtualMemory

[0703] - MiGetPageProtection

[0704] - MiQueryAddressState

[0705] - MiSetProtectionOnSection

[0706] - MiGetpageProtection

[0707] - MiProtectPrivateMemory

[0708] 따라서 O/S 하위 보안 에이전트(6020)는 메모리 내 그러한 함수들의 위치의 실행에 대해 감시함으로써 그 함수들 중 어느 하나의 실행을 트래핑하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 페이지 테이블 디렉토리(6030) 및 그 안에 포함된 허가 사항들(6032)에 대해 시도되는 어떤 쓰기를 트래핑하도록 구성될 수 있다. 허가 사항들(6032)을 변경하는 데 허가된 함수(위에서 나열된 것들과 같은 함수)의 실행이 트래핑되지 않는데, 허가 사항들(6032)에 대해 시도된 쓰기가 트래핑되면, O/S 하위 보안 에이전트(6020)는 그 쓰기 시도가 악성 소프트웨어를 가리킨다고 판단하도록 구성될 수 있다. 그러한 시도는 메모리 위치에 대한 허가 사항들의 페이지 테이블 디렉토리 내 어느 엔트리를 악성 소프트웨어가 직접 편집한(가령, "읽기"에서 "쓰기"나 "실행"으로) 결과일 수 있고, 그에 따라 운영체제(6012) 함수들과 관련된 보안을 야기하지 않고 해당 위치가 악성 콘텐츠로 기입되거나 실행되게 될 수 있다.

[0709] 메모리로의 쓰기 시도 뒤에 실행 시도가 트래핑되고, 그 시도가 위치 (B)에서의 비할당 메모리 부분(6038)과 같은 메모리의 비할당 부분들에 대해 이루어졌다고 O/S 하위 보안 에이전트(6020)가 판단하는 경우, O/S 하위 보안 에이전트(6020)는 그 시도가 악성이라고 판단하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 콘텐츠 쓰기 및 실행 시도가 위치 (B)의 비할당 부분(6038)과 같은 비할당 공간에 대해 이루어졌는지 여부를 판단하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 메모리 맵(6010)을 액세스하거나 앞서 트래핑된 할당 함수들의 실행을 참조함으로써 그러한 판단을 내리도록 구성될 수 있다. 일 실시예에서, 시도된 쓰기나 실행이 할당되지 않은 것으로 판단된 메모리(6003)에서 이루어진다면, 악성 소프트웨어가 운영체제(6012)에 의해 정의된 루틴들의 사용 없이 메모리를 할당하였을 것이다. 그러한 실시예에서 O/S 하위 보안 에이전트(6020)는 그 시도가 실행할 코드를 메모리(6003)로 로딩하는 악의적 시도라고 판단할 수 있다. 다른 실시예에서, 쓰기 및 실행 시도가 메모리 맵(6010)에 보여진 바와 같이 전자 장치(6001)의 어느 개체에 이미 할당된 메모리(6003)에서 이루어진 것이면, O/S 하위 보안 에이전트(6020)는 명령어의 호출자가 다른 개체로의 코드 쓰기에 대해 권리가 있는지 혹은 허가되는지 여부를 검사하도록 구성될 수 있다. 그런 권리가 없다면, 그러한 시도는 악성 소프트웨어가 다른 함수에 속하는 메모리(6003)의 부분 안에 악의적일 수 있는 코드를 주입하려고 시도하는 것임을 나타낼 수 있다.

[0710] O/S 하위 보안 에이전트(6020)는 위치 (B)의 부분(6038)과 같이 할당되지 않은 것으로 지정된 메모리(6003) 부분들에서의 어떤 쓰기, 읽기 또는 실행 시도, 또는 허가 사항들(6032) 중 미할당 부분들(6038)에 대한 허가 사항들을 변경하려는 시도들을 트래핑하도록 구성될 수 있다. 운영체제(6012)의 허가된 개체들에 의해 호출되는 허가된 할당 함수들이 아닌 개체들로부터 일어난 그러한 시도들은 호출자의 악성 소프트웨어 상태가 알려지지 않은 것인지 여부와 무관하게 악성이라고 판단될 수 있다. 메모리 위치가 쓰기 시도를 경험하는 경우, 그 시도는 코드가 메모리에 로딩되는 표시라고 판단될 수 있다. 메모리 위치가 실행 시도를 경험하는 경우, 그 시도는 새로 로딩되는 코드가 실행되는 표시라고 판단될 수 있다.

[0711] O/S 하위 보안 에이전트(6020)는 비신뢰 드라이버(6040)에 의한 어떤 쓰기나 실행 시도들, 또는 비신뢰 드라이버(6040)와 관련된 메모리 위치 (C)에 대한 허가 사항들(6032)의 변경을 트래핑하도록 구성될 수 있다. 비신뢰 드라이버(6040)는 악성 소프트웨어 상태가 알려지지 않은 드라이버를 포함할 수 있다. 비신뢰 드라이버(6040)의 서명이나 해시에 대해 블랙리스트나 화이트리스트에 아무 엔트리들도 존재 하지 않을 수 있다. 비신뢰 드라이버(6040)가 시스템(6000)이나 유사 시스템들에 의해 경험되지 않은 안전한 새 드라이버일 수 있고, 아니면 비신뢰 드라이버(6040)가 시스템(6000)에 의해 경험되지 않은 악성 소프트웨어의 어떤 치환일 수 있다. 비신뢰 드라이버(6040)의 쓰기나 실행 시도가 트래핑되면, O/S 하위 보안 에이전트(6020)는 그러한 트래핑된 동작들을 평가하기 위해 추가 보안 규칙들(6008)을 적용하도록 구성될 수 있다. 특히, 코드 쓰기 및 이어지는 코드 실행 시도들이 트래핑될 수 있고, 그러한 동작들이 운영체제(6012)의 커널 모듈들이나 알려진 드라이버들과 같은 알려진 구성요소들에 대해 시도되는지 여부를 판단하기 위해 O/S 하위 보안 에이전트(6020)에 의해 자세히 검사될 수 있다. 알려진 구성요소들에 대해 시도되는 경우, 그러한 시도들은 차단될 수 있고, 비신뢰 드라이버(6040)가 악성 소프트웨어라고 판단될 수 있다.

[0712] O/S 하위 보안 에이전트(6020)는 어드레스 (D)의 부분(6042)과 같이 비존재 콘텐츠를 가진 메모리(6003)의 부분들에 대한 쓰기나 실행 시도를 트래핑하도록 구성될 수 있다. 메모리(6003)의 부분(6042)은 가상 메모리 할당 맥락에서 할당될 수 있지만 부분(6042)의 콘텐츠는 메모리(6003)의 물리적 메모리 안에 실제로 존재하지 않을

수 있다. 대신 그 부분(6042)의 콘텐츠가 스토리지(6044) 안과 같은 어느 다른 곳에 존재할 수 있다. 비존재 콘텐츠는 스토리지(6044) 안에서 페이지 스왑 동작의 일부로서 스왑된 콘텐츠(6048)를 포함하는 스왑 파일 안에 상주할 수 있으며, 여기서 가상 메모리(6003)의 콘텐츠는 메모리(6003)의 물리적 메모리 안에 다른 요소들을 위한 공간을 만들어 주기 위해 디스크로 옮겨진다. 운영체제(6012)는 그러한 스와핑 동작들을 수행하도록 구성될 수 있고, 필요 시 물리적 메모리 안에 그 스왑된 콘텐츠(6048)를 재로딩하도록 구성될 수 있다. 그에 따라 부분(6042)이 비존재 콘텐츠를 포함하더라도, 부분(6042)에 대한 쓰기나 실행 시도들이 O/S 하위 보안 에이전트(6020)에 의해 트래핑될 수 있다. 그러한 쓰기 시도가 페이지 스왑을 수행하는 운영체제(6012)가 아닌 다른 개체로부터 발생했다면, 그 쓰기는 악성일 수 있다. 예를 들어 운영체제(6012)의 가상 메모리 관리자로부터 발생되지 않은 부분(6042)으로의 쓰기 시도들은 거부될 수 있다. 또한, 콘텐츠가 존재하지 않는데 그 부분(6042)에 대한 실행 시도가 이루어지는 경우, 그 실행 시도는 악성일 수 있다.

[0713] 또한 스토리지(6044) 상의 스왑된 파일(6046)의 스왑된 콘텐츠(6048)로의 쓰기와 같은 액세스 시도가 O/S 하위 보안 에이전트(6020)에 의해 트래핑될 수 있다. 그러한 액세스 시도가 허가된 개체가 아닌 다른 개체로부터 발생했다면, 그 쓰기는 악성일 수 있다. 예를 들어 운영체제(6012)의 가상 메모리 관리자로부터 발생되지 않은 스왑된 콘텐츠(6048)로의 쓰기 시도들은 거부될 수 있고 악성 소프트웨어를 가리킨다고 판단될 수 있다. 그러한 트래핑은 예컨대 디스크 쓰기 함수들을 트래핑하거나 스토리지(6044) 내 입/출력 명령들을 트래핑함으로써 이루어질 수 있다. 스토리지(6044) 내 입/출력 명령들의 트래핑은 예컨대 스토리지(6044) 상에서 실행되는 펌웨어 보안 에이전트에 의해 이루어질 수 있다.

[0714] 도 63은 코드를 주입하기 위해 스왑된 콘텐츠에 대해 이루어지는 악성 공격들의 추가적 예를 도시한다. 커널 가상 메모리(6304)는 가상화되었을 때의 메모리(6003)의 부분들을 나타낼 수 있다. 커널 가상 메모리(6304)의 콘텐츠가 그들이 상주하는 위치들에 매핑될 수 있다. 그러한 콘텐츠는 예컨대 물리적 메모리(6302) 및/또는 디스크(6344)로 매핑될 수 있다. 물리적 메모리(6302)는 전자 장치(6001) 안에 물리적으로 상주할 때의 메모리(6003)의 물리적 레이아웃을 나타낼 수 있다. 디스크(6344)는 도 60의 스토리지(6044)에 의해 구현될 수 있다. 커널 가상 메모리(6304)의 어떤 부분들(6306, 6310, 6314, 6318)은 할당되지 않고 가상 메모리의 사용자가 사용할 수 없다. 커널 가상 메모리(6304)의 다른 부분들은 운영체제 커널(6308) 및 드라이버1(6312), 드라이버2(6316) 및 드라이버3(6320)과 같은 드라이버들을 위한 부분들을 포함할 수 있다. 커널 가상 메모리(6304) 안에서 할당된 메모리의 부분들은 물리적 메모리(6302) 및/또는 디스크(6344)의 다양한 비인접 부분들로 매핑할 수 있다. 드라이버2(6320)에 포함되는 페이지들의 예가 페이지들(6322, 6324 및 6326)을 포함하는 것으로 보여진다. 드라이버3 페이지 0에 대응하는 페이지(6326)가 디스크(6344) 상의 스왑 파일(6350)로 매핑될 수 있다. 드라이버3 페이지 1에 대응하는 페이지(6324)는 어드레스 (A)에 있고 계속해서 어드레스 (B)로 이어지는 물리적 메모리(6302)의 어드레스로 매핑될 수 있다. 드라이버3 페이지 2에 대응하는 페이지(6322)가 디스크(6344) 내 스왑 파일(6348)로 매핑될 수 있다.

[0715] 스왑 파일 동작이 수행될 때, "XYZ"일 수 있는 드라이버3 페이지 0의 콘텐츠가 스왑 파일(6350)으로 기입될 수 있다. 드라이버 3 페이지 0의 콘텐츠는 그에 따라 존재하지 않게 될 수 있다. 한편 악성 소프트웨어(6352)는 스왑 파일(6350)의 콘텐츠를 "PDQ"로 재기입할 수 있다. 따라서 스왑 파일 동작이 거꾸로 되어 콘텐츠가 디스크(6344) 및 스왑 파일(6350)로부터 판독될 때, 새 코드가 커널 가상 메모리(6304) 안에 로딩될 것이다. 또한 가능한 것이 값들을 물리적 메모리(6302)로 페이지 6324와 같은 다른 페이지들에 대해 바로 기입하려는 악성 소프트웨어(6352)의 액션일 수 있다.

[0716] 도 60으로 돌아가면, O/S 하위 보안 에이전트(6020)는 존재하지 않는 콘텐츠(6042)를 가진 페이지들과 관련된 스왑 파일들(6046) 안으로 코드를 로딩하는 것을 막도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 메모리(6003)로부터 스왑 파일들(6046)로 혹은 그 반대로 시도된 쓰기 동작들을 트래핑하고, 콘텐츠의 스냅샷, 서명, 암호화 해시, 체섬 또는 다른 표지를 판단할 수 있다. O/S 하위 보안 에이전트(6020)는 스토리지(6044) 상의 위치들에 대응하는 플래그들을 세팅하거나 메모리 페이지들을 스왑하기 위한 함수들이나 루틴들 상에서 그러한 시도를 트래핑하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 비존재 콘텐츠(6042)에 대해 스왑 파일(6046)로부터 시도된 읽기 동작들 및/또는 메모리 부분들로 다시 쓰기 시도를 트래핑하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 이러한 작업들을 달성하기 위해 시도된 스왑 함수들의 실행을 트래핑하도록 구성될 수 있다. 그러한 함수들은 예컨대 IoPageRead(), IoAsynchronousPageWrite() 또는 IoAsynchronousPageWrite()를 포함하거나 호출할 수 있다. O/S 하위 보안 에이전트(6020)는 스왑 파일(6046)로부터 읽혀지고/거나 비존재 콘텐츠(6042)에 쓰여지는 콘텐츠의 스냅샷, 서명, 암호화 해시, 체섬, 또는 다른 표지를 판단하고 그 콘텐츠들이 변경되었는지 여부를 판단하도록 구성될 수 있다. 변경된 경우, O/S 하위 보안

에이전트(6020)는 스왑 파일(6046)의 콘텐츠에 코드가 주입되었다고 판단하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 새로운 콘텐츠의 실행을 차단하거나 다른 교정 조치들을 수행하도록 구성될 수 있다.

[0717] O/S 하위 보안 에이전트(6020)는 실행 허가를 주기 위해 콘텐츠의 메모리 위치에 대한 허가 사항들(6032)을 변경하려는 후속 시도를 트래핑함으로써, 메모리(6003)로의 특정 쓰기가 코드의 로딩이라고 판단하도록 구성될 수 있다. 또한 O/S 하위 보안 에이전트(6020)는 메모리(6003)에 쓰고 이어서 허가 사항(6032)에 대한 쓰기를 시도하는 개체가 해당 메모리가 실행될 수 있게 하는 것이라고 판단하도록 구성될 수 있다. 그러한 개체는 통상적으로 운영체제 로더와 같이 알려진 신뢰 개체일 수 있다. 따라서 O/S 하위 보안 에이전트(6020)는 메모리(6003) 안으로의 코드 로딩 시도가 운영체제 로더와 같이 알려진 신뢰 개체에 의해 발생되었는지 여부를 판단하도록 구성될 수 있다. 그런 신뢰 개체에 의해 발생되지 않은 경우, O/S 하위 보안 에이전트(6020)는 그러한 시도를 거부하고 해당 시도 및/또는 개체가 악성 소프트웨어를 가리킨다고 판단하도록 구성될 수 있다.

[0718] O/S 하위 보안 에이전트(6020)는 메모리(6003)로부터 합법적 함수 코드를 복사하고, 그 함수 코드를 새로 할당된 부분(6034)과 같은 새 위치에 복사하며, 그런 다음 그 복사된 코드를 실행하려는 시도를 트래핑하도록 구성될 수 있다. 그러한 시도는 예컨대 시스템 함수를 소유하는 운영체제(6012)나 드라이버로부터 허가를 얻을 필요 없이 그 시스템 함수를 실행하려는 악성 소프트웨어에 의한 시도일 수 있다.

[0719] 시도된 동작을 트래핑할 때, O/S 하위 보안 에이전트(6020)는 O/S 내 보안 에이전트(6019)로부터의 정황 정보, 드라이버들의 호출 스택들 및/또는 메모리 맵(6010)에 기반하여 그 동작의 호출자를 식별하도록 구성될 수 있다. 동향 규칙들, 화이트리스트나 블랙리스트를 포함하는 보안 규칙들(6008)을 이용하여, O/S 하위 보안 에이전트(6020)는 호출자가 악성이라고 알려진 것인지 여부를 판단하도록 구성될 수 있다. 그러한 악성 상태가 알려진 것이 아니면, O/S 하위 보안 에이전트(6020)는 쓰기 동작이 계속되게 허가하도록 구성될 수 있다. 그러나 O/S 하위 보안 에이전트(6020)는 이어지는 실행 시도를 중지하도록 구성될 수 있다. 어떤 동작들은 주입된 코드를 완전히 로딩하기 위해 여러 번의 쓰기 시도들을 수반할 수 있다. O/S 하위 보안 에이전트(6020)는 주입된 코드를 충분히 판단하여 특징짓기 위해 실행 중지 전에 그러한 쓰기들을 허용할 수 있다.

[0720] 메모리(6003)가 할당되거나 할당 해제되거나 신뢰 또는 비신뢰 개체들에 의해 기입될 때, O/S 하위 보안 에이전트(6020)는 메모리 맵(6010)에 그러한 정보를 기록하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 실행 허가 없이 코드 주입 공격에 민감한 메모리(6003)의 부분들을 마크할 수 있다. 그러한 메모리(6003) 부분들에 대한 실행 시도는 트래핑 동작을 일으킬 수 있다. O/S 하위 보안 에이전트(6020)는 시도된 실행과 관련된 메모리 맵(6010) 내 메모리 부분을 찾고 관련 개체를 판단하도록 구성될 수 있다. O/S 하위 보안 에이전트(6020)는 알려진 악성 소프트웨어 표시에 대해 코드 바이트들을 검색하거나 코드 쓰기 시 관찰된 가능한 악성 동향을 평가할 수 있다.

[0721] 코드 로딩 시도 및 이어지는 실행 시도가 의심스럽다거나 악성 소프트웨어를 나타내는 것으로 판단되면, O/S 하위 보안 에이전트(6020)는 적절한 교정 액션을 수행하도록 구성될 수 있다. 일 실시예에서 O/S 하위 보안 에이전트(6020)는 기입 코드에 "NOOP" 명령어들이나 다른 패턴들과 같은 더미 정보를 채우도록 구성될 수 있다. 또한 해당 시도를 수행한 개체의 메모리 위치들에 더미 정보가 채워질 수 있다. 다른 실시예에서 O/S 하위 보안 에이전트(6020)는 실행을 해당 시도들의 소스 및 타깃을 청소하고 검역하는 교정 엔진으로 넘기도록 구성될 수 있다. 또 다른 실시예에서 O/S 하위 보안 에이전트(6020)는 호출자와 타깃의 메모리 위치들을 읽기, 쓰기 또는 실행을 불허하는 허가 사항들(6032)로 마크하도록 구성될 수 있다. 마찬가지로 O/S 하위 보안 에이전트(6020)는 그러한 시도들의 호출자나 타깃의 메모리 위치들에 대한 어떤 후속 읽기, 쓰기 또는 실행 시도들을 트래핑하도록 구성될 수 있다. 그 시도들과 관련된 메모리 위치들은 메모리 맵(6010)은 악성으로 마크될 수 있다.

[0722] 도 64는 메모리의 한 부분이 악성이라고 판단된 뒤 도 60의 메모리 맵(6010)과 같은 메모리 맵(6400)의 실시예이다. 메모리 맵(6400)의 부분들(6402, 6406, 6410, 6414, 6418, 6422)이 할당되지 않은 것으로 보여질 수 있다. 메모리 맵은 운영체제 커널(6404), 드라이버1(6408), 드라이버2(6412) 및 드라이버3(6416)의 위치들을 보일 수 있다. 코드를 주입하고 실행하려는 악성 시도와 관련된 것으로 판단된 메모리의 부분은 악성 소프트웨어(6420)라고 지명될 수 있다. 악성 소프트웨어 섹션(6420)에 대한 읽기, 쓰기 및 실행을 거부하는 허가 사항들이 세팅될 수 있다. 악성 소프트웨어 섹션(6420)은 예컨대 코드가 주입되었던 메모리의 부분들이나 주입 시도가 이루어졌었던 메모리 부분들을 반영할 수 있다. 이어서, 알려지지 않은 섹션(6424)로부터 악성 소프트웨어 섹션(6420)에 대해 시도되는 액세스가 도 60의 O/S 하위 보안 에이전트(6020)와 같은 O/S 하위 보안 에이전트에 의해 트래핑될 수 있다. 그러한 O/S 하위 보안 에이전트는 악성 소프트웨어 섹션(6420)에 대응하는 메모리(6003)의 부분들에 대한 액세스를 트래핑하기 위해 플래그들을 세팅하도록 구성될 수 있다. 예를 들어, 섹션

(6420)에 대해 앞서 기입된 코드를 읽으려는 시도가 알려지지 않은 섹션(6424)에 의해 이루어질 수 있다. 다른 예에서, 섹션(6420)에 대해 앞서 기입된 코드를 실행하려는 시도가 알려지지 않은 섹션(6424)에 의해 이루어질 수 있다. O/S 하위 보안 에이전트는 악성 소프트웨어로 지명된 메모리에 대한 어떤 그러한 액세스 시도는 자체가 악성이라고 판단하도록 구성될 수 있다. 따라서 O/S 하위 보안 에이전트는 그 시도를 차단하고 이전에 알려지지 않은 섹션(6424)을 악성 소프트웨어 섹션으로 재지정할 수 있다. O/S 하위 보안 에이전트는 새로 지정된 악성 소프트웨어 섹션(6424)에 대해 그것을 더미 데이터로 채우고, 그것을 청소 및 검역하기 위한 프로세스로 보내고, 그 악성 소프트웨어 섹션(6424)를 읽기, 쓰기 또는 실행 액세스를 거부하라는 허가 사항들로 마크하고/거나 악성 소프트웨어 섹션(6424)에 대한 후속 액세스를 트래핑하는 것과 같은 교정 액션들을 수행하도록 구성될 수 있다.

[0723] 도 65는 전자 장치에서 메모리 내 코드 로딩 및 실행에 대한 운영체제 하위 트래핑을 위한 방법(6500)의 실시예이다. 단계 6505에서 메모리 내 코드 로딩 및 실행과 관련된 자원들을 판단하기 위해 보안 규칙들이 액세스될 수 있다. 그러한 보안 규칙들은 자원들, 및 자원에 대해 시도된 액세스가 트래핑되고 평가되게 할 기준을 식별할 수 있다.

[0724] 단계 6510에서 전자 장치 안의 운영체제들의 레벨 아래의 제어 구조 안에서 플래그들이 세팅될 수 있다. 플래그들은 예컨대 코드의 주입 및 후속 실행 시도를 트래핑하기 위해 세팅될 수 있다. 플래그들은 상술한 시도들에 대응하는 메모리 페이지들을 통한 가상 메모리 액세스 및/또는 메모리 어드레스들을 통한 물리적 메모리 액세스에 대해 세팅될 수 있다.

[0725] 단계 6515에서 전자 장치가 메모리 안으로의 코드 주입과 관련된 자원들을 액세스하려는 시도들의 트래핑을 위해 감시될 수 있다. 단계 6520에서 어떤 시도들도 트래핑되지 않았으면, 프로세스(6500)는 트래핑된 시도들에 대해 계속 감시하도록 단계 6515로 진행할 수 있다. 시도가 트래핑되었으면, 그 시도는 단계 6525에서 시작하여 처리될 수 있다. 그러한 처리는 전자 장치의 운영체제들의 레벨 아래에서 수행될 수 있다. 단계 6525에서, 그 시도가 악성인지 여부를 분석하기 위해 유용한 정보가 수집될 수 있다. 예를 들어, 그 시도의 타깃과 함께, 그 시도를 행하는 프로세스, 애플리케이션, 드라이버 또는 루틴이 판단될 수 있다. 전자 장치의 운영체제 안으로부터의 정황 정보가 O/S 내 보안 에이전트로부터 획득될 수 있다. 코드를 주입하려는 시도가 이루어졌으면, 호출자의 이미지가 검색될 수 있다.

[0726] 단계 6530에서, 로딩 또는 주입 시도를 행한 개체가 그러한 시도를 수행하는 것이 허가되지 않았다고 알려진 것인지 여부가 판단될 수 있다. 허가되지 않았다고 알려진 경우, 단계 6565에서 해당 시도는 거부되고 어떤 적절한 교정 액션이 수행될 수 있다. 그렇지 않은 경우는 개체의 악성 소프트웨어 상태가 아직 알려지지 않을 수 있는 경우이며, 단계 6535에서 로딩 시도가 허용될 수 있고, 이어지는 실행 시도의 상황에 따라 그 로딩 시도가 잠정적으로 의심스러운 것인지 여부가 판단될 수 있다. 로딩 시도가 잠정적으로 의심스럽지 않은 경우, 이 방법은 전자 장치 감시를 계속하기 위해 단계 6515로 진행할 수 있다.

[0727] 로딩 시도가 여전히 잠정적으로 의심스러운 경우, 단계 6545에서 전자 장치는 기입되었던 코드에 대한 후속 실행 시도에 대해 감시될 수 있다. 어떤 시도들도 트래핑되지 않았으면, 프로세스(6500)는 단계 6545를 반복하거나 트래핑된 시도들에 대해 계속 감시하도록 나란히 단계 6515로 진행할 수 있다. 실행 시도가 트래핑되었으면, 그 시도는 단계 6550에서 시작하여 처리될 수 있다. 그러한 처리는 전자 장치의 운영체제들의 레벨 아래에서 수행될 수 있다. 단계 6550에서, 로딩 시도와 함께 그 실행 시도가 악성인지 여부를 분석하기 위해 유용한 정보가 수집될 수 있다. 예를 들어, 그 시도의 타깃과 함께, 그 시도를 행하는 프로세스, 애플리케이션, 드라이버 또는 루틴이 판단될 수 있다. 전자 장치의 운영체제 안으로부터의 정황 정보가 O/S 내 보안 에이전트로부터 획득될 수 있다. 실행 시도의 호출자 이미지가 악성 소프트웨어 표시에 대해 검색될 수 있다.

[0728] 단계 6555에서 시도된 로딩과 함께 시도된 실행이 악성 소프트웨어를 나타내는지 그렇지 않은지 여부가 판단될 수 있다. 악성 소프트웨어를 가리키는 경우, 단계 6565에서 해당 시도는 거부되고 어떤 적절한 교정 액션이 수행될 수 있다. 악성 소프트웨어를 가리키지 않은 경우, 단계 6560에서 실행 및 로딩 시도들은 하용될 수 있고, 방법(6500)은 옵션으로서 전자 장치를 계속 감시하도록 단계 6515로 이어질 수 있다.

[0729] 방법들(300, 600, 800, 1100, 1400, 1900, 2400, 2500, 2800, 3000, 3200, 3400, 3600, 3700, 4000, 4200, 4300, 4400, 4600, 4900, 5300, 5700, 5900, 6500)은 도 1-2, 4-5, 7, 9-10, 12-13, IS18, 20-23, 26-27, 29, 31, 33, 35, 38-39, 41, 45, 47-48, 50-52, 54-56, 58 또는 60-64의 시스템들 중 어느 하나나 방법들(300, 600, 800, 1100, 1400, 1900, 2400, 2500, 2800, 3000, 3200, 3400, 3600, 3700, 4000, 4200, 4300, 4400, 4600, 4900, 5300, 5700, 5900, 6500)을 구현하기 위해 작동되는 어떤 다른 시스템을 이용하여 구현될 수

있다. 그로써, 방법들(300, 600, 800, 1100, 1400, 1900, 2400, 2500, 2800, 3000, 3200, 3400, 3600, 3700, 4000, 4200, 4300, 4400, 4600, 4900, 5300, 5700, 5900, 6500)에 대한 바람직한 초기화 지점 및 이들 각자의 단계들의 순서는 선택된 구현 예에 달려있을 수 있다. 도 3, 6, 8, 11, 14, 19, 24-25, 28, 30, 32, 34, 36-37, 40, 42-44, 46, 49, 53, 57, 59 및 65가 방법들의 예들(300, 600, 800, 1100, 1400, 1900, 2400, 2500, 2800, 3000, 3200, 3400, 3600, 3700, 4000, 4200, 4300, 4400, 4600, 4900, 5300, 5700, 5900 및 6500)과 관련하여 취해질 특정 수의 단계들을 개시하고 있지만, 그 방법들은 그 도면들에 도시된 것보다 더 많거나 더 적은 단계들로 실행될 수 있다. 또한 도 3, 6, 8, 11, 14, 19, 24-25, 28, 30, 32, 34, 36-37, 40, 42-44, 46, 49, 53, 57, 59 및 65는 그 방법들에 관해 취해질 단계들의 어떤 순서를 개시하며 그러한 방법들을 이루는 단계들은 어떤 적절한 순서로 이행될 수 있다. 또한 방법들(300, 600, 800, 1100, 1400, 1900, 2400, 2500, 2800, 3000, 3200, 3400, 3600, 3700, 4000, 4200, 4300, 4400, 4600, 4900, 5300, 5700, 5900 및 6500)의 일부 또는 모든 단계들은 다른 방법들의 단계들과 결합될 수 있다. 일부 실시예들에서 어떤 단계들은 옵션으로서 생략되거나, 반복되거나 결합될 수 있다. 일부 실시예들에서 방법들(300, 600, 800, 1100, 1400, 1900, 2400, 2500, 2800, 3000, 3200, 3400, 3600, 3700, 4000, 4200, 4300, 4400, 4600, 4900, 5300, 5700, 5900 및 6500) 중 하나 이상의 어떤 단계들은 서로의 다른 단계들과 나란히 실행될 수 있다. 소정 실시예들에서, 방법들(300, 600, 800, 1100, 1400, 1900, 2400, 2500, 2800, 3000, 3200, 3400, 3600, 3700, 4000, 4200, 4300, 4400, 4600, 4900, 5300, 5700, 5900 및 6500)은 컴퓨터 판독 가능 매체에서 실시되는 소프트웨어를 통해 부분적으로나 전체적으로 구현될 수 있다.

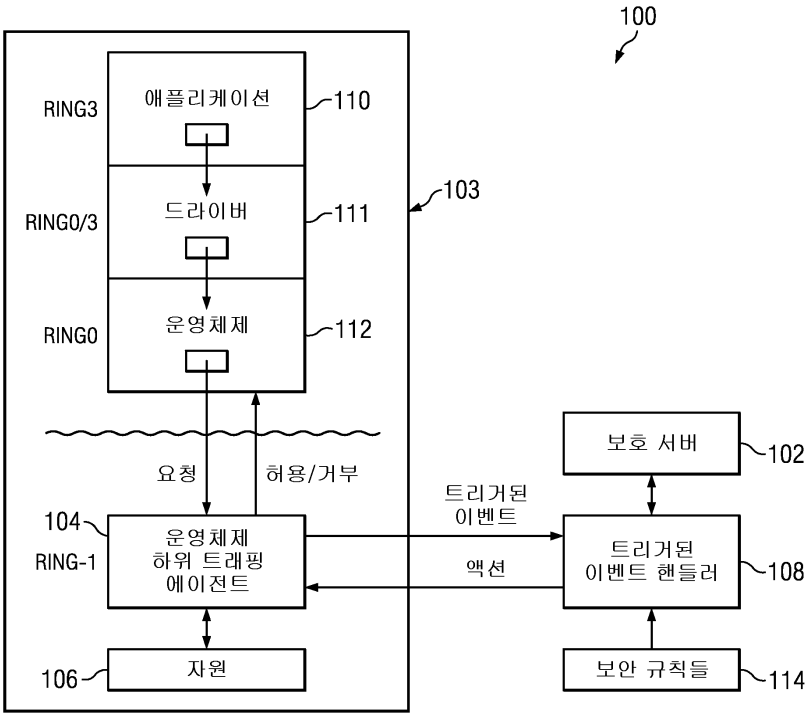
[0730] 본 개시의 목적상, 컴퓨터 판독 가능 매체는 일정 기간 동안 데이터 및/또는 명령어들을 보유할 수 있는 어떤 수단이나 수단들의 집합을 포함할 수 있다. 컴퓨터 판독 가능 매체는 제한 없이 직접 액세스 스토리지 장치(가령, 하드 디스크 드라이브나 플로피 디스크), 순차적 액세스 스토리지 장치(가령, 테이프 디스크 드라이브), 컴팩트 디스크, CD-ROM, DVD, 램(RAM), 롬(ROM), EEPROM(electrically erasable programmable read-only memory) 및/또는 플래시 메모리와 같은 스토리지 매체; 비임시 통신 매체; 및/또는 이들의 조합을 포함할 수 있다.

[0731] 도 1-2, 4-5, 7, 9-10, 12-13, 15-18, 20-23, 26-27, 29, 31, 33, 35, 38-39, 41, 45, 47-48, 50-52, 54-56, 58 또는 60-64의 시스템들 하나 이상은 같은 시스템들의 다른 부분들과 결합될 수 있다.

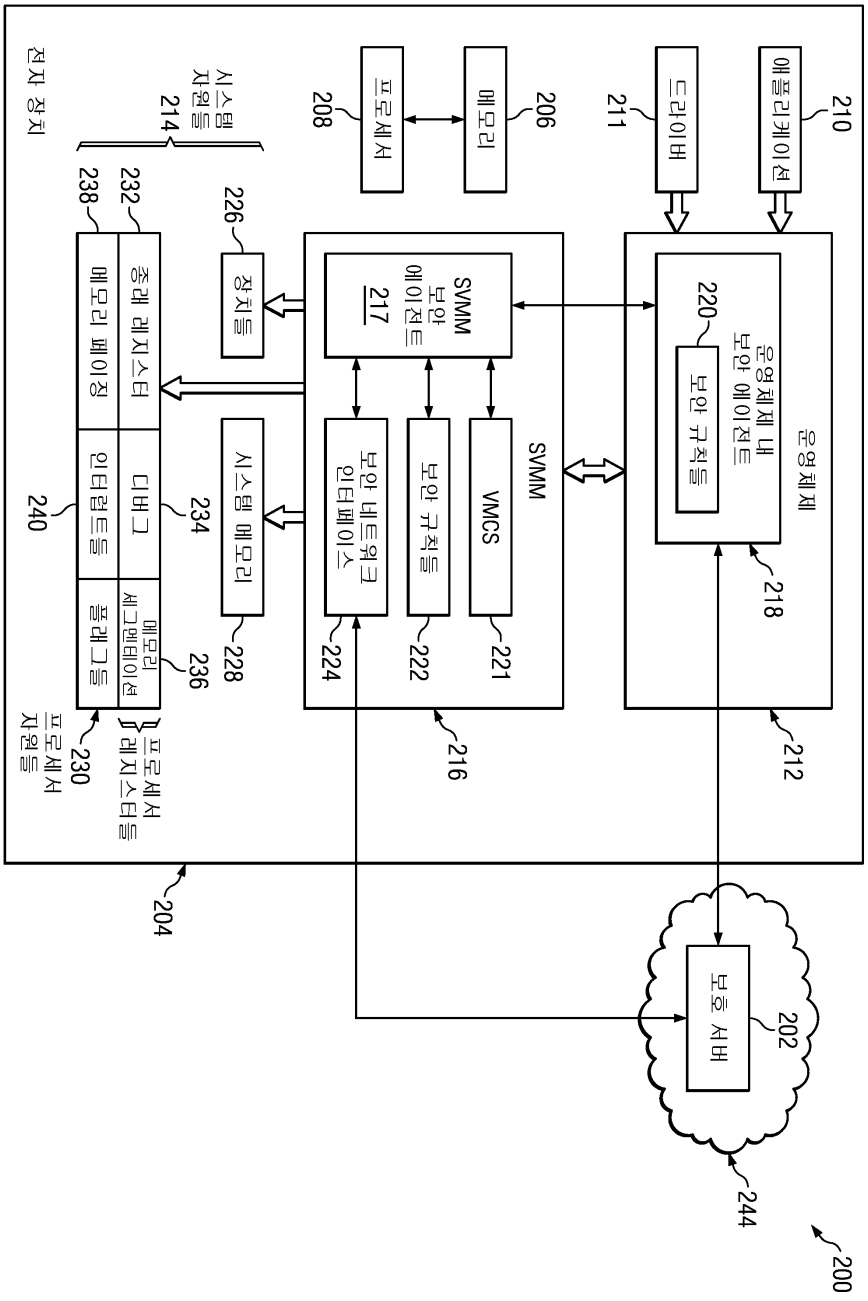
[0732] 본 개시가 상세히 기술되었지만, 첨부된 청구범위에서 정의되는 본 개시의 사상과 범위로부터 벗어나지 않으면서 이에 대해 다양한 변경, 대체, 및 치환이 이루어질 수 있다는 것을 알아야 한다.

도면

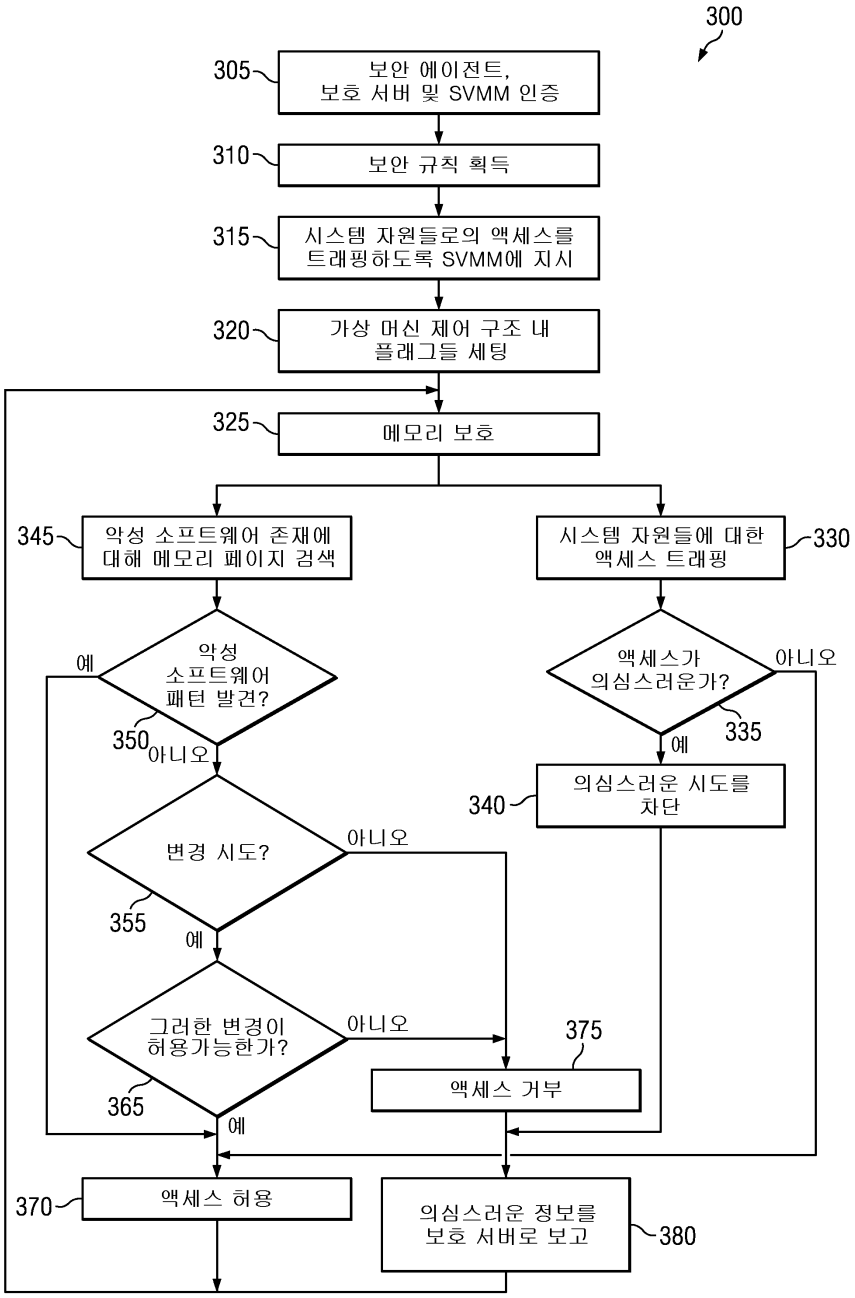
도면1


100
RING3
애플리케이션
110
RING0/3
드라이버
111
103
RING0
운영체제
112
요청
허용/거부
104
RING-1
운영체제
하위 트래핑
에이전트
106
자원
트리거된
이벤트
액션
보호 서버
102
트리거된
이벤트 핸들러
108
보안 규칙들
114

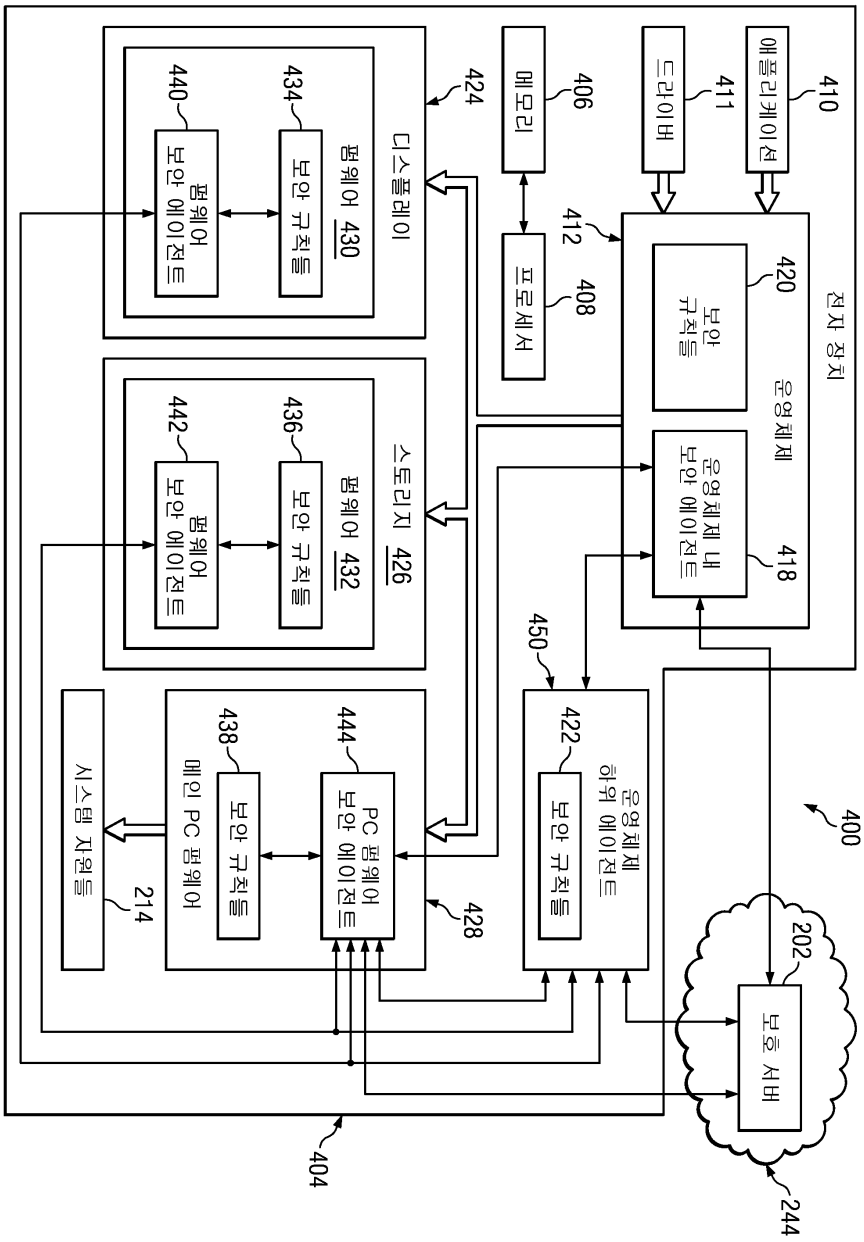
도면2


전자 장치
204
애플리케이션
210
드라이버
211
운영체제
212
운영체제 내
보안 에이전트
218
보안 규칙들
220
SVMM
216
SVMM
보안
에이전트
217
VMCS
221
보안 규칙들
222
보안 네트워크
인터페이스
224
장치들
226
시스템 메모리
228
프로세서
208
메모리
206
시스템
자원들
214
보호 서버
202
244
200
중재 레지스터
232
디버그
234
메모리
세그멘테이션
236
메모리 페이징
238
인터럽트들
240
플래그들
프로세서
자원들
230
프로세서
레지스터들

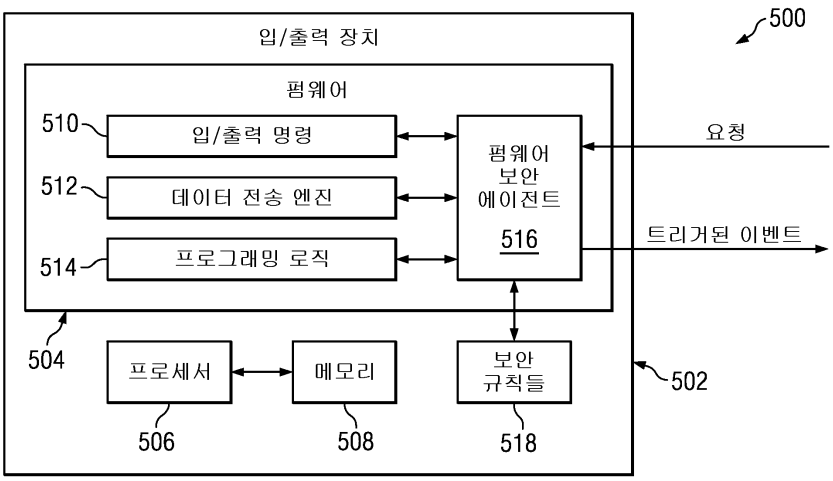
도면3


300
305
보안 에이전트,
보호 서버 및 SVMM 인증
310
보안 규칙 획득
315
시스템 자원들로의 액세스를
트래핑하도록 SVMM에 지시
320
가상 머신 제어 구조 내
플래그들 세팅
325
메모리 보호
345
악성 소프트웨어 존재에
대해 메모리 페이지 검색
330
시스템 자원들에 대한
액세스 트래핑
예
악성
소프트웨어
패턴 발견?
350
아니오
액세스가
의심스러운가?
아니오
335
예
340
의심스러운 시도를
차단
변경 시도?
아니오
355
예
그러한 변경이
허용가능한가?
아니오
375
365
액세스 거부
예
370
액세스 허용
의심스러운 정보를
보호 서버로 보고
380

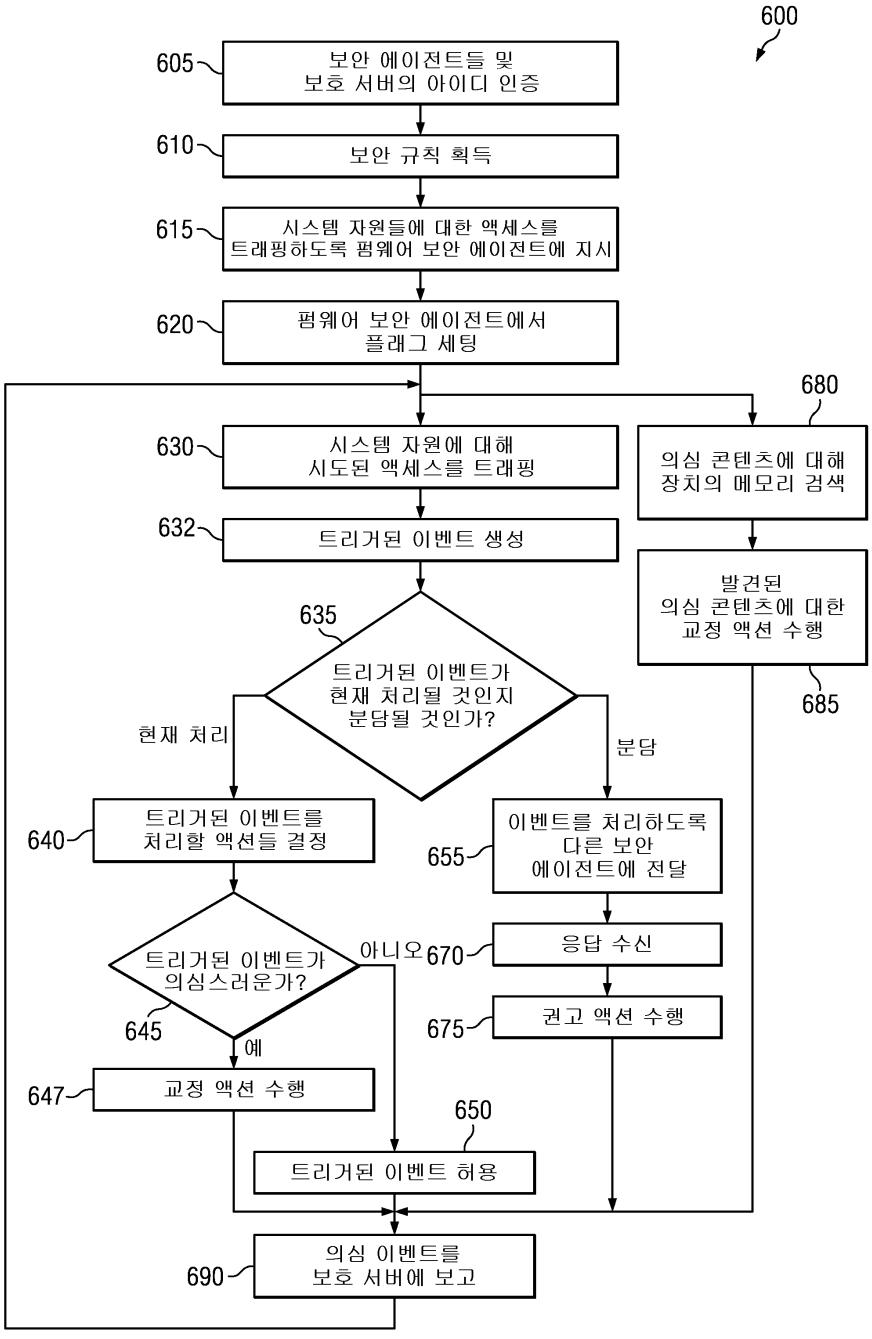
도면4


전자 장치
400
애플리케이션
410
드라이버
411
메모리
406
프로세서
408
412
운영체제
420
보안 규칙들
운영체제 내 보안 에이전트
418
디스플레이
424
펌웨어 430
보안 규칙들
434
펌웨어 보안 에이전트
440
스토리지 426
펌웨어 432
보안 규칙들
436
펌웨어 보안 에이전트
442
운영체제 하위 에이전트
450
보안 규칙들
422
428
메인 PC 펌웨어
보안 규칙들
438
PC 펌웨어 보안 에이전트
444
시스템 자원들
214
404
보호 서버
202
244

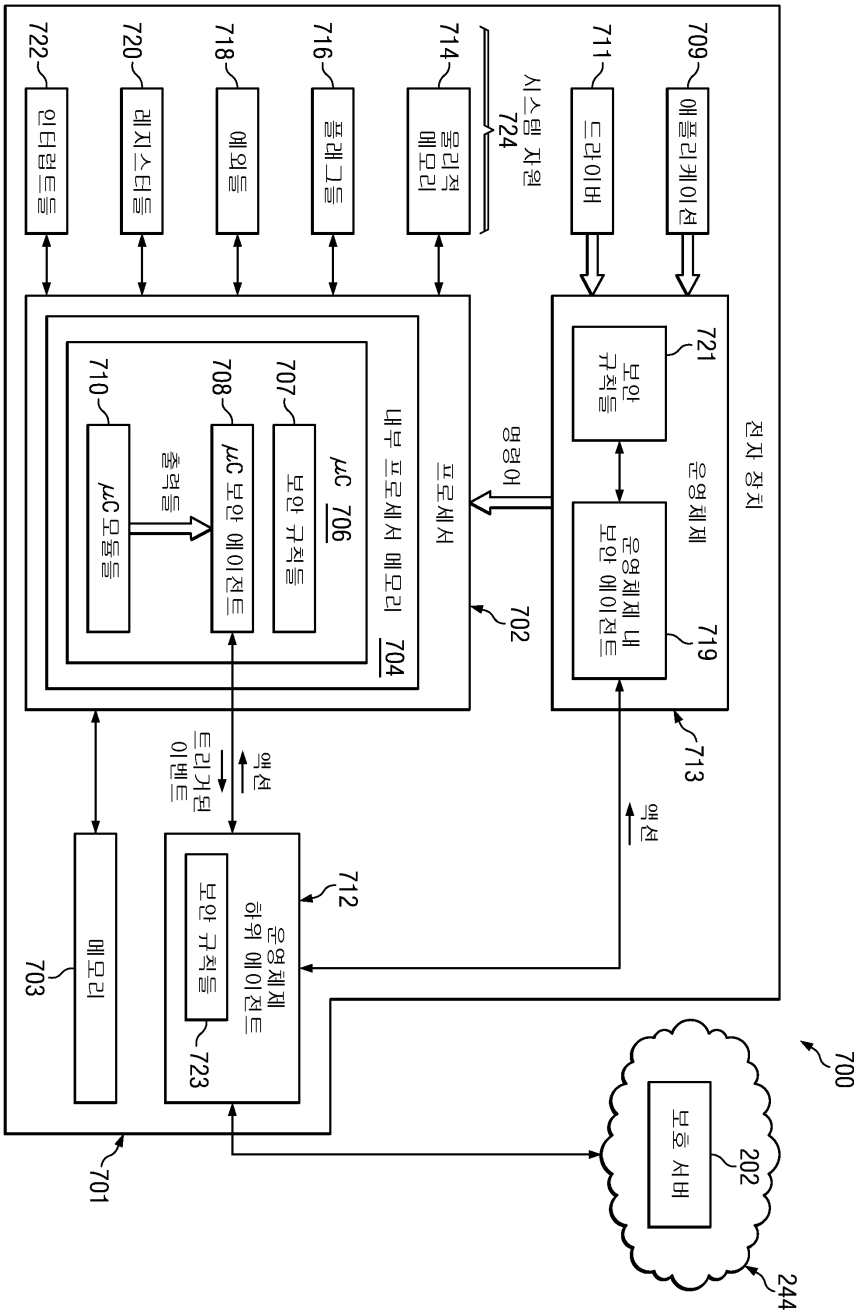
도면5


500
입/출력 장치
펌웨어
510
입/출력 명령
512
데이터 전송 엔진
514
프로그래밍 로직
펌웨어 보안 에이전트
516
요청
트리거된 이벤트
504
프로세서
506
메모리
508
보안 규칙들
518
502

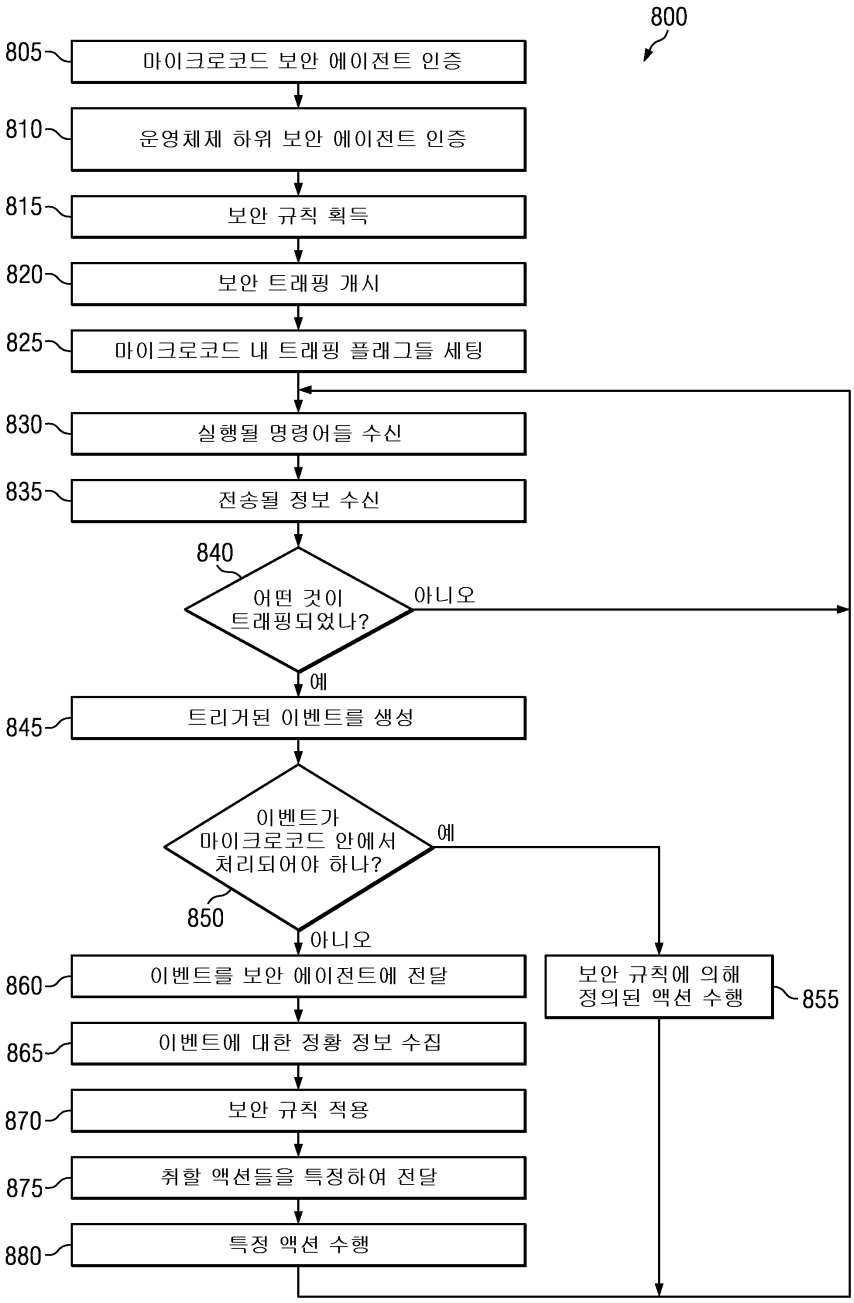
도면6


600
605
보안 에이전트들 및
보호 서버의 아이디 인증
610
보안 규칙 획득
615
시스템 자원들에 대한 액세스를
트래핑하도록 펌웨어 보안 에이전트에 지시
620
펌웨어 보안 에이전트에서
플래그 세팅
630
시스템 자원에 대해
시도된 액세스를 트래핑
632
트리거된 이벤트 생성
635
트리거된 이벤트가
현재 처리될 것인지
분담될 것인가?
현재 처리
분담
640
트리거된 이벤트를
처리할 액션들 결정
645
트리거된 이벤트가
의심스러운가?
아니오
예
647
교정 액션 수행
650
트리거된 이벤트 허용
655
이벤트를 처리하도록
다른 보안
에이전트에 전달
670
응답 수신
675
권고 액션 수행
680
의심 콘텐츠에 대해
장치의 메모리 검색
685
발견된
의심 콘텐츠에 대한
교정 액션 수행
690
의심 이벤트를
보호 서버에 보고

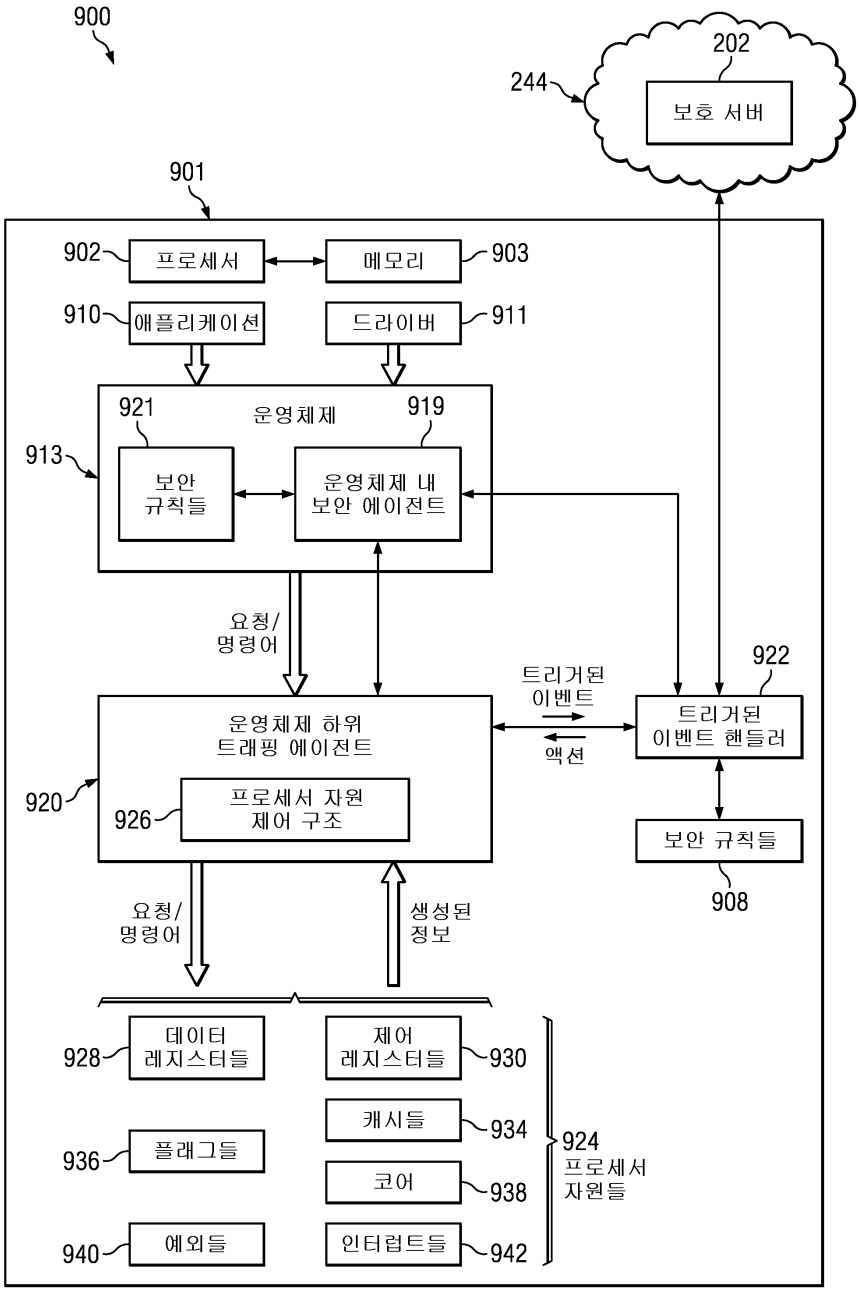
도면7


700
전자 장치
709
애플리케이션
711
드라이버
714
물리적 메모리
716
플래그들
718
예외들
720
레지스터들
722
인터럽트들
724
시스템 자원
721
보안 규칙들
운영체제
719
운영체제 내 보안 에이전트
713
명령어
702
프로세서
704
내부 프로세서 메모리
706
μC 보안 에이전트
707
보안 규칙들
708
710
μC 모듈들
출력들
액션
트리거된 이벤트
712
운영체제 하위 에이전트
723
보안 규칙들
703
메모리
701
보호 서버
202
244

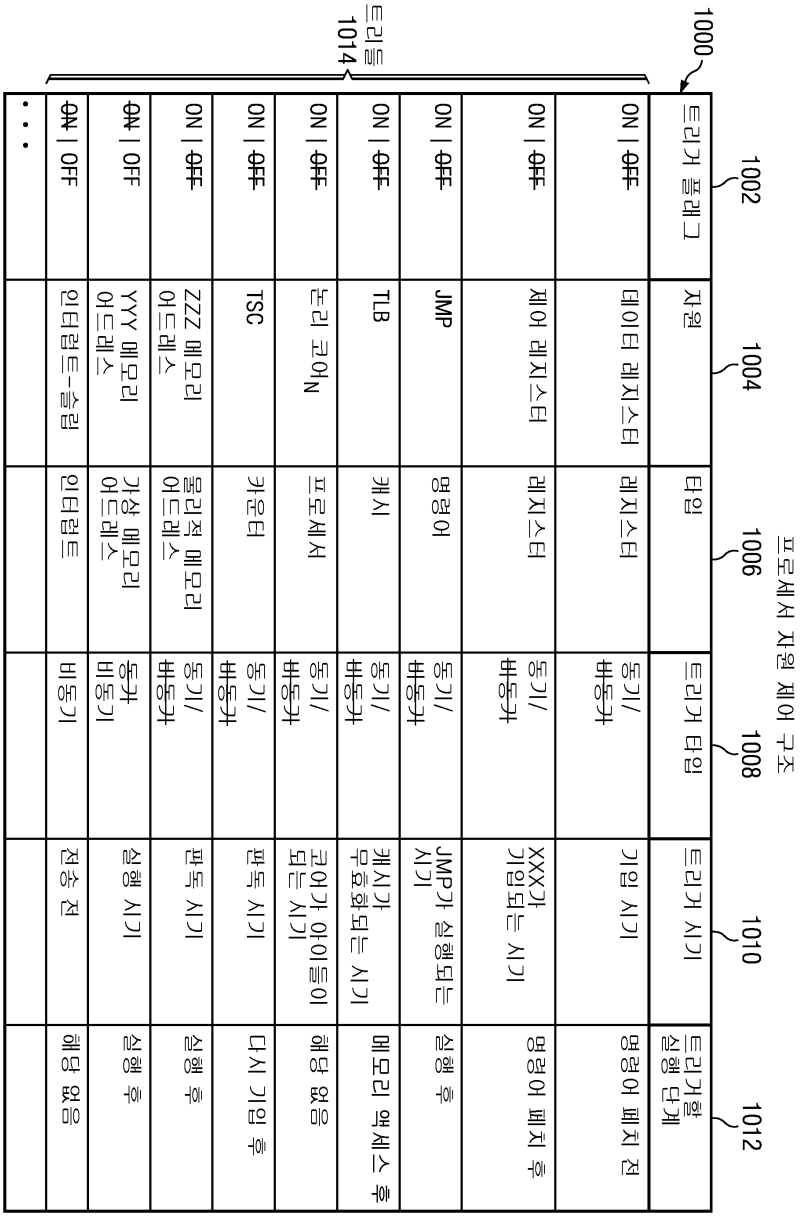
도면8


800
805
마이크로코드 보안 에이전트 인증
810
운영체제 하위 보안 에이전트 인증
815
보안 규칙 획득
820
보안 트래핑 개시
825
마이크로코드 내 트래핑 플래그들 세팅
830
실행될 명령어들 수신
835
전송될 정보 수신
840
어떤 것이 트래핑되었나?
아니오
예
845
트리거된 이벤트를 생성
이벤트가 마이크로코드 안에서 처리되어야 하나?
850
예
아니오
860
이벤트를 보안 에이전트에 전달
865
이벤트에 대한 정황 정보 수집
870
보안 규칙 적용
875
취할 액션들을 특정하여 전달
880
특정 액션 수행
보안 규칙에 의해 정의된 액션 수행
855

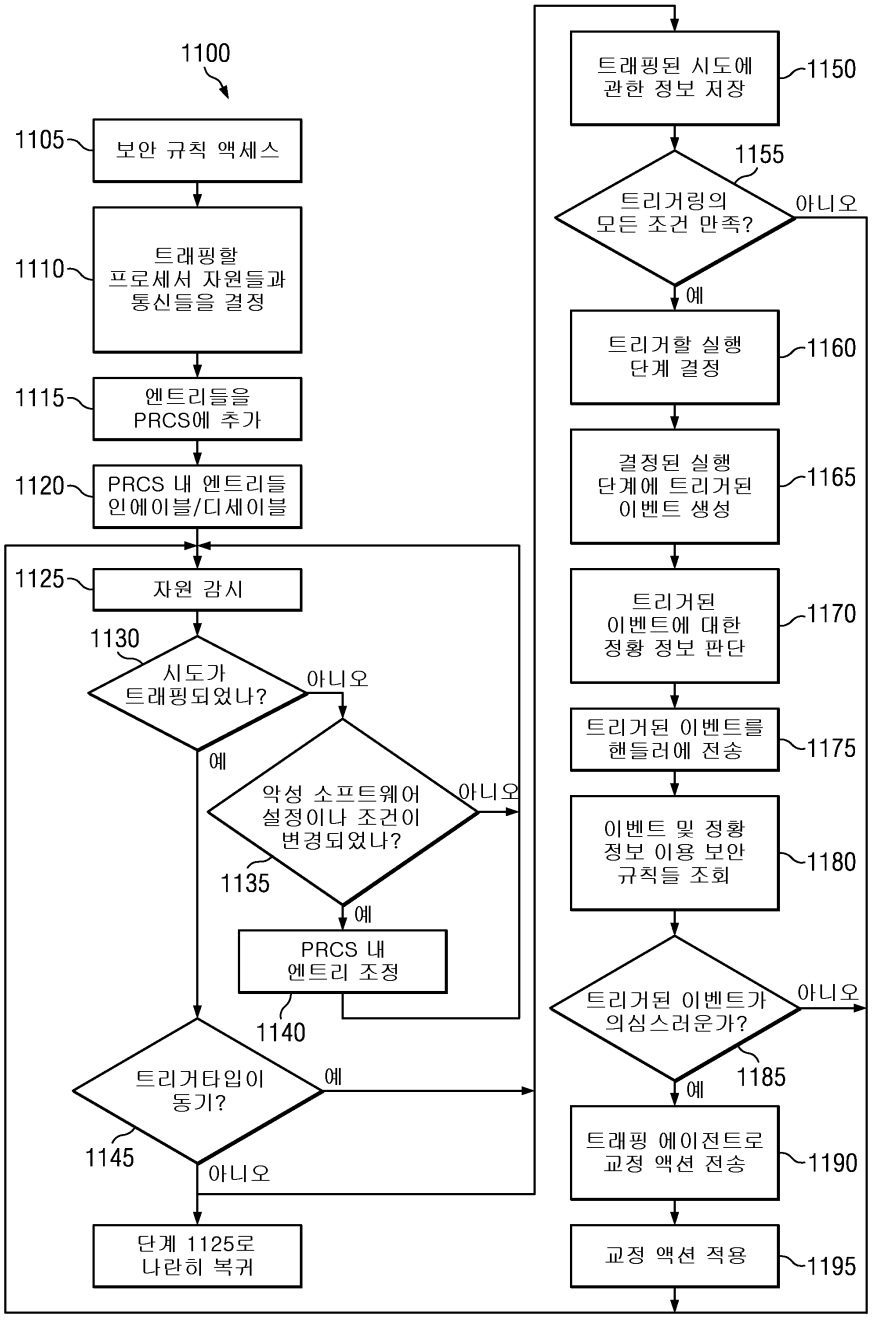
도면9


900
202
244
보호 서버
901
902
프로세서
메모리
903
910
애플리케이션
드라이버
911
921
운영체제
919
913
보안 규칙들
운영체제 내 보안 에이전트
요청/명령어
922
트리거된 이벤트
액션
운영체제 하위 트래핑 에이전트
트리거된 이벤트 핸들러
920
926
프로세서 자원 제어 구조
보안 규칙들
908
요청/명령어
생성된 정보
데이터 레지스터들
928
제어 레지스터들
930
캐시들
934
924
프로세서 자원들
936
플래그들
코어
938
940
예외들
인터럽트들
942

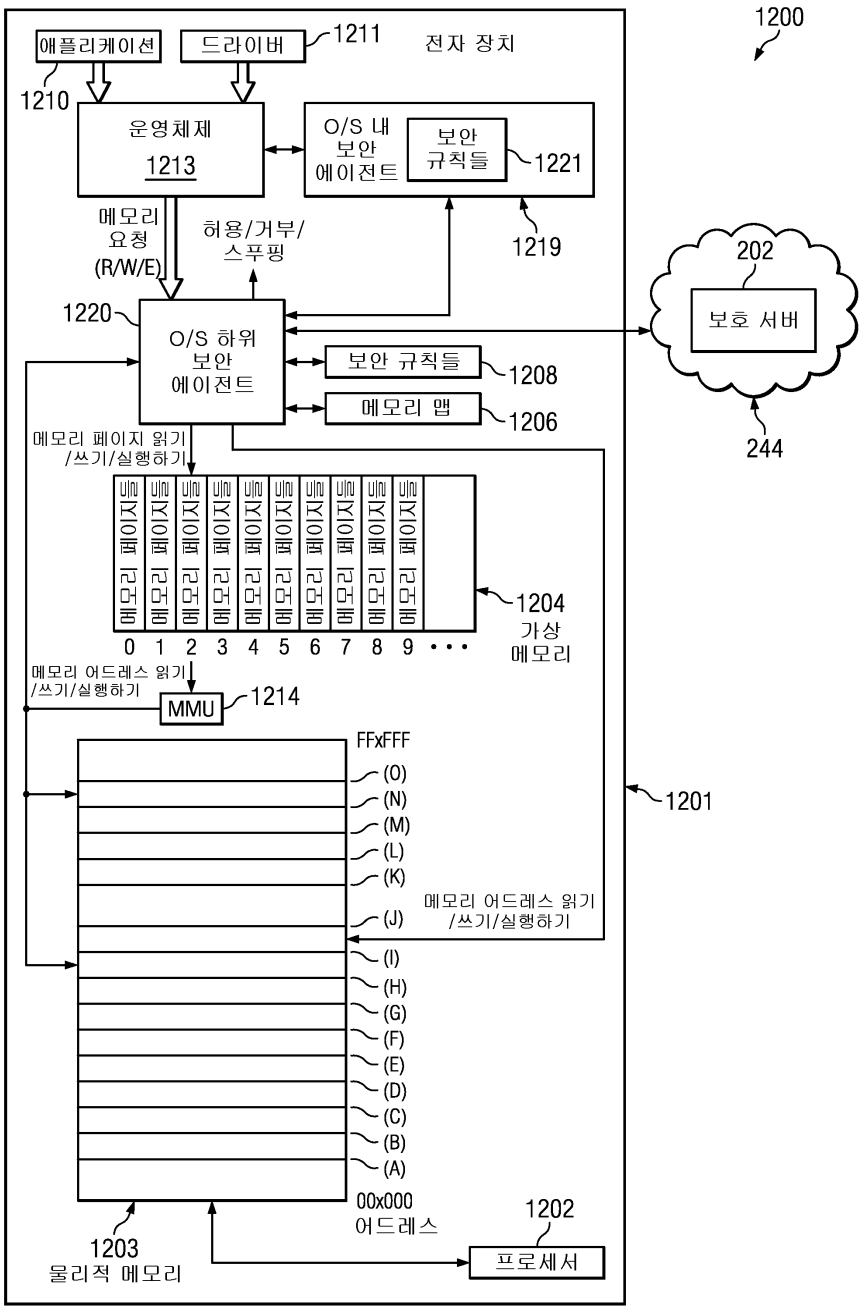
도면10


프로세서 자원 제어 구조
1000
엔트리들
1014
1002
1004
1006
1008
1010
1012
트리거 플래그
자원
타입
트리거 타입
트리거 시기
트리거할 실행 단계
ON | OFF
데이터 레지스터
레지스터
동기/ 비동기
기입 시기
명령어 페치 전
ON | OFF
제어 레지스터
레지스터
동기/ 비동기
XXX가 기입되는 시기
명령어 페치 후
ON | OFF
JMP
명령어
동기/ 비동기
JMP가 실행되는 시기
실행 후
ON | OFF
TLB
캐시
동기/ 비동기
캐시가 무효화되는 시기
메모리 액세스 후
ON | OFF
논리 코어N
프로세서
동기/ 비동기
코어가 아이들이 되는 시기
해당 없음
ON | OFF
TSC
카운터
동기/ 비동기
판독 시기
다시 기입 후
ON | OFF
ZZZ 메모리 어드레스
물리적 메모리 어드레스
동기/ 비동기
판독 시기
실행 후
ON | OFF
YYY 메모리 어드레스
가상 메모리 어드레스
동기 비동기
실행 시기
실행 후
ON | OFF
인터럽트-슬립
인터럽트
비동기
전송 전
해당 없음
• • •

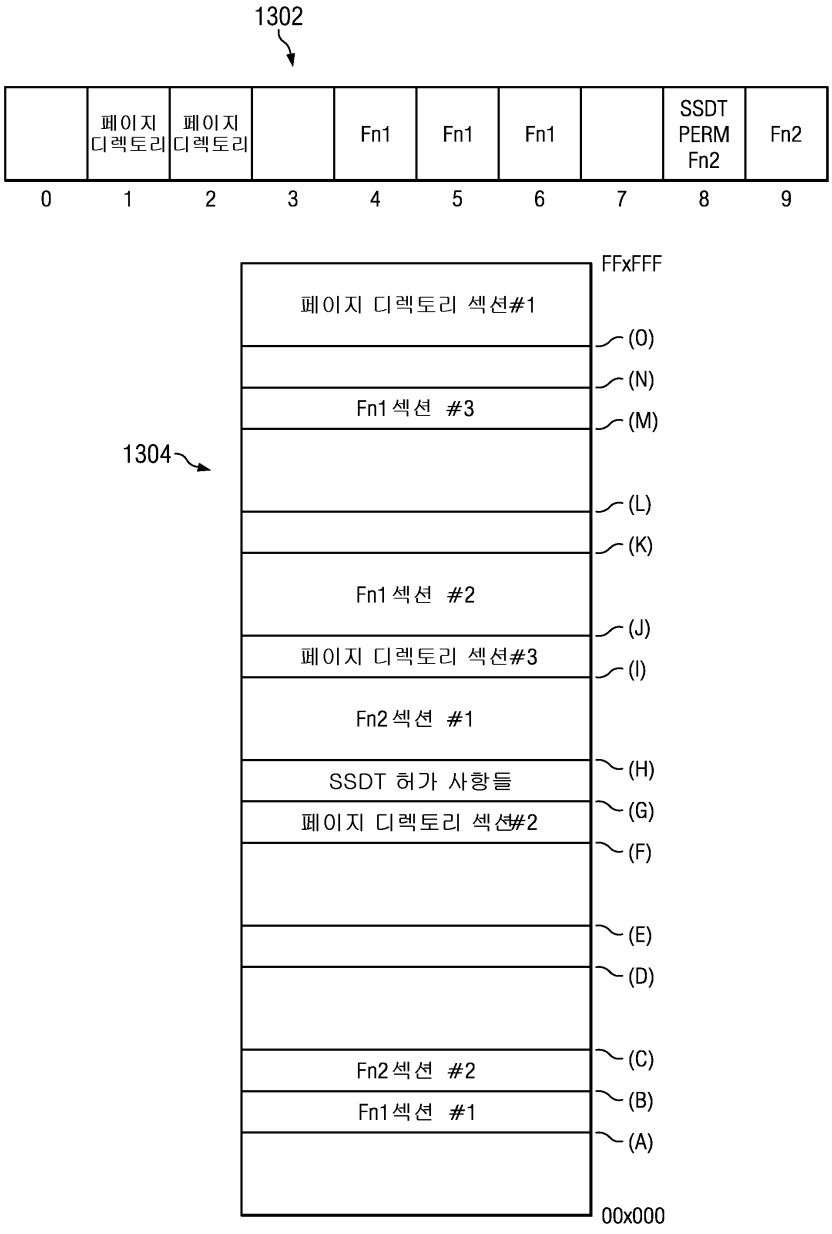
도면11


1100
1105
보안 규칙 액세스
1110
트래핑할 프로세서 자원들과 통신들을 결정
1115
엔트리들을 PRCS에 추가
1120
PRCS 내 엔트리들 인에이블/디세이블
1125
자원 감시
1130
시도가 트래핑되었나?
아니오
예
악성 소프트웨어 설정이나 조건이 변경되었나?
아니오
1135
예
PRCS 내 엔트리 조정
1140
트리거타입이 동기?
예
1145
아니오
단계 1125로 나란히 복귀
트래핑된 시도에 관한 정보 저장
1150
1155
트리거링의 모든 조건 만족?
아니오
예
트리거할 실행 단계 결정
1160
결정된 실행 단계에 트리거된 이벤트 생성
1165
트리거된 이벤트에 대한 정황 정보 판단
1170
트리거된 이벤트를 핸들러에 전송
1175
이벤트 및 정황 정보 이용 보안 규칙들 조회
1180
트리거된 이벤트가 의심스러운가?
아니오
1185
예
트래핑 에이전트로 교정 액션 전송
1190
교정 액션 적용
1195

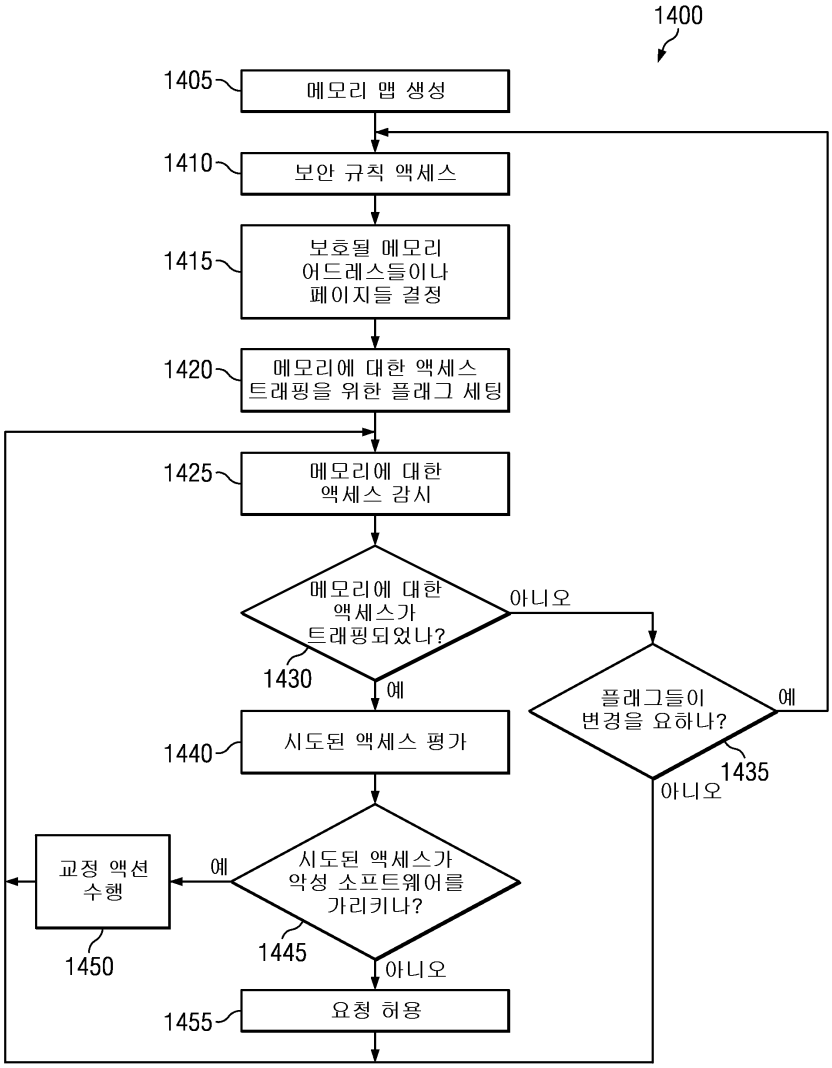
도면12


1200
애플리케이션
드라이버
1211
전자 장치
1210
운영체제
1213
O/S 내
보안
에이전트
보안
규칙들
1221
1219
메모리
요청
(R/W/E)
허용/거부/
스푸핑
202
보호 서버
1220
O/S 하위
보안
에이전트
보안 규칙들
1208
메모리 맵
1206
244
메모리 페이지 읽기
/쓰기/실행하기
메모리 페이지들
0 1 2 3 4 5 6 7 8 9 • • •
1204
가상
메모리
메모리 어드레스 읽기
/쓰기/실행하기
MMU
1214
FFxFFF
(O)
(N)
(M)
(L)
(K)
(J)
(I)
(H)
(G)
(F)
(E)
(D)
(C)
(B)
(A)
00x000
어드레스
1201
메모리 어드레스 읽기
/쓰기/실행하기
1202
프로세서
1203
물리적 메모리

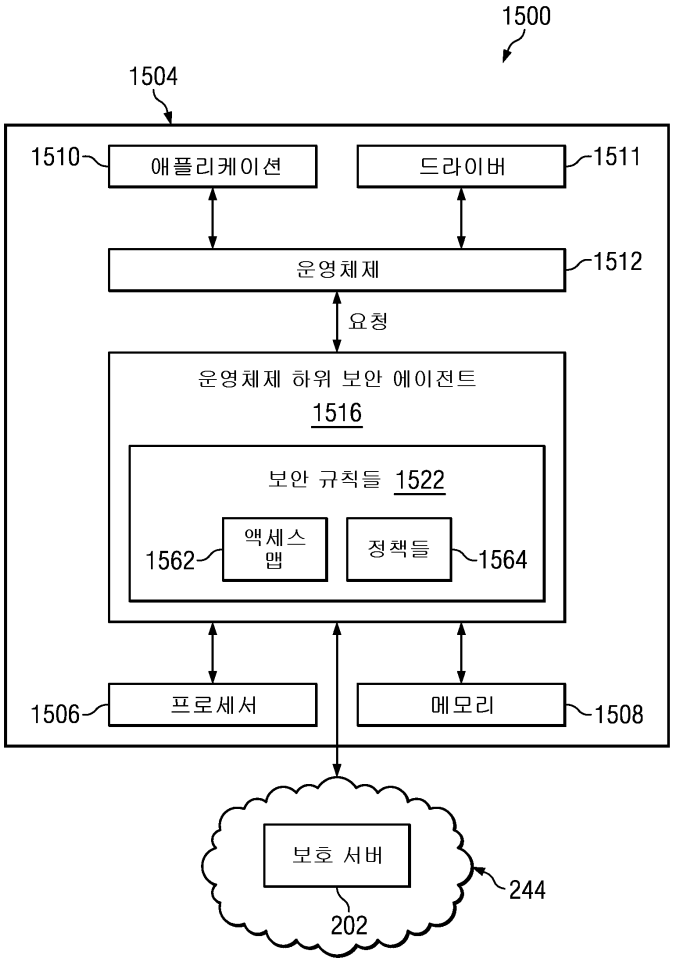
도면13


1302
페이지 디렉토리
페이지 디렉토리
Fn1
Fn1
Fn1
SSDT PERM Fn2
Fn2
0
1
2
3
4
5
6
7
8
9
1304
FFxFFF
페이지 디렉토리 섹션#1
Fn1 섹션 #3
Fn1 섹션 #2
페이지 디렉토리 섹션#3
Fn2 섹션 #1
SSDT 허가 사항들
페이지 디렉토리 섹션#2
Fn2 섹션 #2
Fn1 섹션 #1
00x000
(O)
(N)
(M)
(L)
(K)
(J)
(I)
(H)
(G)
(F)
(E)
(D)
(C)
(B)
(A)

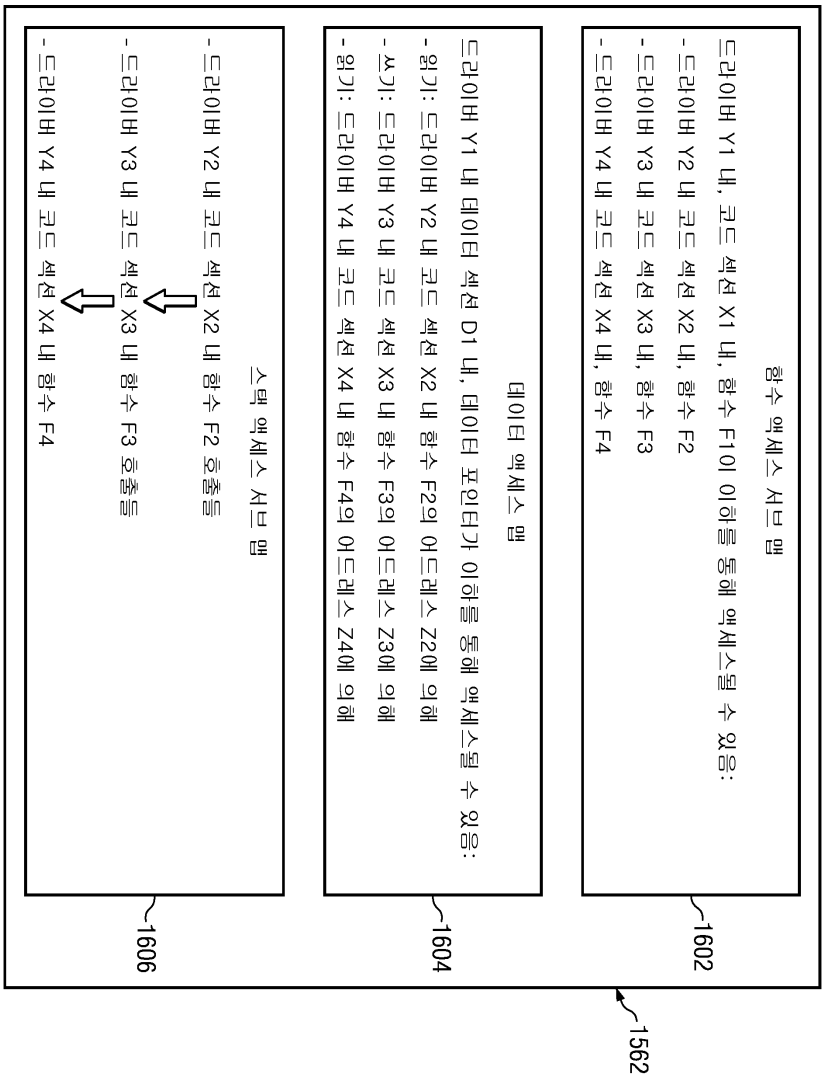
도면14


1400
1405
메모리 맵 생성
1410
보안 규칙 액세스
1415
보호될 메모리 어드레스들이나 페이지들 결정
1420
메모리에 대한 액세스 트래핑을 위한 플래그 세팅
1425
메모리에 대한 액세스 감시
메모리에 대한 액세스가 트래핑되었나?
1430
아니오
예
플래그들이 변경을 요하나?
1435
예
아니오
1440
시도된 액세스 평가
시도된 액세스가 악성 소프트웨어를 가리키나?
1445
예
아니오
교정 액션 수행
1450
1455
요청 허용

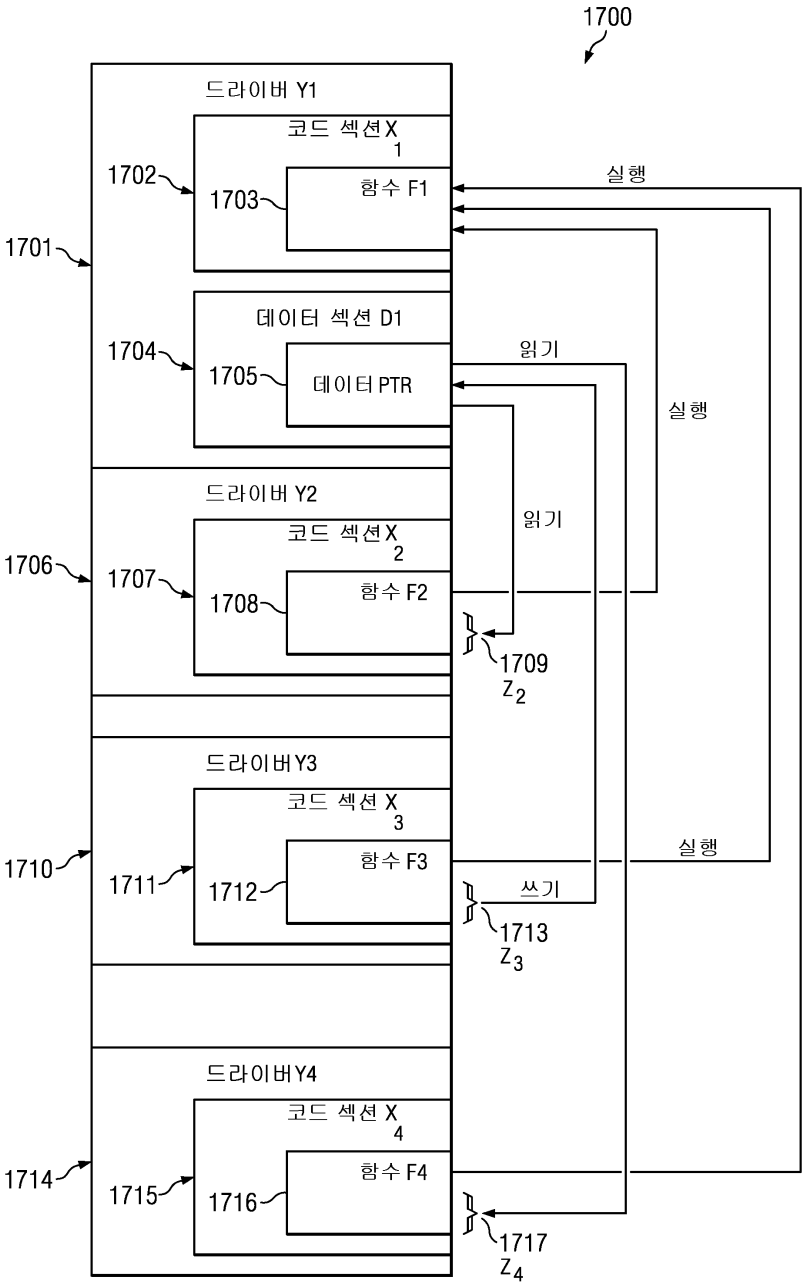
도면15


1500
1504
1510
애플리케이션
드라이버
1511
운영체제
1512
요청
운영체제 하위 보안 에이전트
1516
보안 규칙들 1522
액세스 맵
1562
정책들
1564
프로세서
1506
메모리
1508
보호 서버
202
244

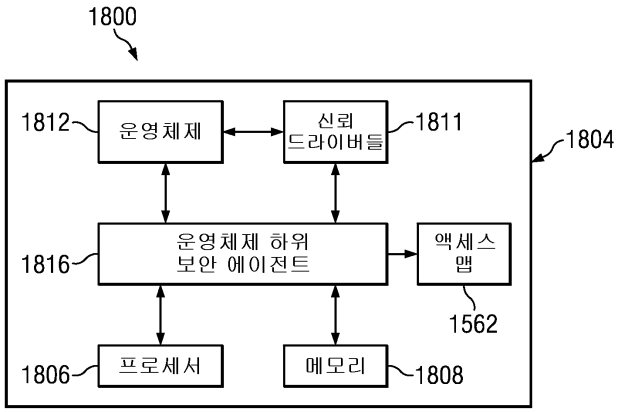
도면16


함수 액세스 서브 맵
드라이버 Y1 내, 코드 섹션 X1 내, 함수 F1이 이하를 통해 액세스될 수 있음:
- 드라이버 Y2 내 코드 섹션 X2 내, 함수 F2
- 드라이버 Y3 내 코드 섹션 X3 내, 함수 F3
- 드라이버 Y4 내 코드 섹션 X4 내, 함수 F4
1602
데이터 액세스 맵
드라이버 Y1 내 데이터 섹션 D1 내, 데이터 포인터가 이하를 통해 액세스될 수 있음:
- 읽기: 드라이버 Y2 내 코드 섹션 X2 내 함수 F2의 어드레스 Z2에 의해
- 쓰기: 드라이버 Y3 내 코드 섹션 X3 내 함수 F3의 어드레스 Z3에 의해
- 읽기: 드라이버 Y4 내 코드 섹션 X4 내 함수 F4의 어드레스 Z4에 의해
1604
스택 액세스 서브 맵
- 드라이버 Y2 내 코드 섹션 X2 내 함수 F2 호출들
- 드라이버 Y3 내 코드 섹션 X3 내 함수 F3 호출들
- 드라이버 Y4 내 코드 섹션 X4 내 함수 F4
1606
1562

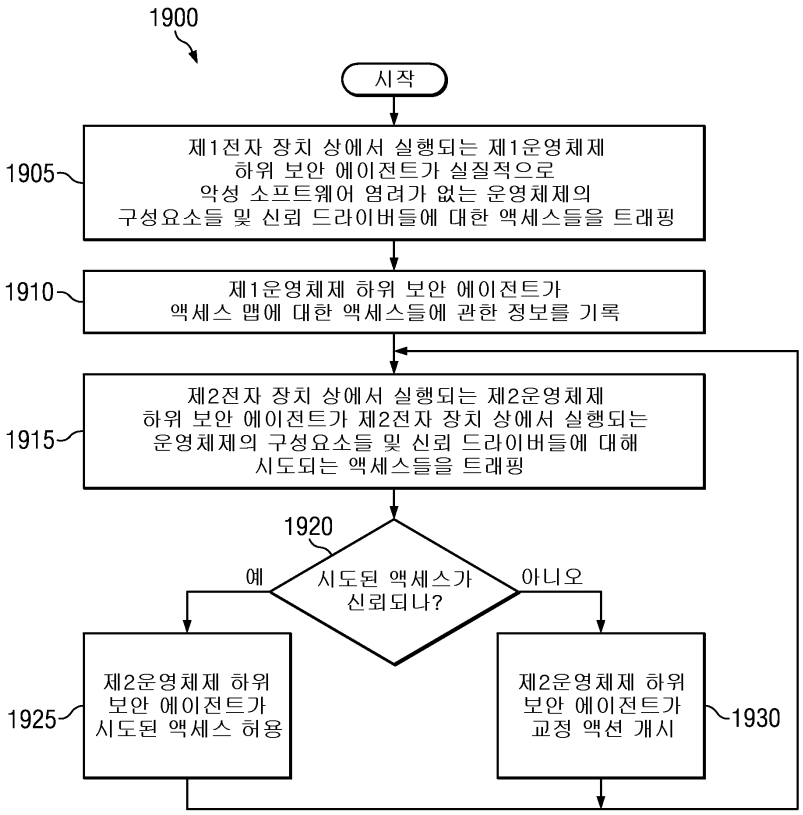
도면17


1700
드라이버 Y1
코드 섹션 X 1
1702
1703
함수 F1
실행
1701
데이터 섹션 D1
1704
1705
데이터 PTR
읽기
실행
드라이버 Y2
읽기
코드 섹션 X 2
1706
1707
1708
함수 F2
1709
Z2
드라이버Y3
코드 섹션 X 3
1710
1711
1712
함수 F3
실행
쓰기
1713
Z3
드라이버Y4
코드 섹션 X 4
1714
1715
1716
함수 F4
1717
Z4

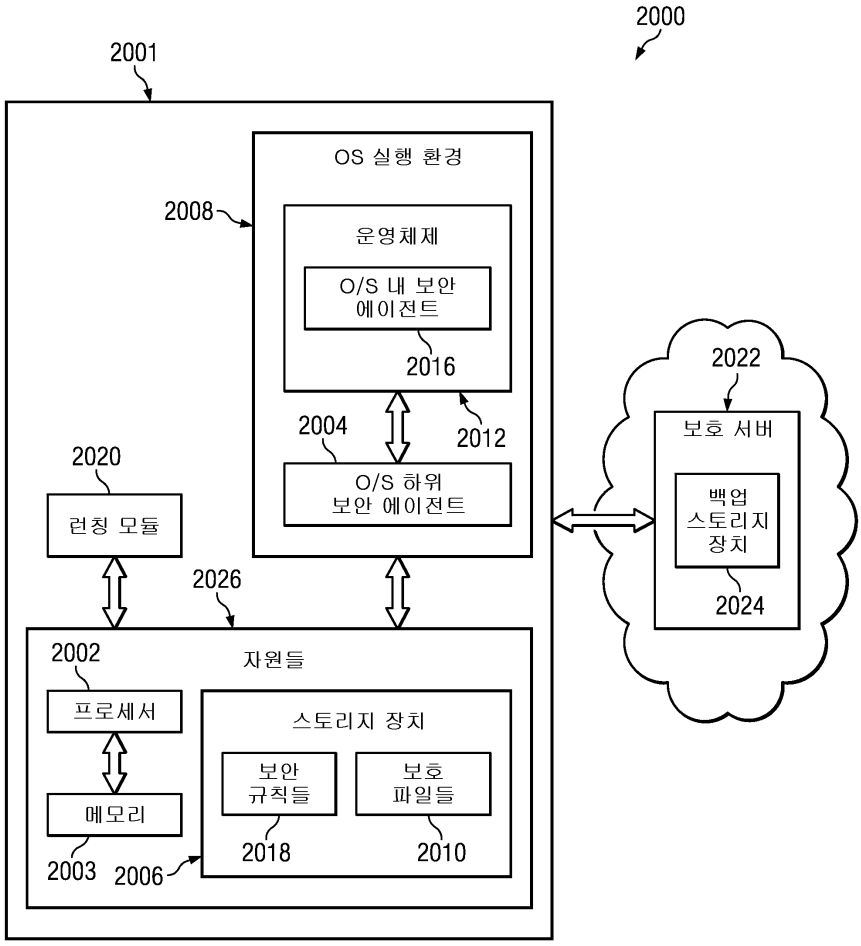
도면18


1800
1812
운영체제
신뢰 드라이버들
1811
1804
운영체제 하위 보안 에이전트
1816
액세스 맵
1562
1806
프로세서
메모리
1808

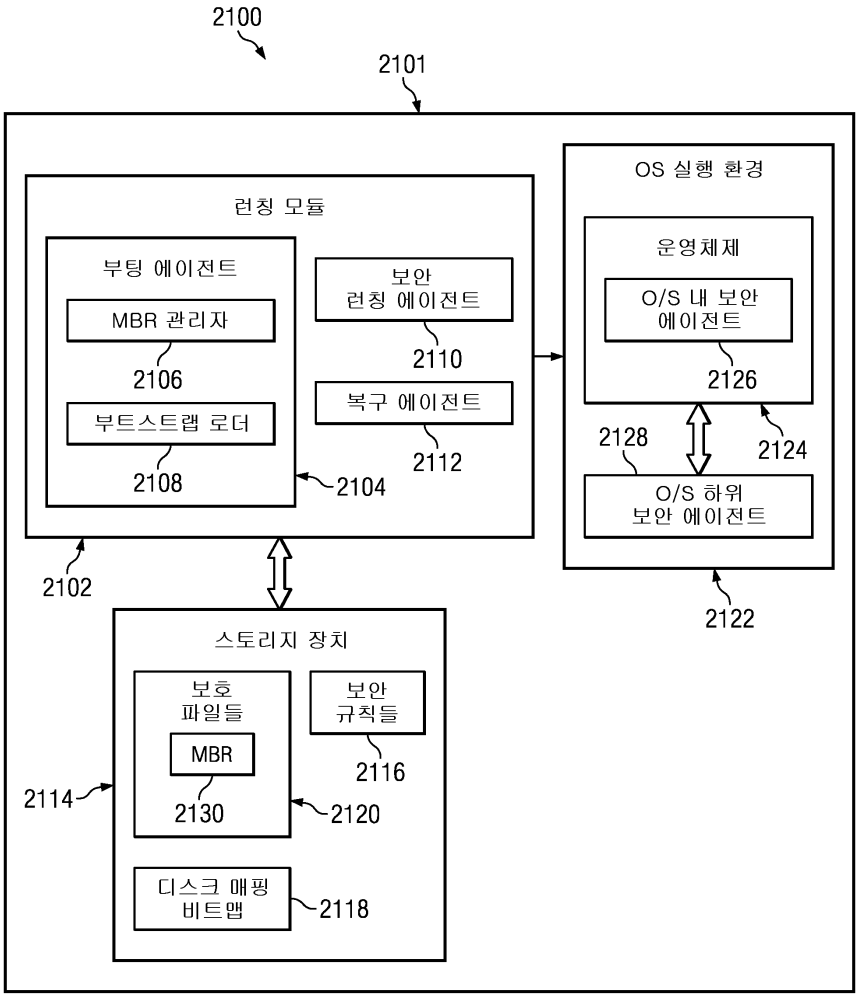
도면19


1900
시작
제1전자 장치 상에서 실행되는 제1운영체제
하위 보안 에이전트가 실질적으로
악성 소프트웨어 염려가 없는 운영체제의
구성요소들 및 신뢰 드라이버들에 대한 액세스들을 트래핑
1905
제1운영체제 하위 보안 에이전트가
액세스 맵에 대한 액세스들에 관한 정보를 기록
1910
제2전자 장치 상에서 실행되는 제2운영체제
하위 보안 에이전트가 제2전자 장치 상에서 실행되는
운영체제의 구성요소들 및 신뢰 드라이버들에 대해
시도되는 액세스들을 트래핑
1915
1920
시도된 액세스가
신뢰되나?
예
아니오
제2운영체제 하위
보안 에이전트가
시도된 액세스 허용
1925
제2운영체제 하위
보안 에이전트가
교정 액션 개시
1930

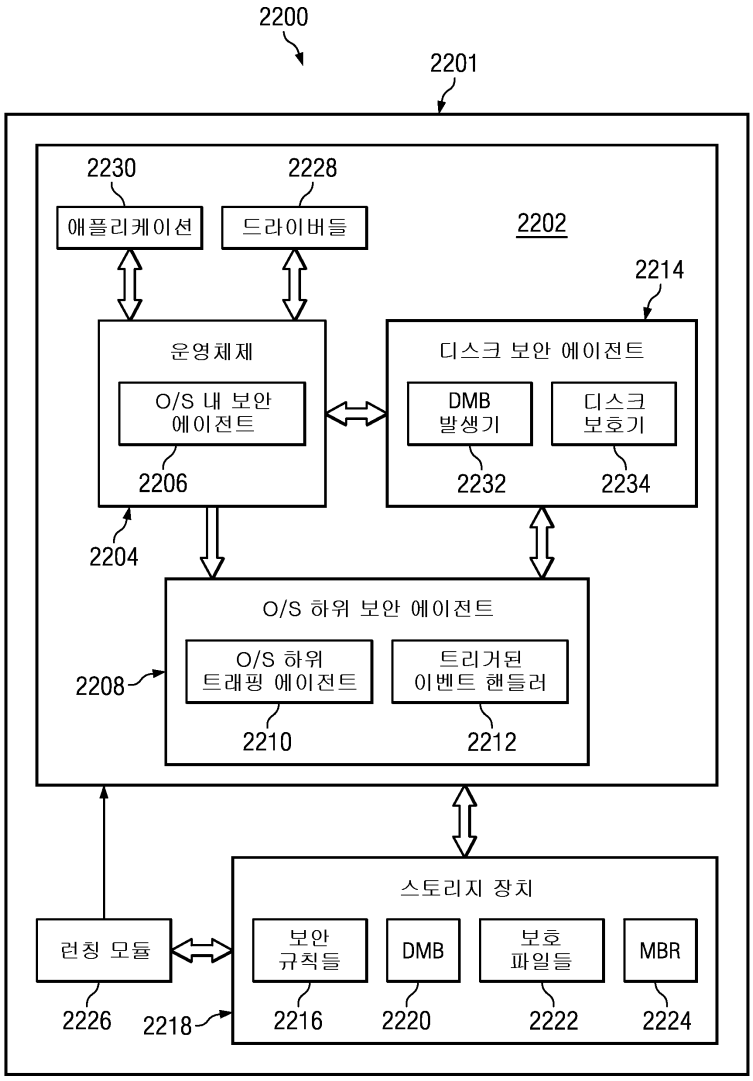
도면20


2000
2001
OS 실행 환경
2008
운영체제
O/S 내 보안
에이전트
2016
2004
2012
O/S 하위
보안 에이전트
2020
런칭 모듈
2026
2022
보호 서버
백업
스토리지
장치
2024
2002
자원들
프로세서
메모리
2003
2006
스토리지 장치
보안
규칙들
보호
파일들
2018
2010

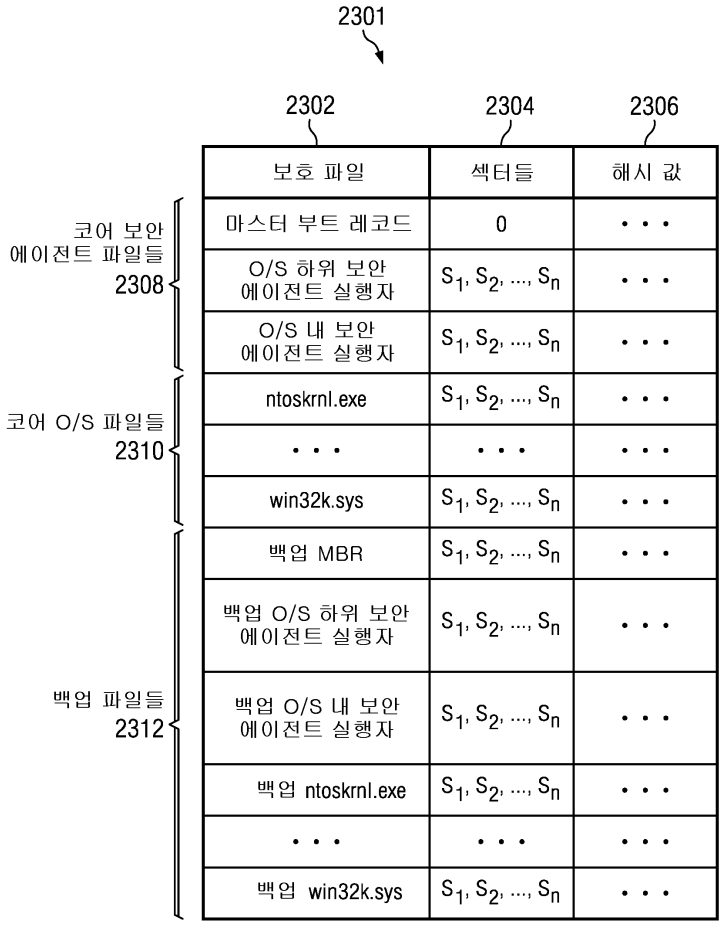
도면21


2100
2101
런칭 모듈
부팅 에이전트
MBR 관리자
2106
부트스트랩 로더
2108
보안
런칭 에이전트
2110
복구 에이전트
2112
2104
OS 실행 환경
운영체제
O/S 내 보안
에이전트
2126
2128
2124
O/S 하위
보안 에이전트
2122
2102
스토리지 장치
보호
파일들
MBR
2130
보안
규칙들
2116
2114
2120
디스크 매핑
비트맵
2118

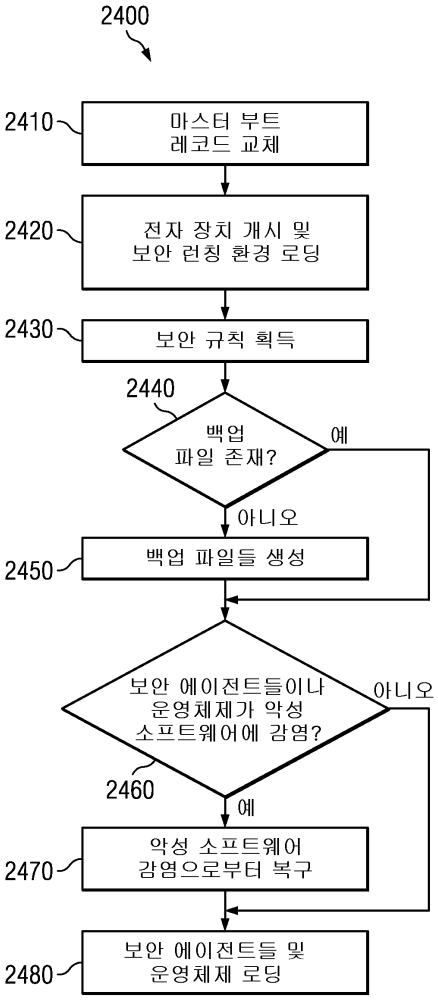
도면22


2200
2201
2230
2228
애플리케이션
드라이버들
2202
2214
운영체제
O/S 내 보안
에이전트
2206
디스크 보안 에이전트
DMB
발생기
디스크
보호기
2232
2234
2204
O/S 하위 보안 에이전트
2208
O/S 하위
트래핑 에이전트
트리거된
이벤트 핸들러
2210
2212
스토리지 장치
런칭 모듈
보안
규칙들
DMB
보호
파일들
MBR
2226
2218
2216
2220
2222
2224

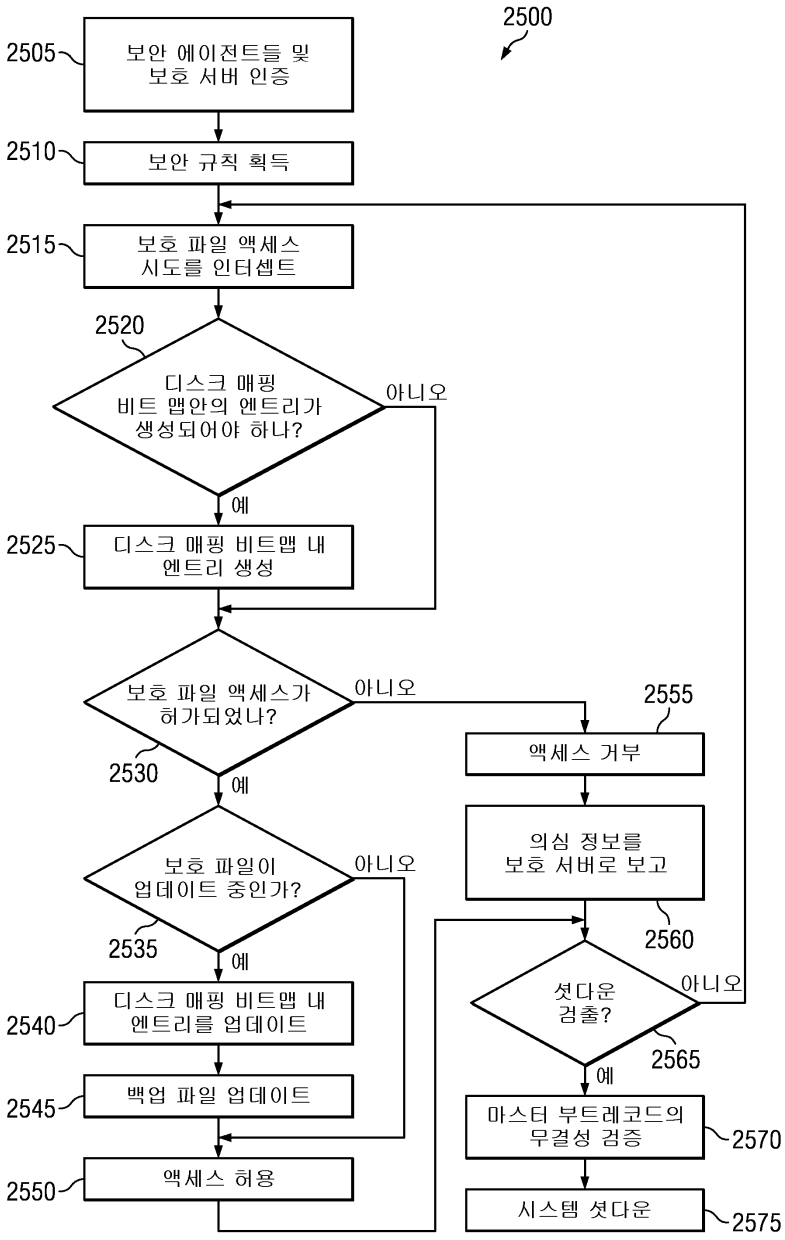
도면23


2301
2302
2304
2306
보호 파일
섹터들
해시 값
코어 보안 에이전트 파일들 2308
마스터 부트 레코드
0
O/S 하위 보안 에이전트 실행자
$S_1, S_2, ..., S_n$
O/S 내 보안 에이전트 실행자
$S_1, S_2, ..., S_n$
코어 O/S 파일들 2310
ntoskrnl.exe
$S_1, S_2, ..., S_n$
win32k.sys
$S_1, S_2, ..., S_n$
백업 파일들 2312
백업 MBR
$S_1, S_2, ..., S_n$
백업 O/S 하위 보안 에이전트 실행자
$S_1, S_2, ..., S_n$
백업 O/S 내 보안 에이전트 실행자
$S_1, S_2, ..., S_n$
백업 ntoskrnl.exe
$S_1, S_2, ..., S_n$
백업 win32k.sys
$S_1, S_2, ..., S_n$

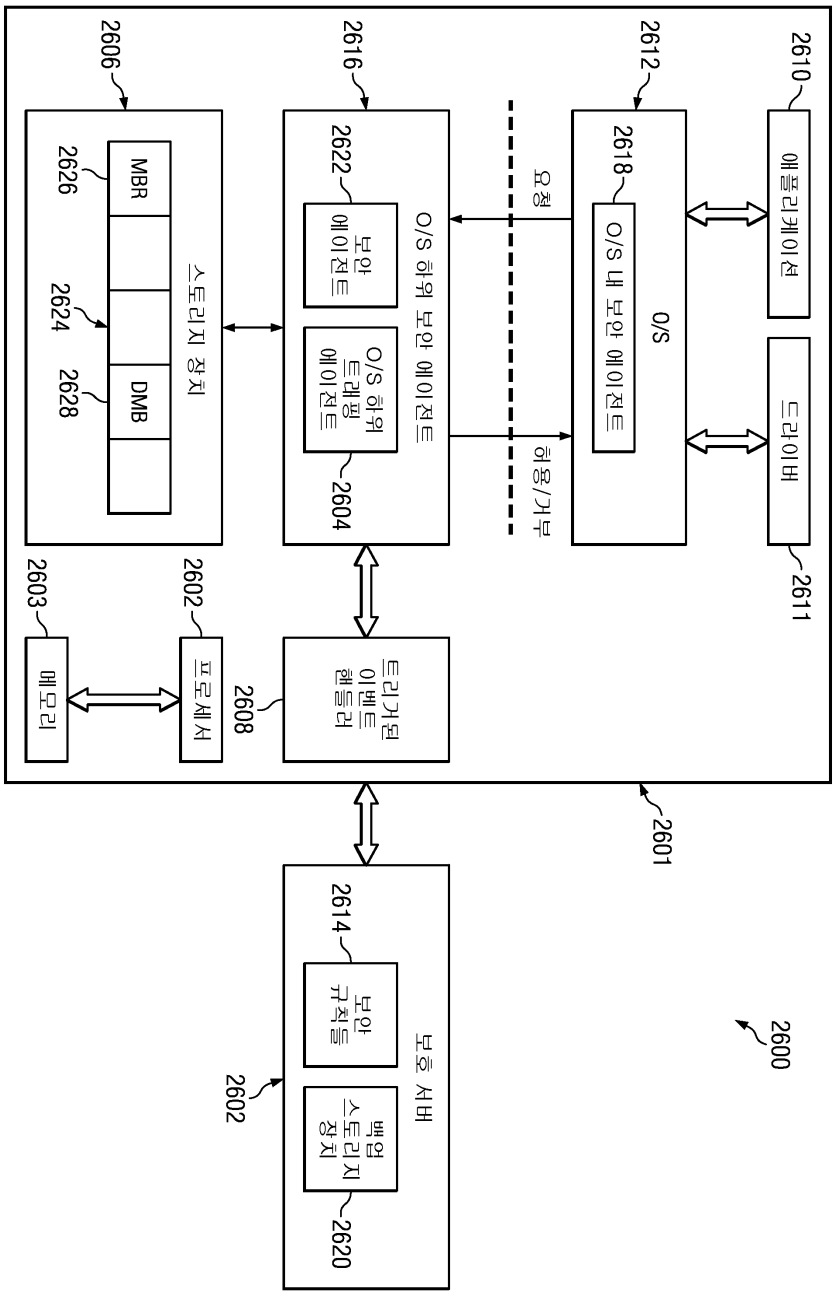
도면24


2400
2410
마스터 부트
레코드 교체
2420
전자 장치 개시 및
보안 런칭 환경 로딩
2430
보안 규칙 획득
2440
백업
파일 존재?
예
아니오
2450
백업 파일들 생성
보안 에이전트들이나
운영체제가 악성
소프트웨어에 감염?
아니오
2460
예
2470
악성 소프트웨어
감염으로부터 복구
2480
보안 에이전트들 및
운영체제 로딩

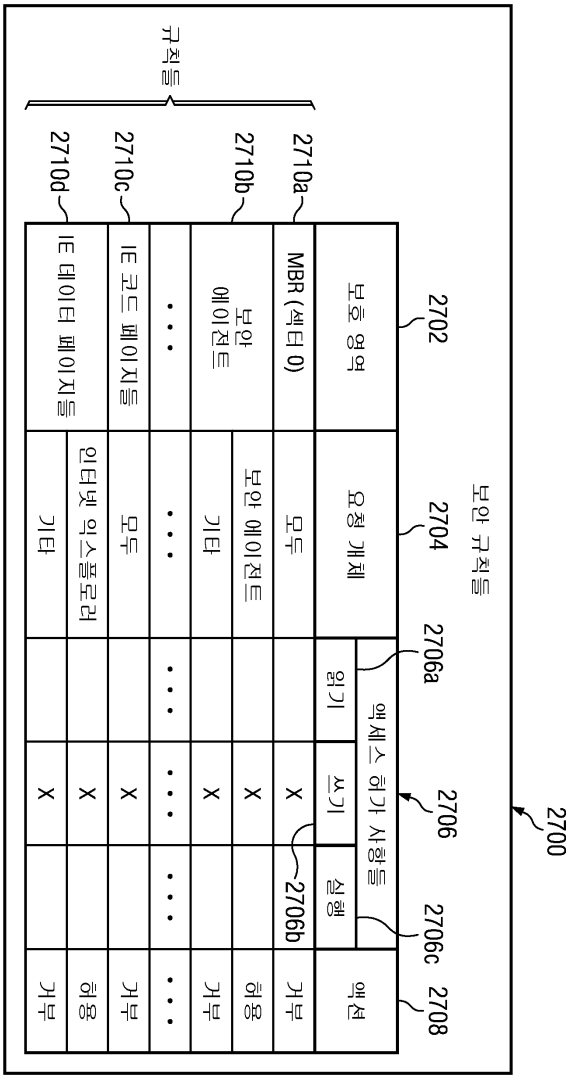
도면25


2500
2505
보안 에이전트들 및 보호 서버 인증
2510
보안 규칙 획득
2515
보호 파일 액세스 시도를 인터셉트
2520
디스크 매핑 비트 맵안의 엔트리가 생성되어야 하나?
아니오
예
2525
디스크 매핑 비트맵 내 엔트리 생성
보호 파일 액세스가 허가되었나?
아니오
2530
예
2555
액세스 거부
보호 파일이 업데이트 중인가?
아니오
2535
예
의심 정보를 보호 서버로 보고
2560
셧다운 검출?
아니오
2565
예
2540
디스크 매핑 비트맵 내 엔트리를 업데이트
2545
백업 파일 업데이트
2550
액세스 허용
마스터 부트레코드의 무결성 검증
2570
시스템 셧다운
2575

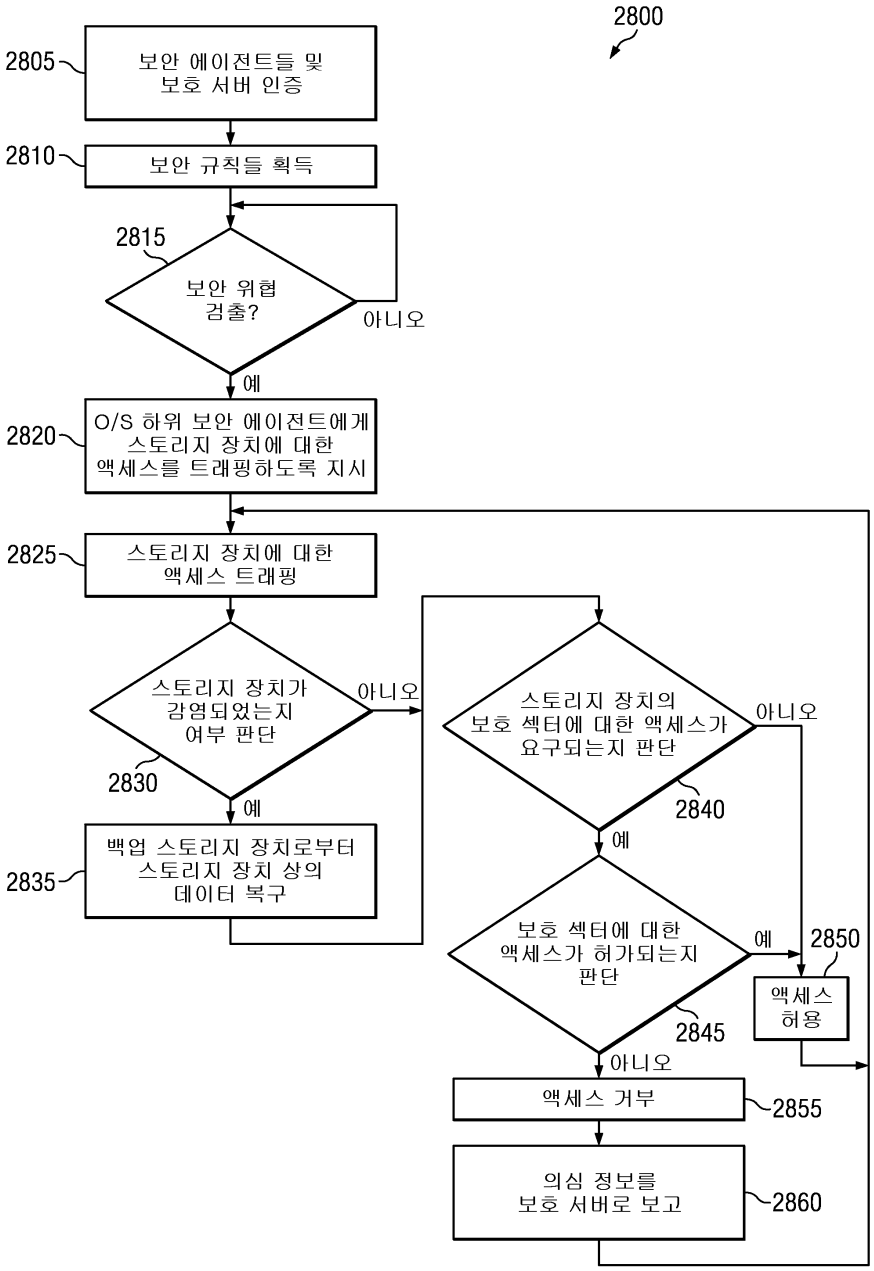
도면26


2600
2601
2602
보호 서버
2614
보안 규칙들
2620
백업 스토리지 장치
2608
트리거링 이벤트 핸들러
2603
메모리
2602
프로세서
2606
스토리지 장치
2626
MBR
2624
2628
DMB
2616
O/S 하위 보안 에이전트
2622
보안 에이전트
2604
O/S 하위 드래곤 에이전트
요청
허용/거부
2612
O/S
2618
O/S 내 보안 에이전트
2610
애플리케이션
2611
드라이버

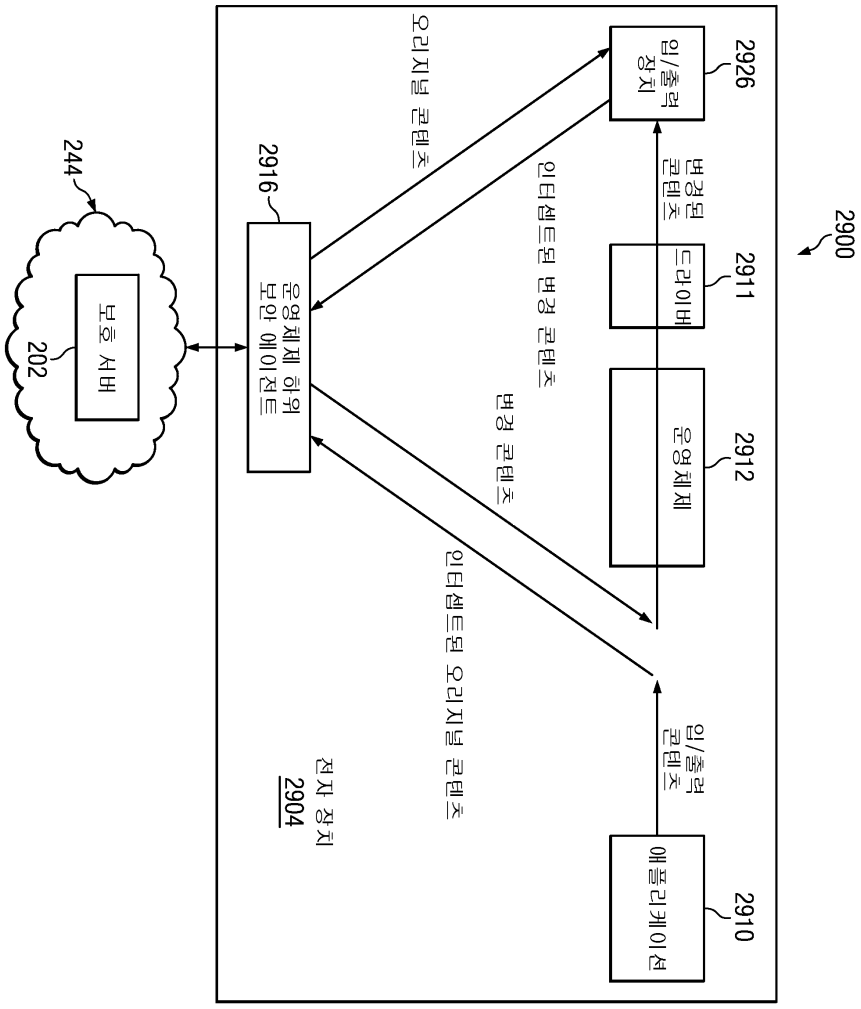
도면27


2700
보안 규칙들
2702
보호 영역
2704
요청 개체
2706
액세스 허가 사항들
2706a
읽기
쓰기
실행
2706b
2706c
2708
액션
2710a
MBR (섹터 0)
모두
X
X
X
거부
2710b
보안 에이전트
보안 에이전트
기타
X
X
X
허용
기타
X
X
X
거부
2710c
IE 코드 페이지들
모두
X
X
X
거부
2710d
IE 데이터 페이지들
인터넷 익스플로러
X
X
X
허용
기타
X
X
X
거부
규칙들

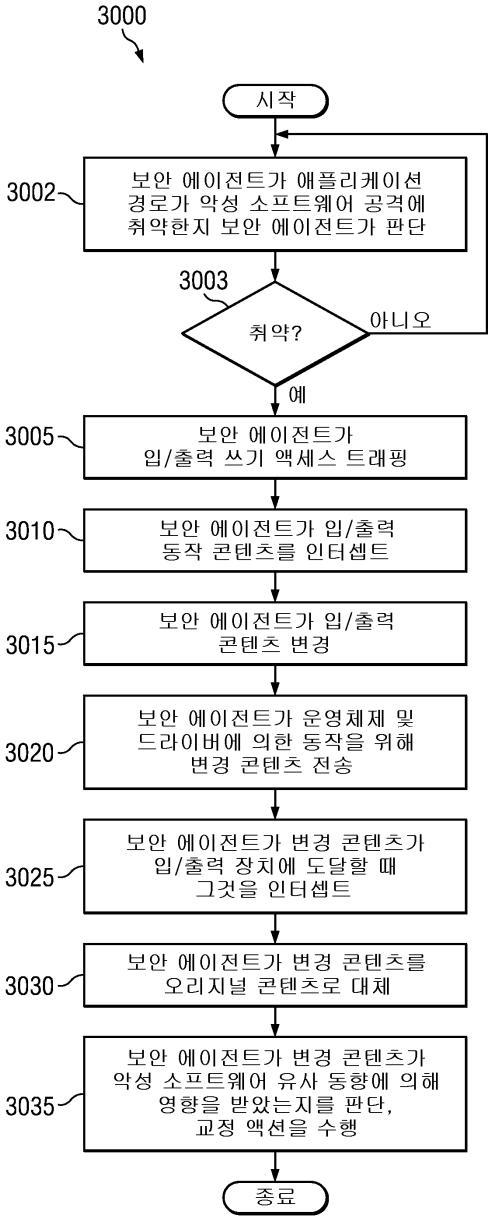
도면28


2800
2805
보안 에이전트들 및 보호 서버 인증
2810
보안 규칙들 획득
2815
보안 위협 검출?
아니오
예
2820
O/S 하위 보안 에이전트에게 스토리지 장치에 대한 액세스를 트래핑하도록 지시
2825
스토리지 장치에 대한 액세스 트래핑
스토리지 장치가 감염되었는지 여부 판단
아니오
2830
예
2835
백업 스토리지 장치로부터 스토리지 장치 상의 데이터 복구
스토리지 장치의 보호 섹터에 대한 액세스가 요구되는지 판단
아니오
2840
예
보호 섹터에 대한 액세스가 허가되는지 판단
예
2850
액세스 허용
2845
아니오
액세스 거부
2855
의심 정보를 보호 서버로 보고
2860

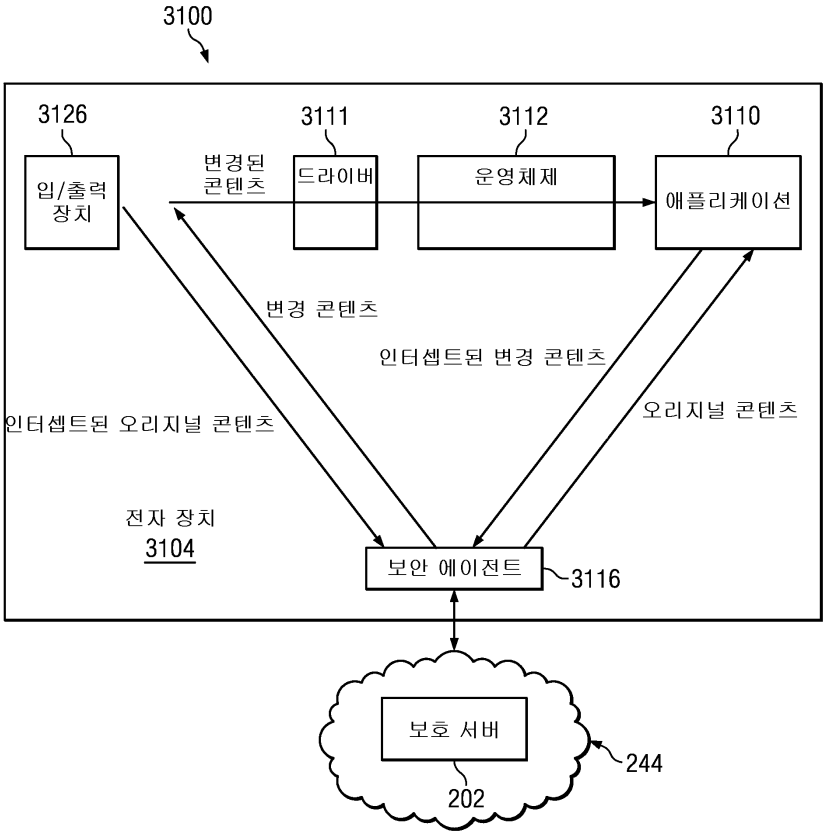
도면29


2900
2926
입/출력 장치
2911
드라이버
2912
운영 체제
2910
애플리케이션
입/출력 콘텐츠
변경된 콘텐츠
인터셉트된 변경 콘텐츠
변경 콘텐츠
인터셉트된 오리지널 콘텐츠
오리지널 콘텐츠
2916
운영체제 하위 보안 에이전트
244
보호 서버
202
전자 장치
2904

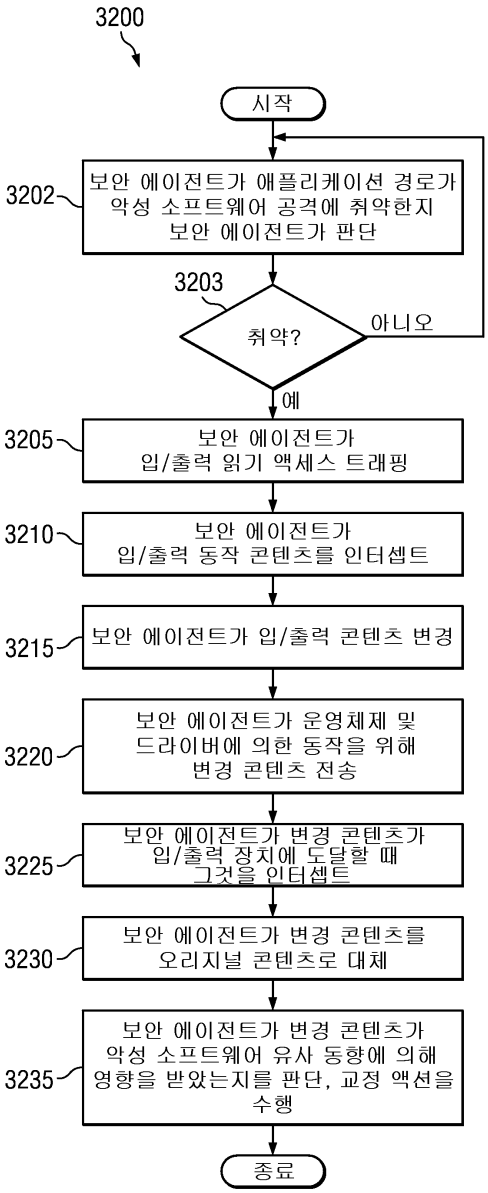
도면30


3000
시작
3002
보안 에이전트가 애플리케이션 경로가 악성 소프트웨어 공격에 취약한지 보안 에이전트가 판단
3003
취약?
아니오
예
3005
보안 에이전트가 입/출력 쓰기 액세스 트래핑
3010
보안 에이전트가 입/출력 동작 콘텐츠를 인터셉트
3015
보안 에이전트가 입/출력 콘텐츠 변경
3020
보안 에이전트가 운영체제 및 드라이버에 의한 동작을 위해 변경 콘텐츠 전송
3025
보안 에이전트가 변경 콘텐츠가 입/출력 장치에 도달할 때 그것을 인터셉트
3030
보안 에이전트가 변경 콘텐츠를 오리지널 콘텐츠로 대체
3035
보안 에이전트가 변경 콘텐츠가 악성 소프트웨어 유사 동향에 의해 영향을 받았는지를 판단, 교정 액션을 수행
종료

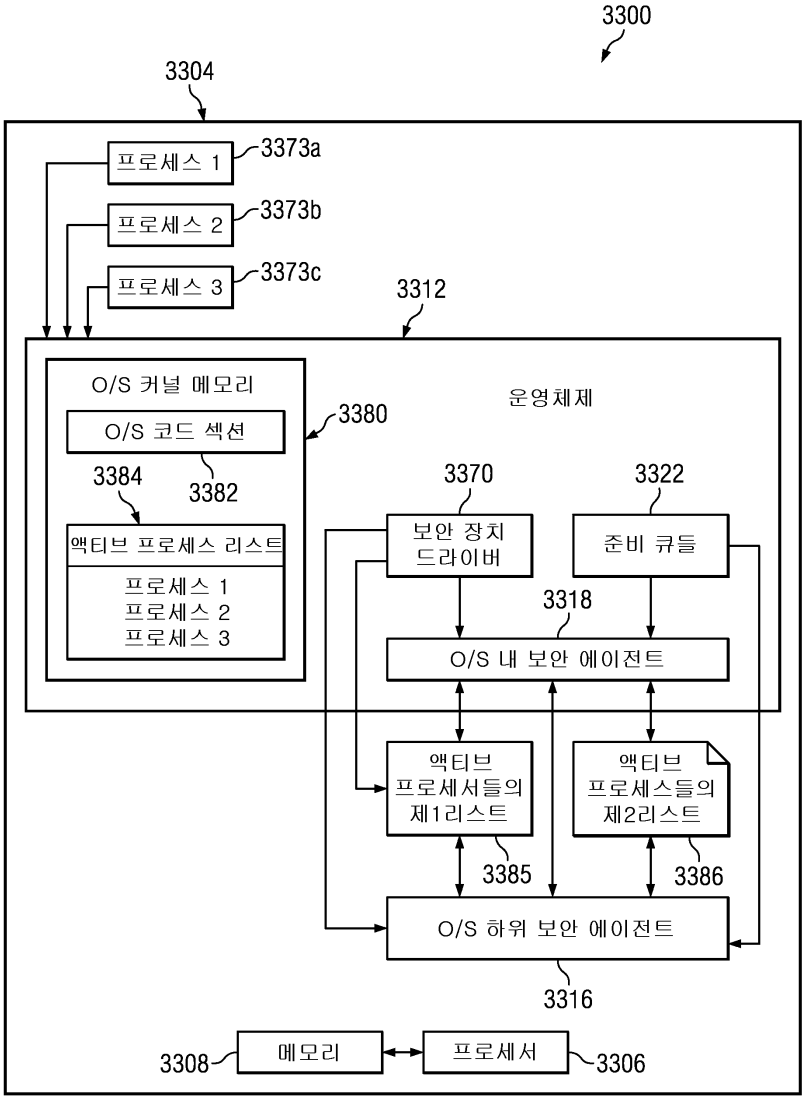
도면31


3100
3126
입/출력 장치
3111
드라이버
3112
운영체제
3110
애플리케이션
변경된 콘텐츠
변경 콘텐츠
인터셉트된 변경 콘텐츠
인터셉트된 오리지널 콘텐츠
오리지널 콘텐츠
전자 장치
3104
보안 에이전트
3116
보호 서버
202
244

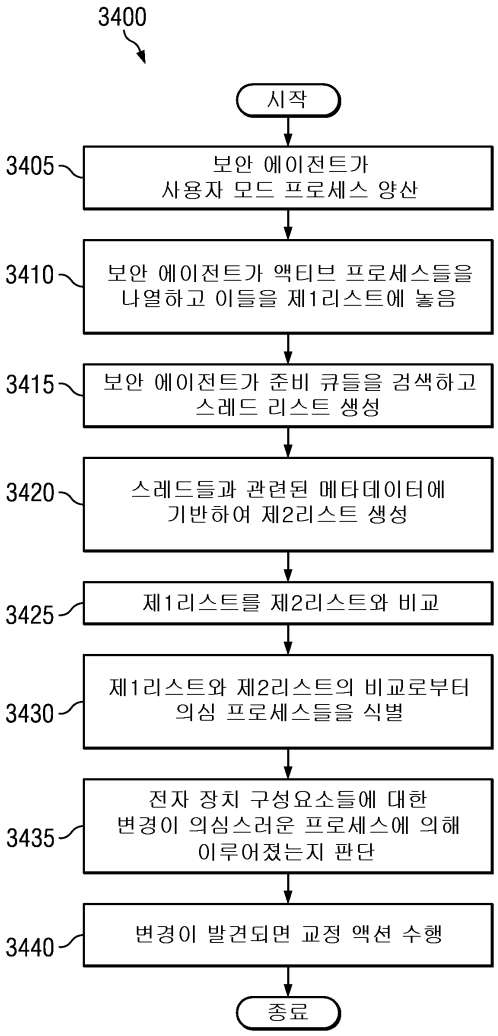
도면32


3200
시작
3202
보안 에이전트가 애플리케이션 경로가 악성 소프트웨어 공격에 취약한지 보안 에이전트가 판단
3203
취약?
아니오
예
3205
보안 에이전트가 입/출력 읽기 액세스 트래핑
3210
보안 에이전트가 입/출력 동작 콘텐츠를 인터셉트
3215
보안 에이전트가 입/출력 콘텐츠 변경
3220
보안 에이전트가 운영체제 및 드라이버에 의한 동작을 위해 변경 콘텐츠 전송
3225
보안 에이전트가 변경 콘텐츠가 입/출력 장치에 도달할 때 그것을 인터셉트
3230
보안 에이전트가 변경 콘텐츠를 오리지널 콘텐츠로 대체
3235
보안 에이전트가 변경 콘텐츠가 악성 소프트웨어 유사 동향에 의해 영향을 받았는지를 판단, 교정 액션을 수행
종료

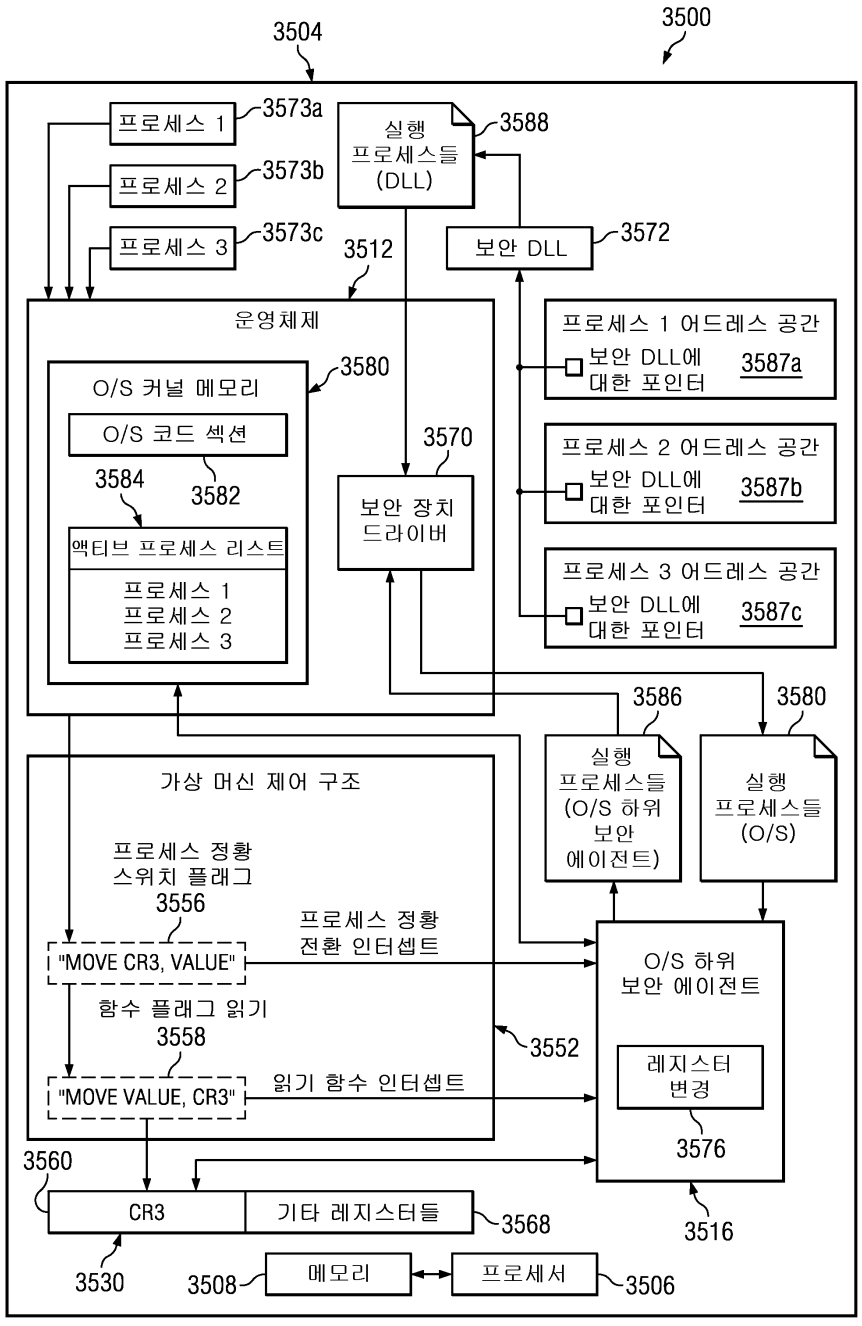
도면33


3300
3304
프로세스 1
3373a
프로세스 2
3373b
프로세스 3
3373c
3312
O/S 커널 메모리
운영체제
O/S 코드 섹션
3380
3384
3382
액티브 프로세스 리스트
프로세스 1
프로세스 2
프로세스 3
3370
3322
보안 장치 드라이버
준비 큐들
3318
O/S 내 보안 에이전트
액티브 프로세서들의 제1리스트
액티브 프로세스들의 제2리스트
3385
3386
O/S 하위 보안 에이전트
3316
3308
메모리
프로세서
3306

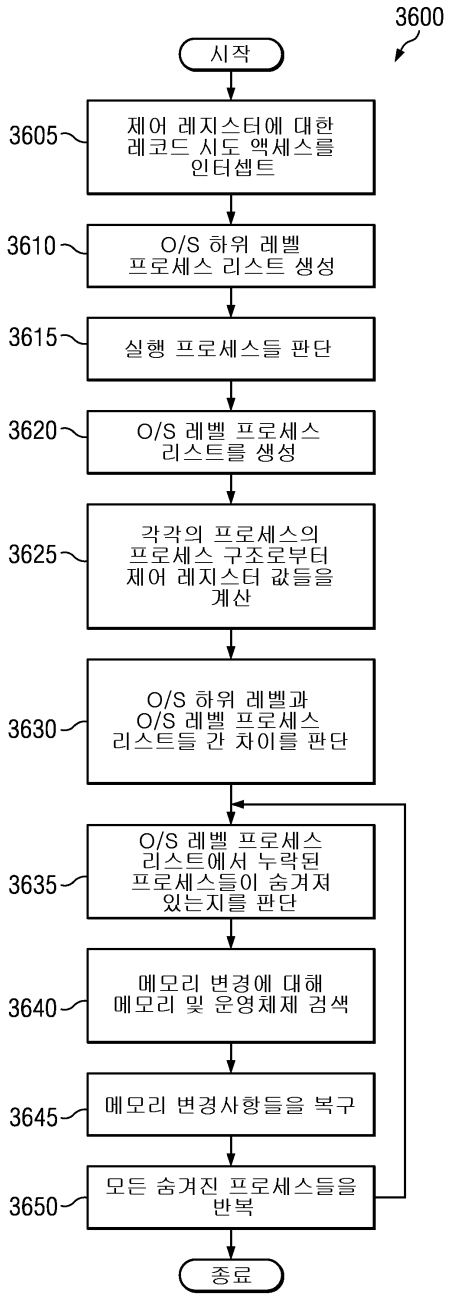
도면34


3400
시작
3405
보안 에이전트가
사용자 모드 프로세스 양산
3410
보안 에이전트가 액티브 프로세스들을
나열하고 이들을 제1리스트에 놓음
3415
보안 에이전트가 준비 큐들을 검색하고
스레드 리스트 생성
3420
스레드들과 관련된 메타데이터에
기반하여 제2리스트 생성
3425
제1리스트를 제2리스트와 비교
3430
제1리스트와 제2리스트의 비교로부터
의심 프로세스들을 식별
3435
전자 장치 구성요소들에 대한
변경이 의심스러운 프로세스에 의해
이루어졌는지 판단
3440
변경이 발견되면 교정 액션 수행
종료

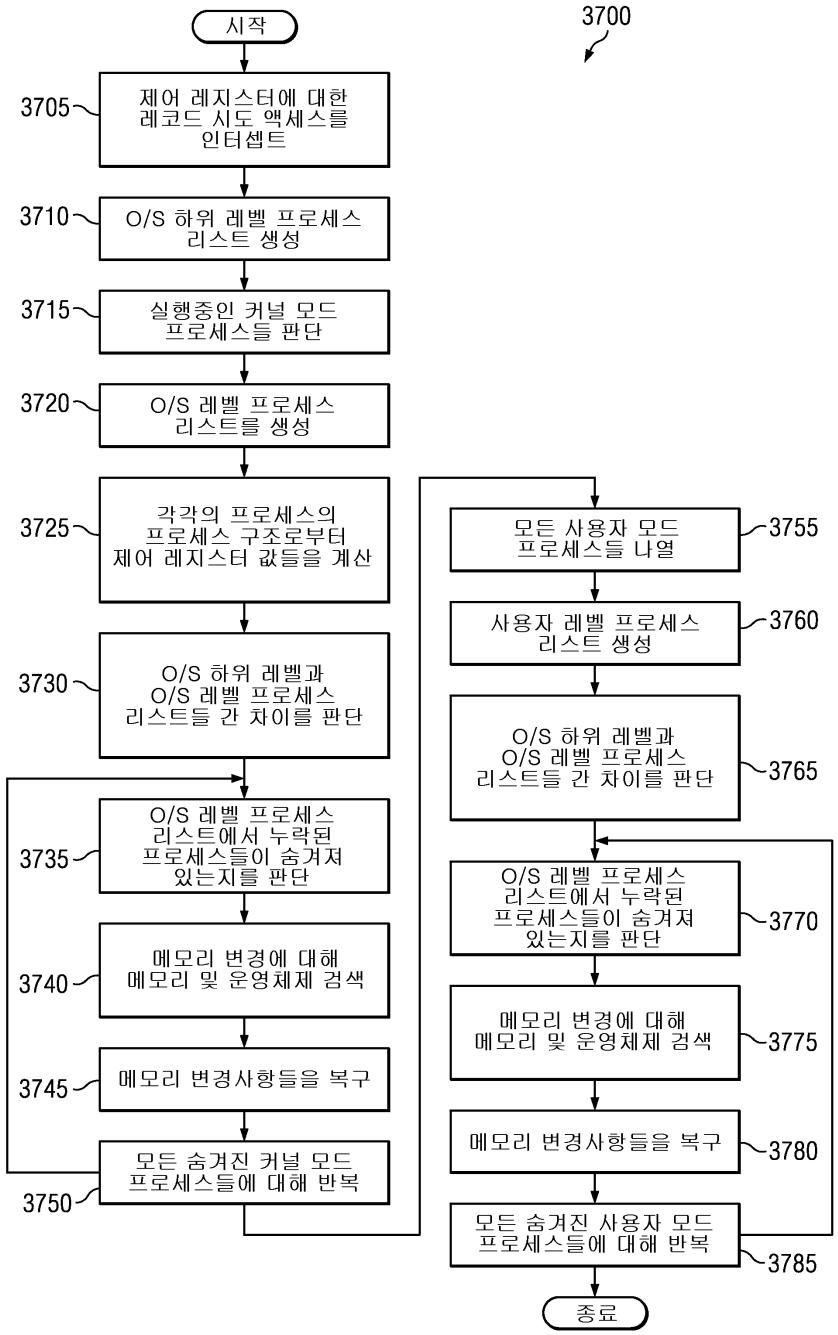
도면35


3504
3500
프로세스 1
3573a
프로세스 2
3573b
프로세스 3
3573c
실행 프로세스들 (DLL)
3588
보안 DLL
3572
3512
운영체제
O/S 커널 메모리
3580
O/S 코드 섹션
3584
3582
액티브 프로세스 리스트
프로세스 1
프로세스 2
프로세스 3
3570
보안 장치 드라이버
프로세스 1 어드레스 공간
보안 DLL에 대한 포인터
3587a
프로세스 2 어드레스 공간
보안 DLL에 대한 포인터
3587b
프로세스 3 어드레스 공간
보안 DLL에 대한 포인터
3587c
3586
3580
실행 프로세스들 (O/S 하위 보안 에이전트)
실행 프로세스들 (O/S)
가상 머신 제어 구조
프로세스 정황 스위치 플래그
3556
"MOVE CR3, VALUE"
프로세스 정황 전환 인터셉트
함수 플래그 읽기
3558
"MOVE VALUE, CR3"
읽기 함수 인터셉트
3552
O/S 하위 보안 에이전트
레지스터 변경
3576
3560
CR3
기타 레지스터들
3568
3516
3530
3508
메모리
프로세서
3506

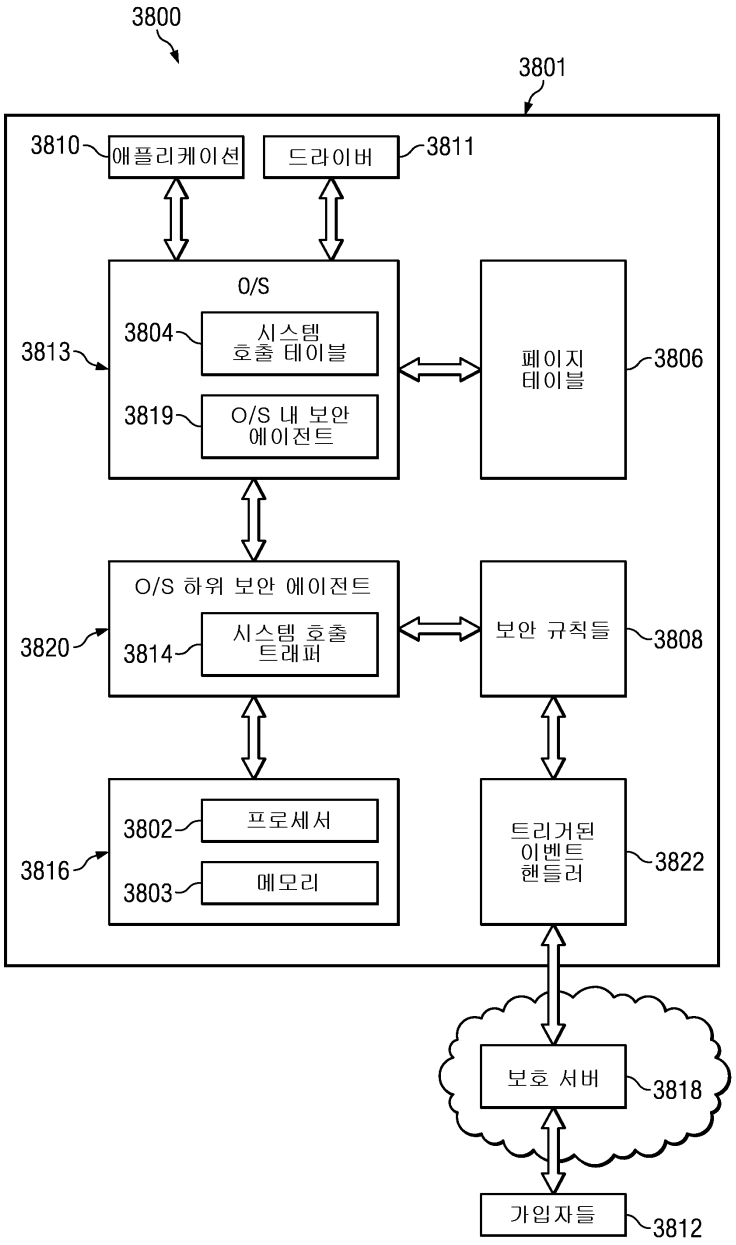
도면36


3600
시작
3605
제어 레지스터에 대한 레코드 시도 액세스를 인터셉트
3610
O/S 하위 레벨 프로세스 리스트 생성
3615
실행 프로세스들 판단
3620
O/S 레벨 프로세스 리스트를 생성
3625
각각의 프로세스의 프로세스 구조로부터 제어 레지스터 값들을 계산
3630
O/S 하위 레벨과 O/S 레벨 프로세스 리스트들 간 차이를 판단
3635
O/S 레벨 프로세스 리스트에서 누락된 프로세스들이 숨겨져 있는지를 판단
3640
메모리 변경에 대해 메모리 및 운영체제 검색
3645
메모리 변경사항들을 복구
3650
모든 숨겨진 프로세스들을 반복
종료

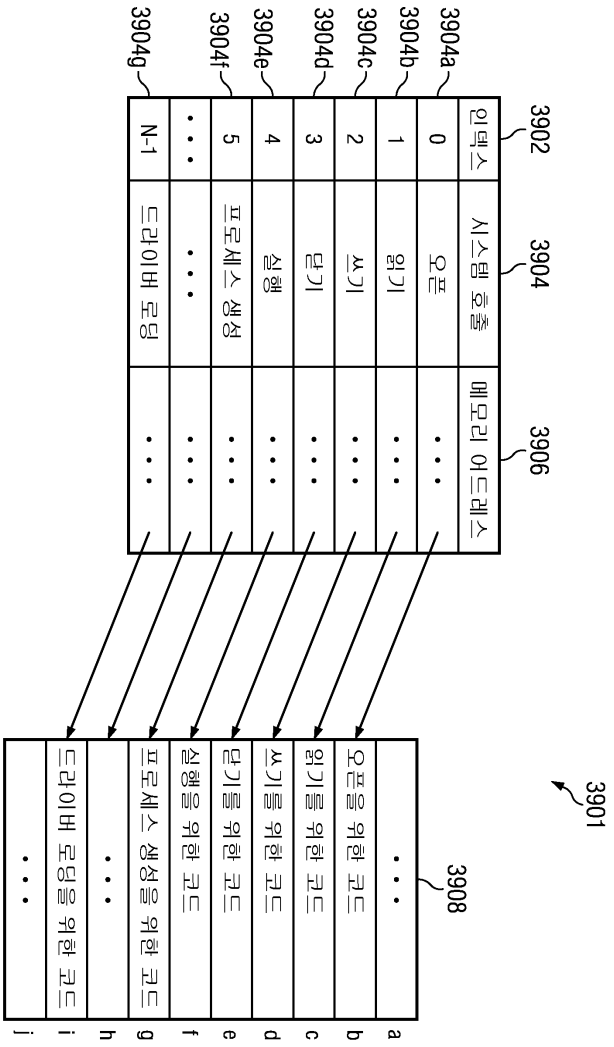
도면37


3700
시작
3705
제어 레지스터에 대한 레코드 시도 액세스를 인터셉트
3710
O/S 하위 레벨 프로세스 리스트 생성
3715
실행중인 커널 모드 프로세스들 판단
3720
O/S 레벨 프로세스 리스트를 생성
3725
각각의 프로세스의 프로세스 구조로부터 제어 레지스터 값들을 계산
3730
O/S 하위 레벨과 O/S 레벨 프로세스 리스트들 간 차이를 판단
3735
O/S 레벨 프로세스 리스트에서 누락된 프로세스들이 숨겨져 있는지를 판단
3740
메모리 변경에 대해 메모리 및 운영체제 검색
3745
메모리 변경사항들을 복구
3750
모든 숨겨진 커널 모드 프로세스들에 대해 반복
3755
모든 사용자 모드 프로세스들 나열
3760
사용자 레벨 프로세스 리스트 생성
3765
O/S 하위 레벨과 O/S 레벨 프로세스 리스트들 간 차이를 판단
3770
O/S 레벨 프로세스 리스트에서 누락된 프로세스들이 숨겨져 있는지를 판단
3775
메모리 변경에 대해 메모리 및 운영체제 검색
3780
메모리 변경사항들을 복구
3785
모든 숨겨진 사용자 모드 프로세스들에 대해 반복
종료

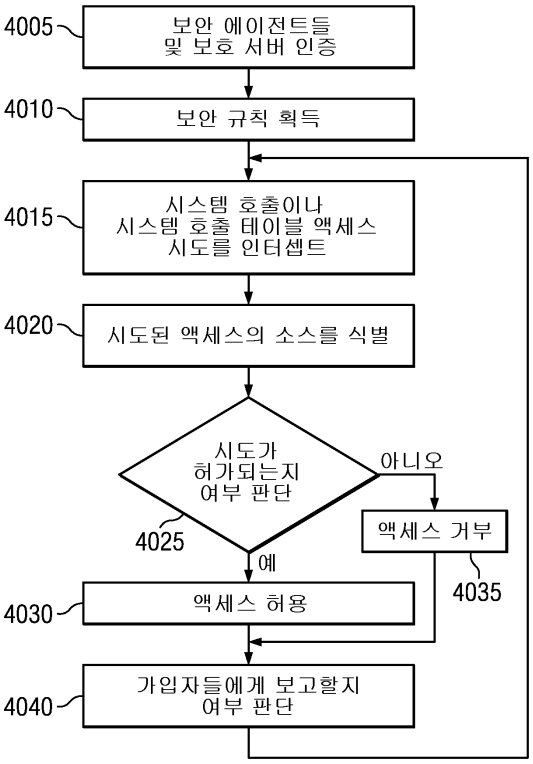
도면38


3800
3801
3810
애플리케이션
드라이버
3811
O/S
3804
시스템
호출 테이블
3813
3819
O/S 내 보안
에이전트
페이지
테이블
3806
O/S 하위 보안 에이전트
3820
3814
시스템 호출
트래퍼
보안 규칙들
3808
3802
프로세서
3816
3803
메모리
트리거된
이벤트
핸들러
3822
보호 서버
3818
가입자들
3812

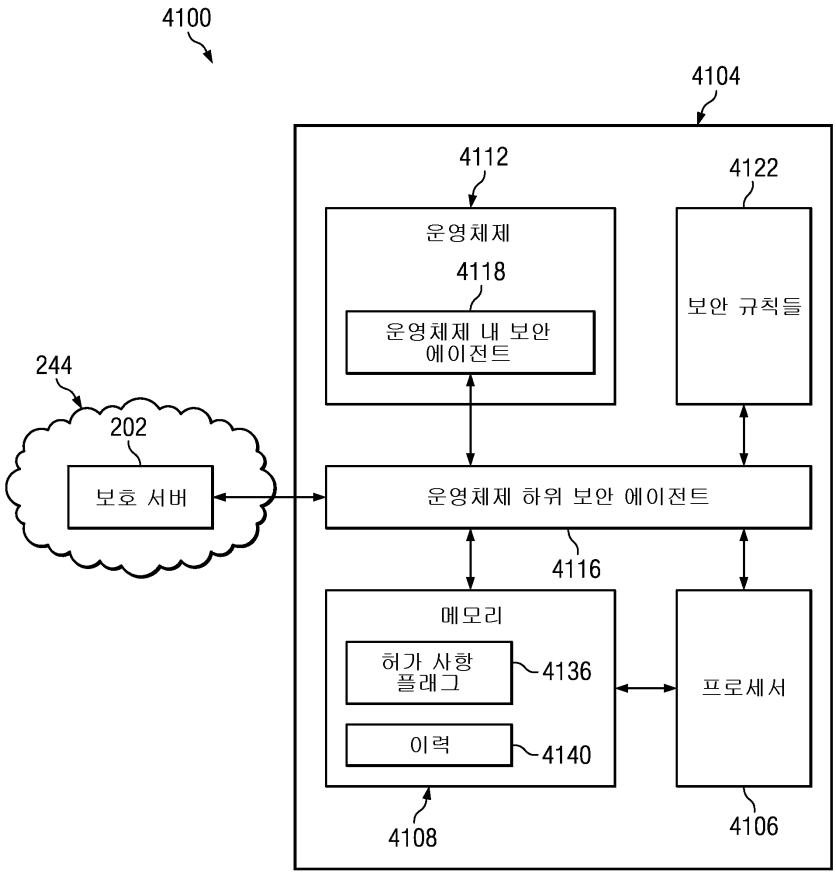
도면39


3901
3902
3904
3906
3908
인덱스
시스템 호출
메모리 어드레스
0
오픈
1
읽기
2
쓰기
3
닫기
4
실행
5
프로세스 생성
N-1
드라이버 로딩
3904a
3904b
3904c
3904d
3904e
3904f
3904g
오픈을 위한 코드
읽기를 위한 코드
쓰기를 위한 코드
닫기를 위한 코드
실행을 위한 코드
프로세스 생성을 위한 코드
드라이버 로딩을 위한 코드
a
b
c
d
e
f
g
h
i
j

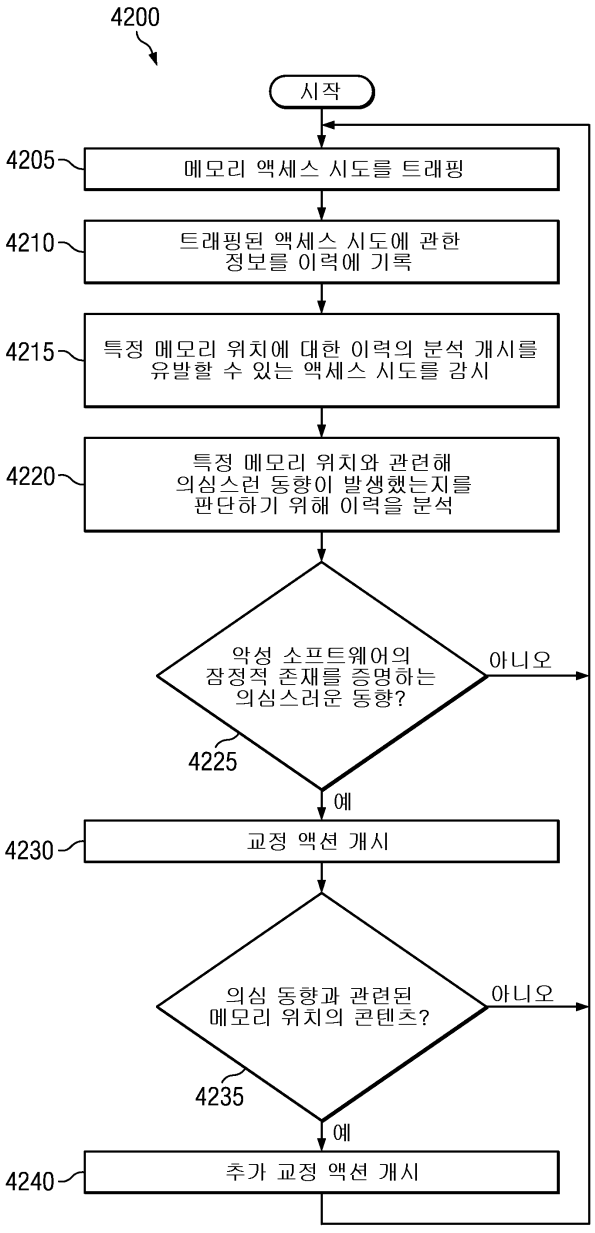
도면40


4005
보안 에이전트들
및 보호 서버 인증
4010
보안 규칙 획득
4015
시스템 호출이나
시스템 호출 테이블 액세스
시도를 인터셉트
4020
시도된 액세스의 소스를 식별
시도가
허가되는지
여부 판단
4025
아니오
액세스 거부
4035
예
4030
액세스 허용
4040
가입자들에게 보고할지
여부 판단

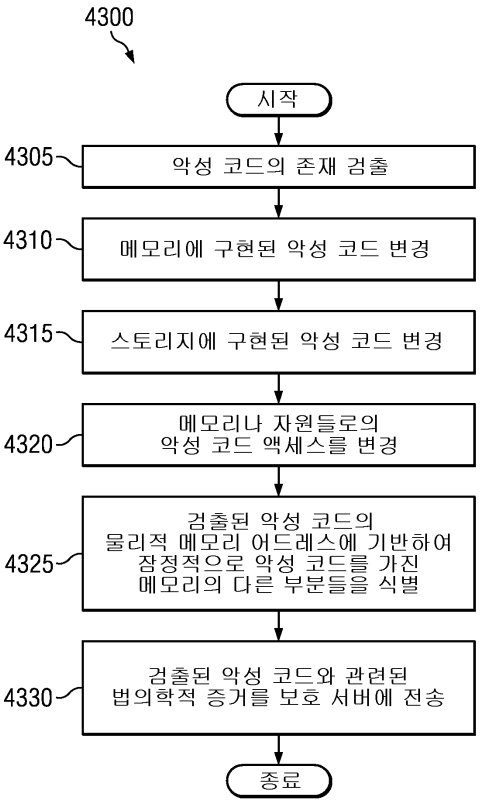
도면41


4100
4104
4112
운영체제
4118
운영체제 내 보안
에이전트
4122
보안 규칙들
244
202
보호 서버
운영체제 하위 보안 에이전트
4116
메모리
허가 사항
플래그
4136
이력
4140
프로세서
4108
4106

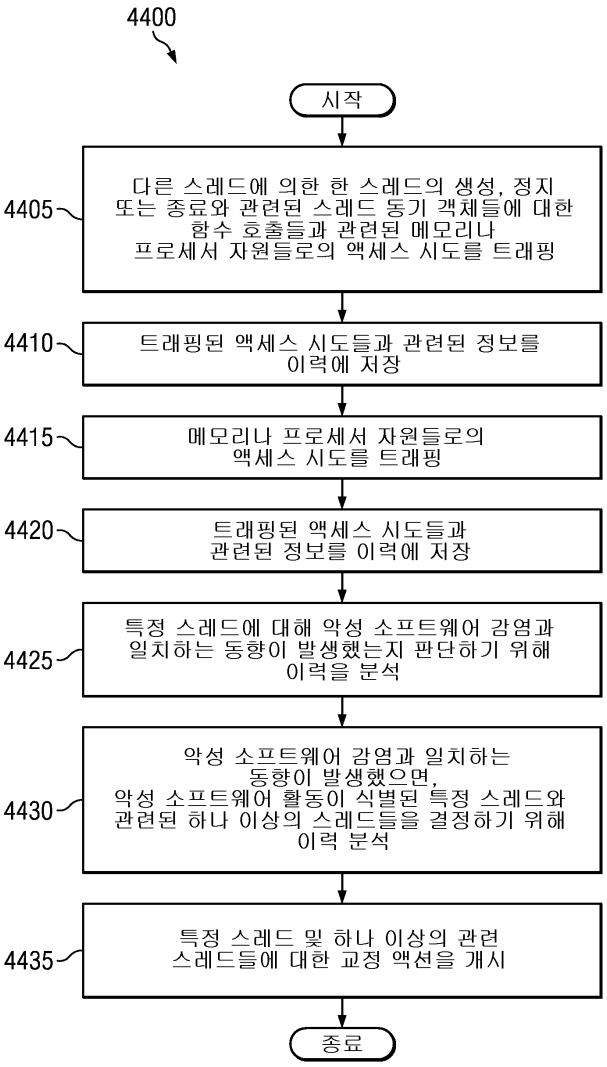
도면42


4200
시작
4205
메모리 액세스 시도를 트래핑
4210
트래핑된 액세스 시도에 관한 정보를 이력에 기록
4215
특정 메모리 위치에 대한 이력의 분석 개시를 유발할 수 있는 액세스 시도를 감시
4220
특정 메모리 위치와 관련해 의심스런 동향이 발생했는지를 판단하기 위해 이력을 분석
악성 소프트웨어의 잠정적 존재를 증명하는 의심스러운 동향?
아니오
4225
예
4230
교정 액션 개시
의심 동향과 관련된 메모리 위치의 콘텐츠?
아니오
4235
예
4240
추가 교정 액션 개시

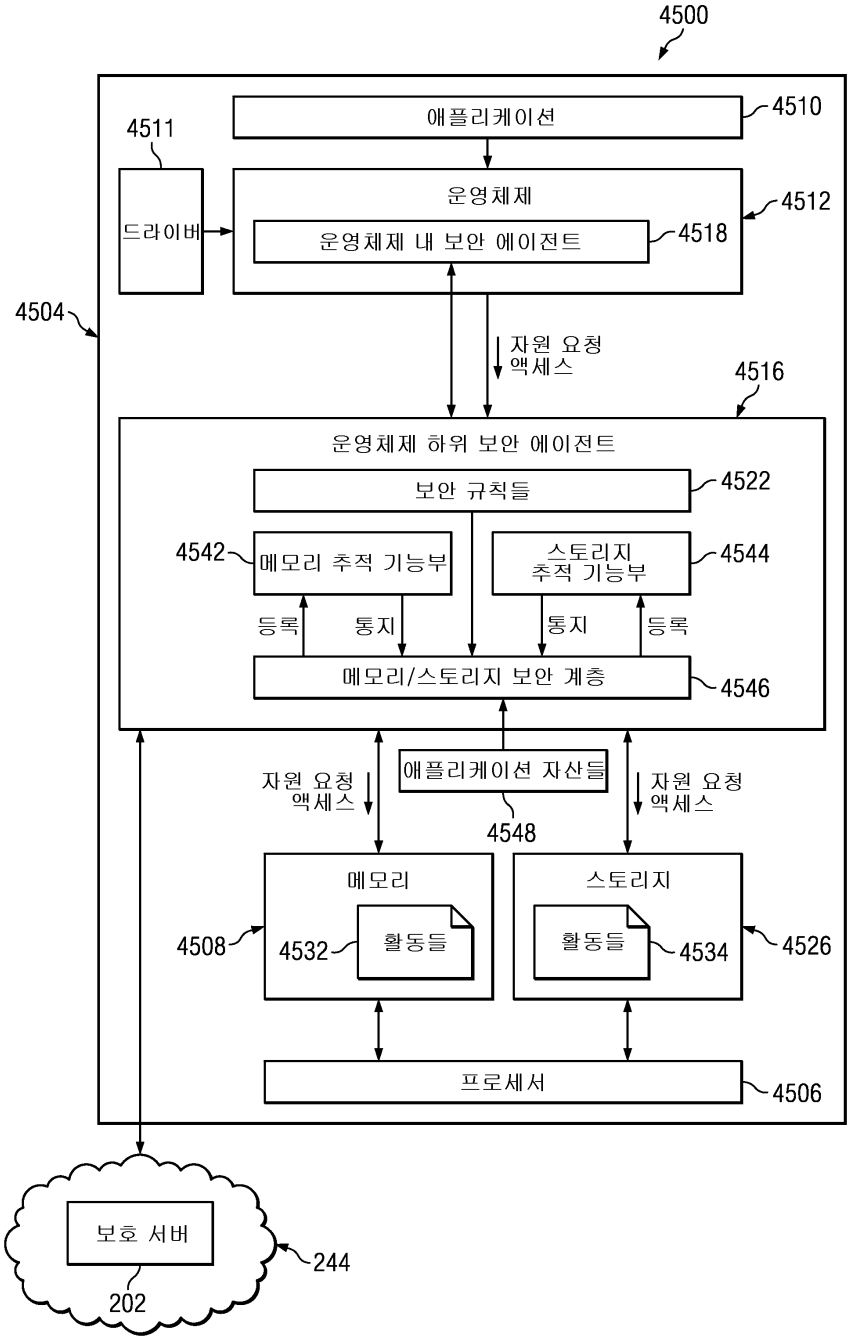
도면43


4300
시작
4305
악성 코드의 존재 검출
4310
메모리에 구현된 악성 코드 변경
4315
스토리지에 구현된 악성 코드 변경
4320
메모리나 자원들로의
악성 코드 액세스를 변경
4325
검출된 악성 코드의
물리적 메모리 어드레스에 기반하여
잠정적으로 악성 코드를 가진
메모리의 다른 부분들을 식별
4330
검출된 악성 코드와 관련된
법의학적 증거를 보호 서버에 전송
종료

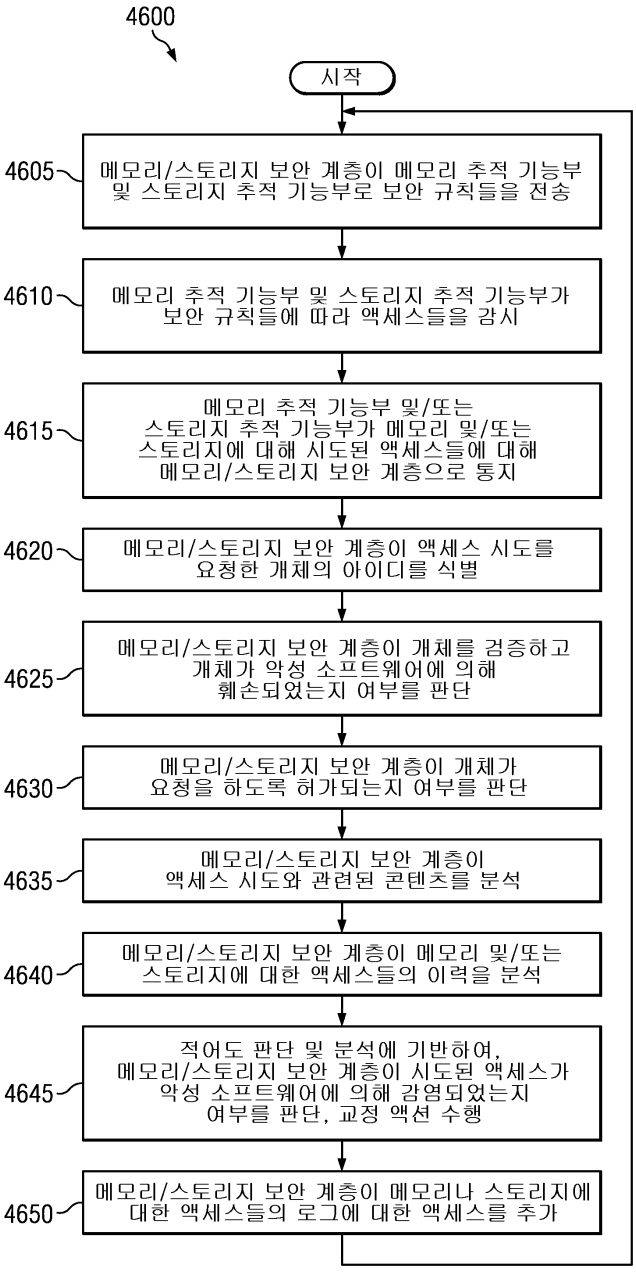
도면44


4400
시작
다른 스레드에 의한 한 스레드의 생성, 정지 또는 종료와 관련된 스레드 동기 객체들에 대한 함수 호출들과 관련된 메모리나 프로세서 자원들로의 액세스 시도를 트래핑
4405
트래핑된 액세스 시도들과 관련된 정보를 이력에 저장
4410
메모리나 프로세서 자원들로의 액세스 시도를 트래핑
4415
트래핑된 액세스 시도들과 관련된 정보를 이력에 저장
4420
특정 스레드에 대해 악성 소프트웨어 감염과 일치하는 동향이 발생했는지 판단하기 위해 이력을 분석
4425
악성 소프트웨어 감염과 일치하는 동향이 발생했으면, 악성 소프트웨어 활동이 식별된 특정 스레드와 관련된 하나 이상의 스레드들을 결정하기 위해 이력 분석
4430
특정 스레드 및 하나 이상의 관련 스레드들에 대한 교정 액션을 개시
4435
종료

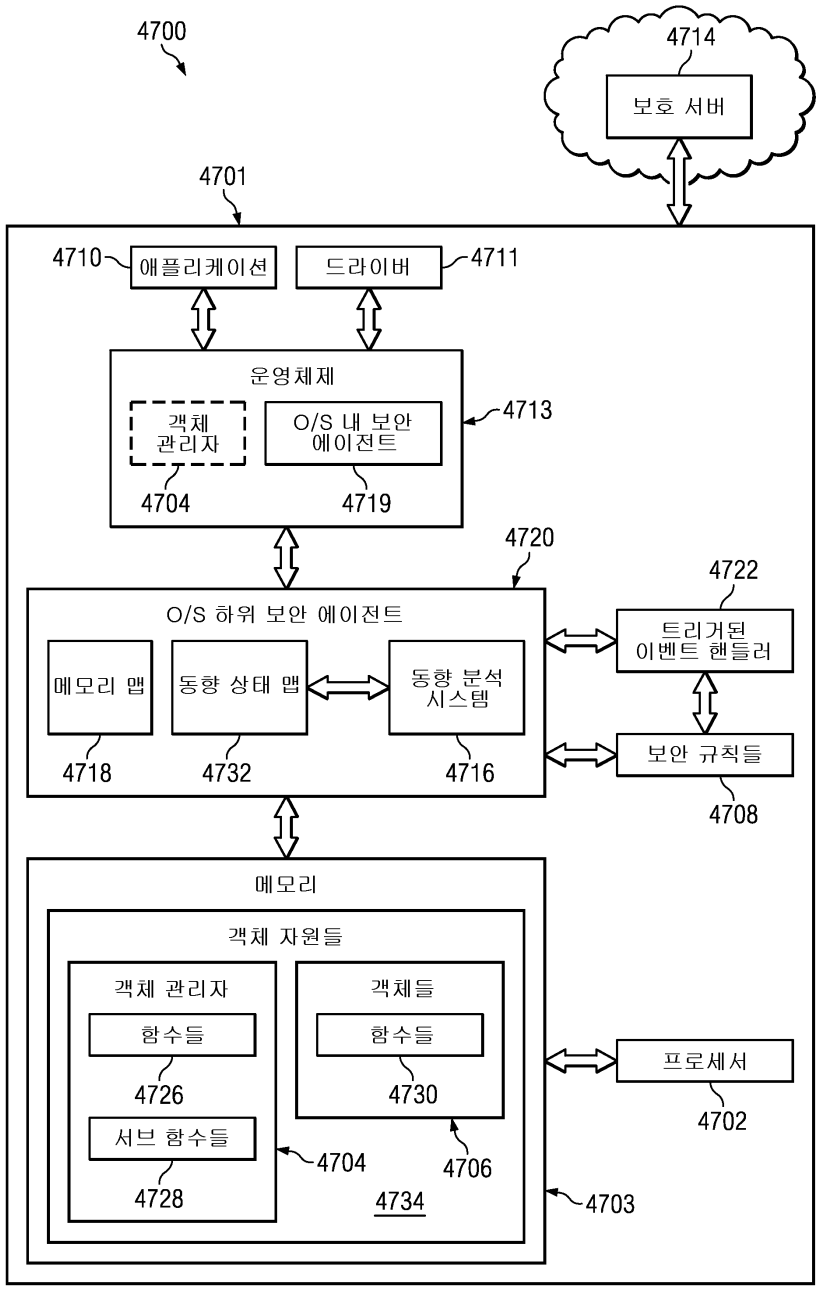
도면45


4500
애플리케이션
4510
4511
드라이버
운영체제
4512
운영체제 내 보안 에이전트
4518
4504
자원 요청
액세스
4516
운영체제 하위 보안 에이전트
보안 규칙들
4522
4542
메모리 추적 기능부
스토리지
추적 기능부
4544
등록
통지
통지
등록
메모리/스토리지 보안 계층
4546
자원 요청
액세스
애플리케이션 자산들
4548
자원 요청
액세스
메모리
스토리지
4508
4532
활동들
활동들
4534
4526
프로세서
4506
보호 서버
202
244

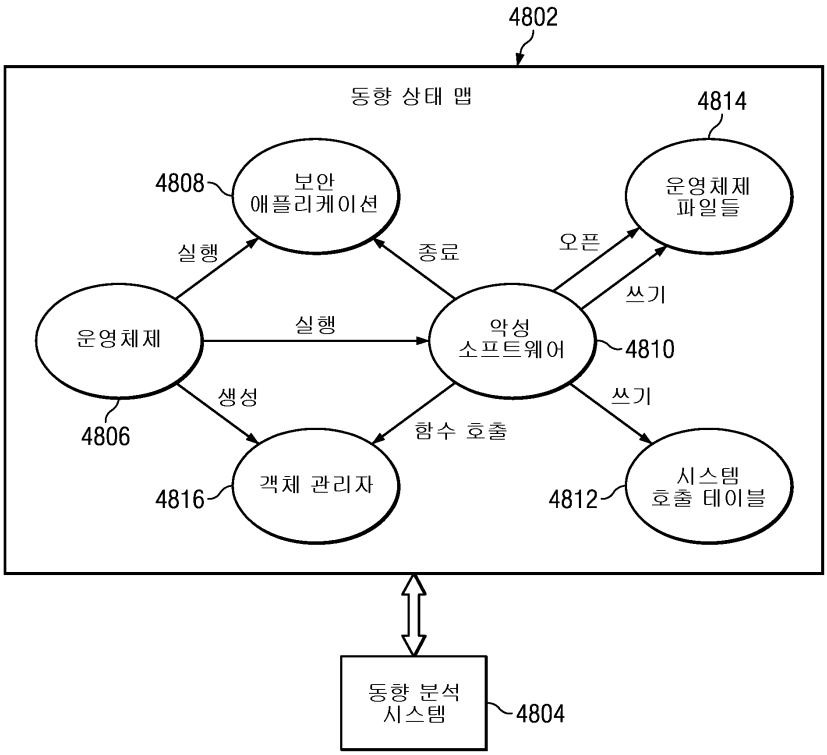
도면46


4600
시작
4605
메모리/스토리지 보안 계층이 메모리 추적 기능부 및 스토리지 추적 기능부로 보안 규칙들을 전송
4610
메모리 추적 기능부 및 스토리지 추적 기능부가 보안 규칙들에 따라 액세스들을 감시
4615
메모리 추적 기능부 및/또는 스토리지 추적 기능부가 메모리 및/또는 스토리지에 대해 시도된 액세스들에 대해 메모리/스토리지 보안 계층으로 통지
4620
메모리/스토리지 보안 계층이 액세스 시도를 요청한 개체의 아이디를 식별
4625
메모리/스토리지 보안 계층이 개체를 검증하고 개체가 악성 소프트웨어에 의해 훼손되었는지 여부를 판단
4630
메모리/스토리지 보안 계층이 개체가 요청을 하도록 허가되는지 여부를 판단
4635
메모리/스토리지 보안 계층이 액세스 시도와 관련된 콘텐츠를 분석
4640
메모리/스토리지 보안 계층이 메모리 및/또는 스토리지에 대한 액세스들의 이력을 분석
4645
적어도 판단 및 분석에 기반하여, 메모리/스토리지 보안 계층이 시도된 액세스가 악성 소프트웨어에 의해 감염되었는지 여부를 판단, 교정 액션 수행
4650
메모리/스토리지 보안 계층이 메모리나 스토리지에 대한 액세스들의 로그에 대한 액세스를 추가

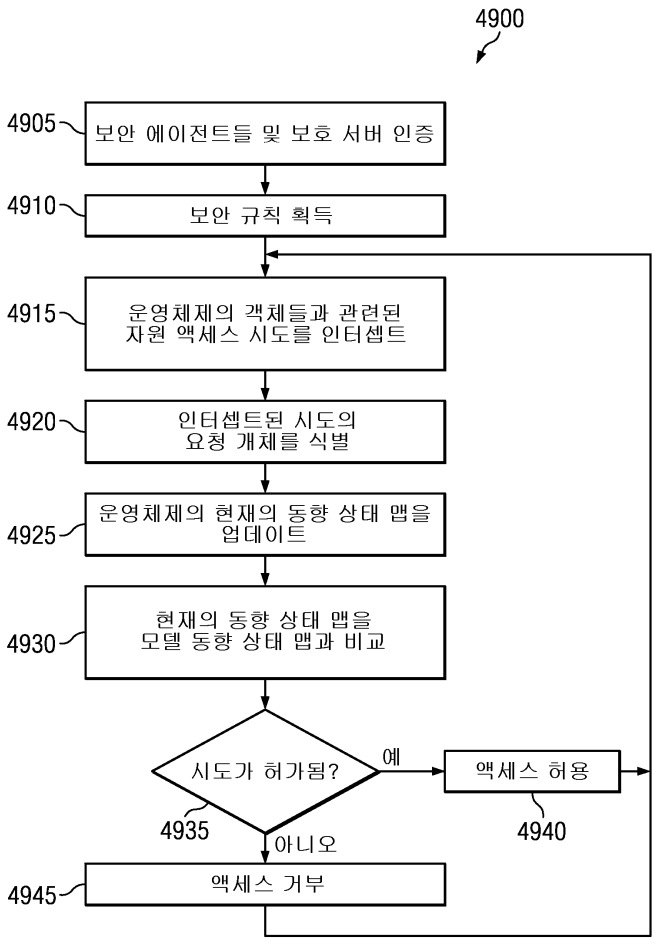
도면47


4700
4714
보호 서버
4701
4710
애플리케이션
드라이버
4711
운영체제
객체 관리자
O/S 내 보안 에이전트
4713
4704
4719
4720
4722
O/S 하위 보안 에이전트
트리거된 이벤트 핸들러
메모리 맵
동향 상태 맵
동향 분석 시스템
보안 규칙들
4718
4732
4716
4708
메모리
객체 자원들
객체 관리자
함수들
객체들
함수들
프로세서
4726
4730
4702
서브 함수들
4704
4706
4728
4734
4703

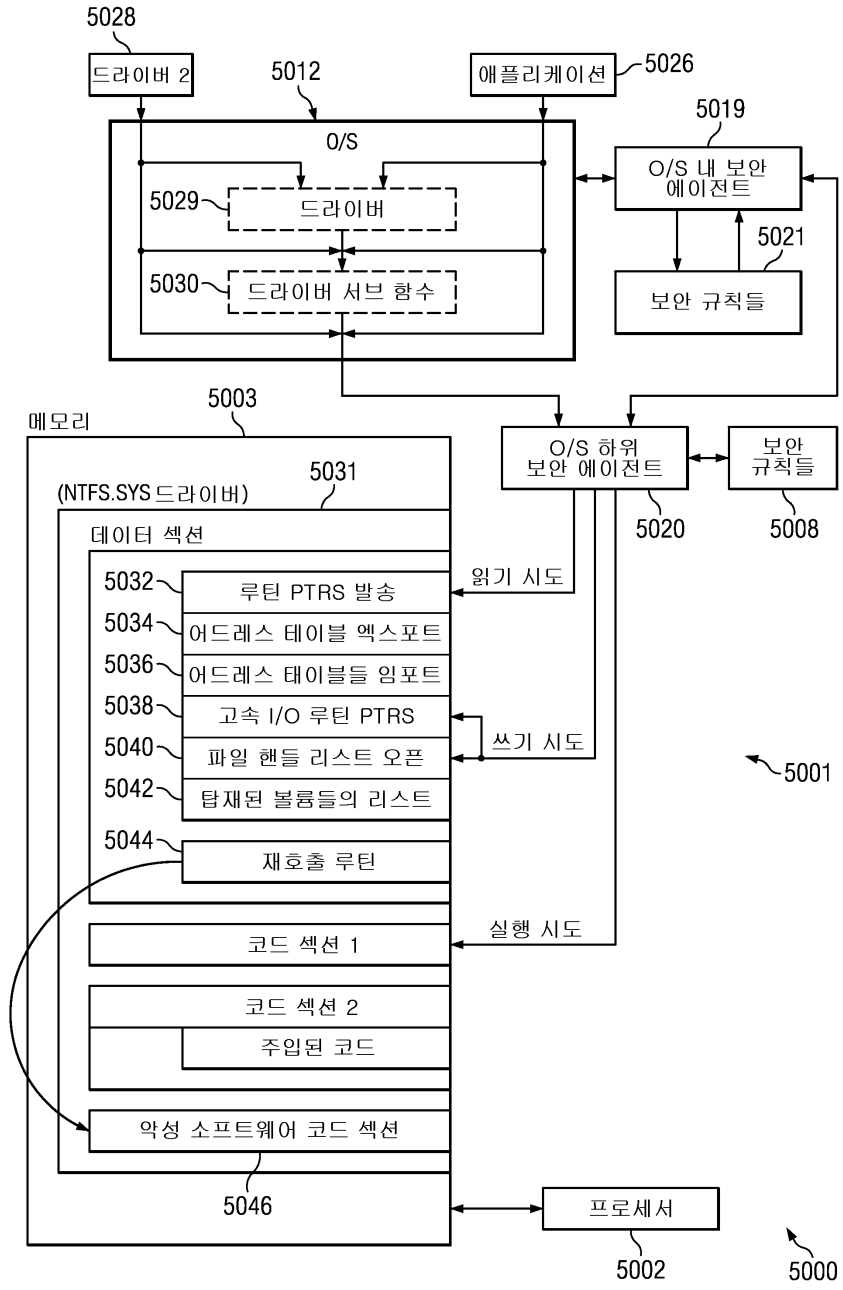
도면48


4802
동향 상태 맵
4814
4808
보안
애플리케이션
운영체제
파일들
실행
종료
오픈
쓰기
운영체제
실행
악성
소프트웨어
4810
생성
쓰기
4806
함수 호출
4816
객체 관리자
4812
시스템
호출 테이블
동향 분석
시스템
4804

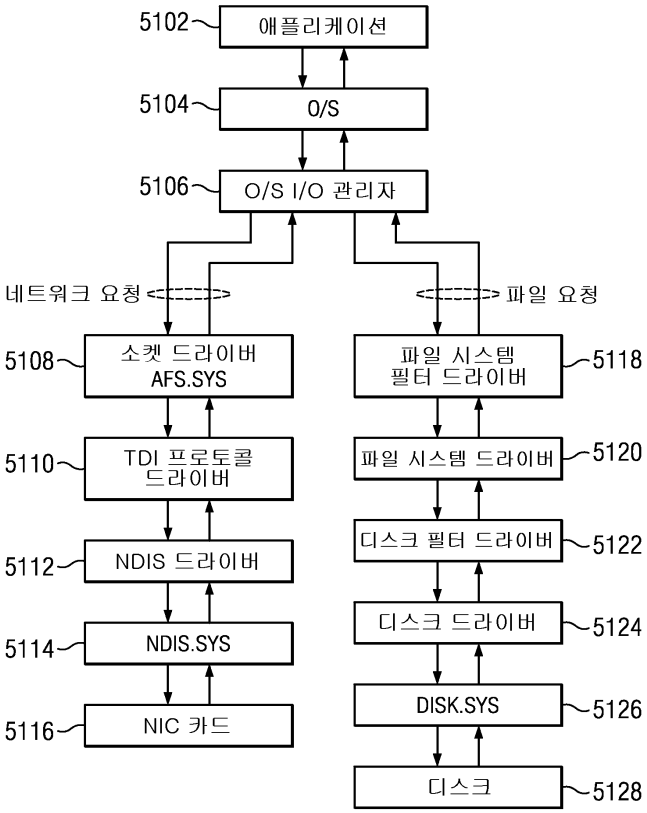
도면49


4900
4905
보안 에이전트들 및 보호 서버 인증
4910
보안 규칙 획득
4915
운영체제의 객체들과 관련된
자원 액세스 시도를 인터셉트
4920
인터셉트된 시도의
요청 개체를 식별
4925
운영체제의 현재의 동향 상태 맵을
업데이트
4930
현재의 동향 상태 맵을
모델 동향 상태 맵과 비교
시도가 허가됨?
예
액세스 허용
4935
4940
아니오
4945
액세스 거부

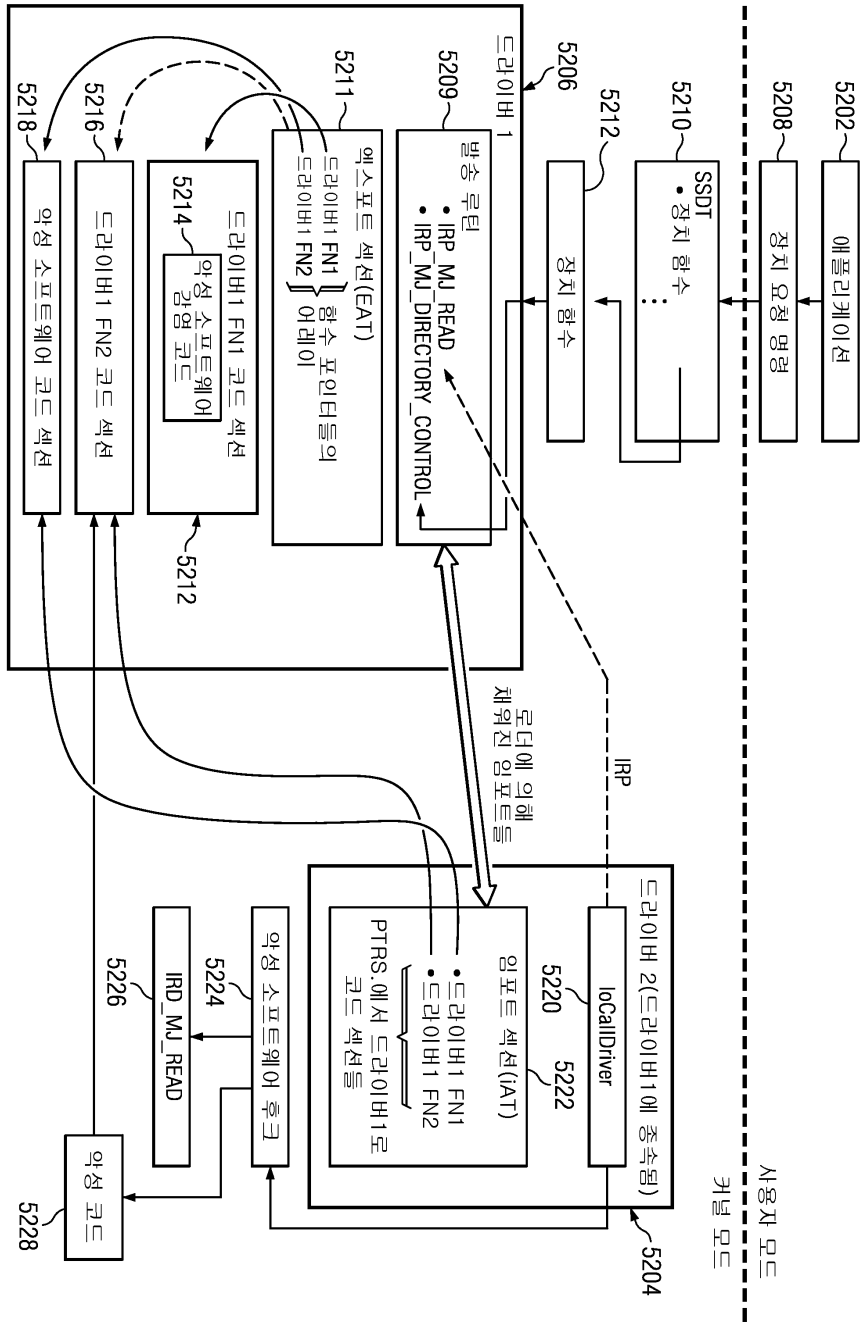
도면50


5028
드라이버 2
5012
애플리케이션
5026
5019
O/S
O/S 내 보안 에이전트
5029
드라이버
5021
5030
드라이버 서브 함수
보안 규칙들
5003
메모리
O/S 하위 보안 에이전트
보안 규칙들
5031
(NTFS.SYS 드라이버)
5020
5008
데이터 섹션
읽기 시도
5032
루틴 PTRS 발송
5034
어드레스 테이블 엑스포트
5036
어드레스 테이블들 임포트
5038
고속 I/O 루틴 PTRS
쓰기 시도
5040
파일 핸들 리스트 오픈
5001
5042
탑재된 볼륨들의 리스트
5044
재호출 루틴
실행 시도
코드 섹션 1
코드 섹션 2
주입된 코드
악성 소프트웨어 코드 섹션
5046
프로세서
5002
5000

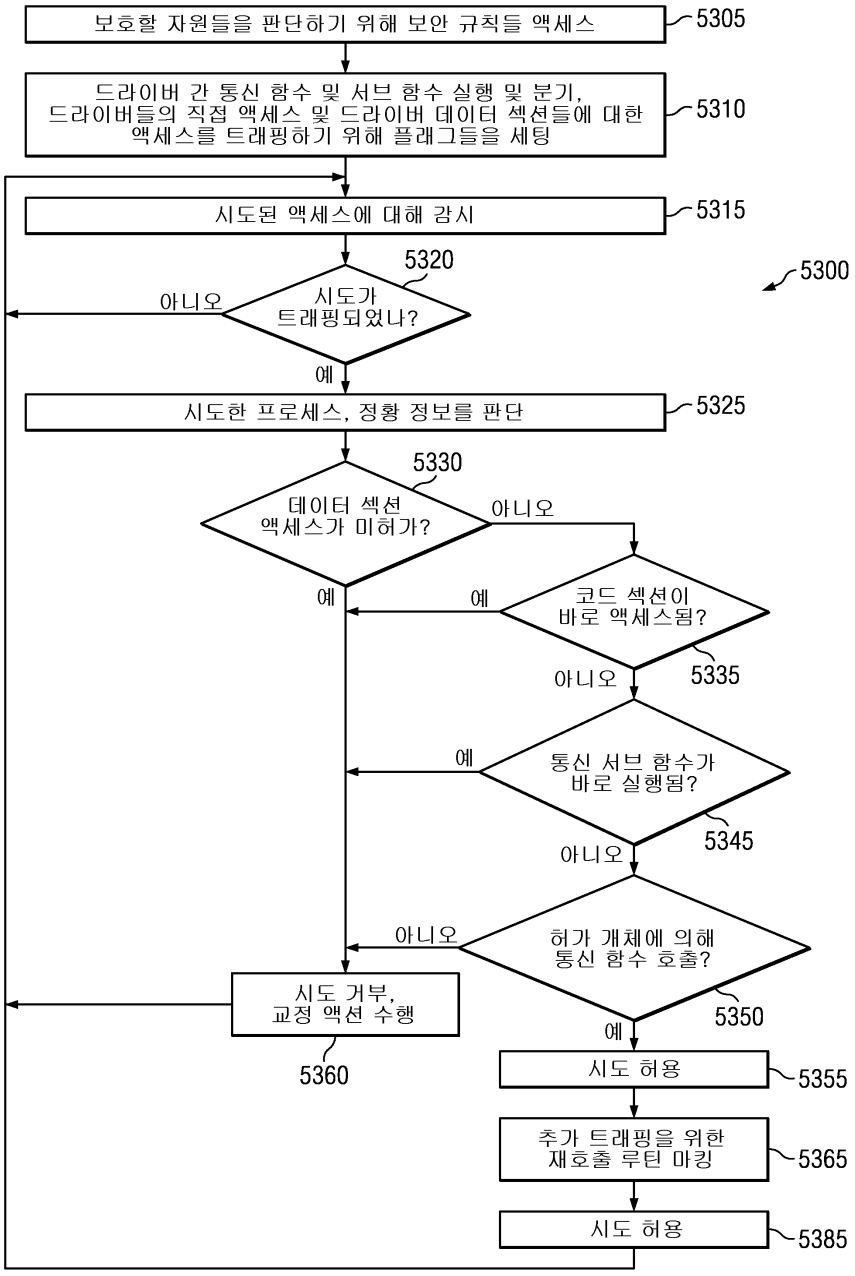
도면51


5102
애플리케이션
5104
O/S
5106
O/S I/O 관리자
네트워크 요청
파일 요청
5108
소켓 드라이버
AFS.SYS
5110
TDI 프로토콜
드라이버
5112
NDIS 드라이버
5114
NDIS.SYS
5116
NIC 카드
파일 시스템
필터 드라이버
5118
파일 시스템 드라이버
5120
디스크 필터 드라이버
5122
디스크 드라이버
5124
DISK.SYS
5126
디스크
5128

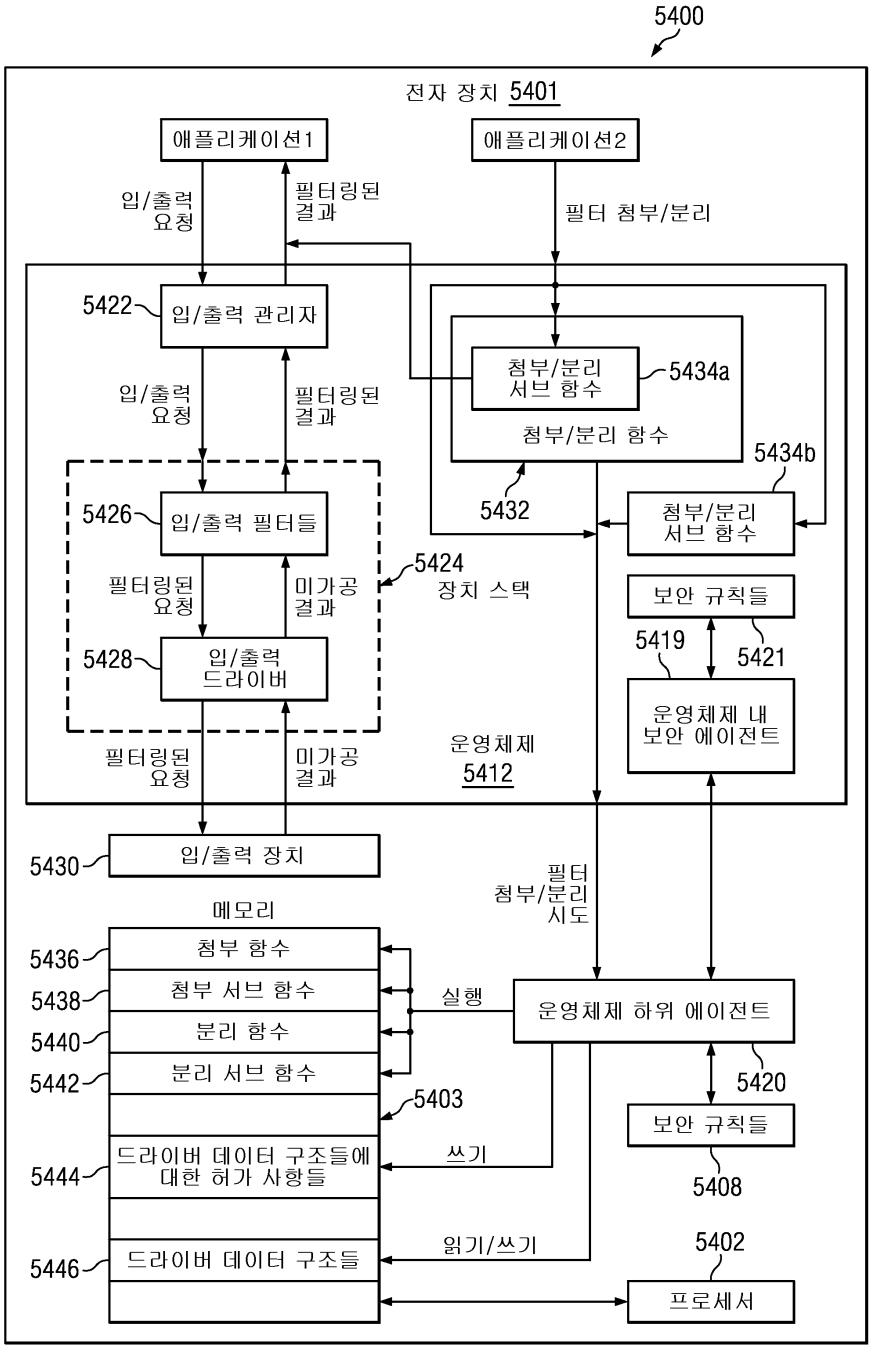
도면52


5202
애플리케이션
5208
장치 요청 명령
5210
SSDT
• 장치 함수
5212
장치 함수
5206
드라이버 1
5209
발송 루틴
• IRP_MJ_READ
• IRP_MJ_DIRECTORY_CONTROL
5211
엑스포트 섹션(EAT)
드라이버1 FN1
드라이버1 FN2
함수 포인터들의 어레이
5216
드라이버1 FN2 코드 섹션
5218
악성 소프트웨어 코드 섹션
5214
드라이버1 FN1 코드 섹션
악성 소프트웨어 감염 코드
5212
사용자 모드
커널 모드
IRP
로더에 의해 채워진 임포트들
드라이버 2(드라이버1에 종속됨)
5204
IoCallDriver
5220
임포트 섹션(iAT)
5222
• 드라이버1 FN1
• 드라이버1 FN2
PTRS.에서 드라이버1로 코드 섹션들
악성 소프트웨어 후크
5224
IRD_MJ_READ
5226
악성 코드
5228

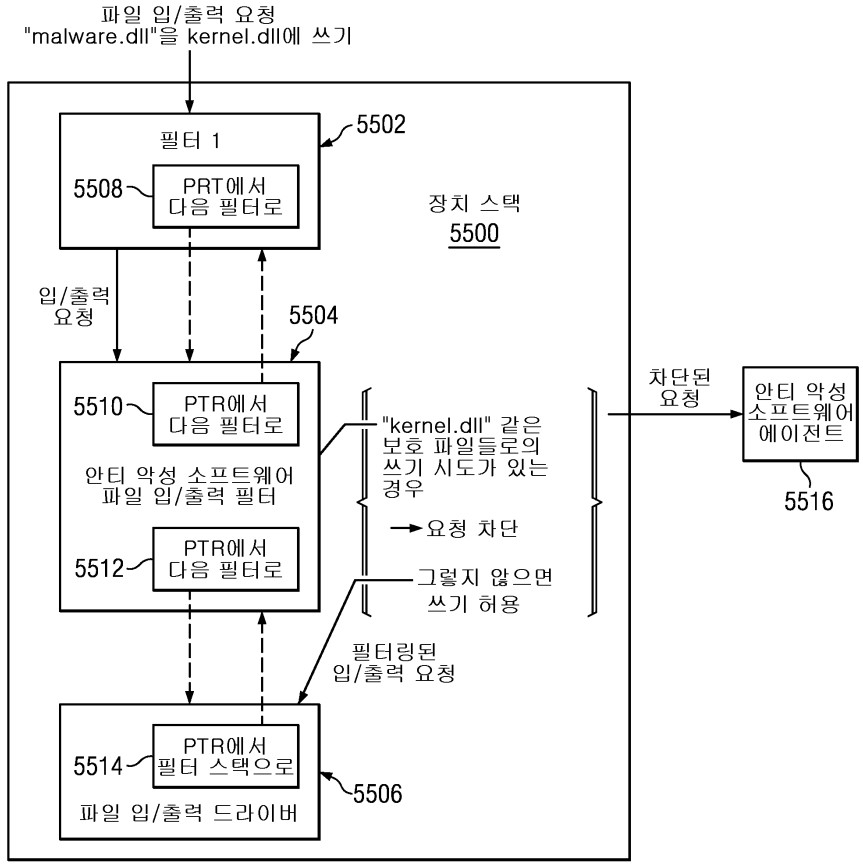
도면53


보호할 자원들을 판단하기 위해 보안 규칙들 액세스
5305
드라이버 간 통신 함수 및 서브 함수 실행 및 분기, 드라이버들의 직접 액세스 및 드라이버 데이터 섹션들에 대한 액세스를 트래핑하기 위해 플래그들을 세팅
5310
시도된 액세스에 대해 감시
5315
5300
5320
시도가 트래핑되었나?
아니오
예
시도한 프로세스, 정황 정보를 판단
5325
5330
데이터 섹션 액세스가 미허가?
아니오
예
예
코드 섹션이 바로 액세스됨?
5335
아니오
예
통신 서브 함수가 바로 실행됨?
5345
아니오
아니오
허가 개체에 의해 통신 함수 호출?
5350
예
시도 거부, 교정 액션 수행
5360
시도 허용
5355
추가 트래핑을 위한 재호출 루틴 마킹
5365
시도 허용
5385

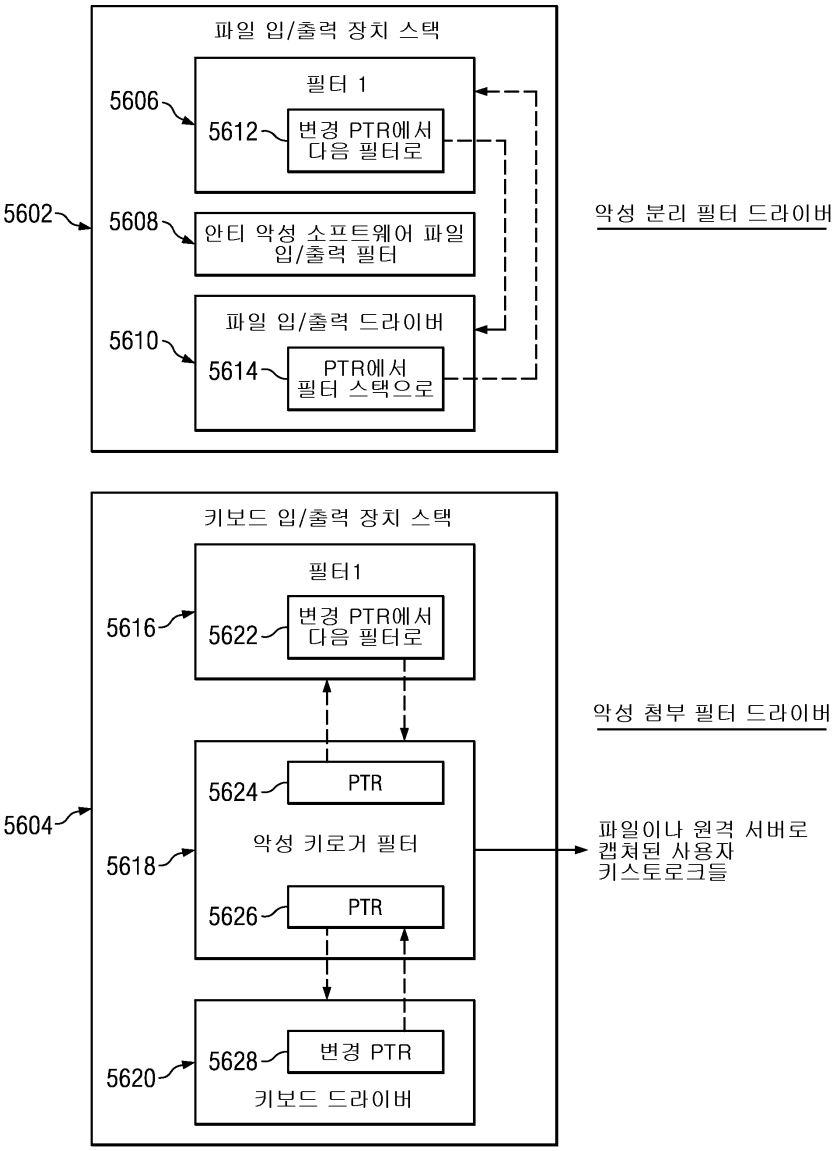
도면54


5400
전자 장치 5401
애플리케이션1
애플리케이션2
입/출력 요청
필터링된 결과
필터 첨부/분리
5422
입/출력 관리자
첨부/분리 서브 함수
5434a
첨부/분리 함수
5434b
5432
첨부/분리 서브 함수
5426
입/출력 필터들
5424
장치 스택
보안 규칙들
5419
5421
5428
입/출력 드라이버
운영체제 내 보안 에이전트
운영체제 5412
5430
입/출력 장치
필터 첨부/분리 시도
메모리
5436
첨부 함수
5438
첨부 서브 함수
5440
분리 함수
5442
분리 서브 함수
실행
운영체제 하위 에이전트
5420
5403
보안 규칙들
5444
드라이버 데이터 구조들에 대한 허가 사항들
쓰기
5408
5446
드라이버 데이터 구조들
읽기/쓰기
5402
프로세서

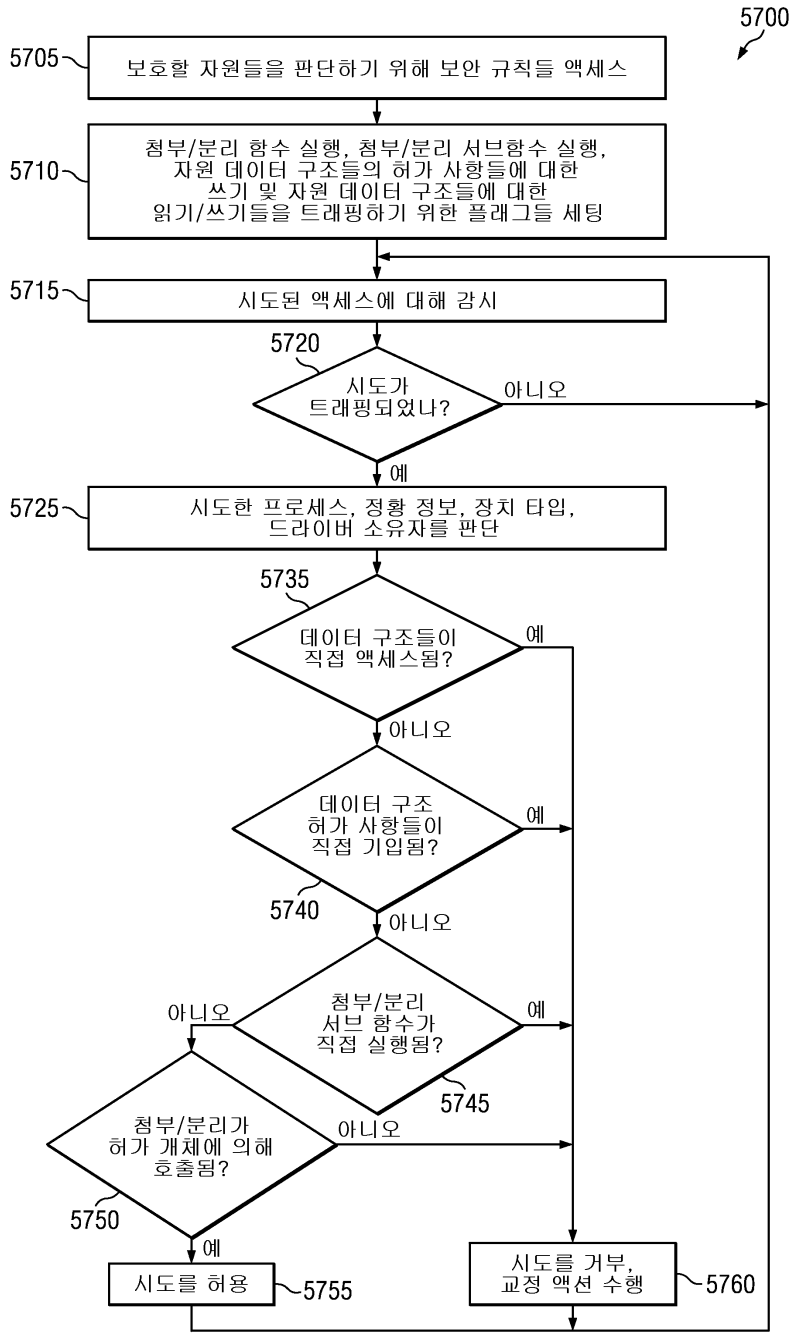
도면55


파일 입/출력 요청
"malware.dll"을 kernel.dll에 쓰기
필터 1
5502
5508
PRT에서
다음 필터로
장치 스택
5500
입/출력
요청
5504
5510
PTR에서
다음 필터로
안티 악성 소프트웨어
파일 입/출력 필터
5512
PTR에서
다음 필터로
"kernel.dll" 같은
보호 파일들로의
쓰기 시도가 있는
경우
요청 차단
그렇지 않으면
쓰기 허용
필터링된
입/출력 요청
차단된
요청
안티 악성
소프트웨어
에이전트
5516
5514
PTR에서
필터 스택으로
5506
파일 입/출력 드라이버

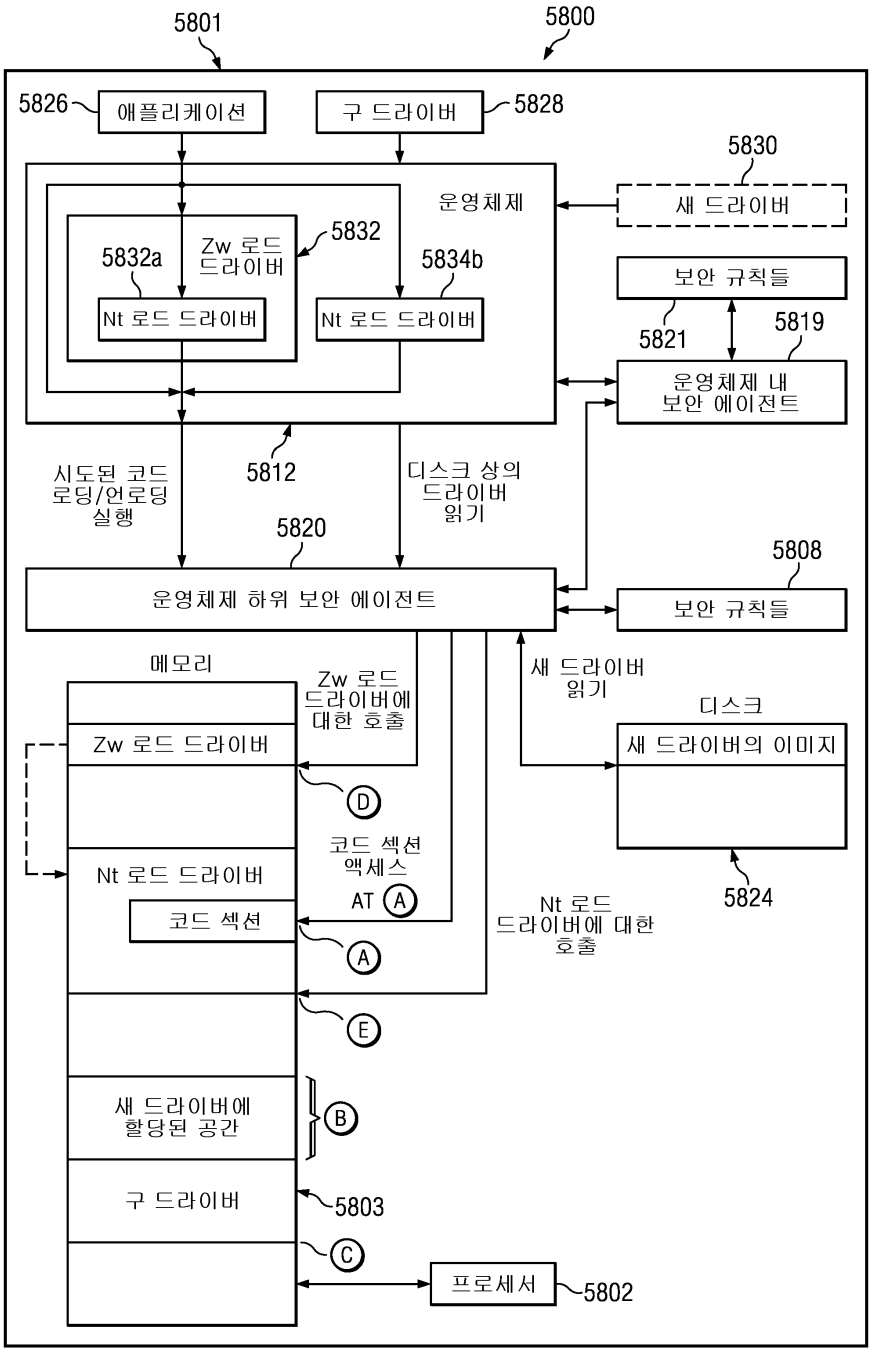
도면56


파일 입/출력 장치 스택
필터 1
5606
5612
변경 PTR에서 다음 필터로
5602
5608
안티 악성 소프트웨어 파일 입/출력 필터
악성 분리 필터 드라이버
파일 입/출력 드라이버
5610
5614
PTR에서 필터 스택으로
키보드 입/출력 장치 스택
필터1
5616
5622
변경 PTR에서 다음 필터로
악성 첨부 필터 드라이버
5624
PTR
5604
악성 키로거 필터
파일이나 원격 서버로 캡쳐된 사용자 키스토로크들
5618
5626
PTR
5620
5628
변경 PTR
키보드 드라이버

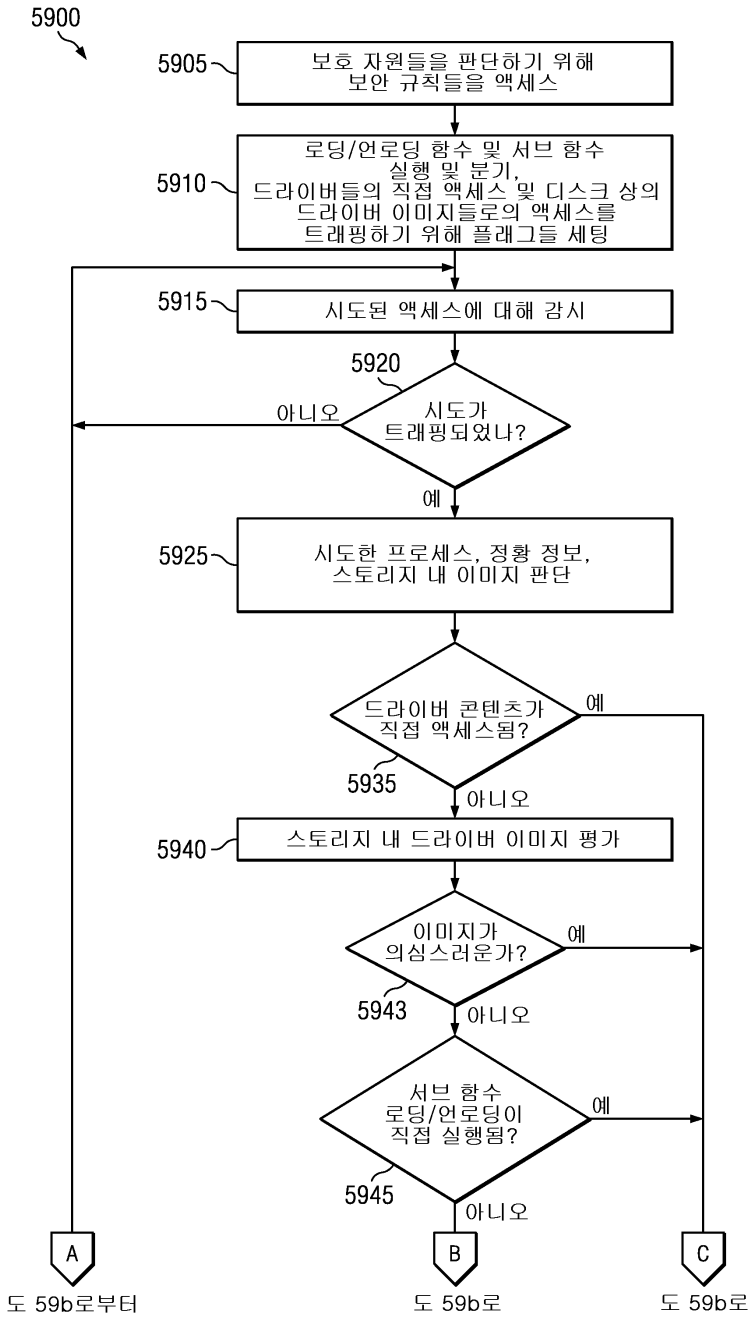
도면57


5700
5705
보호할 자원들을 판단하기 위해 보안 규칙들 액세스
5710
첨부/분리 함수 실행, 첨부/분리 서브함수 실행,
자원 데이터 구조들의 허가 사항들에 대한
쓰기 및 자원 데이터 구조들에 대한
읽기/쓰기들을 트래핑하기 위한 플래그들 세팅
5715
시도된 액세스에 대해 감시
5720
시도가
트래핑되었나?
아니오
예
5725
시도한 프로세스, 정황 정보, 장치 타입,
드라이버 소유자를 판단
5735
데이터 구조들이
직접 액세스됨?
예
아니오
데이터 구조
허가 사항들이
직접 기입됨?
예
5740
아니오
첨부/분리
서브 함수가
직접 실행됨?
예
아니오
5745
첨부/분리가
허가 개체에 의해
호출됨?
아니오
5750
예
시도를 허용
5755
시도를 거부,
교정 액션 수행
5760

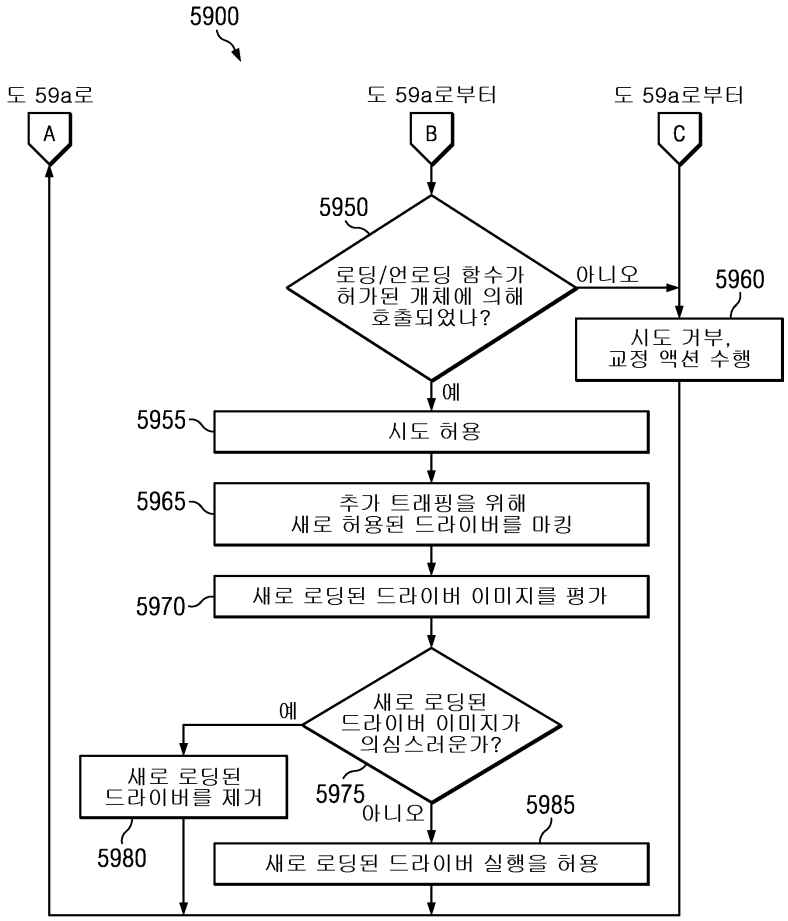
도면58


5801
5800
5826
애플리케이션
구 드라이버
5828
5830
운영체제
새 드라이버
5832
Zw 로드 드라이버
5832a
5834b
Nt 로드 드라이버
Nt 로드 드라이버
보안 규칙들
5821
5819
운영체제 내 보안 에이전트
시도된 코드 로딩/언로딩 실행
5812
디스크 상의 드라이버 읽기
5820
5808
운영체제 하위 보안 에이전트
보안 규칙들
메모리
Zw 로드 드라이버에 대한 호출
새 드라이버 읽기
디스크
Zw 로드 드라이버
새 드라이버의 이미지
D
코드 섹션 액세스
Nt 로드 드라이버
AT A
5824
코드 섹션
Nt 로드 드라이버에 대한 호출
A
E
새 드라이버에 할당된 공간
B
구 드라이버
5803
C
프로세서
5802

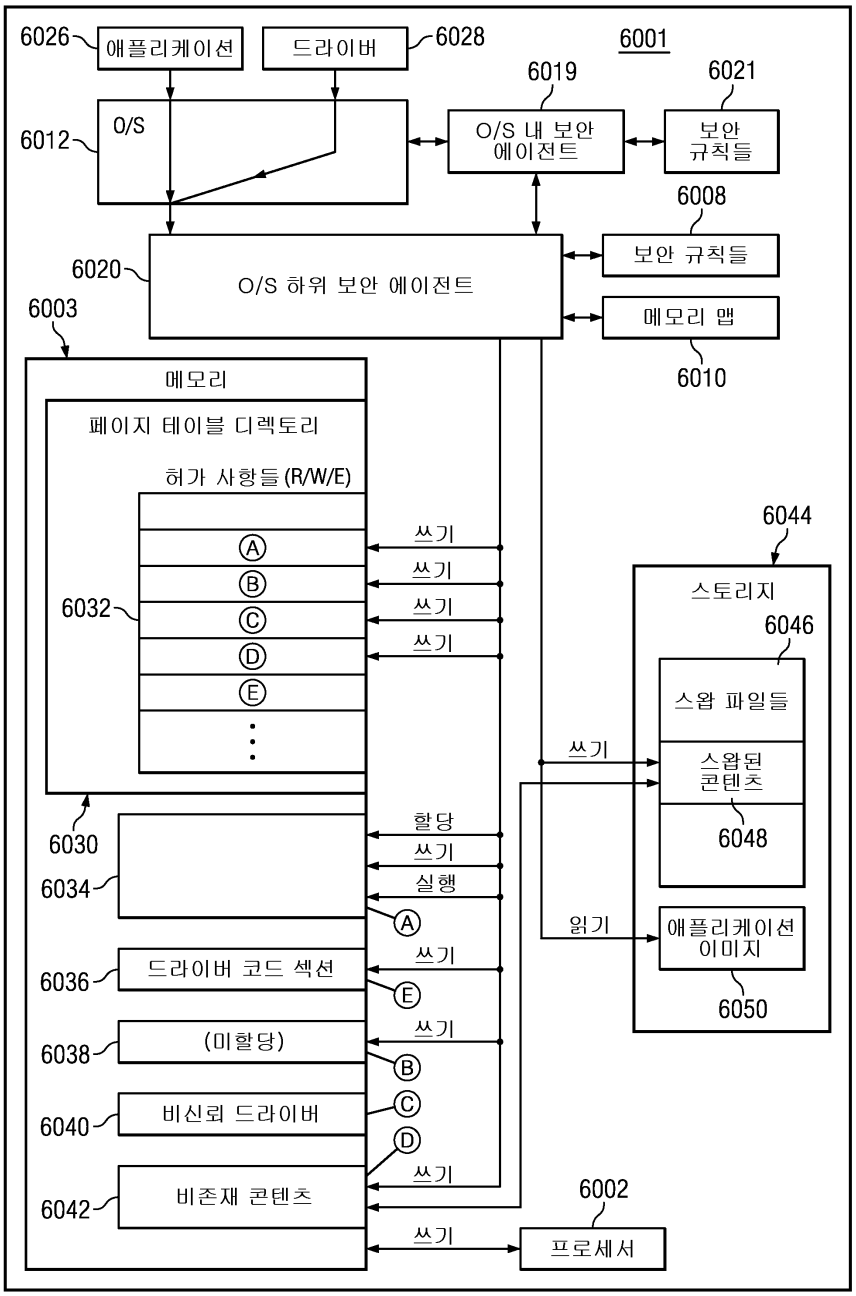
도면59a


5900
5905
보호 자원들을 판단하기 위해 보안 규칙들을 액세스
5910
로딩/언로딩 함수 및 서브 함수 실행 및 분기, 드라이버들의 직접 액세스 및 디스크 상의 드라이버 이미지들로의 액세스를 트래핑하기 위해 플래그들 세팅
5915
시도된 액세스에 대해 감시
5920
시도가 트래핑되었나?
아니오
예
5925
시도한 프로세스, 정황 정보, 스토리지 내 이미지 판단
드라이버 콘텐츠가 직접 액세스됨?
예
5935
아니오
5940
스토리지 내 드라이버 이미지 평가
이미지가 의심스러운가?
예
5943
아니오
서브 함수 로딩/언로딩이 직접 실행됨?
예
5945
아니오
A
도 59b로부터
B
도 59b로
C
도 59b로

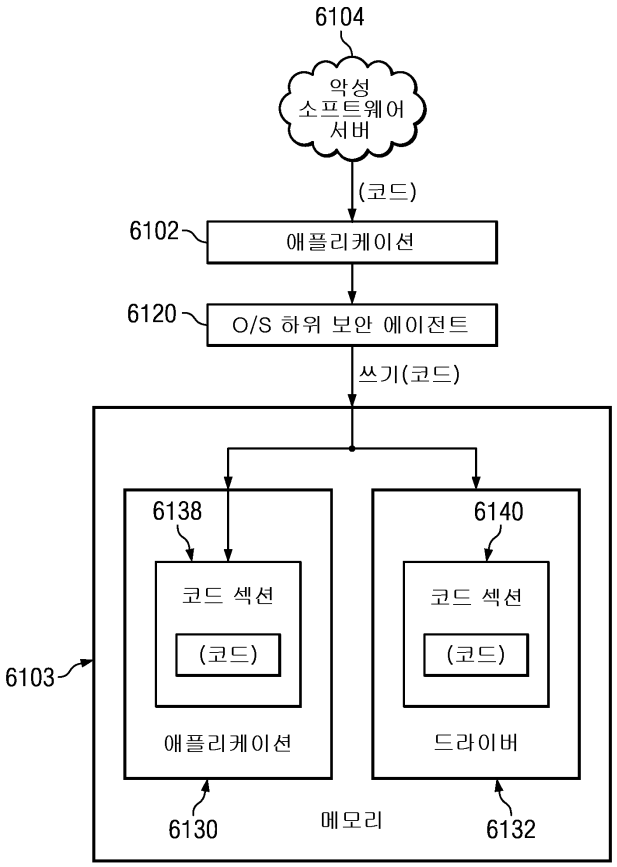
도면59b


5900
도 59a로
A
도 59a로부터
B
도 59a로부터
C
5950
로딩/언로딩 함수가 허가된 개체에 의해 호출되었나?
아니오
5960
시도 거부, 교정 액션 수행
예
5955
시도 허용
5965
추가 트래핑을 위해 새로 허용된 드라이버를 마킹
5970
새로 로딩된 드라이버 이미지를 평가
새로 로딩된 드라이버 이미지가 의심스러운가?
예
5975
아니오
새로 로딩된 드라이버를 제거
5980
5985
새로 로딩된 드라이버 실행을 허용

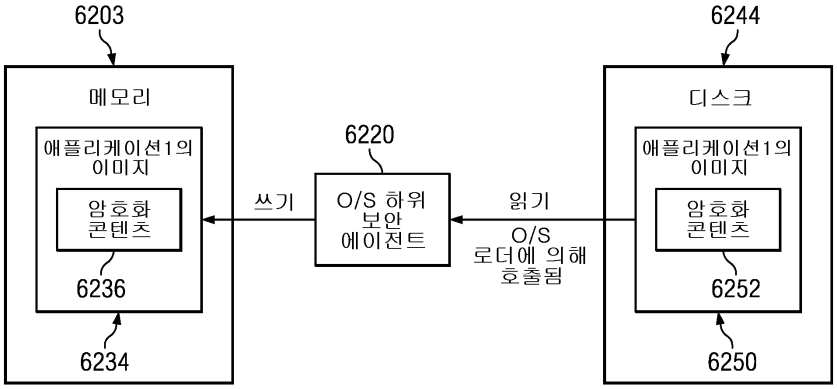
도면60


6026
애플리케이션
드라이버
6028
6001
6019
6021
6012
O/S
O/S 내 보안 에이전트
보안 규칙들
6008
6020
O/S 하위 보안 에이전트
보안 규칙들
메모리 맵
6003
6010
메모리
페이지 테이블 디렉토리
허가 사항들(R/W/E)
6044
스토리지
6046
6032
쓰기
A
B
C
D
E
스왑 파일들
스왑된 콘텐츠
6048
6030
할당
쓰기
실행
6034
A
읽기
애플리케이션 이미지
6050
6036
드라이버 코드 섹션
쓰기
E
6038
(미할당)
쓰기
B
6040
비신뢰 드라이버
C
D
6042
비존재 콘텐츠
쓰기
6002
쓰기
프로세서

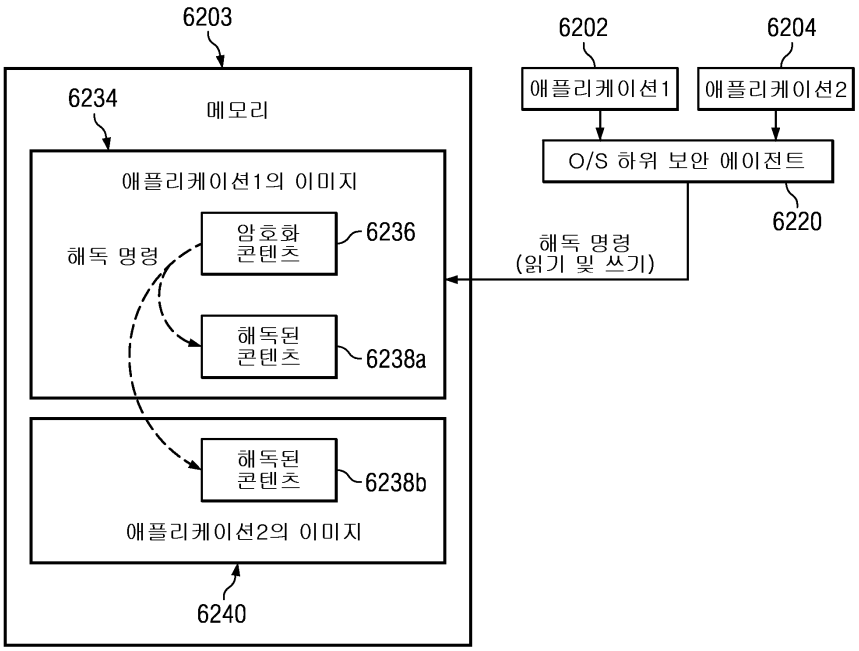
도면61


6104
악성
소프트웨어
서버
(코드)
6102
애플리케이션
6120
O/S 하위 보안 에이전트
쓰기(코드)
6138
6140
코드 섹션
(코드)
코드 섹션
(코드)
6103
애플리케이션
드라이버
6130
메모리
6132

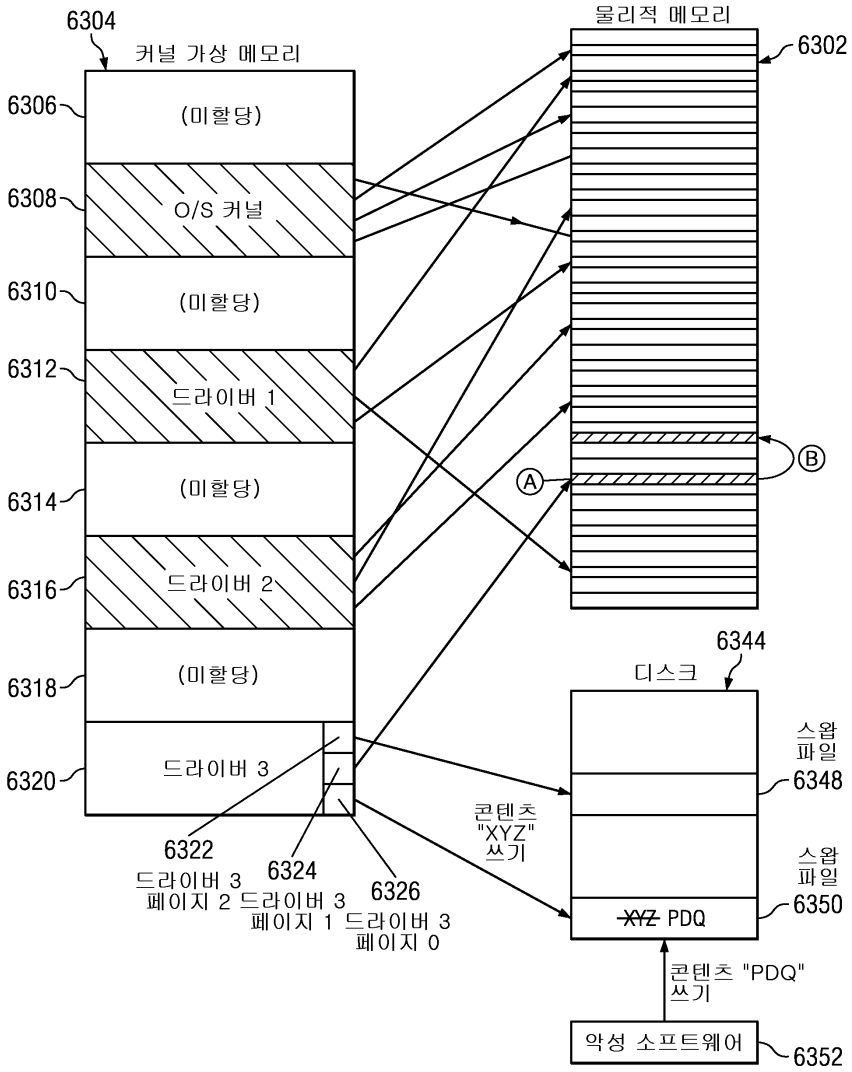
도면62a


6203
메모리
애플리케이션1의
이미지
암호화
콘텐츠
6236
6234
6220
쓰기
O/S 하위
보안
에이전트
읽기
O/S
로더에 의해
호출됨
6244
디스크
애플리케이션1의
이미지
암호화
콘텐츠
6252
6250

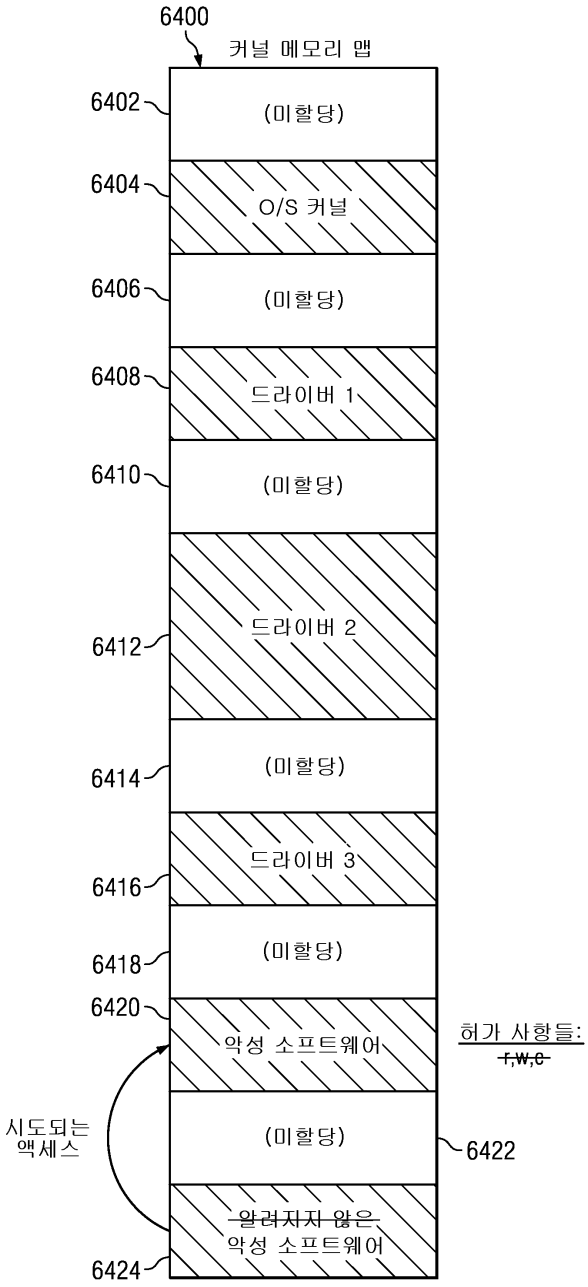
도면62b


6203
6234
메모리
애플리케이션1의 이미지
암호화 콘텐츠
6236
해독 명령
해독된 콘텐츠
6238a
해독된 콘텐츠
6238b
애플리케이션2의 이미지
6240
6202
애플리케이션1
6204
애플리케이션2
O/S 하위 보안 에이전트
6220
해독 명령
(읽기 및 쓰기)

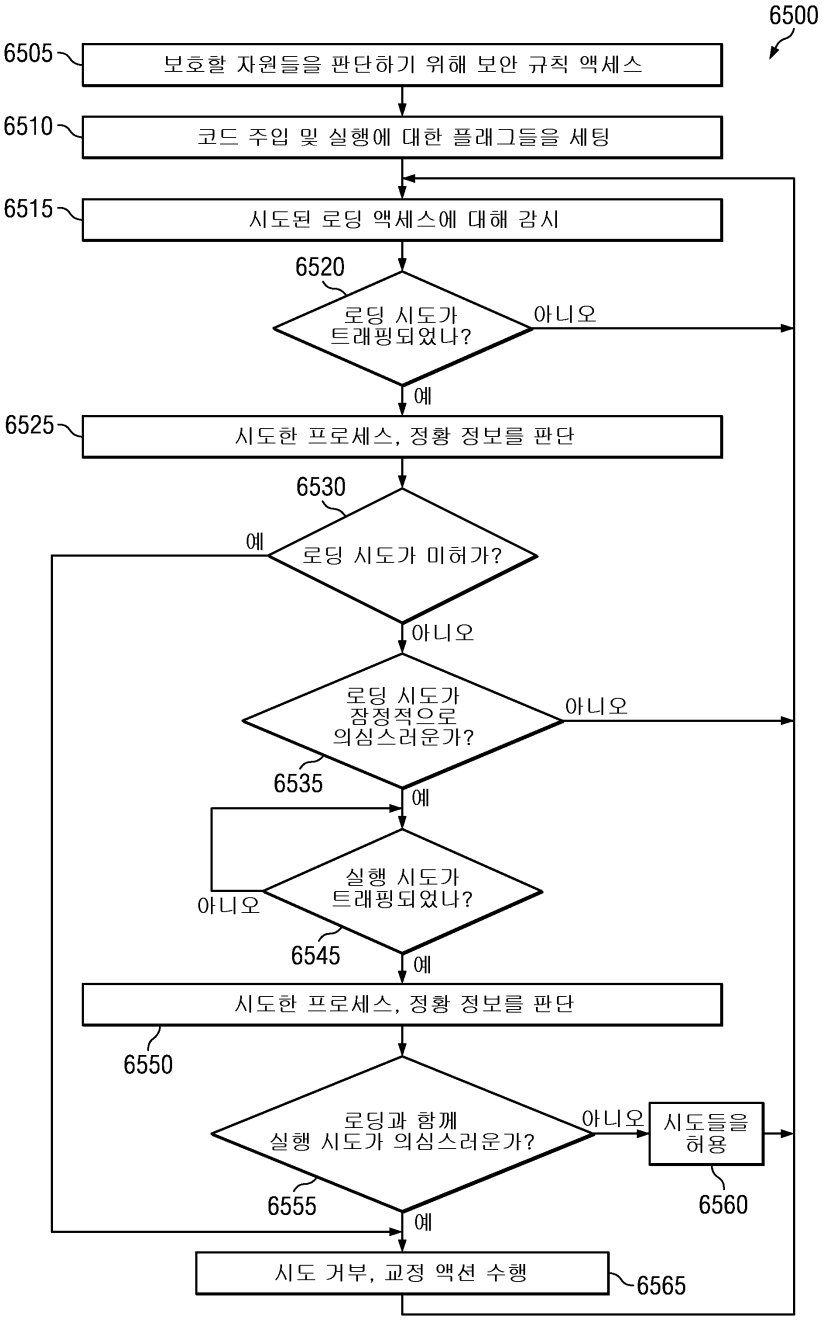
도면63


6304
커널 가상 메모리
6306
(미할당)
6308
O/S 커널
6310
(미할당)
6312
드라이버 1
6314
(미할당)
6316
드라이버 2
6318
(미할당)
6320
드라이버 3
6322
드라이버 3
페이지 2
6324
드라이버 3
페이지 1
6326
드라이버 3
페이지 0
물리적 메모리
6302
Ⓐ
Ⓑ
6344
디스크
스왑
파일
6348
스왑
파일
6350
~~XYZ~~ PDQ
콘텐츠
"XYZ"
쓰기
콘텐츠 "PDQ"
쓰기
악성 소프트웨어
6352

도면64


6400
커널 메모리 맵
6402
(미할당)
6404
O/S 커널
6406
(미할당)
6408
드라이버 1
6410
(미할당)
6412
드라이버 2
6414
(미할당)
6416
드라이버 3
6418
(미할당)
6420
악성 소프트웨어
허가 사항들:
~~r,w,e~~
시도되는
액세스
(미할당)
6422
~~알려지지 않은~~
악성 소프트웨어
6424

도면65


6500
6505
보호할 자원들을 판단하기 위해 보안 규칙 액세스
6510
코드 주입 및 실행에 대한 플래그들을 세팅
6515
시도된 로딩 액세스에 대해 감시
6520
로딩 시도가 트래핑되었나?
아니오
예
6525
시도한 프로세스, 정황 정보를 판단
6530
로딩 시도가 미허가?
예
아니오
로딩 시도가 잠정적으로 의심스러운가?
아니오
6535
예
실행 시도가 트래핑되었나?
아니오
6545
예
시도한 프로세스, 정황 정보를 판단
6550
로딩과 함께 실행 시도가 의심스러운가?
아니오
시도들을 허용
6555
6560
예
시도 거부, 교정 액션 수행
6565