(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2021년11월05일
(11) 등록번호 10-2323276
(24) 등록일자 2021년11월02일

(51) 국제특허분류(Int. Cl.)
G06F 8/72 (2018.01) *G06F 21/54* (2013.01)
G06F 8/41 (2018.01)
(52) CPC특허분류
G06F 8/72 (2013.01)
G06F 21/54 (2013.01)
(21) 출원번호 10-2021-0035161
(22) 출원일자 2021년03월18일
심사청구일자 2021년03월18일
(56) 선행기술조사문헌
KR101436741 B1*
KR1020140075785 A*
KR1020170045437 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
세종대학교산학협력단
서울특별시 광진구 능동로 209 (군자동, 세종대학교)
(72) 발명자
신지선
서울특별시 광진구 능동로 209 세종대학교 대양AI센터 708호
이성훈
서울특별시 동대문구 한천로 248, 103동 102호(휘경동, 주공아파트)
(74) 대리인
두호특허법인

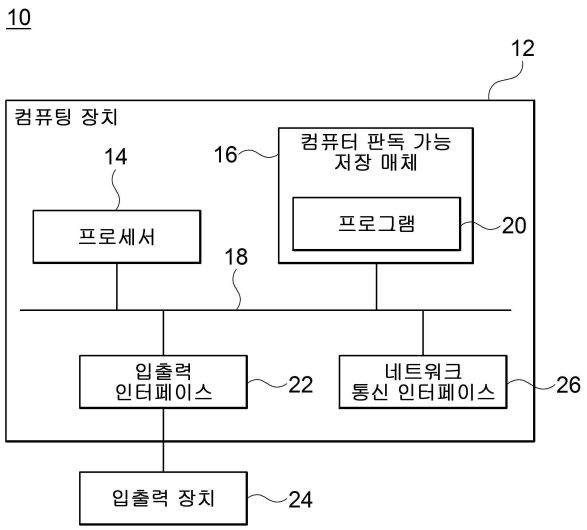
전체 청구항 수 : 총 8 항 심사관 : 지정훈

(54) 발명의 명칭 **애플리케이션 변환 방법 및 장치**

(57) 요 약

애플리케이션 변환 방법 및 장치가 개시된다. 일 실시예에 따른 애플리케이션 변환 방법은, 타겟 애플리케이션 및 상기 타겟 애플리케이션에 추가할 보안 기능을 호출하는 호출 애플리케이션을 각각 디컴파일(decompile)하여 상기 타겟 애플리케이션에 대한 제1 바이트코드(bytecode) 및 상기 호출 애플리케이션에 대한 제2 바이트코드를 생성하는 단계; 상기 제2 바이트코드에서 상기 보안 기능을 호출하기 위한 호출 코드를 추출하는 단계; 상기 추출된 호출 코드를 상기 제1 바이트코드에 추가하는 단계; 상기 호출 코드가 추가된 제1 바이트코드에서 상기 호출 코드와 나머지 코드 사이의 충돌을 방지하기 위한 코드 최적화를 수행하는 단계; 및 상기 코드 최적화가 수행된 제1 바이트코드를 리컴파일(recompile)하는 단계를 포함한다.

대 표 도 - 도1



(52) CPC특허분류
G06F 8/41 (2013.01)
G06F 8/47 (2013.01)
G06F 8/48 (2013.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 1345321135
과제번호 2020R1A6A1A03038540
부처명 교육부
과제관리(전문)기관명 한국연구재단
연구사업명 이공학학술연구기반구축(R&D)
연구과제명 자율지능무인비행체연구소
기 여 율 1/2
과제수행기관명 세종대학교
연구기간 2020.06.01 ~ 2021.02.28
이 발명을 지원한 국가연구개발사업
과제고유번호 1711116145
과제번호 2018-0-01423-003
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보통신방송혁신인재양성(R&D)
연구과제명 지능형 비행로봇 융합기술 연구
기 여 율 1/2
과제수행기관명 세종대학교 산학협력단
연구기간 2020.01.01 ~ 2020.12.31

명 세 서

청구범위

청구항 1

타겟 애플리케이션 및 상기 타겟 애플리케이션에 추가할 보안 기능을 호출하는 호출 애플리케이션을 각각 디컴파일(decompile)하여 상기 타겟 애플리케이션에 대한 제1 바이트코드(bytecode) 및 상기 호출 애플리케이션에 대한 제2 바이트코드를 생성하는 단계;

상기 제2 바이트코드에서 상기 보안 기능을 호출하기 위한 호출 코드를 추출하는 단계;

상기 추출된 호출 코드를 상기 제1 바이트코드에 추가하는 단계;

상기 호출 코드가 추가된 제1 바이트코드에서 상기 호출 코드와 나머지 코드 사이의 충돌을 방지하기 위한 코드 최적화를 수행하는 단계; 및

상기 코드 최적화가 수행된 제1 바이트코드를 리컴파일(recompile)하는 단계를 포함하는, 애플리케이션 변환 방법.

청구항 2

청구항 1에 있어서,

상기 호출 코드는, 상기 보안 기능을 제공하는 보안 애플리케이션에서 상기 보안 기능을 호출하기 위한 코드인, 애플리케이션 변환 방법.

청구항 3

청구항 1에 있어서,

상기 코드 최적화를 수행하는 단계는, 상기 나머지 코드에 포함된 하나 이상의 파라미터를 수정하여 상기 코드 최적화를 수행하는, 애플리케이션 변환 방법.

청구항 4

청구항 1에 있어서,

상기 코드 최적화를 수행하는 단계는, 상기 나머지 코드에서 상기 호출 코드에 포함된 파라미터와 변수 명이 중복되는 파라미터를 식별하는 단계; 및

상기 식별된 파라미터와 관련된 코드를 수정하여 상기 코드 최적화를 수행하는 단계를 포함하는, 애플리케이션 변환 방법.

청구항 5

하나 이상의 프로세서; 및

상기 하나 이상의 프로세서에 의해 실행되는 하나 이상의 프로그램을 저장하는 메모리를 포함하는 장치로서,

상기 하나 이상의 프로그램은,

타겟 애플리케이션 및 상기 타겟 애플리케이션에 추가할 보안 기능을 호출하는 호출 애플리케이션을 각각 디컴파일(decompile)하여 상기 타겟 애플리케이션에 대한 제1 바이트코드(bytecode) 및 상기 호출 애플리케이션에

대한 제2 바이트코드를 생성하는 단계;

상기 제2 바이트코드에서 상기 보안 기능을 호출하기 위한 호출 코드를 추출하는 단계;

상기 추출된 호출 코드를 상기 제1 바이트코드에 추가하는 단계;

상기 호출 코드가 추가된 제1 바이트코드에서 상기 호출 코드와 나머지 코드 사이의 충돌을 방지하기 위한 코드 최적화를 수행하는 단계; 및

상기 코드 최적화가 수행된 제1 바이트코드를 리컴파일(recompile)하는 단계를 실행하기 위한 명령어들을 포함하는, 장치.

청구항 6

청구항 5에 있어서,

상기 호출 코드는, 상기 보안 기능을 제공하는 보안 애플리케이션에서 상기 보안 기능을 호출하기 위한 코드인, 장치.

청구항 7

청구항 5에 있어서,

상기 코드 최적화를 수행하는 단계는, 상기 나머지 코드에 포함된 하나 이상의 파라미터를 수정하여 상기 코드 최적화를 수행하는, 장치.

청구항 8

청구항 5에 있어서,

상기 코드 최적화를 수행하는 단계는, 상기 나머지 코드에서 상기 호출 코드에 포함된 파라미터와 변수 명이 중복되는 파라미터를 식별하는 단계; 및

상기 식별된 파라미터와 관련된 코드를 수정하여 상기 코드 최적화를 수행하는 단계를 포함하는, 장치.

발명의 설명

기 술 분 야

[0001] 본 발명의 실시예들은 앱 래핑(app wrapping) 기술과 관련된다.

배 경 기 술

[0002] 스마트 기기의 대중화에 따라 많은 기업에서 개인의 스마트 기기를 업무에 활용하는 BYOD(Bring Your Own Device) 정책을 도입하고 있다. BYOD는 업무 효율성 향상, 비용 절감 등의 장점이 있지만, 개인의 스마트 기기에 업무용 데이터가 저장됨에 따라 기밀 데이터 유출과 같은 보안 위협이 제기되고 있다.

[0003] 이러한 보안 위협을 해결하기 위해 EMM(Enterprise Mobility Management)과 같은 솔루션을 도입하여 스마트 기기 보안 및 기업 정책에 맞는 맞춤형 보안 서비스를 제공하고 있다. 기업 정책에 따라 보안 서비스를 제공하기 위해서는 애플리케이션의 원본 소스 코드를 분석하여 보안 기능이 필요한 위치에 보안 기능을 추가해야 한다. 이러한 작업은 기업 정책이 변경될 때마다 수행해야 하므로 정책에 따라 애플리케이션을 커스터마이징(customizing)하기 위해 매우 오랜 기간이 소요된다.

[0004] 한편, 애플리케이션의 원본 소스 코드 없이 보안 기능을 적용하는 방안으로 앱 래핑(App wrapping)이 있다. 앱 래핑은 바이트코드(Bytecode) 수준에서 애플리케이션의 원본 소스 코드 없이 애플리케이션 파일에 직접 보안 기능 코드를 추가하는 기술이다. 하지만, 바이트코드 수준에서 보안 기능 코드를 추가한 이후에 애플리케이션이

원활하게 작동되게 하는 것은 어려운 문제로 알려져 있다.

[0005] 특히, 안드로이드(Android)의 바이트코드인 스말리 코드(smali code)는 레지스터(register) 기반으로 각 메소드(method) 영역마다 사용할 수 있는 메모리(memory)가 한정되어 있어, 보안 기능 코드를 추가한 이후에 오버플로우(overflow) 에러가 발생하여 애플리케이션이 정상적으로 실행되지 않는다.

선행기술문헌

특허문헌

[0006] (특허문헌 0001) 대한민국 공개특허공보 제10-2018-0053872호 (2018. 05. 24. 공개)

발명의 내용

해결하려는 과제

[0007] 본 발명의 실시예들은 애플리케이션에 보안 기능을 추가하기 위한 애플리케이션 변환 방법 및 장치를 제공하기 위한 것이다.

과제의 해결 수단

[0008] 일 실시예에 따른 애플리케이션 변환 방법은, 타겟 애플리케이션 및 상기 타겟 애플리케이션에 추가할 보안 기능을 호출하는 호출 애플리케이션을 각각 디컴파일(decompile)하여 상기 타겟 애플리케이션에 대한 제1 바이트코드(bytecode) 및 상기 호출 애플리케이션에 대한 제2 바이트코드를 생성하는 단계; 상기 제2 바이트코드에서 상기 보안 기능을 호출하기 위한 호출 코드를 추출하는 단계; 상기 추출된 호출 코드를 상기 제1 바이트코드에 추가하는 단계; 상기 호출 코드가 추가된 제1 바이트코드에서 상기 호출 코드와 나머지 코드 사이의 충돌을 방지하기 위한 코드 최적화를 수행하는 단계; 및 상기 코드 최적화가 수행된 제1 바이트코드를 리컴파일(recompile)하는 단계를 포함한다.

[0009] 상기 호출 코드는, 상기 보안 기능을 제공하는 보안 애플리케이션에서 상기 보안 기능을 호출하기 위한 코드일 수 있다.

[0010] 상기 코드 최적화를 수행하는 단계는, 상기 나머지 코드에 포함된 하나 이상의 파라미터를 수정하여 상기 코드 최적화를 수행할 수 있다.

[0011] 상기 코드 최적화를 수행하는 단계는, 상기 나머지 코드에서 상기 호출 코드에 포함된 파라미터와 변수 명이 중복되는 파라미터를 식별하는 단계; 및 상기 식별된 파라미터와 관련된 코드를 수정하여 상기 코드 최적화를 수행하는 단계를 포함할 수 있다.

[0012] 일 실시예에 따른 장치는, 하나 이상의 프로세서; 및 상기 하나 이상의 프로세서에 의해 실행되는 하나 이상의 프로그램을 저장하는 메모리를 포함하는 장치로서, 상기 하나 이상의 프로그램은, 타겟 애플리케이션 및 상기 타겟 애플리케이션에 추가할 보안 기능을 호출하는 호출 애플리케이션을 각각 디컴파일(decompile)하여 상기 타겟 애플리케이션에 대한 제1 바이트코드(bytecode) 및 상기 호출 애플리케이션에 대한 제2 바이트코드를 생성하는 단계; 상기 제2 바이트코드에서 상기 보안 기능을 호출하기 위한 호출 코드를 추출하는 단계; 상기 추출된 호출 코드를 상기 제1 바이트코드에 추가하는 단계; 상기 호출 코드가 추가된 제1 바이트코드에서 상기 호출 코드와 나머지 코드 사이의 충돌을 방지하기 위한 코드 최적화를 수행하는 단계; 및 상기 코드 최적화가 수행된 제1 바이트코드를 리컴파일(recompile)하는 단계를 실행하기 위한 명령어들을 포함한다.

[0013] 상기 호출 코드는, 상기 보안 기능을 제공하는 보안 애플리케이션에서 상기 보안 기능을 호출하기 위한 코드일 수 있다.

[0014] 상기 코드 최적화를 수행하는 단계는, 상기 나머지 코드에 포함된 하나 이상의 파라미터를 수정하여 상기 코드 최적화를 수행할 수 있다.

[0015] 상기 코드 최적화를 수행하는 단계는, 상기 나머지 코드에서 상기 호출 코드에 포함된 파라미터와 변수 명이 중복되는 파라미터를 식별하는 단계; 및 상기 식별된 파라미터와 관련된 코드를 수정하여 상기 코드 최적화를 수행하는 단계를 포함할 수 있다.

발명의 효과

[0016] 본 발명의 실시예들에 따르면, 애플리케이션의 원본 소스 코드 없이 바이트코드 수준에서 애플리케이션에 다양한 보안 기능들을 기존 코드와 충돌 없이 적용시킬 수 있다.

도면의 간단한 설명

[0017] 도 1은 예시적인 실시예들에서 사용되기에 적합한 컴퓨팅 장치를 포함하는 컴퓨팅 환경을 예시하여 설명하기 위한 블록도

도 2는 일 실시예에 다른 애플리케이션 변환 방법의 순서도

도 3은 제1 바이트코드에 추가된 호출 코드와 제1 바이트코드의 나머지 코드 사이의 충돌을 예시적으로 설명하기 위한 도면

도 4는 일 실시예에 따른 코드 최적화를 예시적으로 설명하기 위한 도면

발명을 실시하기 위한 구체적인 내용

[0018] 이하, 도면을 참조하여 본 발명의 구체적인 실시형태를 설명하기로 한다. 이하의 상세한 설명은 본 명세서에서 기술된 방법, 장치 및/또는 시스템에 대한 포괄적인 이해를 돕기 위해 제공된다. 그러나 이는 예시에 불과하며 본 발명은 이에 제한되지 않는다.

[0019] 본 발명의 실시예들을 설명함에 있어서, 본 발명과 관련된 공지기술에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략하기로 한다. 그리고, 후술되는 용어들은 본 발명에서의 기능을 고려하여 정의된 용어들로서 이는 사용자, 운용자의 의도 또는 관례 등에 따라 달라질 수 있다. 그러므로 그 정의는 본 명세서 전반에 걸친 내용을 토대로 내려져야 할 것이다. 상세한 설명에서 사용되는 용어는 단지 본 발명의 실시예들을 기술하기 위한 것이며, 결코 제한적이어서는 안 된다. 명확하게 달리 사용되지 않는 한, 단수 형태의 표현은 복수 형태의 의미를 포함한다. 본 설명에서, "포함" 또는 "구비"와 같은 표현은 어떤 특성들, 숫자들, 단계들, 동작들, 요소들, 이들의 일부 또는 조합을 가리키기 위한 것이며, 기술된 것 이외에 하나 또는 그 이상의 다른 특성, 숫자, 단계, 동작, 요소, 이들의 일부 또는 조합의 존재 또는 가능성을 배제하도록 해석되어서는 안 된다.

[0020] 도 1은 예시적인 실시예들에서 사용되기에 적합한 컴퓨팅 장치를 포함하는 컴퓨팅 환경을 예시하여 설명하기 위한 블록도이다.

[0021] 도 1에 도시된 실시예에서, 각 컴포넌트들은 이하에 기술된 것 이외에 상이한 기능 및 능력을 가질 수 있고, 이하에 기술된 것 이외에도 추가적인 컴포넌트를 포함할 수 있다.

[0022] 컴퓨팅 환경(10)은 컴퓨팅 장치(12)를 포함한다. 일 실시예에서, 컴퓨팅 장치(12)는 후술할 애플리케이션 변환 방법을 수행하기 위한 장치일 수 있다.

[0023] 컴퓨팅 장치(12)는 적어도 하나의 프로세서(14), 컴퓨터 판독 가능 저장 매체(16) 및 통신 버스(18)를 포함한다. 프로세서(14)는 컴퓨팅 장치(12)로 하여금 앞서 언급된 예시적인 실시예에 따라 동작하도록 할 수 있다. 예컨대, 프로세서(14)는 컴퓨터 판독 가능 저장 매체(16)에 저장된 하나 이상의 프로그램들을 실행할 수 있다. 상기 하나 이상의 프로그램들은 하나 이상의 컴퓨터 실행 가능 명령어를 포함할 수 있으며, 상기 컴퓨터 실행 가능 명령어는 프로세서(14)에 의해 실행되는 경우 컴퓨팅 장치(12)로 하여금 예시적인 실시예에 따른 동작들을 수행하도록 구성될 수 있다.

[0024] 컴퓨터 판독 가능 저장 매체(16)는 컴퓨터 실행 가능 명령어 내지 프로그램 코드, 프로그램 데이터 및/또는 다른 적합한 형태의 정보를 저장하도록 구성된다. 컴퓨터 판독 가능 저장 매체(16)에 저장된 프로그램(20)은 프로세서(14)에 의해 실행 가능한 명령어의 집합을 포함한다. 일 실시예에서, 컴퓨터 판독 가능 저장 매체(16)는 메모리(랜덤 액세스 메모리와 같은 휘발성 메모리, 비휘발성 메모리, 또는 이들의 적절한 조합), 하나 이상의 자기 디스크 저장 디바이스들, 광학 디스크 저장 디바이스들, 플래시 메모리 디바이스들, 그 밖에 컴퓨팅 장치(12)에 의해 액세스되고 원하는 정보를 저장할 수 있는 다른 형태의 저장 매체, 또는 이들의 적합한 조합일 수 있다.

[0025] 통신 버스(18)는 프로세서(14), 컴퓨터 판독 가능 저장 매체(16)를 포함하여 컴퓨팅 장치(12)의 다른 다양한 컴

포넌트들을 상호 연결한다.

[0026] 컴퓨팅 장치(12)는 또한 하나 이상의 입출력 장치(24)를 위한 인터페이스를 제공하는 하나 이상의 입출력 인터페이스(22) 및 하나 이상의 네트워크 통신 인터페이스(26)를 포함할 수 있다. 입출력 인터페이스(22) 및 네트워크 통신 인터페이스(26)는 통신 버스(18)에 연결된다. 입출력 장치(24)는 입출력 인터페이스(22)를 통해 컴퓨팅 장치(12)의 다른 컴포넌트들에 연결될 수 있다. 예시적인 입출력 장치(24)는 포인팅 장치(마우스 또는 트랙패드 등), 키보드, 터치 입력 장치(터치패드 또는 터치스크린 등), 음성 또는 소리 입력 장치, 다양한 종류의 센서 장치 및/또는 촬영 장치와 같은 입력 장치, 및/또는 디스플레이 장치, 프린터, 스피커 및/또는 네트워크 카드와 같은 출력 장치를 포함할 수 있다. 예시적인 입출력 장치(24)는 컴퓨팅 장치(12)를 구성하는 일 컴포넌트로서 컴퓨팅 장치(12)의 내부에 포함될 수도 있고, 컴퓨팅 장치(12)와는 구별되는 별개의 장치로 컴퓨팅 장치(12)와 연결될 수도 있다.

[0027] 도 2는 일 실시예에 따른 애플리케이션 변환 방법의 순서도이다.

[0028] 도 2에 도시된 애플리케이션 변환 방법은 변환 대상인 타겟 애플리케이션을 보안 기능이 추가된 애플리케이션으로 변환하기 위한 것으로 예를 들어, 도 1에 도시된 컴퓨팅 장치(12)에 의해 수행될 수 있다.

[0029] 도 2를 참조하면, 우선, 컴퓨팅 장치(12)는 타겟 애플리케이션과 호출 애플리케이션을 각각 디컴파일(decompile)하여 타겟 애플리케이션에 대한 바이트코드(bytecode)(이하, 제1 바이트코드) 및 호출 애플리케이션에 대한 바이트코드(이하, 제2 바이트코드)를 생성한다(210).

[0030] 이때, 타겟 애플리케이션은 예를 들어, EMM(Enterprise Mobility Management), MDM(Mobile Device Management) 등과 같이 보안 애플리케이션에 의해 제공되는 하나 이상의 보안 기능이 적용될 애플리케이션을 의미할 수 있다.

[0031] 또한, 호출 애플리케이션은 보안 애플리케이션에 의해 제공되는 보안 기능을 호출하기 위한 호출 코드를 포함하는 애플리케이션을 의미할 수 있다.

[0032] 한편, 일 실시예에 따르면, 타겟 애플리케이션 및 호출 애플리케이션은 예를 들어, 구글(Google) 사에서 공개한 안드로이드(Android) 운영체제에서 실행되는 APK(Android Application Package) 형식의 파일일 수 있으며, 제1 바이트코드 및 제2 바이트코드는 예를 들어, 스말리 코드(Smali code)일 수 있으나 반드시 이에 한정되는 것은 아니다.

[0033] 또한, 일 실시예에 따르면, 보안 애플리케이션에 의해 제공되는 보안 기능은 예를 들어, API(Application Programming Interface) 형태일 수 있으나, 실시예에 따라, 안드로이드 액티비티(Android Activity)의 형태일 수 있다.

[0034] 이후, 컴퓨팅 장치(12)는 제2 바이트코드에서 타겟 애플리케이션에 추가할 보안 기능을 호출하기 위한 호출 코드를 추출한 후, 추출된 호출 코드를 제1 바이트코드에 추가한다(220).

[0035] 이후, 컴퓨팅 장치(12)는 제1 바이트코드에 추가된 호출 코드와 제1 바이트코드의 나머지 코드 사이의 충돌을 방지하기 위한 코드 최적화를 수행한다(230).

[0036] 이때, 일 실시예에 따르면, 컴퓨팅 장치(12)는 제1 바이트코드의 나머지 코드에 포함된 하나 이상의 파라미터를 수정하여 코드 최적화를 수행할 수 있다.

[0037] 구체적으로, 컴퓨팅 장치(12)는 제 바이트코드에 추가된 호출 코드와 제1 바이트코드의 나머지 코드에서 변수명이 중복되는 파라미터를 식별하고, 나머지 코드에서 해당 파라미터와 관련된 코드를 수정하여 코드 최적화를 수행할 수 있다.

[0038] 도 3은 제1 바이트코드에 추가된 호출 코드와 제1 바이트코드의 나머지 코드 사이의 충돌을 예시적으로 설명하기 위한 도면이다.

[0039] 도 3을 참조하면, 제1 바이트코드(300)에 추가된 호출 코드(310) 중 일부 코드(320)에서 사용되는 파라미터 "p0"는 제1 바이트코드(330)의 나머지 코드 중 일부 코드(330)에서 사용되는 포함된 파라미터 "p0"와 동일한 변수 명을 가지고 있으므로, 애플리케이션 실행 시 에러가 발생하게 된다. 따라서, 호출 코드(310)가 제1 바이트코드(300)에 추가된 경우, 나머지 코드에서 사용되는 파라미터와 충돌이 발생하지 않도록 코드 최적화를 수행할 필요가 있다.

[0040] 도 4는 일 실시예에 따른 코드 최적화를 예시적으로 설명하기 위한 도면이다.

[0041] 도 4를 참조하면, 컴퓨팅 장치(12)는 코드 최적화를 위해 우선 제1 바이트코드의 나머지 코드에서 제1 바이트코드에 추가된 호출 코드에 포함된 파라미터와 변수 명이 중복되는 파라미터인 “activity”를 전역 변수로 선언(410)할 수 있다. 이때, 전역 변수 명은 다른 변수 명과 중복되지 않도록 결정될 수 있다.

[0042] 이후, 컴퓨팅 장치(12)는 나머지 코드에서 해당 변수 명을 가진 파라미터를 “this”로 지정(420)하고, 해당 파라미터가 호출되는 메소드에서 410에서 선언된 전역 변수 명이 사용되도록 코드를 변경(430)함으로써 코드 최적화를 수행할 수 있다.

[0043] 다시 도 2를 참조하면, 컴퓨팅 장치(12)는 코드 최적화가 완료된 경우, 코드 최적화가 수행된 제1 바이트코드를 리컴파일(recompile)하여 보안 기능이 추가된 타겟 애플리케이션을 생성한다(240).

[0044] 이때, 실시예에 따라, 생성된 타겟 애플리케이션에는 전자 서명을 통해 생성된 서명 정보가 추가될 수 있다.

[0045] 한편, 도 2에 도시된 순서도에서 적어도 일부의 단계들은 순서를 바꾸어 수행되거나, 다른 단계와 결합되어 함께 수행되거나, 생략되거나, 세부 단계들로 나뉘어 수행되거나, 또는 도시되지 않은 하나 이상의 단계가 부가되어 수행될 수 있다.

[0046] 이상에서 대표적인 실시예를 통하여 본 발명에 대하여 상세하게 설명하였으나, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자는 전술한 실시예에 대하여 본 발명의 범주에서 벗어나지 않는 한도 내에서 다양한 변형이 가능함을 이해할 것이다. 그러므로 본 발명의 권리범위는 설명된 실시예에 국한되어 정해져서는 안 되며, 후술하는 특허청구범위뿐만 아니라 이 특허청구범위와 균등한 것들에 의해 정해져야 한다.

부호의 설명

[0047] 10: 컴퓨팅 환경

12: 컴퓨팅 장치

14: 프로세서

16: 컴퓨터 판독 가능 저장 매체

18: 통신 버스

20: 프로그램

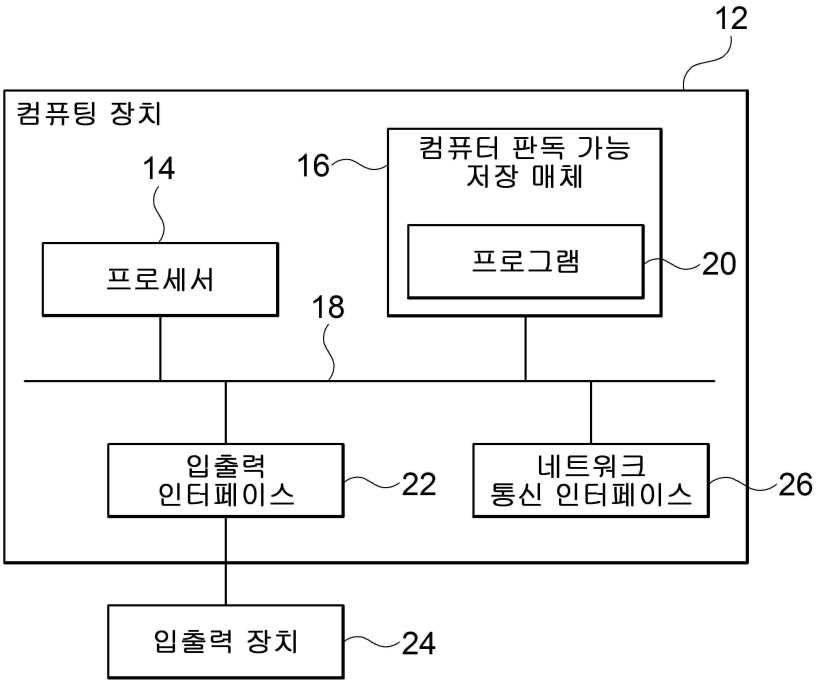
22: 입출력 인터페이스

24: 입출력 장치

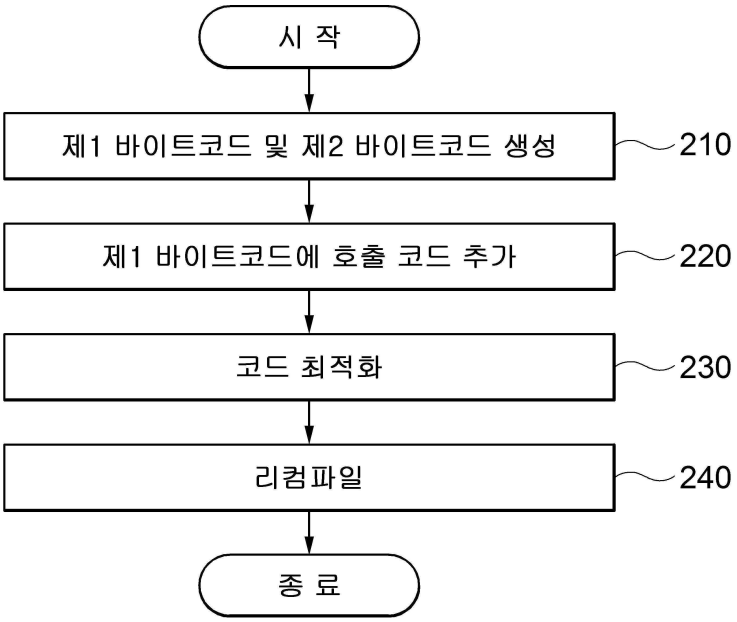
26: 네트워크 통신 인터페이스

도면

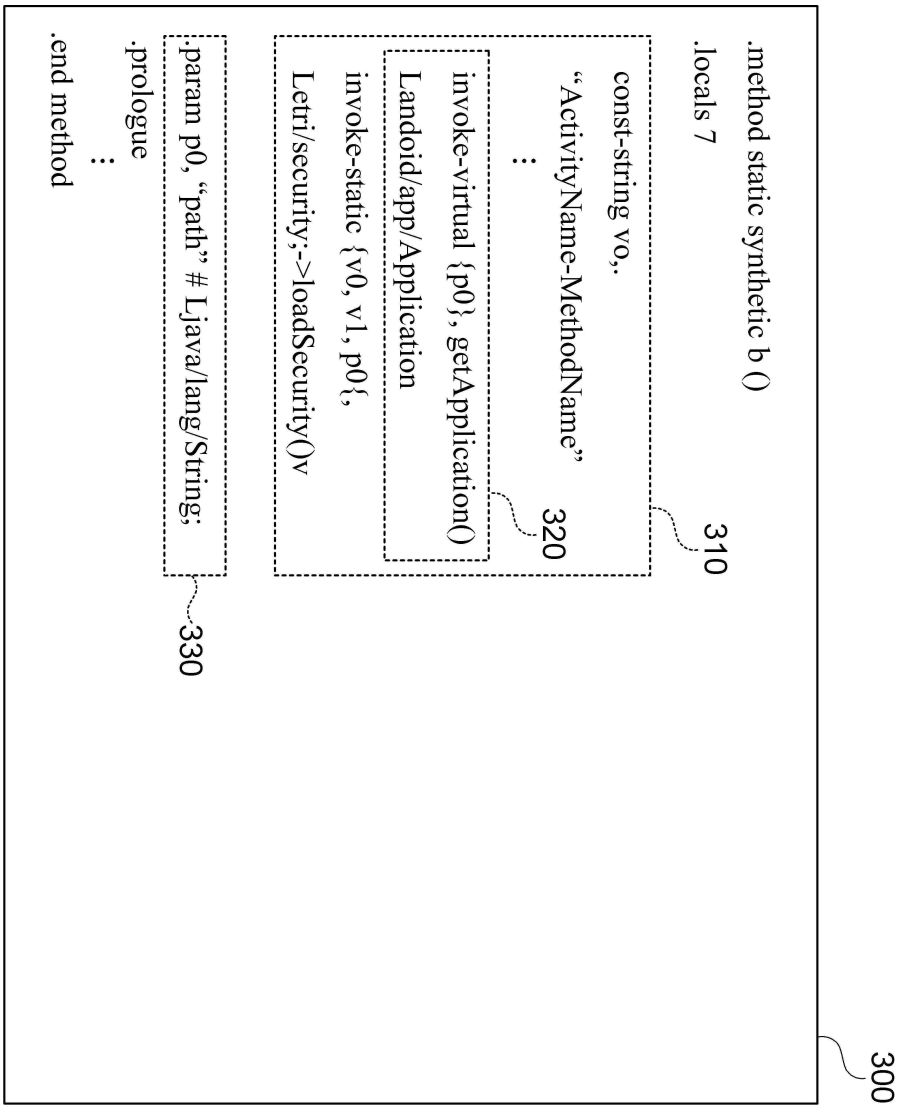
도면1


10
12
컴퓨팅 장치
14
프로세서
16
컴퓨터 판독 가능
저장 매체
프로그램
20
18
입출력
인터페이스
22
네트워크
통신 인터페이스
26
입출력 장치
24

도면2


시 작
제1 바이트코드 및 제2 바이트코드 생성
210
제1 바이트코드에 호출 코드 추가
220
코드 최적화
230
리컴파일
240
종 료

도면3


.method static synthetic b ()
.locals 7
const-string v0,.
"ActivityName-MethodName"
:
invoke-virtual {p0}, getApplication()
Landoid/app/Application
invoke-static {v0, v1, p0{,
Letri/security;->loadSecurity()v
.param p0, "path" # Ljava/lang/String;
.prologue
:
.end method
310
320
330
300

도면4

public static Activity activity;
410
@override
protected void onCreate(Bundle saveInstanceState) {
super.onCreate(saveInstanceState);
setContentView(R.layout.activity_main);
activity=this;
420
String currentmethod="test.etri.mainactivity-onCreate";
new JavaReflection().loadTxt(currentmethod, getApplicationContext(), activity: this);
}
public static void a(Context paramContext) {
String currentmethod="test.etri. mainactivity-onCreate";
430
new JavaReflection().loadTxt(currentmethod,paramContext.getApplicationContext(), activity);
paramContext.startActivity(new Intent(paramContext, MainActivity.class));
400