(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2011-0040767
(43) 공개일자 2011년04월20일

(51) Int. Cl.
G06F 21/22 (2006.01) *G06F 9/30* (2006.01)
(21) 출원번호 10-2010-7028644
(22) 출원일자(국제출원일자) 2009년06월24일
심사청구일자 없음
(85) 번역문제출일자 2010년12월20일
(86) 국제출원번호 PCT/US2009/048461
(87) 국제공개번호 WO 2009/158405
국제공개일자 2009년12월30일
(30) 우선권주장
12/163,164 2008년06월27일 미국(US)

(71) 출원인
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
콜스, 닐, 로렌스
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 국제 특허 내
쉘, 스캇 랜들
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 국제 특허 내
(뒷면에 계속)
(74) 대리인
양영준, 백만기

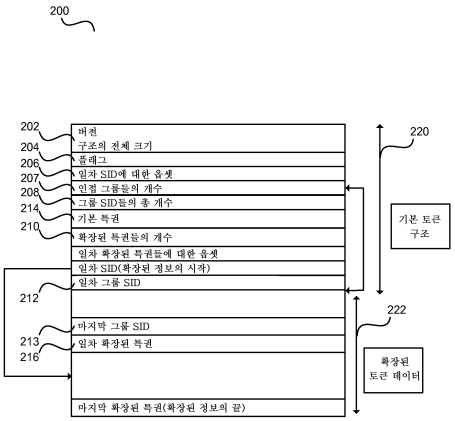
전체 청구항 수 : 총 20 항

(54) 컴퓨팅 프로세스의 최소 특권의 액세스 허가

(57) 요 약

실시예들은 애플리케이션에 의해 어떤 자원이 액세스될 수 있는지 그리고 애플리케이션이 호출할 수 있는 API가 무엇인지를 제어하기 위해 기존의 오퍼레이팅 시스템의 상단에서 구동하도록 구성될 수 있는 보안 기반구조를 제공한다. 보안 판단은 현재 스레드의 아이덴티티 및 현재 스레드의 콜 체인 콘텍스트를 둘다 고려함으로써 수행되어 디폴트로 특권을 최소화할 수 있도록 한다. 현재 스레드 콘텍스트가 캡쳐되고 이것의 카피가 생성되어 비동기적으로 보안 체크를 수행하는데 사용된다. 시스템 내 모든 스레드는 연관된 아이덴티티를 가지고 있다. 특정 자원에의 액세스를 획득하기 위하여, 각각의 호출자 및 스레드가 그 자원에 액세스하도록 하기 위해 현재 스레드 상의 모든 호출자들이 분석된다. 각각의 호출자 및 스레드가 그 자원에 액세스할 때만 호출자에게 그 자원에의 액세스가 주어진다.

대 표 도 - 도2



(72) 발명자

산다디, 우펜더 레디

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 국제 특허내

발스 안젤로 레나토

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 국제 특허내

리온스, 매튜, 지.

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 국제 특허내

조던, 크리스토퍼 로스

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 국제 특허내

로저스, 앤드류

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 국제 특허내

고팔란, 야두

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 국제 특허내

시에, 보-밍

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 국제 특허내

특허청구의 범위

청구항 1

실행가능한 명령어를 포함하는 컴퓨터 판독가능 매체로서,

상기 실행가능한 명령어는, 실행될 때,

연관된 주체의 고유 식별자를 규정하며 연관된 주체와 연관된 계정을 명시하는 보안 식별자(100)를 제공하는 단계;

보안 식별자(100)에 배정된 기본 특권(214) 및 확장된 특권(216)을 결정하기 위해 계정 데이터베이스 내 엔트리에 고유 식별자를 맵핑하는 단계;

연관된 주체의 이동(migration)에 따른 연관된 주체들의 아이덴티티들(210, 212)을 누적하는 단계;

보안 식별자(100)에 배정된 기본 특권(214) 및 확장된 특권(216)의 결정에 따라서 아이덴티티들(210, 212) 세트 및 아이덴티티들 세트에 배정된 특권들의 집합을 규정하는 보안 토큰(200)을 생성하는 단계; 및

연관된 주체의 모든 누적된 아이덴티티들(210, 212) 및 특권들(214, 216)의 집합을 교차시킴으로써 연관된 주체의 액세스 특권을 결정하는 단계를 수행함으로써,

연관된 주체에 의한 자원에의 액세스를 제어하기 위한 보안 기반구조를 제공하는

컴퓨터 판독가능 매체.

청구항 2

제1항에 있어서, 연관된 주체는 프로세스 또는 프로세스의 스레드(thread)를 포함하는 컴퓨터 판독가능 매체.

청구항 3

제1항에 있어서, 연관된 주체의 액세스 특권들(214, 216)을 결정하는 단계는,

현재 연관된 주체의 콜 체인 콘텍스트를 캡쳐하는 단계, 및

현재 연관된 주체를 분석하여 콜 체인에서 누적된 각각의 아이덴티티(210, 212) 및 각각의 연관된 주체가 요청된 자원에 액세스하는 것을 검증한 다음에 자원에의 액세스를 허가함으로써 디폴트로 특권을 최소화할 수 있도록 하는 단계를 더 포함하는 컴퓨터 판독가능 매체.

청구항 4

제1항에 있어서, 보안 토큰(200)과 보안 디스크립터(300) 사이의 관계는 연관된 주체에 의한 요청된 자원에의 액세스 권리를 제어하는 것을 결정하는 컴퓨터 판독가능 매체.

청구항 5

제1항에 있어서, 액세스 특권들(214, 216)을 결정하는 단계는,

보안 토큰(200) 내 각 보안 식별자(100)를 이용하여 보안 토큰이 보안 디스크립터(300)에 의해 명시된 자원에의 액세스를 요청했는지를 결정하며,

콘택트 체인 내 모든 호출자가 요청된 자원에의 액세스를 허가하는 특권을 가질 때만 요청된 자원에의 액세스가 허가되는 컴퓨터 판독가능 매체.

청구항 6

제1항에 있어서,

자원에의 액세스 권리를 규정하는 보안 디스크립터(300)를 구축하는 단계;

보안 토큰(200)에 따라 연관된 주체를 식별하는 단계;

보안 디스크립터(300)에서 규정된 아이덴티티(320, 330)가 보안 디스크립터 내 액세스 제어 엔트리(500)에 포함되어 있는지를 결정하는 단계;

액세스 제어 엔트리에 근거하여 아이덴티티에 이용가능한 액세스 권리를 결정하는 단계; 및

결정된 액세스 권리에 따라서 아이덴티티(320, 330)에게 액세스 권리를 허가하는 단계를 더 포함하는 컴퓨터 판독가능 매체.

청구항 7

제1항에 있어서,

보안 토큰 리스트(700)에서 연관된 주체에 관련된 모든 토큰들(200)의 리스트를 유지하는 단계를 더 포함하며,

현재 액티브 토큰(710)은 항상 보안 토큰 리스트의 맨 앞에 있으며, 현재 연관된 주체의 모든 액세스 및 특권 체크는 보안 토큰 리스트(700)의 맨 앞에 있는 현재 액티브 보안 토큰에 대하여 연관된 주체가 액세스를 요청한 자원을 비교함으로써 현재 액티브 보안 토큰(710) 및 연관된 주체의 현재 콘텍스트를 이용하여 처리되는 컴퓨터 판독가능 매체.

청구항 8

제1항에 있어서,

클라이언트 측(1020)에서, 메시지 큐 내 연관된 주체의 보안 콘텍스트를 기록하는 단계,

서버 측(1030)에서, 메시지 큐로부터 연관된 주체의 보안 콘텍스트를 비동기적으로 검색하는 단계, 및

위장될 연관된 주체의 검색된 보안 콘텍스트를 카피함으로써 연관된 주체를 위장하는 단계를 더 포함하는 컴퓨터 판독가능 매체.

청구항 9

제1항에 있어서,

연관된 주체의 현재 보안 토큰(710)과 연관된 모든 아이덴티티들(210, 212)이 요청 자원에 액세스할 때,

현재 연관된 주체의 콜스택(800) 내 모든 챔버들과 연관된 모든 아이덴티티들이 요청 자원에 액세스할 때, 및

연관된 주체의 저장된 콘텍스트와 연관된 모든 아이덴티티들이 요청된 자원에 액세스할 때,

요청된 자원에의 액세스를 허가하는 단계를 더 포함하는 컴퓨터 판독가능 매체.

청구항 10

제1항에 있어서, 오프라인 처리를 위해 보안 콘택트들을 오프라인 데이터베이스에 저장하는 단계를 더 포함하는 컴퓨터 판독가능 매체.

청구항 11

보안 기반구조로서,

연관된 주체의 고유 식별자를 규정하며 연관된 주체와 연관된 계정을 명시하는 보안 식별자(100);

보안 식별자(100)에 배정된 기본 특권(214) 및 확장된 특권(216)을 결정하는 것에 기초하여 아이덴티티들 세트 및 아이덴티티들 세트에 배정된 특권들의 집합을 규정하는 보안 토큰(200);

요청된 자원에 액세스한 계정들 및 프로세스에 관한 규칙을 규정하는 보안 디스크립터; 및

보안 식별자(100)의 액세스 권리를 식별하는 적어도 하나의 액세스 제어 엔트리(500)를 포함하는 액세스 제어 리스트(400)

를 포함하는 보안 기반구조.

청구항 12

제11항에 있어서, 보안 토큰(200)은 구조의 버전(202), 플래그(204), 옵셋(206), 인접 그룹들의 개수(207) 및 그룹 식별자들의 총 개수(208)를 식별하는 필드들을 포함하는 보안 기반구조.

청구항 13

제12항에 있어서, 보안 토큰(200)은 일차 소유자 보안 식별자(210), 그룹 보안 식별자(212), 기본 특권(214) 및 확장된 특권(216)을 더 포함하는 보안 기반구조.

청구항 14

제13항에 있어서, 일차 보안 식별자(210) 및 그룹 보안 식별자(212)는 보안 토큰(200)의 연관된 주체들의 아이덴티티들을 규정하는 보안 기반구조.

청구항 15

제13항에 있어서, 기본 특권(214)은 보안 토큰(200)에서 명시된 아이덴티티들에 유효한 특권 세트를 포함하는 보안 기반구조.

청구항 16

제13항에 있어서, 확장된 특권(216)은 보안 토큰(200) 내 보안 식별자들(100)에 대해 규정된 관례적 특권을 포함하는 보안 기반구조.

청구항 17

연관된 주체의 최소 특권의 액세스를 허가하는 방법으로서,

연관된 주체의 고유 식별자를 규정하고 연관된 주체와 연관된 계정을 명시하는 보안 식별자(100)를 제공하는 단계;

보안 식별자(100)에 배정된 기본 특권(214) 및 확장된 특권(216)을 결정하기 위해 계정 데이터베이스 내 엔트리에 고유 식별자를 맵핑하는 단계;

연관된 주체의 이동(migration)에 따른 연관된 주체들의 아이덴티티들(210, 212)을 누적하는 단계;

보안 식별자(100)에 배정된 기본 특권(214) 및 확장된 특권(216)의 결정에 근거하여 아이덴티티들(210, 212) 세트 및 아이덴티티들 세트에 배정된 특권들의 집합을 규정하는 보안 토큰(200)을 생성하는 단계; 및

연관된 주체의 누적된 식별자들(210, 212) 모두와 특권들(214, 216)의 집합을 교차시킴으로써 연관된 주체의 액세스 특권을 결정하는 단계

를 포함하는 방법.

청구항 18

제17항에 있어서, 연관된 주체의 액세스 특권들(214, 216)을 결정하는 단계는,

현재 연관된 주체의 콜 체인 콘텍스트를 캡처하는 단계, 및

현재 연관된 주체를 분석하여 콜 체인에서 누적된 각각의 아이덴티티(210, 212) 및 각각의 연관된 주체가 요청된 자원에 액세스한 것을 검증한 다음에 자원에의 액세스를 허가함으로써 디폴트로 특권을 최소화할 수 있게 하는 단계를 더 포함하는 방법.

청구항 19

제1항에 있어서,

클라이언트 측(1020)에서, 메시지 큐 내 연관된 주체의 보안 콘텍스트를 기록하는 단계,

서버 측(1030)에서, 메시지 큐로부터 연관된 주체의 보안 콘텍스트를 비동기적으로 검색하는 단계,

위장될 연관된 주체의 검색된 보안 콘텍스트를 카피함으로써 연관된 주체를 위장하는 단계,

카피된 보안 콘텍스트(750)를 분석하여 요청 자원에의 액세스 권리를 규정하는 보안 디스크립터(300)를 구축하

는 단계,

보안 토큰(200)에 따라 연관된 주체를 식별하는 단계,

보안 디스크립터(300)에서 규정된 아이덴티티(320,330)가 보안 디스크립터 내 액세스 제어 엔트리(500)에 포함되어 있는지를 결정하는 단계, 및

액세스 제어 엔트리(500)에 근거하여 아이덴티티(320,330)에 이용가능한 액세스 권리를 결정하는 단계를 더 포함하는 방법.

청구항 20

제1항에 있어서, 연관된 주체의 현재 보안 토큰(710)과 연관된 모든 아이덴티티들(210, 212)이 요청 자원에 액세스할 때,

현재 연관된 주체의 콜스택(800) 내 모든 챔버들과 연관된 모든 아이덴티티들이 요청 자원에 액세스할 때, 및

연관된 주체의 저장된 콘텍스트와 연관된 모든 아이덴티티들이 요청된 자원에 액세스할 때,

요청된 자원에의 액세스를 허가하는 단계를 더 포함하는 방법.

명 세 서

배 경 기 술

[0001] 장치에는 자원, 애플리케이션 등에의 액세스를 제어하는 보안 기반구조(security infrastructure)가 설정될 수 있다. 현재 모바일 장치에서, 코드 아이덴티티(code identity)를 기반으로 하여 개개의 애플리케이션에는 신뢰가 부여된다. 보안 기반구조는 장치에서 어떤 애플리케이션이 구동할 수 있는지, 어떤 애플리케이션이 통제(lock out)될 수 있는지, 어떤 애플리케이션이 어떤 콘텍스트에서 구동될 수 있는지, 그리고 그러한 애플리케이션이 액세스할 수 있는 자원이 무엇인지를 결정한다. 현재 신뢰도는 서명없는 모듈에 대해 "신뢰(trusted)" "보통(normal)", 및 "무신뢰(untrusted)"를 포함한다. 애플리케이션 또는 모듈은 신뢰할 수 있는 인증서 또는 신뢰할 수 있는 인증 스토어(certification store)에 연쇄(chain)한 인증서로 서명될 수 있다. 이 경우, 그 모듈은 신뢰할 수 있다고 간주되며 그와 같이 그 모듈 내의 어떤 코드라도 시스템에서 특권있는 API 및 자원들 모두에 액세스할 수 있다. 신뢰의 판단은 호출자의 애플리케이션 신뢰도를 바탕으로 한다.

[0002] 그러나, 전술한 현재의 보안 모델에는 여러 문제가 있다. 스레드(thread)가 시스템 내 다수의 보호 서버 라이브러리(protected server library: PSL)를 통해 이동(migrate)할 때, 콜 체인(call chain)에서 인접한 호출 프로세스가 시스템 자원에 액세스하는 것을 허용하는 것은 가능하지만, 전체 콜 체인 콘텍스트를 시험하는 것은 자원이 액세스가능하지 않아야 한다고 드러내는 것일 수 있다. 그러한 보안은 만일 보안 판단이 항상 전적으로 인접한 호출자의 콘텍스트를 근거로 한다면 강화될 수 없다. 또한, 현재의 보안 모델은 상이한 아이덴티티를 위장하는(impersonating), 즉, 낮은 특권의 애플리케이션이 시스템 서비스를 호출하는 것을 제공하지 않으며, 이 경우 시스템 서비스는 호출자의 콘텍스트 또는 그 자신의 콘텍스트를 근거로 한 요청을 처리하도록 요청받는다. 또한, 비동기 액세스 요청의 경우, 현재의 보안 모델은 호출자 콘텍스트가 완전히 이용가능하지 않은 경우에 이차 스레드에서 보안 체크를 제공하지 않는다.

[0003] 이러한 고려사항 및 다른 고려사항에 대해 본 발명이 만들어졌다.

발명의 내용

과제의 해결 수단

[0004] 본 요약은 간략한 형태로 아래의 상세한 설명에서 더 상세히 기술되는 개념을 발췌하여 소개하고자 한다. 본 요약이 청구된 주제의 중요한 특징 또는 필수적인 특징과 같은 것이라는 것은 아니고, 청구된 주제의 범위를 결정하는데 도움을 주고자 하는 것도 아니다.

[0005] 실시예들은 디폴트로 특권을 최소로 할 수 있도록 하기 위해 현재 스레드의 아이덴티티 및 현재 스레드의 콜 체인을 고려함으로써 보안을 판단하도록 제공된다. 현재 스레드 콘텍스트가 캡쳐되고 이것의 카피가 생성되어 비동기적으로 보안 체크를 수행하는데 사용된다. 시스템 내 모든 스레드는 연관된 아이덴티티를 갖는다. 초기에 이러한 아이덴티티는 패런트 프로세스로부터 유도된다. 그러나, 스레드 수명이 경과함에 따라, 이 아이덴티티

는 그 스레드의 모든 위장에 따라 변경할 수 있는 것이 가능하다.

[0006] 이러한 특징과 다른 특징 및 장점은 다음의 상세한 설명을 읽어보고 이와 관련된 도면을 검토함으로써 자명해질 것이다. 전술한 개괄적인 설명과 다음의 상세한 설명은 모두 예시적인 것일 뿐이며 청구한 바와 같은 본 발명을 제한하는 것이 아님은 물론이다.

도면의 간단한 설명

[0007] 도 1은 본 발명의 실시예에 따른 보안 식별자(SID)(100)를 예시한다.

도 2는 본 발명의 실시예에 따른 보안 토큰 구조(200)의 레이아웃을 예시한다.

도 3은 본 발명의 실시예에 따른 보안 디스크립터 레이아웃(300)의 간략한 도면을 예시한다.

도 4는 본 발명의 실시예에 따른 액세스 제어 리스트(ACL)(400)를 예시한다.

도 5는 본 발명의 실시예에 따른 액세스 제어 엔트리(ACE)(500)의 구조를 예시한다.

도 6은 본 발명의 실시예에 따른 보안 디스크립터(SD)(600)의 구조적 레이아웃을 예시한다.

도 7은 본 발명의 실시예에 따른 스레드(700)의 토큰 리스트 배열을 예시한다.

도 8은 본 발명의 실시예에 따른 PSL 콜(800) 내 스레드의 콜스택 리스트를 예시한다.

도 9는 본 발명의 실시예에 따른 위장 리스트와 스레드(900)의 콜스택 리스트 사이의 링크를 예시하는 블록도이다.

도 10은 본 발명의 실시예에 따른 메시지 큐 시스템(1000)을 예시한다.

도 11은 본 발명의 실시예들이 구현될 수 있는 컴퓨팅 환경을 예시한다.

발명을 실시하기 위한 구체적인 내용

[0008] 실시예들은 애플리케이션에 의해 어떤 자원이 액세스될 수 있는지 그리고 애플리케이션이 호출할 수 있는 API가 무엇인지를 제어하기 위해 기존의 오퍼레이팅 시스템의 상단에서 구동하도록 구성될 수 있는 보안 기반구조를 제공한다. 보안 판단은 현재 스레드의 아이덴티티 및 현재 스레드의 콜 체인 콘텍스트를 둘다 고려함으로써 수행되어 디폴트로 특권을 최소화할 수 있다. 현재 스레드 콘텍스트가 캡쳐되고 이것의 카피가 생성되어 비동기적으로 보안 체크를 수행하는데 사용된다. 시스템 내 모든 스레드는 연관된 아이덴티티를 가지고 있다. 초기에 이러한 아이덴티티는 패런트 프로세스로부터 유도되지만, 스레드 수명이 경과함에 따라 스레드가 행하는 모든 위장에 따라 이 아이덴티티가 변경할 수 있는 것이 가능하다.

[0009] 실시예들은 콜 또는 애플리케이션이 무엇을 요청하고 있는지를 분석할 뿐만아니라 그 콜이 어디에서부터 발생하는지를 분석한다. 실시예들은 또한 조정가능한(open-ended) 시스템용 보안 기반구조를 제공하며 그러므로 최종 사용자 또는 최종 운영자는 자신들의 제어하에 있지 않을 수 있는 상이한 애플리케이션을 설치할 수 있다. 따라서, 시스템 내 모든 스레드들은 디폴트로 최소의 특권을 가지고 구동되어 스레드가 액세스 권한을 갖지 않은 자원으로부터 그 스레드가 잘못 액세스되는 것으로 인한 보안 문제가 발생하는 것을 방지할 수 있다. 오히려, 특정 자원을 액세스하기 위하여, 각각의 호출자 및 스레드가 그 자원을 액세스할 수 있도록 하기 위해 현재 스레드 상의 모든 호출자들이 분석된다. 각각의 호출자 및 스레드가 그 자원에 액세스할 때만 호출자에게 그 자원에의 액세스 권한이 주어진다.

[0010] 본 발명의 실시예들은 애플리케이션에 의해 어떤 자원이 액세스될 수 있는지 그리고 애플리케이션이 호출할 수 있는 API가 무엇인지를 제어하기 위한 보안 기반구조를 제공하는 데이터 구조를 이용하여 구현될 수 있다. 이 데이터 구조는 다음과 같은 구성요소, 즉, 보안 식별자(Security Identifier: SID), 보안 토큰(Security Token), 보안 디스크립터(Security Descriptor: SD), 액세스 제어 리스트(Access Control List: ACL), 및 액세스 제어 엔트리(Access Control Entry: ACE)를 포함한다.

[0011] 도 1은 본 발명의 실시예에 따른 보안 식별자(Security Identifier: SID)(100)를 예시한다. SID(100)는 시스템 내 고유 ID를 규정하고 어떤 계정에서 특정 스레드가 구동하고 있는지를 명시하는 가변 길이 데이터 구조(110)이다. SID(100)는 또한 사용자 계정과 같은 것으로 간주될 수 있다. 예를 들어, 누군가 컴퓨터에 로그온할 때, 적절한 로그온 패스워드를 이용하여 여러 사용자 계정들이 액세스가능해질 수 있다. 사용자 계정은

컴퓨터에 로그 온하는 사람을 식별한다. SID(100)는 단지 스레드 또는 프로세스를 식별하는 계정일 뿐이다. SID(100)는 특정 디바이스 내에서 오로지 유일무이하다. 그러나, 당업자들이라면 본 발명의 실시예들의 설명을 분석한 후 전역적인 보안 식별자가 사용될 수 있음을 인식할 것이다.

[0012] 데스크탑에서, SID(100)는 개개 사용자 계정, 소정 시스템 계정, 및 소정 사용자 그룹에 배정될 수 있다. 예를 들어, WINDOWS® CE 오퍼레이팅 시스템에서 계정은 데스크탑 WINDOWS® 오퍼레이팅 시스템과 관련하여 사용된 정의와 반드시 동일하지는 않음을 주목하자. 그럼에도 불구하고, 당업자들이라면 무엇이 계정을 구성하는지를 이해할 것이다. 본 명세서에서, SID(100)는 계정 데이터베이스 내 엔트리에 맵핑하는 전체 오퍼레이팅 시스템, 예컨대, WINDOWS® CE 오퍼레이팅 시스템 전체에서 고유 식별자인 것으로 가정한다. 계정 데이터베이스 내 엔트리는 어떤 기본 특권 및 확장된 특권이 특정 계정(다른 말로 SID)에 배정되는지를 명시한다.

[0013] 도 2는 본 발명의 실시예에 따른 보안 토큰 구조(200)의 레이아웃을 예시한다. 보안 토큰(200)은 한 세트의 아이덴티티와 이들 아이덴티티들에 배정된 특권들의 집합을 규정하는데 사용된다. 전형적으로, 보안 토큰(200)은 프로세스, 스레드, 동기화와 같은 런타임 객체들과 연관되며 메시지 큐 내 개개의 메시지들과도 연관된다. 보안 토큰(200)은 이 구조의 버전(202), 플래그(204), 옵셋(206), 인접 그룹들의 개수(207), 및 그룹 ID들의 총 개수(208)를 식별하는 필드를 포함한다. 보안 토큰(200)의 구조는 또한 일차 소유자 SID(210), 그룹 SID(들)(212), 기본 특권(214) 및 확장된 특권(216)을 저장할 수 있다.

[0014] 일차 SID(210) 및 그룹 SID(들)(212)는 이 보안 토큰(200)을 갖는 객체와 연관된 아이덴티티를 규정한다. 기본 특권(214) 및 확장된 특권(216)은 이 보안 토큰(200)과 연관된 객체에게 어떤 특권이 부여되는지를 규정한다. 기본 특권(214)은 보안 토큰(200)에서 명시된 아이덴티티들에게 유효한 한 세트의 특권들이다. 유사하게, 확장된 특권(216)은 보안 토큰(200) 내 SID들의 리스트에 대해 규정된 관례적인 특권들이다.

[0015] 기본적인 보안 토큰 구조(200)는 보안 토큰(200)의 확장된(선택적) 데이터, 이를 테면, 이 보안 토큰(200)과 연관된 일차 SID(210) 및 그룹 SID(212), 및 이 보안 토큰(200)과 연관된 확장된 특권(216)에 대한 옵셋 포인터(206)를 갖는다. 보안 토큰(200) 내 개개의 SID(210, 212)는 각기 AccessCheck API 콜에서 사용되어 소정 보안 토큰(200)이 SD가 명시된 객체에의 액세스를 희망하는지를 판단한다. 또한, 보안 토큰(200) 내 기본 특권(214) 및 확장된 특권(216)은 PrivilegeCheck API 콜에서 사용되어 소정 보안 토큰(200)이 특권을 요청했는지를 판단한다. API들 및 보안 토큰(200)의 용도는 아래의 본 명세서에서 기술된 토큰 API들의 설명을 참조하여 더 상세히 기술될 것이다.

[0016] 계정의 스트링 표현(string representation)이 생성되며 이 스트링 표현은 보안 토큰 구조(200)에서 사용된 DWord에 맵핑된다. 그러면, 보안 토큰(200)은 시스템 내 판독 스레드와 연관된 객체가 된다. 그러므로, 스레드는 모두 특정 보안 토큰(200)에서부터 시작한다. 보안 토큰(200)은 이러한 특정 스레드가 특정 계정 ID를 갖는 특정 챔버에 속한다는 것과, 또한 스레드가 소정 그룹의 멤버임을 나타내는 아이덴티티들의 리스트를 갖는 기본 토큰 구조(220)를 포함한다. 각각의 보안 토큰(200)은 세트별로 표현된다. 따라서, 보안 토큰 구조는 구조 및 다수의 세트를 포함한다. 제1 세트는 소유자 세트이다. 이것은 시스템 내 각 챔버에 또는 시스템 내 각 계정에 고유 ID가 주어진다는 것을 의미한다. 그래서, 그룹 멤버쉽, 그룹 계정 ID들이 하나도 없거나 많을 수 있다. 이것들이 확장된 토큰 데이터(222)에서 주어진다.

[0017] 데스크탑에서도 보안 토큰 구조는 유사하다. 예를 들어, 데스크탑을 리셋하기 위해 규정된 특권 또는 드라이버를 설치하기 위해 규정된 특권이 있을 수 있다. 따라서, 만일 특정한 호출자가 특정한 권한 세트를 갖는다면, 그 호출자만이 그 API 콜을 할 수 있으며 그 호출자만이 디바이스를 리세트 또는 드라이버를 설치할 수 있다. 이러한 것들은 특정 특권들이다. 기본 특권은 단 하나의 DWord일 수 있으며, 그러므로 기본 특권들(214)과 연결되며, 확장된 토큰 데이터(222)의 일부인 확장된 특권들(216)은 몇 개라도 있을 수 있다.

[0018] 많은 특권들(214, 216)이 있을 수 있지만, 특권들(214, 216)은 자원들에 대한 액세스 체크를 행하는 방법을 정확하게 수행한다. 따라서, 만일 특정 API가 특별한 특권을 요구하거나 또는 특정 호출자가 특정 특권을 갖고자 요청한다면, 콘택트 체인에서 호출자는 모두 그 특정 특권을 가져야 한다. 모든 호출자들이 그 특권을 가졌을 때만 그 호출이 통과하도록 허용될 것이다. 모든 토큰은 적어도 기본 토큰 구조(220)와 같은 정도의 정보를 갖고 있어야 한다.

[0019] 확장된 토큰 구조(222)는 선택적인 설정(optional setting)이다. 그러므로, 예로서, 계정은 하나 이상의 그룹들의 한 멤버일 수 있다. 따라서, 확장된 토큰 구조는 가변가능한 크기를 갖는데, 즉, 그 크기는 고정되지 않지만, 기본 토큰 구조의 크기는 고정된다. 보안 토큰(200)은 모든 프로세스와 연관되며 그래서 프로세스 내 모

든 스레드는 시작 시점에서 보안 토큰(200)을 수신 또는 복사한다. 보안 토큰(200)이 상이한 서버들로 이동할 때, 보안 토큰(200)은 변할 수 있다. 이동한 보안 토큰(200)은 서버의 보안 토큰과 교차된다. 실제 데이터 구조는 변하지 않는다.

[0020] 도 3은 본 발명의 실시예에 따른 보안 디스크립터 레이아웃(300)의 간략한 도면을 예시한다. 보안 디스크립터(300)는 자원에 허가를 연관시키는데 사용된 데이터 구조이다. 특정 자원의 경우, 보안 디스크립터(300)는 그 자원에 액세스하는 것이 어떤 계정인지를 규정하며 또한 그 특정 객체와 관련하는 모든 규칙을 규정한다. 전형적으로, 객체의 보안 디스크립터(300)는 소유자 SID(320), 그룹 SID(330) 및 연관된 ACL(340)을 규정한다.

[0021] 이러한 것들 모두가 선택적 엔트리들이기 때문에, SD(300)는 그 구조의 끝부분에 소유지 SID(320), 그룹 SID(330), 및 ACL(340)이 리스트된 가변가능한 크기의 구조로서 규정된다. 버전 필드(310) 및 플래그 필드(312)는 SD 구조(300)에서 제공되어 어떤 값들이 SD(300)에 포함되는지를 명시한다. 필드(314)는 또한 SD 구조(300)의 전체 크기를 규정하기 위해 제공된다.

[0022] 보안 토큰(즉, 도 2의 (200))과 보안 디스크립터(300) 간의 관계는 액세스 권리의 제어를 결정한다. 스레드가 자원(예를 들어, 사진)을 액세스할 때, 파일, (예컨대, /windows/myphotos/)이 생성되며, 그래서 그 호출은 결국 서버에 들어가고 서버는 호출자가 이 자원에 액세스할 수 있는지를 결정한다. 서버는 그 호출로부터 자원에의 액세스 권리를 규정하는 보안 디스크립터(300)를 구축한다. 그러므로, 일단 서버가 이 자원을 누가 액세스할 수 있는지를 식별하는 보안 디스크립터(300)를 갖게 되면, 서버는 호출자의 보안 토큰(즉, 도 2의 (200))을 탐색하여 그 호출자의 아이덴티티를 결정한다. 그런 다음, 서버는 보안 디스크립터에서 규정된 아이덴티티가 ACE들 중 하나에 있는지를 결정하려 한다. 만일 보안 디스크립터에서 규정된 아이덴티티가 ACE들 중 하나에 있다면, 서버는 액세스 권리들이 그 아이덴티티의 것인지를 결정한다. 만일 그 아이덴티티의 액세스가 부정되면, 호출이 거부된다. 만일 그 아이덴티티의 액세스가 판독하는 것이라면, 그 호출자에게는 판독 허가만이 주어진다.

[0023] 따라서, 호출과 그 호출을 수신하는 특정 자원 간의 교차는 반드시 보안 토큰을 이용하여 식별되어야 하며 그 아이덴티티가 보안 디스크립터(300)에서 갖고 있는 허가가 무엇인지 결정되어야 한다. 오퍼레이팅 시스템 내에서, 다수의 계정 ID들이 제공될 수 있다. 어떤 프로세스 내에서 구동을 시작하는 모든 스레드는 그 프로세스에 대한 계정 ID를 갖는 토큰을 승계할 것이다. 스레드가 서비스의 상이한 필드들을 호출할 때, 그 토큰은 갱신된다.

[0024] 도 4는 본 발명의 실시예에 따른 액세스 제어 리스트(ACL)(400)를 예시한다. ACL(400)은 ACE 헤더들(420, 422) 및 ACE와 연관된 SID들(430, 432)을 포함하는 액세스 제어 엔트리들(Access Control Entries: ACE)의 집합이다. ACE들은 소정 SID에 대해 어떤 액세스 권리가 규정되어있는지를 식별한다. 도 4에서, ACL(400)은 개정 필드(402), 미사용 필드(404), 구조의 전체 크기를 규정하는 필드(406), ACL의 끝부분에서 ACE들의 개수를 규정하는 필드(408) 및 제2 미사용 필드(410)를 포함한다. ACE들의 개수는 사전에 규정되지 않기 때문에, ACL(400)은 가변가능한 길이의 구조이다.

[0025] 각각의 ACE는 계정 ID 및 그 계정 ID가 그 객체에 대해 갖고 있는 허가가 무엇인지를 명시한다. 예를 들어, 하나의 특정 ACE는 그 특정 객체에 대해서만 판독하는 액세스인 계정 ID를 획득할 수 있다. 또 다른 특정 ACE는 계정 ID 관리자 액세스 판독/기록일 수 있으며, 이는 상이한 모든 ACE들의 집합을 제공하며 이들 ACE들 각각은 특정 계정에 대해 어떤 액세스가 이 자원에 명시되었는지를 규정한다.

[0026] 데스크탑에서, ACE는 각 엔트리마다 상이한 액세스 권리를 제공한다. 상이한 액세스 권리는 허용(Allow), 거부(Deny), 및 감사(Audit)를 포함한다. WINDOWS® CE 오퍼레이팅 시스템에서, 권리는 소정 ACE를 대상으로 승인된다. ACL에서 액세스 마스크를 요청한 ACE가 없을 때 디폴트 리턴(default return)은 "거부"이다.

[0027] 도 5는 본 발명의 실시예에 따른 액세스 제어 엔트리(ACE)(500)의 구조를 예시한다. 각각의 ACE(500)는 이 ACE(510)에 의해 어떤 형태의 액세스가 (SID(520)에 의해 제공된) 어느 아이덴티티에 허용되는지를 규정한다. SID(520)는 가변가능한 길이의 데이터 아이템이기 때문에, ACE(500)도 역시 가변가능한 길이의 구조이다. ACE(500)와 연관된 SID(520) 데이터는 ACE 구조(500)의 끝부분에서부터 시작한다. ACE(500)는 또한 플래그 필드(530), 구조의 전체 크기를 식별하는 필드(532) 및 마스크 필드(534)를 포함한다.

[0028] 도 6은 본 발명의 실시예에 따른 보안 디스크립터(SD)(600)의 구조적 레이아웃을 예시한다. 보안 디스크립터(SD)(600)는 헤더(610)를 포함한다. SD 헤더(610)는 SD(600)의 버전, 플래그 및 크기를 식별한다. SD(600)는 그 다음으로 소유자 보안 식별자(SID)(620) 및 그룹 보안 식별자(SID)(630)를 포함한다. 나머지 데이터는 액세

스 제어 리스트(ACL)(640)를 형성한다. ACL(640)은 ACL 헤더(642) 및 하나 이상의 액세스 제어 엔트리들(ACEs)(650)을 포함한다. ACL 헤더(642)는 ACE(650)의 버전, 크기 및 개수를 식별한다. 각각의 ACE(650)는 ACE 헤더(652) 및 연관된 보안 식별자(SID)(654)를 포함한다.

[0029] 본 발명의 실시예에 따른 보안 기반구조는 프로세스 또는 스레드가 생성될 때 보안 디스크립터, 보안 식별자, 액세스 제어 리스트 및 액세스 제어 엔트리들에 의해 제공된다. 예를 들어, 프로세스 토큰은 불변이며 프로세스가 생성될 때 배정된다. 디폴트로, 어떠한 보안도 제공되지 않으면, 모든 프로세스들은 (토큰 특권들의 경우와) 동일하게 취급되며 프로세스는 사전에 규정된 시스템 토큰을 배정받는다. 이것은 보안이 가능하지 않은 시스템에서의 기본적인 동작 특성이다. 그러한 시스템에서, 신뢰의 경계(trust boundary)는 사용자 모드와 커널 모드 사이의 천이 시점이다.

[0030] 프로세스 토큰은 잠재적으로 아래와 같은 여러 데이터 포인트들이 결합된 것일 수 있다.

[0031] • 계정 데이터베이스 내 ID에 맵핑시킬 실행 증거(Exe evidence)(경로, 해시, 인증서)

[0032] • 계정 데이터베이스 내 소정 계정의 기본 특권들

[0033] • 계정 데이터베이스 내 소정 계정의 확장된 특권들

[0034] • 호출자 토큰에 기반한 그룹 ID들의 리스트

[0035] 주어진 실행가능(executable)의 증거가 되는 제 1편의 정보는 시큐어 로더 컴포넌트(secure loader component)에 의해 결정되며 이 정보는 본 명세서의 범주를 벗어난다. 이러한 특징의 목적상, 증거는 계정 데이터베이스 내 ID에 맵핑하는 것으로 가정한다. 이렇게 가정하면, OS는 계정 데이터베이스 내 계정 정보로부터 토큰을 생성할 것이다. 이 토큰은 프로세스가 생성될 때 프로세스 객체와 연관되며 프로세스의 수명 동안 내내 변하지 않는다.

[0036] 스레드 토큰은 스레드가 생성될 때 생성되고 스레드 객체와 연관된다. 디폴트로, 스레드 토큰은 스레드의 소유자 프로세스와 연관된 토큰과 동일하다. 프로세스 토큰과 스레드 토큰 간의 주요한 차이점은 스레드 토큰이 스레드의 수명 동안 변할 수 있는 반면, 프로세스 토큰은 그 프로세스의 수명 동안 변하지 않는다는 것이다. 예를 들어, 스레드 토큰은 아래에 따라 변할 수 있다.

[0037] • 소정 토큰을 위장하려는 콜: 이것은 호출하는 스레드의 액티브 토큰을 위장하려는 호출에서 통과된 보안 토큰으로 변경할 것이다.

[0038] • 이전의 위장을 되돌리려는 콜: 이것은 스레드 토큰을 위장 콜(impersonation call) 이전의 토큰으로 갱신할 것이다.

[0039] • 현재 프로세스를 위장하려는 호출: 이것은 현재 스레드 토큰을 현재 액티브 프로세스의 스레드 토큰이 되게 갱신할 것이다.

[0040] • API 콜로부터 스레드 리턴: API 콜을 리턴할 때, 커널은 스레드가 리턴하는 PSL 콘텍스트와 연관된 모든 스레드 토큰들을 자동으로 삭제할 것이다. 이 경우, 스레드와 연관된 현재 토큰 또한 API 콜 이전의 토큰으로 갱신된다.

[0041] 도 7은 본 발명의 실시예에 따른 스레드의 토큰 리스트 배열(700)을 예시한다. 소정 스레드의 다수의 토큰들을 관리하기 위해, 토큰 리스트(700)가 제공된다. 토큰 리스트(700)는 하나의 스레드와 연관된 모든 토큰들의 링크된 리스트이다. 스레드의 현재 액티브 토큰(710)은 항상 토큰 리스트(700)의 맨 앞에 있는 토큰 노드이다. 다시 말해서, 토큰 리스트(700)는 리스트에 마지막으로 추가된 토큰이 스레드의 현재 토큰이 되는 LIFO 큐 (다른 말로 스택)처럼 동작한다. 현재 스레드의 모든 액세스 및 특권은 그 특정한 토큰만을 그리고 스레드의 현재 콘텍스트를 이용하여 체크된다. 또한, 스레드가 API 콜로부터 리턴할 때, 커널은 콘텍스트에서 보호된 서버 라이브러리(protected server library: PSL)(712)로의 위장 호출들에 의해 이 리스트에 추가된 모든 토큰 노드들을 자동으로 삭제한다. 이러한 자동 되돌림은 API 콜 상에서의 임의의 특권의 누출을 방지한다.

[0042] 오퍼레이팅 시스템, 예를 들어, WINDOWS® CE에서, 스레드는 API 콜을 다루는 서버 프로세스로 천이함으로써 대응하는 API 콜을 발생할 수 있다. 각각의 스레드는 토큰에 대해 규정된 한 세트의 아이덴티티/특권들과 연관

된 그 토큰을 갖기 때문에, API 서버는 API 콜이 현재 스레드에서 허용되는지 여부를 판단하기 위해 스레드의 현재 특권을 고려할 필요가 있다. API 서버가 API 콜을 완료하기 위해 사용할 수 있는 두 가지 가능한 토큰은 아래와 같다.

[0043] • 호출자의 토큰: 이 경우, API 서버는 CeAccessCheck (GetCurrentToken(),...)를 호출한다. 이것은 만일 API 콜이 높은 특권의 챔버로부터 낮은 특권의 챔버로 발생하면 잠재적인 보안 위험에 처했을 것이다. 실제로 높거나 낮은 특권의 개념은 없다. 그러나, 이러한 논의의 목적상 높은 특권의 챔버가 낮은 특권의 챔버의 모든 특권들 및 어떤 부가적인 특권들을 갖는다고 가정한다.

[0044] • 현재 프로세스 토큰: 이 경우, API 서버는 CeAccessCheck (ImpersonateCurrentProcess(), ...)를 호출하며, 여기서 현재 스레드의 토큰은 현재 프로세스(API 서버의 토큰)로 갱신된다. 이것은 API 콜이 낮은 특권의 챔버에서높은 특권의 챔버로 발생하는 경우에도 역시 잠재적인 보안 위험에 처했을 것이다.

[0045] 모든 스레드는 처음에 소유자 프로세스 토큰(720)에서부터 시작한다. 그러므로, 소유자 프로세스 토큰(720)은 스레드(700)의 토큰 레이아웃을 도시하는 도 7의 하단에서 리스트된다. 스레드는 상이한 호출자로부터 또는 메시지 큐로부터 비동기적으로 토큰을 수신할 수 있다. 그러면, 스레드가 원할 때 그 토큰에 근거하여 자원에 액세스할 수 있다. 스레드는 프로세스 토큰을 사용하기를 원하지 않고, 그 대신 그것의 아이덴티티를 변경하기를 원한다. 그러므로, 스레드는 API들 중 하나를 호출하여 위장할 수 있으며, 이에 따라 토큰을 토큰 리스트의 상단으로 밀어올린다. 따라서, 스레드가 토큰을 호출하고 위장할 때마다, 그 토큰은 리스트의 상단에 놓이게 된다. 도 7에서, 위장된 토큰(730)은 소유자 프로세스 토큰(720) 위에 도시된다. 도 7에서, N개의 일련의 위장된 토큰들(730-750)이 도시되어 있다.

[0046] 스레드가 액세스한 어떤 자원이라도 상단의 토큰(750)에 대해, 즉, 현재 토큰(710) 내 상단 토큰에 대해 체크된다. 위장이 실시될 때마다, 위장 토큰(750)은 항상 소유자 프로세스 토큰(720)과 교차된다. 따라서, 스레드가 자원에 액세스할 때마다, 소유자 프로세스 토큰(720)은 그 자원에서 액세스 권리가 있음을 보여주어야 하며 방금 위장된 토큰(750)도 역시 그 자원에 액세스해야 한다. 이러한 방식으로, 스레드는 상이한 서버들로 천이하는 것처럼 상이한 서버에 API 콜을 발생할 수 있다. 또한, 만일 콘택트 체인 상에 어떤 서버들이라도 존재한다면, 모든 서버들은 그 자원에 액세스해야 한다.

[0047] 만일 세 번의 체크가 모두 검증되면, 스레드로의 액세스가 허용된다. 중요하게, 이 프로세스는 개개의 특권을 제공하는 것과 상이하다. 예를 들어, 만일 행위가 전적으로 현재 토큰(750)만을 근거로 하면 그리고 만일 현재 토큰(750)이 공교롭게도 더 높은 특권을 가지면, 특권 상승이 있을 것이다. 이와 반대로, 만일 행위가 전적으로 호출자의 토큰(720)만을 근거로 하면, 높은 특권의 서버에서 낮은 특권의 서버로 특권이 이동한 것이다. 따라서, 두 가지 경우에 있어서, 있을 수 있는 특권의 상승 때문에 개별 특권만이 사용될 때 보안 위험이 생긴다.

[0048] 일단 API 콜이 완료되면, 그리고 API 콜이 리턴되면, 토큰은 어떤 반복적인 특권의 누출을 방지하기 위해 서버에서 이루어진 어떤 위장된 채로 되돌아간다. 따라서, API 콜이 리턴될 때, 서버가 위장되었던 토큰을 되돌리는 것을 잊어버릴지라도, 서버는 자동적으로 되돌려진다. 이러한 방식으로, 클라이언트는 API 콜이 그 토큰을 리턴할 때 API 콜 이전에 행해졌던 것이 무언이던지 항상 다시 되돌려질 것이다.

[0049] 알 수 있는 바와 같이, 만일 호출자의 스레드 토큰만을 사용하거나 또는 현재 프로세스의 스레드 토큰만을 사용하는 경우에는 보안 문제가 있다. 이러한 문제를 해결하기 위해, 소정 스레드에 의한 자원에의 액세스를 체크할 때 다음과 같은 사항을 고려할 것을 제안한다.

[0050] • 스레드의 현재 토큰과 연관된 모든 아이덴티티들은 자원에 액세스해야 한다.

[0051] • (위장 경계까지의) 현재 스레드 콜스택 내 모든 챔버들과 연관된 모든 아이덴티티들은 자원에 액세스해야 한다.

[0052] • 그 스레드에 대해 저장된 콘텍스트와 연관된 모든 아이덴티티는 자원에 액세스해야 한다. 이러한 체크는 액세스 체크가 콜링 스레드와 다른 상이한 스레드에서 호출자 대신 액세스 체크가 수행될 때 주로 사용된다.

[0053] 이러한 변경은 소정 스레드 및 객체에 대해 액세스/특권이 체크되는 방법에 영향을 미칠 것이다. 어떤 객체에 액세스하려는 코드의 경우, 스레드의 현재 토큰(750) 및 API 서버의 토큰, 즉, 소유자 프로세스/스레드 토큰(720)은 둘다 그 객체에 액세스해야 한다. 어떤 특권을 획득하려는 코드의 경우, 스레드의 현재 토큰(750) 및

(이 실시예에서) API 서버의 토큰(720)은 둘다 그 특권 세트를 가져야 한다. 그 결과, 디폴트로, 스레드의 현재 토큰(750)이 항상 현재 스레드의 콜스택 체인(700)과 결합될 때 코드는 항상 최소한의 액세스/특권 세트를 가지고 구동될 것이다. 이로 인해 현재 스레드에 대해 액세스/특권 체크를 수행할 때 호출자의 토큰(720)만을 이용하여 또는 현재 프로세스 토큰(750)만을 이용함으로써 일어나는 보안 문제를 다루는 안전한 프로세스를 제공하게 된다. 본 명세서에서 토큰 API들 중 일부는 아래에서 표 1을 참조하여 설명하기로 한다.

표 1

[0054]

이름	개요	리턴
CeCreateToken(pToken, flags)	토큰 구조에 대한 포인터, 플래그들	토큰에 대한 핸들
CeImpersonateToken(hToken)	토큰의 핸들	위장이 성공적이면 TRUE; 그렇치않으면 FALSE
CeImpersonateCurrentProcess(void)	없음	스레드 토큰이 갱신되면 TRUE; 그렇치않으면 FALSE
CeRevertToSelf(void)	없음	없음
CeGetProcessAccount(hProcess)	프로세스의 핸들	소정 프로세스와 연관된 소유자 계정
CeGetThreadAccount(hThread)	스레드의 핸들	소정 스레드와 연관된 소유자 계정
CeGetOwnerAccount(hToken)	토큰의 핸들	소정 토큰과 연관된 소유자계정
CeGetGroupAccount(hToken, idx)	토큰의 핸들, 및 그룹 계정의 인덱스	소정 토큰과 연관된 그룹 계정
CeAccessCheck(pSD, hToken, AccessRequired)	SD에 대한 포인터, 토큰에 대한 핸들, 및 필요한 액세스 마스크	소정 토큰이 소정 SD에서 액세스를 원하면 TRUE; 그렇치 않으면 FALSE 이 API는 소정 토큰 내 각 계정 및 소정 SD에 리스트된 각 SD의 소정 액세스를 대상으로 체크한다.
CePrivilegeCheck(hToken, pPrivileges, cPrivileges)	토큰에 대한 핸들, 체크하려는 특권들의 어레이, 및 특권들의 개수	소정 토큰이 특권을 요청했으면 TRUE; 그렇치 않으면 FALSE 이 API는 소정 토큰 내 소정 특권 액세스를 대상으로 체크한다. API가 성공하려면, (pPriviledges 어레이에서 명시된) 모든 소정 특권들은 토큰의 기본특권 및 확장된 특권 리스트에 존재하여야 한다.
GetCurrentToken(void)	없음	스레드 현재 토큰에 대한 의사 핸들

[0055] 다음의 설명은 핵심적인 위장 및 액세스 체크 API들에 관한 것이다. 표 1에 리스트된 첫 번째 토큰은 토큰 구조에 대한 포인터, 및 플래그를 제공하는 CeCreateToken (pToken, flags) 필드이다. 토큰에 대한 핸들(handle to the token)이 리턴된다.

[0056] CeImpersonate Token은 소정 토큰을 위장하려는 모든 애플리케이션들에 의해 사용될 수 있다. 이것은 또한 호출자를 대신하여 비동기적 콜을 수행하는데에도 사용된다. 이 API를 구현하면 이 토큰을 현재 스레드의 토큰 리스트의 스택의 최상단으로 밀어올리게 되며 이 토큰을 스레드의 현재 토큰으로 만든다. 이러한 API 콜은 항상 현재 프로세스 토큰을 이 API 콜에 명시된 토큰과 병합할 것이며 병합된 토큰은 현재 스레드의 토큰으로서 취급된다.

[0057] 또한, (현재 프로세스 토큰에 의해 주어진) 현재 프로세스의 아이덴티티는 토큰 아이덴티티들 중에서 통과된 아이덴티티에 암시적으로 추가된다. 이렇게 하는 주된 이유는 호출자가 이 API 콜을 이용하여 특권을 얻지 못하게 하는 것이다. 다시 말해서, 이 API는 낮아진 특권으로 제한될 것이며 또는 기껏해야 현재 프로세스와 동일한 특권을 가지게 될 것이다. 이것은 디폴트로 가장 낮은 특권 레벨에서 코드를 실행할 수 있게 한다. 이것을 위한 용도는 다음과 같을 것이다. 즉, API 콜의 시점에서, 현재 스레드의 토큰은 은닉되며 나중에 API 서버에서 비동기 스레드에 의해 사용되어 콜의 시점에서 호출자의 콘텍스트를 이용하여 API 콜을 실행한다.

[0058] CeImpersonateCurrentProcess(void) 토큰은 현재 스레드의 토큰이 갱신되는 경우 참(TRUE)을 리턴하며; 그렇지 않으면 거짓(FALSE)을 리턴한다. CeRevertToSelf Token은 현재 스레드의 토큰 리스트의 토큰을 "꺼내기(pop)"하는데 사용된다. 전형적으로 이것은 API 서버들에 의해 사용되어 CeImpersonateToken or CeImpersonateCurrentProcess API 콜들을 매개로 하여 이루어진 어떤 토큰 위장을 원상태로 되돌리는데 사용된다. CeImpersonateCurrentProcess Token은 현재 스레드의 토큰 리스트를 현재 프로세스 토큰으로 절단(truncate)하는데 사용된다. 이것은 현재 프로세스 토큰을 현재 스레드의 토큰 리스트의 맨 앞으로 효과적으로 "밀어올린다(pushes)".

[0059] CeAccessCheck Token은 세 가지 변수(argument)가 있다. 첫 번째 변수는 액세스를 요청하는 객체에 대해 허가 세트를 나타내는 SD이다. 두 번째 변수는 객체에 액세스를 요청하는 아이덴티티인 토큰이다. 세 번째 변수는 객체에 액세스하기를 희망하는 것이다. CeAccessCheck Token은 소정 토큰 내 각 계정에 대한 소정 액세스, 그리고 소정 SD에 리스트된 각 ACE의 소정 액세스를 체크하는데 사용된다. 이 API는 소정 토큰 내 각 계정의 소정 액세스 그리고 소정 SD 내에 리스트된 각 ACE의 소정 액세스를 체크한다.

[0060] CePrivilegeCheck Token은 소정 토큰 내 소정 특권 액세스를 체크한다. 이 API는 소정 토큰 내 소정 특권 액세스를 체크한다. API가 성공하려면, (pPrivileges 어레이에 명시된) 소정 특권들 모두 토큰의 기본 또는 확장된 특권 리스트 내에 존재하여야 한다.

[0061] CeGetProcessAccount(hProcess) 토큰은 소정 프로세스와 연관된 소유자 계정을 리턴한다. CeGetThreadAccount(hThread) 토큰은 소정 스레드와 연관된 소유자 계정을 리턴한다. CeGetOwnerAccount(hToken) 토큰은 소정 토큰과 연관된 소유자 계정을 리턴한다. CeGetGroupAccount(hToken, idx) 토큰은 토큰의 핸들 및 그룹 계정의 인덱스를 사용하여 소정 토큰과 연관된 그룹 계정을 리턴한다. GetCurrentToken(void) 토큰은 스레드의 현재 토큰에 대한 의사 핸들(pseudo-handle)을 제공한다.

[0062] 도 8은 본 발명의 실시예에 따른 PSL 콜(800) 내 스레드의 콜스택 리스트를 예시한다. 동작 시, 커널은 모든 스레드마다 콜스택 구조의 리스트를 유지한다. 모든 PSL 콜 및 리턴에 따라, 스레드의 콜스택 리스트(800)가 갱신된다. 도 8에서, 네 개의 콜스택(810, 820, 830, 840)이 도시된다. 마지막 콜스택(840)은 최상단에 있으며 현재 콜스택이다. PSL 콜이 입력되면, 새로운 콜스택 구조가 추가된다. API 콜이 리턴하면, 최상단 콜스택 구조가 삭제된다. 콜스택 구조(800)는 스레드가 이동한 하나의 PSL 서버에 대응한다.

[0063] 도 9는 위장 리스트와 본 발명의 실시예에 따른 스레드(900)의 콜스택 리스트 간의 링크를 예시하는 블록도이다. 콜스택 구조와 유사하게, 스레드의 토큰들은 토큰 리스트의 상단에서 현재 위장의 상태대로 리스트에서 유지된다. 도 9에서, 네 개의 콜스택(910, 920, 930, 940)이 도시된다. 마지막 콜스택(940)이 상단에 있으며 현재 콜스택이다. 어떤 특권도 PSL 콜들 전체에서 누출되지 않도록 보장하기 위해, 커널은 현재 스레드의 위장 리스트와 현재 스레드의 콜스택 리스트 간의 링크를 유지한다. 각각의 위장 노드는 위장 콜이 발생한 콜스택 구조와 연관된다.

[0064] 따라서, 도 9에서, 애플리케이션에 속하는 스레드는 PSL 서버 S1(910)로 이동하였다. PSL 서버 S1은 위장 API를 호출하며 그 결과 위장 노드(912)가 위장 리스트에 추가되었으며 연관된 콜스택 구조 S1은 위장 노드 T1(912)에서 표시된다. 그 다음, PSL 서버 S1은 PSL 서버 S2(920)로 이동했다. S2는 어떤 위장 콜도 발생하지 않는다. 그 다음, 동일한 콜에서 스레드가 S3(930)로 이동할 때, S3 서버는 위장 토큰을 호출한다. 새로운 위장 노드(932)가 토큰 리스트에 추가되며 연관된 콜스택 구조 S3(930)가 T2(932)에 표시된다.

[0065] 자원 체크는 스레드 수명의 상이한 단계에서 수행된다. 예를 들어, S1(910)부터 액세스를 체크하는 것을 고려해 본다. 이 경우, 스레드가 S1 프로세스(910)로 이동했을 때 자원은 액세스를 대상으로 체크된다. 다음과 같은 체크가 수행된다.

[0066] • 토큰 T1 내 모든 아이덴티티들은 자원에 액세스해야 한다.

[0067] • 프로세스 S1과 연관된 토큰 내 모든 아이덴티티들은 자원에 액세스해야 한다.

[0068] 그 다음, S2(920)부터 액세스 체크하는 것을 고려해 본다. 이 경우, 스레드가 S1(910) 프로세스를 경유하여 S2(920) 프로세스로 이동했을 때 자원은 액세스를 대상으로 체크된다. 이 경우, 다음과 같은 체크가 수행된다.

[0069] • 토큰 T1 내 모든 아이덴티티들은 자원에 액세스해야 한다.

[0070] • 프로세스 S2와 연관된 토큰 내 모든 아이덴티티들은 자원에 액세스해야 한다.

[0071] • 프로세스 S1과 연관된 토큰 내 모든 아이덴티티들은 자원에 액세스해야 한다.

[0072] S3(930)부터 액세스 체크를 수행할 때, 스레드는 S1(910)을 경유하고 S2(920)를 경유하여 프로세스 S3(930)로 이동한다. 액세스 체크는 S3(930)부터 호출되었다. 다음과 같은 체크가 수행된다.

[0073] • 토큰 T2 내 모든 아이덴티티들은 자원에 액세스해야 한다.

[0074] • 프로세스 S3와 연관된 토큰 내 모든 아이덴티티들은 자원에 액세스해야 한다.

[0075] 이 경우 비록 스레드 풀 콘텍스트(thread full context)가 그의 콜 체인에서 프로세스들 S2(920) 및 S1(910)을 가지고 있을지라도, S3(930)이 토큰을 위장하고 있으며 그 위장이 콜스택 체인을 효과적으로 차단하기 때문에 이들 프로세스들에 대해 액세스 체크는 수행되지 않음을 주목하자.

[0076] (콜스택 상단(940)으로서 리스트된) 현재 프로세스부터 액세스 체크를 수행할 때, 스레드 콜스택 체인(900) 내 S3 프로세스(930)와 현재 프로세스(940) 사이에는 더 이상의 토큰 위장이 없다. 다음과 같은 체크가 수행된다.

[0077] • 토큰 T2 내 모든 아이덴티티들은 자원에 액세스해야 한다.

[0078] • 콜스택 체인 내 현재 프로세스에서 프로세스 S3까지 연관된 토큰 내 모든 아이덴티티들은 자원에 액세스해야 한다.

[0079] 그러므로 액세스 체크를 위한 기본적 규칙은 현재 스레드의 토큰 리스트의 맨 앞에 있는 토큰인 현재 토큰이 자원에 액세스해야 하는지 그리고 현재 프로세스부터 위장을 호출했던 마지막 프로세스(이것도 포함하여)까지의 모든 프로세스들이 자원에 액세스해야 하는지를 포함한다.

[0080] CeGetAccessMask API는 소정 SD의 토큰에 배정된 최대 액세스를 리턴하기 위해 제공될 수 있다. 그러한 API는 세 가지 변수를 이용하여 구현될 수 있다.

[0081] • 액세스를 체크를 원하는 아이덴티티들의 리스트를 갖는 토큰

[0082] • 소정 자원의 상이한 ID들과 연관된 ACE들을 리스트하는 보안 디스크립터

[0083] • 소정 SD 및 토큰 조합의 최대 액세스 세트를 리턴하는 [out] 파라미터

[0084] 이 API의 구현은 간단하다. 즉, 토큰 내 각 ID 마다 체크가 수행되어 SD 내에서 허용 ACE가 있는지를 결정한다. 만일 있다면, 그 ACE와 연관된 액세스 마스크는 리턴될 액세스 마스크 값에 추가된다. 이 단계는 (마지막 위장까지) 현재 스레드 콘텍스트 내 각 토큰마다 반복되며 모든 토큰들 내에 설정된 액세스 마스크만을 리턴한다. CeAccessCheck와 달리 이 API 콜은 토큰으로부터 매칭 ID를 갖는 모든 액세스 마스크들의 조합을 구하기 위해 SD 내 모든 ACE들을 스캔할 필요가 있음을 주목하자.

[0085] 또한, CeDuplicateToken API는 다음과 같은 변수들을 취함으로써 제공될 수 있다.

[0086] • 소스 토큰 호출자에 대한 핸들은 카피하려 시도한다.

[0087] • 새로 카피된 토큰 객체의 핸들에 대한 [out] 포인터는 콜링 프로세스 내 토큰 객체를 카피한다.

[0088] 도 10은 본 발명의 실시예에 따른 메시지 큐 시스템(1000)을 예시한다. 메시지 큐(1010)는 클라이언트(1020)와 본 발명의 실시예에 따른 서버(1030) 프로세스 사이에서 비동기적 메시징을 제공한다. 메시지 큐(1010)는 클라이언트(1020)로 하여금 메시지 큐(1010)의 라이트 엔드(write end)를 열고 서버(1030)로 하여금 메시지 큐(1010)의 리드 엔드(read end)를 열게 해준다. 클라이언트(1020) 및 서버(1030)는 메시지 큐(1010) 내에 메시지들 및 다른 데이터를 게재(post)할 수 있다. 다수의 메시지 큐(1010)가 제공될 수 있으며 그러한 메시지 큐(1010)는 비동기적으로 동작한다. 따라서, 클라이언트(1020)가 메시지를 작성한 다음 공개할 때, 클라이언트(1020)는 승인(acknowledgement) 또는 어떤 형태의 신호를 기다리지 않는다. 그 다음 서버(1030)는 메시지를 픽업하고 메시지 검색에 따라 어떤 행위를 수행한다. 서버(1030)가 메시지를 픽업할 때, 서버(1030)는 그 메시지의 작성 시점에서 그 메시지를 실제로 보유한 스레드의 보안 자격을 가져야 한다. 콘택트는 서버(1030)가 메

시지를 읽는 시간까지 변할 수 있어서 콘택 스냅샷(snapshot)의 카피가 만들어져야 하고 메시지와 함께 기록되어야만, 서버(1030)가 그 메시지를 읽을 때, 서버는 실제로 그 스냅샷에 액세스하고, 그 스냅샷을 위장하고 그런 다음 그 메시지를 저술할 수 있으며 그 결과를 다시 보낼 수 있다.

[0089] 메시지 및 그 메시지를 전송한 아이덴티티를 서버(1030)에 비동기적으로 전달하여 서버(1030)가 그들 대신 동작할 수 있도록 하는 프로세스에 추가하여, 그 메시지 및 그 메시지가 게재될 때의 시간에 존재한 그 메시지의 게재자(poster)의 전체 보안 콘택트 또한 상기 프로세스의 일부가 된다. 이러한 방식으로, 나중에 더 많은 보안 판단이 이루어질 수 있다. 그러나, 당업자들이라면 본 발명이 본 명세서에 기술된 메시지 큐를 이용하는 것으로 제한하려는 의도가 아님을 인식할 것이다. 오히려, 서버에게 스레드의 콘택트 체인을 제공하는 다른 기술이 가능하다.

[0090] 프로세스가 시작할 때, 프로세스에는 어떤 아이덴티티가 배정되고 그 프로세스 내 각각의 스레드에게 그 아이덴티티가 주어진다. 시스템 상에서 스레드들이 다른 프로세스들로 이동할 때, 이들 스레드들은 이들의 다른 프로세스들의 아이덴티티들을 누적하여 스레드들에 관련한 보안 판단이 개시될 때 이들 아이덴티티들 모두가 스레드의 액세스 특권을 결정하기 위해 교차될 수 있도록 한다. 또한, 프로세스가 어떤 특정한 서버에 특정하다는 것을 의미하지 않는다. 그 보다는, 어떤 서버라도 이 모델을 이용할 수 있다. 예를 들어, WINDOWS® 오퍼레이팅 시스템에서, 만일 데이터 토큰이 입력되면, 특정한 특권, 즉, 위장 특권이 존재하여야 한다. 위장은 최악의 경우 권리가 낮아지는 결과를 가져오기 때문에 위장이 허용된다.

[0091] 다수의 호출자를 포함하는 시나리오에서, 데스크탑에서, 예를 들어, 애플리케이션 콜들이 오퍼레이팅 시스템에 의해 익스포트된 파일 또는 어떤 다른 API들을 생성할 때, 스레드는 커널 모드에서 사용자의 경계에서 정지하고 커널 내 또 다른 스레드는 스핀오프할 것이며 또한 그 API에 필요한 행위를 수행할 것이며 그런 다음 콜을 다시 스레드로 리턴할 것이다. 따라서, 데스크탑에서 스레드는 실제로 하나의 프로세스에서 다른 프로세스로 천이한다. 반면, WINDOWS® CE 오퍼레이팅 시스템에서, 애플리케이션이 API 콜을 발생할때, 스레드는 실제로 서버 프로세스로 이동한다.

[0092] 전형적으로, 서버 프로세스들은 클라이언트들보다 특권이 더 높다. 따라서, 만일 애플리케이션이 레지스트리를 오픈하기 위한 레지스터 오픈 키를 호출하면 그리고 특정 레지스트리가 낮은 특권의 애플리케이션이 읽을 수 없는 키로 보호되고 있다면, 스레드 상의 콜 및 스레드는 실제로 레지스트리 API들을 실행하는 서버 프로세스로 이동할 것이다. 만일 그 서버 프로세스가 액세스 키를 가지고 있다면, 이것은 호출자의 보안 콘택트가 고려되지 않은 것이기 때문에 같은 종류의 행위를 수행한다. 그러나, 만일 호출자의 보안 콘택트가 현재 보안 콘택트와 비교되면, 그 보호된 자원에의 액세스가 거부될 것이다.

[0093] 테스크탑 WINDOWS® 오퍼레이팅 시스템 및 Windows Mobile® 오퍼레이팅 시스템은 다른데, 그 이유는 Windows Mobile® 오퍼레이팅 시스템의 스레드는 실제로 이동하며 보안 콘택트 정보는 그 스레드 내에 유지되어야 하기 때문이다. 그래야만 자원에의 액세스가 요청될 때 전체 콘택트에 기반한 판단이 이루어질 수 있다. 데스크탑의 경우, 사용자 모드에서 커널 모드로 천이할 시점에서, 현재 스레드 콘택트들은 카피되어야 하며 그런 다음 커널 스레드에서, 호출자가 천이되고 마치 호출자를 대신하여 완료되는 것처럼 행위가 완료된다. 일단 행위가 완료되면, 원래의 보안 콘택트들로 다시 천이가 수행되고 그 결과는 다시 호출자에게 리턴된다.

[0094] 특정 스레드의 보안 콘택트들은 직렬화 및 역직렬화된다(serialized and deserialized). Windows Mobile® 오퍼레이팅 시스템에서, 예를 들어, 데이터베이스 레코드들은 그 데이터베이스 레코드들을 갱신하고 편집할 수 있는 주체가 제한되도록 보호된다. 보안을 제공하기 위해, 레코드들의 보안 콘택트들은 레코드들과 함께 저장된다. 따라서, 예를 들어, 레코드를 갱신하라는 호출이 들어올 때, 만일 호출자가 요청한 데이터베이스 동작을 수행하도록 충분히 액세스할 수 있는지를 알아보기 위해 레코드 및 호출자의 보안 콘택트들이 전달될 수 있다. 이러한 프로세스를 성취하기 위해, 오프라인 데이터베이스가 일관성 있기 때문에 보안 콘택트들은 오프라인 데이터베이스에서 처리될 수 있어야 한다. 따라서, Windows Mobile® 오퍼레이팅 시스템을 구동하는 장치가 그렇게 설정되며 이 장치가 다시 기동할 때, 데이터베이스 레코드들이 동일한 콘택트들을 그대로 보존하고 있는지가 판단된다. 호출자가 식별될 때, 그 콘택트 체인 내 모든 계정들이 결정되며 그리고 나서 파일로서 저장된다. 새로운 호출이 들어오면, 파일이 다시 생성되며 호출자가 콘택트 체인 내 계정들의 서브세트임을 보장하기 위해 저장된 파일과 비교된다.

[0095] 만일 애플리케이션이 서버를 호출하면, 서버는 스레드에서 행위를 실행한다. 상이한 스레드에서는, 모든 자원

들이 호출자를 대신하여 액세스되도록 호출자는 위장된다. 클라이언트로부터 보안 콘택트들의 스냅샷은 기본적으로 발신자의 이름, 현재 콘택트 체인 내 모든 호출자들, 어떤 계정 또는 어떤 상이한 계정들이 콘택트 체인에 있는지 그리고 상이한 서버로 이동했는지를 읽는 과정을 포함할 것이다. 예를 들어, 애플리케이션 4RXE는 서버를 호출할 수 있으며 그러면 동일한 스레드가 또 다른 서버를 호출한다. 따라서, 콘택트 체인에는 다수의 서버가 있을 수 있다. 그러나, 이들 서버들 모두 적절히 액세스하여야 한다. 따라서, 콘택트 스냅샷이 행해질 때, 현재 콘택트 내에 있는 모든 콘택트가 캡쳐되어야 한다.

[0096] 특히, 스레드가 시스템을 통해 이동함에 따라, 그의 보안 콘택트들은 변하게 되는데, 이것은 스레드가 상이한 서버들을 통해 이동하고 상이한 권리를 갖기 때문이다. 이들 권리 모두 교차되는지 결정되어야 한다. 어느 한 시점의 스냅샷을 행하는 역량은 보안을 결정하는데 필요하다. 다시 말해서, 보안 체크가 이루어질 때, 검증되는 모든 ID들은 저장될 필요가 있으며 그러므로 체크는 비동기적으로 또는 추후에도 실행될 수 있다.

[0097] 전술한 시스템 및 컴포넌트는 네트워크형 환경, 분산형 환경 또는 다른 컴퓨터 구현 환경의 부분으로서 구현될 수 있다. 시스템 및 컴포넌트는 유선, 무선, 및/또는 통신 네트워크들의 조합을 통해 통신할 수 있다. 데스크탑 컴퓨터, 랩탑, 핸드헬드, 또는 다른 스마트 장치를 포함하는 다수의 클라이언트 컴퓨팅 장치들은 시스템의 일부와 상호작용할 수 있으며 및/또는 시스템의 일부로서 포함될 수 있다. 대안의 실시예에서, 희망하는 구현에 따라 여러 컴포넌트들이 조합 및/또는 구성될 수 있다. 다른 실시예 및 구성이 이용가능하다.

[0098] 이제 도 11을 참조하면, 다음의 설명에서 본 발명의 실시예가 구현될 수 있는 적합한 컴퓨팅 환경의 간략하고, 일반적인 설명을 제공하고자 한다. 본 발명이 개인용 컴퓨터에서 오퍼레이팅 시스템에서 구동하는 프로그램 모듈들과 관련하여 실행하는 프로그램 모듈들의 일반적 문맥 내에서 기술될 것이지만, 당업자들이라면 본 발명이 또한 다른 형태의 컴퓨터 시스템 및 프로그램 모듈들과도 함께 구현될 수 있음을 인식할 것이다.

[0099] 일반적으로, 프로그램 모듈은 특정 작업을 수행하거나 또는 특정한 추상 데이터 형태(abstract data types)를 실행하는 루틴, 프로그램, 컴포넌트, 데이터 구조, 및 다른 형태의 구조를 포함한다. 더욱이, 당업자들이라면 본 발명이 핸드헬드 장치, 멀티프로세서 시스템, 마이크로 프로세서 기반 또는 프로그램 가능 소비자 전자장치, 미니컴퓨터, 및 메인프레임 컴퓨터 등을 포함하는 다른 컴퓨터 시스템 구성과 함께 실시될 수 있음을 인식할 것이다. 본 발명은 또한 통신 네트워크를 통해 링크된 원격 프로세싱 장치에 의해 작업들이 수행되는 분산 컴퓨팅 환경에서도 실시될 수 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈은 로컬 및 원격 메모리 스토리지 장치에 배치될 수 있다.

[0100] 이제 도 11을 참조하면, 본 발명의 실시예의 예시적인 동작 환경이 기술될 것이다. 도 11에 도시된 바와 같이, 컴퓨터(1100)는 하나 이상의 애플리케이션 프로그램을 실행할 수 있는 범용 데스크탑, 랩탑, 핸드헬드, 또는 다른 형태의 컴퓨터를 포함한다. 컴퓨터(1100)는 적어도 하나의 중앙 처리 유닛("CPU")(1108), 랜덤 액세스 메모리("RAM")(1118) 및 리드 온리 메모리("ROM")(1120)를 포함하는 시스템 메모리(1112), 및 메모리를 CPU(1108)에 연결하는 시스템 버스(1110)를 포함한다. 컴퓨터 내 구성요소들 사이에서 정보를, 이를 테면, 기동 중에 전송해주는 기본 루틴들을 포함하는 기본 입력/출력 시스템은 ROM(1120)에 저장된다. 컴퓨터(1100)는 오퍼레이팅 시스템(1132), 애플리케이션 프로그램, 및 다른 프로그램 모듈을 저장하는 대용량 스토리지 장치(1114)를 더 포함한다.

[0101] 대용량 스토리지 장치(1114)는 버스(1110)에 연결된 대용량 스토리지 장치 제어기(도시되지 않음)를 통해 CPU(1108)에 연결된다. 대용량 스토리지 장치(1114) 및 그의 연관된 컴퓨터 판독가능 매체는 컴퓨터(1100)에게 비휘발성 스토리지를 제공한다. 비록 본 명세서에서 포함된 컴퓨터 판독가능 매체에 대한 기술은 하드디스크 또는 CD-ROM 드라이브와 같은 대용량 스토리지 장치를 지칭할지라도, 당업자들은 컴퓨터 판독가능 매체가 컴퓨터(1100)에 의해 액세스될 수 있거나 또는 이용될 수 있는 어떤 이용가능한 매체라도 될 수 있음을 인식하여야 한다.

[0102] 예를 들어, 제한하지 않지만, 컴퓨터 판독가능 매체는 컴퓨터 스토리지 매체 및 통신 매체를 포함할 수 있다. 컴퓨터 스토리지 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 다른 데이터와 같은 정보의 저장을 위한 어떠한 방법 또는 기술로도 구현된 휘발성 및 비휘발성의 제거가능 및 제거가능하지 않은 매체를 포함한다. 컴퓨터 스토리지 매체는, 다음으로 제한되지 않지만, 원하는 정보를 저장하는데 사용될 수 있고 컴퓨터(1100)에 의해 액세스될 수 있는 RAM, ROM, EPROM, EEPROM, 플래시 메모리 또는 다른 고상 메모리 기술, CD-ROM, 디지털 다기능 디스크("DVD"), 또는 다른 광학 스토리지, 자기 카세트, 자기 테이프, 자기 디스크 스토리지 또는 다른 자기 스토리지 장치들, 또는 어떤 다른 매체를 포함한다.

[0103] 본 발명의 여러 실시예들에 따르면, 컴퓨터(1100)는, 예를 들어, 네트워크(1104), 이를 테면, 로컬 네트워크, 인터넷 등을 통해 원격 컴퓨터와의 논리적 연결을 이용하여 네트워크형 환경에서 동작할 수 있다. 컴퓨터(1100)는 버스(1110)에 연결된 네트워크 인터페이스 유닛(1116)을 통해 네트워크(1104)에 연결될 수 있다. 네트워크 인터페이스 유닛(1116)은 또한 다른 형태의 네트워크 및 원격 컴퓨팅 시스템에 연결하는데에도 이용될 수 있음을 인식하여야 한다. 컴퓨터(1100)는 또한 키보드, 마우스 등(도시되지 않음)을 포함하는 다수의 다른 장치들로부터 입력을 수신하고 그 입력을 처리하는 입력/출력 제어기(1122)를 포함할 수 있다. 유사하게, 입력/출력 제어기(1122)는 디스플레이 스크린, 프린터, 또는 다른 형태의 출력 장치에 출력을 제공할 수 있다.

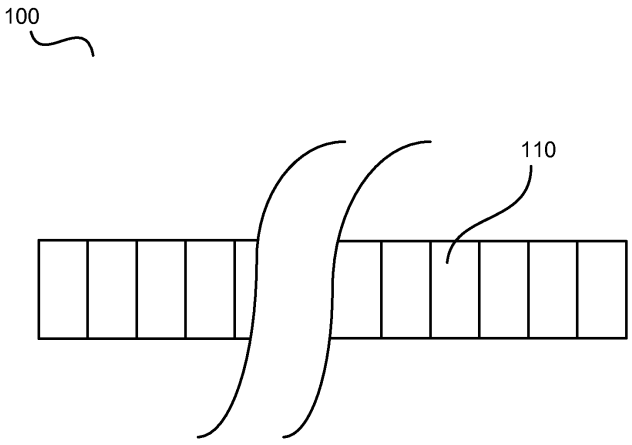
[0104] 앞에서 간략하게 설명한 바와 같이, 네트워크형 개인용 컴퓨터의 동작을 제어하기에 적합한 오퍼레이팅 시스템(1132), 이를 테면, 워싱턴, 레드몬드 소재의 마이크로소프트 코포레이션으로부터 입수가능한 WINDOWS® 오퍼레이팅 시스템을 포함하는 다수의 프로그램 모듈 및 데이터 파일은 컴퓨터(1100)의 대용량 스토리지 장치(1114) 및 RAM(1118)에 저장될 수 있다. 대용량 스토리지 장치(1114) 및 RAM(1118)은 또한 하나 이상의 프로그램 모듈을 저장할 수 있다. 특히, 대용량 스토리지 장치(1114) 및 RAM(1118)은 클라이언트 애플리케이션 프로그램(1140) 및 다른 소프트웨어 애플리케이션(1142)을 저장할 수 있다. 도 11에 예시된 바와 같은 컴퓨터(1100)는 도 1 내지 도 10에서 기술된 본 발명의 실시예들에 따른 보안 기반구조를 제공하는 명령어를 실행하도록 구성될 수 있다.

[0105] 본 발명의 여러 실시예들은 (1) 일련의 컴퓨터로 구현된 행위 또는 컴퓨팅 시스템에서 구동하는 프로그램 모듈로서, 및/또는 (2) 컴퓨팅 시스템 내에서 상호연결된 머신 로직 회로 또는 회로 모듈로서 구현될 수 있음을 인식하여야 한다. 이러한 구현은 본 발명을 구현하는 컴퓨팅 시스템의 성능 요건에 따른 선택의 문제이다. 따라서, 관련된 알고리즘을 포함하는 논리 동작은 다양하게 동작, 구조적 장치, 행위 또는 모듈이라고 지칭될 수 있다. 당업자들이라면 본 명세서에서 기술된 청구범위 내에서 언급된 바와 같이 본 발명의 정신 및 범주를 일탈함이 없이도 이러한 동작, 구조적 장치, 행위 및 모듈은 소프트웨어, 펌웨어, 특수목적 디지털 로직, 및 이들의 어떤 조합으로 구현될 수 있음을 인식할 것이다.

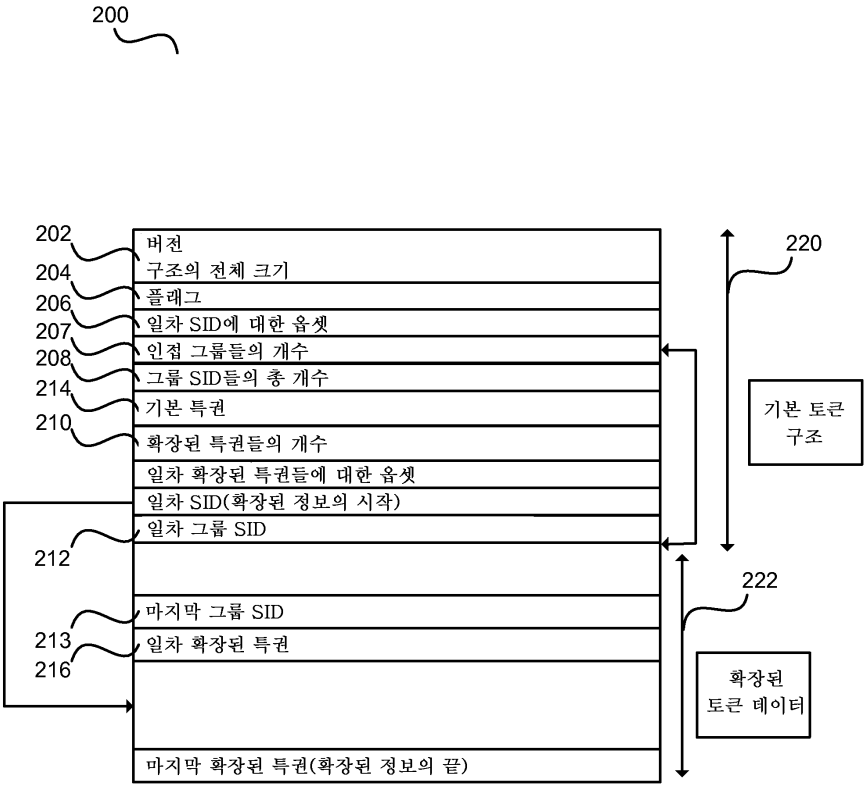
[0106] 비록 본 발명이 여러 예시적인 실시예와 관련하여 기술되었을지라도, 당업자들은 다음의 청구범위의 범주 내에서 많은 변형이 이루어질 수 있음을 이해할 것이다. 따라서, 본 발명의 범주는 어떤 방식으로든지 전술한 설명에 의해 제한되는 것으로 의도하지 않으며, 그 대신 전적으로 다음의 청구범위를 참조하여 결정된다.

도면

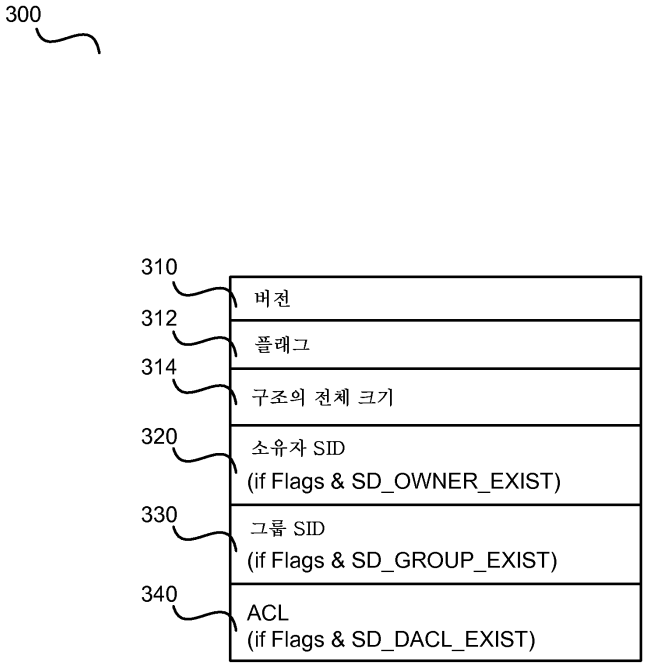
도면1



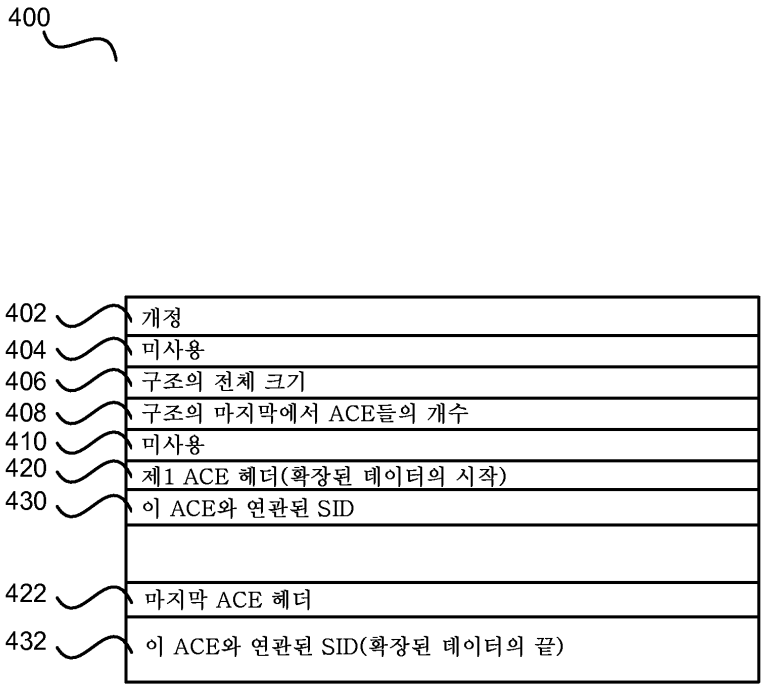
도면2


200
202
버전
구조의 전체 크기
204
플래그
206
일차 SID에 대한 옵셋
207
인접 그룹들의 개수
208
그룹 SID들의 총 개수
214
기본 특권
210
확장된 특권들의 개수
일차 확장된 특권들에 대한 옵셋
일차 SID(확장된 정보의 시작)
일차 그룹 SID
212
마지막 그룹 SID
213
일차 확장된 특권
216
마지막 확장된 특권(확장된 정보의 끝)
220
기본 토큰 구조
222
확장된 토큰 데이터

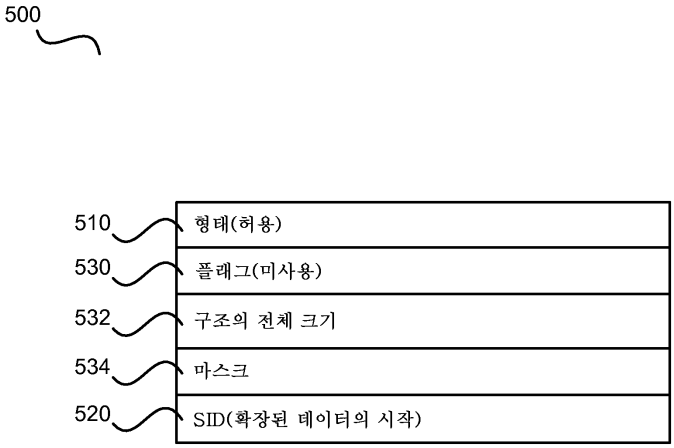
도면3


300
310
버전
312
플래그
314
구조의 전체 크기
320
소유자 SID
(if Flags & SD_OWNER_EXIST)
330
그룹 SID
(if Flags & SD_GROUP_EXIST)
340
ACL
(if Flags & SD_DACL_EXIST)

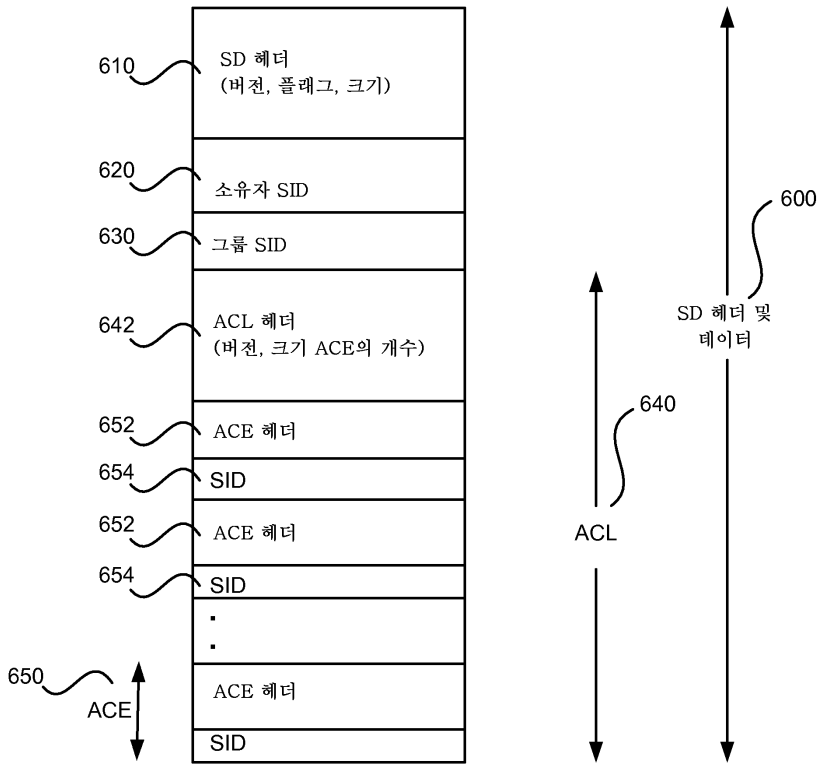
도면4


400
402
개정
404
미사용
406
구조의 전체 크기
408
구조의 마지막에서 ACE들의 개수
410
미사용
420
제1 ACE 헤더(확장된 데이터의 시작)
430
이 ACE와 연관된 SID
422
마지막 ACE 헤더
432
이 ACE와 연관된 SID(확장된 데이터의 끝)

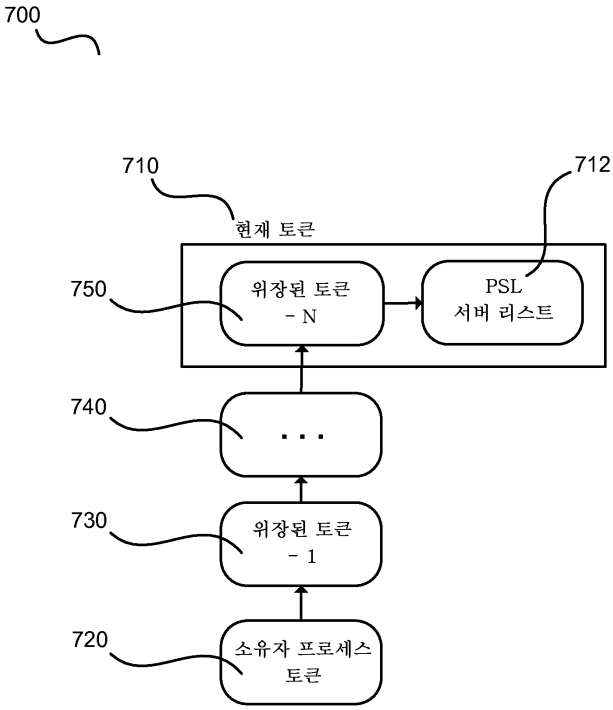
도면5


500
510
형태(허용)
530
플래그(미사용)
532
구조의 전체 크기
534
마스크
520
SID(확장된 데이터의 시작)

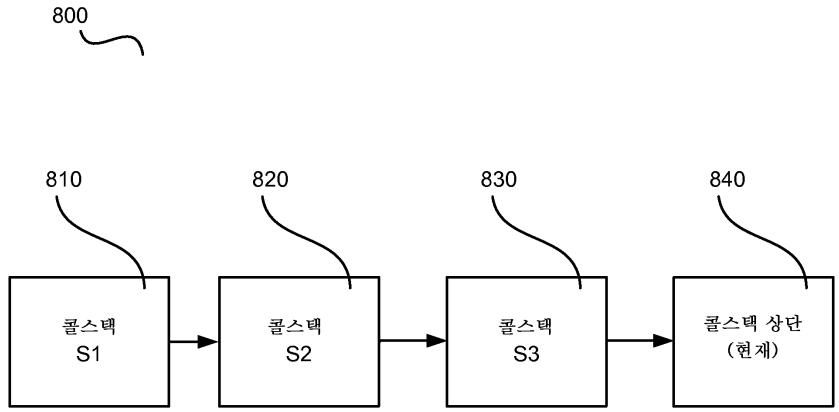
도면6


610
SD 헤더
(버전, 플래그, 크기)
620
소유자 SID
630
그룹 SID
642
ACL 헤더
(버전, 크기 ACE의 개수)
652
ACE 헤더
654
SID
652
ACE 헤더
654
SID
650
ACE
ACE 헤더
SID
600
SD 헤더 및
데이터
640
ACL

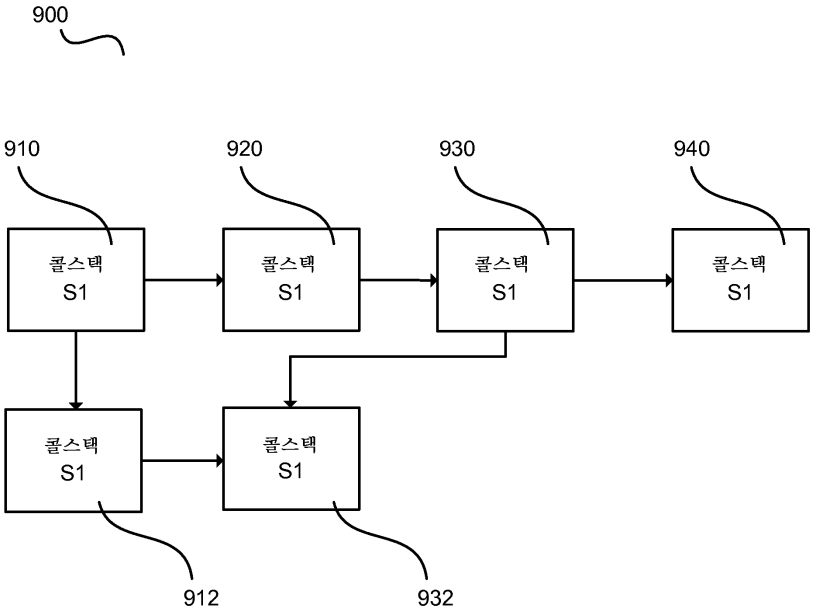
도면7


700
710
현재 토큰
712
750
위장된 토큰
- N
PSL
서버 리스트
740
. . .
730
위장된 토큰
- 1
720
소유자 프로세스
토큰

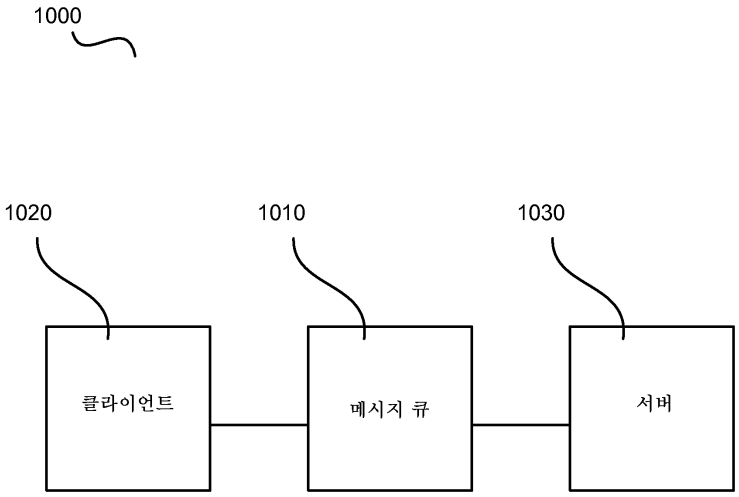
도면8


800
810
820
830
840
콜스택
S1
콜스택
S2
콜스택
S3
콜스택 상단
(현재)

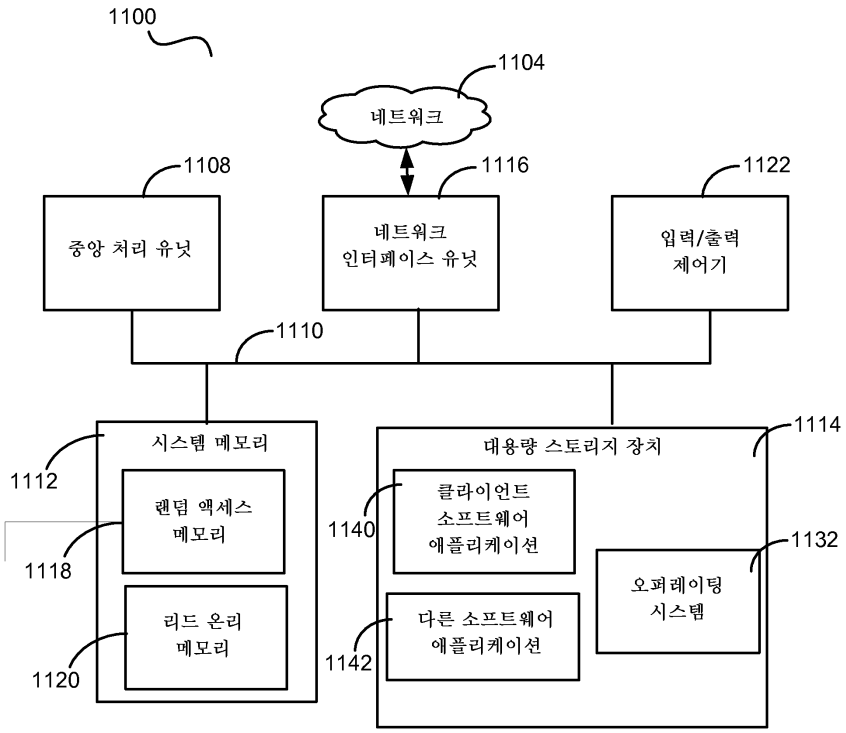
도면9


900
910
920
930
940
콜스택
S1
콜스택
S1
콜스택
S1
콜스택
S1
콜스택
S1
콜스택
S1
912
932

도면10


1000
1020
1010
1030
클라이언트
메시지 큐
서버

도면11


1100
1104
네트워크
1108
1116
1122
중앙 처리 유닛
네트워크
인터페이스 유닛
입력/출력
제어기
1110
시스템 메모리
1112
랜덤 액세스
메모리
1118
리드 온리
메모리
1120
대용량 스토리지 장치
1114
1140
클라이언트
소프트웨어
애플리케이션
1132
오퍼레이팅
시스템
다른 소프트웨어
애플리케이션
1142