정 보 보 호 학 회 지
제25권 제6호, 2015. 12

안드로이드 모바일 악성앱 동적분석 회피기술 동향

김 미 주*, 신 영 상*, 이 태 진*, 염 흥 열**

요 약

스마트폰 사용이 대중화됨에 따라 스마트폰 사용인구 증가와 함께 우리의 일상생활과 밀접한 관계를 가지며 영향력을 넓혀가고 있는 가운데, 악성앱을 이용해 개인정보 유출, 불법 과금 유발, 스팸 발송 등 스마트폰 사용자에 피해를 입히며 사회적인 문제를 유발하는 보안 위협의 출현 또한 지속적으로 증가하고 있다. 이러한 문제를 해결하기 위해 전 세계 보안업체, 연구소, 학계 등에서는 스마트폰 악성앱을 탐지하고 대응하기 위한 기술을 연구개발하고, 앱 마켓에서는 악성앱을 탐지하기 위한 분석 시스템을 도입하는 등 다양한 활동이 진행되고 있다. 하지만 악성앱 또한 기존의 탐지 및 대응 기술을 우회하는 등 생존율을 높이기 위한 방향으로 점차 지능화·정교화되는 양상을 보이고 있다. 최근 이러한 특징은 앱 마켓 등에서 도입하고 있는 대량의 앱에 대한 자동화된 런타임 분석을 수행하는 동적분석 시스템/서비스를 대상으로 많이 발생되고 있는데, 동적분석의 환경적, 시간적 제약 등을 이용하여 분석기술을 회피하는 기법을 주로 사용하고 있다. 이와 관련하여 본 논문에서는 기존의 동적분석 기술을 우회하는 악성앱 분석회피 행위 유형을 분류하고, 이와 관련된 연구 동향에 대한 정보를 제공하고자 한다.

Ⅰ. 서 론

전 세계적으로 스마트폰 시장이 포화상태에 접어들었다고 판단될 만큼 스마트폰은 우리의 일상생활과 밀접한 관계를 가지며 발전해왔다. 스마트폰을 가지고 행선지에 가기 위한 최적의 길을 찾고, 운동량을 체크하고, 금융거래를 하고, IoT(Internet of Things)를 비롯한 다양한 IT 융합 기술들과 결합되어 새로운 부가가치 서비스를 창출하는 수단이 되기도 하는 등 우리 생활패턴에 큰 변화를 가져다주었다.

스마트폰을 사용함으로써 사람들은 생활의 편리함을 누리게 되었지만, 개인정보, 계좌정보 등 민감 정보의 유출, 문자메시지 탈취를 통한 사생활 침해, 악성코드 감염, 소액결제 등 과금 유발, 불법적인 권한 획득을 통한 단말 제어, 스미싱 등 다양한 형태의 보안 위협에 노출되는 역기능도 경험하게 되었다. 이러한 스마트폰 보안 위협은 피해의 규모 및 정도가 점점 심각해지고 있어 사회적인 문제가 되고 있다. 특히 스마트폰 보안 위협은 안드로이드 단말을 대상으로 많이 발생되고 있는데 이러한 현상은 안드로이드의 개방성 및 높은 시장 점유율을 이유로 들 수 있다. IDC에서 분석한 2015년 2분기 스마트폰 OS 시장 점유율[1]에 따르면 안드로이드가 82.8%로 가장 점유율이 높은 것으로 나타났으며, 2014년 10월 공개된 카스퍼스키와 인터폴의 합동 보고서[2]에 따르면 98.05%의 모바일 악성앱이 안드로이드 기반의 스마트폰 사용자를 대상으로 하고 있음을 알 수 있다.

안드로이드를 비롯한 스마트폰 보안 위협을 최소화시키기 위해 전 세계 보안업체, 연구소, 학계 등에서는 다양한 대응기술에 대한 연구개발을 하고 있으며, 앱 마켓에서는 악성앱을 탐지하기 위한 분석 시스템을 도입하는 등 다양한 활동이 진행되고 있다.

하지만 PC 기반의 악성코드에서 휴먼 인터랙션 탐지, 설정 확인, 가상 환경 확인, VM(Vurtual Machine) 웨어 설정 확인 등을 통해 샌드박스 기반 행위분석을 우회하는 기술들[3]이 나타났듯이, 모바일 기반의 악성앱도 기존의 탐지 및 대응 기술을 우회하는 방식으로 생존율을 높이는 자가보호 기술을 탑재하는 등 점차 지

이 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.R0132-15-1004, 모바일 결제사기 공격 역추적 및 피해 방지를 위한 프로파일링 기반 통합대응 기술 개발)
* 한국인터넷진흥원 정보보호R&D기술공유센터 사이버보안기술개발팀 ({mijoo.kim, ysshin, tjlee}@kisa.or.kr)
** 순천향대학교 정보보호학과 (hyyoum@sch.ac.kr)

능화·정교화되는 양상을 보이고 있다.

NTT 그룹에서 발표한 2014 글로벌 위협 인텔리전스 리포트[4]에 따르면, 허니팟을 통해 수집된 악성코드를 대상으로 샌드박스 기반의 안티바이러스로 분석하면 탐지율이 거의 절반 수준으로 줄어드는 것을 알 수 있다. 모바일 기반 악성앱을 대상으로 직접적인 통계를 비교분석한 공식자료는 아직 발표되고 있지 않지만, 분석회피를 통한 자가보호 기술 등 지능화되는 양상을 보았을 때 PC 기반의 악성코드와 유사할 것으로 추측할 수 있다.

모바일 악성앱에 대한 분석회피 기법은 앱 마켓 등에서 도입하고 있는 대량의 앱에 대한 자동화된 런타임 분석을 수행하는 동적분석 시스템/서비스를 대상으로 많이 발생되는데, 주로 동적분석의 환경적, 시간적 제약 등을 이용해 분석회피 행위가 이뤄진다. 2012년 2월 구글은 자사에서 운영 중인 안드로이드 마켓에 악성앱 분석 시스템인 '바운서'을 발표하였으며[5], 바운서의 도입으로 40%의 악성이 감소하였다고 밝혔다. 바운서는 안드로이드 마켓에 등록되는 앱에 대하여 구글 인프라 기반의 에뮬레이터에서 동적분석을 수행하며, 5분 동안 앱을 실행해봄으로써 해당 앱이 악성행위를 하는지 여부를 확인하고, 외부 네트워크 접속을 허용하는 특징을 가진다. 하지만 그 후 바운서의 앱 실행환경 탐지를 통한 분석회피 등 구글 바운서를 우회하는 다양한 기술들이 발표되고 있다. 또한, 안드로이드 기반 모바일 악성앱 탐지 및 분석을 위해 다양한 동적분석 도구, 서비스들이 개발되었지만 바운서와 같이 우회기술들을 통해 무력화될 수 있음이 논문 등을 통해 발표되고 있다.

이에 본 논문에서는 기존의 모바일 악성앱 동적분석 회피 유형을 분류하고, 각 분석회피 기술에 대한 연구 동향에 대한 정보를 제공하고자 한다.

Ⅱ. 모바일 악성앱 동적분석 회피 유형

모바일 악성앱 동적분석 회피기술에 대한 연구 동향 및 사례들을 살펴보면 [표 1]과 같이 크게 3가지 유형으로 분류할 수 있다.

첫 번째 유형은 앱 실행환경을 탐지하여 실제단말이 아닌 경우 동작하지 않거나 악성행위를 하지 않도록 하

〔표 1〕 모바일 악성앱 동적분석 회피 유형 분류

유형	세부 유형		
앱 실행환경 탐지를 통한 분석회피	시스템		단말 ID, 전화번호, 빌드값, QEMU, 시스템 파일 등 앱 실행 시스템에 대한 탐지를 통해 가상환경인 경우 동작하지 않음
	안드로이드 프레임워크	센서	가속도센서, 자기장센서, 회전벡터, 자이로스코프 등 센서에 대한 리턴값 확인을 통해 가상환경인 경우 동작하지 않음
		데이터 (콘텐츠)	주소록, 통화목록, SMS 내역, 배터리 레벨, 시그널 강도 등 데이터 확인을 통해 가상환경인 경우 동작하지 않음
	네트워크		라우팅 테이블 등 네트워크 속성 확인을 통해 가상환경인 경우 동작하지 않음
사용자 인터랙션 탐지를 통한 분석회피	사용자가 정확한 버튼 터치를 해야만 악성행위 개시		
	사용자가 정보를 입력해야만 악성행위 개시		
시간차 공격을 통한 분석회피	단순지연		앱 실행 후 일정 시간 후 악성행위 개시
			특정 시점 이후 악성행위 개시
	(조건부 동작) 공격자 제어		공격자로부터 명령을 받아 악성행위 개시
			(공격자로부터)특정 키워드를 포함하는 문자 수신 시 악성행위 개시
			(공격자로부터)특정 번호로부터 통화 수신 시 악성행위 개시
	(조건부 동작) 단말 동작 탐지		사용자가 임의의 앱 실행 시 악성행위 개시
			사용자가 특정사이트에 접속 시 악성행위 개시
			네트워크 환경 변경(LTE → Wi-Fi 등) 시 악성행위 개시
			GPS 사용 시 악성행위 개시

는 경우이다. 이러한 유형의 대표적인 예로, Jon O.와 Charlie M.이 SummerCon2012에서 발표[6]한 구글 바운서 우회방법이 있다. 바운서 우회방법은 구글에서 운영 중인 안드로이드 마켓에 앱 등록 시 검증단계에서 앱이 실행되고 있는 환경 정보를 받아 코드를 수정함으로써 검증 시스템을 우회하고, 안드로이드 마켓에 악성앱이 등록될 수 있도록 한다[7]. 이러한 유형은 앱 동적분석 시스템이 주로 에뮬레이터 등 가상환경에서 이루어진다는 점에 착안하여 실제 단말과 가상 환경의 차이점을 이용해 발생한다. 세부 유형으로는 단말ID, 빌드값, 버전, QEMU(Quick Emulator) 스케줄링 등 앱이 실행되고 있는 시스템, 가속도센서, 회전벡터, 자이로스코프 등과 같은 센서, 주소록, 통화목록, SMS 내역, 배터리 레벨, 시그널 강도 등의 데이터, 라우팅 테이블 등 실제 단말과 가상 환경 간의 차이점을 이용해 가상환경인 경우 동작하지 않음으로써 분석을 회피한다.

두 번째 유형은 정확한 버튼 터치, 정보 입력 등 사용자의 지능적인 인터랙션을 탐지하여 실제단말이 아닌 경우 동작하지 않거나 악성행위를 하지 않도록 하는 경우이다. 이러한 유형의 대표적인 예로, 별자리 운세 정보를 제공하는 앱을 가장하여 정보유출을 시도하는 'Horoscope' 앱이 있다. Horoscope 앱은 사용자가 자신의 별자리 및 운세 정보 제공 기간에 대한 연속적인 버튼 터치하도록 유도하고 있다. 하지만 사용자가 두 번의 연속적인 버튼 터치를 수행하면 스마트폰 내 정보유출을 시도한다. 사용자 인터랙션을 탐지하여 분석회피를 시도하는 또 다른 유형으로는 은행, 카드사를 사칭하여 정보입력을 유도하고 해당 정보에 대한 유출을 시도하는 '보안둘리' 앱이 있다. 보안둘리 앱은 사용자에게 이름, 주민등록번호, 계좌번호, 계좌비밀번호, 이체비밀번호, 카드정보 등 개인정보 및 금융정보의 입력을 유도하고 사용자가 정보를 입력하고 확인버튼을 누르면 입력된 정보들이 유출된다.

이러한 유형의 앱들은 사용자의 지능적인 인터랙션이 발생하지 않으면 악성행위를 하지 않기 때문에 단순 구동 혹은 임의의 터치만을 수행하는 기존의 동적분석 기술로는 탐지가 불가능하다.

세 번째 유형은 앱 실행 후 바로 악성행위를 하지 않고 일정 시간이 지나거나 특정 상황에서 조건부로 동작하도록 함으로써 분석을 회피하는 경우이다. 이러한 유형은 동적분석을 위해 수행되는 앱 실행시간의 제한을 이용하여 발생하는데, 한 예로, 구글 바운서의 경우 동적분석을 위해 하나의 앱에 대해서 5분 동안 실행시켜봄으로써 악성행위를 하는지 판단한다. 이는 앱이 실행 후 5분 이상이 지난 이후부터 악성행위를 하도록 제작된 경우 동적분석을 수행하는 동안에는 악성행위를 하지 않기 때문에 정상앱으로 판단하여 마켓에 등록될 수 있다. 이러한 시간차를 이용한 분석회피 행위는 앱 구동 후 일정 시간이 지난 후 악성행위를 시작하도록 하거나 특정 시점 이후에만 악성행위를 시작하도록 하는 등 단순히 시간을 지연시키는 형태, 공격자로부터 명령을 받거나 특정 키워드가 삽입된 문자 수신시에만 악성행위를 시작하도록 하는 등 공격자 제어 기반으로 악성행위를 시작하는 형태, 그리고 사용자가 앱 구동, 바탕화면 페이지 전환, 네트워크 환경 변경, GPS 사용 등과 같은 단말 동작이 탐지되었을 경우에만 악성행위를 시작하도록 하는 형태가 있다.

다음의 Ⅲ, Ⅳ, Ⅴ절에서는 본 절에서 분류한 세 가지 유형의 모바일 악성앱 동적분석 회피 유형과 관련된 연구동향에 대해서 기술한다.

Ⅲ. 앱 실행환경 탐지를 통한 분석회피

본 절에서는 앱 실행환경 탐지를 통해 실제 단말이 아닌 경우 악성행위를 하지 않도록 함으로써 동적분석을 우회하는 유형에 대한 연구 동향에 대해서 기술한다.

3.1. 실제 단말과 가상 단말 간의 차이점 분석

Timothy V.와 Nicolas C.[8]는 실제 단말과 가상 단말 간의 행위, 성능, 하드웨어 및 소프트웨어 컴포넌트, 시스템 설계방식 상의 차이점을 분석하여 가상 단말인 동적분석 시스템을 탐지하여 분석을 회피할 수 있음을 보였다.

첫 번째로, 행위 상의 차이점에 기반하여 동적분석 시스템을 탐지하는 방법에는 안드로이드 API를 통해 에뮬레이터를 탐지하는 방법, 네트워크 에뮬레이션을 탐지하는 방법, 하위(underlying) 에뮬레이터를 탐지하는 방법 등이 있다.

안드로이드 API를 이용해 탐지하는 방법은 에뮬레이터가 API 리턴값으로 실제 단말과는 다른 특정 값을 반환하게 되는데, 이러한 차이점을 파악하여 에뮬레이터

〔표 2〕 에뮬레이터 탐지에 사용될 수 있는 API 목록

API	Value	의미
Build.ABI	armeabi	에뮬레이터의심
Build.ABI2	unknown	에뮬레이터의심
Build.BOARD	unknown	에뮬레이터
Build.BRAND	generic	에뮬레이터
Build.DEVICE	generic	에뮬레이터
Build.FINGERPRINT	generic	에뮬레이터
Build.HARDWARE	goldfish	에뮬레이터
Build.HOST	android-test	에뮬레이터의심
Build.ID	FRF91	에뮬레이터
Build.MANUFACTURER	unknown	에뮬레이터
Build.MODEL	sdk	에뮬레이터
Build.PRODUCT	sdk	에뮬레이터
Build.RADIO	unknown	에뮬레이터
Build.SERIAL	null	에뮬레이터
Build.TAGS	test-keys	에뮬레이터
Build.USER	android-build	에뮬레이터
TelephonyManager.getDeviceId()	All 0's	에뮬레이터
TelephonyManager.gerLine1 Number()	155552155xx	에뮬레이터
TelephonyManager.getSubscriberId()	310260000000000	에뮬레이터
TelephonyManager.gerVoiceMailNumber()	15552175049	에뮬레이터

탐지에 활용하는 것이다. 예를 들면, TelephonyManager.getDeviceId() API에 대해서 실제 단말이 없기 때문에 에뮬레이터는 모두 0으로 구성된 값을 반환하게 되는데 이러한 특징을 파악하여 실제 단말이 아님을 확인하여 악성행위를 하지 않음으로써 분석을 회피할 수 있다.

또한, TelephonyManager.gerLine1Number() API에 대해서 에뮬레이터 상에서는 폰번호를 에뮬레이터에서 사용하는 ADB(Android Debug Bridge)포트를 기반으로 생성하게 되는데 에뮬레이터는 5554/5555로 시작하는 TCP 포트를 디버깅 목적으로 예약해두고 1-555-521-5554 형태로 생성하여 반환하게 되는데 실제 지역번호 555값은 존재하지 않기 때문에 에뮬레이터임을 확인할 수 있다.

네트워크 에뮬레이션도 실제 단말과 차이점을 보이는데, 에뮬레이터는 네트워크 주소에 대해서 항상 10.0.2/24을 반환하며, 가상 라우터의 마지막 옥텟(octet), 호스트 루프백, DNS 리졸버(resolvers), 에뮬레이터 주소가 각각 1, 2, 3-6, 15로 고정되어 있다. 이러한 에뮬레이터의 IP 주소는 에뮬레이터 탐지에 활용되어 분석을 회피하는데 사용될 수 있다.

하위 에뮬레이터 탐지는 하드웨어 에뮬레이션을 처리하기 위해 적용되는 QEMU 등을 탐지하는 것으로, CPU 버그, 가상 주소, QEMU를 Dalvik 가상머신이 아닌 프로세스 상에 동작시키기 위한 네이티브 코드 사용 여부 확인 등을 통해 탐지할 수 있다.

두 번째로, 성능 상의 차이를 이용해 에뮬레이터를 탐지하는 방법이 있다. 하드웨어를 지원하지 않는 에뮬레이터는 태생적으로 처리 속도 상의 약점을 갖는다는 특징에 기인하여 CPU 및 그래픽 성능을 측정해본 결과, 실제 단말과 가상 단말 간에는 상당한 차이가 있는 있음을 보였다. CPU 성능 측정을 위해 AGM(Arithmetic-Geometric Mean)을 계산해본 결과, PC는 가장 빠르고, 실제 단말은 중간, 에뮬레이터는 가장 느린 수치로 계산되었다. 또한, 그래픽에 대한 FPS(Frame Per Second) 측정 결과에서도 안드로이드 버전에 따라 실제 단말과 에뮬레이터 간의 수치, 형태 등에 큰 차이가 있음을 보였다.

세 번째로, 하드웨어 및 소프트웨어 컴포넌트 상의 차이점을 이용해 에뮬레이터를 탐지하는 방법이 있다. 주로 에뮬레이터에서는 제공하지 않는 실제 단말에서의 복잡하고, 특징적인 하드웨어 및 소프트웨어 컴포넌트 식별을 통해 에뮬레이터 여부를 확인한다. 특히, 실제 단말은 무수히 많은 동작, 환경적, 위치적 상황을 감지를 하는 하드웨어 센서를 사용하고 있고, 심지어 GPS, 블루투스, NFC 칩, 공기압 센서, 습도 센서 등을 가지는데 반해 에뮬레이터는 이러한 값들에 대한 변화를 에뮬레이션하는데 한계를 가진다. 또한 하드웨어 관련 소프트웨어 컴포넌트로 카메라, 배터리 레벨 등의 확인을 통해서 에뮬레이터임을 확인할 수 있다. 하드웨어를 지원하는 소프트웨어뿐 아니라 실제 단말에는 출고와 함께 추가되는 DRM 지원, 클라우드 자원과의 인터페이

스, 구글의 다양한 지원 소프트웨어 등 다양한 소프트웨어가 추가되지만 에뮬레이터에서는 이러한 소프트웨어는 포함되어 있지 않기 때문에 이러한 차이점을 기반으로 에뮬레이터임을 탐지하여 회피를 시도할 수 있다.

마지막으로 PC 기반 악성코드와 모바일 악성앱 공격 목적의 차이에 따른 런타임 분석 시스템 설계 상의 차이점을 이용하여 탐지하는 방법이 있다.

3.2. 동적분석 시스템 탐지를 위한 정적, 동적, Hypervisor 요소 분석

Thanasis Petsas 등[9]은 동적분석 시스템을 탐지하기 위한 정적(static) 요소, 동적(dynamic) 요소, 하이퍼바이저(hypervisor) 요소를 도출하였다.

정적 요소는 실제 단말과 구분되는 가상 단말의 고정된 값으로, IMEI(International Mobile Station Equipment Identity), IMSI(International Mobile Subscriber Identity), 라우팅 테이블 등이 있다.

동적 요소는 실제 단말에는 동적으로 변하는 값이나, 가상 디바이스에서는 고정된 상태이거나 제공되지 않는 값으로, 가속도 센서, 자기장 센서, 회전 벡터, 근접 센서, 자이로스코프 등 각종 센서들이 이에 해당된다.

하이퍼바이저 요소는 VM 에뮬레이션에 대해 실제 OS와의 구성 상 차이점을 파악하여 탐지하는 방법으로 QEMU 스케줄링 및 실행을 식별하는 방법 등이 있다.

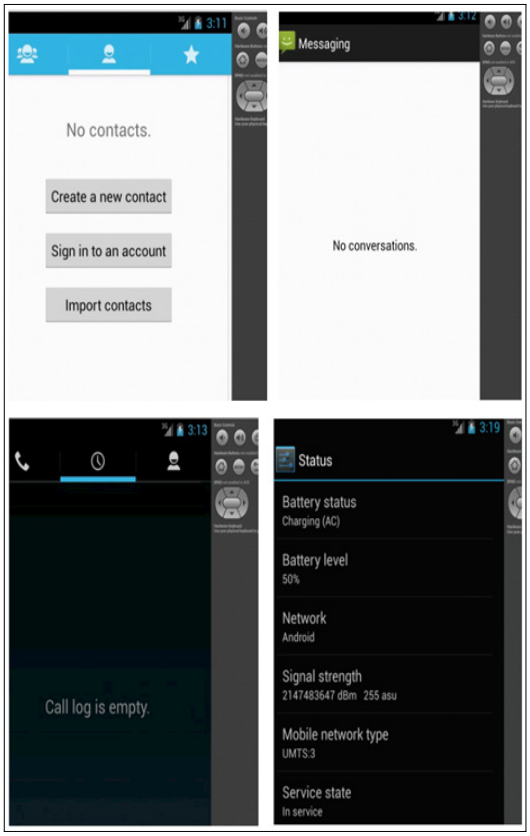
논문에서는 정적, 동적, 하이퍼바이저 요소를 이용해 탐지회피를 시도하는 10가지 종류의 악성앱을 가지고 DroidBox[10], TaintDroid[11], Andrubis[12], ApkScan[13], CopperDroid[14], Apk Analyzer[15] 등 12종의 악성앱 동적분석 도구들을 대상으로 실험해본 결과, [표 3]과 같이 IMEI와 같은 아주 간단한 정적 요소를 이용한 탐지회피 시도를 제외하고는 거의 모든 동적분석 도구들이 탐지회피 시도 행위를 감지하지 못하는 것을 확인하였다.

3.3. 안드로이드 에뮬레이터 탐지 휴리스틱 자동 생성 프레임워크(Morpheus)

Yiming Zing 등[16]은 실제 단말과 가상 단말 간의 차이점을 분석해 안드로이드 에뮬레이터를 탐지하는 휴리스틱을 자동으로 생성하는 프레임워크인 Morpheus를 제안하였다. Morpheus를 이용하여 네트워크, 파워 관리, 오디오, USB, 라디오, 소프트웨어 컴포넌트 및 설정 등과 같은 기본적인 휴리스틱부터 QEMU, Goldfish 가상 하드웨어, 블루투스, NFC, 바이브레이터 등 QEMU를 탐지하는 휴리스틱, VirtualBox, PC 하드웨어 등 VirtualBox를 탐지하는 휴리스틱에 이르기까지 10,000가지 이상의 다양한 휴리스틱을 도출함으로써, 가상화 환경에 이루어지는 악성앱 동적분석 시스템에 다양한 회피기술이 적용 가능함을 보였다.

〔표 3〕 악성앱 동적분석 회피기법에 대한 탐지 결과

	정적 요소			동적 요소					하이퍼바이저 요소	
	IMEI	하드웨어	네트워크	가속도 센서	자기장 센서	회전벡터	근접센서	자이로 스코프	QEMU 스케줄링	QEMU 실행
DroidBox	√	X	X	X	X	X	X	X	판단불가	판단불가
DroidScope	X	X	X	X	X	X	X	X	X	X
TaintDroid	X	X	X	X	X	X	X	X	판단불가	판단불가
Andrubis	√	X	X	X	X	X	X	X	X	X
SandDroid	√	X	X	X	X	X	X	X	X	X
ApkScan	√	X	X	X	X	X	X	X	판단불가	판단불가
VisualThreat	X	X	X	X	X	X	X	X	X	X
Tracedroid	X	X	X	X	X	X	X	X	X	X
CopperDroid	X	X	X	X	X	X	X	X	X	X
Apk Analyzer	√	√	√	X	X	X	X	X	판단불가	판단불가
ForeSafe	X	X	X	X	X	X	X	X	X	X
Mobile Sandbox	√	X	X	X	X	X	X	X	판단불가	판단불가



〔그림 1〕 단말 콘텐츠 확인을 통한 에뮬레이터 탐지

3.4. 시스템 속성, QEMU 파이프, 콘텐츠 확인을 통한 에뮬레이터 탐지

Tim Strazzere는 HITCON2013에서 에뮬레이터를 회피하는 다양한 방법들을 발표[17]하였는데, 시스템 속성 확인, 호스트 환경과 통신하기 위한 QEMU 파이프 확인, 그리고 연락처 정보, SMS 전송내역, 통화목록, 배터리 레벨 등 단말의 콘텐츠 확인을 통해 에뮬레이터를 탐지할 수 있음을 보였다.

Ⅳ. 사용자 인터랙션 탐지를 통한 분석회피

본 절에서는 정확한 버튼 터치, 정보 입력 등 사용자 인터랙션 탐지를 통해 동적분석을 우회하는 유형과 관련된 연구 동향에 대해서 기술한다.

해당 유형은 기존의 PC기반 악성코드에서 마우스 클릭, 대화상자에 대한 지능적인 응답 등과 같은 인간 사용자의 개입을 탐지할 때까지 잠복함으로써 샌드박스를 통한 분석을 우회하는 기법과 유사하다.

모바일에서의 인터랙션은 화면에 대한 터치, 팝업에 대한 터치, 정보 입력 등을 통해 이뤄지는데, 팝업, 버튼 터치 등의 단순한 형태의 인터랙션만을 요구하는 경우는 임의의 좌표값에 대한 이벤트를 발생하는 monkey 등을 통해 사용자 인터랙션을 만들어낼 수 있지만, 사용자의 판단에 의한 연속되고 정확한 버튼 터치, 정보 입력 등 지능적인 인터랙션을 요구하는 경우 monkey 로 대응하는데 한계가 있다.

Cong Zheng 등[18]은 정적분석을 통해 행위(Activity)와 함수(Function)들에 대한 호출 그래프(call graph)를 그린 후, 이를 기반으로 행위 흐름 제어 및 UI(User Interface) 인터랙션에 대한 시뮬레이션을 수행함으로써 UI 기반 트리거 조건을 만족시키는 자동화된 시스템을 제안하였다.

Ⅴ. 시간차 공격을 통한 분석회피

본 절에서는 앱 실행 후 바로 악성행위를 하지 않고, 악성행위 시작 시점을 임의로 지연시키거나 특정 상황에서 조건부로 동작하도록 함으로써 분석을 회피하는 유형과 관련된 연구 동향에 대해서 기술한다. 이러한 유형은 주로 앱을 단시간 구동해보는 동적분석의 시간적 제약 등을 이용함으로써 발생한다.

시간차 공격을 통해 동적분석을 회피하는 기술에는 크게 세 가지 유형이 있다.

첫 번째로, PC 악성코드에서 타임아웃 매개변수를 동적분석 실행 시간보다 긴 시간으로 설정 후 악성행위를 하는 함수를 실행을 지연시키는 휴면호출(Sleep Call) 방식 및 정해진 날짜와 시간 후에만 악성코드를 실행시켜 그 시간 전에는 샌드박스가 비정상 행위를 탐지하지 못하게 하는 타임 트리거(Time trigger, 시한폭탄) 방식을 모바일 환경에 적용해 단순히 악성행위의 시작 시점을 지연하는 형태가 있다.

두 번째로, 공격자가 악성행위 시작 시점을 제어해 동적분석이 이뤄지는 동안에는 악성행위를 하지 않도록 함으로써 분석을 회피하는 유형이다. 이러한 유형에는 공격자로부터 명령을 받아 악성행위를 시작하는 형태가 있을 수 있는데, 코드 내부에 악성행위에 대한 코드가 숨겨져 있는 형태와 다운로드를 통해 악성코드를 받아서 악성행위를 하는 형태가 있다. 최근 구글 바운서를

우회해 200만대 이상의 기기 및 100만 명 이상의 사용자가 감염된 것으로 확인된 'BrainTest' 앱의 경우에도 구글 플레이에서 검증이 수행되는 동안에는 악성행위를 하지 않다가 앱 배포 이후 공격자의 명령을 받아 악성코드가 실행되도록 제작되었다. 또한 공격자가 특정 키워드가 삽입된 문자를 보내거나, 특정 번호로 중복적인 통화 및 문자를 보냄으로써 악성행위를 개시하는 형태도 있을 수 있다.

마지막으로 단말에 대해 사전에 정의된 특정 조건을 만족하는 동작 행위가 탐지되었을 때 악성행위를 시작하도록 하는 유형이다. 이러한 유형에는 사용자의 임의의 앱 실행, 단말의 바탕화면 페이지 전환, 특정 사이트 접속, 통화 시도, 네트워크 환경 변경(LTE->Wi-Fi 등), GPS 사용 등 실제 단말에서 발생할 수 있는 다양한 단말 동작 조건에 대해서 사전에 정의하여 해당 조건을 만족하는 동작 이벤트가 발생하는 경우 탐지하여 악성행위를 시작하는 경우이다.

Ⅵ. 결 론

본 논문에서는 지능화·정교화되는 모바일 악성앱의 자가보호 기술로, 기존의 동적분석 시스템을 우회함으로써 악성앱 탐지 및 분석을 무력화시키는 다양한 동적분석 회피 유형을 기술하였다. 동적분석 회피 유형에는 앱 실행환경 및 사용자 인터랙션 탐지를 통해 분석을 회피하는 유형들과 시간차 공격을 통해 분석을 회피하는 유형이 존재하였고, 각 유형별 관련 연구 동향을 소개함으로써 현 악성앱 동적분석 기술의 한계를 확인하였다.

본 논문에서 기술한 안드로이드 모바일 악성앱 동적분석 회피기술 동향을 기반으로 지능화되는 악성앱 기술 동향 및 동적분석 기술의 한계에 대한 연구를 진행하는데 유익하게 활용되어, 기존의 동적분석의 한계점들을 극복한 고도화된 분석기술이 개발되기를 기대해 본다.

참 고 문 헌

[1] IDC, http://www.idc.com/prodserv/smartphone-market-share.jsp.

[2] Kaspersky, "MOBILE CYBER THREATS", Kaspersky Lab&INTETPOL Joint Report, 2014.

[3] FireEye, "파일 기반의 샌드박스를 쉽게 회피하는 악성코드", 2013.

[4] NTT Group, "2014 Global Threat Intelligence Report", 2014.

[5] Google Mobile Blog, "Android and Security", 2012.

[6] J. Oberheide, C. Miller, "Dissection the Android Bouncer", SummerCon, 2012.

[7] 배한철, "안드로이드 모바일 악성 앱 분석 방법에 대한 연구", Internet & Security Focus 2013년 6월호, pp. 59-84, 2013.

[8] T. Vidas, N. Christin, "Evading Android Runtime Analysis via Sandbox Detection", ASIA CCS'14, 2014.

[9] T. Petsas, G. Voyatzis, E. Athanasopoulos, M. Polychronakis, S. Ioannidis, "Rage Against the Virtual Machine: Hindering Dynamic Analysis of Android Malware", EuroSec'14, 2014.

[10] DroidBox, https://github.com/pjlantz/droidbox.

[11] TaintDroid, http://appanalysis.org/.

[12] Andrubis, http://anubis.iseclab.org/.

[13] ApkScan, https://apkscan.nviso.be/.

[14] CopperDroid, http://copperdroid.isg.rhul.ac.uk/copperdroid/.

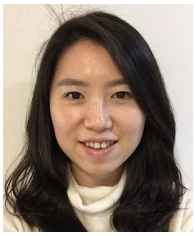
[15] Apk Analyzer, https://www.apk-analyzer.net/.

[16] Y. Jing, Z. Zhao, G. Ahn, H. Hu, "Morpheus: Automatically Generating Heuristics to Detect Android Emulators", ACSAC'14, 2014.

[17] T. Strazzere, "DEX EDUCATION 201 ANTI-EMULATION", HITCON2013, 2013.

[18] C. Zhengm, S. Zhu, S. Dai, G. Gu, X. Gong, X. Han, W. Zou, "SmartDroid: an Automatic System for Revealing UI-based Trigger Conditions in Android Applications", SPSM'12, "2012.

〈저 자 소 개〉



김 미 주 (Mijoo Kim)
정회원

2006년 : 순천향대학교 정보보호학과 학사 졸업
2008년 : 순천향대학교 정보보호학과 석사 졸업
2008년~현재 : 순천향대학교 정보보호학과 박사과정
2008년 4월~현재 : 한국인터넷진흥원 선임연구원
관심분야 : 인터넷 보안, 사이버보안, 모바일·스마트폰 보안, 스마트그리드 보안

신 영 상 (Young Sang Shin)
정회원

1998년 : 부산대학교 컴퓨터공학 학사 졸업
2000년 : 부산대학교 컴퓨터공학 석사 졸업
2004년 : University of Wisconsin - Madison, Computer Science 석사 졸업
2011년 : Indiana University - Bloomington, Computer Science 박사 졸업
2011년 12월~현재 : 한국인터넷진흥원 책임연구원
관심분야 : 네트워크/시스템 침입방지, 클라우드 보안, 모바일 보안, 웹 보안, 금융 보안

이 태 진 (Tae Jin Lee)
정회원

2003년 : POSTECH 컴퓨터공학 학사 졸업
2008년 : 연세대학교 컴퓨터공학 석사 졸업
2003년 1월~현재 : 한국인터넷진흥원 팀장
관심분야 : 악성코드, 네트워크 보안, 시스템 보안

염 흥 열 (Heung Youl Youm)
종신회원

1981년 : 한양대학교 전자공학과 학사 졸업
1983년 : 한양대학교 대학원 전자공학과 석사 졸업
1990년 : 한양대학교 대학원 전자공학과 박사 졸업
1982년 12월~1990년 9월 : 한국전자통신연구소 선임연구원
1990년 9월~현재 : 순천향대학교 공과대학 정보보호학과 정교수
1997년 3월~2000년 3월 : 순천향대학교 산업기술연구소 소장
2000년 4월~2006년 2월 : 순천향대학교 산학연컨소시엄센터 소장
1997년 3월~현재 : 한국정보보호학회 총무이사, 학술이사, 교육이사, 논문지편집위원회 위원장, 수석부회장(역), 학회장(역), 명예회장(현)
2005년~2008년 : ITU-T SG17 Q.9 Rapporteur(역)
2006년 11월~2009년 2월 : (구) 정통부 정보보호 PM/, 정보통신연구진흥원 정보보호전문위원
2011년 1월~12월 : 한국정보보호학회 회장(역)
2008년 7월~현재 : 방송통신위원회 자체평가위원회
2008년 7월~2013년 2월 : 행정안전부 정책자문위원회
2013년 5월~현재 : 미래창조과학부 자체평가위원회
2009년 5월~현재 : 국정원 암호검증위원회 위원
2009년~현재 : ITU-T SG17 부의장/SG17 WP3 의장
2012년 6월~2015년 3월 : 정보보안산업표준 포럼 의장
관심분야 : 인터넷 보안, USN 보안, IPTV 보안, 홈네트워크 보안, 암호 프로토콜