보안공학연구논문지 *Journal of Security Engineering*
Vol.11, No.6 (2014), pp.551-566
http://dx.doi.org/10.14257/jse.2014.12.11

TCP-Flow 시퀀스를 이용한 네트워크 행위 기반 안드로이드 악성코드 탐지

성명재[1], 박해룡[2], 최보민[3], 임을규[4]

Android Malware Detection Using TCP-flow Sequence on Network Behavior-based

MyeongJae Seong[1], Haeryong Park[2], Bomin Choi[3], Eul Gyu Im[4]

요 약

안드로이드 악성코드는 빠르게 증가하고 있다. 많은 안드로이드 악성코드들은 불분명한 곳으로부터 악성코드를 설치하도록 유도하고 있다. 따라서 안드로이드장치들은 항상 악성코드 위협에 노출되어있다. 안드로이드 운영체제에서 악성코드를 차단하기 위해서 많은 연구가 진행 중이며 이런 연구 중 본 논문에서 네트워크 행위 기반 안드로이드 악성코드 탐지 방법을 연구하였다. 네트워크 행위 기반 안드로이드 악성코드 탐지방법은 악성코드가 행위를 수행할 때 발생되는 네트워크 패킷을 이용하여 특징을 생성하였으며, 이 특징은 TCP-Flow에 패킷 크기를 사상하고 코드변환을 통해 코드를 나열시킨, TCP-Flow 시퀀스(TCP-flow sequence)와 도메인 네임(Domain Name)을 사용해 하나의 특징으로 구성된다. 본 논문에서는 이 특징을 이용해 네트워크 행위 기반 악성코드 탐지 시스템을 구현하였다.

핵심어 : 안드로이드 악성코드, 네트워크 행위, TCP-Flow, Levenshtein distance

Abstract

The number of Android malware is increasing rapidly. Android malware is spreaded through unclear sources or markets. Therefore, Android devices are always exposed to malware threats. In order to prevent malware from damaging the Android operating system, there are many ongoing researches. In this paper, we propose Android malware detection method based on network behavior. this method generates features

접수일(2014년10월28일), 심사의뢰일(2014년10월29일), 심사완료일(1차:2014년11월14일)
게재일(2014년12월31일)

[1]133-791 서울특별시 성동구 왕십리로 222 IT/BT관, 한양대학교 컴퓨터 소프트웨어학과
email: mjseong@hanyang.ac.kr
[2]138-950 서울특별시 송파구 중대로 135 아이티벤처타워, 한국인터넷진흥원 정보보호기술개발팀
email: hrpark@kisa.or.kr
[3]138-950 서울특별시 송파구 중대로 135 아이티벤처타워, 한국인터넷진흥원 정보보호기술개발팀
email: bmchoi@kisa.or.kr
[4](교신저자) 133-791 서울특별시 성동구 왕십리로 222 IT/BT관, 한양대학교 컴퓨터공학부
email: imeg@hanyang.ac.kr
* 본 연구는 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [10044938, 악성코드 프로파일링 및 대용량 보안이벤트 분석을 통한 공격징후 탐지기술 개발]

ISSN: 1738-7531 JSE

using network packets which occur during executions of malware. these features were consisted from a pair of domain names and TCP packet-flow sequences such as TCP flows. We Implemented the proposed method, and experimented with test data.

Keywords : Android malware, Network-behavior, TCP-Flow, Levenshtein distance

1. 서론

안드로이드 악성코드증가 속도는 매우 빠르게 증가하고 있다. 현재 안드로이드 운영체제를 이용하는 사용자가 다른 스마트폰 운영체제 사용자에 비해 빠르게 증가하고 있으며, 안드로이드 운영체제에서는 Google 공식 마켓 외에도 블랙마켓으로부터 앱을 다운로드할 수 있기 때문에 악성코드들이 유입 경로가 다중화 되어 있다. 이외에도 안드로이드 운영체제는 합법적이지 않은 불분명한 곳에서 APK파일을 다운로드받아 어플리케이션을 설치 할 수 있기 때문에 보안이 취약하며, 현재까지도 다수의 악성코드가 생성되고 배포되고 있다. 따라서 안드로이드 운영체제의 취약점을 보완하기 위해 많은 안드로이드 운영체제 환경에서 악성코드를 탐지하는 연구가 활발하게 진행 중이다.

지금까지 제안된 여러 악성코드 탐지 방법들 중에 APK 파일 기반 검사와 권한부여 검사 그리고 행위기반 탐지 방법이 있다.

APK파일 기반 검사방법은 JAVA byte code 단위의 정적 분석을 이용하여 호출되는 API의 특징을 생성하고 탐지하는 방법[1]과 APK파일에 대한 역어셈블을 통해 클래스 단위로 Dalvik 바이트코드 명령어에 대한 빈도수를 추출하여 시그니처를 생성하고 이 생성된 시그니처를 이용해 악성코드를 탐지하는 방법[2]이 연구되었다. 권한부여 검사 방법은 안드로이드 운영체제에서 사용하는 권한부여(permission)을 이용 여부를 분석하여 탐지하는 방법[3]이다. 그리고 행위기반 탐지 방법은 악성코드가 활동하는 행위를 특징으로 생성하고 행위와 특징을 비교하여 탐지하는 방법이다.

악성코드 행위탐지는 안드로이드 운영체제의 자원을 이용할 때 API호출과 메모리사용률, CPU 사용률, PID 및 기타자원에 접근하는 특징을 분석해 탐지 하는 방법과 악성코드가 네트워크를 이용해 데이터를 전송할 때 발생되는 패킷을 분석하여 탐지하는 방법이 연구되고 있다. 이러한 악성코드 행위 탐지 연구 중, FTRACE를 이용하여 안드로이드 운영체제의 커널에서 악성코드가 메모리를 할당하는 것과 네트워크 장치를 이용하는 것을 로그에 기록해 악성코드의 행위를 분석하는 연구[4]가 수행 되었다.

본 논문에서는 악성코드가 행위기반 탐지 방법 중 네트워크 패킷을 분석하여 안드로이드 운영체제에서 활동하는 악성코드의 탐지 방법에 대해서 제안한다.

본 연구의 네트워크 행위기반 탐지 방법은 다수 악성코드를 네트워크상에서 탐지할 수 있고 악성코드가 활동하는 특정 모바일 장치를 탐지하고 신속하게 악성코드 존재여부를 파악할 수 있다. 따라서 악성코드가 발생시키는 패킷을 수집하고 분석된 특징을 이용해 악성코드가 활동하는 네트워크 환경에서 악성코드에 감염된 모바일 장치를 탐지할 수 있는 시스템에 대해 제안한다.

2. 관련 연구

2.1 일반적인 네트워크 기반 악성코드 탐지

일반적인 네트워크 기반 악성코드 탐지 기법은 패킷의 Payload를 분석하고 특정 Payload 내부의 바이너리 값을 특징으로 생성해 악성코드를 탐지하거나 특정 네트워크 Port를 검사하여 악성코드 공격을 탐지하는 방법을 주로 이용한다. 또한 패킷의 프로토콜 정보 중 OSI 7계층의 네트워크 계층과, 전송 계층에 해당하는 프로토콜 기본 정보들을 특징으로 생성하고 군집화한 후 악성코드의 공통점을 찾아 탐지 방법[5][6]이 연구되었다.

현재의 악성코드들이 네트워크를 통해 데이터를 전송할 때 주요 특징은 TCP프로토콜을 이용하면서 일반적인 Port 번호를 위장하기 때문에 간단하게 Port 검사로는 악성코드탐지에 어려움이 있다. 따라서 많은 네트워크 기반 탐지 방법들은 패킷의 속성에 대한 특징을 찾아 악성코드를 탐지하는 방법을 연구하고 있다.

다수의 네트워크 기반 탐지 방법은 OSI 7계층에서 전송 계층의 TCP에서 정보를 추출하고 응용계층에서는 HTTP 프로토콜에서 정보를 추출한다. 일반적인 네트워크기반 악성코드 탐지 방법은 봇넷(Botnet)탐지 방법[7][8]이 다수 제안되었다. 봇넷탐지 방법은 봇넷의 HTTP 프로토콜 데이터패턴정보와 FTP프로토콜을 이용 여부 그리고 특정 취약점이 존재하는 port번호를 이용하는 것을 종합적으로 분석해 탐지한다.

2.2 네트워크 기반 안드로이드 악성코드 탐지

네트워크 기반 안드로이드 악성코드 탐지 방법은 일반적인 네트워크 기반 악성코드 탐지 방법과 유사하지만 각 안드로이드 어플리케이션별 특징을 생성할 수 있어 악성코드가 활동하는 어플리케이션단위로 탐지 할 수 있다.

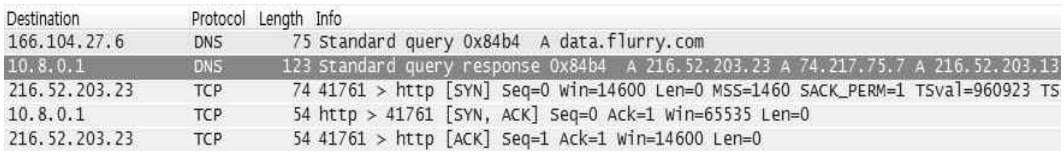
안드로이드 악성코드를 탐지하기 위해 악성코드가 통신을 시도한 패킷을 분석하고 그 분석결과를 통해 모바일장치 정보(IMEI)와 개인정보(주소록)를 유출시키는 악성행위에 대한 보고가[9]되었다. 그리고 안드로이드 악성코드가 발생시킨 패킷에서 패킷 시작시간, 종료시간, 보낸 패킷의 플로우(up flow), 받은 패킷의 플로우(down flow), IP주소, Port번호들의 특징을 주로 사용하여 악성코드를 탐지하려는 네트워크 기반 연구들 중 트로이목마와 같은 악성행위로 발생되는 패킷에서 DNS응답패킷의 정보(RESOURCE SELECTION에 포함된 IP주소와 TTL, 기타정보)를 네트워크 특징으로 이용해 악성코드들의 Geo matrix를 생성하여 ICA분석을 이용한 탐지방법연구[10]와 PID와 네트워크 연결된 시작시간과 종료시간, up flow, down flow, IP주소, Port번호, 프로토콜 종류들을 병합하여 하나의 Vector로 생성한 특징을 이용한 탐지 연구[11]가 기존에 수행 되었다.

ISSN: 1738-7531 JSE

3. 네트워크 기반 안드로이드 악성코드 분석 및 특징 생성

3.1 안드로이드 악성코드 행위 분석

안드로이드 악성코드 비중을 보면 잠재력 위협 프로그램과 트로이목마 그리고 광고 프로그램 이외 기타 프로그램 등으로 구분될 수 있다. 이 분류 중 다수 차지하고 있는 트로이목마는 안드로이드 사용자의 정보를 몰래 유출하는 행위를 하며 사용자의 스마트폰을 제어할 수 있게 서버로부터 명령을 받고 관리자권한 얻는 행위(rooting)를 시도하거나 프로그램을 종료 시켜버리는 행위를 한다. 이런 트로이목마 악성코드는 정보를 유출시키거나 서버로부터 명령을 받기 위해 네트워크 접속을 시도 한다.



Destination	Protocol	Length	Info
166.104.27.6	DNS	75	Standard query 0x84b4 A data.flurry.com
10.8.0.1	DNS	123	Standard query response 0x84b4 A 216.52.203.23 A 74.217.75.7 A 216.52.203.13
216.52.203.23	TCP	74	41761 > http [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=960923 TS
10.8.0.1	TCP	54	http > 41761 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
216.52.203.23	TCP	54	41761 > http [ACK] Seq=1 Ack=1 Win=14600 Len=0

[그림 1] 안드로이드 악성코드의 통신 행위

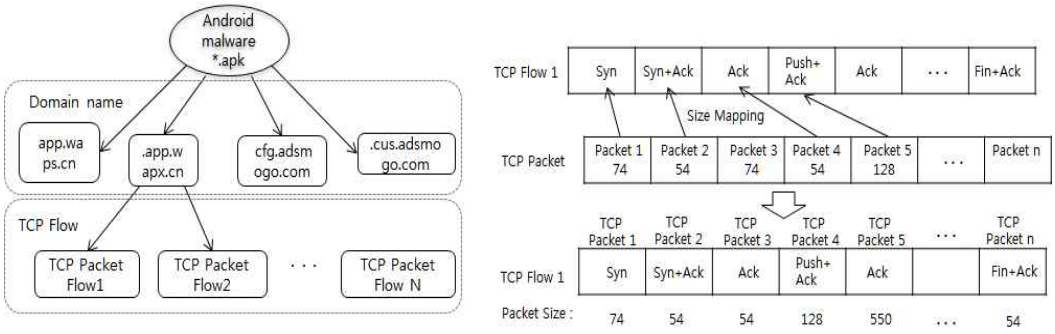
[Fig. 1] Communication behavior of Android malware

대부분의 악성코드들은 그림 1과 같이 DNS프로토콜을 이용하여 서버 IP를 내려 받아 통신을 한다. 따라서 안드로이드 악성코드가 하는 행위들은 도메인네임 질의(domain name query)를 DNS 서버에 요청하고 악성코드 서버가 등록된 도메인 네임과 일치하는 IP주소를 응답으로 받아 통신을 시작하게 된다. 다수의 악성코드들은 이와 같은 방법을 사용해 네트워크 통신을 수행하게 된다.

3.2 안드로이드 악성코드 특징 생성

안드로이드 악성코드를 네트워크 환경에서 탐지하려면 특징을 생성해야 한다. 따라서 안드로이드 악성코드 행위를 특징으로 생성하기 위해 악성코드가 사용하는 DNS프로토콜의 도메인 네임 질의(domain name request)통해 얻을 수 있는 도메인 네임(domain name)을 이용해야 한다.

악성코드가 서버와 통신하기 위해서는 IP주소와 port 번호를 이용하지만 IP주소와 port 번호는 변경되는 빈도가 높아 특징으로 사용하기 어렵다. 그렇기 때문에 위에서 언급한 도메인 네임을 이용해야한다.



[그림 2] 안드로이드 악성코드의 특징

[Fig. 2] The feature of Android malware

안드로이드 악성코드마다 도메인네임 질의 요청을 하기 때문에 유사한 악성코드들은 동일한 도메인네임을 사용할 가능성이 매우 높다. 추가적으로 도메인네임이 악성코드 마다 중복되는 경우를 예상하여 다른 특징을 고려해볼 수 있다. 그 다른 특징으로는 TCP-Flow를 이용하는 것으로 TCP-Flow는 TCP프로토콜이 통신한 시작과 끝을 알 수 있게 하여 TCP패킷의 흐름을 단위별로 파악할 수 있게 한다. 따라서 TCP-Flow를 이용하면 또 하나의 특징을 추가할 수 있다.

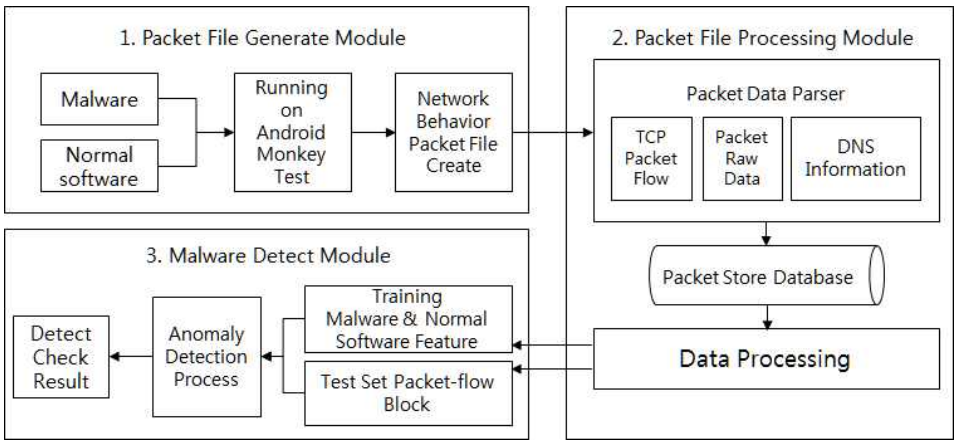
본 논문에서 제안하는 특징은 DNS패킷에서 추출된 도메인 네임과 TCP-Flow에 해당하는 패킷별 크기를 사상(Mapping)하여 TCP-Flow별 형태를 생성한 것을 쌍으로 구성하여 특징으로 생성한다. 패킷 크기를 이용한 기존의 네트워크기반 탐지연구와 다르게, IP주소마다 통신한 전체 패킷 Flow에 해당하는 패킷별 최대, 최소 크기만 특징으로 사용하는 것이 아니라 TCP-Flow에 해당하는 패킷별 크기를 이용해서 TCP-Flow의 형태를 생성한다. 이것은 그림 2.와 같이 안드로이드 악성코드마다 도메인 네임과 TCP-Flow 형태를 쌍을 구성하여 각 안드로이드 악성코드가 발생하는 네트워크의 특징의 구성을 보여주고 있으며, 패킷 크기가 사상된 TCP-Flow를 볼 수 있다.

본 논문의 악성코드 특징 생성에 대한 자세한 과정은 다음 4절의 안드로이드 악성코드 탐지 시스템에서 설명하겠다.

4. 안드로이드 악성코드 탐지 시스템

4.1 안드로이드 악성코드 탐지 시스템 구조

네트워크 기반 안드로이드 악성코드 탐지 시스템은 그림 3과 같이 안드로이드 악성코드가 통신을 통해 발생된 패킷을 수집하는 모듈과 수집된 패킷의 정보를 가공하고 저장하는 모듈 그리고 가공된 패킷의 정보를 악성코드의 특징으로 생성하고 탐지하는 모듈로 구성되어 있다.



[그림 3] 네트워크 기반 악성코드 탐지 시스템 구조

[Fig. 3] Malware Detection System Architecture Based Network

4.1.1 패킷 파일 생성

패킷 파일 생성 모듈은 안드로이드 악성코드와 일반 프로그램의 실행을 통해 프로그램이 통신을 했을 때 발생하는 패킷을 수집하고 그 수집된 정보를 .pcap 포맷파일로 생성시키는 역할을 수행한다.

프로그램 실행은 안드로이드에서 제공되는 ADB shell의 Monkey test 도구를 통해 악성코드와 정상 프로그램의 동작을 자동으로 수행할 수 있다.

4.1.2 패킷 정보 가공 및 저장

패킷 파일 처리는 패킷 데이터 분석기와 추출 데이터 처리로 구분할 수 있다. 패킷이 수집된 .pcap파일을 통해 수집된 악성코드와 정상프로그램의 통신정보를 이용해 특징을 생성하고 저장하는 역할을 수행한다.

패킷 데이터 분석기는 .pcap파일에서 TCP프로토콜의 sequence 번호를 이용해 TCP 패킷의 flow를 생성하고 DNS 프로토콜에서 사용된 도메인 네임과 IP주소를 추출한다. 또한 기본 패킷 정보인 IP주소, Port번호, 패킷크기 그리고 필요한 다른 정보를 추출해 데이터베이스에 저장한다.

추출 데이터 처리는 앞서 패킷 데이터 분석기에서 추출된 정보를 데이터베이스에서 읽고 그 정보를 이용해 안드로이드 악성코드와 정상프로그램의 특징 및 탐지할 대상의 비교 데이터를 생성한다.

4.1.3 악성코드 탐지

악성코드탐지 모듈은 패킷 파일 처리 부분에서 생성된 안드로이드 악성코드와 일반프로그램의 특징과 탐지 대상의 비교 데이터를 이용하여 악성코드의 존재를 탐지하는 역할을 수행한다.

비정상 행위 탐지 처리는 특징이 수집된 정보와 탐지 대상의 데이터를 검사하고 특징과 유사한 정보가 탐지되면 결과를 도출한다.

4.2 안드로이드 악성코드 패킷 생성 과정

안드로이드 악성코드의 패킷을 생성하려면 안드로이드 장치 또는 에뮬레이터에 설치해야 한다. 또한 설치된 프로그램을 실행하기 위해서 직접실행을 수행하고 프로그램의 이벤트를 발생시켜야 한다. 따라서 앞에서 언급된 안드로이드 ADB-Shell의 Monkey test 도구를 이용해 자동화된 패킷 생성 방법을 사용한다.

Monkey test는 안드로이드에 설치된 프로그램을 무작위로 이벤트를 발생시켜주는 test도구로 일정 시간동안 악성코드를 실행시켜 악성코드가 발생시킬 수 있는 이벤트를 수행하게 한다. 이렇게 일정시간동안 악성코드가 실행되어 생성된 패킷을 수집한다.

패킷을 수집하는 방법은 안드로이드에서 지원하는 TCP dump 또는 안드로이드 어플리케이션을 사용해 패킷을 수집할 수 있다. 그리고 수집된 패킷은 .pcap포맷 파일로 변환되어 저장된다.

4.3 안드로이드 악성코드 특징 생성 과정

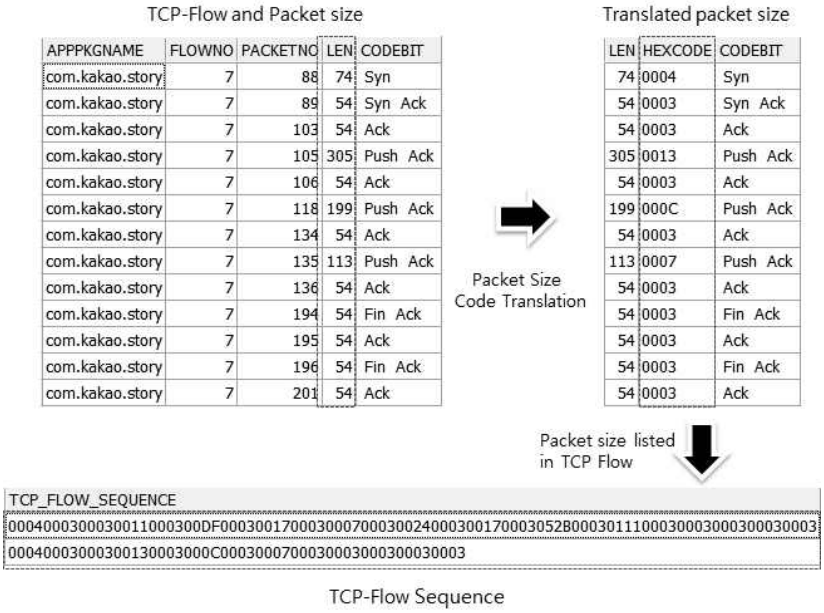
안드로이드 악성코드별 특징을 생성하기 위해서는 2가지 제약이 발생한다. 첫 번째, 악성코드를 단독으로 실행시켜 패킷을 발생시켜야 한다. 그 이유는 다른 악성코드와 같이 실행되면 악성코드가 발생시킨 패킷의 구분이 어렵다. 두 번째, 악성코드를 반복 수행시키고 발생되는 동일한 TCP-Flow가 생성되어 일치해야한다. 악성코드특징은 동일한 TCP-Flow만 사용할 수 있다. 따라서 동일하지 못한 TCP-Flow 경우 특징으로 사용할 수 없다.

패킷 데이터 가공 부분에서 위의 조건에 맞는 악성코드들은 추출된 정보를 이용하여 특징을 생성할 수 있다. 악성코드에서 발생된 DNS프로토콜 패킷에서는 도메인 네임별 IP 주소를 다수 보유하고 있다. 따라서 IP주소마다 통신된 TCP프로토콜의 TCP-Flow를 일치 시킬 수 있다. 그리고 TCP-Flow의 형태를 생성하기 위해 TCP-Flow에 해당하는 패킷 크기를 사상하고 사상된 TCP-Flow와 패킷 크기는 다시 일정한 자릿수 형태를 갖는 코드로 변환한다.

코드변환 과정은 패킷 크기의 10진수 값을 16진수로 변환하여 일정한 자릿수를 유지하는 코드로 변환된다. 그리고 이 변환된 코드를 나열하게 되면 그림4와 같은 나열된 TCP-Flow를 확인할 수 있다. 이렇게 나열된 TCP-Flow를 TCP-Flow 시퀀스라고 본 논문에서 명명하였다.

TCP-Flow 시퀀스와 도메인 네임을 쌍으로 구성한 특징을 이용해 각 악성코드별 구분할 수 있다. 대부분 악성코드들은 도메인네임을 이용해 IP주소를 얻어 통신을 수행한다. 따라서 간단히 도메인 네임만으로 특징으로 사용할 수 있겠지만 악성코드의 다른 그룹에서 동일한 도메인 네임을 사용할 가능성이 존재하기 때문에 TCP-Flow 시퀀스를 특징이 필요하다.

ISSN: 1738-7531 JSE



APPPKGNAME	FLOWNO	PACKETNO	LEN	CODEBIT
com.kakao.story	7	88	74	Syn
com.kakao.story	7	89	54	Syn Ack
com.kakao.story	7	103	54	Ack
com.kakao.story	7	105	305	Push Ack
com.kakao.story	7	106	54	Ack
com.kakao.story	7	118	199	Push Ack
com.kakao.story	7	134	54	Ack
com.kakao.story	7	135	113	Push Ack
com.kakao.story	7	136	54	Ack
com.kakao.story	7	194	54	Fin Ack
com.kakao.story	7	195	54	Ack
com.kakao.story	7	196	54	Fin Ack
com.kakao.story	7	201	54	Ack

LEN	HEXCODE	CODEBIT
74	0004	Syn
54	0003	Syn Ack
54	0003	Ack
305	0013	Push Ack
54	0003	Ack
199	000C	Push Ack
54	0003	Ack
113	0007	Push Ack
54	0003	Ack
54	0003	Fin Ack
54	0003	Ack
54	0003	Fin Ack
54	0003	Ack

TCP_FLOW_SEQUENCE
00040003000300110003000DF0003001700030007000300240003001700030052B0003011100030003000300030003
0004000300030013000300C0003000700030003000300030003

[그림 4] TCP-Flow 시퀀스 생성 과정

[Fig. 4] Sequence generating process of TCP-Flow

또한 안드로이드 악성코드에서 생성된 특징 중 같은 그룹에서 유사한 TCP-Flow 시퀀스도 존재한다. 따라서 이런 TCP-Flow 시퀀스를 특징으로 생성하기 위해 2가지 방법을 사용하였다.

첫 번째는 같은 그룹간의 TCP-Flow 시퀀스를 비교하는 과정으로 동일한 길이의 TCP-Flow 시퀀스를 Levenshtein distance 알고리즘(1)[12]을 사용해 거리를 산정하였으며 유사한 두 시퀀스간의 최소 거리가 1이상의 시퀀스는 제외하는 방법을 사용하였다.

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) \\ \min\begin{cases} lev_{a,b}(i-1,i)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} \end{cases}$$

(1) Levenshtein distance algorithm

특별히 최소 거리 1이상의 시퀀스를 제외 한 이유는 16진수로 변환된 코드간의 실제 패킷 크기 차이가 2자리 이상이면 전송되는 값의 크기가 다르기 때문이다.

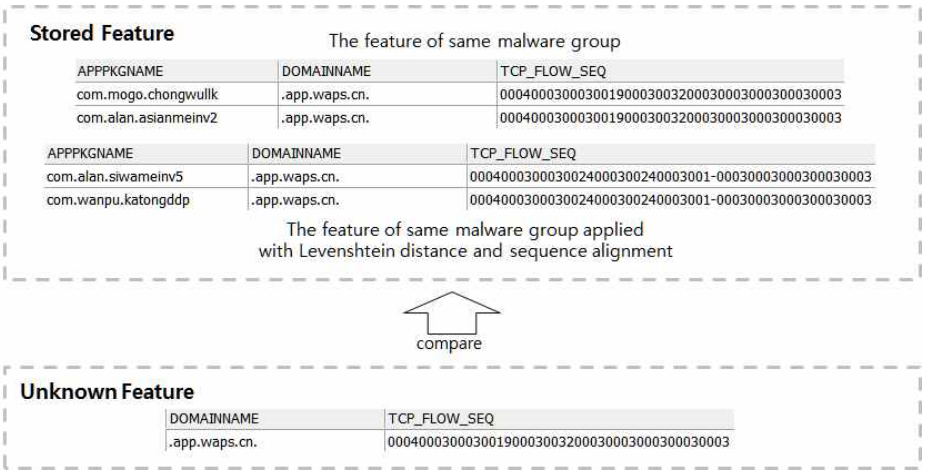
두 번째는 Levenshtein distance로 최소 거리 1인 TCP-Flow 시퀀스를 즉시 특징으로 생성하지 않고 거리차이가 1 발생된 코드를 찾아 Sequence Alignment[13]적용 시켜 특징으로 생성하였다. Sequence Alignment는 유사한 TCP-Flow 시퀀스를 동일한 TCP-Flow 시퀀스로 생성해 줄 수 있도

록 TCP-Flow 시퀀스 대상에서 거리가 1인 16진수 코드를 '-'으로 표기하여 동일한 TCP-Flow시퀀스로 보이도록 한다. 이렇게 생성된 특징은 다음 4.4절 안드로이드 악성코드 탐지과정에서 실제 2가지 방법이 적용된 특징을 그림 5를 통해 확인 할 수 있다.

4.4 안드로이드 악성코드 탐지 과정

안드로이드 악성코드의 특징이 생성되면 이 특징을 이용해 네트워크상의 악성코드가 설치된 장치의 감염여부를 탐지할 수 있다.

악성코드가 설치된 장치에서 패킷을 수집하고 그 수집된 정보는 앞서 설명한 패킷 파일 처리과정을 통해 TCP-Flow 시퀀스와 도메인 네임의 쌍으로 구성된 정보를 그림5.와 같이 얻을 수 있다. 이렇게 얻은 정보를 특징과 비교하면 신속하게 악성코드를 탐지할 수 있다. 특징으로 생성된 TCP-Flow 시퀀스와 도메인 네임을 검사 대상에서 얻은 정보와 일치시켜 일치하는 경우 악성코드를 탐지한 결과를 도출한다.



[그림 5] 실제 같은 악성코드 그룹 별 특징과 지정되지 않은 악성코드 특징 비교

[Fig. 5] Comparison of the feature with stored same malware group and unknown malware

검사대상에서 수집된 정보가 불일치해 미 탐지될 경우 TCP-Flow 시퀀스의 유사도 검사를 한다. 이 방법을 사용하는 것은 동일한 도메인 네임을 사용하고 TCP-Flow 시퀀스의 길이가 같은 경우 악성코드 특징생성과정에서와 같이 시퀀스를 Levenshtein distance 알고리즘과 Sequence Alignment을 사용해 유사한 정보를 얻을 수 있다. 이렇게 얻어진 TCP-Flow 시퀀스는 다시 Sequence Alignment된 TCP-Flow 시퀀스 특징과 비교하게 되며 일치하는 TCP-Flow 시퀀스가 존재하는 경우 악성코드가 감염되었다는 결과를 도출한다.

5. 안드로이드 악성코드 탐지 실험

5.1 실험 환경 및 방법

본 논문에서 제안한 네트워크 기반 안드로이드 악성코드 탐지 시스템의 실험을 하기위해 안드로이드 젤리빈 4.1.1 버전의 genymotion 에뮬레이터와 젤리빈 4.1.2 버전의 갤럭시 S2에 실험환경을 구성하였다. 그리고 안드로이드 악성코드 343개와 정상어플리케이션 11개를 대상으로 패킷을 수집을 시도하였다.

안드로이드 악성코드 샘플 343개중 158개는 통신이 되지 않거나 동작하지 않는 샘플로 확인 되었으며, 실제 패킷이 수집된 샘플의 수는 악성코드와 정상어플리케이션 모두 총 195개이며 샘플에서 일정한 시간 동안에 각 악성코드 패킷을 수집할 수 있게 Monkey test 스크립트를 작성하여 실행하였다. 그리고 Monkey test 횟수는 20회 이벤트 발생 값은 300으로 10분 이상 각 악성코드마다 실행하고 패킷을 수집하였다.

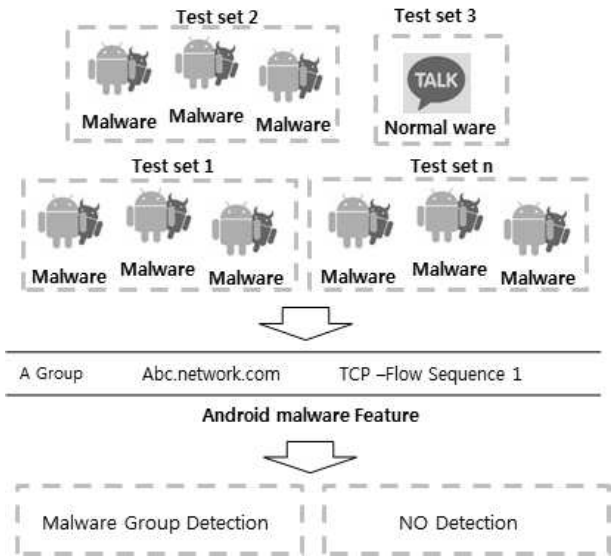
[표 1] 안드로이드 프로그램의 실험 그룹

[Table. 1] test group of Android application

그룹 이름	실험 그룹
Group1	Normal Software
Group2	DroidKungfu Family
Group3	Anserver Family
Group4	DroidDream Family
Group5	GoldDream Family

실험을 위해 우선 특징을 생성할 악성코드를 선정하였고 악성코드는 사전에 구분된 그룹에서 선택하였다. 그리고 표 1.과 같이 안드로이드 악성코드 그룹은 Droidkungfu, DroidDream, Anserver, Golddream이며 정상 어플리케이션 그룹은 google play스토어에서 다운로드받은 어플리케이션으로 구분하였다. 이렇게 구분된 그룹에서 악성코드 특징이 생성될 샘플 일부분을 선택해 특징을 생성하고 정상 어플리케이션 특징은 모두 특징으로 생성하였다.

각 그룹별 샘플에 대한 특징 생성이 완료 된 후 탐지 실험을 수행하기 위해서 그룹별 나눠진 악성코드와 정상어플리케이션을 각 test set으로 그룹화 하였다. 각 test set마다 그룹별 샘플 1개씩 선택하였고 1차부터 3차까지 test set과 training set을 증가 시키는 실험을 수행하였으며, test set 수는 총 40개가 사용되었다.



[그림 6] 안드로이드 악성코드의 실험 구성

[Fig. 6] Experiment composition of Android malware

그림6과 같이 test set은 각각 안드로이드를 사용하는 장치로 볼 수 있으며, test set마다 안드로이드 악성코드가 설치되어 test set 단위로 탐지 여부를 파악할 수 있다. 그리고 test set에 포함된 각 악성코드 그룹별 샘플들이 실제 네트워크에서 발생시키는 패킷처럼 보일 수 있도록 test set별 악성코드를 Monkey test 도구를 이용하여 패킷을 수집하였다.

5.2 실험 결과 및 분석

본 논문에서는 안드로이드 악성코드 test set과 정상 어플리케이션 test set으로 구분한 실험을 수행하였으며 test set 마다 악성코드 샘플이 일정한 개수로 존재 하지 않는 관계로 test set 마다 다르게 악성코드 그룹을 나누어 선정하였다.

탐지실험 전 각 그룹별 샘플을 training 하여 특징을 저장하였다. 표 2.는 실험 1차부터 3차까지 수행되면서 그룹별로 training된 샘플수를 나타낸 것으로 실험 1차부터 3차까지 샘플 수를 증가시키면서 training 수당 test 샘플 수의 탐지확률이 증가 되는 것을 표 3.과 연계하여 확인 할 수 있다.

표 3.과 같이 1차 실험에서는 test set 수는 12개이고 샘플 수는 36개로 실험을 수행하였다. 1차 실험결과 test set의 샘플보다 학습된 샘플의 수가 많았지만 전체 탐지확률은 52%를 보이며 낮은 성능을 보였다. 1차 실험 후 악성코드의 샘플 중 탐지확률이 가장 낮은 Anserver의 특징을 재생성하였다.

[표 2] 실험을 위한 학습된 샘플 수

[Table. 2] Number of trained sample for experiment

Group Name	Test1	Test2	Test3
	Number of sample	Number of sample	Number of sample
Group1	-	11	11
Group2	22	22	32
Group3	6	8	8
Group4	11	11	11
Group5	-	5	10
Total	39	57	73

2차 실험에서는 test set 수는 30개, 샘플 수는 92개로 증가시켜 실험을 수행하였다. 2차 실험 결과 정상 어플리케이션에서는 각 정상어플리케이션 마다 탐지를 하였으며, 1차 실험에서 Anserver 안드로이드 악성코드의 탐지성능이 낮았지만 2차 실험에서 가장 높은 탐지 성능을 보였다. 그리고 전체 탐지확률은 71%를 보이며 1차보다 높은 탐지확률을 보였다.

3차 실험에서는 총 test set수는 40개, 샘플 수는 122개로 증가시켰다. 3차 실험결과, 2차 실험에서 탐지확률이 낮았던 Droidkungfu와 GoldDream 악성코드 그룹의 특징을 추가하여 탐지성능을 개선하였다. 그 결과 3차 실험까지 총 122개 test 샘플 중 84%의 탐지확률 보이고 있다.

[표 3] 안드로이드 악성코드와 정상 안드로이드 프로그램의 탐지 실험 결과

[Table. 3] The result of experiment using Android malware and normal software

Group Name	Test 1		Test 2		Test 3	
	Number of sample	Detection rate	Number of sample	Detection rate	Number of sample	Detection rate
Group1	-	-	11	1.0	11	1.0
Group2	12	0.67	27	0.52	37	0.76
Group3	12	0.17	27	0.89	37	0.92
Group4	12	0.75	12	0.75	12	0.75
Group5	-	-	15	0.47	25	0.80
Total	36	0.52	92	0.71	122	0.84

본 논문에서 제안한 네트워크 기반 안드로이드 악성코드 탐지의 실험 결과, training 샘플 수를 증가 할수록 탐지확률이 증가하는 것을 확인할 수 있었으며 각 test set마다 악성코드가 설치된 그룹을 확인할 수 있어 악성코드가 활동 중인 안드로이드 장치에서 네트워크를 통해 악성코드가 탐지할 수 있다.

6. 결론

본 논문에서 제안한 네트워크 기반 안드로이드 악성코드 탐지 시스템은 네트워크를 이용하는 안드로이드 악성코드를 탐지하는 방법으로 사용될 수 있다. 또한 실험결과와 같이 네트워크 통신

을 하는 안드로이드 악성코드는 항상 패킷을 발생시키기 때문에 어떤 악성코드가 감염 되어있는지 간단히 확인할 수 있고, 네트워크상에서 패킷이 이동되는 경로 어디에서든 패킷을 수집할 수 있는 곳이라면 악성코드를 탐지 할 수 있다.

TCP-Flow 시퀀스(TCP-Flow sequence) 와 도메인 네임(Domain Name)의 사용으로 악성코드의 페이로드(payload)를 자세히 검사하지 않고 TCP-Flow의 형태만 검사하기 때문에 간단한 방법으로 악성코드를 검사할 수 있으며, 패킷의 흐름에 대한 감시를 수행하는 IDS와 IPS 시스템과 연계하여 장치에서 악성코드가 동작하는지 파악할 수 있을 것으로 예상된다. 그리고 이 제안된 악성코드 탐지 방법을 응용하여 정상어플리케이션을 구분할 수 있어 네트워크에서 사용 중인 모바일기기에서 어떤 어플리케이션을 사용 중 인지 확인할 수 있다.

본 논문의 연구에서 제한사항으로는 안드로이드에서 패킷을 실시간으로 수집하고 전달하는 시스템으로 구현되지 않았다. 하지만 실시간 패킷이 수집되고 수집된 패킷의 정보를 .pcap포맷 형식의 파일로 전달한다면 본 연구에서 제안된 방법을 이용해 실시간 탐지 시스템으로 구성할 수 있을 것이다. 또한 악성코드의 TCP-Flow 시퀀스(TCP-Flow sequence) 특징을 생성하기 위해 반복적으로 악성코드를 실행해 특징을 생성할 때 장시간 소요되는 것이 문제가 되었다.

따라서 향후 연구에서는 본 시스템을 실시간으로 패킷 수집기능을 추가되면 자동으로 수집된 패킷의 정보를 이용하여 특징을 생성할 수 있을 것으로 예상된다.

ISSN: 1738-7531 JSE

References

[1] Dong_Jie Wu and Ching-Hao Mao and Te-En Wei and Hahn-Ming Lee and Kuo-Ping Wu, DroidMat: Android Malware Detection through Manifest and API Calls Tracing. In proceedings of the 2012 Seventh Asia Joint Conference in Information Security(AsiaJCIS), **(2012)** Aug, pp. 62-69.

[2] Jungtae Kim and Eul Gyu Im, Malicious Family Detection Based on Android Using Similar Class Information, Journal of Security Engineering, **(2013)** August, Vol.10, No.4, pp. 441-453.

[3] Zarni Aung and Win Zaw, Permission-Based Android Malware Detection, International Journal of Scientific & Technology Research. **(2013)** March, Vol. 2, Issue 3.

[4] Hyunsoo Lee and Eul Gyu Im, The Analysis Possibility Of Android Application Malignant BehaviorThrough The FTRACE, In proceedings of the KIISE Korea Computer Congress, **(2014)** June, pp. 990-991

[5] R Perdisci and W Lee and N Feamster, Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. In proceedings of the 2010 USENIX Symposium on Networked Systems Design and Implementation **(2010)** April.

[6] R Perdisci and D Ariu and G Giacinto, Scalable fine-grained behavioral clustering of HTTP-based malware, International Journal of Computer and Telecommunications Networkin. **(2013)** February, pp. 487-500.

[7] Florian Tegeler and Xiaoming Fu and Giovanni Vigna and Christopher Kruegel, BotFinder: finding bots in network traffic without deep packet inspection. In proceedings of the 2012 international conference on Emerging networking experiments and technologies, **(2012)** December, pp. 349-360

[8] Konrad Rieck and Guido Schwenk and Tobias Limmer and Thorsten Holz and Pavel Laskov, Botzilla: detecting the "phoning home" of malicious software. In proceedings of the 2010 ACM Symposium on Applied Computing, **(2010)** March, pp 1978-1984.

[9] http://cs.ucsb.edu/~iland/AndroidMalwareDetection.pdf, December **(2011)**.

[10] Te-En Wei and Ching-Hao Mao and Albert B. Jeng and Hahn-Ming Lee and Horng-Tzer Wang and Dong-Jie Wu, Android Malware Detection via a Latent Network Behavior Analysis. In proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications(TrustCom), **(2012)** June, pp 1251-1258.

[11] Yincheng Qi and Mingjing Cao and Can Zhang and Ruping Wu, A Design of Network Behavior-Based Malware Detection System for Android. In proceedings of the 2014 14th International Conference on Algorithms and Architectures for Parallel Processing(ICA3PP), **(2014)** August, pp 590-600.

[12] http://xlinux.nist.gov/dads/HTML/Levenshtein.html, August **(2013)**.

[13] http://en.wikipedia.org/wiki/Sequence_alignment, March **(2009)**.

Authors

성명재 (Myeong Jae Seong)

2011년 2월 : 공주대학교 정보통신학과 학사
2012년 9월 : STS반도체통신 근무
2013년 3월~현재 : 한양대학교 컴퓨터·소프트웨어학과 석사과정
관심분야 : 정보보호, 네트워크 보안, 모바일 보안, 데이터베이스

박해룡 (Haeryong Park)

1999년 2월 : 전남대학교 이학사
2001년 2월 : 서울대학교 이학석사
2006년 2월 : 전남대학교 이학박사
2000년~현재 : 한국인터넷진흥원 정보보호기술개발팀장
관심분야 : 암호알고리즘 설계 및 분석, 사이버 블랙박스 기술 개발, 클라우드 서비스 보안 등

최보민 (Bomin Choi)

2012년 2월 : 가천대학교 학사
2014년 2월 : 가천대학교 공학석사
2014년~현재 : 한국인터넷진흥원 정보보호기술개발팀
관심분야 : 악성코드 프로파일링, 빅데이터. 지능형 알고리즘

임을규 (Eul Gyu Im)

1992년 : 서울대학교 컴퓨터공학과 학사
1994년 : 서울대학교 컴퓨터공학과 석사
2002년 : University of Southern California, Computer Science Ph.D.
2005년~현재 : 한양대학교 컴퓨터공학부 부교수
관심분야 : 제어시스템 보안, 악성코드, 정보 보호, 소프트웨어 취약 점검

ISSN: 1738-7531 JSE