안드로이드 악성코드 분석 시스템

숭실대학교 Soongsil University

안드로이드 악성코드 분석 시스템

CNSL(COMMUNICATION NETWORK SEQURITY LABORATORY) 연구실

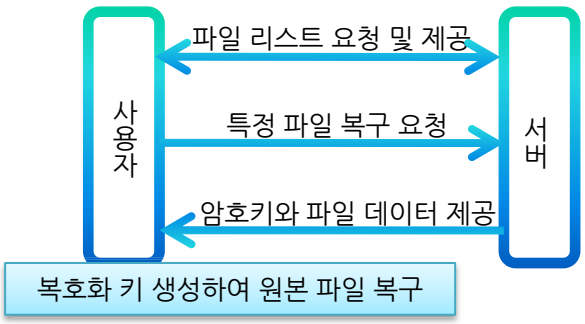
- 주요 연구분야
  - 안드로이드 악성코드 분석 플랫폼에 대한 연구
  - 클라우드 기반 보안 서비스 플랫폼 개발
  - 유무선 환경에서의 보안과 인증에 대한 연구

1. 머신러닝을 이용한 안드로이드 악성코드 탐지 기술

- 악성코드에서 사용되는 API와 권한 특징을 추출하여 머신러닝을 구축
- 98%이상의 악성코드 탐지율을 보임

2. 클라우드 기반 랜섬웨어 복구 시스템

- 사용자 파일 생성 시 자동으로 클라우드 서버에 암호화하여 백업
- 저장된 시점과 데이터의 Hash값을 기반으로 원본파일 복구



[클라우드 서버로부터 원본파일을 얻는 과정]

3. 클라우드 기반 보안 서비스 플랫폼 SecaaS

- 테넌트가 보안 서비스를 선택적으로 사용할 수 있도록 하는 기술
- SFC를 이용한 보안 서비스의 효율적 활용 가능

특허 정보

- 안드로이드 보안을 위한 듀오 OS모델 및 이를 탑재한 모바일 장치, 이를 이용한 모바일 장치의 보안 방법 (등록 제 10-1895893 호)
- 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼 및 이를 이용한 모바일 장치의 보안 방법 (등록 제 10-1857009 호)

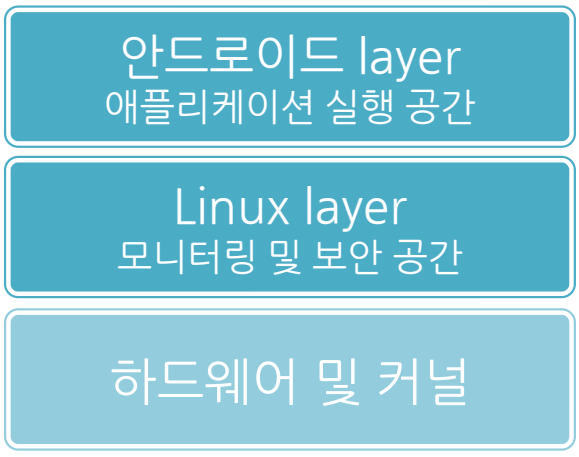
기술 개요

- **악성코드 분석 시 기존의 가상화 방식과 모바일 기기 방식은 비효율적임**

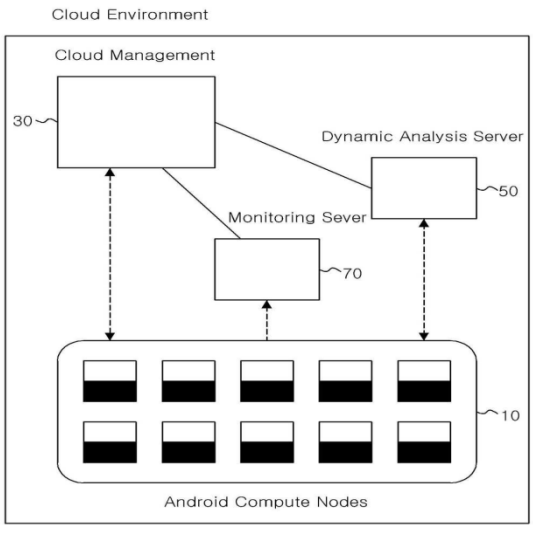
  - 지능화된 악성코드는 실제 모바일 하드웨어와 에뮬레이터를 구별함
  - 모바일 기기 사용은 비용이 많이 발생하고, 프로세스를 자동화하거나 클라우드 환경에 적용하기가 어려움

- **혁신적인 안드로이드 악성코드 진단 컨테이너 플랫폼**

  - 안드로이드 운영체제 레벨 가상화에 기반한 듀오 OS 모델을 이용하여 악성 어플리케이션을 탐지하는 기술
  - 루트되지 않은 안드로이드 공간에서 악성 어플리케이션이 실행되도록 하고 같은 시간에 리눅스 공간에서 탐지 수행
  - 애플리케이션과 이를 실행하는데 필요한 요소를 묶은 컨테이너 기술 사용



[듀오 OS 모델의 구조]

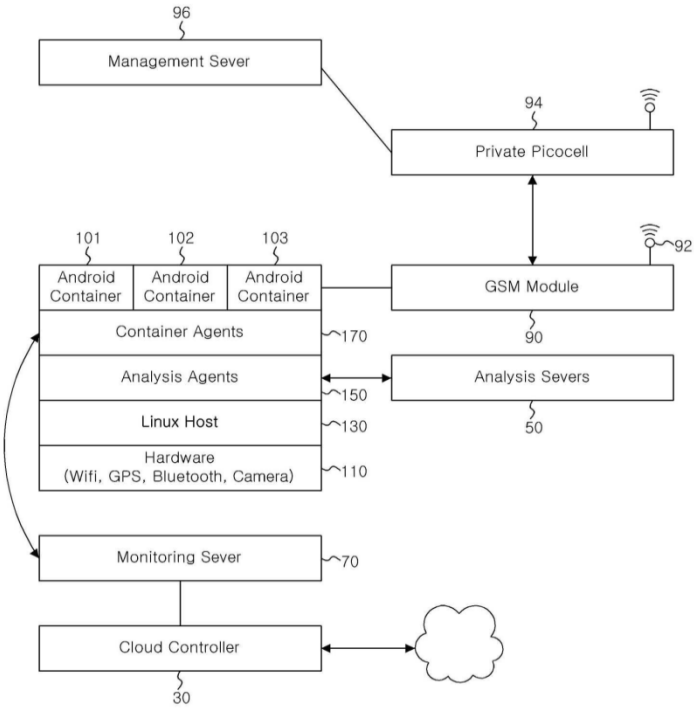


[컨테이너 플랫폼의 개념도]

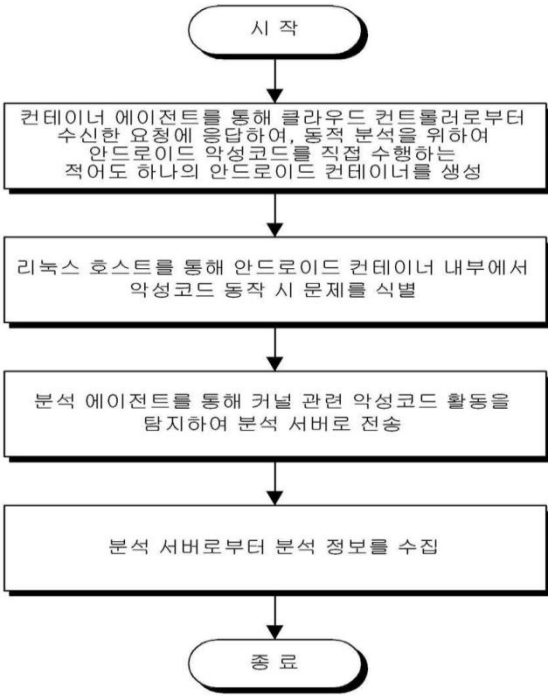
기술 특장점

▪ 컨테이너 기술을 이용

- 하드웨어 가상화 기술이 필요 없음
- 이식성이 높음
- 다양한 공격에서부터 간단히 복구가 가능
- 장애 발생 시 컨테이너를 하나 새로 만들어 빠른 대응이 가능
- 집적도가 높고, 배포 속도가 빠름



[안드로이드 컨테이너 플랫폼의 블록도]



[컨테이너 플랫폼 보안 방법의 흐름도]

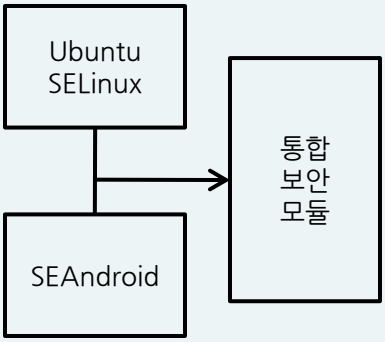
▪ 듀오 OS 시스템의 장점

- 빠른 속도로 더 안전하게 유해 프로그램 탐지 가능
- 모니터링 및 보안 애플리케이션을 쉽게 개발할 수 있는 리눅스 공간 관리 기능 제공
- 안드로이드 공간에 치명적인 공격으로부터 시스템 복구를 도와줄 수 있음
- 분석 환경 탐지 기능 무력화
- 안드로이드 장치 상에서 안정성과 유연성이 뛰어남

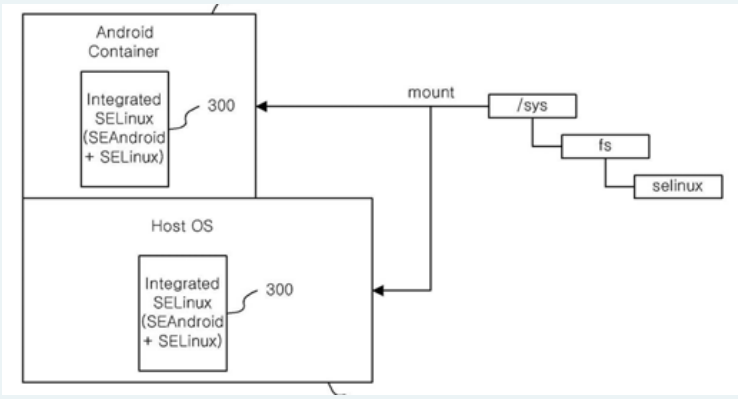
기술개발 현황

▪ 리눅스 보안 기술과 안드로이드 보안 기술을 결합한 통합 보안 모듈 개발

- SELinux와 SEAndroid 파일 시스템의 충돌 문제 해결
- 통합 보안 모듈을 리눅스 호스트와 안드로이드 컨테이너가 공유하는 디렉토리에 마운트함으로써 보안 시스템을 구축함



[통합 보안 모듈 생성]



[안드로이드 컨테이너 플랫폼의 보안 시스템 구축 상태]

▪ 안드로이드 악성코드 분석기 A-POT 개발

- 지능화된 악성코드 분석을 위한 Bare-metal 솔루션
- APK 파일을 분석 상자에 넣고 클릭 버튼을 누르면 분석 완료
- wifi, uSIM 등의 센서를 이용하여 실제 폰과 같은 환경 제공

A-POT	
지능형 악성코드 실행률	93%
상업용 앱 실행률	98%
악성코드 랜덤 샘플 실행률	75.6%
구글 플레이 스토어 앱 실행률	92.08%
악성코드 탐지율	96.21%

[A-POT 이미지]

[악성코드 분석 실행 화면]

종래 기술 대비 우수성

종래 기술

- 지능화된 악성 앱이 실제 모바일 기기와 에뮬레이터를 구별함
- ADB와 같은 안드로이드 분석 툴 필요
- 분석 후 환경을 재구성하는 데 시간이 오래 걸림
- Native 분석 불가능
- Flow-Graph 제공 안함

개발 기술

- **악성 앱이 본 기기를 에뮬레이터로 구별하지 못함**
- **다수의 악성코드를 동시에 분석함**
- **2분 이내로 초기화 가능**
- **Native 코드 영역 분석가능**
- **분석 보고서 및 Flow-Graph 제공**

기술 활용 전망

- **시장규모 및 전망**

- 2023년 국내시장은 1,037억 원, 해외시장은 약 8.5조의 시장 규모 예상
- 국내 시장에 비하여 해외 시장의 증가율이 매우 클 전망

단위 : 억 원

구분	시장 구분	2017년	2020년	2023년	연평균 증가율
제품(기술)	국내 시장	983	1,019	1,037	00.91%
	해외 시장	30,655	69,893	85,588	29.86%

자료 : Anti-Malware Market by Operating System, by Organization Size, by Vertical and by Region - Global Forecast to 2020

- **기술 활용 분야**

- 다양한 App Store에 등록 전 검색기로 활용
- 악성코드 조기 탐지 및 상세 분석에 활용
- 앱 취약점 분석을 통하여, 금융권 App의 정보 유출과 위협 요소에 선제적 대응 가능

보유 특허

No	국가	출원(등록)번호	명칭
1	대한민국	10-2017-0027824	안드로이드 동적 로딩 파일 추출 방법, 이를 수행하기 위한 기록 매체 및 시스템
2	대한민국	10-2017-0008996	안드로이드 악성코드 분석을 위한 컨테이너 플랫폼 및 이를 이용한 모바일 장치의 보안 방법
3	대한민국	10-2017-0005820	안드로이드 보안을 위한 듀오 OS모델 및 이를 탑재한 모바일 장치, 이를 이용한 모바일 장치의 보안 방법
4	대한민국	10-2016-0181213	루팅 탐지 장치 및 방법, 이를 수행하기 위한 기록 매체
5	대한민국	10-2016-0175370	구글 플레이 스토어 루팅 앱 탐지 방법, 이를 수행하기 위한 기록 매체, 클라우드 서버 및 시스템
6	대한민국	10-2016-0173585	애플리케이션 악성 판단 시스템 및 방법, 이를 수행하기 위한 기록 매체
7	대한민국	10-2016-0155743	클라우드 기반의 클라이언트 장치 및 백업 방법, 이를 수행하기 위한 기록매체

수행연구과제

No	과제명	수행 연도	주관기관
1	인터넷 인프라 시스템 기술 개발 및 전문 인력 양성	2017~2020	숭실대학교
2	머신러닝 기반 지능형 악성코드 분석 통합 플랫폼	2017~2019	NSHC
3	맞춤형 보안서비스 제공을 위한 클라우드 기반 지능형 보안 기술 개발	2016~2019	ETRI
4	Optimizing Native Analysis with Android Container	2017	ACMLC
5	Design and Implementation of Log Collection and System Statistics Model for Android Containers in OpenStack	2017	SETH

(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2018년10월24일
(11) 등록번호 10-1895893
(24) 등록일자 2018년08월31일

(51) 국제특허분류(Int. Cl.)
G06F 21/50 (2013.01) *G06F 21/60* (2013.01)
(52) CPC특허분류
G06F 21/50 (2013.01)
G06F 21/60 (2013.01)
(21) 출원번호 10-2017-0005820
(22) 출원일자 2017년01월13일
심사청구일자 2017년01월13일
(65) 공개번호 10-2018-0055627
(43) 공개일자 2018년05월25일
(30) 우선권주장
1020160152410 2016년11월16일 대한민국(KR)
(56) 선행기술조사문헌
KR1020130101648 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
숭실대학교산학협력단
서울특별시 동작구 상도로 369 (상도동)
(72) 발명자
정수환
서울특별시 동작구 상도로 369, 숭실대학교 형남공학관 1105호
차오녹투
서울특별시 동작구 상도로 369, 숭실대학교 형남공학관 1102호
박정수
서울특별시 동작구 상도로 369, 숭실대학교 형남공학관 1102호
(74) 대리인
윤귀상

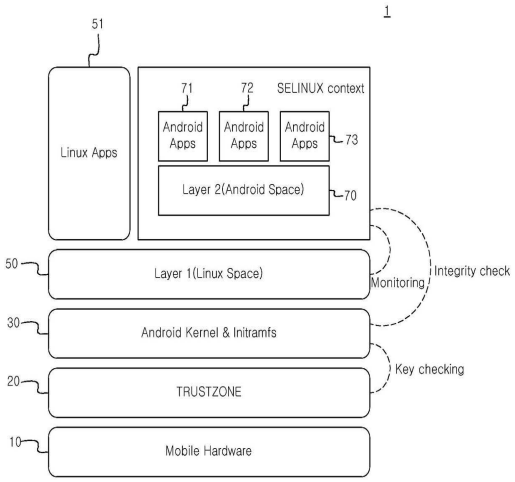
전체 청구항 수 : 총 11 항 심사관 : 문남두

(54) 발명의 명칭 **안드로이드 보안을 위한 듀오 OS 모델 및 이를 탑재한 모바일 장치, 이를 이용한 모바일 장치의 보안 방법**

(57) 요 약

안드로이드 보안을 위한 듀오 OS 모델은, 모바일 장치의 하드웨어와 인터페이스를 위한 드라이버를 제공하는 안드로이드 커널; 리눅스 공간의 커널 부팅 전에 무결성 체크를 위한 보안 키를 저장하는 보안 플랫폼; 모바일 장치의 모니터링 및 보안 기능을 제공하는 리눅스 기반의 운영체제인 제1 레이어; 및 안드로이드 어플리케이션을 통하여 모바일 사용자들과 상호작용하는 사용자 공간층으로 컨테이너 기반의 안드로이드 운영체제인 제2 레이어를 포함한다. 이에 따라, 빠른 속도로 더 안전하게 유해 프로그램을 탐지할 수 있으며, 보안 및 모니터링 어플리케이션을 쉽게 개발하는 수 있는 리눅스 공간 관리 기능을 제공할 수 있다.

대 표 도 - 도1



이 발명을 지원한 국가연구개발사업
과제고유번호 H8501-16-1008 / 1711035246
부처명 미래창조과학부
연구관리전문기관 정보통신기술진흥센터
연구사업명 대학 ICT연구센터 육성 지원사업
연구과제명 클라우드 환경의 스마트 기기와 서비스 보안 기술 개발 및 연구 인력양성
기 여 율 1/2
주관기관 숭실대학교 산학협력단
연구기간 2016.01.01 ~ 2016.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 B0717-16-0108 / 1711037875
부처명 미래창조과학부
연구관리전문기관 정보통신기술진흥센터
연구사업명 정보통신·방송 연구개발 사업
연구과제명 맞춤형 보안서비스 제공을 위한 클라우드 기반 지능형 보안 기술 개발
기 여 율 1/2
주관기관 한국전자통신연구원
연구기간 2016.04.01 ~ 2016.12.31

명 세 서

청구범위

청구항 1

모바일 장치의 하드웨어와 인터페이스를 위한 드라이버를 제공하는 안드로이드 커널;

리눅스 공간의 커널 부팅 전에 무결성 체크를 위한 보안 키를 저장하는 보안 플랫폼;

모바일 장치의 모니터링 및 보안 기능을 제공하는 리눅스 기반의 운영체제인 제1 레이어; 및

안드로이드 어플리케이션을 통하여 모바일 사용자들과 상호작용하는 사용자 공간층으로 컨테이너 기반의 안드로이드 운영체제인 제2 레이어를 포함하고,

상기 제2 레이어는,

상기 제1 레이어에 제공된 호스트 운영체제와 상이한 운영체제가 제공된 LXC(LinuX Containers)를 포함하는, 안드로이드 보안을 위한 듀오 OS 모델을 실행시키기 위하여 비일시적 컴퓨터 판독가능 저장매체에 저장된 컴퓨터 프로그램.

청구항 2

제1항에 있어서,

상기 제1 레이어는 배포, 관리, 보안, 모니터링 및 기타 확장을 위해 사용되는 안드로이드 공간의 툴을 포함하는, 안드로이드 보안을 위한 듀오 OS 모델을 실행시키기 위하여 비일시적 컴퓨터 판독가능 저장매체에 저장된 컴퓨터프로그램.

청구항 3

제1항에 있어서,

상기 제2 레이어는 안드로이드 어플리케이션을 포함하는, 안드로이드 보안을 위한 듀오 OS 모델을 실행시키기 위하여 비일시적 컴퓨터 판독가능 저장매체에 저장된 컴퓨터프로그램.

청구항 4

제1항에 있어서,

MDM(Mobile Device Management, 모바일 단말 관리) 모듈은 제2 레이어의 활동을 통제하기 위해 상기 제1 레이어의 상단에 설치되는, 안드로이드 보안을 위한 듀오 OS 모델을 실행시키기 위하여 비일시적 컴퓨터 판독가능 저장매체에 저장된 컴퓨터프로그램.

청구항 5

제1항에 있어서,

Malware Analysis(악성 코드 분석) 모듈은 제2 레이어에서 동작하는 APK 파일 및 SO 라이브러리를 분석하기 위해 상기 제1 레이어의 상단에 설치되는, 안드로이드 보안을 위한 듀오 OS 모델을 실행시키기 위하여 비일시적 컴퓨터 판독가능 저장매체에 저장된 컴퓨터프로그램.

청구항 6

제1항에 있어서,

Multiple OS single Platform(단일 플랫폼에 멀티 OS)은 상기 제1 레이어의 상단에 설치되는, 안드로이드 보안을 위한 듀오 OS 모델을 실행시키기 위하여 비일시적 컴퓨터 판독가능 저장매체에 저장된 컴퓨터프로그램.

청구항 7

제1항 내지 제6항 중 어느 하나의 항에 따른 안드로이드 보안을 위한 듀오 OS 모델을 실행시키기 위하여 매체에 저장된 컴퓨터프로그램을 메모리에 탑재한 모바일 장치.

청구항 8

모바일 장치의 모니터링 및 보안 기능을 제공하는 리눅스 기반의 운영체제인 제1 레이어 및 안드로이드 어플리케이션을 통하여 모바일 사용자들과 상호작용하는 사용자 공간층으로 컨테이너 기반의 안드로이드 운영체제인 제2 레이어를 포함하여, 상기 제2 레이어가 상기 제1 레이어에 제공된 호스트 운영체제와 상이한 운영체제가 제공된 LXC(LinuX Containers)를 포함하는, 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법은,

안드로이드 커널에서 보안 키를 저장하는 보안 플랫폼을 통해 리눅스 공간의 커널 부팅 전에 안드로이드 컨테이너의 무결성을 검사하는 단계;

상기 제2 레이어에 설치된 안드로이드 어플리케이션을 구동시키는 단계; 및

상기 제1 레이어에 설치된 모니터링 및 보안 어플리케이션을 통해 상기 제2 레이어에서 구동되는 안드로이드 어플리케이션의 유해 여부를 탐지하는 단계를 포함하는, 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법.

청구항 9

제8항에 있어서,

상기 제1 레이어에 설치된 관리자 프로그램은 지속적으로 동작하며, 상기 제2 레이어에서 동작하는 APK 파일 및 SO 라이브러리를 분석하는 단계를 더 포함하는, 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법.

청구항 10

제8항에 있어서,

배포, 관리, 보안, 모니터링 및 기타 확장을 위해 사용되는 안드로이드 공간의 툴 및 모바일 장치의 모니터링 및 보안 어플리케이션이 모바일 장치에 다운로드되는 경우, 상기 제1 레이어에 설치하는 단계를 더 포함하는, 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법.

청구항 11

제8항에 있어서,

안드로이드 어플리케이션이 모바일 장치에 다운로드되는 경우, 상기 제2 레이어에 설치하는 단계를 더 포함하는, 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 안드로이드 보안을 위한 듀오 OS 모델 및 이를 탑재한 모바일 장치, 이를 이용한 모바일 장치의 보안 방법에 관한 것으로서, 더욱 상세하게는 모바일 장치를 위한 아키텍처로서 운영체제 레벨 가상화 기술을 사용하여 하나 이상의 운영체제 상단에 다른 운영체제를 실행시키는 기술에 관한 것이다.

배 경 기 술

[0002] 기존 모바일 장치는 운영체제가 동작하는 하나의 층만을 제공하고, 해당 층에서 모니터링 및 보안 어플리케이션이 모바일 보안 상태(Rooting or unrooting, jailbreak or not jailbreak)에 상관없이 정상 및 악성 어플리케이션과 동일한 장소에서 동작하였다.

[0003] 이러한 이유로, 보안 활동을 하는 사람들은 보안 어플리케이션에 대하여 주장하였지만, 이론적으로 보안 어플리케이션은 다른 어플리케이션에 비교하여 어떠한 장점도 가지지 못한다.

[0004] 모바일 운영체제인 안드로이드 경우 리눅스 커널에 발전에도 불구하고 여전히 일반적인 리눅스 운영체제의 어플리케이션과 툴들의 재사용이 가능하지 않다. 또한, 요즘에는 하나의 장치에 멀티 플랫폼이 필요한 경우가 있다.

[0005] AVD(안드로이드 가상 장치) 또는 기타 QEMU 기반 에뮬레이터 등의 안드로이드 에뮬레이터는 일반적으로 악성 코드 분석을 위해 사용된다. 그러나, 대부분의 안드로이드 악성 코드는 안드로이드 에뮬레이터를 감지하는 안티 에뮬레이터 기술을 적용하고 있기 때문에 안드로이드 에뮬레이터 우회 기술이 적용된 악성코드에 대한 분석이 어렵다.

[0006] Ubuntu Touch, Condroid, Cells은 각기 다른 목적으로 안드로이드 컨테이너를 지원하는 솔루션이다.

[0007] Condroid와 Cells은 각기 다른 용도로 사용할 수 있는 하나 이상의 안드로이드 환경을 제공하기 위해 운영체제 수준의 가상화 방법을 이용한다. Ubuntu Touch는 LXC 컨테이너와 libhybris의 도움으로 안드로이드에 Ubuntu-liked 어플리케이션을 제공하는 것을 목적으로 하는 솔루션이다.

[0008] Cells과 Condroid는 안드로이드 컨테이너를 구축하기 위하여 안드로이드용 LXC(LinuX Containers) 기술을 이용한다. 그러나, 안드로이드용 LXC가 LXC의 서브 프로젝트이고 주된 관심사가 아니기 때문에, 안드로이드 소스 코드용 LXC에는 호환성 문제가 존재한다.

[0009] 예를 들면, 안드로이드 LXC 몇 가지 버전에서는 Systemd 서비스 존재를 확인하고, Systemd 서비스가 존재하지 않을 경우 오류가 발생한다. 또한 LXC에서 개발된 안드로이드 환경의 프로젝트들의 경우, 리눅스 환경에서 제공하는 다수의 어플리케이션을 실행시키지 못한다는 한계점이 있다. 또한, Ubuntu Touch는 완전히 격리되지 않은 부분 컨테이너 솔루션을 제공함으로써 GUI 솔루션을 해결하는데 초점을 맞추고 있을 뿐이다.

선행기술문헌

특허문헌

[0010] (특허문헌 0001) KR 10-1216581 B1

(특허문헌 0002) KR 10-1223981 B1

비특허문헌

[0011] (비특허문헌 0001) 김호중, 컨테이너에 대한 6가지 오해, IDG Summary, 2016.04.28.

발명의 내용

해결하려는 과제

[0012] 이에, 본 발명의 기술적 과제는 이러한 점에서 착안된 것으로 본 발명의 목적은 안드로이드 보안을 위한 듀오 OS 모델을 제공하는 것이다.

[0013] 본 발명의 다른 목적은 상기 안드로이드 보안을 위한 듀오 OS 모델을 탑재한 모바일 장치를 제공하는 것이다.

[0014] 본 발명의 또 다른 목적은 상기 안드로이드 보안을 위한 듀오 OS 모델을 이용한 모바일 장치의 보안 방법을 수행하기 위한 장치를 제공하는 것이다.

과제의 해결 수단

[0015] 상기한 본 발명의 목적을 실현하기 위한 일 실시예에 따른 안드로이드 보안을 위한 듀오 OS 모델은, 모바일 장치의 하드웨어와 인터페이스를 위한 드라이버를 제공하는 안드로이드 커널; 리눅스 공간의 커널 부팅 전에 무결성 체크를 위한 보안 키를 저장하는 보안 플랫폼; 모바일 장치의 모니터링 및 보안 기능을 제공하는 리눅스 기반의 운영체제인 제1 레이어; 및 안드로이드 어플리케이션을 통하여 모바일 사용자들과 상호작용하는 사용자 공간층으로 컨테이너 기반의 안드로이드 운영체제인 제2 레이어를 포함한다.

[0016] 본 발명의 실시예에서, 상기 제1 레이어는 배포, 관리, 보안, 모니터링 및 기타 확장을 위해 사용되는 안드로이드 공간의 툴을 포함할 수 있다.

[0017] 본 발명의 실시예에서, 상기 제2 레이어는 안드로이드 어플리케이션을 포함할 수 있다.

[0018] 본 발명의 실시예에서, MDM(Mobile Device Management, 모바일 단말 관리) 모듈은 제2 레이어의 활동을 통제하기 위해 상기 제1 레이어의 상단에 설치될 수 있다.

[0019] 본 발명의 실시예에서, Malware Analysis(악성 코드 분석) 모듈은 제2 레이어에서 동작하는 APK 파일 및 SO 라이브러리를 분석하기 위해 상기 제1 레이어의 상단에 설치될 수 있다.

[0020] 본 발명의 실시예에서, Multiple OS single Platform(단일 플랫폼에 멀티 OS)은 상기 제1 레이어의 상단에 설치될 수 있다.

[0021] 상기한 본 발명의 다른 목적을 실현하기 위한 일 실시예에 따른 모바일 장치는, 상기 안드로이드 보안을 위한 듀오 OS 모델을 탑재하고 있다.

[0022] 상기한 본 발명의 또 다른 목적을 실현하기 위한 일 실시예에 따른 모바일 장치의 모니터링 및 보안 기능을 제공하는 리눅스 기반의 운영체제인 제1 레이어 및 안드로이드 어플리케이션을 통하여 모바일 사용자들과 상호작용하는 사용자 공간층으로 컨테이너 기반의 안드로이드 운영체제인 제2 레이어를 포함하는 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법은, 안드로이드 커널에서 보안 키를 저장하는 보안 플랫폼을 통해 리눅스 공간의 커널 부팅 전에 안드로이드 컨테이너의 무결성을 검사하는 단계; 상기 제2 레이어에 설치된 안드로이드 어플리케이션을 구동시키는 단계; 및 상기 제1 레이어에 설치된 모니터링 및 보안 어플리케이션을 통해 상기 제2 레이어에서 구동되는 안드로이드 어플리케이션의 유해 여부를 탐지하는 단계를 포함한다.

[0023] 본 발명의 실시예에서, 상기 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법은, 상기 제1 레이어에 설치된 관리자 프로그램은 지속적으로 동작하며, 상기 제2 레이어에서 동작하는 APK 파일 및 SO 라이브러리를 분석하는 단계를 더 포함할 수 있다.

[0024] 본 발명의 실시예에서, 상기 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법은, 배포, 관리, 보안, 모니터링 및 기타 확장을 위해 사용되는 안드로이드 공간의 툴 및 모바일 장치의 모니터링 및 보안 어플리케이션이 모바일 장치에 다운로드되는 경우, 상기 제1 레이어에 설치하는 단계를 더 포함할 수 있다.

[0025] 본 발명의 실시예에서, 상기 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법은, 안드로이드 어플리케이션이 모바일 장치에 다운로드되는 경우, 상기 제2 레이어에 설치하는 단계를 더 포함할 수 있다.

발명의 효과

[0027] 이와 같은 안드로이드 보안을 위한 듀오 OS 모델에 따르면, 안드로이드 운영체제 레벨 가상화 기술을 사용하여 서로 다른 OS를 2개의 층으로 제공한다. 구체적으로, 모니터링 및 보안 어플리케이션은 리눅스 공간에 설치되고, 안드로이드 어플리케이션은 안드로이드 공간에 분리되어 설치되어, 루트되지 않은 안드로이드 공간을 제공함으로써 안드로이드 공간에서 악성 어플리케이션이 실행되도록 하고, 같은 시간에 리눅스 공간에서 악성 어플리케이션에 의한 상태 변화를 확인 할 수 있다.

[0028] 이는, 가상화 기술보다 속도가 빠르며, 보안 및 모니터링 어플리케이션을 쉽게 개발하는 수 있는 리눅스 공간 관리 기능을 제공한다.

[0029] 또한, 컨테이너 기술을 이용하므로, 안드로이드 뿐만 아니라 리눅스 시스템을 멀티 파티션 생성없이 배포할 수 있으며, 안드로이드 장치 상에서 안정성과 유연성이 뛰어나다.

도면의 간단한 설명

[0031] 도 1은 본 발명에 따른 안드로이드 보안을 위한 듀오 OS 모델의 개념도이다.

도 2는 본 발명에서 이용하는 컨테이너 기술과 가상화 기술을 비교하여 설명하기 위한 도면이다.

도 3은 본 발명의 일 실시예에 따른 MDM(Mobile Device Management, 모바일 단말 관리) 솔루션을 구현한 도면이다.

도 4는 본 발명의 다른 실시예에 따른 Malware Analysis(악성 코드 분석) 솔루션을 구현한 도면이다.

도 5는 본 발명의 또 다른 실시예에 따른 Multiple OS single Platform(단일 플랫폼에 멀티 OS) 솔루션을 구현한 도면이다.

도 6은 본 발명의 일 실시예에 따른 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법의 흐름도이다.

도 7은 도 6의 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법의 상세한 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0032] 후술하는 본 발명에 대한 상세한 설명은, 본 발명이 실시될 수 있는 특정 실시예를 예시로서 도시하는 첨부 도면을 참조한다. 이들 실시예는 당업자가 본 발명을 실시할 수 있기에 충분하도록 상세히 설명된다. 본 발명의 다양한 실시예는 서로 다르지만 상호 배타적일 필요는 없음이 이해되어야 한다. 예를 들어, 여기에 기재되어 있는 특정 형상, 구조 및 특성은 일 실시예에 관련하여 본 발명의 정신 및 범위를 벗어나지 않으면서 다른 실시예로 구현될 수 있다. 또한, 각각의 개시된 실시예 내의 개별 구성요소의 위치 또는 배치는 본 발명의 정신 및 범위를 벗어나지 않으면서 변경될 수 있음이 이해되어야 한다. 따라서, 후술하는 상세한 설명은 한정적인 의미로서 취하려는 것이 아니며, 본 발명의 범위는, 적절하게 설명된다면, 그 청구항들이 주장하는 것과 균등한 모든 범위와 더불어 첨부된 청구항에 의해서만 한정된다. 도면에서 유사한 참조부호는 여러 측면에 걸쳐서 동일하거나 유사한 기능을 지칭한다.

[0033] 이하, 도면들을 참조하여 본 발명의 바람직한 실시예들을 보다 상세하게 설명하기로 한다.

[0034] 도 1은 본 발명에 따른 안드로이드 보안을 위한 듀오 OS 모델의 개념도이다.

[0035] 본 발명에 따른 안드로이드 보안을 위한 듀오 OS 모델(1, Duo OS Model)은 모바일 장치를 위한 아키텍처로서 운영체제 레벨 가상화 기술을 사용하여 하나 이상의 운영체제 상단에 다른 운영체제를 실행시키는 구조이다.

[0036] 본 발명에 따른 듀오 OS 모델(1)은 각 안드로이드 장비에 서로 다른 OS를 두 개의 층으로 제공한다. 제1 레이어(Layer 1)는 모바일 장치 전체의 모니터 및 보안 기능을 제공하는 리눅스 기반의 시스템이다. 제2 레이어(Layer 2)는 컨테이너 기반의 안드로이드로 안드로이드 어플리케이션을 통하여 모바일 장치의 사용자들과 상호작용하는 사용자 공간 층이다.

[0037] 본 발명에 따른 듀오 OS 모델(1)의 장점은 다음과 같다. 첫째로, 어플리케이션 공간(안드로이드 공간)과 모니터링 공간(리눅스 공간)이 구별되어 있다. 둘째로, 리눅스에서 기존의 모니터링 및 보안 도구를 재사용할 수 있다. 셋째로, 컨테이너 기술을 사용하여 유연한 OS 배포를 할 수 있다.

[0038] 상기 모바일 장치는 안드로이드 디바이스일 수 있으며, 무선 통신이 가능한 장치로서, 스마트 폰, 휴대 전화, 태블릿 컴퓨터, 노트북, 넷북, 피디에이(PDA), 피엠피(PMP), 피에스피(PSP), 엠피쓰리(MP3) 플레이어, 이북(e-book) 리더, 내비게이션, 스마트 카메라, 전자사전, 전자시계, 게임기 등 다양한 형태의 모바일(mobile) 장치를 포함할 수 있다.

[0039] 상기 모바일 장치는 이동성을 가지며, 디바이스(device), 기구(apparatus), 단말(terminal), UE(user equipment), MS(mobile station), 무선기기(wireless device), 휴대기기(handheld device) 등 다른 용어로 불릴 수 있다.

[0040] 상기 모바일 장치는 운영체제(Operation System; OS)를 기반으로 다양한 응용 프로그램을 실행할 수 있다. 상기 운영체제는 응용 프로그램이 단말 장치의 하드웨어를 사용할 수 있도록 하기 위한 시스템 프로그램으로서, 본

발명에서는 안드로이드 OS를 기반으로 할 수 있다.

[0041] 상기 응용 프로그램은 모바일 장치를 이용하여 특정한 작업을 수행할 수 있도록 개발된 프로그램으로서, 각종 어플리케이션, 툴, 프로세스 및 서비스 객체뿐 아니라 게임, 동영상, 사진 등의 각종 멀티미디어 컨텐츠(contents) 또는 상기 멀티미디어 컨텐츠를 실행하는 이미지 뷰어, 동영상 재생기 등의 실행 프로그램을 모두 포함할 수 있다.

[0042] 상기 모바일 장치는 무선 통신을 지원하는 디스플레이 장치로서 표시부를 통해 미디어 데이터를 표시하거나 사용자에게 사용자 인터페이스(UI; user interface)를 제공할 수 있다.

[0043] 상기 표시부는 액정 디스플레이(Liquid Crystal Display; LCD) 패널, 플라즈마 디스플레이 패널(Plasma Display Panel; PDP), 유기발광 다이오드(Organic Light-Emitting Diode; OLED) 디스플레이 패널 등을 포함할 수 있다.

[0044] 또한, 사용자 입력을 처리하기 위하여 터치 스크린 기능이 상기 표시부에 포함되거나 별도의 터치 패드 장치로 제공될 수 있다. 이와 다르게, 상기 모바일 장치는 상기 표시부와 별도로 형성되어 사용자의 입력을 받는 키패드 등의 입력부(미도시)를 포함할 수도 있다.

[0045] 도 1을 참조하면, 본 발명에 따른 본 발명에 따른 듀오 OS 모델(1)은 모바일 장치의 하드웨어(10)를 기반으로 하는 안드로이드 커널(30), 보안 플랫폼(20), 리눅스 공간인 제1 레이어(50) 및 안드로이드 공간인 제2 레이어(70)를 포함한다.

[0046] 상기 듀오 OS 모델(1)은 단말의 일부 모듈이거나 별도의 단말일 수 있다. 또한, 상기 보안 플랫폼(20), 상기 안드로이드 커널(30), 상기 제1 레이어(50) 및 상기 제2 레이어(70)의 구성은 통합 모듈로 형성되거나, 하나 이상의 모듈로 이루어 질 수 있다. 그러나, 이와 반대로 각 구성은 별도의 모듈로 이루어질 수도 있다.

[0047] 상기 안드로이드 커널(30)은 안드로이드를 실행시키기 위한 리눅스 기반의 드라이버 패키지로서, 모바일 장치의 하드웨어(10)와 인터페이스를 위한 드라이버를 제공한다.

[0048] 운영체제의 커널(kernel) 영역은, 인터럽트 처리, 프로세스 관리, 메모리 관리, 파일 시스템 관리, 프로그래밍 인터페이스를 제공하는 등 장치의 가장 기본적인 기능들을 관리하고 제어하는 영역이다. 커널 영역은 일반적으로 접근이 불가능한 메모리에 로드(load)되며, 하드웨어를 제어하기 위한 일종의 API(Application Program Interface)라고 볼 수 있다.

[0049] 상기 보안 플랫폼(20)(예를 들어, Trustzone)은 신뢰할 수 있는 실행 환경을 가진 모바일 장치의 하드웨어(10)를 위해, 보안 키를 저장하는데 사용될 수 있다. 상기 보안 키는 리눅스 공간인 제1 레이어(50)의 커널 부팅 전에 무결성 체크를 위해 사용될 수 있다.

[0050] 상기 제1 레이어(50)는 리눅스 공간으로서, 모바일 장치 전체의 모니터 및 보안 기능을 제공하는 리눅스 기반의 시스템이다. 배포, 관리, 보안, 모니터링 및 기타 확장을 위해 사용되는 안드로이드 공간의 툴(51)들이 이 리눅스 공간에 배치된다.

[0051] 상기 제2 레이어(70)는 안드로이드 공간으로서, 컨테이너 기반의 안드로이드로 안드로이드 어플리케이션을 통하여 모바일 사용자들과 상호작용하는 사용자 공간 층이다. 안드로이드 어플리케이션(71, 72, 73)이 이러한 안드로이드 공간에 배치된다.

[0052] 컨테이너 기술은, 운영체제 가상화 기반에서 어플리케이션과 이를 실행하는데 필요한 요소를 묶은 일종의 패키징 기술이다. 여러 개 실행시키거나 다른 운영환경으로 옮겨서 실행할 수도 있어 간편하게 어플리케이션을 운영, 확장할 수 있다.

[0053] 컨테이너 기술이 급부상한 또 다른 이유는 개발과 운영의 틈새를 메우는 이른바 '데브옵스(DevOps)'에 도움이 되기 때문이다. 그동안 개발자가 만든 시스템을 운영 환경으로 넘겨 배포하는 과정에서 문제가 생겨 시스템 가동 시기가 늦어지는 일이 많았다. 이때 컨테이너를 이용하면 패키징한 어플리케이션을 운영 환경에 그대로 이식해 실행할 수 있다.

[0054] 이에 따라, 개발자는 개발에 더 집중하고, 관리자는 손쉽게 배포, 관리할 수 있다. IT 아키텍트는 테스트 기간과 배포 시 오류를 줄이면서도 필요에 따라 유연하게 인프라를 확장할 수 있다.

[0055] 컨테이너 기술은 집적도와 배포 속도, 성능면에서 매우 유리하다. 컨테이너는 VM(Virtual Machine: 가상 머신)

보다 10배 정도 집적도가 높다. 기존에는 서버 1대에 VM을 10대 설치했다면 컨테이너는 100개 이상 운영할 수 있다. 이는 컨테이너 구조에 하이퍼 바이저와 VM운영체제 계층이 없어 물리적인 자원을 덜 사용하기 때문이다.

[0056] 일반적으로 가상화 환경에서는 이들이 전체 자원의 10~20% 정도를 사용하는 것으로 알려졌다. 물리 서버에 운영환경을 구축하려면 보통 27시간 정도 소요된다. VM에서는 템플릿 기능을 이용한다고 해도 12분 정도 걸린다. 반면 컨테이너 인스턴스를 새로 만드는 데는 약 10초에 가능하다.

[0057] 또한, 컨테이너 기술은 가상화 기술보다 새로운 환경을 빠르게 구성할 수 있고, 이러한 시간을 더욱 단축시킬 수 있다. 또한, VM은 환경을 구성한 후 데이터를 올리고 다른 시스템과 연동하는 등의 수작업이 필요하지만, 컨테이너를 쓰면 이런 작업까지 컨테이너 이미지에 넣어 생략할 수 있어 효율적이다.

[0058] 컨테이너 기술은 성능 면에서 많은 장점이 있다. 컨테이너 기술은 하드웨어 위에 호스트 운영체제를 설치하고 그 위에 어플리케이션과 라이브러리를 패키지로 묶은 컨테이너를 올리는 구조로 형성된다.

[0059] 기존의 베어메탈 환경이라면 SSL 라이브러리에 문제가 생겼을 때 당장 이와 연결된 모든 어플리케이션에서 문제가 발생한다. 단일 라이브러리를 공유하는 구조이기 때문이다. 반면, 컨테이너는 내부 라이브러리를 컨테이너별로 사용하므로, 한 컨테이너가 장애를 일으켜도 다른 서비스는 영향을 받지 않는다.

[0060] 컨테이너 기술을 가상화 기술과 비교하는 경우, 컨테이너 기술의 가장 큰 장점은 실시간적인 민첩성과 대응이 필요한 곳에 적용할 수 있는 확장성이다.

[0061] 도 2를 참조하면, 보통 가상화는 자원이 부족하면 해당 VM에 자원을 더 할당하는 '스케일 업(Scale Up)' 방식으로 확장한다. 반면 컨테이너는 같은 역할을 하는 컨테이너를 하나 더 만들어 실행하는 '스케일 아웃(Scale Out)' 방식이다.

[0062] 스케일 아웃 방식은 장애 대응이나 서비스 확장 시 더 유연하게 대응할 수 있다. 컨테이너는 장애가 발생했을 때 기존 컨테이너를 복구하는 대신 새로 하나를 만들어 실행하는 것이 더 빠를 만큼 가볍고 유연하다.

[0063] 한편, Ubuntu(우분투) Touch, Condroid, Cells은 각기 다른 목적으로 안드로이드 컨테이너를 지원하는 솔루션이다.

[0064] Condroid와 Cells은 각기 다른 용도로 사용할 수 있는 하나 이상의 안드로이드 환경을 제공하기 위해 운영체제 수준의 가상화 방법을 이용한다. Ubuntu Touch는 LXC(LinuX Containers)와 libhybris의 도움으로 안드로이드에 Ubuntu-liked 어플리케이션을 제공하는 것을 목적으로 하는 솔루션이다.

[0065] Ubuntu Touch에서 안드로이드 SurfaceFlinger와 일부 서비스는 안드로이드 디스플레이 하드웨어에 Ubuntu 어플리케이션들을 디스플레이하고 실행되게 하려고 리눅스 공간에서 실행된다. 이 Ubuntu Touch와 본 발명의 듀오 OS 모델(1)은 유사해 보이지만 확실한 차이가 존재한다.

[0066] Ubuntu Touch가 SurfaceFlinger를 리눅스 공간에서 실행하기 위해 부분적으로 컨테이너를 제공한다면, 본 발명의 듀오 OS 모델(1)에서는 안드로이드를 위해 전체 LXC 컨테이너를 제공된다.

[0067] Ubuntu Touch가 우분투 시스템에 안드로이드 장치를 구축하는 데 초점을 맞추는 반면, 본 발명의 듀오 OS 모델(1)은 안드로이드 폰에 안정성과 유연성 향상에 초점을 맞추고 있다.

[0068] 본 발명의 듀오 OS 모델(1)은 보안된 안드로이드 공간 구축에 초점을 맞추고 있으므로, 컨테이너 루트 파일 시스템의 무결성 체크를 하는 보안 플랫폼(20)(예를 들어, Trustzone)에 의존하여 무결성 검사를 통과하지 못한 루트 파일 시스템을 거절할 수 있다. 이에 따라, 유해 프로그램의 탐지를 더욱 견고히 수행할 수 있다.

[0069] 본 발명에 따른 듀오 OS 모델(1)은 모바일 장치의 새로운 운영체제의 개념을 제공하며, 아래와 같은 특징을 가진다.

[0070] 먼저, 기존 모바일 장치는 운영체제가 동작하는 하나의 층만을 제공하고, 해당 층에서 모니터링 및 보안 어플리케이션이 모바일 보안 상태(Rooting or unrooting, jailbreak or not jailbreak)에 상관없이 정상 및 악성 어플리케이션과 동일한 장소에서 동작하였다.

[0071] 반면, 본 발명에 따른 듀오 OS 모델(1)에서는 모니터링 및 보안 어플리케이션은 리눅스 공간(50)에 설치되고, 안드로이드 어플리케이션은 안드로이드 공간(70)에 설치되어 서로 분리되어 있다.

[0072] 모바일 운영체제인 안드로이드 경우 리눅스 커널에 발전에도 불구하고 여전히 일반적인 리눅스 운영체제의 어플리케이션과 툴들의 재사용이 가능하지 않다. 그러나, 본 발명에 따른 듀오 OS 모델(1)에서는 리눅스에서 기존의

모니터링 및 보안 도구를 재사용할 수 있다.

[0073] 또한, 요즘에는 하나의 장치에 멀티 플랫폼이 필요한 경우가 있다. 본 발명에 따른 듀오 OS 모델(1)에서는 컨테이너 기술을 사용하므로, 안드로이드 뿐만 리눅스 시스템을 멀티 파티션 생성없이 배포할 수 있다. 이에 따라, 유연한 OS를 배포할 수 있다.

[0074] AVD(안드로이드 가상 장치) 또는 기타 QEMU 기반 에뮬레이터 등의 안드로이드 에뮬레이터는 일반적으로 악성 코드 분석을 위해 사용된다. 그러나, 대부분의 안드로이드 악성 코드는 안드로이드 에뮬레이터를 감지하는 안티 에뮬레이터 기술을 적용하고 있다. 안티 에뮬레이터 기술은 일반적으로 실제 모바일 하드웨어 및 에뮬레이터의 하드웨어 정보를 구분하려고 하고 있다.

[0075] 반면, 본 발명에 따른 듀오 OS 모델(1)에서는 모바일 장치의 실제 하드웨어 정보를 추출하는 컨테이너 기술을 사용하고 있으므로, 본 발명에 따른 듀오 OS 모델(1)과 실제 모바일 장치를 구별할 수 없다.

[0076] 본 발명에 따른 듀오 OS 모델(1)에서는 다음 솔루션을 사용할 수 있다.

[0077] - MDM(모바일 단말 관리)

[0078] - Malicious Analysis(악성코드 분석)

[0079] - Multiple OS on single Platform(단일 플랫폼에 멀티 OS)

[0080] 먼저, MDM (Mobile Device Management)에 대해 설명한다. 기존의 Cells 및 Condroid에 비교하여 본 발명에 따른 듀오 OS 모델(1)은 보안 및 모니터링 어플리케이션을 쉽게 개발하는 수 있는 리눅스 공간 관리 기능을 제공한다.

[0081] 또한, Ubuntu Touch는 사용자 경험에 초점을 맞추는 반면, 본 발명에 따른 듀오 OS 모델(1)은 사용자 응용 프로그램 보안에 초점을 맞춘다. 본 발명에 따른 듀오 OS 모델(1)은 공유 서비스가 존재하지 않는 전체 컨테이너 솔루션을 제공한다.

[0082] Malware Analysis(악성 코드 분석)과 관련하여, 기존 QEMU-based virtualization는 실제 모바일 하드웨어 및 에뮬레이터의 하드웨어 정보를 구분하려고 하고 있다. 반면, 본 발명에 따른 듀오 OS 모델(1)은 실제 디바이스와 구별되지 않는 실제 하드웨어 정보를 제공한다. 또한, 컨테이너 기술은 가상화 기술보다 훨씬 빠르다.

[0083] 실제 모바일(Real Mobile) 장치에서 본 발명에 따른 듀오 OS 모델(1)은 클라우드 통합한 유용한 솔루션과 실제 디바이스와 비교하여 개발하기 쉬운 솔루션을 제공한다.

[0084] 이하에서는 상기 세 가지 솔루션을 모바일 장치에서 구현한 실시예들을 도면을 참조하여 상세히 설명한다.

[0085] 도 3은 본 발명의 일 실시예에 따른 MDM(Mobile Device Management, 모바일 단말 관리) 솔루션을 구현한 도면이다.

[0086] 도 3을 참조하면, 본 발명에 따른 듀오 OS 모델(1)에서 MDM 어플리케이션(40)은 안드로이드 공간(70)의 활동을 통제하기 위해서 리눅스 공간(50)의 상단에 배포될 수 있다. 안드로이드 공간(70)이 LXC(LinuX Containers)에 의해 배치된 이래로, 컨테이너의 모든 정보는 /proc/$container_id/ 또는 LXC-실행 명령어를 통하여 검사할 수 있다.

[0087] 안드로이드 악성 어플리케이션 중 일부는 루트환경에서의 탐지를 피하기 위해서 노력한다. 본 발명은 루트되지 않은 안드로이드 공간(70)을 제공함으로써 안드로이드 공간(70)에서 악성 어플리케이션이 실행되도록 하고 같은 시간에 MDM 어플리케이션(40)을 통해 리눅스 공간(50)에서 탐지를 수행할 수 있다.

[0088] 이에 따라, 본 발명은 안드로이드 공간(70)의 루트 상태를 체크할 수 있고, 악성 응용 어플리케이션의 탐지를 회피할 수 있다. 또한, 리눅스 공간(50)에서 안드로이드 공간(70)에서 일어나는 악의적인 행동을 검사할 수 있고, ARM TrustZone(20)을 통해 안드로이드 컨테이너의 무결성을 검사할 수 있는 장점을 제공한다.

[0089] 도 4는 본 발명의 다른 실시예에 따른 Malware Analysis(악성 코드 분석) 솔루션을 구현한 도면이다.

[0090] 도 4를 참조하면, 본 발명에 따른 듀오 OS 모델(1)에서는 악성 코드 분석을 위한 어플리케이션(60)이 안드로이드 공간(70)에서 동작하는 APK 파일이나 SO 라이브러리를 분석하기 위하여 리눅스 공간(50)의 상단에 배치될 수 있다.

[0091] 이에 따라, 악성 코드 분석을 위한 어플리케이션을 제1 레이어에 설치하고, 안드로이드 공간인 제2 레이어에 설

치된 안드로이드 기반의 악성 어플리케이션을 탐지할 수 있다. 예를 들어, 안드로이드 단말의 OS 상에 존재할 수 있는 악성코드가 보안이 필요한 어플리케이션(예컨대, 악성앱 탐지 어플리케이션)에 접근하는 것을 방지할 수 있다.

[0092] 컨테이너 기술을 사용하여 안드로이드 컨테이너를 발전시켰기 때문에 관리자 프로그램이 리눅스 공간(50)에서 지속적으로 동작하면서 안드로이드 공간(70)에 유해한 시스템 콜과 커널 관련 활동을 막는 것이 훨씬 쉬워졌다. 또한, 컨테이너 기술을 이용하여 컨테이너 백업을 쉽게 할 수 있다.

[0093] 따라서, 본 발명에 따라 안드로이드 공간에 치명적인 공격으로부터 모바일 시스템 복구를 도와줄 수 있다. 안드로이드 컨테이너가 처리 시스템과 실제 하드웨어에 직접적인 접근이 가능해졌기 때문에, QEMU보다 배치가 빠르고 실제 디바이스와 비교하여 구별할 수 없게 되었다.

[0094] 도 5는 본 발명의 또 다른 실시예에 따른 Multiple OS single Platform(단일 플랫폼에 멀티 OS) 솔루션을 구현한 도면이다.

[0095] 도 5를 참조하면, OS 배포 모듈(80)을 리눅스 공간(50)의 상단에 배치할 수 있다. 이에 따라, ARM Trustzone(20)과 OS 배포 모듈(80, 리눅스 공간)을 사용하여, 단일 플랫폼에 무결성 검사와 리눅스 또는 안드로이드 버전의 배포가 더욱 용이해 진다.

[0096] 본 발명에 따라, 안드로이드 운영체제 레벨 가상화 기술을 사용하여 서로 다른 OS를 2개의 층으로 제공한다. 이에 따라, 모니터링 및 보안 어플리케이션은 리눅스 공간에 설치되고, 안드로이드 어플리케이션은 안드로이드 공간에 분리되어 설치되어, 루트되지 않은 안드로이드 공간을 제공함으로써 안드로이드 공간에서 악성 어플리케이션이 실행되도록 하고 같은 시간에 리눅스 공간에서 탐지를 수행할 수 있다.

[0097] 이는, 가상화 기술보다 속도가 빠르며, 보안 및 모니터링 어플리케이션을 쉽게 개발하는 수 있는 리눅스 공간 관리 기능을 제공한다.

[0098] 또한, 컨테이너 기술을 이용하므로, 안드로이드 뿐만 아니라 리눅스 시스템을 멀티 파티션 생성없이 배포할 수 있으며, 안드로이드 장치 상에서 안정성과 유연성이 뛰어나다.

[0100] 도 6은 본 발명의 일 실시예에 따른 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법의 흐름도이다. 도 7은 도 6의 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법의 상세한 흐름도이다.

[0101] 본 실시예에 따른 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법은, 도 1의 안드로이드 보안을 위한 듀오 OS 모델(1, Duo OS Model)과 실질적으로 동일한 구성에서 진행될 수 있다. 따라서, 도 1의 듀오 OS 모델(1)과 동일한 구성요소는 동일한 도면부호를 부여하고, 반복되는 설명은 생략한다.

[0102] 본 실시예에 따른 모바일 장치의 보안 방법은, 각 안드로이드 장비에 서로 다른 OS를 두 개의 층으로 제공되는 듀오 OS 모델(1)을 통해 수행된다. 구체적으로, 모바일 장치의 모니터링 및 보안 기능을 제공하는 리눅스 기반의 운영체제인 제1 레이어(50) 및 안드로이드 어플리케이션을 통하여 모바일 사용자들과 상호작용하는 사용자 공간층으로 컨테이너 기반의 안드로이드 운영체제인 제2 레이어(70)를 포함하는 듀오 OS 모델(1)을 탑재한 모바일 장치에서 수행된다.

[0103] 본 발명에서, 상기 제1 레이어(50)는 리눅스 공간으로서, 모바일 장치 전체의 모니터 및 보안 기능을 제공하는 리눅스 기반의 시스템이다. 배포, 관리, 보안, 모니터링 및 기타 확장을 위해 사용되는 안드로이드 공간의 툴(51)들이 이 리눅스 공간에 배치된다.

[0104] 반면, 상기 제2 레이어(70)는 안드로이드 공간으로서, 컨테이너 기반의 안드로이드로 안드로이드 어플리케이션을 통하여 모바일 사용자들과 상호작용하는 사용자 공간 층이다. 안드로이드 어플리케이션(71, 72, 73)이 이 안드로이드 공간에 배치된다.

[0105] 도 6을 참조하면, 본 실시예에 따른 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법은, 안드로이드 커널에서 보안 키를 저장하는 보안 플랫폼(20)을 통해 리눅스 공간(50)의 커널 부팅 전에 안드로이드 컨테이너의 무결성을 검사한다(단계 S10).

[0106] 신뢰할 수 있는 실행 환경을 가진 모바일 장치의 하드웨어(10)를 위해, 상기 보안 플랫폼(20)(예를 들어, Trustzone)에 보안 키를 저장하고, 상기 보안 키는 리눅스 공간인 제1 레이어(50)의 커널 부팅 전에 무결성 체크를 위해 사용될 수 있다.

[0107] 도 7을 참조하면, 모바일 장치에 프로그램이 다운로드 되는 경우, 다운로드 되는 어플리케이션이 모바일 장치의 모니터링 및 보안 어플리케이션인지 확인한다(단계 S21).

[0108] 이 경우, 모바일 장치의 모니터링 및 보안 어플리케이션에는, 다운로드 되는 프로그램이 배포, 관리, 보안, 모니터링 및 기타 확장을 위해 사용되는 안드로이드 공간의 툴인 경우도 포함된다.

[0109] 다운로드 되는 어플리케이션이 모바일 장치의 모니터링 및 보안 어플리케이션인 경우, 듀오 OS 모델(1)의 상기 제1 레이어(50)에 설치한다(단계 S23).

[0110] 반면, 모바일 장치에 안드로이드 어플리케이션이 다운로드되는 경우(단계 S25), 듀오 OS 모델(1)의 상기 제2 레이어(70)에 설치한다(단계 S27).

[0111] 이에 따라, 상기 제2 레이어(70)에 설치된 안드로이드 어플리케이션을 구동시키며(단계 S30), 동시에 상기 제1 레이어(50)에 설치된 모니터링 및 보안 어플리케이션을 통해 상기 제2 레이어에서 구동되는 안드로이드 어플리케이션의 유해 여부를 탐지할 수 있다(단계 S50).

[0112] 본 발명의 듀오 OS 모델(1)은 보안된 안드로이드 공간 구축에 초점을 맞추고 있으므로, 컨테이너 루트 파일 시스템의 무결성 체크를 하는 보안 플랫폼(20)(예를 들어, Trustzone)에 의존하여 무결성 검사를 통과하지 못한 루트 파일 시스템을 거절할 수 있다.

[0113] 본 발명에 따른 듀오 OS 모델(1)에서는 안드로이드 공간(70)의 활동을 통제하기 위해서 MDM(모바일 단말 관리), Malicious Analysis(악성코드 분석) 및 Multiple OS on single Platform(단일 플랫폼에 멀티 OS)의 솔루션 중 적어도 하나를 사용할 수 있으며, 이는 리눅스 공간(50)의 상단에 배포될 수 있다.

[0114] 먼저, MDM (Mobile Device Management)에 대해 설명한다. 기존의 Cells 및 Condroid에 비교하여 본 발명에 따른 듀오 OS 모델(1)은 보안 및 모니터링 어플리케이션을 쉽게 개발하는 수 있게 리눅스 공간 관리 기능을 제공한다.

[0115] 안드로이드 악성 어플리케이션 중 일부는 루트환경에서의 탐지를 피하기 위해서 노력한다. 본 발명은 루트되지 않은 안드로이드 공간(70)을 제공함으로써 안드로이드 공간(70)에서 악성 어플리케이션이 실행되도록 하고 같은 시간에 MDM 어플리케이션(40)을 통해 리눅스 공간(50)에서 탐지를 수행할 수 있다.

[0116] 이에 따라, 본 발명은 안드로이드 공간(70)의 루트 상태를 체크할 수 있고, 악성 응용 어플리케이션의 탐지를 회피할 수 있다. 또한, 리눅스 공간(50)에서 안드로이드 공간(70)에서 일어나는 악의적인 행동을 검사할 수 있고, ARM TrustZone(20)을 통해 안드로이드 컨테이너의 무결성을 검사할 수 있는 장점을 제공한다.

[0117] 또한, Ubuntu Touch는 사용자 경험에 초점을 맞추는 반면, 본 발명에 따른 듀오 OS 모델(1)은 사용자 응용 프로그램 보안에 초점을 맞춘다. 본 발명에 따른 듀오 OS 모델(1)은 공유 서비스가 존재하지 않는 전체 컨테이너 솔루션을 제공한다.

[0118] Malware Analysis(악성 코드 분석)과 관련하여, 기존 QEMU-based virtualization는 실제 모바일 하드웨어 및 에뮬레이터의 하드웨어 정보를 구분하려고 하고 있다. 반면, 본 발명에 따른 듀오 OS 모델(1)은 실제 디바이스와 구별되지 않는 실제 하드웨어 정보를 제공한다. 또한, 컨테이너 기술은 가상화 기술보다 훨씬 빠르다.

[0119] 본 발명에 따른 듀오 OS 모델(1)에서는 악성 코드 분석을 위한 어플리케이션(60)이 리눅스 공간(50) 상단에 배치되어 지속적으로 동작하며, 안드로이드 공간(70)에서 동작하는 APK 파일이나 SO 라이브러리를 분석할 수 있다(단계 S70).

[0120] 이에 따라, 안드로이드 단말의 OS 상에 존재할 수 있는 악성코드가 보안이 필요한 어플리케이션(예컨대, 악성앱 탐지 어플리케이션)에 접근하는 것을 방지할 수 있다.

[0121] 본 발명은 컨테이너 기술을 사용하여 안드로이드 컨테이너를 발전시켰기 때문에 관리자 프로그램이 리눅스 공간(50)에서 지속적으로 동작하면서 안드로이드 공간(70)에 유해한 시스템 콜과 커널 관련 활동을 막는 것이 훨씬 쉬워졌다. 또한, 컨테이너 기술을 이용하여 컨테이너 백업을 쉽게 할 수 있다.

[0122] 따라서, 본 발명에 따라 안드로이드 공간에 치명적인 공격으로부터 모바일 시스템 복구를 도와줄 수 있다. 안드로이드 컨테이너가 처리 시스템과 실제 하드웨어에 직접적인 접근이 가능해졌기 때문에, QEMU보다 배치가 빠르고 실제 디바이스와 비교하여 구별할 수 없게 되었다.

[0123] 또한, 실제 모바일(Real Mobile) 장치에서 본 발명에 따른 듀오 OS 모델(1)은 클라우드 통합한 유용한 솔루션과

실제 디바이스와 비교하여 개발하기 쉬운 솔루션을 제공한다.

[0124] 또한, Multiple OS single Platform(단일 플랫폼에 멀티 OS 솔루션을 리눅스 공간(50) 상단에 배치하여, ARM Trustzone(20)과 OS 배포 모듈(80, 리눅스 공간)을 사용하여, 단일 플랫폼에 무결성 검사와 리눅스 또는 안드로이드 버전의 배포가 더욱 용이해 진다.

[0125] 본 발명에 따른 듀오 OS 모델(1)은 어플리케이션 공간(안드로이드 공간)과 모니터링 공간(리눅스 공간)이 구별되어 루트되지 않은 안드로이드 공간을 제공함으로써 안드로이드 공간에서 악성 어플리케이션이 실행되도록 하고 같은 시간에 리눅스 공간에서 탐지를 수행할 수 있다.

[0126] 또한, 리눅스에서 기존의 모니터링 및 보안 도구를 재사용할 수 있고, 컨테이너 기술을 사용하여 유연한 OS 배포를 할 수 있다.

[0127] 이와 같은, 듀오 OS 모델을 탑재한 모바일 장치의 보안 방법은 어플리케이션으로 구현되거나 다양한 컴퓨터 구성요소를 통하여 수행될 수 있는 프로그램 명령어의 형태로 구현되어 컴퓨터 판독 가능한 기록 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능한 기록 매체는 프로그램 명령어, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다.

[0128] 상기 컴퓨터 판독 가능한 기록 매체에 기록되는 프로그램 명령어는 본 발명을 위하여 특별히 설계되고 구성된 것들이거니와 컴퓨터 소프트웨어 분야의 당업자에게 공지되어 사용 가능한 것일 수도 있다.

[0129] 컴퓨터 판독 가능한 기록 매체의 예에는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM, DVD와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 ROM, RAM, 플래시 메모리 등과 같은 프로그램 명령어를 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다.

[0130] 프로그램 명령어의 예에는, 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드도 포함된다. 상기 하드웨어 장치는 본 발명에 따른 처리를 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0131] 이상에서는 실시예들을 참조하여 설명하였지만, 해당 기술 분야의 숙련된 당업자는 하기의 특허 청구의 범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위 내에서 본 발명을 다양하게 수정 및 변경시킬 수 있음을 이해할 수 있을 것이다.

산업상 이용가능성

[0133] 본 발명은 안드로이드 운영체제 레벨 가상화 기술을 사용하여 서로 다른 OS를 2개의 층에 제공하여, 안드로이드뿐만 아니라 리눅스 시스템을 멀티 파티션 생성없이 배포할 수 있으며, 안드로이드 장치 상에서 안정성과 유연성이 뛰어나다. 따라서, 안드로이드 악성코드 분석에 유용하게 적용할 수 있을 것으로 기대된다.

부호의 설명

[0135] 1: 안드로이드 보안을 위한 듀오 OS 모델

10: 모바일 장치의 하드웨어

20: 보안 플랫폼

30: 안드로이드 커널

50: 리눅스 공간인 제1 레이어

70: 안드로이드 공간인 제2 레이어

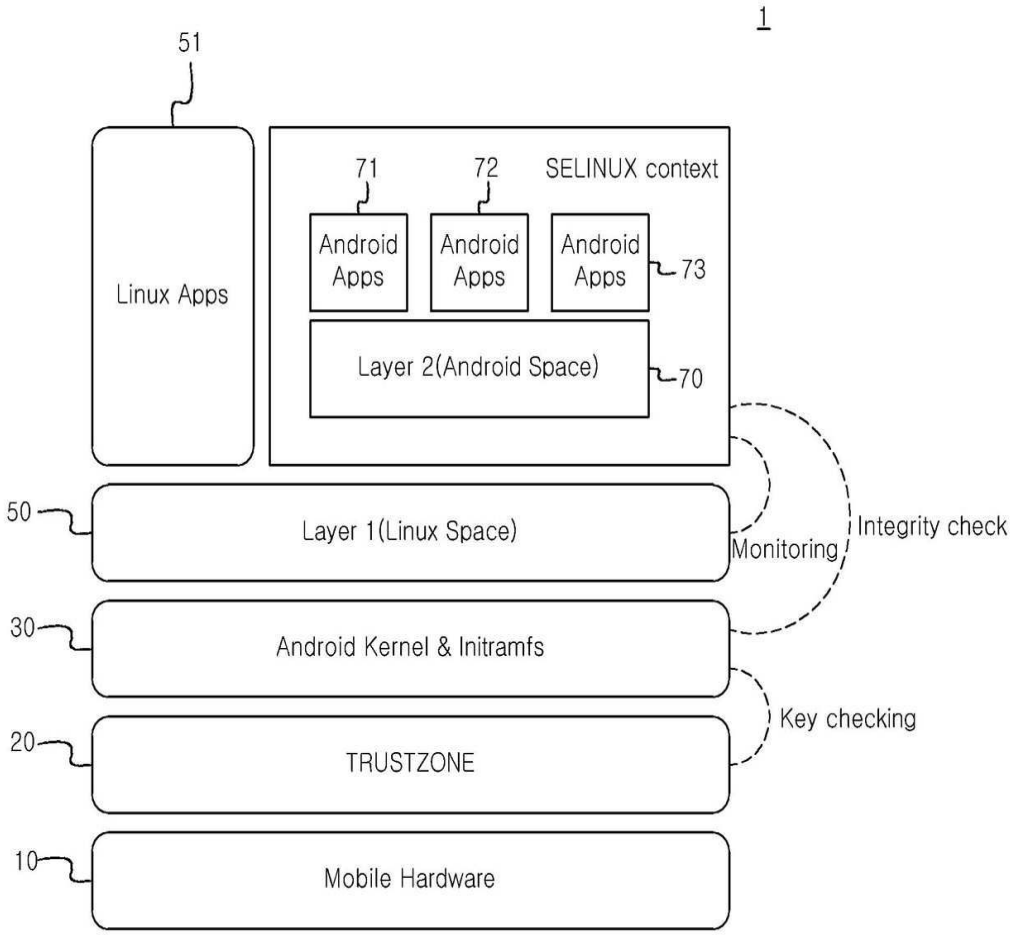
40: MDM 어플리케이션

60: 악성 코드 분석을 위한 어플리케이션

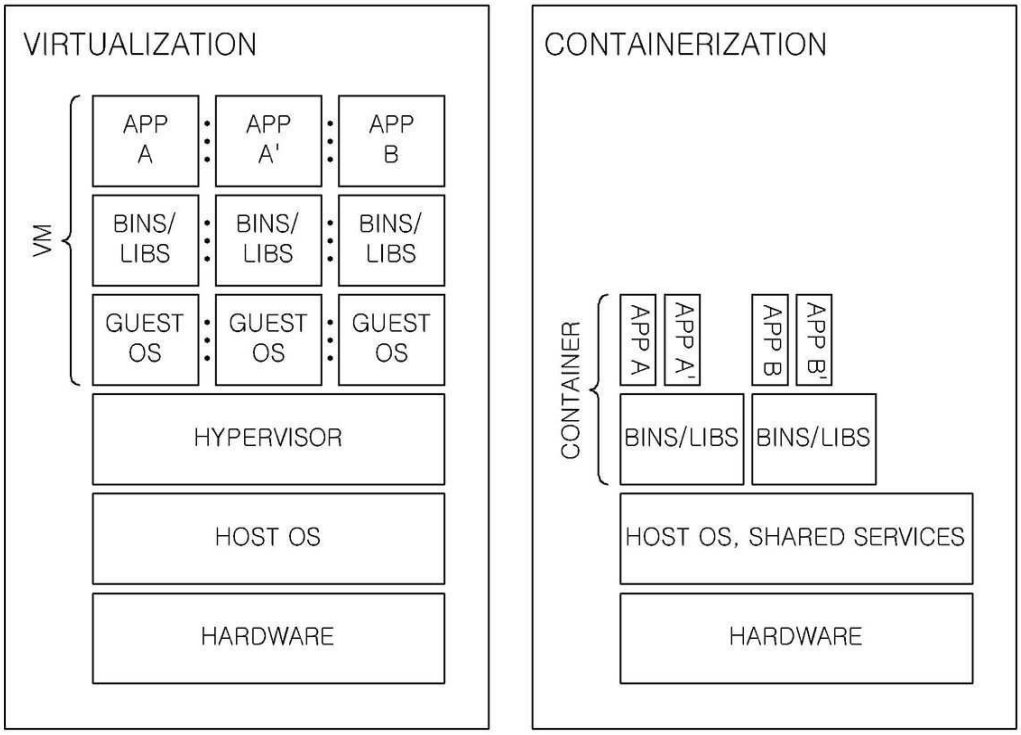
80: OS 배포 모듈

도면

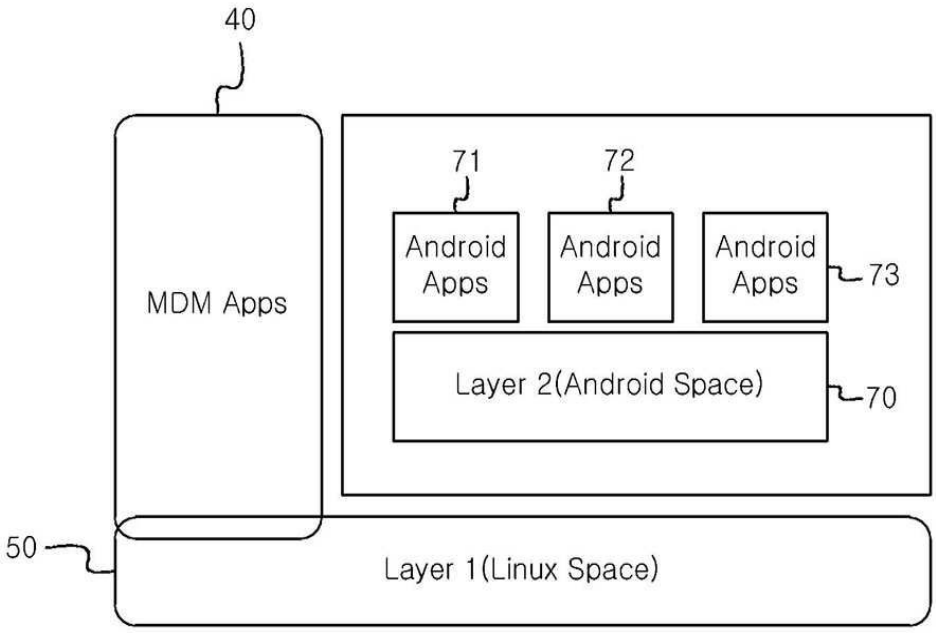
도면1


1
51
Linux Apps
71
72
SELINUX context
Android Apps
Android Apps
Android Apps
73
Layer 2(Android Space)
70
50
Layer 1(Linux Space)
Monitoring
Integrity check
30
Android Kernel & Initramfs
Key checking
20
TRUSTZONE
10
Mobile Hardware

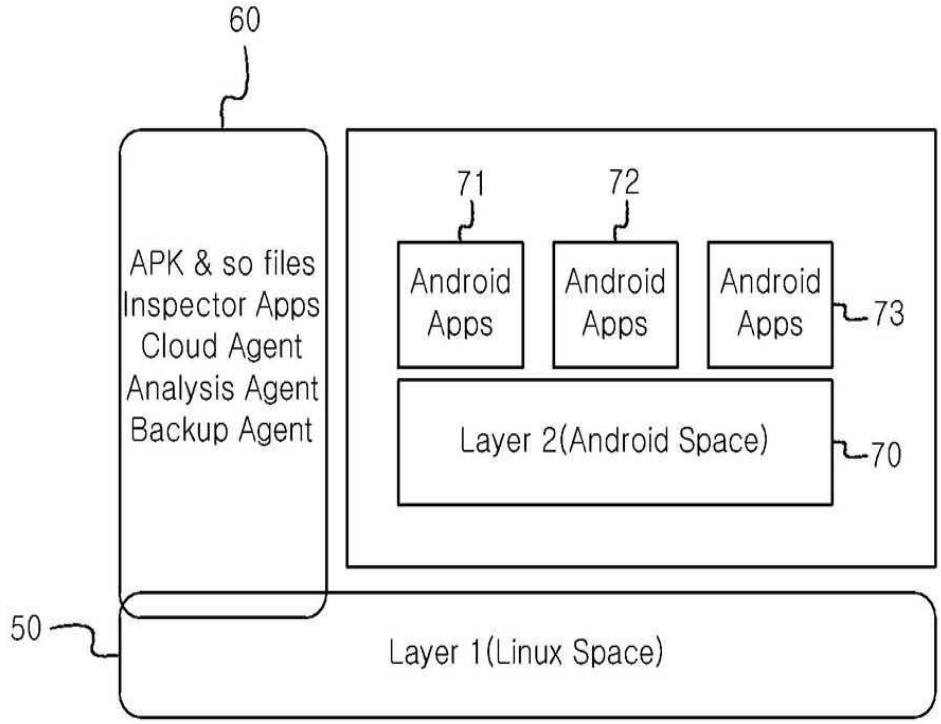
도면2


VIRTUALIZATION
APP A
APP A'
APP B
VM
BINS/ LIBS
BINS/ LIBS
BINS/ LIBS
GUEST OS
GUEST OS
GUEST OS
HYPERVISOR
HOST OS
HARDWARE
CONTAINERIZATION
CONTAINER
APP A
APP A'
APP B
APP B'
BINS/LIBS
BINS/LIBS
HOST OS, SHARED SERVICES
HARDWARE

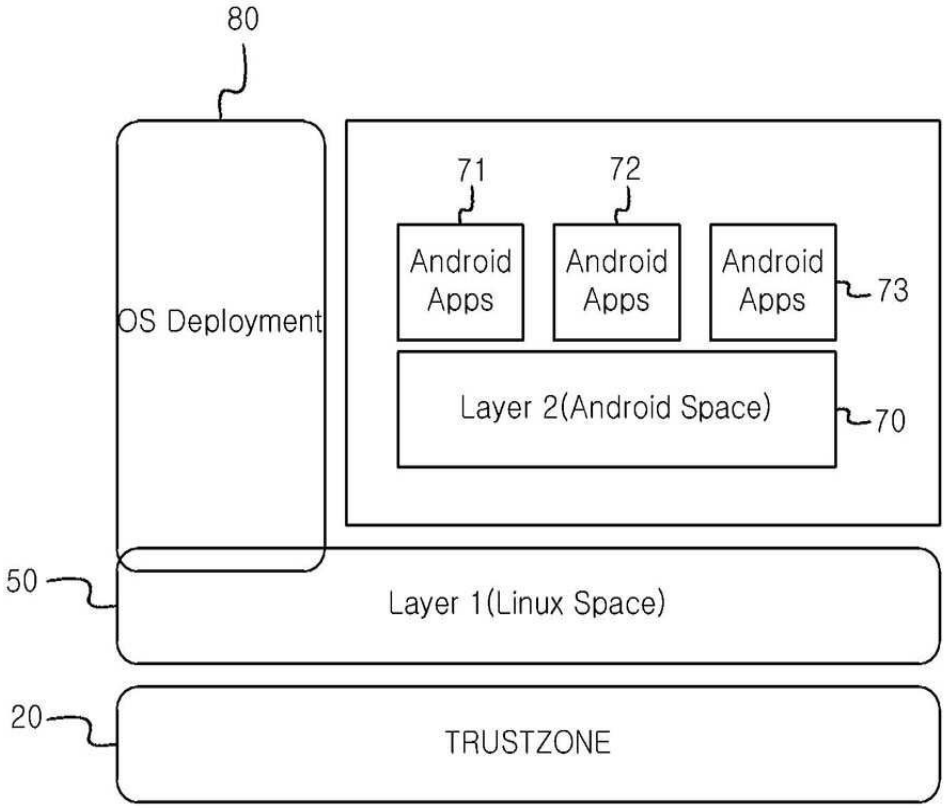
도면3


40
MDM Apps
71
72
Android Apps
Android Apps
Android Apps
73
Layer 2(Android Space)
70
50
Layer 1(Linux Space)

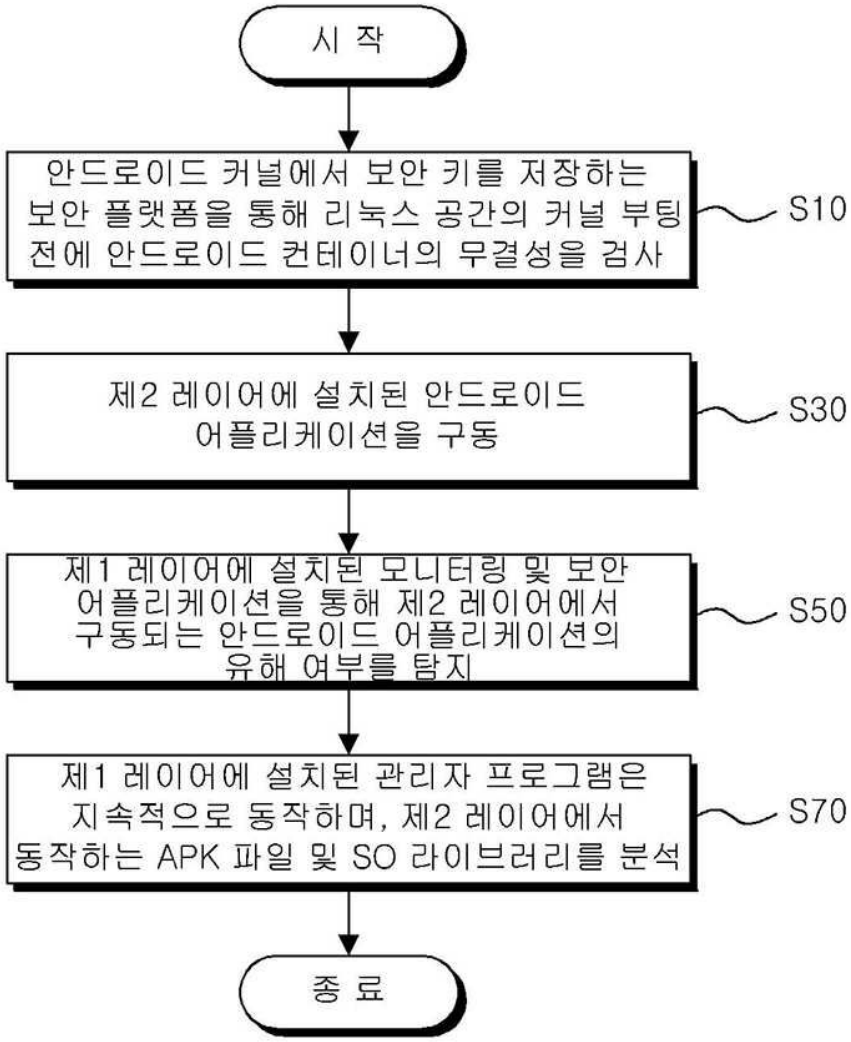
도면4


60
APK & so files
Inspector Apps
Cloud Agent
Analysis Agent
Backup Agent
71
72
Android Apps
Android Apps
Android Apps
73
Layer 2(Android Space)
70
50
Layer 1(Linux Space)

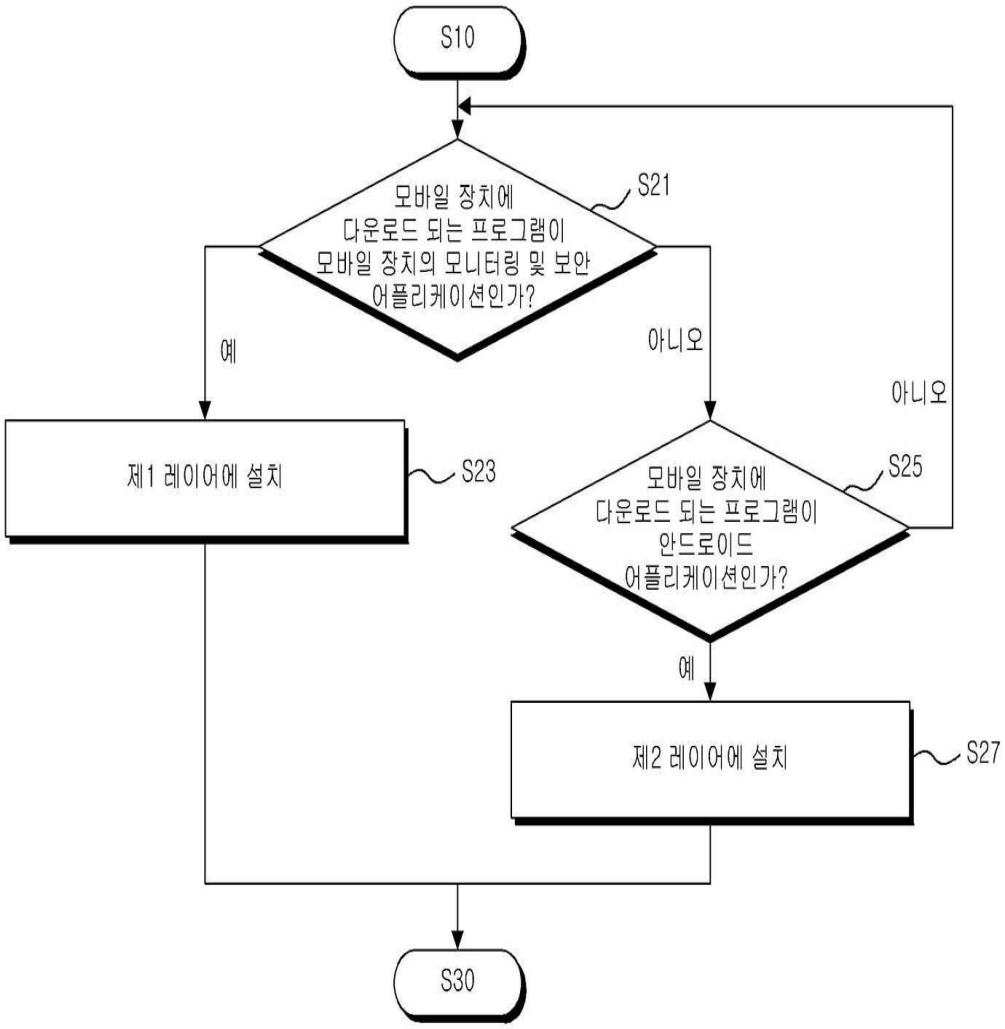
도면5


80
OS Deployment
71
72
Android Apps
Android Apps
Android Apps
73
Layer 2(Android Space)
70
50
Layer 1(Linux Space)
20
TRUSTZONE

도면6


시 작
안드로이드 커널에서 보안 키를 저장하는 보안 플랫폼을 통해 리눅스 공간의 커널 부팅 전에 안드로이드 컨테이너의 무결성을 검사
S10
제2 레이어에 설치된 안드로이드 어플리케이션을 구동
S30
제1 레이어에 설치된 모니터링 및 보안 어플리케이션을 통해 제2 레이어에서 구동되는 안드로이드 어플리케이션의 유해 여부를 탐지
S50
제1 레이어에 설치된 관리자 프로그램은 지속적으로 동작하며, 제2 레이어에서 동작하는 APK 파일 및 SO 라이브러리를 분석
S70
종 료

도면7



【심사관 직권보정사항】

【직권보정 1】

【보정항목】 청구범위

【보정세부항목】 청구항 1 내지 6

【변경전】

매체

【변경후】

비일시적 컴퓨터 판독가능 저장매체

(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2018년05월11일
(11) 등록번호 10-1857009
(24) 등록일자 2018년05월04일

(51) 국제특허분류(Int. Cl.)
G06F 21/56 (2013.01) *H04L 29/06* (2006.01)
(52) CPC특허분류
G06F 21/567 (2013.01)
H04L 63/1408 (2013.01)
(21) 출원번호 10-2017-0008996
(22) 출원일자 2017년01월19일
심사청구일자 2017년01월19일
(56) 선행기술조사문헌
KR1020140037442 A
KR1020120046807 A
US20100122343 A1
KR1020110084693 A

(73) 특허권자
숭실대학교산학협력단
서울특별시 동작구 상도로 369 (상도동)
(72) 발명자
정수환
서울특별시 동작구 상도로 369, 숭실대학교 형남공학관 1105호
차오녹투
서울특별시 동작구 상도로 369, 숭실대학교 형남공학관 1102호
박정수
서울특별시 동작구 상도로 369, 숭실대학교 형남공학관 1102호
(74) 대리인
윤귀상

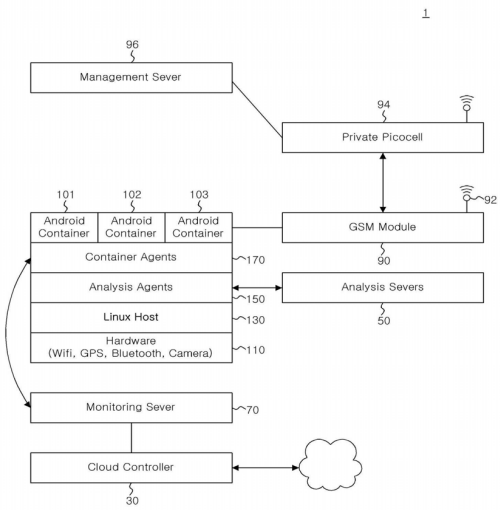
전체 청구항 수 : 총 8 항 심사관 : 윤혜숙

(54) 발명의 명칭 **안드로이드 악성코드 분석을 위한 컨테이너 플랫폼 및 이를 이용한 모바일 장치의 보안 방법**

(57) 요 약

안드로이드 악성코드 분석을 위한 컨테이너 플랫폼에 있어서, 각 모바일 장치에 설치된 안드로이드 컴퓨트 노드는, 클라우드 컨트롤러로부터 수신한 요청에 응답하여, 동적 분석을 위하여 안드로이드 악성코드를 직접 수행하는 적어도 하나의 안드로이드 컨테이너를 생성하고, 상태를 확인하는 컨테이너 에이전트; 상기 모바일 장치의 하드웨어 기반 기능을 제공하는 하드웨어 모듈; 상기 하드웨어 모듈 상에 운영체제가 설치된 리눅스 호스트; 및 상기 리눅스 호스트를 통해 상기 안드로이드 컨테이너 내부에서 악성코드 동작 시 문제를 식별하고, 커널 관련 악성코드 활동을 탐지하여 분석 서버로 전송하는 분석 에이전트를 포함한다. 이에 따라, 컨테이너 기술을 이용하므로, 클라우드 환경에 적용하여 확장성과 탄력성을 높일 수 있다.

대 표 도 - 도2



(52) CPC특허분류
H04L 63/20 (2013.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 B0717-16-0108/1711037875
부처명 미래창조과학부
연구관리전문기관 정보통신기술진흥센터
연구사업명 정보통신·방송 연구개발 사업
연구과제명 맞춤형 보안서비스 제공을 위한 클라두드 기반 지능형 보안 기술 개발
기 여 율 1/2
주관기관 한국전자통신연구원
연구기간 2016.04.01 ~ 2016.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 H8501-16-1008/1711035246
부처명 미래창조과학부
연구관리전문기관 정보통신기술진흥센터
연구사업명 대학 ICT연구센터 육성 지원사업
연구과제명 클라우드 환경의 스마트 기기와 서비스 보안 기술 개발 및 연구 인력양성
기 여 율 1/2
주관기관 숭실대학교 산학협력단
연구기간 2016.01.01 ~ 2016.12.31

명 세 서

청구범위

청구항 1

안드로이드 악성코드 분석을 위한 컨테이너 플랫폼에 있어서, 상기 컨테이너 플랫폼은 적어도 하나의 모바일 장치를 포함하며, 각 모바일 장치에 설치된 안드로이드 컴퓨트 노드는,

클라우드 컨트롤러로부터 수신한 요청에 응답하여, 동적 분석을 위하여 안드로이드 악성코드를 직접 수행하는 적어도 하나의 안드로이드 컨테이너를 생성하고, 상태를 확인하는 컨테이너 에이전트;

상기 모바일 장치의 하드웨어 기반 기능을 제공하는 하드웨어 모듈;

상기 하드웨어 모듈 상에 운영체제가 설치된 리눅스 호스트; 및

상기 리눅스 호스트를 통해 상기 안드로이드 컨테이너 내부에서 악성코드 동작 시 문제를 식별하고, 커널 관련 악성코드 활동을 탐지하여 분석 서버로 전송하는 분석 에이전트를 포함하는, 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼.

청구항 2

제1항에 있어서,

상기 컨테이너 에이전트를 통해 상기 안드로이드 컨테이너를 관리 및 구성하는 상기 클라우드 컨트롤러를 더 포함하는, 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼.

청구항 3

제1항에 있어서,

상기 컨테이너 에이전트와 연결되어 상기 안드로이드 컨테이너의 상태 정보에 대하여 모니터링하여, 상기 안드로이드 컴퓨트 노드의 분석 충돌을 해결하는 모니터링 서버를 더 포함하는, 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼.

청구항 4

제1항에 있어서,

상기 분석 에이전트가 보낸 정보를 수신하여 악성코드 분석을 위한 동적 분석 및 정적 분석이 수행되며, 상기 안드로이드 컨테이너에 APK 설치 명령을 지시하는 분석 서버를 더 포함하는, 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼.

청구항 5

제1항에 있어서,

모바일 네트워크 기능을 상기 안드로이드 컨테이너에 제공하는 사설 피코셀(Private Picocell) 네트워크를 더 포함하는, 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼.

청구항 6

[청구항 6은(는) 설정등록료 납부시 포기되었습니다.]

제5항에 있어서, 상기 사설 피코셀 네트워크는,

SDR(Software-Defined Radio)을 기반으로 하는 RF 송수신기;

GSM/LTE 모듈로 연결 가능한 개인 모바일 네트워크로 작동하는 피코셀 플랫폼; 및

상기 피코셀 플랫폼으로부터 GSM/LTE 정보를 수집하여 악의적인 행동을 탐지하거나 상기 개인 모바일 네트워크 문제를 감시하는 관리 서버를 포함하는, 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼.

청구항 7

삭제

청구항 8

안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안 방법은,

상기 모바일 장치의 컨테이너 에이전트를 통해 클라우드 컨트롤러로부터 수신한 요청에 응답하여, 동적 분석을 위하여 안드로이드 악성코드를 직접 수행하는 적어도 하나의 안드로이드 컨테이너를 생성하는 단계;

상기 모바일 장치의 리눅스 호스트를 통해 상기 안드로이드 컨테이너 내부에서 악성코드 동작 시 문제를 식별하는 단계;

상기 모바일 장치의 분석 에이전트를 통해 커널 관련 악성코드 활동을 탐지하여 분석 서버로 전송하는 단계; 및

상기 모바일 장치에서 상기 분석 서버로부터 분석 정보를 수집하는 단계를 포함하는, 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안 방법.

청구항 9

제8항에 있어서,

상기 모바일 장치에서 상기 안드로이드 컨테이너의 상태 정보를 모니터링 서버에 전송하는 단계를 더 포함하는, 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안 방법.

청구항 10

제8항에 있어서,

상기 모바일 장치에서 상기 분석 서버의 명령에 따라 상기 안드로이드 컨테이너에 APK 파일을 설치하는 단계를 더 포함하는, 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼 및 이를 탑재한 모바일 장치, 이를 이용한 모바일 장치의 보안 방법에 관한 것으로서, 더욱 상세하게는 리눅스 컨테이너 기법을 사용하여 모바일 분석을 위한 새로운 디자인에 관한 것이다.

배 경 기 술

[0002] 가상화 플랫폼인 AVD, QEMU 기반 에뮬레이터에서 실행되는 안드로이드(Android) X86과 같은 안드로이드 악성코드 분석 플랫폼은 안티 에뮬레이터 기술을 이용하여 많은 우회기법이 등장하였다.

[0003] 안티 에뮬레이터 기술은 실제 모바일 하드웨어와 에뮬레이터 된 하드웨어 정보를 구별할 뿐 아니라, 8가지 센서

기능을 제공하는지 여부를 확인하여 구별한다. 이러한 안티 에뮬레이터를 방지하고자 실제 모바일 기기를 사용하는 것은 많은 양의 안드로이드 앱을 분석하는데 비용이 많이 발생할 뿐 아니라 비효율적이다.

[0004] 대표적으로, Bare QEMU-based, Real Device with automation tools, Intel-based Android and Virtualbox/VMware이 사용되고 있다.

[0005] Bare QEMU의 대표적인 예는 AVD, 블루 스택, AMIDuOS, Nox 등이 있다. QEMU는 안드로이드 가상 장치의 하드웨어를 가상화하는데 사용할 수 있는 에뮬레이터 및 가상화 도구로, QEMU를 통해 에뮬레이터는 가상화 환경을 구성할 수 있다.

[0006] Real Device with automation tools는 모든 응용 프로그램이 호환성 문제 없이 실행가능하기 때문에 안드로이드 어플리케이션 분석을 실행하는 효과적인 방법이다. 악성코드의 지능화로 에뮬레이터 환경을 우회하는 기술이 발달하기 때문에 가장 안전하게 분석할 수 있다. 하지만 효율성이 떨어지는 단점이 있다.

[0007] 인텔기반 안드로이드 버전은 모바일 장치뿐 아니라 노트북, 개인용 컴퓨터에서도 실행 가능하다. RemixOS와 같은 Android-x86의 유형이 있다. 인텔기반 안드로이드는 ISO 파일로 제공되기 때문에, Virtualbox 혹은 VMware와 같은 가상화 도구에서 실행 가능하다.

[0008] 그러나, 기존 안드로이드 악성코드 분석의 문제점은 아래와 같다.

[0009] 지능화 된 악성코드는 가상 환경 정보를 확인하는 기본적인 방법부터 빌드 환경, 장비 및 하드웨어 정보를 검사하는 등 다양한 방법으로 에뮬레이터 환경을 검사한다(예 : / dev / qemu_pipe, / dev / socket / qemud가 있는지 확인하거나 getprop 명령을 사용하여 qemu 정보를 확인). 일부 앱은 루팅 된 환경에서 작동하지 않도록 되어 있기 때문에, QEMU-based에서는 분석이 어렵다.

[0010] 또한, 실제 안드로이드 기기(Real Android Device)에서의 앱 분석이 더 나은 결과를 가져 오지만, 사전 분석 및 사후 분석 프로세스를 자동화하는 것이 어려운 단점이 있다. 또한, 클라우드 환경에 적용하여 확장성과 탄력성을 높이는 것도 어렵기 때문에 분석환경에 적합하지 않은 문제점이 있다.

선행기술문헌

특허문헌

[0011] (특허문헌 0001) KR 2015-0099440 A

(특허문헌 0002) KR 1386605 B1

(특허문헌 0003) KR 1626067 B1

비특허문헌

[0012] (비특허문헌 0001) 김호중, 컨테이너에 대한 6가지 오해, IDG Summary, 2016.04.28.

발명의 내용

해결하려는 과제

[0013] 이에, 본 발명의 기술적 과제는 이러한 점에서 착안된 것으로 본 발명의 목적은 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 제공하는 것이다.

[0014] 본 발명의 다른 목적은 상기 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 탑재한 모바일 장치를 제공하는 것이다.

[0015] 본 발명의 또 다른 목적은 상기 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안 방법을 수행하기 위한 장치를 제공하는 것이다.

과제의 해결 수단

[0016] 상기한 본 발명의 목적을 실현하기 위한 일 실시예에 따른 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼에 있어서, 각 모바일 장치에 설치된 안드로이드 컴퓨트 노드는, 클라우드 컨트롤러로부터 수신한 요청에 응답하여, 동적 분석을 위하여 안드로이드 악성코드를 직접 수행하는 적어도 하나의 안드로이드 컨테이너를 생성하고, 상태를 확인하는 컨테이너 에이전트; 상기 모바일 장치의 하드웨어 기반 기능을 제공하는 하드웨어 모듈; 상기 하드웨어 모듈 상에 운영체제가 설치된 리눅스 호스트; 및 상기 리눅스 호스트를 통해 상기 안드로이드 컨테이너 내부에서 악성코드 동작 시 문제를 식별하고, 커널 관련 악성코드 활동을 탐지하여 분석 서버로 전송하는 분석 에이전트를 포함한다.

[0017] 본 발명의 실시예에서, 상기 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼은, 상기 컨테이너 에이전트를 통해 상기 안드로이드 컨테이너를 관리 및 구성하는 상기 클라우드 컨트롤러를 더 포함할 수 있다.

[0018] 본 발명의 실시예에서, 상기 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼은, 상기 컨테이너 에이전트와 연결되어 상기 안드로이드 컨테이너의 상태 정보에 대하여 모니터링하여, 상기 안드로이드 컴퓨트 노드의 분석 충돌을 해결하는 모니터링 서버를 더 포함할 수 있다.

[0019] 본 발명의 실시예에서, 상기 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼은, 상기 분석 에이전트가 보낸 정보를 수신하여 악성코드 분석을 위한 동적 분석 및 정적 분석이 수행되며, 상기 안드로이드 컨테이너에 APK 설치 명령을 지시하는 분석 서버를 더 포함할 수 있다.

[0020] 본 발명의 실시예에서, 상기 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼은, 모바일 네트워크 기능을 상기 안드로이드 컨테이너에 제공하는 사설 피코셀(Private Picocell) 네트워크를 더 포함할 수 있다.

[0021] 본 발명의 실시예에서, 상기 사설 피코셀 네트워크는, SDR(Software-Defined Radio)을 기반으로 하는 RF 송수신기; GSM/LTE 모듈로 연결 가능한 개인 모바일 네트워크로 작동하는 피코셀 플랫폼; 및 상기 피코셀 플랫폼으로부터 GSM/LTE 정보를 수집하여 악의적인 행동을 탐지하거나 상기 개인 모바일 네트워크 문제를 감시하는 관리 서버를 포함할 수 있다.

[0022] 상기한 본 발명의 다른 목적을 실현하기 위한 일 실시예에 따른 모바일 장치는, 상기 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 탑재하고 있다.

[0023] 상기한 본 발명의 또 다른 목적을 실현하기 위한 일 실시예에 따른 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안 방법은, 컨테이너 에이전트를 통해 클라우드 컨트롤러로부터 수신한 요청에 응답하여, 동적 분석을 위하여 안드로이드 악성코드를 직접 수행하는 적어도 하나의 안드로이드 컨테이너를 생성하는 단계; 리눅스 호스트를 통해 상기 안드로이드 컨테이너 내부에서 악성코드 동작 시 문제를 식별하는 단계; 분석 에이전트를 통해 커널 관련 악성코드 활동을 탐지하여 분석 서버로 전송하는 단계; 및 상기 분석 서버로부터 분석 정보를 수집하는 단계를 포함한다.

[0024] 본 발명의 실시예에서, 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안 방법은, 상기 안드로이드 컨테이너의 상태 정보를 모니터링 서버에 전송하는 단계를 더 포함할 수 있다.

[0025] 본 발명의 실시예에서, 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안 방법은, 상기 분석 서버의 명령에 따라 상기 안드로이드 컨테이너에 APK 파일을 설치하는 단계를 더 포함할 수 있다.

발명의 효과

[0026] 이와 같은 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼에 따르면, 컨테이너 기술을 이용하므로, 기존의 QEMU 기술과 같이 하드웨어 가상화 기술이 필요 없으며, 실제 폰에서 동작하는 것과 같이 다양한 장비를 요구하지 않아 확장성과 탄력성을 제공할 수 있게 된다.

[0027] 또한, 본 발명에 따라 안드로이드 컨테이너에서 악성코드를 분석하게 될 경우, 안드로이드 공간에서 실행되는 APK 및 .so 라이브러리에 대한 분석이 가능해지며, 컨테이너를 쉽게 백업할 수 있어 다양한 공격에서부터 간단히 복구가 가능하다.

[0028] 또한, 본 발명은 실제 하드웨어에서 정보를 직접 받아오기 때문에, 가상환경에 비하여 빠르며, 에뮬레이터 우회 기술에 대비할 수 있다. 나아가, 본 발명은 피코셀(Picocell) 네트워크를 사용하기 때문에 실제 모바일 네트워크를 통한 공격에 대응 가능하며, 컨테이너 간 사설 네트워크를 생성하는 것 또한 가능하다.

도면의 간단한 설명

[0029] 도 1은 본 발명의 일 실시예에 따른 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼의 개념도이다.

도 2는 도 1의 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 자세히 도시한 블록도이다.

도 3은 도 2의 안드로이드 컴퓨트 노드의 블록도이다.

도 4는 본 발명에서 이용하는 컨테이너 기술과 가상화 기술을 비교하여 설명하기 위한 도면이다.

도 5는 본 발명의 일 실시예에 따른 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안 방법의 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0030] 후술하는 본 발명에 대한 상세한 설명은, 본 발명이 실시될 수 있는 특정 실시예를 예시로서 도시하는 첨부 도면을 참조한다. 이들 실시예는 당업자가 본 발명을 실시할 수 있기에 충분하도록 상세히 설명된다. 본 발명의 다양한 실시예는 서로 다르지만 상호 배타적일 필요는 없음이 이해되어야 한다. 예를 들어, 여기에 기재되어 있는 특정 형상, 구조 및 특성은 일 실시예에 관련하여 본 발명의 정신 및 범위를 벗어나지 않으면서 다른 실시예로 구현될 수 있다. 또한, 각각의 개시된 실시예 내의 개별 구성요소의 위치 또는 배치는 본 발명의 정신 및 범위를 벗어나지 않으면서 변경될 수 있음이 이해되어야 한다. 따라서, 후술하는 상세한 설명은 한정적인 의미로서 취하려는 것이 아니며, 본 발명의 범위는, 적절하게 설명된다면, 그 청구항들이 주장하는 것과 균등한 모든 범위와 더불어 첨부된 청구항에 의해서만 한정된다. 도면에서 유사한 참조부호는 여러 측면에 걸쳐서 동일하거나 유사한 기능을 지칭한다.

[0031] 이하, 도면들을 참조하여 본 발명의 바람직한 실시예들을 보다 상세하게 설명하기로 한다.

[0032] 도 1은 본 발명의 일 실시예에 따른 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼의 개념도이다. 도 2는 도 1의 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 자세히 도시한 블록도이다. 도 3은 도 2의 안드로이드 컴퓨트 노드의 블록도이다.

[0033] 본 발명에 따른 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼(1, 이하 컨테이너 플랫폼)은 컨테이너 기법에 기초하여 안드로이드 악성코드 분석을 위한 플랫폼이다.

[0034] 구체적으로, 본 발명은 가상화 방식보다 빠르고 모바일 기기 방식보다 유연한 플랫폼으로, 모바일 악성코드 분석을 위한 새로운 분석 및 모니터링 플랫폼을 제안한다.

[0035] 본 발명은 QEMU 기반 에뮬레이터, Android-x86 및 실제 모바일 장치와 같은 기존 솔루션 대신 안드로이드 어플리케이션을 분석하기 위한 안드로이드 컨테이너를 사용한다.

[0036] 도 1을 참조하면, 본 발명에 따른 컨테이너 플랫폼(1)은 안드로이드 컴퓨트 노드들(10), 클라우드 컨트롤러(30), 분석 서버(50), 모니터링 서버(70) 및 사설 피코셀(Private Picocell) 네트워크(도 2 참조)를 포함한다.

[0037] 상기 안드로이드 컴퓨트 노드들(10)은 안드로이드 컨테이너를 생성하고, 안드로이드 컨테이너와 컨테이너 호스트에서 분석 정보를 수집하는 역할을 수행한다. 상기 안드로이드 컴퓨트 노드들(10)은 생성된 모든 안드로이드 컨테이너를 관리한다.

[0038] 상기 클라우드 컨트롤러(30)는 상기 안드로이드 컴퓨트 노드들(10)을 연결하고, 안드로이드 컨테이너를 관리하게 된다.

[0039] 상기 분석 서버(50)는 안드로이드 악성코드 분석을 위한 동적 분석 또는/및 정적 분석이 수행되며, 상기 안드로이드 컨테이너에 APK 설치 명령을 지시한다.

[0040] 상기 모니터링 서버(70)는 상기 안드로이드 컴퓨트 노드들(10)에서 실행 중인 모니터링 에이전트가 보낸 정보를 수집하여 각 안드로이드 컨테이너의 상태 정보에 대하여 모니터링하고, 상기 안드로이드 컴퓨트 노드들(10)의 분석 충돌 및 기타 문제를 해결한다.

[0041] 상기 사설 피코셀(Private Picocell) 네트워크에서는 모바일 네트워크 기능을 안드로이드 컨테이너에 제공한다.

[0043] 도 2를 참조하여, 도 1의 컨테이너 플랫폼(1)의 구성 및 구성 간의 관계를 좀 더 자세하게 설명한다. 도 2 및 도 3에서는 하나의 상기 안드로이드 컴퓨트 노드(10)를 대표적으로 설명한다.

[0044] 도 2 및 도 3을 참조하면, 상기 안드로이드 컴퓨트 노드(10)는 컨테이너 에이전트(170), 하드웨어 모듈(110), 리눅스 호스트(130) 및 분석 에이전트(150)를 포함한다.

[0045] 상기 안드로이드 컴퓨트 노드(10)는 각 모바일 장치의 단말기이거나 일부 모듈일 수 있다. 또한, 상기 컨테이너 에이전트(170), 상기 하드웨어 모듈(110), 상기 리눅스 호스트(130) 및 상기 분석 에이전트(150)의 구성은 통합 모듈로 형성되거나, 하나 이상의 모듈로 이루어 질 수 있다. 그러나, 이와 반대로 각 구성은 별도의 모듈로 이루어질 수도 있다.

[0046] 상기 모바일 장치는 안드로이드 디바이스일 수 있으며, 무선 통신이 가능한 장치로서, 스마트 폰, 휴대 전화, 태블릿 컴퓨터, 노트북, 넷북, 피디에이(PDA), 피엠피(PMP), 피에스피(PSP), 엠피쓰리(MP3) 플레이어, 이북(e-book) 리더, 내비게이션, 스마트 카메라, 전자사전, 전자시계, 게임기 등 다양한 형태의 모바일(mobile) 장치를 포함할 수 있다.

[0047] 상기 모바일 장치는 이동성을 가지며, 디바이스(device), 기구(apparatus), 단말(terminal), UE(user equipment), MS(mobile station), 무선기기(wireless device), 휴대기기(handheld device) 등 다른 용어로 불릴 수 있다.

[0048] 상기 모바일 장치는 운영체제(Operation System; OS)를 기반으로 다양한 응용 프로그램을 실행할 수 있다. 상기 운영체제는 응용 프로그램이 단말 장치의 하드웨어를 사용할 수 있도록 하기 위한 시스템 프로그램으로서, 본 발명에서는 안드로이드 OS를 기반으로 할 수 있다.

[0049] 상기 응용 프로그램은 모바일 장치를 이용하여 특정한 작업을 수행할 수 있도록 개발된 프로그램으로서, 각종 어플리케이션, 툴, 프로세스 및 서비스 객체뿐 아니라 게임, 동영상, 사진 등의 각종 멀티미디어 컨텐츠(contents) 또는 상기 멀티미디어 컨텐츠를 실행하는 이미지 뷰어, 동영상 재생기 등의 실행 프로그램을 모두 포함할 수 있다.

[0050] 상기 모바일 장치는 무선 통신을 지원하는 디스플레이 장치로서 표시부를 통해 미디어 데이터를 표시하거나 사용자에게 사용자 인터페이스(UI; user interface)를 제공할 수 있다.

[0051] 상기 표시부는 액정 디스플레이(Liquid Crystal Display; LCD) 패널, 플라즈마 디스플레이 패널(Plasma Display Panel; PDP), 유기발광 다이오드(Organic Light-Emitting Diode; OLED) 디스플레이 패널 등을 포함할 수 있다.

[0052] 또한, 사용자 입력을 처리하기 위하여 터치 스크린 기능이 상기 표시부에 포함되거나 별도의 터치 패드 장치로 제공될 수 있다. 이와 다르게, 상기 모바일 장치는 상기 표시부와 별도로 형성되어 사용자의 입력을 받는 키패드 등의 입력부(미도시)를 포함할 수도 있다.

[0053] 상기 컨테이너 에이전트(170)는 상기 클라우드 컨트롤러(30)로부터 수신한 요청에 응답하여, 동적 분석을 위하여 안드로이드 악성코드를 직접 수행하는 적어도 하나의 안드로이드 컨테이너(101, 102, 103)를 생성한다.

[0054] 또한, 상기 컨테이너 에이전트(170)는 상기 클라우드 컨트롤러(30)와 안드로이드 컨테이너(101, 102, 103) 간 연결을 통하여 안드로이드 컨테이너(101, 102, 103)의 상태를 확인할 수 있도록 한다.

[0055] 상기 안드로이드 컨테이너(101, 102, 103)는 복수 개일 수 있고, 동적 분석을 위하여 안드로이드 악성코드를 직접 수행할 수 있도록 설계된다. 안드로이드 컨테이너(101, 102, 103)의 상태는 OS, 빌드 버전, 루틴, 루팅 상태, 하드웨어 정보 등에 따라 각각 다르게 생성 가능하다.

[0056] 컨테이너 기술은, 운영체제 가상화 기반에서 어플리케이션과 이를 실행하는데 필요한 요소를 묶은 일종의 패키징 기술이다. 여러 개 실행시키거나 다른 운영환경으로 옮겨서 실행할 수도 있어 간편하게 어플리케이션을 운영, 확장할 수 있다.

[0057] 컨테이너 기술이 급부상한 또 다른 이유는 개발과 운영의 틈새를 메우는 이른바 '데브옵스(DevOps)'에 도움이 되기 때문이다. 그 동안 개발자가 만든 시스템을 운영 환경으로 넘겨 배포하는 과정에서 문제가 생겨 시스템 가동 시기가 늦어지는 일이 많았다. 이때 컨테이너를 이용하면 패키징한 어플리케이션을 운영 환경에 그대로 이식해 실행할 수 있다.

[0058] 이에 따라, 개발자는 개발에 더 집중하고, 관리자는 손쉽게 배포, 관리할 수 있다. IT 아키텍트는 테스트 기간과 배포 시 오류를 줄이면서도 필요에 따라 유연하게 인프라를 확장할 수 있다.

[0059] 컨테이너 기술은 집적도와 배포 속도, 성능면에서 매우 유리하다. 컨테이너는 VM(Virtual Machine: 가상 머신)보다 10배 정도 집적도가 높다. 기존에는 서버 1대에 VM을 10대 설치했다면 컨테이너는 100개 이상 운영할 수 있다. 이는 컨테이너 구조에 하이퍼 바이저와 VM운영체제 계층이 없어 물리적인 자원을 덜 사용하기 때문이다.

[0060] 일반적으로 가상화 환경에서는 이들이 전체 자원의 10~20% 정도를 사용하는 것으로 알려졌다. 물리 서버에 운영환경을 구축하려면 보통 27시간 정도 소요된다. VM에서는 템플릿 기능을 이용한다고 해도 12분 정도 걸린다. 반면 컨테이너 인스턴스를 새로 만드는 데는 약 10초에 가능하다.

[0061] 또한, 컨테이너 기술은 가상화 기술보다 새로운 환경을 빠르게 구성할 수 있고, 이러한 시간을 더욱 단축시킬 수 있다. 또한, VM은 환경을 구성한 후 데이터를 올리고 다른 시스템과 연동하는 등의 수작업이 필요하지만, 컨테이너를 쓰면 이런 작업까지 컨테이너 이미지에 넣어 생략할 수 있어 효율적이다.

[0062] 컨테이너 기술은 성능 면에서 많은 장점이 있다. 컨테이너 기술은 하드웨어 위에 호스트 운영체제를 설치하고 그 위에 어플리케이션과 라이브러리를 패키지로 묶은 컨테이너를 올리는 구조로 형성된다.

[0063] 기존의 베어메탈 환경이라면 SSL 라이브러리에 문제가 생겼을 때 당장 이와 연결된 모든 어플리케이션에서 문제가 발생한다. 단일 라이브러리를 공유하는 구조이기 때문이다. 반면, 컨테이너는 내부 라이브러리를 컨테이너별로 사용하므로, 한 컨테이너가 장애를 일으켜도 다른 서비스는 영향을 받지 않는다.

[0064] 컨테이너 기술을 가상화 기술과 비교하는 경우, 컨테이너 기술의 가장 큰 장점은 실시간적인 민첩성과 대응이 필요한 곳에 적용할 수 있는 확장성이다.

[0065] 도 4를 참조하면, 보통 가상화는 자원이 부족하면 해당 VM에 자원을 더 할당하는 '스케일 업(Scale Up)' 방식으로 확장한다. 반면 컨테이너는 같은 역할을 하는 컨테이너를 하나 더 만들어 실행하는 '스케일 아웃(Scale Out)' 방식이다.

[0066] 스케일 아웃 방식은 장애 대응이나 서비스 확장 시 더 유연하게 대응할 수 있다. 컨테이너는 장애가 발생했을 때 기존 컨테이너를 복구하는 대신 새로 하나를 만들어 실행하는 것이 더 빠를 만큼 가볍고 유연하다.

[0067] 상기 하드웨어 모듈(110)은 상기 안드로이드 컴퓨트 노드(10)에 설치되어, Wifi, GPS, 블루투스, 카메라, GSM 연결과 같은 모바일 장치의 하드웨어 기반 기능을 제공한다.

[0068] 상기 리눅스 호스트(130)는 상기 하드웨어 모듈(110) 상에 운영체제가 설치되고, 상기 분석 에이전트(150)는 상기 리눅스 호스트(130)를 통해 상기 안드로이드 컨테이너(101, 102, 103) 내부에서 악성코드 동작 시 문제를 식별하고, 커널 관련 악성코드 활동을 탐지하여 분석 서버(50)로 전송한다.

[0069] 상기 클라우드 컨트롤러(30)는 상기 컨테이너 에이전트(170)를 통해 안드로이드 컨테이너(101, 102, 103)를 관리 및 구성하는데 사용된다.

[0070] 상기 분석 서버(50)는 상기 분석 에이전트(150)가 보낸 정보를 수신하여 악성코드 분석을 위한 동적 분석 및 정적 분석 수행을 통해 문제를 감지한다. 또한, 상기 안드로이드 컨테이너(101, 102, 103)에 APK 설치 명령을 지시한다.

[0071] 상기 모니터링 서버(70)는 상기 컨테이너 에이전트(170)와 연결되어 상기 안드로이드 컨테이너(101, 102, 103)의 상태 정보에 대하여 모니터링하여, 상기 안드로이드 컴퓨트 노드(10)의 분석 충돌 및 기타 문제를 해결한다.

[0072] 상기 사설 피코셀(Private Picocell) 네트워크는 상기 안드로이드 컨테이너(101, 102, 103)에 모바일 네트워크 기능을 제공한다.

[0073] 상기 사설 피코셀 네트워크는, GSM 모듈(90)을 이용하는 SDR(Software-Defined Radio) 기반의 RF 송수신기(92), GSM/LTE 모듈로 연결 가능한 개인 모바일 네트워크로 작동하는 피코셀 플랫폼(94) 및 상기 피코셀 플랫폼(94)으로부터 GSM/LTE 정보를 수집하여 악의적인 행동을 탐지하거나 상기 개인 모바일 네트워크 문제를 감시하는 관리 서버(96)를 포함할 수 있다.

[0074] 상기와 같은 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼에 따르면, 컨테이너 기술을 이용하므로, 기존의 QEMU 기술과 같이 하드웨어 가상화 기술이 필요 없으며, 실제 폰에서 동작하는 것과 같이 다양한 장비를 요구하지 않아 확장성과 탄력성을 제공할 수 있게 된다.

[0075] 또한, 본 발명에 따라 안드로이드 컨테이너에서 악성코드를 분석하게 될 경우, 안드로이드 공간에서 실행되는 APK 및 .so 라이브러리에 대한 분석이 가능해지며, 컨테이너를 쉽게 백업할 수 있어 다양한 공격에서부터 간단

히 복구가 가능하다.

[0076] 또한, 본 발명은 실제 하드웨어에서 정보를 직접 받아오기 때문에, 가상환경에 비하여 빠르며 에뮬레이터 우회 기술에 대비할 수 있다. 나아가, 본 발명은 피코셀(Picocell) 네트워크를 사용하기 때문에 실제 모바일 네트워크를 통한 공격에 대응 가능하며, 컨테이너 간 사설 네트워크를 생성하는 것 또한 가능하다.

[0078] 도 5는 본 발명의 일 실시예에 따른 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안 방법의 흐름도이다.

[0079] 본 실시예에 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안 방법은, 도 2의 컨테이너 플랫폼(1) 환경에서 도 3의 안드로이드 컴퓨트 노드(10)와 실질적으로 동일한 구성에서 진행될 수 있다.

[0080] 따라서, 도 2의 컨테이너 플랫폼(1) 환경에서 도 3의 안드로이드 컴퓨트 노드(10)와 동일한 구성요소는 동일한 도면부호를 부여하고, 반복되는 설명은 생략한다.

[0081] 또한, 본 실시예에 따른 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안을 수행하기 위한 소프트웨어(애플리케이션)에 의해 실행될 수 있다.

[0082] 도 5를 참조하면, 본 실시예에 따른 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 이용한 모바일 장치의 보안 방법은, 안드로이드 컴퓨트 노드(10)의 컨테이너 에이전트(170)를 통해 클라우드 컨트롤러(30)로부터 수신한 요청에 응답하여, 동적 분석을 위하여 안드로이드 악성코드를 직접 수행하는 적어도 하나의 안드로이드 컨테이너(101, 102, 103)를 생성한다(단계 S10).

[0083] 상기 클라우드 컨트롤러(30)는 상기 컨테이너 에이전트(170)를 통해 안드로이드 컨테이너(101, 102, 103)를 관리 및 구성한다. 상기 안드로이드 컨테이너(101, 102, 103)는 복수 개일 수 있고, 동적 분석을 위하여 안드로이드 악성코드를 직접 수행할 수 있도록 설계된다.

[0084] 안드로이드 컨테이너(101, 102, 103)의 상태는 OS, 빌드 버전, 루틴, 루팅 상태, 하드웨어 정보 등에 따라 각각 다르게 생성 가능하다.

[0085] 상기 안드로이드 컨테이너(101, 102, 103)가 생성되면, 리눅스 호스트(130)를 통해, 상기 안드로이드 컨테이너(101, 102, 103) 내부에서 악성코드 동작 시 문제를 식별한다(단계 S30). 상기 리눅스 호스트(130)는 상기 하드웨어 모듈(110) 상에 운영체제가 설치된다.

[0086] 또한, 분석 에이전트(150)를 통해 커널 관련 악성코드 활동을 탐지하여 분석 서버(50)로 전송한다(단계 S50). 상기 분석 서버(50)는 상기 분석 에이전트(150)가 보낸 정보를 수신하여 악성코드 분석을 위한 동적 분석 또는/및 정적 분석 수행을 통해 문제를 감지한다. 또한, 상기 안드로이드 컨테이너(101, 102, 103)에 APK 설치 명령을 지시할 수 있다.

[0087] 상기 안드로이드 컴퓨트 노드(10)의 분석 에이전트(150)는 상기 분석 서버(50)로부터 분석 정보를 수집한다(단계 S70).

[0088] 또한, 상기 컨테이너 에이전트(170)는 상기 안드로이드 컨테이너(101, 102, 103)의 상태 정보에 대해 모니터링 서버(70)에 전송할 수 있다. 상기 모니터링 서버(70)는 상기 컨테이너 에이전트(170)와 연결되어 상기 안드로이드 컨테이너(101, 102, 103)의 상태 정보에 대하여 모니터링하여, 상기 안드로이드 컴퓨트 노드(10)의 분석 충돌 및 기타 문제를 해결할 수 있다.

[0089] 이와 같은, 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼을 탑재한 모바일 장치의 보안 방법은 애플리케이션으로 구현되거나 다양한 컴퓨터 구성요소를 통하여 수행될 수 있는 프로그램 명령어의 형태로 구현되어 컴퓨터 판독 가능한 기록 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능한 기록 매체는 프로그램 명령어, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다.

[0090] 상기 컴퓨터 판독 가능한 기록 매체에 기록되는 프로그램 명령어는 본 발명을 위하여 특별히 설계되고 구성된 것들이거나와 컴퓨터 소프트웨어 분야의 당업자에게 공지되어 사용 가능한 것일 수도 있다.

[0091] 컴퓨터 판독 가능한 기록 매체의 예에는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM, DVD와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 ROM, RAM, 플래시 메모리 등과 같은 프로그램 명령어를 저장하고 수행하도록 특별히 구성된 하드웨

어 장치가 포함된다.

[0092] 프로그램 명령어의 예에는, 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드도 포함된다. 상기 하드웨어 장치는 본 발명에 따른 처리를 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0093] 이상에서는 실시예들을 참조하여 설명하였지만, 해당 기술 분야의 숙련된 당업자는 하기의 특허 청구의 범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위 내에서 본 발명을 다양하게 수정 및 변경시킬 수 있음을 이해할 수 있을 것이다.

산업상 이용가능성

[0094] 본 발명은 컨테이너 기술을 적용하여 안드로이드 악성코드 분석을 위한 플랫폼을 제안하므로, 안드로이드 장치상에서 안정성과 유연성이 뛰어나다. 따라서, 안드로이드 악성코드 분석에 유용하게 적용할 수 있을 것으로 기대된다.

부호의 설명

[0095] 1: 안드로이드 악성코드 분석을 위한 컨테이너 플랫폼

10: 안드로이드 컴퓨트 노드들

101, 102, 103: 안드로이드 컨테이너들

110: 하드웨어 모듈

130: 리눅스 호스트

150: 분석 에이전트

170: 컨테이너 에이전트

30: 클라우드 컨트롤러

50: 분석 서버

70: 모니터링 서버

90: GSM 모듈

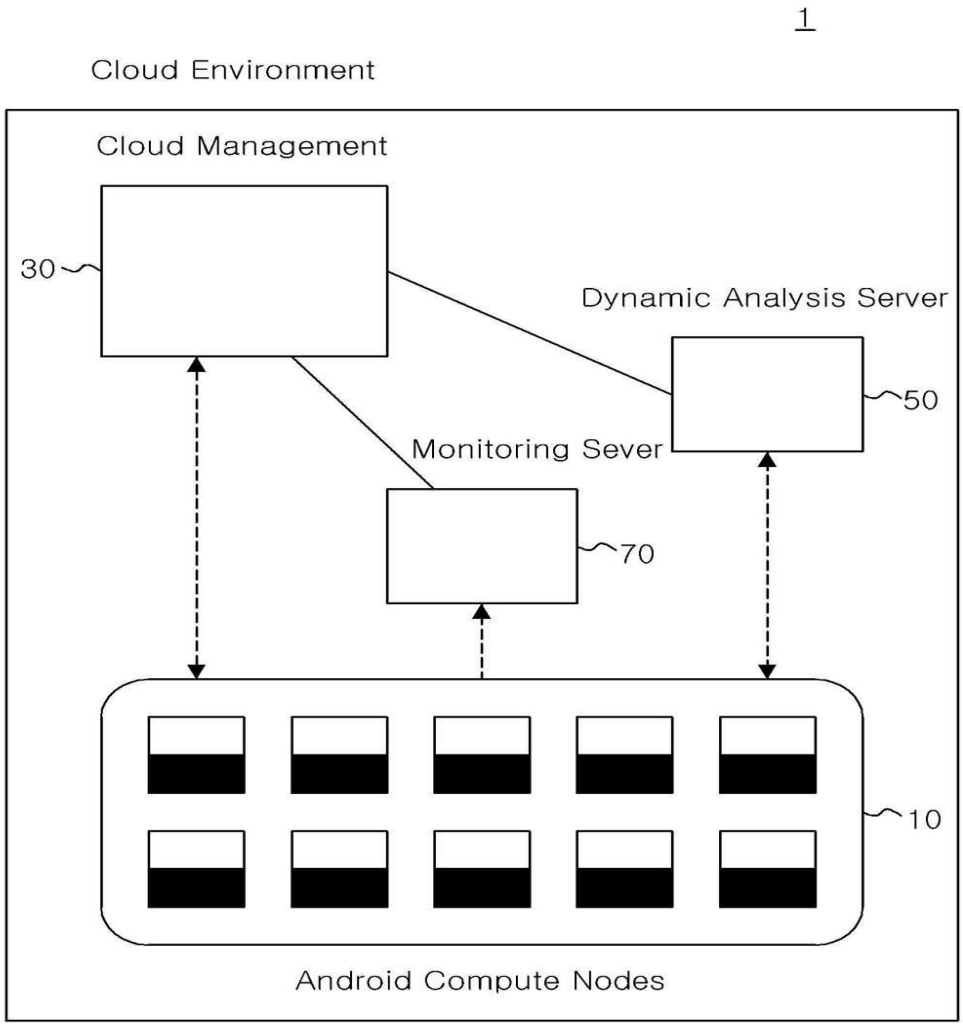
92: RF 송수신기

94: 피코셀 플랫폼

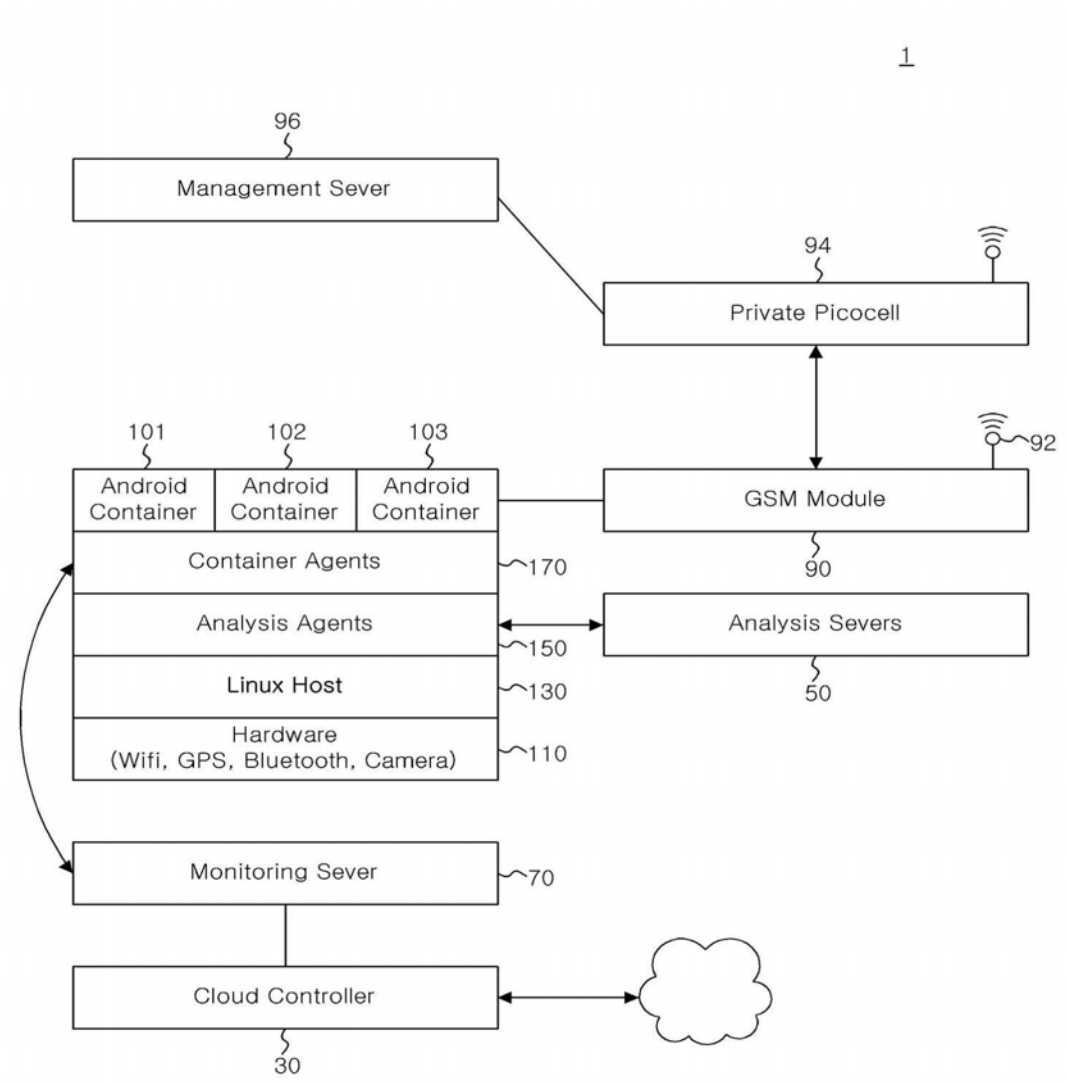
96: 관리 서버

도면

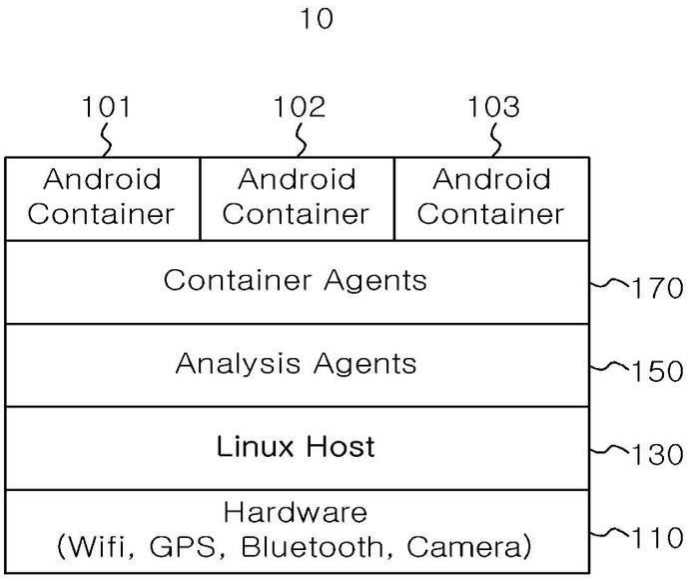
도면1


1
Cloud Environment
Cloud Management
30
Dynamic Analysis Server
50
Monitoring Sever
70
10
Android Compute Nodes

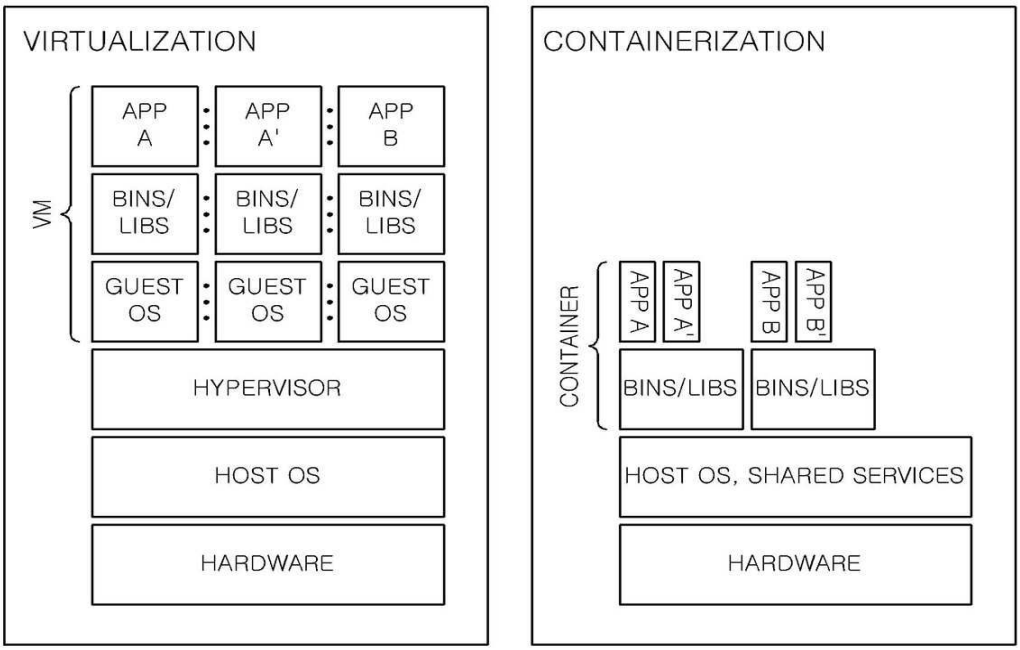
도면2


1
96
Management Sever
94
Private Picocell
92
101
102
103
Android Container
Android Container
Android Container
GSM Module
90
Container Agents
170
Analysis Agents
150
Analysis Severs
50
Linux Host
130
Hardware
(Wifi, GPS, Bluetooth, Camera)
110
Monitoring Sever
70
Cloud Controller
30

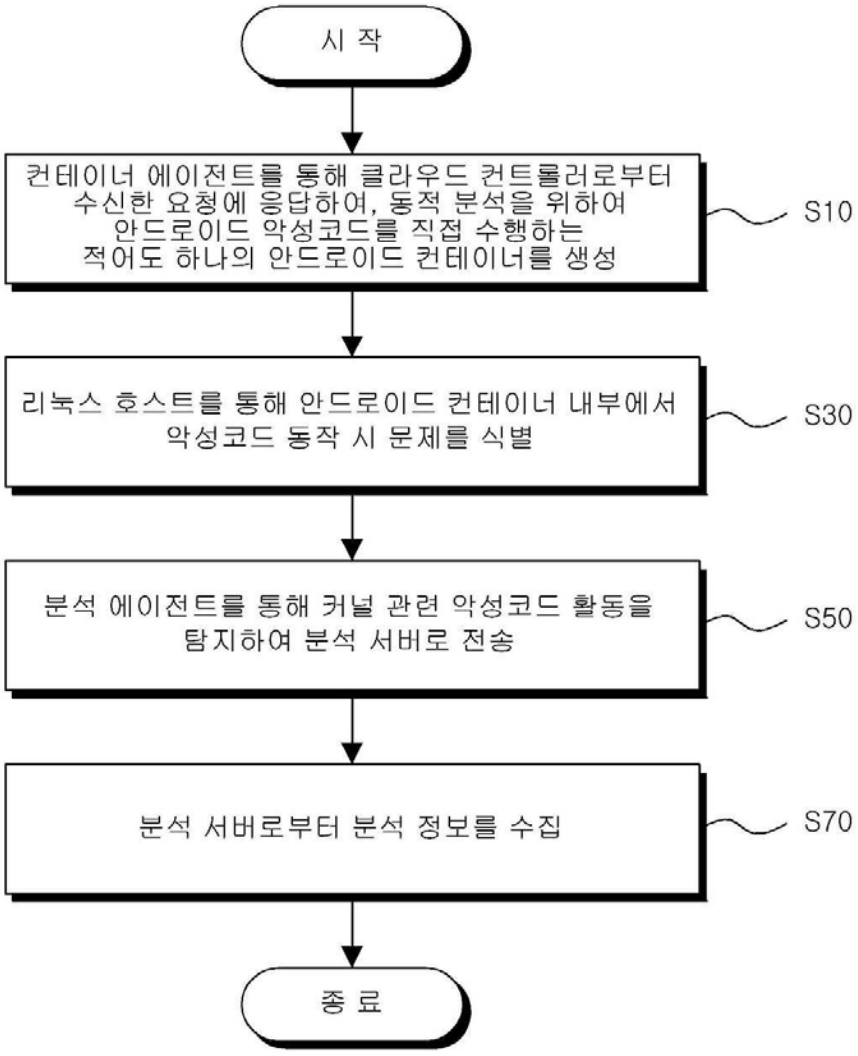
도면3


10
101
102
103
Android Container
Android Container
Android Container
Container Agents
170
Analysis Agents
150
Linux Host
130
Hardware
(Wifi, GPS, Bluetooth, Camera)
110

도면4


VIRTUALIZATION
VM
APP A
APP A'
APP B
BINS/ LIBS
BINS/ LIBS
BINS/ LIBS
GUEST OS
GUEST OS
GUEST OS
HYPERVISOR
HOST OS
HARDWARE
CONTAINERIZATION
CONTAINER
APP A
APP A'
APP B
APP B'
BINS/LIBS
BINS/LIBS
HOST OS, SHARED SERVICES
HARDWARE

도면5


시 작
컨테이너 에이전트를 통해 클라우드 컨트롤러로부터 수신한 요청에 응답하여, 동적 분석을 위하여 안드로이드 악성코드를 직접 수행하는 적어도 하나의 안드로이드 컨테이너를 생성
S10
리눅스 호스트를 통해 안드로이드 컨테이너 내부에서 악성코드 동작 시 문제를 식별
S30
분석 에이전트를 통해 커널 관련 악성코드 활동을 탐지하여 분석 서버로 전송
S50
분석 서버로부터 분석 정보를 수집
S70
종 료