Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.25, NO.6, Dec. 2015 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2015.25.6.1341

Androfilter: 유효마켓데이터를 이용한 안드로이드 악성코드 필터*

양 원 우,† 김 지 혜‡
국민대학교 보안-스마트전기자동차학과

Androfilter: Android Malware Filter using Valid Market Data*

Wonwoo Yang,† Jihye Kim‡
Kookmin University Secured Smart Electric Vehicle specialist Education

요 약

스마트폰의 대중화로 다양한 애플리케이션이 증가하면서, Third party App Market 이나 블랙마켓을 통한 악성 애플리케이션 또한 급격한 증가세에 있다. 본 논문에서는 APK파일의 변조여부를 효과적으로 검출할 수 있는 검사 필터인 Androfilter를 제안한다. Androfilter는 대부분의 안티바이러스 소프트웨어들이 사용하는 수집, 분석, 업데이트 서버등을 사용하지 않고, Google Play를 신뢰 기관으로 가정하여 대응되는 애플리케이션의 조회만으로 애플리케이션의 변조여부를 판단 한다. 실험 결과에 따르면 변조된 애플리케이션을 감지함으로 보고되지 않은 신종 악성코드를 차단할 수 있다.

ABSTRACT

As the popularization of smartphone increases the number of various applications, the number of malicious applications also grows rapidly through the third party App Market or black market. This paper suggests an investigation filter, Androfilter, that detects the fabrication of APK file effectively. Whereas the most of antivirus software uses a separate server to collect, analyze, and update malicious applications, Androfilter assumes Google Play as the trusted party and verifies integrity of an application through a simple query to Google Play. Experiment results show that Androfilter blocks brand new malicious applications that have not been reported yet as well as known malicious applications.

Keywords: Android, APK, Malware, Filter, Anti-malware, Meta-data, live-data, ESD

I. 서 론

1.1 스마트폰 보급과 위협

전 세계적으로 스마트폰이 빠르게 보급되고 있다. eMarketer의 자료에 의하면, 2014년 기준 전 세계 인구의 25%인 약 18억 명이 스마트폰을 사용하고 있으며, 2017년까지 전 세계 인구의 34% 이상이 스마트폰 사용자일 것으로 예상하고 있다[1].

스마트폰의 급속한 보급으로 스마트폰을 대상으로 하는 악성 코드의 위협 또한 증가세에 있다. 상당량의 스마트폰 서비스들은 개인에게 특화된 서비스를 제공하기 위해 사용자의 생활과 관련된 정보를 접근 또는 수집한다. 하지만 개인의 프라이버시와 관련된 민감한 정보들이 안전하게 보호되지 않는다면 사용자들의 개인정보 유출이나, 재산피해 등을 야기할 수 있다. 실제로 스토킹 앱이나 스마트TV 해킹 등으로

Received(05 .19. 2015), Modified(1st: 09. 03. 2015, 2nd: 10. 05. 2015), Accepted(10. 19. 2015)
* 본 연구는 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임. (2012R1A1B5000634).
† 주저자, noiz@kookmin.ac.kr
‡ 교신저자, jihyek@kookmin.ac.kr(Corresponding author)

사용자의 사생활을 훔쳐보거나 스미싱, 파밍 공격과 같은 방법으로 직접 금전적 피해를 끼치는 사례가 급증하고 있다.

스마트폰을 대상으로 한 공격은 개인의 수준에 그치지 않고 사회적인 위협으로 확대될 수 있다. 예를 들어 공격자가 교통 인프라의 마비를 목적으로 교통정보 서비스를 점거하여 정상정보 대신 허위정보로 속여서 제공하면 단 한 개의 서비스만 공격했음에도 스마트폰으로 교통정보를 서비스받는 모든 사용자에게 광범위한 사회적 혼란을 일으킬 수 있게 된다.

이런 공격을 가능하게 하는 매체 중 가장 영향력이 큰 것은 악성 코드 유포를 이용한 공격이다.

1.2 공격의 형태

스마트폰을 대상으로 한 공격을 막기 위해 Android 와 같은 OS에는 자체적으로 보안장치들을 만들어놓고 운용 하고 있다. 따라서 정상적인 OS 사용 환경에서 공격자가 시스템 취약점을 찾는 것은 쉽지 않은 작업이며 발견되는 취약점 또한 업데이트를 통해 지속적으로 패치가 된다. 공격자 입장에서 시스템 취약점을 이용하는 방법은 그 난이도 자체도 높으며, 지속 가능한 공격방법도 아니기 때문에 비경제적이다. 따라서 공격자는 사회 공학적 방법으로 피해자를 꾀어 단말기에 악성 애플리케이션을 설치하게끔 유도하여 공격을 성립시키는 방법을 선호한다.

사회공학적인 공격방법은 사용자에게 악성코드를 설치하게끔 특정한 행동을 유도해야하기 때문에 많은 제약이 존재한다. 주로 사용자에게 유료 애플리케이션을 공짜로 쓸 수 있다고 유혹하거나, 배송 조회 서비스, 공공 기관 서비스를 사칭하는 URL에서 악성 애플리케이션을 다운받고 설치하도록 사용자를 유도하여 공격을 성공시킨다. 사용자가 해적판 애플리케이션을 다운로드 받아서 사용하면 공격자가 원본 애플리케이션을 변조하여 악의적인 코드를 삽입할 수 있기 때문에 사용자는 정상 애플리케이션을 사용한다고 생각하게 만들면서 뒤로는 악의적인 동작을 할 수 있다. 심지어 설치시에 보이는 애플리케이션의 아이콘과 이름만 유사하게 꾸미고 설치가 끝나면 애플리케이션 목록에서 숨기는 식으로 설치가 안 된 것처럼 피해자를 속이는 방법도 있다.

이렇게 제한된 방법으로 악성코드가 유포됨에도 불구하고 스마트폰의 특성상 사용자들이 항상 휴대하며 주기적으로 확인하기 때문에 악성 애플리케이션이 유포되는 속도는 엄청나다. 일례로 Droid Dream 이라는 Android 악성코드는 불과 48시간 만에 26만대의 단말기를 감염시킨 전력이 있다.[9] 이런 악성코드는 일단 감염이 되면 피해자의 주소록에 있는 연락처로 SMS등을 보내 2차 피해를 발생시키는 사례를 빈번히 보이는데, 아는 사람에게 보내는 행사 초대장 등을 빙자하기 때문에 2차 피해자의 경계를 느슨하게 하는 효과가 있다. 이렇듯 인간의 심리를 이용한 방법도 동원되기 때문에 공격 성공률과 유포 속도는 더욱 가속된다.

빠른 유포 속도에 비해, 탐지는 더욱 어려워지고 있다. 여러 개의 변종을 만들어 배포하여 백신을 회피하게 만들거나 난독 화를 하여 분석을 힘들게 하는 방법 등이 사용된다. 이 때문에 악성코드를 검사하기 위해 더욱 정교한 분석을 요구하여, 분석결과의 신뢰도를 높이기 위해 컴퓨팅 자원이 많이 필요한 작업들이 필요해지게 되고 있다.

1.3 현재의 문제

1.3.1 검사 시스템의 한계

전통적인 백신은 악성코드를 수집하고, 분석해서 악성코드의 signature를 생성하여 백신 클라이언트에 주기적으로 업데이트한다.[12] 클라이언트는 업데이트된 signature 정보와 다운로드 받은 애플리케이션을 대조하여 애플리케이션의 악성유무를 가린다. 악성코드가 유포되면 발견즉시 signature가 빠르게 업데이트되기 때문에 대부분의 경우 통계적으로 2시간 이후에는 백신업체가 샘플을 수집하게 되어 그때부터는 유포를 차단할 수 있게 된다[2].

하지만, 스마트폰의 경우 제한된 배터리와 연산 자원을 가지고 있기 때문에 항상 최신의 signature 정보를 유지하는 것이 힘들다. 만에 하나 최신 signature 정보 유지가 가능하다고 가정하더라도, 백신은 악성코드가 수집, 분석을 거쳐 최종적으로 패턴이 업데이트되기 전까지의 공백 기간 동안 무방비로 노출되는 시간이 존재하는 구조적 한계를 가지고 있다. 이 때문에, PC환경에서는 동적 분석[4][7][8], 정적 분석[6][11][15]을 사용하며 최근에는 기계학습을 이용한 분석방식[3][4][5][7] 까지 동원하여 이런 보안 공백을 보완하는 중이다. 그러나, 앞서 언급한 것처럼 스마트폰 에서는 이런 복잡한 분석 시스템은 컴퓨팅 비용이 너무 크기 때문에

사용할 수가 없다.

컴퓨팅 자원, 배터리 문제를 극복하기 위해 연산 능력이 강력한 분석서버에 검사를 위탁하는 접근 방식도 있다. 즉, 분석서버에 애플리케이션을 전송하고, 분석서버는 분석결과를 스마트폰에 반환한다. 이와 같은 방식은 결과를 받는 동안의 시간이 비교적 오래 걸리고, 애플리케이션의 크기만큼 통신 트래픽이 발생한다는 단점이 있다.[16]

Google 차원에서도 노력이 이루어지고 있다 Android 4.2 버전이후부터 설치하는 애플리케이션의 악성여부를 간단히 검사해주는 Verify Apps 라는 기능을 제공하기 시작했지만 2015년 8월 현재 4.2 미만의 버전의 사용자의 22%가 이 기능의 보호를 받지 못하고 있으며, 그 이상의 버전을 사용하는 기기의 경우에도 본 논문에서 사용한 악성코드 300여개를 Android 5.1.1 버전의 기기(Nexus 7 2013)에 설치를 시도 해 본 결과 5개를 제외한 악성코드의 98.3%의 설치를 차단하지 못하여 제대로 본래의 역할을 하지 못함을 보이고 있다.

1.3.2 사용 환경상의 문제

본래 Android 호환성 테스트를 통과하여 GooglePlay 를 사용할 수 있는 기기라면 GooglePlay 이외의 경로로 애플리케이션을 설치하는 것이 차단되어 있는 상태로 출고되는 경우가 대부분이다. 임의로 다운받은 APK 파일을 직접 설치하려면 환경설정의 보안메뉴에서 "알 수 없는 출처의 애플리케이션 설치" 옵션을 허용해야 하며, 일반적인 환경이라면 사용자가 이 옵션을 선택하지 않아도 기기를 이용하는데 지장이 없다. 피싱, 스미싱에 사용되는 악성 애플리케이션의 절대 다수가 이 옵션을 허용하지 않으면 설치 자체가 되지 않기 때문에 이 옵션을 허가하지 않는 것 자체가 일종의 보안장벽과 같은 구실을 해 준다.

하지만 이런 정상기기를 사용하는 사용자 또한, 통신사나, 모바일 기기 제조사들이 운영하는 3rd party market을 사용하려면 "알 수 없는 출처의 애플리케이션 설치"를 허용하라고 안내 받게 된다. 이런 보안 설정을 완화하는 행위가 큰 보안 취약점으로 작용할 수 있다는 점이 충분히 안내가 되고 있지 않고, 오히려 3rd party market 을 운영하는 주체들은 각자의 이익을 위해 이 옵션의 허용을 요구하고 있는 것이 현실이다. 심지어 일부 저가형 제품들의 경우 Google 의 인증을 받지 않은 채 3rd party market 만 설치된 채로 유통되는 경우도 많다. 이 경우에는 문제가 더 심각하다.

1.4 Androfilter 제안

본 논문에서는 입력된 APK 파일이 GooglePlay와 같은 신뢰기관에서 정상 배포중인 파일을 변조한 것인지 여부를 특별한 서버 없이 확인 가능한 Androfilter를 제안한다. Androfilter는 사용자에게 GooglePlay에 등록된 유효한 APK인지 여부와 그 변조 가능성을 알려주어 비정상적으로 배포되는 APK 파일의 설치를 방지해 준다. 정상적인 APK 파일을 임의로 변조했다는 것은 선의로 이루어지지 않은 경우가 대부분이기 때문에 변조여부를 악성을 판별하는 근거로 사용가능하다. 본 연구의 실험 결과에 의하면, 실제로 대부분의 악성코드들이 Androfilter 정도만으로도 충분히 탐지가 가능하다.

1.5 Androfilter의 특장점

본 논문에서는 Androfilter의 신뢰기관으로 GooglePlay를 가정하여 설치할 애플리케이션의 원본 정보를 조회하고 Meta-Data를 비교하여, 다운받은 애플리케이션의 변조여부를 판단 한다. 제안하는 필터의 특징은 다음과 같이 요약된다.

1.5.1 서버를 운용 할 필요가 없다.

기존의 방식은 서비스를 운용하기 위해 여러 종류의 서버가 필요하다. 예를 들어 정상 APK를 수집하는 수집서버, 수집된 APK를 분석하여 Signature를 생성하는 분석서버, 백신 클라이언트에 signature를 넘겨주는 업데이트 서버 등이다.[10] 제안하는 Androfilter의 경우에는 단말에서 바로 변조여부를 조사할 수 있기 때문에 위의 서버가 모두 없이 인터넷에 접속할 수 있는 단말만 있어도 악성애플리케이션을 검사할 수 있다.

운용비용 상의 문제뿐만 아니라 보안적인 측면에서도 서버가 필요 없는 시스템은 추가적인 이점을 가지고 있다. 공격자가 업데이트 서버를 점거하여 해당 서비스의 사용자 전부에게 업데이트를 가장하여 악성코드를 유포해서 공격하는 행위에 대해 원천적으로 면역을 가질 수 있기 때문이다. 또한, 업데이트 서버

의 장애로 인한 보안공백으로 위협에 노출되는 사고도 염려하지 않아도 된다.

1.5.2 분석을 위해 APK파일을 저장할 필요가 없다.

GooglePlay를 기반으로 모든 APK를 수집하는 기존의 방식에서는 유료 애플리케이션을 분석하기 위해 애플리케이션을 구매하는 추가적인 비용이 발생할 수 있다. 하지만 사실상 market 에 올라와있는 모든 유료 애플리케이션을 구매하여 분석한다는 것은 사실상 불가능하며, 하루에도 셀 수 없이 업데이트되는 애플리케이션을 매번 다시 수집하여 분석하는 것도 실용하기 어려운 이유가 된다.

제안하는 Androfilter는 분석하는 시점에 애플리케이션의 Live-Data 만 조회하기 때문에 구매로 인해 발생하는 비용도 없으며 검사에 필요한 데이터만을 조회하기 때문에 검사를 할지 안할지도 모르는 대상의 signature를 클라이언트가 전부 가지고 있거나 업데이트를 할 필요가 없다. 기존의 수집서버와는 다르게 APK 파일을 직접 수집하지 않아도 되기 때문에 네트워크 트래픽 오버헤드를 최소화한다.

1.5.3 변종에 강하며, 보안 공백이 줄어든다.

효과적인 악성코드가 하나 등장하면 이것을 변조시켜 Signature 검사를 우회하게 해주는 Packer 유형의 툴들이 다수 존재한다. 이런 변조 툴 때문에 사실상 같은 악성코드임에도 불구하고 지속적으로 변종이 등장하여 재차 피해를 입히는 경우가 발생한다. Androfilter 는 APK파일이 주장하는 애플리케이션을 패키지 명으로 조회하기 때문에 변종이 발생하더라도 패키지 명을 변경하지 않는 이상 같은 대상을 가리키게 되어 동일한 정상 애플리케이션이 아니면 필터를 통과할 수 없게 된다.

Androfilter는 APK를 설치하려고 시도를 하는 시점에서 신뢰기관(GooglePlay)에 대응되는 패키지 명을 조회하여 분석하고 검사하기 때문에 새로 유포된 신종 변조 애플리케이션도 효과적으로 검사 해낼 수 있다. 즉, 신종 변조 애플리케이션이나, 이렇게 생성된 악성 애플리케이션들은 보안 공백이 없이 바로 구별가능하다.

1.5.4 기존의 시스템과 연동 가능하다.

서비스를 제공하기 위한 서버를 사용하지 않으며 독립적으로 동작할 수 있기 때문에 기존 대형 검사 시스템의 일련된 검사 단계중 하나로 추가하여 사용할 수 있다. 예를 들어 연산량이 적고 빠른 정적 분석 단계와 연산량이 많은 동적 분석 단계 사이에 Androfilter를 추가하여 전체 검사 시스템의 효율을 높일 수 있다.

II. Androfilter

2.1 가정 및 전제

본 연구에서는 신뢰기관에서 검사대상과 대응되는 애플리케이션의 원본 정보를 조회하여 변조여부를 판단하는 Androfilter를 제안한다. 실제로 배포되는 악성애플리케이션의 절대 다수가 GooglePlay에 등록이 되어있지 않고, 남은 소수는 이미 등록된 정상 애플리케이션으로 패키지 명을 속이고 있다.

Androfilter는 복수의 신뢰기관을 두고 구현 될 수 있으며 여러 가지 유통 경로에 대해 정책을 다르게 세워서 운용할 수 있다. 하지만 본 논문에서는 구현의 편의성과 변인통제를 위해 GooglePlay를 유일한 신뢰기관으로 가정하도록 정책을 세워 운용한다. 이때, GooglePlay에 등록되어 있지 않고 블랙마켓에서만 유통되는 애플리케이션은 악성으로 분류하도록 한다. 또, 정상등록 되어있는 애플리케이션이라고 할지라도 최신 버전보다 더 높은 버전이 유통되는 것은 악성으로 분류하는 정책을 사용한다. 배포되고 있는 버전보다 낮은 경우에는 낮은 버전의 APK의 정보를 조회하거나 파일을 요청하여 대조할 Live-Data를 얻을 수 있다.

2.2 Androfilter 시스템

Androfilter의 전체 시스템 구성도는 Fig. 1.과 같다. Androfilter는 Web이나 Black Market에서 다운로드 받은 APK 파일을 검사 대상으로 입력받아 APK내부의 Meta-Data를 추출하여 이를 GooglePlay와 같은 신뢰기관에서 조회한 Live-Data 와 비교하여 APK이 변조 되었는지 여부를 판단하게 된다.

이런 시스템은 Android Device에 탑재될 수 있으며 대형 검사 시스템의 일련 된 분석 방법 중 하나로 사용될 수도 있다.

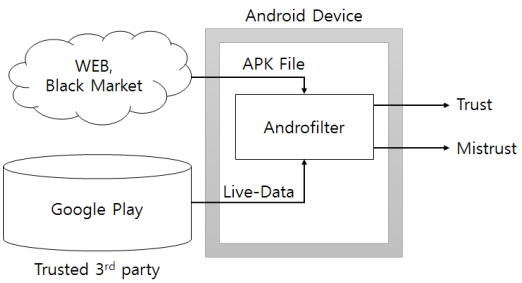

Fig. 1. Androfilter System

2.3 Meta-Data

Androfilter에서 변조여부를 확인하기 위해 사용되는 Meta-Data들의 항목들은 다음과 같다.

2.3.1 패키지 명 (Package Name)

애플리케이션은 고유한 패키지 명을 가지고 이를 애플리케이션을 서로 구분하는 수단으로 사용한다. GooglePaly와 같은 market 에서 조회 할 때에도 이 패키지 명을 이용한다. 하지만 이것은 공격자가 정상 애플리케이션과 동일하게 조작할 수 있기 때문에 다른 정보를 함께 대조 해 보아야만 한다.

2.3.2 버전 (Version)

버전정보는 총 3가지이다. APK 빌드에 사용된 SDK의 버전, Version Name (문자열), Version Code(숫자).

유포되는 악성 APK를 더 최신버전으로 속여 업데이트를 유도하거나 임의의 값으로 버전이 변조되어 있는 경우 알아낼 수 있다. 하지만 이 버전정보 역시 공격자가 임의로 바꿀 수 있다.

2.3.3 애플리케이션 요청 권한 (Permission)

Permission은 스마트폰의 특정 기능을 이용하기 위해 애플리케이션이 얻어야 하는 허가이다. 예를 들어 애플리케이션이 네트워크 트래픽을 사용하기 위해서는 "android.permission.INTERNET" 이라는 Permission을 얻어야 한다. 이런 Permission 정보는 Meta-Data로 추출이 가능하다.

APK 파일이 GooglePlay에 존재하는 정상 애플리케이션과 같은 버전, 같은 패키지 명을 주장하더라도 요구하는 Permission의 개수나 종류가 다르다면 변조된 애플리케이션이라고 판단할 수 있다.

정상적으로 애플리케이션이 요구하는 Permission이라고 할지라도 SMS, 주소록, 파일접근권한 처럼 주장하는 기능과 상관없음에도 개인정보 유출에 사용 될 수 있는 Permission들을 요구하는 경우에는 정책에 따라 Grey ware로 구분하여 악성이 아니라도 관심을 가지고 분석하게 정책을 세울 수도 있다. 모바일 환경에서는 다시 한 번 사용자에게 알려주어 위협이 될 수 있다고 경고하는 용도로도 사용 될 수 있다.

2.3.4 Filesize & Hash

APK파일로 packing 되어 있는 내부 파일들의 hash값을 확인할 수 있고, 신뢰기관에서 이 파일들의 hash 값을 제공한다면 완벽하게 APK파일의 무결성을 증명할 수 있다. 하지만 이러한 정보를 얻을 수 있는 방법이 없기 때문에 본 논문에서는 이 데이터를 대체하여 APK 파일의 정확한 크기를 사용한다.

차선책으로는 Trusted Party에서 APK파일을 다운로드 받아 Hash를 구하거나 Hash를 얻을 수 있는 API를 이용하는 방법이 있었으나 현재 구글측에서 차단해놓은 상태라 사용 할 수 없다.

III. 구 현

Androfilter를 구현하기 위해 필터내부의 각 기능을 Fig.2. 와 같이 크게 3가지 구성요소로 분류하여 설계하였다.

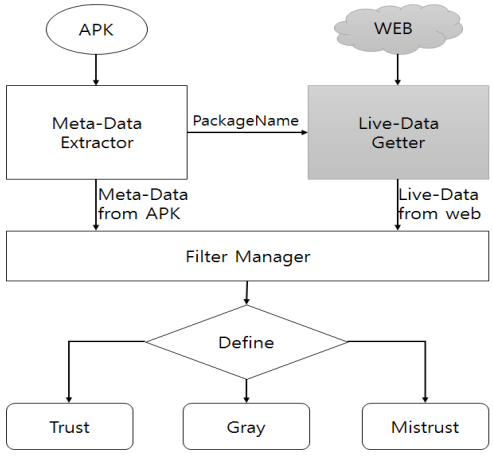

Fig. 2. Components of Androfilter System

필터의 동작을 제어하고 입출력 결과를 관장하는 Filter Manager, APK를 분석하는 Meta-Data Extractor(APK Anatomy), 온라인에서 정상 애플리케이션의 Live-Data를 조회하는 Live-Data Getter로 구분된다.

필터는 Python 2.7.x version으로 Linux (Ubuntu) 위에서 동작하도록 구현되었다.

3.1 Meta-Data Extractor

APK파일에서 내부정보를 추출하는 모듈이며 APK파일을 de-packaging 하여 내부의 파일들을 분리해내고 분석하여 Meta-Data를 얻게 된다. 더 나아가 내부의 3rd Party Dynamic Library도 분리 해 낼수 있다.

오픈소스로 배포되는 APK inspector 나 APKTool의 일부에 인터페이스를 달아서 구현할 수 있으며 본 연구에서는 APKTool을 사용하였다.

추출된 Meta-Data는 Fig. 3. 과 같은 정보를 포함하며 이는 Filter Manager에게 전달되어 악성여부 판단에 사용된다.

```
>>>
filename:       D:\tj.apk
app version c:  1
app version n:  1.0
packager:       com.Copon
permission:     ('android.permission.INTERNET\nand
nandroid.permission.RECEIVE_SMS\nandroid.permissio
receiver:       ('com.Copon.SMS', 1)
service:        ('com.Copon.clService', 1)
valid !!
D:\tj.apk is not mal APK
>>>
```

Fig. 3. Execution of Meta-Data Extractor

3.2 Live-Data Getter

Live-Data Getter에서 얻을 수 있는 정보는 Fig. 4.와 같으며 2가지 방법으로 구현되었다. 하나는 Selenium WebDriver 를 이용한 웹 파싱이고 다른 하나는 (Unofficial) android client API를 이용한 방식이다.

```
>>>
launch browser
login complite
find app
click install button
get permission name start
//div[@id='base-dialog-body-content']/div/div[2]/div/
Wi-Fi 사용 설정 여부나 연결된 Wi-Fi 기기의 이름 등 Wi-Fi 네
program finish
```

Fig. 4. Execution of Live-Data Getter

3.2.1 Selenium WebDriver

Selenium WebDriver를 이용한 구현은 GooglePlay와 같은 Trusted Party에서 웹페이지를 패키지명으로 조회하여 애플리케이션의 정보를 추출해오게 구현하는 방법이다.

http://play.google.com/store/apps/details?id=〔PACKAGE_NAME〕 과 같은 형식으로 HTTP 요청을 하여 애플리케이션의 정보를 조회할 수 있는 웹페이지를 얻어서 정보를 추출(parsing)한다.

하지만 이렇게 얻은 애플리케이션의 정보는 APK파일에서 추출한 정보와 표기형식이 다소 다르다. permission의 경우에는 Android 내부에서 사용되는 상수명이 아니라 사용자가 읽고 이해할 수 있게 해석되어 표기되어 있기 때문에 Meta-Data와 비교하기 위해서는 형식을 통일시켜주는 Dictionary 를 만들어 변환작업을 거쳐야 한다.

3.2.2 Unofficial Android Client API

실제 Android Device와 GooglePlay간의 트래픽을 분석하여 역설계하여 만들어진 Unofficial Android Client API를 이용하여 구현된다. 유저의 GooglePlay 계정 인증토큰과 검사하고자 하는 애플리케이션의 패키지 이름, Android 디바이스의 식별정보 (android_id =IMEI or UDID)를 가지고 android.client.google.com 도메인의 API 로 요청하여 애플리케이션의 Live-Data를 얻어오는 방식이다. 이 방식으로 얻은 정보는 Permission들이 상수의 이름 형태로 넘어오기 때문에 사용하기 쉽고 파일의 크기등도 근사치가 아닌 정확한 수치다. 하지만 Google 측에서 공개한 정식 API가 아니기 때문에 반복해서 사용 할 경우 IP기반으로 일정시간 동안 사용할 수 없도록 차단된다. 또, 과거에는 SHA-1 Hash값을 얻을 수 있었지만 현재는 지원하지 않는다.

이런 비 공식적인 방법이 아니라 유사한 방식으로 Google이 API를 공개한다면 좀 더 양질의 데이터를 사용하여 좋은 결과를 낼 수 있다.

3.3 Filter Manager

Meta-Data Extractor와 Live-Data Getter에서 얻은 정보들을 서로 대조, 분석하여 악성이나 무고성 여부를 최종 판단하는 모듈이다.

검사의 강도나 보안정책에 검사 결과의 처리까지 수행 한다. 악성파일을 보관할것인지 삭제할것인지, 악성은 아니라고 판단했지만 잠재적으로 개인정보 유출의 위험이 있는 permisson들을 많이 요구한다던지 하는 항목에 대해서 사용자에게 경고를 하는등의 조치를 취할 수 있도록 도와주는 기능도 여기서 운용된다.

IV. 실험 및 결과

구현한 필터가 실제로 사용 가능한 수준의 성능을 보이는지 알아보기 위해 악성코드와 정상애플리케이션 샘플을 가지고 실험을 수행한다.

실험에 사용한 샘플은 세 종류로 하나는 배포 초기에 수집된 미보고 악성 애플리케이션이고 다른 하나는 과거에 많은 사용자를 감염시킨 유명 악성코드이다. 이 두 개의 악성코드가 검사되는 과정을 보여 실제로 어떻게 검사가 이루어지는지 보인다.

마지막으로 대량의 악성코드와 정상파일 들을 함께 검사하여 실용성이 있는지 확인한다.

4.1 미보고 악성 애플리케이션

Fig. 5.와 같은 문자로 받은 URL에서 다운로드 하여 스미싱 목적으로 유포되는 악성코드를 수집하였다. 수집한 악성 APK는 SMS를 통해서 유포되는 것이고 택배의 배송 정보를 알려주는 것으로 위장하여 링크의 클릭을 유도하게 된다. 이 샘플을 수집하

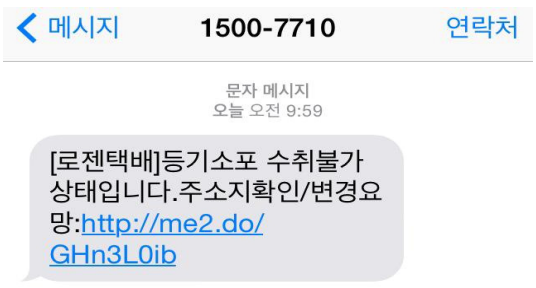

Fig. 5. Smishing Application Sample

였을 때는 악성코드의 배포가 시작된 지 4시간이 지나지 않은 상태였다.

백신들이 얼마나 잡아낼 수 있는지 확인하기 위해 업로드한 파일을 여러개의 백신으로 검사를 하여 결과를 보여주는 Virustotal 이라는 서비스를 이용한다. Naver.APK는 Virustotal에서 14-08-22에 최초로 분석 되었으며 이때는 55개의 백신중 19개가 악성이라고 판단을 내렸다. 4일이 지난 14-08-26에는 19개보다 6개 많아진 25개의 백신에서 악성으로 판단하였다. 이렇게 악성 판단을 다르게 하는 양상은 배포 초기에는 악성으로 판단하지 못하는 백신도 존재한다는 것을 보여주고 있다.

Fig. 6.처럼 Meta-Data Extractor가 APK파일을 분석하여 얻은 패키지 명으로 Live-Data Getter를 수행하면 GooglePlay에 등록이 되어있지 않은 애플리케이션이라는 것을 알 수 있고, 앞서 세운 정책에 의해 악성으로 판단할 수 있다.

이 과정 중에 악성코드에 대한 어떠한 Signature 나 Pattern Matching을 하지 않았음에도 악성여부를 판단 할 수 있었다.

```
fileName : naver.apk
package: name='com.android.slientinstall'
versionCode='1'
versionName='1.0'
sdkVersion:'8'
targetSdkVersion:'18'
uses-permission:'android.permission.MOUNT_UNMOUNT_FILESYSTEMS'
uses-permission:'android.permission.WRITE_EXTERNAL_STORAGE'
uses-permission:'android.permission.READ_EXTERNAL_STORAGE'
...
```

Fig. 6. Meta-Data Extract from naver.APK

4.2 큰 피해를 냈던 유명한 악성 애플리케이션

과거의 큰 피해를 냈던 비교적 정교하게 제작된 악성 애플리케이션에도 필터가 잘 작동하는지 확인해 본다.

분석에 사용된 악성 애플리케이션은 Droid Kung-fu의 변종중 하나다.

분석은 Meta-Data Extractor가 Fig. 7.처럼 Meta-Data를 분석하는 것으로 시작한다. 여기서 얻은 패키지 명을 Live-Data Getter로 GooglePlay 에서 조회를 시도하면 이전과는 다르게 정상적으로 애플리케이션의 정보가 조회된다.

하지만 다른 정보를 비교해보면 비정상적인 부분을 발견할 수 있다. 우선 APK파일이 주장하는 버전은 1.0.5 이지만 GooglePlay에서 배포되고 있는 최신버전은 1.0.4이다 뿐만 아니라 APK파일은

```
package: name='com.aijiaoyou.android.sipphone'
versionCode='1005'
versionName='1.0.5'
sdkVersion:'5'
application-label:'???????
application-icon-120:'res/drawable/hllinphone.png'
application-icon-160:'res/drawable/hllinphone.png'
application-icon-240:'res/drawable-hdpi/hllinphone.png'
application: label='???????' icon='res/drawable/hllinphone.p
launchable-activity: name='com.aijiaoyou.android.sipphone.l
label='' icon=''
uses-permission:'android.permission.INTERNET'
uses-permission:'android.permission.RECORD_AUDIO'
uses-permission:'android.permission.READ_PHONE_STATE'
uses-permission:'android.permission.MODIFY_AUDIO_SETTINGS'
uses-permission:'android.permission.ACCESS_NETWORK_STATE'
uses-permission:'android.permission.WAKE_LOCK'
uses-permission:'android.permission.BOOT_COMPLETED'
uses-permission:'android.permission.VIBRATE'
uses-permission:'android.permission.GET_TASKS'
uses-permission:'android.permission.WRITE_EXTERNAL_STORAGE'
uses-permission:'android.permission.ACCESS_WIFI_STATE'
uses-permission:'android.permission.CHANGE_WIFI_STATE'
uses-permission:'android.permission.INSTALL_PACKAGES'
uses-permission:'android.permission.READ_EXTERNAL_STORAGE'
uses-implied-permission:'android.permission.READ_EXTERNAL_S
                        'requested WRITE_EXTERNAL_STORAGE'
...|
```

Fig. 7. Meta-Data Extract from Droid Kung-fu series sample

2.2MB이지만 GooglePlay에 등록된 버전은 5.9MB로 큰 차이를 보인다. 분명히 배포되고 있는 정상 애플리케이션으로 위장했지만 Meta-Data를 비교하여 조작이 가해진 악성 애플리케이션임을 알아낼 수 있다.

이렇게 많은 피해자를 발생시켰던 정교한 악성코드도 필터로 변조여부를 확인하는 것이 가능하다.

4.3 대량의 악성애플리케이션 샘플

유포되었던 악성코드들을 대량으로 수집하여 필터를 통과시켜보아 필터가 제대로 동작하는지를 확인한다. 실험에 사용 된 샘플은 VirusShare 에서 수집하여 제공하는 14906개의 악성 APK 들과[13] GooglePlay에서 다운로드 받은 정상 애플리케이션 100개를 사용 했다.

검사결과 False Positive 는 1개 False Negative는 4개가 나왔고 나머지는 정상적으로 변조여부를 구분하였다. 탐지율은 99.966%이다. 상위 10개 상용 백신들의 탐지율이 99.8% ~ 97.4% 임을 감안하면 충분히 실용 가능한 신뢰도를 보여준다.[17]

Table 1. Statistics of the Androfilter

	Malware	NormalAPK
Malicious	14905 (sensitivity)	4 (FalseNegative)
Safe	1 (False Positive)	96 (specificity)

Table 2. Statistics of the Filtered Malware Sample

item	count	percentage
Retrieval failure from Live-Data	13985	93.823 %
Retrieval success from Live-Data	921	6.176%

실험 결과중 악성애플리케이션의 93.8% 가 GooglePlay에서 조회가 불가능 한 메타데이터를 가지고 있는 것으로 나타났으며. 조회가 가능한 나머지 6.176% 중에서도 단 1개, 즉 전체의 0.108%만이 라이브 데이터와 유사한 출력 값을 가진다.

Live-Data를 조회할 수 있었던 나머지 6.176%의 항목들 중 Meta-Data와 비교했을 때 일치하는 값을 가지는 경우는 Chart. 3. 과 같다. 이중에서 동시에 4가지 항목을 모두 만족시킨 파일은 단 1개로 전체의 0.108% 이었다. 하지만 이 파일은 오탐으로 인해 악성으로 분류 되었었던 잘못 수집된 샘플이었으며 Virustotal에서 재검사 해 보아도 55개의 백신이 모두 안전한 파일로 결과를 냈다.

실험결과중 정상 파일은 모두 Live-Data가 가능했고 경우 SDK Version, 버전네임, 버전코드 모두 일치했다. Chart. 4. 의 내용과 같이 False-Negative 로 분류된 4건은 파일 사이즈가 달랐으며 이중 한 개는 Permission의 개수도 달랐다.

False-Negative가 발생하는 이유는 Live-Data를 요청할 때 특정 기기의 id를 가지고 조회를 하기 때문에 해당 기기에 설치 가능한 APK파일의 Live-Data가 조회되기 때문이다. GooglePlay에서는 기기의 종류마다 (스마트폰인지, 태블릿인지,

Table 3. Statistics of the Filtered Malware Sample with Live-Data

item	count	percentage
SDK version matching	583	3.915 %
file size matching within 5 percent of errors	216	1.452 %
App version matching	155	1.040 %
user-permissions matching	67	0.452 %
matching of all Meta-Data (SDk version, App version, file size, user-permissions)	1	0.108 %

Table 4. Statistics of the Filtered NormalAPK Sample

	count	percentage
Retrieval Failure from Live-Data	0	0%
SDK Version	0	0%
VersionName	0	0%
VersionCode	0	0%
Permissions	1	1%
FileSize	4	4%
AnySuspicion	4	4%

스마트TV인지), 화면의 해상도마다(dpi), OS의 버전, 심지어 특정 기기모델 마다 서로 다른 APK 파일을 배포할 수 있는 기능이 있는데 이 실험의 경우 APK를 다운받은데 사용한 스마트폰과 검색에 사용한 android_id의 기기가 서로 다른 모델이기 때문이다.

이런 문제는 Google이 기기별, 버전별로 Live-Data를 제공하는 공식 API를 제공하면 False-Negative를 해결할 수 있다.

V. 결 론

본 논문에서는 스마트폰과 같은 모바일 기기에서 악성코드를 백신을 운용하기위한 서버나 바이러스 패턴 없이 작은 비용으로 효과적으로 검사 하고 설치를 방지할 수 있는 필터를 제안하였고 실험을 통해 실용가능함을 보였다.

만약 GooglePlay측에서 기기별, 버전별로 배포되었던 모든 APK의 Live-Data를 조회할 수 있게 API등의 인터페이스를 제공한다면 완벽한 무결성을 제공할 수 있게 되어 False-Negative를 줄일 수 있을 것이며, 동작 속도도 개선할 수 있을 것으로 기대한다.

Android 뿐만 아니라 MAC, Windows, Linux진영 에서도 각자의 ESD(Electronic Software Delivery)를 운영하고 확대 해 나가고 있기 때문에 Androfilter의 모델을 확장하여 Desktop 환경에서도 유사하게 유통의 주체가 있는 컨텐츠의 위, 변조를 검사할 수 있는 방법으로 사용할 수 있을 것이다.

References

[1] eMarketer, "Worldwide Smartphone Usage to Grow 25% in 2014." (2014). http://www.emarketer.com/Article/Worldwide-Smartphone-Usage-Grow-25-2014/1010920

[2] FireEye, "Ghost-Hunting With Anti-Virus." 2014.05, "FireEye Blog", https://www.fireeye.com/blog/executive-perspective/2014/05/ghost-hunting-with-anti-virus.html

[3] Sahs, Justin, and Latifur Khan. "A machine learning approach to android malware detection." In Intelligence and Security Informatics Conference (EISIC), 2012 European, pp. 141-147. IEEE, Aug. 2012.

[4] Burguera, Iker, Urko Zurutuza, and Simin Nadjm-Tehrani. "Crowdroid: behavior-based malware detection system for android." In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, pp. 15-26. ACM, Oct. 2011.

[5] Zhou, Yajin, and Xuxian Jiang. "Dissecting android malware: Characterization and evolution." In Security and Privacy (SP), 2012 IEEE Symposium on, pp. 95-109. IEEE, May. 2012.

[6] Wu, Dong-Jie, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee, and Kuo-Ping Wu. "Droidmat: Android malware detection through manifest and api calls tracing." In Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on, pp. 62-69. IEEE, Aug. 2012.

[7] Yan, Lok-Kwong, and Heng Yin. "DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis." In USENIX security symposium, pp. 569-584. Aug. 2012.

〔8〕 Isohara, Takamasa, Keisuke Takemori, and Ayumu Kubota. "Kernel-based behavior analysis for android malware detection." In Computational Intelligence and Security (CIS), 2011 Seventh International Conference on, pp. 1011-1015. IEEE, Dec. 2011.

〔9〕 Grace, Michael, Yajin Zhou, Qiang Zhang, Shihong Zou, and Xuxian Jiang. "Riskranker: scalable and accurate zero-day android malware detection." In Proceedings of the 10th international conference on Mobile systems, applications, and services, pp. 281-294. ACM, June 2012.

〔10〕 Zhou, Yajin, Zhi Wang, Wu Zhou, and Xuxian Jiang. "Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets." In NDSS. Feb. 2012.

〔11〕 Pack Seoungsoo, Hwansoo Han , "Similarity Detection for Large Scale Software Using Abstracted Source Code." Journal Korea Information Science Society, 39(1A), 39-41. 2012

〔12〕 Soongchunhang University Industry Academy Cooperation Foundation "Method for Detecting Malicious Code by Permission Management" Korea Patent 1013866050000, 2014.04.11

〔13〕 Jang, Jae-wook, Jaesung Yun, Jiyoung Woo, and Huy Kang Kim. "Andro-profiler: anti-malware system based on behavior profiling of mobile malware." In Proceedings of the companion publication of the 23rd international conference on World wide web companion, pp. 737-738. International World Wide Web Conferences Steering Committee, Apr. 2014.

〔14〕 www.android.com, "Platform Versions" 2015.08.03. , https://developer.android.com/about/dashboards/index.html?utm_source=suzunone#Platform

〔15〕 Sato, Ryo, Daiki Chiba, and Shigeki Goto. "Detecting android malware by analyzing manifest files." Proceedings of the Asia-Pacific Advanced Network 36 (2013): 23-31. Dec. 2013.

〔16〕 Morrissey, Michael Gerard, Richard Cannings, Joseph Benjamin Gruver, Angana Ghosh, Jonathan Bruce Larimer, Andrew Devron Stadler, Panayiotis Mavrommatis, Niels Holger Gerhard Konstantin Provos, and Adrian Ludwig. "Protecting users from undesirable content." U.S. Patent Application 13/633,093, filed October 1, 2012.

〔17〕 Anti-Virus Comparative File Detection Test of Malicious Software including false alarm test 2015.10.15. http://www.av-comparatives.org/wp-content/uploads/2015/10/avc_fdt_201509_en.pdf

〈저자소개〉

양 원 우 (Wonwoo Yang) 학생회원
2013년 2월: 국민대학교 전자공학부 졸업
2013년 3월~현재: 국민대학교 보안-스마트전기자동차학과 석사과정
〈관심분야〉 네트워크, 기계학습, 정보보호, 암호

김 지 혜 (Jihye Kim) 정회원
1999년 2월: 서울대학교 컴퓨터공학부 졸업
2003년 2월: 서울대학교 전자컴퓨터공학부 석사
2008년 8월: UC Irvine Dept. of Information & Computer Science 박사
2008년 9월~2008년 11월: UC Irvine Post Doc.
2008년 11월~2011년 8월: 서울대학교 수학연구소 Post Doc.
2011년 9월~현재: 국민대학교 전자공학부 조교수
〈관심분야〉 정보보호기술, 응용암호, 분산시스템