불법 복제 앱 전송 탐지 시스템 구조

김성민*, 김은회*, 최재영*
*숭실대학교 컴퓨터학부
e-mail : smkim@ssu.ac.kr

Illegal apk file detectioning system architecture

Sungmin Kim*, Eunhoe Kim*, Jaeyoung Choi*
*Dept. of Computer Science, SoongSil University

요 약

스마트폰이 일반화되면서 앱 시장의 활성화와 함께 불법 앱 배포 및 사용으로 인한 많은 문제들이 발생하고 있다. 본 논문에서는 이러한 불법 앱의 배포를 막고 저작권을 강화하기 위한 하나의 방법으로써 네트워크에서 전송 중인 불법 앱을 탐지하고 신고하는 불법 앱 전송 탐지 시스템을 제안한다. HTTP 와 FTP 같이 파일 전송에 많이 사용되는 통신 프로토콜을 대상으로 패킷 스니핑, 분석, 조합 과정을 통해 네트워크로 전송중인 앱 파일을 복원하며, 내부에 삽입된 워터마크를 추출하여 저작권 정보를 확인함으로써 불법 앱 전송 여부를 판단한다. 본 논문에서 제안하는 불법 복제 앱 전송 탐지 시스템은 네트워크 디바이스 수준에서 패킷 스니핑을 수행하여, 불법 앱 전송 탐지를 위한 시스템의 성능 저하를 최소화한다. 또한 스니핑한 패킷들을 효율적으로 조합하고 앱 파일 여부를 식별할 수 있는 기준을 제안하며, 중앙 신고 서버를 통한 탐지 결과 확인을 위한 서비스를 제공한다.

1. 서론

일반적으로 파일 형식의 정보데이터는 손쉽게 복사 및 복제가 가능하며, 사용자들의 데이터 저작권 보호 의식이 점점 약화됨에 따라 고속 네트워크 환경을 통한 손쉬운 배포를 통해 불법 복제 데이터의 무분별한 사용 문제가 더욱 더 증가하고 있다. 불법 복제 파일로 인한 이러한 문제는 정보 데이터의 저작권 보호를 저하시키고, 정상적인 데이터 유통을 통한 수익을 감소시킴으로써 개발자의 새로운 정보 데이터 창출을 방해하고 있다. 또한 일부 악의적인 코드가 삽입된 불법 복제 데이터 배포를 통해 사용자의 시스템에 문제를 발생시키고 개인 정보와 같은 고유 정보를 강제로 유출하여 악용하는 사례도 발생하고 있다. 현재 급속도로 보급되고 있는 스마트폰 어플리케이션(이하 앱)에서도 불법 복제 앱으로 인한 문제들이 발생하고 있으며, 점차 스마트폰의 사용량이 증가됨에 따라 심각성이 더욱 커지고 있다. 그 중 한 예로, 대표적인 스마트폰 운영체제인 안드로이드는 구글의 오픈 지향적인 성격을 가지고 있어 데이터 파일을 메모리에 손쉽게 이동할 수 있는데, 불법 복제된 앱 역시 이동가능하며 설치 역시 아무런 제재없이 손쉽게 가능하다. 불법 적인 사용을 제재하고자 시스템 메모리 영역에는 함부로 접근하지 못하도록 안드로이드 자체적으로 보호하고 있으나, 루팅(rooting)이라는 관리자 권한을 획득하여 데이터를 추출 또는 개발자를 위한 SDK 도구 중 하나인 adb 를 이용하여 강제로 불법 추출할 수 있다. 일부 백신 프로그램과 같은 감시 프로그램들은 불법 앱 또는 유해 앱에 대한 판별 및 차단 기능을 제공하여 유해 앱으로 인한 문제점을 해결하고자 하고 있으나 이는 정확성이 떨어지고 수동적인 시스템으로써 사용자에 의해 강제로 실행이 정지될 수 있다는 약점을 가지고 있다. 또한 앱을 설치하는 과정에서 안드로이드 시스템 자체에 접근하여 불법 복제 앱을 강제로 차단하는 연구가 진행되었으나 제조사와 연관된 문제가 발생가능하고 이미 보급화된 스마트 폰에 대한 호환 문제로 인해 추가 비용 문제가 발생할 수 있다는 문제점을 가지고 있다.

본 논문에서는 불법 복제된 앱 데이터가 네트워크를 통해 전송되는 것을 탐지함으로써 이러한 현식 제약적인 문제를 회피하고 불법 복제 안드로이드 앱의 사용을 줄이기 위한 저작권 보호 방법을 제시한다. 불법 데이터 유출자 추적을 위한 기술 등과 같이 정보 데이터에 시그니쳐 기법을 적용한 고유 정보 삽입기술을 이용하여 전송 탐지된 앱 데이터의 불법 전송 여부를 판별하며, 패킷 조합을 통해 얻어진 오브젝트 파일이 앱 파일인지 아닌지 판별할 수 있도록 안드로이드 앱 파일이 가지는 고유한 특징점을 분석한다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구를 살펴볼 것이며, 3 장에서는 전송되는 앱 파일에 대한 특징점을 살펴보고, 4 장에서는 3 장에서 제시한 특징점을 적용하여 불법 앱 전송을 탐지하기 위한 전송 탐지 시스템을 제시한다. 5 장에서는 불법 앱 전송 탐지 시스템을 이용한 실험 결과를 제시하며, 6 장의 결론을 통해 논문을 매듭짓는다.

2. 관련연구

네트워크 모니터링을 통해 전송되는 앱이 불법인지 판별할 수 있는 기술 연구는 아직까지 미흡한 상황이지만, 불법 컨텐츠를 추적하거나[1] 특정 응용 어플리케이션에 대한 트래픽을 분석하는[2][3] 연구들이 진행되어 왔다.

[1]에서 제안하는 불법 콘텐츠 추적시스템은 웹 페이지, P2P, 웹 하드 등에서 디지털 콘텐츠를 키워드를 통해 탐색하고 수집하여 특징점 기반의 식별 코드를 이용함으로써 해당 콘텐츠인지 확인한 뒤, 핑거 프린팅 정보를 추출함으로써 불법 배포자를 추적한다. 본 논문은 네트워크에서 무작위로 전송되는 모든 종류의 패킷들을 대상으로 수행한다는 점에서 [1]과 차이점을 보이고 있으나, 컨텐츠를 식별해

내고 불법 여부를 판별하기 위한 특징점 기반의 식별 방법과 핑거프린팅과 같은 시그네쳐 기술을 활용한다는 면에서 유사하다.

[2][3] 에서 제시하는 네트워크 트래픽 분석 도구들은 특징점 기반의 분석 방법을 통해 정확한 트래픽 분석 수행을 제안하고 있다. 특징점 기반의 분석 방법이란 특정 응용프로그램에서 발생하는 트래픽을 수집하여 해당 트래픽에서 나타나는 고유한 특징을 특징점으로 추출하고 분석 대상 트래픽과 비교함으로써 응용을 판단하는 방법이다. [2][3]에서 제안하는 트래픽 분석 도구들은 네트워크상에서 무작위로 전송되는 패킷들을 수집하고 조합하여 특징점을 기반으로 식별해 낸다는 점에서 본 연구와 매우 유사하다. 그러나 [1][2][3]은 모두 특정한 컨텐츠의 불법 여부와 특정 네트워크 응용에 대한 식별 방법을 제안하고 있어 네트워크로 전송되는 안드로이드 앱에 대한 불법 전송 여부 판별을 위해서는 안드로이드 앱의 특징점을 연구하고 이를 기반으로 네트워크 패킷 분석 연구가 필요하다.

따라서 본 논문에서는 apk 파일의 고유한 특징점을 추출하여 분석한 뒤, 네트워크에서 전송되는 안드로이드 앱 파일에 대한 전송 여부를 판별할 수 있도록 한다.

3. 안드로이드 앱 파일의 고유 특징점

안드로이드 앱 설치 파일은 apk 라는 확장자를 가진다. 구글에서 제공하는 개발도구에 의하여 패키지화 되는 안드로이드 파일은 jar 파일 형식으로 압축된다는 특징과 모든 안드로이드 앱에 대하여 공통적으로 고유한 폴더 구조 및 파일 이름을 가진다는 특징이 있다. 이러한 특징을 이용하여 오브젝트 파일에 대한 앱 파일 판별을 시도한다.

3.1 안드로이드 앱 압축 방식

안드로이드 개발도구에 의하여 안드로이드 어플리케이션이 패키징될 때, jar 파일의 압축 형식이 사용된다. 자바 라이브러리 클래스를 묶어서 배포하기 위해 일반적으로 사용되는 jar 압축방식은 zip 방식의 압축구조를 사용하고 있으므로, 결과적으로 안드로이드 설치 파일은 하나의 zip 형식을 따르는 압축파일이다. 이러한 특징을 이용하기 위하여 zip 파일의 구조를 살펴보면 다음과 같다. Zip 파일의 구조는 Local Header 영역, Centural header 영역, End of Record 영역으로 크게 3 개의 영역으로 구분된다. 첫 번째 영역인 Local Header 영역은 압축 파일에 포함된 각가의 세부 파일들이 실제로 압축된 데이터 필드와 이를 위한 헤더가 나열되어 있는 영역이다. 각각의 파일을 나타내는 local header 의 세부 영역들은 [50 4B 03 04]라는 고유 코드로 시작하며, 각각의 헤더 부분에 압축된 파일 이름, 압축률, 파일 크기 등의 세부 정보를 담고 있다. 두번째 영역인 Central header 영역에는 이전 Local header 영역의 각각의 세부 영역들의 시작 위치(offset)와 크기, 추가 헤더 정보등을 모두 순서대로 나열하여 저장하고 있으며 [50 4B 05 06]이라는 고유 코드로 시작한다. 마지막으로 End of record 영역은 전체 Local header 의 길이 값(Centural header 의 시작 위치), 전체 파일 크기, CRC 코드, 압축 파일의 개수 등 파일의 전체적인 구조 정보를 포함하고 있으며, [50 4B 01 02]라는 고유 코드로 시작한다. 이처럼 zip 파일 구조는 헤더 데이터가 마지막에 위치하고 있으며 각 고유 코드 영역으로 시작하고, 각각의 파일에 대해 독립적으로 압축하여 정보를 헤더 내부에 명시하고 있다.

(1) Local header 는 zip 파일 구조의 첫 시작이므로 apk 파일은 [50 4B 03 04]로 시작한다. 또한 [50 4B 01 02], [50 4B 01 02]와 같은 고유 코드 값이 반드시 함께 포함되어야 한다.

Local header 의 경우, 실제 데이터는 압축되어 기록되지만 헤더의 파일명 필드는 압축되지 않은채로 그대로 기록되는데 이특징을 이용하여 안드로이드 앱에 대한 판별을 수행한다.

3.2 apk 파일의 고유 스트링

안드로이드 어플리케이션은 구글에서 제공하는 개발도구에 의하여 패키징된다. 이 때 개발자는 패키지 구조를 안드로이드에서 명시하고 있는 구조에 따라 어플리케이션을 작성하여 패키징 툴을 통해 apk 확장자 파일을 생성하는데, 이러한 과정에서 모든 안드로이드 어플리케이션은 동일한 내부 구조를 가지게 된다. 개발자에 의해 결정되는 패키지 명이나 정보는 모두 Manifest.xml 파일에 기록되며, 커스텀하게 생성된 실행클래스 파일들은 전부 classes.dex 파일로 결합되어 포함된다. 첨부했던 리소스 데이터들은 res 하위 구조에 각각의 위치에 맞게 저장되며 정보를 담고 있는 resource.rsc 가 함께 생성된다. 이를 정리하면 [표 1]과 같다.

(2) <표 1>apk 파일이 가지는 고유 특징점(feature)

고유 스트링	사용 목적
res/	앱 제작시 사용된 실제 리소스 파일들이 각 기능에 맞추어 폴더 구조에 담겨 있음.
assets/	앱 내부 static 한 저장 공간. 동적인 사용 불가
AndroidManifest.xml	앱에 대한 속성 및 설정 값을 모두 포함
resources.arsc	각각의 리소스 파일에 대한 링크 데이터.
classes.dex	실행에 필요한 java 코드들이 모두 병합되어 하나의 파일로 변형.

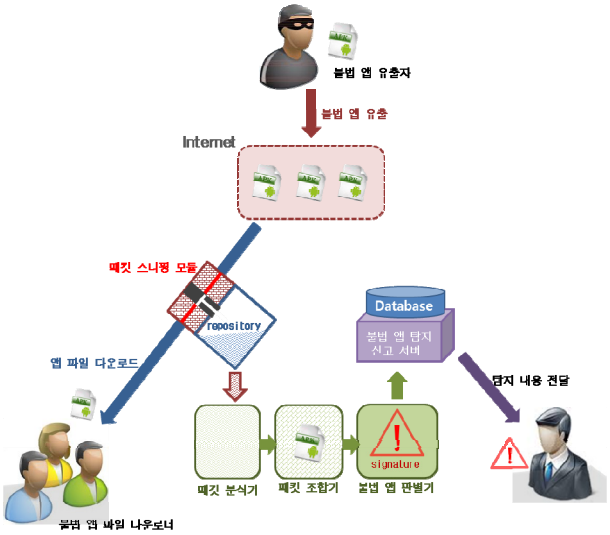
위에서 살펴본 zip 파일 형식을 가지는 특징 (1)과 apk 가 가지는 고유스트링 (2)를 이용하여 앱 파일에 대한 판별을 수행할 수 있다. 앱 설치 파일의 이름과 확장자가 바뀌더라도 바이너리 타입의 데이터를 가지고 판별할 수 있는 방법으로써 사용성이 높으며, 압축 파일 내부의 실제 파일명이 바뀌거나 데이터가 교체되는 등 원본 데이터에 변형이 생길 경우, 안드로이드 시스템 자체에서 실행하지 못하므로 변경할 수 없는 정보임을 보장받을 수 있다.

4. 불법 앱 전송 탐지 시스템

3 에서 제시한 특징점을 기반으로 한 앱 파일 판별 방법을 이용하여 네트워크에서 존송되는 앱 파일 탐지를 수행한다. 본 논문에서 제시하는 탐지 시스템은 크게 패킷 스니핑 모듈, 패킷 분석 모듈, 패킷 조합 모듈, 불법 판별 모듈, 그리고 신고 모듈로 구성된다.

첫 번째, 패킷 스니핑 모듈은 네트워크에서 현재 전송중인 패킷을 네트워크 디바이스 수준에서 모두 스니핑하여 레퍼지토리에 저장한다. 두 번째, 패킷 분석 모듈은 레퍼지토리에 저장된 패킷 헤더와 데이터 필드를 분석하여 데이터 베이스에 저장한다. 세 번째, 패킷 조합 모듈은 데이터 베이스에 저장된 패킷 헤더 정보들을 참조하여 데이터

들을 그룹화 하고 순서를 정렬하여 각각의 후보군 오브젝트로 조립한다. 네 번째, 워터마킹 추출 모듈은 후보군 오브젝트에서 워터마킹 정보를 추출하여 불법 앱 전송 여부를 판단하며, 마지막 다섯 번째인 탐지 신고 모듈을 통해 탐지 내용을 중앙 신고 서버에 전달하고 모니터링 툴을 통해 결과를 제공한다.



(그림 1) 불법 앱 전송 탐지 시스템 구조

4.1 패킷 스니핑

패킷 스니핑 모듈은 네트워크 디바이스에 도착한 모든 패킷을 강제로 복사하여 레퍼지토리에 저장하는 모듈이다. 패킷의 검출률을 높이기 위해 게이트웨이의 위치에 설치하며, 스니핑 과정에서 발생할 수 있는 네트워크 속도 저하 문제를 최소화 하기 위해 네트워크 디바이스 수준에서 패킷 스니핑을 수행한다.

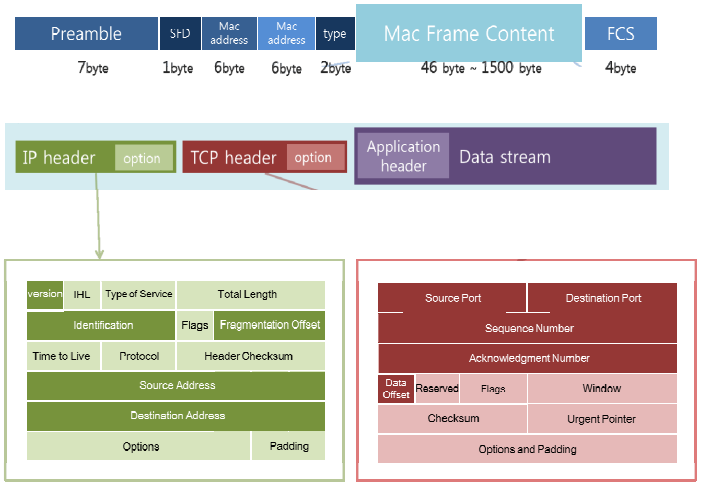
리눅스 커널에서는 네트워크 모듈이 다양한 드라이버가 동일하게 사용 할 수 있도록 공용 라이브러리 형태로 제공되고 있다. 따라서 다양한 네트워크 디바이스 환경에 대해 호환성을 높일 수 있도록 스니핑 모듈을 네트워크 공용 모듈의 최하위에 위치시킴으로써 실제 네트워크 디바이스 수준에서 스니핑을 수행하고 발생할 수 있는 속도 저하문제를 최소화 시킨다.

4.2 패킷 분석 모듈

패킷 분석 모듈은 스니핑 단계에서 얻어진 바이너리 타입의 패킷 데이터들을 분석하여 데이터 베이스에 저장하는 모듈이다. 이 때, HTTP, FTP 프로토콜의 사용 여부 판단하고 그 외의 불필요한 패킷 헤더 정보들을 제거한다. 이 후 조합단계에서의 효율을 높이기 위해서 IP, TCP 헤더 요소 중 패킷 조합에 필요한 요소들은 (그림 2)와 같다.

IP 헤더에서는 먼저 IP4v 가 아닌 모든 패킷을 제외하기 위해 버전 필드를 확인한다. 이어지는 헤더 길이 값을 참조하여 TCP 헤더 시작 위치를 계산한다. 패킷의 전송 순서를 정렬하기 위해 identification 과 fragmentation offset 번호를 분석하며, 프로토콜 필드 값을 참조하여 TCP 이외의 모든 패킷을 제외한다. 마지막으로 source address 와 destination address 를 추출한 뒤, TCP 헤더 시작 위치로 이동한다.

TCP 헤더에서는 Source port 와 Destination port 번호를 추출하고, 패킷 데이터의 중복 문제 및 순서 정렬을 위하여 sequence number 와 acknowledgement number 을 추출한다. 마지막으로 tcp 헤더의 길이 값을 참조하여 데이터 컨텐츠 필드의 시작 위치로 이동한다.

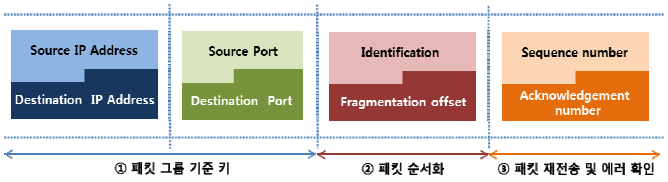


(그림 2) IP,TCP 헤더 추출 요소

4.3 패킷 조합 모듈

패킷 조합 모듈은 데이터베이스에 저장된 패킷 정보들을 이용하여 각각의 후보군 오브젝트들로 조합한다.

첫 번째 단계인 패킷 그룹핑은 패킷들을 각각 해당하는 데이터 그룹으로 묶어내는 과정이다. IP 헤더의 source IP 와 destination IP 를 이용하고, TCP 헤더의 source port 와 destination port 를 패킷들을 그룹화 한다. 두 번째 단계인 패킷 오더링 과정에서는 그룹화 된 패킷들의 순서를 정렬하여 데이터 컨텐츠 필드를 하나의 오브젝트 파일로써 조합하는 단계이다. IP 헤더의 identification number 와 fragment offset 을 이용하여 1 차적으로 패킷의 순서를 정렬하며, TCP 헤더의 sequence number 와 acknowlegement number 를 이용하여 패킷 중복 및 재전송 문제를 해결한다.



(그림 3) 그룹화 기준키와 순서화 기준키

패킷 그룹들을 조합한 후 오브젝트 파일을 대상으로 앱 판별 과정을 수행하기 위하여 3 번에서 살펴본 앱 파일 판별 방법을 적용한다.

4.4 워터마킹 추출기

패킷 조합기의 결과물인 앱 오브젝트 파일에서 내부 dex 파일에 기록된 워터마크 정보를 추출한다. 워터마크 정보에는 앱 개발자에 대한 정보가 삽입되어 있으며, 이를 추출을 위하여 불법 앱 전송 탐지 시스템에서는 AWL 1.0(Android Watermark Library)를 이용하였다. 불법 앱 전송에 대한 판별을 수행하기 위하여 해당 오브젝트 조합시 패킷 그룹의 source ip address 를 참조한다. 공식적으로 마켓을 통해 구입하여 사용자에게 전송되는 앱 파일이 아닌 경우 모두 비공식적인 불법 전송 앱에 해당하므로, 공식적인 마켓의 ip 가 아닐 경우 모두 불법으로 판단한다.

4.5 탐지 신고 모듈

탐지 신고 모듈은 워터마킹 추출기로부터 추출된 개발자 정보와 조합된 오브젝트 파일의 네트워크 전송 정보를 포함하여 중앙 신고 서버에 전달한다. 전달하는 네트워크 전송 정보에는 source ip address 와 destination ip address, 프로토콜의 종류 그리고 전송 날짜 및 시간이 포함된다. 전달된 신고 데이터는 중앙 신고 서버 내부 데이터베이스에 누적되어 보관되며, 웹 페이지 형식의 모니터링 툴을 사용하여 현재까지의 불법 앱 탐지 신고 내용을 확인할 수 있다.

5. 실험 결과 및 분석

불법 앱 전송 탐지 시스템을 실험하기 위해 사용된 머신은 Inter(R) core(TM)2 DUO CPU E8400 (3.00GHz, 4GB)이며, 리눅스 커널 버전은 2.6.32-33 이다. 네트워크 카드는 Intel(R) Pro/1000 PT DUAL PORT 20GByte 를 사용하였으며, 네트워크 디바이스 드라이버는 e1000e 이다. 준비된 머신에 라우터를 설치한 후 테스트용 클라이언트를 통해 FTP, HTTP 프로토콜을 사용하여 1 시간 동안 무작위로 워터마크가 적용된 apk 파일의 전송 요청을 시도하였다.

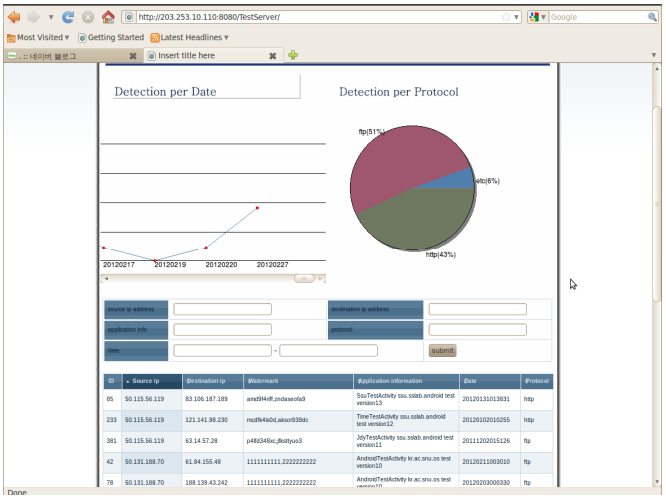
실험 결과는 <표 2>와 같이 총 400 개의 앱 파일을 전송했을 때 조합에 성공한 apk 파일의 수는 385 개로 96.2%의 성공률을 보였으며, 워터마크 추출까지 성공한 apk 파일의 소눈 382 개로 95.5%의 성공률을 보였다.

<표 2>실험 결과

실험 결과 항목	결과 값
스니핑 된 총 패킷 수	225420 개
조합된 apk 파일 수	385 개 (96.2%)
추출된 워터 마킹 수	382 개 (95.5%)

실험 결과에서 apk 파일 조합과 워터마킹 추출 성공률이 100%에 이르지 못한 첫번째 원인은 스니핑 모듈의 패킷 복사 속도와 레퍼지토리에 기록하는 속도의 차이로 인한 패킷 손실이다. 두번째 원인은 패킷 조합의 결과물로써 앱 오브젝트 파일이 생성되었으나 데이터 컨텐츠 필드의 일부분이 패킷 기록 중 잘못되어 완전한 apk 을 구성하지 못해 AWL1.0 라이브러리가 인식하지 못할 수 있기 때문이다.

실험 과정에서 전송을 시도한 400 개의 앱은 공식 마켓이 아닌 웹 페이지 서버와 FTP 를 사용하였으므로 모두 불법 앱 전송으로 판별하게 되므로 탐지 신고 모듈에 의하여 중앙 신고 서버에 탐지 내용을 모두 전달하게 된다. 중앙 신고 서버에서 웹 페이지 방식을 이용하여 개발자에게 제공하는 모니터링 서비스는 (그림 4)과 같다.



(그림 4) 불법 앱 탐지 결과 모니터링 페이지

6. 결론

본 논문에서는 불법 복제된 앱 파일의 배포 문제를 해결하기 위한 하나의 방법으로써 네트워크를 통해 전송 되는 불법 앱 전송 탐지 시스템을 제안하였다. 패킷 조합 및 분석 방안을 제안하고 중앙신고 서버를 통해 탐지 결과를 확인할 수 있는 불법 앱 전송 탐지 시스템은 고속 네트워크 환경에서 속도 저하를 최소화 하면서 뛰어난 불법 앱 탐지 기능을 수행할 수 있다. 이를 위해 네트워크 인터페이스 레이어에서 패킹 스니핑을 수행하며, 패킷의 분석, 조합 및 앱 판별, 워터마킹 추출 과정을 통해 불법 앱 전송 여부를 판별하는 방법을 제안하였으며, 중앙 신고 서버와 모니터링 페이지를 이용하여 개발자에게 불법 앱 유출 사실을 전달하였다. 이러한 불법 앱 전송 탐지 시스템을 통해 불법 앱 파일의 배포 상황을 파악함으로써 불법 앱 배포에 대한 방어책 마련과 안드로이드 마켓의 투명성 및 개발자의 저작권을 강화하는데 이용 할 수 있을 것이다. 추후 데이터 전송과 관련된 다양한 프로토콜에 대해 적용 가능하도록 패킷 분석 및 조합 과정을 연구하고 핑거프린팅 추출 성공률을 높이기 위해 판별 기술을 고도화 할 것이다.

참고문헌

[1] 정혜원, 이준석, 서영호, "불법콘텐츠 추적 기술 연구 동향", 전자통신통향분석 제 20 권, 제 4 호, pp. 120-128, 2005 년 8 월.

[2] 박준상, 윤성호, 김명섭, “탐색 공간 최적화를 통한 시그니쳐기반 트래픽 분석 시스템 성능 향상” , 한국인터넷 정보학회, 12 권, 3 호, 2011 년 4 월

[3] 박준상, 박진완, 윤성호, 이현신, 김명섭, “응용 레벨 트래픽 분류를 위한 시그니쳐 생성 및 갱신 시스템 개발 “, 정보처리학회논문지, 17-C 권, 제 1 호(20102), 2009 년 10 월

[4] Bruce A. Harvey, Stephen B. Wicker, "Packet Combining Systems Based on the Viterbi Decoder", IEEE Transactions on communications, VOL.42, NO. 2/3/4, 1994.

[5] Frederick J. Block, "Packet Combining Considerations for Random-Access Networks", Military Communications Conference, pp.133-139, 2010.

[6] 정상준, 권영헌, 최혁수, 이정협, 김종근, "실시간 망 관리를 위한 패킷 분석 시스템의 설계 및 구현", 춘계학술발표논문집, pp. 3-671, 2001 년 6 월.

[7] 김원규, 김승휘, 이재혁, 황태철, 추영열, "리눅스 기반 호스트 IPS 구현", 추계학술발표논문집, pp. 1-967, 2006 년 11 월.

[8] 변상익, 함유식, 김정인, 설순욱, 김명철, "리눅스에서의 분할패킷 재조합 성능 개선 “, 한국정보과학회 추계학술대회, 제 27 권, 2000 년 10 월.

[9] 김명섭, 강훈정, 홍원기, “Flow Grouping 을 통한 P2P 트래픽 분석 방법에 관한 연구” , Proc. of KNOM 2003 Conference, Daejeon, Korea, pp.210-218, May 22-23, 2003.