



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2016-0031590

(43) 공개일자 2016년03월23일

(51) 국제특허분류(Int. Cl.)

G06F 21/56 (2013.01)

(21) 출원번호 10-2014-0120951

(22) 출원일자 2014년09월12일

심사청구일자 2014년09월12일

(71) 출원인

고려대학교 산학협력단

서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)

(72) 발명자

김휘강

서울 서초구 바우피로 91, 111동 1105호 (양재동, 우성아파트)

윤재성

경기도 고양시 일산동구 성현로 659번길 186, D동 402호(사리현동, SL그린빌)

(뒷면에 계속)

(74) 대리인

김동용

전체 청구항 수 : 총 11 항

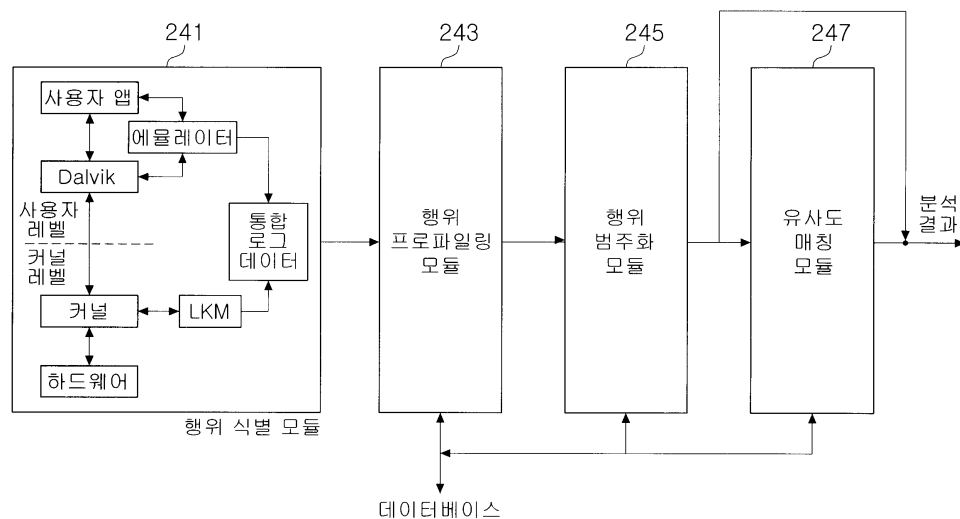
(54) 발명의 명칭 악성 앱 분류 장치 및 악성 앱 분류 방법

(57) 요약

본 발명은 앱의 분석에 기초한 데이터로부터 앱의 하나 이상의 악성 행위(operation)를 식별할 수 있는 행위 프로파일을 생성하는 행위 프로파일링 모듈 및 행위 프로파일로부터 분석되는 앱의 악성 행위 패턴에 따라 앱을 분류하는 행위 범주화 모듈을 포함하는 악성 앱 분류 장치 및 악성 앱 분류 방법에 관한 것이다.

본 발명을 이용함으로써 행위 기반의 프로파일링 기법을 적용하여 높은 정확도로 모바일 악성 앱을 탐지하고 분류할 수 있는 효과가 있다.

대표도 - 도2



(72) 발명자

장재욱

서울특별시 영등포구 가마산로 540, 104동 107호(신길동, 바다마을아파트)

우지영

충남 아산시 배방읍 용연로 37, 205동 603호 (용연마을휴먼시아2단지아파트)

이 발명을 지원한 국가연구개발사업

과제고유번호 H0301-14-1004

부처명 미래창조과학부

연구관리전문기관 정보통신산업진흥원

연구사업명 대학 IT 연구센터 육성지원

연구과제명 제어시스템 보안연구

기 여 율 1/1

주관기관 고려대 산학협력단

연구기간 2014.01.01 ~ 2014.12.31

명세서

청구범위

청구항 1

앱의 분석에 기초한 데이터로부터 상기 앱의 하나 이상의 악성 행위(operation)를 식별할 수 있는 행위 프로파일을 생성하는 행위 프로파일링 모듈; 및

상기 행위 프로파일로부터 분석되는 상기 앱의 악성 행위 패턴에 따라 상기 앱을 분류하는 행위 범주화 모듈;을 포함하는,

악성 앱 분류 장치.

청구항 2

제1항에 있어서,

상기 앱의 악성 행위를 식별할 수 있도록 하는 상기 데이터를 생성하는 행위 식별 모듈;을 더 포함하며,

상기 행위 식별 모듈은 상기 앱을 애플레이터를 통해 실행하여 사용자 레벨에서 수집되는 사용자 로그 데이터와 커널 레벨에서 수집되는 시스템 로그 데이터를 통합한 통합 로그 데이터를 생성하여 상기 행위 프로파일링 모듈로 전달하며,

상기 통합 로그 데이터는 시스템 콜과 시스템 콜에서 이용되는 매개변수를 포함하는,

악성 앱 분류 장치.

청구항 3

제1항에 있어서,

상기 악성 행위는 과금 전화, 과금 문자, 개인 정보 전송 또는 데이터 변환을 나타내고,

상기 행위 범주화 모듈은 상기 앱에서 수행되는 과금 전화, 과금 문자, 개인 정보 전송 및 데이터 변환의 하나 이상의 조합으로 인식되는 악성 행위 패턴으로 상기 앱을 분류하는,

악성 앱 분류 장치.

청구항 4

제1항에 있어서,

카테고리별로 악성 앱들을 분류하고 카테고리 내에서 상기 악성 앱들을 복수의 서브-카테고리로 분류하며 서브-카테고리별 대표 프로파일을 관리하는 데이터베이스; 및

상기 행위 범주화 모듈에 의해 분류된 앱의 대응 카테고리 내에서 서브-카테고리별 대표 프로파일과의 유사도 비교로 상기 분류된 앱의 서브-카테고리를 결정하는 유사도 매칭 모듈;을 더 포함하는,

악성 앱 분류 장치.

청구항 5

제4항에 있어서,

상기 유사도 매칭 모듈은, 과금 전화, 과금 문자, 개인 정보 전송 및 데이터 변환 악성 행위별로, 상기 분류된 앱과 대표 프로파일간의 유사성(similarity)의 합산을 통해 유사도를 계산하고,

과금 전화 악성 행위 및 과금 문자 악성 행위의 유사성에 적용되는 가중치는 개인 정보 전송 악성 행위 및 데이터 변환 악성 행위에 적용되는 가중치보다 큰,

악성 앱 분류 장치.

청구항 6

제4항에 있어서,

상기 앱의 서브-카테고리 결정에 따라 상기 대표 프로파일은 업데이트되며,

상기 대표 프로파일은 결정된 서브-카테고리 내의 앱들의 프로파일의 교집합(Intersection) 또는 합집합(Union)으로 업데이트되는,

악성 앱 분류 장치.

청구항 7

제5항에 있어서,

상기 악성 행위는 이름, 목표 및 속성으로 구성되고

악성 행위별로 계산되는 유사성은 적어도 상기 악성 행위에 포함된 속성에 기초하여 계산되는,

악성 앱 분류 장치.

청구항 8

제4항에 있어서,

클라이언트로부터 앱 정보를 수신하고 상기 앱 정보에 따른 분석 결과를 송신하는 송수신부; 및

인터넷을 통해 앱들을 수집하는 크롤러(crawler);를 더 포함하며,

상기 악성 앱 분류 장치는 크롤러를 통해 수집된 앱들을 데이터베이스에 분류하고 수신된 앱 정보의 해쉬 값으로 식별되는 앱이 이미 상기 데이터베이스에서 분류된 경우에 상기 데이터베이스로부터 결정되는 분석 결과를 클라이언트로 송수신부를 통해 전송하는,

악성 앱 분류 장치.

청구항 9

(b) 앱의 분석에 기초한 데이터로부터 상기 앱의 하나 이상의 악성 행위(operation)를 식별할 수 있는 행위 프로파일을 생성하는 단계; 및

(c) 상기 행위 프로파일로부터 분석되는 상기 앱의 악성 행위 패턴에 따라 상기 앱을 분류하는 단계;를 포함하는,

악성 앱 분류 방법.

청구항 10

제9항에 있어서,

상기 단계 (b) 이전에, (a) 상기 앱의 악성 행위를 식별할 수 있도록 하는 상기 데이터를 생성하는 단계;를 더 포함하며,

상기 단계 (a)는 상기 앱을 에뮬레이터를 통해 실행하여 사용자 레벨에서 수집되는 사용자 로그 데이터와 커널 레벨에서 수집되는 시스템 로그 데이터를 통합한 통합 로그 데이터를 상기 앱의 분석에 기초한 데이터로 생성하며,

상기 통합 로그 데이터는 시스템 콜과 시스템 콜에서 이용되는 매개변수를 포함하는,

악성 앱 분류 방법.

청구항 11

제9항에 있어서,

분류된 앱의 대응 카테고리 내에서 서브-카테고리별 대표 프로파일과의 유사도 비교로 분류된 앱의 서브-카테고리

리를 결정하는 단계;를 더 포함하는,
악성 앱 분류 방법.

발명의 설명

기술 분야

[0001] 본 발명은 악성 앱 분류 장치 및 악성 앱 분류 방법에 관한 것으로서, 구체적으로는 앱의 로그 데이터부터 행위 기반의 프로파일을 구성하여 악성 앱을 탐지 및 분류할 수 있도록 하는 악성 앱 분류 장치 및 악성 앱 분류 방법에 관한 것이다.

배경 기술

[0002] 스마트폰의 대중화로 언제 어디서나 정보를 액세스하고 교환할 수 있는 환경이 구축되었다. 스마트폰의 보급은 기하급수적으로 증가하고 있지만 모바일 악성 앱에 의한 정보유출 등과 같은 해킹 위협 또한 증가하고 있다.

[0003] 특히 안드로이드 플랫폼을 기반으로 하는 스마트폰이 해킹 위협에 더 많이 노출되어 있다. 이는 안드로이드 플랫폼의 개방 정책에 따라 어떤 개발자나 안드로이드 마켓에 간단한 인증으로 손쉽게 악성 앱을 업로드할 수 있고 앱 자체의 크랙과 리패키징이 용이하기 때문이다.

[0004] 기하급수적으로 증가하는 악성 앱의 확산을 막기 위해서, 백신 회사는 기존 PC에서 적용한 악성코드 탐지방법인 시그니처(signature) 기반의 탐지방법을 모바일 환경에서도 확대 적용하고 있다. 시그니처 기반의 탐지방법은 지속적으로 데이터베이스를 업데이트 해야하고 알려지지 않은 공격(Zero-day)에 대해서는 탐지가 어렵고 악성 행위 자체는 유지한 채 앱 코드의 일부 또는 전부를 변경한 악성 앱의 경우 탐지율이 현저히 떨어지는 문제가 존재한다.

[0005] 이러한 시그니처 기반의 탐지방법에 대응하기 위해서 많은 선행연구가 진행되고 있다. 선행연구는 주로 퍼미션(permission) 기반, API(Application Programming Interface) 기반 또는 시스템 콜 기반으로 악성 앱을 탐지하고 있다.

[0006] 퍼미션 기반의 악성 앱 탐지 방법은 정상(Benign) 앱을 악성 앱으로 분류하는 오탐(false positive)이 많고 API 기반의 악성 앱 탐지 방법은 디컴파일과 디스어셈블리 과정이 완료되어야 하고 변조 및 난독화 기술에 취약하다. 또한 시스템 콜 기반의 악성 앱 탐지 방법은 시스템 콜 빈도수에만 기초하고 다른 시스템 콜 인자(Argument)를 고려하지 않기에 정확도가 떨어진다.

[0007] 이와 같이 기존의 악성 앱 탐지방법의 선행 연구에서 발견되는 한계들을 해소할 수 있고 정확하게 악성 앱을 분류할 수 있도록 하는 새로운 악성 앱 분류 장치 및 악성 앱 분류 방법이 필요하다.

발명의 내용

해결하려는 과제

[0008] 본 발명은 상술한 문제점을 해결하기 위해서 안출한 것으로서, 앱에 대한 동적 분석으로 출력되는 사용자 레벨 및 커널 레벨의 로그 데이터로부터 행위 기반의 프로파일을 생성하여 악성 앱을 탐지하고 분류할 수 있도록 하는 악성 앱 분류 장치 및 악성 앱 분류 방법을 제공하는 데 그 목적이 있다.

과제의 해결 수단

[0009] 상기와 같은 목적을 달성하기 위한 악성 앱 분류 장치는 앱의 분석에 기초한 데이터로부터 앱의 하나 이상의 악성 행위(operation)를 식별할 수 있는 행위 프로파일을 생성하는 행위 프로파일링 모듈 및 행위 프로파일로부터 분석되는 앱의 악성 행위 패턴에 따라 앱을 분류하는 행위 범주화 모듈을 포함한다.

[0010] 또한 상기와 같은 목적을 달성하기 위한 악성 앱 분류 방법은 (b) 앱의 분석에 기초한 데이터로부터 앱의 하나 이상의 악성 행위(operation)를 식별할 수 있는 행위 프로파일을 생성하는 단계 및 (c) 행위 프로파일로부터 분석되는 앱의 악성 행위 패턴에 따라 앱을 분류하는 단계를 포함한다.

발명의 효과

[0011] 상기와 같은 본 발명에 따른 악성 앱 분류 장치 및 악성 앱 분류 방법은 앱에 대한 동적 분석으로 출력되는 사용자 레벨 및 커널 레벨의 로그 데이터로부터 행위 기반의 프로파일을 생성하여 악성 앱을 탐지하고 분류할 수 있도록 하는 효과가 있다.

도면의 간단한 설명

[0012] 도 1은 본 발명에 따라 악성 앱 탐지 및 분류를 위한 예시적인 시스템 블록도와 개략적인 내부 구조를 도시한 도면이다.

도 2는 앱 분석부의 예시적인 기능 블록도를 도시한 도면이다.

도 3은 악성 앱 탐지 및 분류를 위한 예시적인 흐름도를 도시한 도면이다.

도 4는 특정 악성 앱에 대하여 생성된 행위 프로파일의 예를 도시한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0013] 상술한 목적, 특징 및 장점은 첨부된 도면을 참조하여 상세하게 후술 되어 있는 상세한 설명을 통하여 더욱 명확해 질 것이며, 그에 따라 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 본 발명의 기술적 사상을 용이하게 실시할 수 있을 것이다. 또한, 본 발명을 설명함에 있어서 본 발명과 관련된 공지 기술에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에 그 상세한 설명을 생략하기로 한다. 이하, 첨부된 도면을 참조하여 본 발명에 따른 바람직한 실시 예를 상세히 설명하기로 한다.

[0014] 도 1은 본 발명에 따라 악성 앱 탐지 및 분류를 위한 예시적인 시스템 블록도와 개략적인 내부 구조를 도시한 도면이다.

[0015] 도 1에 따르면 악성 앱 탐지 및 분류를 위한 시스템은 하나 이상의 클라이언트(100)(Client), 악성 앱 분류 장치(200) 및 통신망(300)을 포함한다.

[0016] 시스템의 각 블록에 대해서 살펴보면, 클라이언트(100)는 일반 사용자가 이용하는 단말기로서 예를 들어 스마트폰, 태블릿 PC 등일 수 있다. 클라이언트(100)는 프로세서를 내장하고 각종 응용 프로그램을 수행할 수 있다. 이하 응용 프로그램을 앱(App)으로 지칭한다. 클라이언트(100)는 바람직하게 이동중에도 이동통신망을 통한 전화, SMS(Short Message Service) 또는 MMS(Multimedia Message Service)가 가능한 휴대형 단말기이다. 클라이언트(100)는 이동통신망이나 인터넷망을 통해 각종 앱을 다운받을 수 있고 다운받은 앱을 실행시킬 수 있다.

[0017] 클라이언트(100)는 다운받은 앱의 실행 이전에 웹 브라우저를 통해 먼저 악성 앱 분류 장치(200)로 다운받은 앱이 악성 앱인지 또는 악성 앱인 경우 어떤 악성 앱인지를 알기 위해 앱 정보를 전송하고 해당 앱에 대한 분석 결과를 수신하도록 구성될 수 있다. 앱 정보는 예를 들어 다운받은 앱의 이름과 앱의 해쉬값(hash value)을 포함한다. 또는 앱 정보는 다운받은 앱의 패키지(Package)를 포함할 수도 있다.

[0018] 악성 앱 분류 장치(200)는 통신망(300)을 통해 하나 이상의 클라이언트(100)에 연결되어 클라이언트(100)로부터의 앱 탐지 및 분류 요청을 수신하고 수신된 앱 정보를 이용하여 앱이 악성인지 탐지하고 및/또는 악성인 경우 어떠한 악성 앱인지를 분류하여 그 분석 결과를 클라이언트(100)로 전송하도록 구성된다. 악성 앱 분류 장치(200)는 소위 서버를 구성하고 예를 들어 웹 페이지를 HTTP(hyper text transfer protocol)를 통해 제공하는 웹 서버로서 동작할 수 있다.

[0019] 악성 앱 분류 장치(200)는 적어도 통신망(300)에 인터페이스 하기 위한 통신 인터페이스, 각종 프로그램을 수행할 수 있도록 하는 하나 이상의 프로세서 및 각종 데이터를 저장하기 위하여 하나 이상의 하드 디스크로 그 하드웨어가 구성된다.

[0020] 악성 앱 분류 장치(200)는 기능 블록으로서 송수신부(210), 데이터베이스(220), 앱 확인부(230), 앱 분석부(240) 및 크롤러(250)(crawler)를 포함한다. 각각의 기능 블록들은 바람직하게는 악성 앱 분류 장치(200)의 하드웨어와 결합하여 구성된다. 예를 들어 송수신부(210)는 통신 인터페이스와 통신 인터페이스상에서의 통신패킷을 처리하여 통신망(300)을 통해 데이터를 송수신한다. 데이터베이스(220)는 하드디스크 등을 이용하여 각종 데이터를 액세스 및 관리를 할 수 있도록 한다. 그리고 크롤러(250), 앱 확인부(230) 및 앱 분석부(240)는 프로세서에서 실행할 수 있는 프로그램의 형태로 구성된다.

- [0021] 각각의 기능 블록의 기능과 그 관계를 좀 더 살펴보면, 크롤러(250)는 인터넷망을 통해 공식/비공식 마켓으로부터 앱 파일들(예를 들어 APK(Android PacKage) 파일)을 지속적으로 수집하고 수집된 앱 파일들을 데이터베이스(220)나 하드디스크 등에 저장한다. 크롤러(250)는 클라이언트(100)에 의한 분석 요청 이전에 먼저 앱 파일들을 수집하여 앱 분석부(240)를 통해 미리 악성 앱인지를 결정하고 악성 앱 인 경우 해당 앱을 분류할 수 있도록 한다.
- [0022] 송수신부(210)는 클라이언트(100)로부터 앱의 분석 대상을 나타내는 앱 정보를 수신하고 수신된 앱 정보를 데이터베이스(220)나 데이터베이스(220)를 구성하는 하드디스크 등에 저장한다. 앱 정보는 예를 들어 앱 파일의 이름(식별자)과 앱 파일의 해쉬값을 포함한다. 또는 앱 정보는 앱 파일(APK 파일) 자체를 나타낼 수도 있다. 앱 정보는 또한 앱 확인부(230)로 바로 전달될 수도 있다. 송수신부(210)는 또한 앱 확인부(230) 또는 앱 분석부(240)를 통해 생성된 분석 결과를 요청한 클라이언트(100)로 전송한다.
- [0023] 앱 확인부(230)와 앱 분석부(240)는 클라이언트(100)로부터 앱 분석 요청에 따른 앱 정보를 이용하여 앱이 악성인지 나아가 앱이 악성인 경우에 해당 앱을 분류하여 카테고리를 유사도에 따라 결정할 수 있다.
- [0024] 앱 확인부(230)는 클라이언트(100)로부터의 앱 정보의 해쉬값과 데이터베이스(220)에 이미 분석이 완료되어 카테고리별로 분류된 앱들의 해쉬값들을 비교한다. 만일 동일한 해쉬값이 데이터베이스(220)에 존재하는 경우에 데이터베이스(220)로부터 추출되는 정보를 이용하여 결정되는 분석 결과를 앱 확인부(230)가 생성한다. 이후 앱 확인부(230)는 분석 결과를 송수신부(210)를 통해 클라이언트(100)로 전송한다.
- [0025] 클라이언트(100)로부터의 앱 정보가 데이터베이스(220)에서 식별되지 않는 경우에, 앱 확인부(230)는 크롤러(250)를 이용하여 앱 파일을 인터넷망의 공식/비공식 마켓으로부터 자동 추출하여 추출된 앱 파일을 데이터베이스(220)에 저장하고 또한 앱 파일을 앱 분석부(240)로 전달한다. 만일 공식/비공식 마켓으로부터 앱 파일이 추출되지 않는 경우 앱 확인부(230)는 클라이언트(100) 측으로 앱 파일을 앱 정보로서 요청할 수도 있다.
- [0026] 앱 분석부(240)는 클라이언트(100)로부터 탐지 및 분류 요청된 앱 파일 또는 크롤러(250)를 통해 수집된 앱 파일들에 대해서 행위 기반의 프로파일링을 통해 카테고리로 분류하고 각 카테고리 내에서 서브-카테고리를 결정할 수 있다. 앱 분석부(240)를 통한 분석 결과는 데이터베이스(220)에 해쉬값과 같이 저장되고 또한 클라이언트(100)로 송수신부(210)를 통해 전송될 수 있다.
- [0027] 클라이언트(100)는 수신된 분석결과를 화면상에 디스플레이하고 이후 다운로드된 앱의 설치 여부를 사용자가 분석결과에 따라 결정할 수 있다.
- [0028] 앱 분석부(240)에 대해서는 도 2 및 도 3을 통해서 상세히 살펴볼도록 한다.
- [0029] 통신망(300)은 악성 앱 분류 장치(200)와 클라이언트(100) 사이의 데이터를 송수신할 수 있는 유선, 무선 또는 유무선 망이다. 통신망(300)은 이동통신망 또는 인터넷망을 포함한다. 물론 이동통신망은 인터넷망을 구성할 수도 있다.
- [0030] 도 2는 앱 분석부(240)의 예시적인 기능 블록도를 도시한 도면이다.
- [0031] 도 2에 따르면 앱 분석부(240)는 행위 식별 모듈(241), 행위 프로파일링 모듈(243), 행위 범주화 모듈(245) 및 유사도 매칭 모듈(247)을 포함한다.
- [0032] 각 모듈들을 살펴보면, 행위 식별 모듈(241)은 행위 프로파일 생성에 이용될 로그 데이터를 생성한다. 이 로그 데이터는 이후 분석 대상인 앱의 행위를 식별할 수 있도록 한다.
- [0033] 좀 더 구체적으로 살펴보면, 행위 식별 모듈(241)은 분석 대상인 앱을 에뮬레이터(emulator) 환경에서 실행하고 사용자 레벨에서의 사용자 로그 데이터와 커널(Kernel) 레벨에서의 시스템 로그 데이터를 통합하여 통합 로그 데이터를 생성한다. 생성된 통합 로그 데이터는 이후 행위 프로파일링을 위해 행위 프로파일링 모듈(243)로 전달된다.
- [0034] 예를 들어 행위 식별 모듈(241)은 사용자 레벨에서 수행되는 에뮬레이터와 커널에서 동적으로 결합되는 적재가 가능한 커널 모듈(LKM : Loadable Kernel Module)를 이용하여 일정 시간 이상 앱을 실행하여 에뮬레이터로부터 사용자 로그 데이터를 수집하고 커널 모듈로부터 시스템 로그 데이터를 수집한다. 에뮬레이터는 예를 들어 안드로이드 앱 실행이 에뮬레이팅 가능한 Droidbox이다.

[0035] 행위 식별 모듈(241)은 사용자 로그 데이터 중 데이터 누수(leak) 내역, SMS/통화 내역을 수집하고 동적 커널 모듈로부터 시스템 콜과 매개변수를 수집하여 이를 통합 로그 데이터로 구성한다. 생성된 통합 로그 데이터는 적어도 시스템 콜(system call)과 시스템 콜에서 이용되는 매개변수를 포함한다.

[0036] 일반적으로 악성 앱 제작자는 악성행위와 관련있는 API 호출 사실을 사용자에게 숨길 수 있기 때문에 앱의 악성 행위를 파악하기 용이치 않다. 그래서 시스템 콜 후킹(Hooking)을 통해 어떤 시스템 함수가 호출되는 지 그리고 어떠한 매개변수를 가지는 지를 로그로 기록하면 API 함수 호출을 숨기는 경우에도 정확한 행위를 파악할 수 있다.

[0037] 행위 프로파일링 모듈(243)은 행위 식별 모듈(241)의 통합 로그 데이터로부터 분석 대상인 앱의 하나 이상의 행위(Operation)를 식별할 수 있는 행위 프로파일을 생성한다. 즉 행위 프로파일링 모듈(243)은 통합 로그 데이터를 파싱하고 파싱된 데이터로부터 행위 프로파일을 구성한다.

[0038] 행위 프로파일링 모듈(243)에서 생성되는 행위 프로파일에 대해서 여기서 구체적으로 살펴보도록 한다.

[0039] 행위 프로파일(P)는 $P=(O, OP, \Gamma, \Delta)$ 와 같이 4개의 원소를 갖는 집합으로 구성된다. O 는 분석 대상인 앱에서 파악되는 모든 객체 셋(Set)을 의미하고 OP 는 모든 오퍼레이션(Operation) 집합을 나타내며 키(Key)와 값(Value)로 구성된 내재된 사전 구조(nested dictionary)를 가진다. $\Gamma \subseteq (O \times OP)$ 는 객체와 오퍼레이션과의 매핑 관계를 의미하며 $\Delta \subseteq ((O \times OP), (O \times OP))$ 는 순서(Sequence)와 관련 없는 전체 객체-오퍼레이션 관계 집합을 의미한다. Δ 는 통합 로그 데이터로 표현된다.

[0040] 객체(Object)는 악성 앱이 각각의 행위를 하는데 필요한 추상적인 기능(functionality)을 의미한다. 객체는 특정 행위에 대한 추상화를 의미하고 아래와 같이 기호화하여 설명할 수 있다. 이하 살펴볼 바와 같이, 악성 앱의 행위 패턴은 과금 전화번호로 전화를 거는 행위(이하 '과금 전화'라고도 함), 과금 전화번호로 SMS 보내는 행위(이하 '과금 문자'라고도 함), 민감한 개인 정보를 수집하여 다른 곳으로 전송하는 행위 (이하 '개인 정보 전송'이라고도 함) 및 전송을 위하여 데이터를 변환하는 행위(이하 '데이터 변환'이라고도 함)의 하나 이상의 조합으로 결정된다.

Object	::=	Object-type
Object-type	::=	Telephony Phone Network

[0041]

[0042] 오퍼레이션은 구체적인 악성 행위를 나타낸다. 이하에서는 오퍼레이션을 '악성 행위'라고 지칭할 수도 있다. 오퍼레이션은 이름(name), 목표(target) 및 속성(attribute)로 구성된다. 이름은 악성 행위의 식별자(또는 구분자)를 의미하고 목표는 악성 앱의 공격 목표를 의미한다. 예를 들어 클라이언트(100)의 외부 메모리의 내용이나 시스템 정보 등은 공격 목표가 될 수 있다. 속성은 공격 목표에 대하여 추출된 데이터, 예를 들어 과금/문자 전화번호, 국가 코드나 유심 정보 등 악성 앱이 목표로 하는 민감한 개인 정보를 자체를 의미한다. 오퍼레이션은 다음과 같이 기호화하여 설명될 수 있다.

```
Operation ::= { Operation-name : {
                    Operation-target :
                    Operation-attribute } }
```

```
Operation-name ::= Sending SMS |
Calling number|Sending sensitive
info.|Converting data
```

```
Operation-target ::= Premium-rate
SMS/number|Device ID|IMEI|IMSI |
MMC | MNC | .... | etc
```

[0043]

[0044]

아래 표 1은 네트워크 객체와 대응되는 오퍼레이션간의 관계(Γ)에 대한 예제이다.

표 1

타입(Type)	이름(Name)	목표(Target)	속성(Attribute)
네트워크	개인 정보 전송	Device ID	6A27B03C155564A0E484 EB56AA59C6D4
		Device	Android
		IMEI	357242043237517
		IMSI	310005123456789
		MCC	310
		MNC	260
		OS version	10
		SDK version	2.3.3
		Carrier	Android
		Country code	en
		Location	GPS Coordinates
	데이터 변환	암호화 방식	No, DES, AES, Blowfish
		목적지 URL	http://my365image.com
		포트(Port)	80
		인코딩 방식	gzip

[0045]

[0046]

표 1에서 알 수 있는 바와 같이, 네트워크 객체는 두 개의 오퍼레이션에 매핑되고 각각의 오퍼레이션들은 다시 하나 혹은 다수의 속성들을 전송하거나 변환하는 것을 알 수 있다.

[0047]

행위 프로파일링 모듈(243)은 통합 로그 데이터로부터 객체 및 오퍼레이션(악성 행위)을 식별하고 객체 및 악성 행위 간의 매핑 관계 그리고 악성 행위의 이름, 목표, 속성 등을 내제된(nested) 형식으로 구성한다. 예를 들어 행위 프로파일링 모듈(243)은 파이썬(PYTHON) 프로그래밍 언어의 내제된 사전구조(nested dictionary)로 이러한 행위 프로파일(P)을 표현할 수 있다.

[0048]

각각의 악성 행위는 에뮬레이션(emulation) 과정(또한 커널 후킹을 통해)에서 식별될 수 있다. 악성 행위는 예를 들어 클라이언트(100)를 이용하는 사용자에게 그 행위를 알리지 않고 과금 문자, 과금 전화, 개인 정보 전송 또는 데이터 변환을 수행하는 행위이다. 이러한 행위는 앞서 살펴본 바와 같은 4가지 악성 행위로 범주화되고 앱은 행위 프로파일을 통해 4가지 악성 행위의 조합으로 표현된다.

- [0049] 도 4는 특정 악성 앱에 대하여 생성된 행위 프로파일의 예를 도시한 도면이다. 도 4의 예는 자료 수집형 악성 앱의 일종인 FakeBattScar에 대해서 수집된 통합 로그 데이터로 행위 프로파일링 모듈(243)에서 생성된 행위 프로파일의 예이다. FakeBattScar는 광고를 디바이스 화면에 띄우고, 디바이스 모델, 제조사, 이동통신사 정보, 위치정보, IMEI, OS/SDK 버전 등을 악성 앱 제작자의 서버로 전송한다. 도 4에서 알 수 있는 바와 같이 행위 프로파일을 통해 분석가는 직관적으로 해당 앱의 악성 행위 유무를 파악할 수 있다.
- [0050] 행위 프로파일링 모듈(243)에 의해서 분석 대상인 앱에 대한 행위 프로파일이 구성되고 행위 범주화 모듈(245)로 이 행위 프로파일(P)이 전달된다.
- [0051] 행위 범주화 모듈(245)은 행위 프로파일로부터 분석되는 앱의 행위 패턴에 따라 분석 대상인 앱을 특정 카테고리로 분류한다. 대분류로 이용되는 카테고리들은 예를 들어 4가지 악성 행위의 조합으로 구성되고 16가지의 행위 패턴(하나의 행위 패턴은 정상 앱을 나타내고 나머지 15가지의 행위 패턴은 4가지 악성 행위에 대한 1가지 이상의 조합을 나타냄)에 각각 대응한다.
- [0052] 이와 같이 행위 범주화 모듈(245)은 앱에서 수행되는 과금 전화, 과금 문자, 개인 정보 전송 또는 데이터 변환의 악성 행위 조합으로 인식되는 행위 패턴으로 분석 대상인 앱을 분류한다.
- [0053] 만일 행위 프로파일에 사전 정의된 4가지 악성 행위가 전혀 존재하지 않는 경우 행위 범주화 모듈(245)은 해당 앱을 정상 앱으로 간주하고 정상 앱을 나타내는 카테고리에 해당 앱 정보(예를 들어 앱 이름 및 해쉬값)를 저장한다. 그리고 행위 범주화 모듈(245)은 이후 유사도 매칭 모듈(247)을 구동하지 않고 정상임을 나타내는 분석 결과를 송수신부(210)를 통해 클라이언트(100)로 전달할 수 있다.
- [0054] 유사도 매칭 모듈(247)은 행위 범주화 모듈(245)에 의해서 분류된 분석 대상 앱의 대응 카테고리 내에서 복수의 서브-카테고리(sub-category)별 대표 (행위) 프로파일과의 유사도 비교로 행위 범주화 모듈(245)에서 분류된 앱의 서브-카테고리를 결정한다.
- [0055] 여기서, 악성 앱 분류 장치(200)에 포함된 데이터베이스(220)를 좀 더 구체적으로 살펴보면, 데이터베이스(220)는 크롤러(250) 또는 클라이언트(100)로부터 요청된 앱들을 16가지 카테고리로 대분류하고 카테고리 내 앱들을 다시 군집별로 서브-카테고리로 분류한다. 16가지 카테고리 중 하나는 정상 앱을 위한 카테고리이며 나머지 15가지는 4가지 악성 행위의 하나 이상의 조합의 악성 행위 패턴을 나타낸다.
- [0056] 악성 행위 패턴의 각 카테고리 내의 서브-카테고리 각각은 예를 들어 악성 앱의 진단명을 나타낼 수 있다. 특정 서브-카테고리에 분류된 앱들은 앱 파일에 대한 해쉬값 및 앱에 대하여 행위 프로파일링 모듈(243)에 의해서 생성된 행위 프로파일로 데이터베이스(220)에 저장된다. 데이터베이스(220)에서 특정 서브-카테고리 각각은 대표 프로파일을 또한 저장한다.
- [0057] 대표 프로파일은 서브-카테고리 내의 앱들의 프로파일에 의해서 결정된다. 대표 프로파일은 서브-카테고리의 그룹의 공통적인 성격을 가지도록 구성된다. 예를 들어 대표 프로파일은 분석 대상인 앱이 특정 서브-카테고리로 분류 결정됨에 따라 업데이트되며 업데이트 방식은 현재 대표 프로파일과 분석 대상인 앱의 행위 프로파일의 합집합(Union) 또는 교집합(Intersection)으로 업데이트된다.
- [0058] 교집합 방식은 악성 앱의 변종으로 인해 발생하는 노이즈에 용이하게 대응할 수 있고 해당 군(서브-카테고리)에 속하는 악성 앱들의 공통적인 성격을 유지하는 방향으로 업데이트한다. 합집합 방식은 다양한 변종을 수용할 수 있는 방향으로 업데이트한다.
- [0059] 유사도 매칭 모듈(247)은 유사도 비교로 분석 대상 앱의 서브 카테고리 결정하게 되는 데, 유사도 매칭 모듈(247)은 과금 전화, 과금 문자, 개인 정보 전송 및 데이터 변환 악성 행위별로 서브-카테고리별 대표 프로파일과의 유사성 점수의 합산으로 유사도를 계산한다.
- [0060] 유사성 점수는 전화나 카메라 등과 같은 하드웨어 자원과 시스템 또는 개인 정보에 접근할 수 있는 정도로 수치화된다.
- [0061] 서브-카테고리별 유사도(S)는

수학식 1

$$S = \sum_i w_i \cdot BFS_i$$

로 계산되고, $\sum_i w_i = 1$ 이다. BFS_i 와 w_i 는 i 번째 행위 유사성 요소와 가중치를 나타낸다. BFS(Behavior Factor Similarity)는 행위 유사성 요소로서 인덱스 i에 따라 4가지를 나타낸다. i 인덱스 값에 따라 즉 과금 전화 악성 행위에 대한 유사성(CS : Similarity of Calling premium-rate number), 과금 문자 악성 행위에 대한 유사성(SS : Similarity of Sending premium-rate SMS), 개인 정보 전송 악성 행위에 대한 유사성(SIS : Similarity of collecting Sensitvie Information), 데이터 변환 악성 행위에 대한 유사성(CDS : Similarity of Converting Data)을 나타낸다.

각 유사성에 적용되는 가중치는 달리 적용될 수 있다. 예를 들어 SS는 0.33, CS는 0.33, SIS는 0.21, CDS는 0.13으로 결정될 수 있다. 이와 같이 SS와 CS는 SIS와 CDS의 가중치에 비해서 크다. 특히 CDS의 가중치가 가장 낮는데 그 이유는 다음과 같다. 먼저 개인 정보 전송을 위하여 통신하는 과정에서 에러 발생시에 전송을 위한 자료 변환이 불필요하기에 다른 유사성 요소에 비해 가중치를 낮게 두었고 둘째 일반적으로 정상 파일에 대해서 서버간 통신이 필요할 때 인코딩 또는 암호화가 적용되는 경우가 많아 자료 변환 만으로 악성 행위로 보기 힘들기 때문이다.

표 2

행위 요소	목 표	유사성 기준
과금 문자	과 금	Binary (0 or 1)
과금 전화	과 금	Binary (0 or 1)
개인 정보 전송	시스템 정보, 개인 정보	Jacard index (0, 1)
데이터 변환	목적지 URL	Longest prefix matching + Levenshtein distance (0, 1)
	암호화 모드 (DES, AES, Blowfish)	Binary (0 or 1)
	인코딩 모드 (Gzip or not)	Binary (0 or 1)

표 2는 각 요소별 유사성 측정 방법을 나타낸다. 과금 번호로 SMS을 보내는 SS와 과금 번호로 전화를 거는 CS의 유사성 점수는 해당 하드웨어 자원에 접근할 수 있느냐 없느냐를 바이너리로 비교한다. 악성 행위의 속성 등에 포함되는 전화번호와 같은 문자열은 한 글자라도 다르면 다른 전화번호가 되기에 부분 매칭은 유사성 비교측면에서 유의한 의미를 제공하지 못한다. 따라서 서브-카테고리별 대표 프로파일과의 비교에서 악성 행위의 속성에 기초하여 동일한 행위를 한다면 1 그렇지 않다면 0이 부여된다.

민감한 데이터를 전송하는 SIS는 자카드 인덱스 방법을 이용하여 유사성 점수를 계산한다. CDS는 악성 행위의 속성에 포함되는 목적 URL 주소, 암호화 모드와 인코딩 모드에 대한 유사성을 각각 구하여 더한 평균값으로 유사성 점수를 계산한다. 목적 URL 주소의 유사성 점수는 Longest Prefix Matching 방법을 적용한 후에 불일치되는 부분에 대해서는 레빈슈타인 거리(Levenshtein distance) 방법을 적용한다. 암호화 모드 및 인코딩 모드에 대해서는 바이너리 비교를 통해 동일한 행위를 한다면 1 아니라면 0을 부여한다.

이와 같이 유사도 매칭 모듈(247)은 서브-카테고리별 유사도(S) 점수를 계산하고 서브-카테고리별 유사도 점수를 임계치와 비교한다. 유사도 매칭 모듈(247)은 서브-카테고리별 유사도 점수가 임계치를 초과하고 가장 높은

유사도 점수를 가지는 서브-카테고리를 분석 대상인 앱의 서브-카테고리로 결정할 수 있다. 임계치는 예를 들어 0.85 등이 될 수 있다. 만일 임계치를 초과하는 서브-카테고리가 존재하지 않는 경우에 유사도 매칭 모듈(247)은 새로운 서브-카테고리를 데이터베이스(220)에 생성하고 해당 앱의 행위 프로파일을 새로운 서브-카테고리에 저장한다. 서브-카테고리의 결정에 따라 유사도 매칭 모듈(247)은 합집합 또는 교집합 방식에 따라 대표 프로파일을 업데이트한다.

- [0069] 이후 유사도 매칭 모듈(247)은 분석 결과를 클라이언트(100)에게 송수신부(210)를 통해 전송할 수 있다.
- [0070] 본 발명에 따른 앱 분석부(240)를 구현하여 실험한 결과, 악성 앱의 서브-카테고리(예를 들어 악성 앱 진단명) 간 유사도 점수는 최대 0.8 까지 인 것으로 나타났고 악성 앱의 서브-카테고리와 정상 앱 간의 유사도 점수는 0.1 정도인 것으로 나타났다. 또한 악성 앱의 유사도 비교에 따른 분류에서 정확도가 99% 이상임이 확인되었고 정상 앱과 악성 앱의 분류 정확도가 98 % 이상임이 나타났고 1MB 앱을 58초 안에 분류 가능하였다. 실험에 이용되고 서브-카테고리일 수 있는 진단명은 예를 들어 AdWo, AirPush, Boxer, FakeBattScar 등이 있다.
- [0071] 이와 같이 행위 프로파일 기법을 악성 앱 탐지 및 분류에 적용하여 악성코드 샘플들에 대해서 신속하고 효율적인 대처가 가능하고 행위 프로파일링 기법을 활용하여 다양한 기존 문제점을 해소하면서 신종 악성 앱을 탐지하고 및 분류할 수 있도록 한다.
- [0072] 도 3은 악성 앱 탐지 및 분류를 위한 예시적인 흐름도를 도시한 도면이다.
- [0073] 이미 도 2를 통해 탐지 및 분류를 위한 모듈 및 제어를 상세히 살펴보았으므로 여기서는 도 2의 설명과 중복되는 부분은 생략하고 흐름을 중심으로 간단히 살펴보도록 한다. 도 3의 흐름도는 악성 앱 분류 장치(200)의 하드웨어 상에서 수행되며 바람직하게는 하드디스크 등에 저장되어 있는 프로그램을 프로세서상에서 실행하여 수행된다.
- [0074] 먼저 악성 앱 분류 장치(200)는 클라이언트(100)로부터 앱 정보를 수신(S101)한다.
- [0075] 앱 정보의 수신 후에 앱 정보에 대응하는 앱이 데이터베이스(220)에 존재하는 지를 결정(S103)한다. 존재 여부의 결정은 앱 정보에 포함된 해쉬값과 데이터베이스(220)에 포함된 해쉬값의 비교로 결정된다.
- [0076] 만일 이미 존재하여 데이터베이스(220)에 서브-카테고리로 분류되어 있는 경우에 악성 앱 분류 장치(200)는 데이터베이스(220)로부터 분석 결과를 생성(S117)한다.
- [0077] 데이터베이스(220)에 존재하지 않는 경우, 악성 앱 분류 장치(200)는 앱 정보에 대응하는 앱의 악성 행위를 식별할 수 있도록 하는 데이터인 통합 로그 데이터를 생성(S105)한다. 악성 앱 분류 장치(200)는 앱을 크롤러(250) 등을 통해 또는 클라이언트(100)로부터 수신하여 앱을 에뮬레이터를 통해 실행한다. 에뮬레이터에서의 실행에 따라 악성 앱 분류 장치(200)는 사용자 레벨에서의 사용자 로그 데이터와 LKM에 의해 커널 레벨에서 수집되는 시스템 로그 데이터를 통합하여 통합 로그 데이터를 생성한다. 통합 로그 데이터는 적어도 시스템 콜과 시스템 콜에서 이용되는 매개변수를 포함한다.
- [0078] 이후 악성 앱 분류 장치(200)는 앱 분석에 기초한 데이터인 통합 로그 데이터로부터 앱의 악성 행위를 식별할 수 있는 행위 프로파일을 생성(S107)하고 행위 프로파일로부터 분석되는 악성 행위 패턴에 따라 앱을 카테고리 분류(S109)한다. 이에 따라 분석 대상인 앱은 정상 앱인지 또는 악성 앱인지를 탐지할 수 있고 앱의 대분류를 식별할 수 있다.
- [0079] 분류된 카테고리 내에서 악성 앱 분류 장치(200)는 서브-카테고리별 대표 프로파일과의 유사도 비교로 분류된 앱의 서브-카테고리를 결정(S111)한다. 이때 악성 앱 분류 장치(200)는 임계치를 이용하여 특정 서브-카테고리를 결정하거나 새로운 서브-카테고리를 생성할 수 있다.
- [0080] 서브-카테고리의 결정에 따라 악성 앱 분류 장치(200)는 결정된 서브-카테고리의 대표 프로파일을 업데이트(S113)한다. 이러한 업데이트는 예를 들어 교집합 방식이거나 합집합 방식일 수 있다.
- [0081] 이후 악성 앱 분류 장치(200)는 서브-카테고리의 결정 및 대표 프로파일의 업데이트를 반영하도록 데이터베이스(220)를 업데이트하고 분석 대상인 앱에 대한 분석 결과를 생성(S115)한다.
- [0082] 분석 결과의 생성에 따라 악성 앱 분류 장치(200)는 분석 결과를 클라이언트(100)로 송수신부(210)를 경유하여 전송(S119)한다. 분석 결과를 수신한 클라이언트(100)는 분석 결과를 화면 등에 표시하고 사용자에게 알릴 수

있다.

[0083]

이상에서 설명한 본 발명은, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에게 있어 본 발명의 기술적 사상을 벗어나지 않는 범위 내에서 여러 가지 치환, 변형 및 변경이 가능하므로 전술한 실시 예 및 첨부된 도면에 의해 한정되는 것이 아니다.

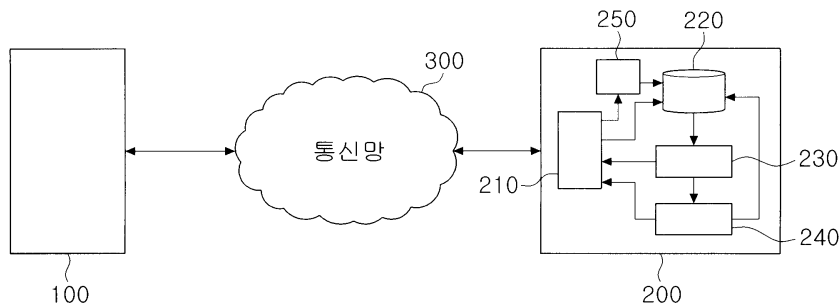
부호의 설명

[0084]

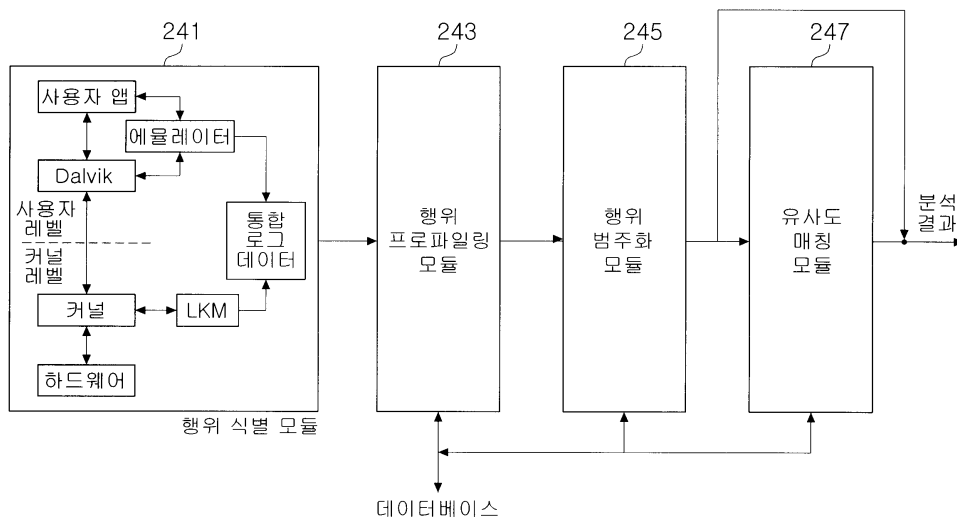
100 : 클라이언트 200 : 악성 앱 분류 장치
 210 : 송수신부 220 : 데이터베이스
 230 : 앱 확인부 240 : 앱 분석부
 241 : 행위 식별 모듈 243 : 행위 프로파일링 모듈
 245 : 행위 범주화 모듈 247 : 유사도 매칭 모듈
 250 : 크롤러
 300 : 통신망

도면

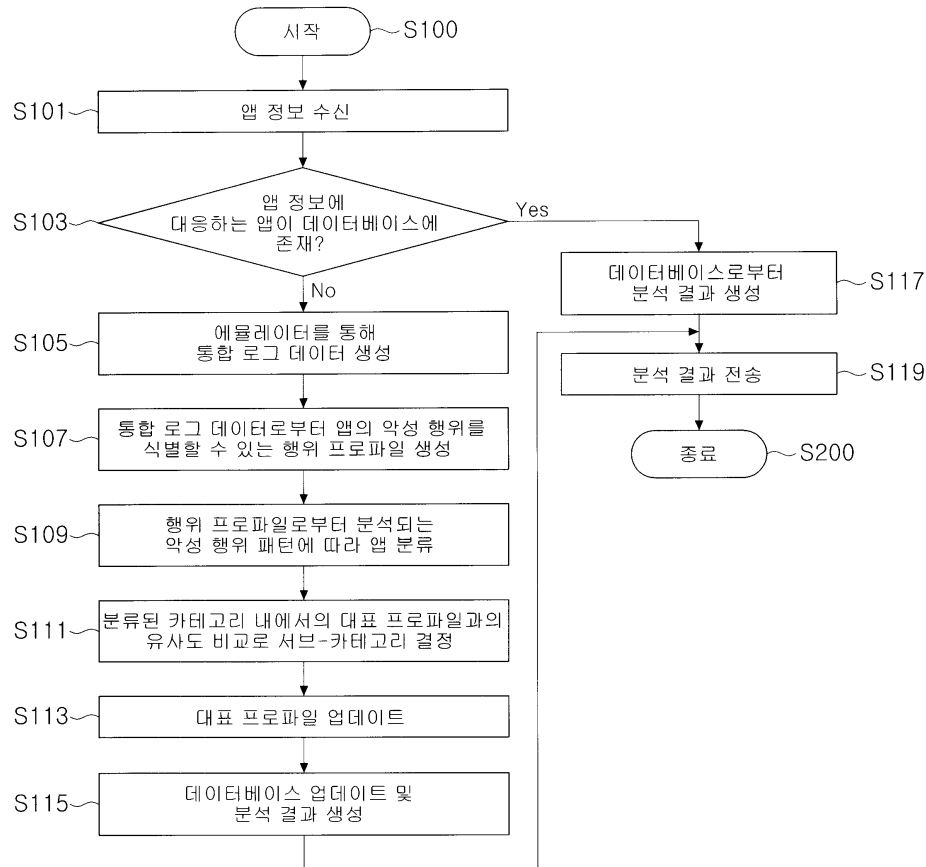
도면1



도면2



도면3



도면4

```

1  ==Parser Start==
2  =====Visualization=====
3
4  SHA : 444938eb2178c770d507695054ae4c5810bf3b28543b926be2cba32a347289fe Hash digest(SHA256)
5  =====
6
7  [Network] Object
8  =====
9
10 =Sending Data=
11 ['<string name="imei">357242043237517</string>', '<string name="imsi">310005123456789</string>',
12 '<string name="model">generic</string>', '<string name="manufacturer">unknown</string>',
13 '<string name="version_sdk">10</string>', '<string name="version_release">2.3.4</string>',
14 '<string name="cpu">armeabi</string>', '<string name="networktype">internet</string>',
15 '<string name="device">generic</string>', '<string name="simnum">8901410321118510720</string>',
16 '<int name="config_next" value="60" />', '<int name="get_config_limit" value="60" />',
17 '<boolean name="gotList" value="false" />', '<boolean name="gotAlert" value="false" />',
18 '<string name="package_man">1</string>', '<string name="cpuid">357242043237517</string>',
19 '<string name="telnum">15555215554</string>', '<long name="firstRunTime" value="1381827671388" />',
20 '<long name="getConfigTime" value="1381827671568" />', '<long name="getAlertTime" value="1381827671735" />',
21 '<boolean name="gotList" value="false", <long name="getSoftListTime" value="1381827671890" />',
22 '<long name="getUpListTime" value="1381827672060" />', '<long name="getConfigTime" value="1381827672949" />',
23 '<int name="ChainFace_SETTINGFLAG" value="1" />', '<int name="ChainFace_SETTINGLASTLEVEL" value="1" />']
24
25 =====
26
27 =====Visualization End=====
    
```