ETRI

2013
Electronics and
Telecommunications
Trends

사이버 보안 기술 특집

안드로이드 기반 모바일 악성코드 탐지 기술 동향

A Survey of Mobile Malware Detection Techniques

Ⅰ. 서론
Ⅱ. 모바일 악성코드 특징
Ⅲ. 모바일 악성코드 탐지 기법
Ⅳ. 결론

문화신 (H.S. Moon) 모바일보안연구실 선임연구원
정보흥 (B.H. Jung) 모바일보안연구실 책임연구원
전용성 (Y.S. Jeon) 모바일보안연구실 책임연구원
김정녀 (J.N. Kim) 모바일보안연구실 실장

* 본 연구는 미래창조과학부의 방송통신기술개발사업의 일환으로 수행되었음(12-912-06-001).

스마트폰은 언제 어디서나 이용 가능한 소형 컴퓨터로 진화함으로써, 해커들의 주요 공격 타깃이 되고 있다. 해커는 스마트폰에 설치된 악성코드를 통해 개인정보를 탈취할 수 있을 뿐만 아니라, 휴대폰 소액 결제 및 프리미엄 SMS 서비스를 이용하여 금전적 이득을 취할 수 있다. 악성코드에 감염된 스마트폰으로부터 얻을 수 있는 이러한 이득과 함께 모바일 악성코드는 그 수가 급속히 증가하고 있다. 특히, 안드로이드 마켓의 개방성과 안드로이드 단말의 높은 시장 점유율은 악성코드의 유포를 용이하게 하며, 이러한 이유로 모바일 악성코드의 대부분은 안드로이드 단말을 공격 대상으로 삼고 있다. 본고에서는 이렇게 급속히 증가하고 있는 안드로이드 기반 모바일 악성코드의 특징을 살펴보고, 이들을 탐지하기 위하여 연구되고 있는 다양한 보안 기법들을 소개하고자 한다.

Ⅰ. 서론

1994년 출시된 사이먼 이래로 스마트폰의 성능은 비약적으로 향상되어 왔다. 스마트폰의 성능 향상은 무선 네트워크 인프라 확장과 맞물려 전화, 문자와 같은 기존 통신 서비스뿐만 아니라, 인터넷 검색, SNS, 게임, 모바일 뱅킹 및 지불 서비스 등으로 스마트폰의 이용 범위를 확장시켰다. 그러나 이러한 기술적 발전은 일반 사용자뿐만 아니라 해커들에게도 스마트폰에 대한 관심을 증가시키는 결과를 초래하였다.

스마트폰 플랫폼 중 하나인 안드로이드는 빠른 시장 확장을 위하여 '플랫폼 소스 공개' 및 '다양한 애플리케이션 유포 경로'의 특징을 갖는 개방형 구조를 선택하였으며 그 결과 50%가 넘는 점유율을 획득했다[1],[2]. 그러나 개방형 구조는 악성코드의 작성 및 유포를 쉽게 하는 특징 또한 가져, 안드로이드에게 악성코드 취약이라는 오명 역시 부여하였다. 실제 모바일 악성코드의 대부분이 안드로이드를 공격 대상으로 삼고 있으며[3] 그 수는 급속히 증가하고 있다. 일례로, 안드로이드 모바일 악성코드는 2011년 6월 400여 개에서 같은 해 12월 13,302개로 급속히 증가하였다[4].

그러나 안드로이드 기반 악성코드의 빠른 진화와 달리, 이들을 탐지하기 위한 모바일 보안 솔루션은 아직 초기 단계에 머물러 있다. 특히, 참고문헌 [5]의 실험 결과에 따르면 1,260개의 악성코드에 대한 기존 모바일 안티바이러스 제품의 탐지율은 20.2%에서 79.6% 밖에 보이지 못하고 있다. 이러한 이유로 애플리케이션 유통 과정 전반에 걸쳐 모바일 악성코드의 빠른 탐지를 위한 다양한 보안 기법들이 현재 연구되고 있다. 본고에서는 모바일 단말 보안의 현주소를 파악하고 종합적인 보안 솔루션 연구에 도움이 되고자 현재 유포되고 있는 모바일 악성코드의 특징과 함께 이들을 탐지하기 위하여 연구되고 있는 다양한 보안 기법에 대해 소개하고자 한다.

Ⅱ. 모바일 악성코드 특징

모바일 악성코드는 단말에 설치되어 개인정보 탈취, 스마트폰 원격 제어, 사용자 과금 유발 등의 행동을 수행한다. 이러한 모바일 악성코드의 행동은 해커에게 금전을 비롯한 다양한 이득을 제공한다. 일례로, 해커는 특정 번호로 문자를 보내면 보낸 사람에게 과금이 발생하고 받는 사람과 통신사에 수익이 생기는 프리미엄 SMS 서비스를 이용하게 함으로써 직접적인 금전적 이득을 획득할 수 있다. 또한, 휴대폰 인증번호를 가로채 소액 결제를 이용함으로써 직접적인 금전적 이득을 취할 수 있다. 이 외에도 스마트폰을 좀비폰으로 활용함으로써 DDoS(Distributed Denial of Service) 공격 및 악성 스팸 대량 유포 등과 같은 작업을 수행할 수 있다.

모바일 악성코드의 유포 방법은 악성코드가 생산하는 위와 같은 이득과 함께 빠른 진화를 보이고 있다. 본 장에서는 악성코드의 빠른 유포를 위해서 현재 사용되고 있는 3가지 사회공학적(social engineering) 기법에 대해 설명한다.

1. 위변조(Repackaging)

위변조는 모바일 악성코드 유포를 위하여 매우 일반적으로 사용되는 기법이다[5]. 특히 참고문헌 [5]에 따르면 분석한 1,260개의 악성코드 중 86%가 위변조 기법을 사용하고 있었다. 애플리케이션을 위변조하기 위해서 해커는 (1) 위변조할 애플리케이션을 선정하여 마켓 등을 통해 입수한 후 (2) unzip, ded, dex2jar, JD-GUI와 같은 도구를 이용하여 애플리케이션의 소스 코드를 추출/분석한다. 이후 (3)기존 애플리케이션의 특정 부분에 원하는 악성코드를 삽입하거나 특정 보안 기능을 무력화하는 작업을 수행하며, (4) 변경된 소스 코드를 서명하여 위변조 된 새로운 애플리케이션을 생성한다. 특히, 해커들은 악성코드의 빠른 유포를 위해 앵그

리버드, 인스타그램과 같은 높은 인기를 가지는 애플리케이션이나 구글 월렛과 같은 신뢰성 있는 애플리케이션을 위변조 대상으로 설정하고 있다. Arxan 보고 자료에 따르면, 안드로이드 마켓 상위 100개의 유료 앱에 대한 위변조 애플리케이션이 모두 존재하는 것으로 밝혀졌으며[6], 심지어 올해 1월에는 구글 플레이에 뱅킹 앱을 위변조한 애플리케이션이 다수 등록된 것으로 보고되고 있다[7].

2. 애플리케이션 업데이트

해커의 목적은 악성코드임이 탐지되지 않은 채로 많은 단말을 감염시키는 것일 것이다. 악성코드 탐지를 회피하면서 악성코드를 유포하기 위해 주로 사용되는 기법이 바로 애플리케이션 업데이트이다. 애플리케이션 업데이트는 악의적인 행동을 수행하는 로직을 추후 애플리케이션 업데이트를 통해 단말로 배포하는 방식이다[5]. 즉, 사용자에 의해 처음 설치되는 애플리케이션에는 보안 솔루션의 탐지 대상이 되는 악성코드가 포함되지 않으며, 단지 추후 악성코드를 획득하여 설치하기 위해 사용할 업데이트 로직만을 갖는다. 현재 악성코드 유포를 위해 사용되는 업데이트 방식으로는 설치된 애플리케이션의 업데이트 버전을 다운받는 형식과 애플리케이션의 특정 부분만을 업데이트하는 방식이 있다. 전자의 경우 업데이트에 대한 사용자의 승인이 필요하며, 업데이트 버전은 기존 애플리케이션의 resource나 asset 파일로 저장된다[5]. 반면, 특정 부분만 업데이트하는 방식은 사용자의 승인 절차 없이 조용히 진행될 수 있다[5].

3. 피싱(Phishing)

PC 기반 악성코드와 마찬가지로 모바일 단말 악성코드 역시 피싱을 주요 유포 수단으로 사용하고 있다. 피싱이란 신뢰 가능한 송신자로부터 보내진 메시지로 가장하여 수신자를 현혹한 후, '수신자 정보 탈취' 및 '수신 단말로의 악성코드 설치' 등을 유도하는 행동이다. 현재 피싱을 위해 이용되는 수단으로는 이메일, 카카오톡과 같은 메신저, SMS, 애플리케이션의 광고 링크[5], QR 코드[5] 등이 있다. 특히 모바일 단말의 경우 스미싱이라고 불리는 SMS를 이용한 피싱이 기승을 부리고 있으며, 휴대폰 소액 결제와 연동하여 감염된 사용자의 금전적 손실을 야기하고 있다.

Ⅲ. 모바일 악성코드 탐지 기법

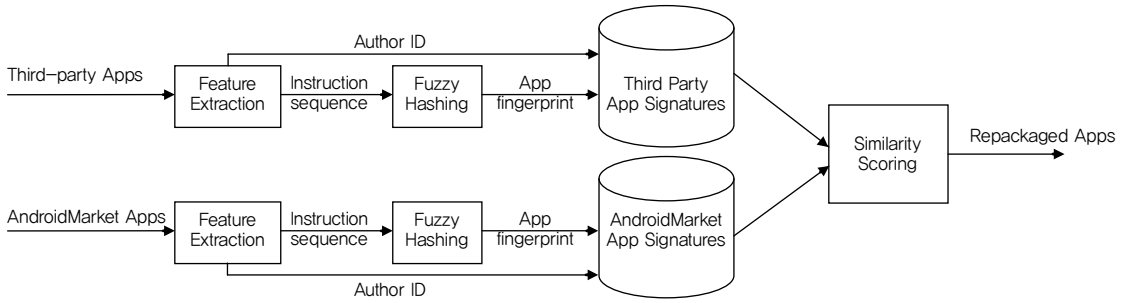
앞 장에서 설명한 바와 같이 최근 악성코드 유포 수단으로 애플리케이션 위변조가 많이 활용되고 있다는 것이 밝혀졌다. 이에 따라 기존의 애플리케이션 소스 코드 및 행동 분석에 기반한 탐지 기법과는 별도로 애플리케이션 위변조에 초점을 둔 모바일 디바이스 보안 기법들이 연구되기 시작하였다. 이러한 이유로 본 장에서는 위변조 탐지 기법들을 별도로 설명한 후, 기존의 악성코드 탐지 기법들을 설명한다.

1. 애플리케이션 위변조 탐지 기법

위변조 애플리케이션 탐지는 크게 (1) 애플리케이션 유사성에 기반한 탐지 방법과 (2) 애플리케이션 무결성을 검증하는 방법의 2가지 접근 방법이 존재한다. 또한, 비록 위변조 여부를 탐지하는 것은 아니지만 애플리케이션을 위변조하기 위하여 필수적으로 수행되는 소스 코드 추출 및 분석을 방지하기 위한 (3) 코드 난독화 기술 역시 연구되고 있다.

가. 유사성 기반 위변조 애플리케이션 탐지

위변조 애플리케이션은 기존 애플리케이션을 바탕으로 특정 소스 코드들이 추가, 삭제, 수정되어 생성된 애



(그림 1) DroidMOSS 시스템 구조[8]

플리케이션으로 볼 수 있다. 따라서 기존 애플리케이션과 위변조 애플리케이션 사이에는 일정 부분 소스 코드를 공유하게 된다. 유사성에 기반한 위변조 애플리케이션 탐지 방법은 이러한 사실에 기초하여 작성자가 다르나 소스 코드가 유사한 애플리케이션들을 찾아내는 방법을 연구한다. 특히 위변조된 애플리케이션은 구글 플레이와 같은 공식 마켓(official market) 보다는 마켓에 등록된 애플리케이션에 대한 검증 작업을 수행하지 않는 제 3자 마켓(third-party market)을 통하여 유통될 확률이 높다. 이러한 이유로 제 3자 마켓에 등록된 애플리케이션들과 공식 마켓에 등록된 애플리케이션 사이에 유사성 측정을 통해 위변조된 애플리케이션을 찾아내는 시스템(DroidMOSS[8])이 최근 제안되었다.

DroidMOSS는 위변조된 애플리케이션 작성 시 해커에 의해 수행될 수 있는 회피 기술(예, 소스 코드 내 함수나 변수명 변경 등)을 고려하여 각 애플리케이션의 유사성 비교 대상으로 소스 코드가 아니라 opcode(Dalvik bytecode 상에서 수행될 명령어를 나타냄)들만을 추출해서 사용하는 특징을 갖는다. 또한, 새롭게 마켓에 등록되는 애플리케이션들의 추가 분석을 고려하여(scalability) 조사 대상인 각 애플리케이션의 fingerprint를 생성 저장하는 단계와, fingerprint들의 유사성을 측정하여 위변조 애플리케이션을 탐지하는 단계가 분리된 구조를 채택하였다. 이 때, 각 애플리케이션의 fingerprint는 유사성 측정 시간 단축과 정확도 모두를 만족시키기 위하여 fuzzy hashing 알고리즘에 의해서 생성된다. (그림 1)은 이와 같은 특징을 갖는 DroidMOSS의 구조를 설명한다. DroidMOSS는 제 3자 마켓으로부터 획득한 1,200개의 애플리케이션과 공식 마켓에서 입수한 68,187개의 애플리케이션의 비교를 통해 10개의 false positive를 포함한 118개의 애플리케이션을 위변조로 탐지하였다. 즉 10% 이내의 false positive rate를 보이고 있다.

나. 애플리케이션 무결성 검증 방법

애플리케이션 무결성 검증 방법은 애플리케이션 실행 시점에 애플리케이션의 위변조 여부를 탐지하는 방법이다. 무결성 검증 방법이 적용된 안드로이드 애플리케이션은 애플리케이션 내부에 위변조 여부를 탐지하기 위한 특별한 코드를 포함한다. 상기 코드는 실행 시점에 애플리케이션의 해시값 을 측정하고 외부 서버에 존재하는 올바른 해시값과 비교함으로써 위변조 여부를 탐지하며, 위변조로 탐지된 경우 애플리케이션의 동작을 중지 또는 제한한다. 현재 무결성 검증 방식은 모바일 뱅킹 애플리케이션에 적용되어 사용되고 있으며 보험 증권과 같은 타 금융 애플리케이션 및 게임 애플리케이션에도 적용되고 있다[9]. 이러한 무결성 검증 방법은 특히 서버를 통해 서비스가 제공되는 애플리케이션에 적합하다. 그러나, 무결성 검증 로직이 애플리케이션 내부에 포함되어 있어 해당 로직 역시 위변조될 가능성이 존재한다. 이러한 문제점은 상기 로직을 코드 난독화 기

술을 통해 보호함으로써 완화될 수 있다. 또한 해당 위변조 탐지 로직을 애플리케이션이 아닌 안드로이드 플랫폼에 포함시키는 방법도 해결 방안으로 고려될 수 있다.

다. 난독화 기술

난독화 기술은 애플리케이션 소스 코드가 역공학(reverse engineering)을 통해 분석되는 것을 어렵게 하기 위해 사용되는 기법으로 크게 구획 난독화(layout obfuscation), 제어 흐름 난독화(control flow obfuscation), 데이터 난독화(data obfuscation), 예방 난독화(preventive obfuscation)로 구분된다[10],[11].

구획 난독화는 소스 코드 분석 시 도움될 수 있는 정보를 제거하는 기법으로 함수, 변수 등의 식별자를 의미 없는 값으로 변경하거나(symbol renaming) 디버깅 정보를 제거하는 방식이 존재한다[10].

제어 흐름 난독화는 수행되지 않는 불필요한 코드를 삽입하는 것과 같은 방식으로 프로그램의 제어 흐름을 변형하여 문맥 판단을 어렵게 하는 기법이다[10].

데이터 난독화는 애플리케이션 내에 포함된 데이터를 유추하기 어렵게 하는 기법이다. 상기 기법의 일례로 기존 애플리케이션에 포함된 텍스트를 암호화하는 방식이 존재한다[10].

마지막으로 예방 난독화는 애플리케이션이 역공학에 이용되는 것을 예방하기 위한 특정 로직을 삽입하는 기법이다. 상기 기법의 일례로 역공학 시 이용되는 디버거들의 사용을 탐지하여 애플리케이션의 동작을 중지 또는 제한하는 로직을 삽입하는 방식이 존재한다.

2. 애플리케이션 행동 분석을 통한 악성코드 탐지 기법

모바일 악성코드 탐지 및 대응에 대한 연구는 분석 방식, 분석 대상 등에 따라 다양한 접근 방법을 가진다. 이러한 접근 방법 중 본 절에서는 악성코드 탐지 분석이 수행되는 시점에 따라 (1) 마켓과 같은 offline에서 단말에 설치되지 않은 애플리케이션을 분석하는 데 사용 가능한 악성코드 탐지 기법과 (2) 모바일 단말상에서 악성코드 탐지 및 대응을 위해 활용 가능한 online 모드의 탐지 기법들로 분류하여 설명한다.

가. Offline 모드 악성코드 탐지 기법

일반적으로 offline에서 적용 가능한 악성코드 탐지 기법은 크게 정적 분석(static analysis)과 동적 분석(dynamic analysis) 방법으로 구분된다[12].

1) 정적 분석 방법

정적 분석 방법은 프로그램에 대한 수행 없이 애플리케이션을 분석하는 방식이다. 안드로이드 애플리케이션에 대한 정적 분석은 Dalvik bytecode 자체나[2],[13] dex2jar, DED[14], soot, JD-GUI와 같은 도구를 사용하여 추출된 애플리케이션 소스 코드를 분석 대상으로 삼는다. 정적 분석의 목표는 애플리케이션의 보안 취약점이나 악의적 행동 여부를 탐지하는 것으로 이를 위하여 root 권한 획득을 위해 사용되는 exploit 탐지[15], permission 오남용 분석[16], 데이터 정보 누출 가능성 탐지[14],[15], IPC(Inter-Process Communication) 통신 취약점 분석[17] 등 다양한 보안 관점을 분석 대상으로 고려하고 있다. 또한, 정적 분석 방법은 악성코드 및 일반 애플리케이션의 분석 결과로부터 습득한 경험적 정보(heuristic)들을 이상 행동 및 취약점 탐지에 활용하며, 이를 위하여 애플리케이션의 permission, semantic, control flow 및 data flow 등을 분석한다. 일례로, 참고문헌 [14]는 데이터 누출 가능성을 탐지하기 위하여 "일반적인 애플리케이션이 hard coding된 특정 목적지로 SMS를 보내지 않는다"는 heuristic을 이용하며, 이를 탐지하기 위하여 애플리케이션 소스 코드의 semantic을 분석한다.

정적 분석 방법은 안드로이드 플랫폼 수정이나 애플리케이션 수행을 필요로 하지 않기 때문에 타 탐지 기법에 비해 분석 시간 및 비용이 적게 드는 장점을 가진다[18]. 따라서 급속하게 증가하고 있는 모바일 애플리케이션 수를 고려하였을 때, 추가적인 세부 분석 대상을 찾아내기 위한 필터링 수단으로서 활용될 수 있다. 그러나, 정적 분석 방법은 분석 결과가 애플리케이션으로부터 추출된 소스 코드의 정확도에 의존하는 문제점을 가진다. 즉, 코드 난독화 기술이 적용된 애플리케이션에 대한 분석이 어렵다는 단점을 가진다. 최근 이러한 난독화 기술을 통한 정적 분석 탐지 우회 문제를 일부 해결한 시스템(RiskRanger[15])이 소개 되었다. 그러나 RiskRanger는 난독화된 코드를 직접적으로 분석하기보다는, 정적 분석 시 사용된 기술을 우회할 수 있는 난독화 코드의 존재 여부를 탐지함으로써 난독화에 따른 false negative를 줄이는 방식으로 동작한다.

2) 동적 분석 방법

동적 분석 방법은 안드로이드 디버거나 에뮬레이터를 이용하여 프로그램을 수행시켜 봄으로써 애플리케이션의 세부 행동을 분석하는 방식이다. 그러나 모바일 단말상에서 디버거를 이용한 직접 적인 분석은 분석 도중 모바일 단말에 발생 가능한 피해 외에도 디버거 탐지 및 우회 로직을 포함한 악성코드들을 분석하지 못하는 단점을 갖는다[2]. 이러한 이유로, PC 기반 악성코드의 동적 분석에서와 같이 안드로이드 기반 악성코드의 동적 분석 역시 가상 머신이 활용되고 있으며, honeynet project[19]와 DroidScope[2]과 같은 가상 머신상에서 안드로이드 애플리케이션의 동적 분석을 돕기 위한 도구들이 개발되고 있다.

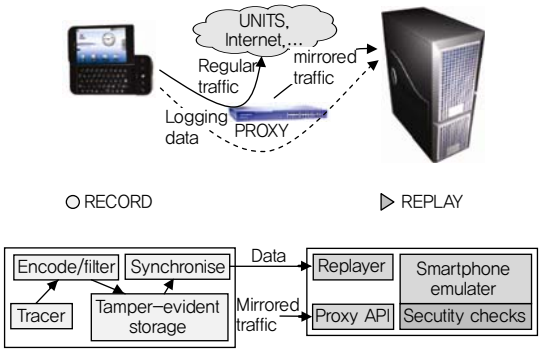
동적 분석 방법은 악성코드의 특징 및 시그니처를 추출하기 위해 많이 이용되고 있으며[12] 코드 난독화가 수행된 악성코드 역시 분석 가능하다는 장점을 갖는다[2]. 그러나 모든 가능한 프로그램 실행 패스를 다 분석하지 못하는 문제점을 가지고 있다. 실제로 초기 동적 분석 방법은 단지 하나의 프로그램 실행 패스만을 분석할 수 있었으며, 현재 동적 분석 범위를 높이기 위한 목적으로 다중 실행 패스 분석 기법들이 연구되고 있다[2],[20].

나. Online 모드 악성코드 탐지 기법

안드로이드 단말에 애플리케이션으로 설치되어 사용되는 안티바이러스 제품들은 시그니처를 기반으로 악성코드를 탐지한다. 그러나 이러한 시그니처 기반 탐지 방식은 새로 발생하는 공격을 탐지하지 못하는 단점을 갖는다[21]. 이러한 이유로 애플리케이션의 동작을 감시, 분석, 규제하는 것과 같은 적극적인 탐지 및 방어를 수행하기 위해 안드로이드 플랫폼을 수정하는 방안이 다양하게 연구되고 있다.

안드로이드 플랫폼 확장 기반 보안 솔루션들은 애플리케이션의 비정상 행동을 탐지하기 위해서, 배터리 소비량[22], 접근 권한[23], 데이터 흐름[24] 등을 조사 분석한다. 이러한 보안 솔루션의 일례로, Quire[23]는 IPC 통신 시 caller의 권한 정보 및 call chain과 같은 정보를 함께 전파함으로써, IPC 통신 취약점을 통한 권한 상승 시도를 탐지 가능하도록 수정된 안드로이드 미들웨어 시스템을 소개한다. 또 다른 예로서, TaintDroid[24]는 위치 정보, 카메라 이미지, 연락처 등과 같은 민감한 정보들을 추적 관리하고, 관리 대상인 정보들이 네트워크 등을 통해 외부로 누출되는 것을 탐지하도록 수정된 안드로이드 플랫폼을 소개한다. 그러나 위와 같은 모바일 단말상에서의 실시간 모니터링 기반 보안 솔루션들은 모바일 단말의 리소스 제약으로 인해 실제 적용되어 사용되지 못하고 있다.

모바일 단말의 리소스는 클라우드 기반 솔루션을 이용함으로써 절약될 수 있다[18],[25],[26]. 클라우드 기반 솔루션들은 리소스 절약을 위하여 모바일 단말에서



(그림 2) Paranoid Android 시스템 구조[25]

는 분석에 사용할 데이터만을 기록하며, 이상 행동 탐지는 기록된 데이터를 기반으로 외부 서버에서 수행한다. 대표적인 cloud 기반 솔루션인 paranoid android[25]는 외부 서버에서 모바일 단말의 행동을 재현 함으로써 악성코드를 탐지하는 방법을 제안하였다. 이를 위해 paranoid는 모바일 단말상에 tracer를 위치하여 애플리케이션의 행동 재현에 필요한 정보를 모두 기록한 후 외부 서버로 전송한다. 이 때, 재현 시 사용될 수 있는 네트워크 트래픽의 경우 그 사이즈가 크기 때문에 트래픽 데이터 전송으로 야기될 수 있는 리소스 소모를 줄이기 위하여 네트워크 트래픽 데이터는 단말이 아닌 프록시 서버에 기록한다. 이를 위해 모바일 단말의 네트워크 트래픽이 모두 프록시 서버를 통과 하도록 설정해야 한다. 외부 서버는 모바일 단말의 tracer 및 프록시 서버로부터 획득한 데이터를 바탕으로 모바일 단말의 행동을 재현하면서 이상 행동 탐지를 수행한다. (그림 2)는 이와 같은 특징을 갖는 paranoid android의 구조를 설명한다. Paranoid는 외부 재현 시 다양한 분석 방법을 이용하여 탐지 성능을 높일 수 있는 장점을 가진다. 그러나 모바일 단말상에서 tracer에 의해 소모되는 리소스(CPU 및 배터리)에 대한 개선이 여전히 필요하다. 또한, 네트워크 트래픽 기록을 통한 프라이버시 문제도 풀어야 할 숙제이다.

Ⅳ. 결론

본고에서는 현재 급속히 증가하고 있는 안드로이드 기반 모바일 악성코드의 특징과 함께 이들을 탐지하기 위해 제안된 다양한 보안 기법들을 살펴보았다.

현재 이러한 보안 솔루션에 대한 연구는 초기 단계에 있으며, 추가적인 탐지 성능 향상과 함께, 상호 유기적으로 연동하여 모바일 단말의 보안을 강화하기 위한 보안 에코 시스템의 형성이 시급히 필요하다.

용어해설

애플리케이션 위변조 개발자가 아닌 제 3자에 의해서 불법적으로 애플리케이션을 수정되는 것

역공학(reverse engineering) 완성된 제품을 분석하여 제품의 기본적인 설계 개념과 적용 기술을 파악하고 재현하는 것

False positive 정상적인 것을 악의적인 것으로 잘못 식별한 것

False negative 악의적인 것을 정상적인 것으로 잘못 식별한 것

약어 정리

DDoS	Distributed Denial of Service
IPC	Inter-Process Communication
QR	Quick Response
SMS	Short Message Service
SNS	Social Network Service

참고문헌

[1] Sophos, “Security Threat Report 2013,” 2013.

[2] L.K. Yan and H. Yin, “DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis,” Proc 21st USENIX conf. Security Symp., Security, 2012.

[3] McAfee, “McAfee Threats Report: First Quarter 2012,” 2012.

[4] Juniper Networks, “2011 Mobile Threats Report,” 2012.

[5] Y. Zhou and X. Jiang, “Dissecting Android Malware: Characterization and Evolution,” Proc 33rd IEEE

Symp Security and Privacy, 2012.

[6] Arxan, "State of Security in the App Economy: Mobile Apps under Attack," 2012.

[7] 금융위원회, "온라인결제 보안강화 종합대책 마련," 2013. 4. 9.

[8] W. Zhou et al., "DroidMOSS: Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces," Proc 2nd ACM Conf. Data Appl. Security Privacy (CODASPY), 2012.

[9] CIOCISO Articles, "앱 위변조 방지 솔루션," 2012. 7. 1.

[10] P. Yuxue, 정진혁, 이정현, "모바일 난독화 기술 동향," 정보와통신, 한국통신학회지, vol. 29, no. 8, 2012. 7, pp. 65-71.

[11] C. Collberg and J. Nagra, Surreptitious Software: Obfuscation, Watermarking, and Tamper proofing for Software Protection, Addison Wesley Professional, 2009.

[12] R. Xu, H. Saïdi, and R. Anderson, "Aurasium: Practical Policy Enforcement for Android Applications," 21st USENIX Security Symp. USENIX, 2012.

[13] Y. Zhou, A. Wang, and X. Jiang, "Hey, You, Get Off My Market: Detecting Malicious Apps in Official and Alternative Android Markets," Proc 19th Netw. Distrib. Syst. Security Symp., 2012.

[14] W. Enck et al., "A Study of Android Application Security," Proc. 20th USENIX Conf. Security, Security, 2011.

[15] M. Grace et al., "RiskRanker: Scalable and Accurate Zero-day Android Malware Detection," Proc. Mobile Syst., Appl., Services, ACM, 2012, pp. 281-294.

[16] A.P. Felt et al., "Android Permissions Demystified," Proc. 18th ACM Conf. Comput. Commun. Security (CCS), 2011.

[17] E. Chin et al.., "Analyzing Inter-Application Communication in Android," Proc. Annual Int. Conf. Mobile Syst., Appli., and Services, 2011.

[18] M. Chandramohan, and H.B.K. Tan, "Detection of Moblie Malware in the Wild," Comput., vol, 45 no. 9, Sept. 2012.

[19] The Honeynet Project. www.honeynet.org

[20] A. Moser, C. Kruegel, and E. Kirda, "Exploring Multiple Execution Paths for Malware Analysis," Proc. IEEE Symp. Security Privacy, SP, 2007, pp. 231-245.

[21] W. Enck, M. Ongtang, and P. McDaniel, "On Lightweight Mobile Phone Application Certification," Proc. 16th ACM Conf. Comput. Commun. Security (CCS), pp. 235-245.

[22] L. Liu et al., "VirusMeter: Preventing Your Cellphone from Spies," Proc. 12th Int. Symp. Recent Adv. Intrusion Detection (RAID), 2009

[23] M. Dietz et al., "Quire: Lightweight Provenance for Smart Phone Operating Systems," Proc 20th USENIX Conf. Security (SEC), 2011.

[24] W. Enck, P. Gilbert, and B. Chun "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones," Proc. 9th USENIX Conf. Operating Syst. Des. Implementation (OSDI), 2010.

[25] G. Portokalidis et al., "Paranoid Android: Versatile Protection for Smartphones," Proc. 26th Annual Comput. Security Appl. Conf. (ACSAC), 2010, pp. 347-356.

[26] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-based Malware Detection System for Android," Proc 1st Workshop Security Privacy Smartphones Mobile Devices (CCS-SPSM), 2011.