논문 15-40-08-12 The Journal of Korean Institute of Communications and Information Sciences '15-08 Vol.40 No.08
http://dx.doi.org/10.7840/kics.2015.40.8.1551

# 자기회귀 이동평균 모델을 이용한 안드로이드 악성코드 탐지 기법

김 환 희◆, 최 미 정°

# Android Malware Detection Using Auto-Regressive Moving-Average Model

Hwan-Hee Kim◆, Mi-Jung Choi°

요 약

최근 스마트 기기가 PC와 유사한 성능을 보이면서, 사용자들은 메신저, SNS(Social Network Service), 은행 업무 등 PC에서 수행했던 업무들을 모바일 기기에서도 수행할 수 있게 되었다. 이 같은 긍정적인 변화와 함께 스마트 기기를 대상으로 하는 공격으로, 보안 위협이 증가하는 부정적인 변화도 나타났다. 대표적으로 사용자의 개인정보 유출, 부당한 과금을 비롯하여 최근에는 DDoS(Distributed Denial of Service) 공격을 발생시키는 봇(Bot)으로 스마트 기기가 활용되면서 모바일 보안에 대한 위협이 증가하는 실정이다. 특히, 스마트 기기의 80% 이상을 차지하는 안드로이드 플랫폼에서의 악성코드를 통한 피해건수가 증가하고 있다. 본 논문에서는 안드로이드의 악성코드를 탐지하기 위해 통계 기반 분석법 중 하나인 시계열 분석법을 제안한다. 시계열 모델 중 기존의 데이터를 기반으로 정확한 예측값을 도출할 수 있는 자기회귀 이동평균 모델을 이용하였으며, Z-Score를 이용한 비정상 데이터 후보군 추출을 통해서 전체 데이터와의 비교 없이 추출된 후보군과의 데이터 비교를 통해서 빠르게 악성코드를 탐지하는 방법을 이용한다. 악성코드 탐지 실험 결과를 통해 제안하는 방법의 타당성을 검증하고자 한다.

**Key Words :** Malware-detection, Android, Auto-regressive, Moving-average, Time-series

ABSTRACT

Recently, the performance of smart devices is almost similar to that of the existing PCs, thus the users of smart devices can perform similar works such as messengers, SNSs(Social Network Services), smart banking, etc. originally performed in PC environment using smart devices. Although the development of smart devices has led to positive impacts, it has caused negative changes such as an increase in security threat aimed at mobile environment. Specifically, the threats of mobile devices, such as leaking private information, generating unfair billing and performing DDoS(Distributed Denial of Service) attacks has continuously increased. Over 80% of the mobile devices use android platform, thus, the number of damage caused by mobile malware in android platform is also increasing. In this paper, we propose android based malware detection mechanism using time-series analysis, which is one of statistical-based detection methods.We use auto-regressive moving-average model which is extracting accurate predictive values based on existing data among time-series model. We also use fast and exact malware detection method by extracting possible malware data through Z-Score. We validate the proposed methods through the experiment results.

※ 이 논문은 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2013R1A1A3011698)
※ 2014년도 강원대학교 학술연구조성비로 연구하였음(과제번호-120140395)
◆ First Author : Dept. of Computer Science, Kangwon National University, hwanhee0920@kangwon.ac.kr, 학생회원
° Corresponding Author : Dept. of Computer Science, Kangwon National University, mjchoi@kangwon.ac.kr, 종신회원
논문번호 : KICS2015-03-058, Received March 22, 2015; Revised June 21, 2015; Accepted August 10, 2015

## Ⅰ. 서 론

스마트폰, 태블릿과 같은 스마트 기기의 발전은 언제 어디서나 인터넷에 접속하여 원하는 서비스를 받을 수 있게 됨으로 우리의 삶의 패턴을 변화시켰다. 특히 최근에 출시된 스마트 기기의 대부분은 쿼드 코어 CPU, 3GB 이상의 메모리를 지원하면서 PC 성능과 거의 흡사한 성능을 보이고 있으며, 이와 동시에 LTE, LTE-A의 보급은 스마트 기기 사용자 수를 증가시키는 큰 요인으로 자리 잡고 있다.

스마트 기기 사용자들은 스마트 기기를 이용하여 기존의 SMS, 전화 서비스 이외에 메신저(Messenger), SNS(Social Network Service), 스마트 뱅킹과 같은 다양한 서비스를 언제 어디서나 활용할 수 있다는 측면에서 긍정적인 효과를 얻었다. 하지만, 동시에 스마트 기기를 대상으로 하는 악성코드 공격이라는 새로운 위협에 직면하게 되었다. 스마트 기기를 대상으로 하는 악성코드 공격은 크게 사용자에게 부당한 과금을 유발하는 유형, 사용자의 개인 정보를 획득하는 유형, 모바일 봇넷(Bot-net)으로 활용하는 유형, 루트 권한을 획득하는 유형으로 구분될 수 있다[1]. 특히 스마트폰이 공격자의 봇(Bot)으로 활용되어 DDoS(Distibuted Denial of Service) 공격을 일으키는 요소 중 하나로 손꼽히며, 이로 인한 피해가 국가적 수준으로 커지고 있어 대비하기 위한 노력이 필요한 상황이다.

그림 1은 핀란드 보안업체 F-secure가 발표한 자료로 신종 악성코드가 2013년과 2014년 사이에 발생 건수를 나타내고 있다[2,3]. 각 분기별 수치를 비교했을 때 안드로이드 이외의 타 플랫폼의 악성코드 발생 건수는 대체적으로 감소하는 추세를 보이나, 안드로이드 플랫폼에서의 신종 악성코드 건수는 계속해서 증가하는 것을 확인할 수 있다.

안드로이드 플랫폼에서의 악성코드 증가 원인[4]은 다음과 같다. 첫 번째는 안드로이드 플랫폼의 개방적인 특징 때문이다. 안드로이드는 오픈 플랫폼 정책으로 사용자가 개발한 써드 파티 어플리케이션의 제작 및 배포가 용이하다. 즉, 공격자들은 오픈 플랫폼 특징을 이용하여 정상적인 어플리케이션에 악의적인 코드를 일부 추가하여 다시 재배포하는 등의 리패키징(Repackaging) 공격을 수행할 수 있다. 또한 Google Play Store에 어플리케이션을 등록하는 과정이 까다롭지 않으며, Play Store 이외에도 특정 블로그, 블랙마켓(Black-market)으로부터 어플리케이션의 다운로드 및 설치가 가능하기 때문에 악성코드 유포에 유리하다고 할 수 있다.

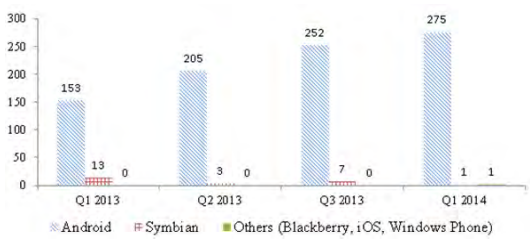


Fig. 1. Quarterly new malware families according to mobile platform

두 번째 원인은 세계 스마트폰 시장에서 안드로이드 플랫폼의 점유율이 높기 때문이다. IDC에서 예측한 전 세계 스마트폰 플랫폼 점유율[5]에 따르면 2014년 안드로이드 플랫폼 점유율이 80%에 육박하며, iOS는 14.9%, 그 외 운영체제는 약 6% 정도를 차지하는 것을 알 수 있다. 안드로이드 플랫폼의 점유율이 가장 높기 때문에 공격자들은 다수의 공격을 광범위하게 수행하고 공격의 효과를 극대화하기 위해서 안드로이드 플랫폼을 대상으로 많은 공격이 수행되는 것으로 해석될 수 있다. 따라서 악성코드 탐지 실험환경으로 안드로이드 플랫폼을 선정하였다.

이렇게 모바일 플랫폼에서의 악성코드 발생건수가 증가하자, 모바일 위협에 대응하기 위해 다양한 연구들이 최근에 활발히 진행되고 있다. 기존 연구는 크게 시그니처 기반 분석[6], 행동 기반 분석[7], 동적 기반 분석[8], 통계 기반 분석[9]으로 나눌 수 있다. 시그니처 기반 분석의 경우 각 악성코드에 대한 정보를 미리 파악하여 악성코드를 탐지할 수 있는 시그니처를 생성해야 하며 이 모든 과정을 관리자가 직접 사전에 수행해야 하는 오버헤드가 있다. 또한 시그니처 기반 분석법은 잘 알려진 악성코드에 대해서는 높은 탐지율을 보이지만, 새로운 악성코드에 대한 탐지가 어렵다는 한계점이 있다. 행동 기반 분석의 경우 적용하는 기계학습 분류기의 종류와 수집된 정보의 종류 및 양에 따라 탐지 성능에 큰 차이를 보인다. 따라서 악성코드 탐지에 필요한 정확한 정보를 수집하고, 상황에 따른 적절한 기계학습 분류기를 선정해야 한다는 한계점이 있다. 안드로이드 버전이 4.0이상으로 업그레이드 되면서 보안을 이유로 추출할 수 있는 자원이 제한되면서 특히 기계학습 분류기를 통한 악성코드 탐지가 어려워졌다. 동적 기반 분석의 경우 각 경우별로 데이터의 흐름을 확인하고 분석하는데 소요되는 시간 및 자원 소모에 대한 오버헤드가 존재한다는 한계점이 있다.

본 논문은 작은 수의 자원 정보만 가지고도 가능한 한 빠르고 정확하게 악성코드를 탐지하기 위한 방안

을 마련하고자 한다. 안드로이드 스마트 기기의 자원 소요 패턴 및 특징을 정확히 판단하여 악성코드를 탐지하기 위해 통계 기반 분석 중 하나인 시계열 분석을 이용한다. 시계열 분석으로 사용되는 모델은 자기회귀 이동평균(Auto-regressive moving-average) 모델[10]이다. 자기회귀 이동평균 모델에서 파라미터 값을 변경시켜 가면서 정상 코드에 대한 최적의 모델을 생성하고, 생성된 모델과 현재 데이터를 비교하여 안드로이드 악성코드를 탐지함으로써 정확도와 탐지율을 높이고자 한다. Z-Score를 이용한 악성코드 후보군 추출을 통해 비교 데이터의 수를 줄이고, 비교 수행을 빠르게 하여 악성코드 탐지 속도를 향상시킨다.

## Ⅱ. 관련 연구

시계열 분석 기법은 시간 흐름에 따라 일정한 간격으로 데이터를 관찰하여 데이터의 패턴 및 특징을 분석하는 방법이다. 본 장에서는 시계열 분석을 기반으로 한 유/무선 환경의 비정상 탐지 논문을 소개한다.

Bahaa Eldin[11]의 논문은 네트워크에서 발생하는 비정상을 탐지하기 위하여 비안정적인 자기회귀(Non-stationary AR, NSAR) 모델을 활용하였다. 먼저 비안정적인 자기회귀 모델의 파라미터를 구하기 위해 RMSE(Root Mean Square Error) 값을 활용하였다. 이후 네트워크 트래픽을 NSAR(p) 모델을 통해 예측하고, 예측값과 실제값의 차이가 특정 임계값을 넘어섰을 때 비정상으로 판단하였다. 실험 데이터 셋은 DARPA 데이터 셋을 이용하였다. 상기 논문은 유선 네트워크에서 발생하는 비정상 트래픽만을 탐지하는 논문이다. 무선에서 발생하는 비정상 트래픽의 특성을 고려하지 않았다. 또한, 실험에 사용된 데이터 셋이 1999년에 수집된 DARPA데이터로 현 유무선 네트워크 트래픽의 특징을 반영하지 못하는 한계점이 존재한다.

Yingxu Lai[12]의 논문은 모바일 네트워크 트래픽의 예측을 통하여 비정상을 탐지하는 논문이다. 무선 트래픽의 비정상을 탐지하기 위해 제시한 정보는 총 110개이다. 110개의 모든 정보를 많은 수의 정보들을 비정상을 탐지하는데 모두 활용하지 않고 상대 편차 거리를 이용하여 정보 선택(Feature selection) 과정을 거치며 최적의 정보들을 먼저 선정한다. 이후 시계열 모델 중 하나인 자기회귀 이동평균 모델을 이용하여 트래픽 데이터를 예측한다. 결과적으로 생성된 모델과 현제 네트워크 데이터를 비교함으로써 트래픽의 비정상을 탐지하였다. 상기 논문에서 활용된 데이터 셋은 정상 네트워크 데이터와 비정상 네트워크 데이터로 나뉘며, 다양한 플로우로 구성되어 있다. 비정상 유형이 웜 공격, DDoS 공격과 같은 네트워크 공격만으로 제한되어 있으며, 자기회귀 이동평균 모형으로 모델링하는 과정에서 ARMA(p, p-1) 만을 고려하여 다양한 모델을 생성하지 않았다는 한계점이 있다.

따라서 본 논문에서는 이와 같은 기존 연구들의 한계점을 해결하는 방안을 제시한다. 첫 번째, Ahnlab ASEC 2013 리포트[13]에서 제시한 악성코드를 활용하여 최근에 발생하는 모바일 악성코드를 대상으로 탐지를 수행하였다. 두 번째, 트로이목마(Trojan), 루트권한 획득을 목적으로 한 악성코드(Exploit)를 추가하여 네트워크 기반 공격이외의 최근에 발생하는 다른 형태의 공격들도 탐지하는 방안을 제시한다. 세 번째, 자기회귀 이동평균 모델의 파라미터를 다양하게 변화시키면서 최적의 모델을 선정하기 위한 방안을 제시하였다. 또한, 탐지를 수행하기 이전에 Z-Score값과 수집된 데이터를 비교하는 선행 작업을 통해 비정상이 예측되는 비교 데이터를 추출함으로써 비교 시간과 탐지 결과의 정확도를 높였다.

## Ⅲ. 자기회귀 이동평균 모델 기반의 악성코드 탐지 기법

본 장에서는 안드로이드 악성코드 탐지를 위해 필요한 자원 정보를 수집하는 모니터링 시스템과 자기회귀 이동평균 모델 기반의 악성코드 탐지 방법에 대해 서술한다.

### 3.1 악성코드 탐지를 위한 자원 정보 모니터링 시스템

자원 정보 모니터링 시스템에서 안드로이드 기기의 악성코드 기기의 악성코드 탐지를 위한 자원 정보는 크게 네트워크, 메모리, CPU에 관한 정보로 분류되며, 총 5개로 구성된다. 네트워크 관련 정보는 다운로드 받은 바이트 수(Rxbytes), 업로드 한 바이트 수(Txbytes)를 추출하며, 메모리 관련 정보는 스마트 기기가 현재 사용 중인 메모리 정보(Usage_memory)를 추출한다. CPU 관련 정보는 안드로이드 기기의 전체 CPU 사용률(Total_CPU)과 사용자가 수행한 어플리케이션이 사용하는 CPU 사용률(User_CPU)을 추출한다. 표 1은 모니터링 시스템에서 추출한 자원 정보를 한 눈에 볼 수 있도록 정리한 것이다. 기존의 시계열 기반의 악성코드 탐지 기법에서는 네트워크 자원을 많이 사용하였지만, 본 논문에서는 다양한 자원 정

Table 1. Resource information for android malware detection

| Category | Resource information |
|---|---|
| Network | Rxbytes, Txbytes |
| Memory | Usage_memory |
| CPU | Total_CPU, User_CPU |

보를 추출하여 실험에 사용하였다.

모니터링 시스템에서 추출한 자원 정보는 리눅스 커널과 안드로이드에서 제공하는 API를 통해 추출한다. 특히 리눅스 커널은 안드로이드 기기의 하드웨어 및 운영체제에 대한 정보를 내포하고 있어 다양한 정보 추출이 가능하다. 네트워크 정보의 경우 '/proc/uid_stat/uid' 폴더에 있는 'tcp_rcv' 파일과 'tcp_snd' 파일을 사용하며, 이는 각각 다운로드 받은 바이트 수와 업로드 한 바이트 수를 의미한다. 메모리 정보는 안드로이드 내부에서 제공하는 ActivityManager.MemoryInfo 클래스를 이용하여 사용 중인 메모리 정보를 추출한다. CPU 정보는 리눅스의 'top' 명령어를 활용하여 정보를 추출한다. 명령어 실행 결과를 파싱하여 스마트 기기 총 CPU 사용률과 사용자가 수행중인 어플리케이션이 사용하는 CPU 사용률을 추출한다.

안드로이드 기기로부터 자원 정보를 수집하기 위해서는 별도의 자원 정보 모니터링 시스템이 요구된다. 그림 2는 안드로이드 기기에서 자원 정보를 모니터링하기 위한 자원 정보 모니터링 시스템의 구조를 나타낸 것이다.

안드로이드 자원 정보 모니터링 시스템의 동작 순서는 다음과 같다. 먼저 앞서 제시한 방법을 통해 안드로이드 자원 정보를 추출한다. 이렇게 추출된 각각의 정보들은 데이터 통합 모듈을 통해 하나의 데이터로 통합된다. 이와 같은 과정은 10초 단위로 반복되어 수행되면서 안드로이드 자원 정보에 대한 시계열 데이터를 생성한다. 여기서 반복 주기를 10초로 설정한 것은 10초 미만으로 주기를 설정하였을 경우 자원 정보 수집 시 스마트 기기내의 모바일 에이전트가 모니터링을 위해 많은 자원을 소모하게 되어 자원 측면의 오버헤드가 크기 때문이다. 또한 10초를 넘어선 시간으로 주기를 설정할 경우 악성코드로 인한 자원의 변화 및 패턴을 파악하기 어렵기 때문이다.

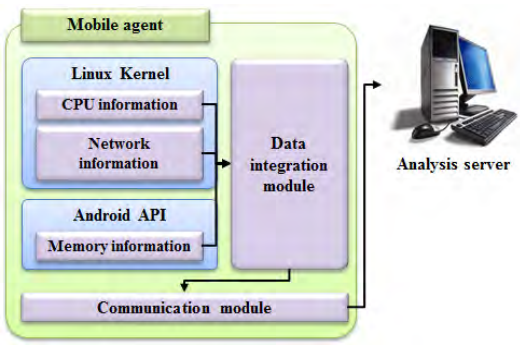


Fig. 2. Structure of android resource information monitoring system

데이터 모듈에 의해 통합된 데이터는 통신 모듈에 의해 분석 서버로 전송된다. 데이터를 분석 서버로 전송하는 것은 데이터 모델링 및 악성코드 분석으로 인한 스마트 기기의 자원 소요들의 오버헤드를 줄이기 위함이다. 분석 서버로 전송된 데이터는 자기회귀 이동평균 모델을 적용하여 악성코드 탐지를 수행한다. 데이터 모델링 및 악성코드 분석을 위해 통계 프로그램인 R[14]을 활용했다.

### 3.2 자기회귀 이동평균 모델을 적용한 악성코드 탐지 알고리즘

악성코드 탐지 프로세스는 크게 4가지로 구성되어 있다. 자원 정보 수집 단계, 데이터 모델링 단계, 후보 데이터 추출 단계, 악성코드 탐지 단계로 구성된다. 먼저 안드로이드 기기로부터 네트워크, 메모리, CPU에 대한 자원 정보를 추출한다. 이후 추출된 데이터는 통합 과정을 거쳐 분석 서버로 전송된다. 분석 서버에서는 정상적인 상황에서의 기기 정보를 바탕으로 각 자원 정보별 자기회귀 이동평균 모델링을 수행한다. 이 과정을 통해 정상 데이터의 각 자원 정보에 따른 자기회귀 이동평균 파라미터 p, d, q가 계산된다. 계산된 파라미터 p, d, q는 자기 회귀 이동평균 최적의 모델을 선정할 때 사용된다.

안드로이드 기기가 정상적인 상황일 때의 데이터와 비정상 어플리케이션이 실행된 상황일 때의 데이터를 바탕으로 악성코드 실행 가능성이 있는 데이터를 추출하기 위한 선행 과정을 수행한다. 이 과정은 Z-Score를 기반으로 수행되며, 악성코드가 실행되었을 가능성이 있는 시간 정보를 추출한다. 이후 자기회귀 이동평균 모형으로 모델링 된 결과와 Z-Score를 기반으로 추출된 시간 정보의 매칭을 통해 악성코드를 탐지한다. 탐지된 결과는 데이터베이스로 전송되며 실제 정답지(Ground truth)와 비교를 통해 악성코드 탐지 성능을 평가한다. 각 세부 과정은 아래 각 절에서 자세히 설명한다.

3.2.1 자기회귀 이동평균 모델을 이용한 데이터 모델링

본 절에서는 자기회귀 이동평균 모델을 이용한 데이터 모델링 수행과정을 자세히 설명한다. 자기회귀 이동평균 모델은 다른 설명 변수의 도입 없이 변수 과거의 값과 오차항만을 가지고 시계열에 적합한 모형을 설정할 수 있는 가장 간편한 방법이며, 점예측과 구간예측 등을 이용하여 정확한 예측값을 도출할 수 있다. 자기회귀 이동평균 모델링을 통해서 악성코드가 수행되지 않은 정상 상황에서의 스마트 기기의 자원 정보에 대해 모델링을 수행하고, 이를 수집한 자원 정보와 비교하여 악성코드가 수행된 비정상 상황을 탐지하게 된다. 정상적인 상황일 때의 기기의 자원 정보를 바탕으로 최적의 자기회귀 이동평균 모델의 파라미터 p, d, q의 값을 구한다. 최적의 파라미터 p, d, q 값을 구하기 위해서는 AIC 값을 이용한다. AIC 값은 주어진 데이터에 대해 시계열 모델이 얼마나 적절하게 피팅되었는지 판단하는 지표이다. 파라미터 추정 수행 결과 AIC 값이 최소일 때, 최적의 모델로 간주한다. 본 논문에서 사용한 안드로이드 기기 자원 정보 데이터의 경우 사용자 별로 사용 패턴이 다르므로, 주기를 가지지 않는 데이터로 간주하고 주기성을 띄는 파라미터인 P, D, Q는 계산하지 않는다. 데이터의 모델링은 통계 프로그램인 R을 이용한다. 그림 3의 알고리즘은 자기회귀 이동평균 모델의 최적의 파라미터를 구하는 과정을 나타낸다.

알고리즘의 입력은 안드로이드 기기가 악성코드에 감염되지 않고 정상적인으로 사용될 때의 시계열 데이터(Data)이다. 먼저 라인 (1)에서 최소 AIC 값을 100으로 설정한다. 이는 AIC의 값을 충분히 큰 수로 가정한 것으로 실험에서는 100 이상 나타나지 않는다는 가정을 기반으로 한다. 라인 (5)에서는 p, d, q 값에 따라 자기회귀 이동평균 모델링을 수행한다. 이 때 자기회귀 파라미터 값은 0에서 8로, 이동평균 파라미터와 적분 파라미터는 0에서 2로 변화시킨다. 이는 여러 번의 실험을 통해서 자기회귀 파라미터가 8을 넘어설 경우에 오히려 낮은 성능을 보였기 때문이다. 또한 이동평균 파라미터 값이 2를 넘어설 경우 일부 결과에 수학적 오류를 내포하고 있어 그 이상의 수치는 대상에서 제외시켰다. 적분 파라미터의 경우 3이상을 고려하지 않았는데, 값이 3이상일 경우 분석 결과 내에 난수가 발생하기 때문이다.

```
Procedure Parameter_estimate()
Input Data : The time-series data of normal status
Output R : The set of estimated parameter
           (AR, MA, Integrated)
(1)  min_AIC=100;
(2)  for p:=0 to 8 do
(3)    for q:=0 to 2 do
(4)      for d:=0 to 2 do
(5)        TEMP=sarima(Data, p, d, q);
(6)        if(TEMP$AIC < min_AIC) do
(7)          min_AIC=TEMP$AIC;
(8)          AR=p; MA=q; Integrated=d;
(9)        end-if
(10)     end-for
(11)   end-for
(12) end-for
(13) R=(AR, Integrated, MA)
```

Fig. 3. Algorithm of calculating optimal para meters of auto-regressive moving average model

3.2.2 Z-Score를 이용한 후보 데이터 추출

본 절에서는 Z-Score를 이용한 악성코드 데이터 후보 추출 과정에 대해 서술한다. 악성코드 데이터를 탐지하기에 앞서, 이 과정을 수행하는 이유는 데이터 분석 시 시간에 따른 오버헤드를 최소화하기 위함이다. 즉, 악성코드가 실행되었을 가능성이 있는 후보 데이터를 미리 추출함으로써, 전체 데이터를 일일이 확인할 필요 없이 악성코드를 탐지할 수 있는 장점이 있다.

본 논문에서는 데이터가 평균과 얼마만큼 일탈하였는가를 알아보는 척도로 Z-Score를 사용하였다. 그림 4의 알고리즘은 Z-Score를 이용하여 후보 데이터를 추출한 과정을 소개한다.

알고리즘의 입력은 안드로이드 기기가 정상적인 상황일 때의 데이터와 악성코드를 내포한 어플리케이션이 실행된 데이터(Data)로 구성된다. 라인 (4)~(7)은 시계열 데이터의 Z-Score가 2보다 큰 시간을 추출하는 부분이다. 여기서 Z-Score가 2이상이라는 것은 전

```
Procedure ZScore_extracting()
Input Data : The time-series data of normal status
             and performed malware
Output R : The set of judged by malware through
           Z-Score
(1)  SC=scores(Data, type='z');
(2)  count=0;
(3)  for i:=1 to length(SC) do
(4)    if(SC[i] > 2) do
(5)      R[i]=i;
(6)      count=count+1;
(7)    end-if
(8)  end-for
(9)  R=(The vector of Z-Score bigger than 2)
```

Fig. 4. Data extraction algorithm using Z-Score

체의 2.275%를 의미하며, 반복적인 실험을 통해 2라는 수치가 모든 악성코드를 후보군으로 추출하면서 적은 양의 후보군 데이터를 추출하는 최적의 값임을 확인하였다.

#### 3.2.3 데이터 매칭을 통한 악성코드 탐지

본 절에서는 앞서 자기회귀 이동평균 모형으로 모델링 된 데이터와 Z-Score를 통해 추출 된 데이터의 비교를 통해 악성코드를 탐지하는 과정을 설명한다. 그림 5의 알고리즘은 데이터의 매칭을 통한 악성코드 탐지 과정을 나타낸다.

알고리즘의 입력은 앞선 과정을 통해 얻어진 데이터(Normal_Data, Zscore_Data, p, d, q)와 임계값을 정의할 때 사용되는 변수(Lamda)이다. 먼저 라인 (1)에서 예측값의 개수를 5로 정의한다. 예측값의 개수를 5개로 정의한 것은 자기회귀 이동평균 모델을 이용한 예측 시 예측값의 개수를 크게 설정하였을 경우 특정 값으로 수렴하는 문제가 발생하기 때문이다. 라인 (4)에서는 예측에 활용되는 정상 데이터의 범위를 정의한다. 즉, Z-Score를 통해 추출된 시간의 100번째 전으로부터 Z-Score로 추출된 시간 전까지 데이터를 활용한다. 라인 (5)~(6)은 라인 (4)에서 정한 범위의 데이터를 바탕으로 향후 6개의 값을 예측한다. 예측값의 수가 5개가 아닌 6개인 것은 임계값을 구할 때 이전의 예측값이 활용되기 때문이다. 라인 (8)~(9)는 악성코드 탐지를 위한 임계값을 계산한다. 라인 (10)에서는 임계값과 실제값의 비교를 통해 악성코드 유무를 판단하는 부분이며, 악성코드로 판단 시 결과 벡터 R에 해당 시간을 저장한다.

```
Procedure Malware_detection()
Input ND : The time-series data of normal status
       ZD : The extracted data using Z-Score from
            all time-series data
       P : Auto-regressive parameter
       D : Integrated parameter
       Q : Moving-average parameter
       Lamda : The constant is used by threshold.
Output R : The set of judged by malware
(1)   forecast_value=5;
(2)   for i:=1 to length(ZD) do
(3)      if(ZD[i]+forecast_value > length(ND)) finish;
(4)      start=ZD[i]-100; end=ZD[i]-1;
(5)      fit=arima(ND[start:end-1], P, D, Q, method="ML");
(6)      prediction=predict(fit, n.ahead=forecast_value+1);
(7)      for j:=1 to forecast_value do
(8)         difference[j]=abs(prediction$pred[j]-ND[end+j=1];
(7)         threshold[j]=(1-Lamda)*difference[j] +
                        Lamda*prediction$pred[j+1];
(8)         if(ND[end+j] > threshold[j]) do
(9)            R[count++]=end+j;
(10)        end-if
(11)     end-for
(12)  end-for
(13)  R = (The set of judged by malware)
```

Fig. 5. Malware detection algorithm

악성코드를 판단하는 기준인 임계값은 수식 (2)와 같다. 임계값은 이전 시간의 예측값과 실제값의 차이에 측정 상수를 곱한 값에 현재 시간의 예측값에 특정 상수를 곱한 값을 더하는 방법으로 계산한다. 임계값은 악성코드가 수행될 경우 현재 시점의 관측값(실제로 측정한 값)과 예측값이 큰 차이를 보인다는 가정하에 세워진 수식이다. 본 논문에서는 실험을 통해 $\lambda$값을 0부터 1까지 0.1씩 변화시킴으로써 최적의 $\lambda$값을 계산하였으며, 그 값은 0.5이다.

$$Zt = Yt\ (\text{관측치}) - Yt^*\ (\text{예측치}) \quad (1)$$

$$\text{Threshold} = (1 - \lambda) \times Zt\text{-}1 + \lambda \times Yt^* \quad (2)$$

## Ⅳ. 성능 평가

본 장에서는 제안한 자기회귀 이동평균 모델링의 결과와 Z-Score를 이용한 악성코드 데이터 추출의 결과, 자기 회귀 이동평균 모델을 이용한 악성코드 탐지 성능을 나타낸다. 실험을 통해 자기회귀 이동평균 모델링 최적의 파리미터를 추출하고, 정상 데이터와 비정상 데이터들 중에서 악성코드를 포함한 데이터를 얼마나 정확하게 탐지하는지 평가하여 제안한 방법의 타당성을 검증한다.

### 4.1 실험 데이터 셋

본 절에서는 실험에 활용된 데이터 셋에 대해 설명한다. 데이터 셋은 크게 정상적인 상황의 자원 정보와 악성코드를 내포한 어플리케이션이 실행된 기기의 자원 정보로 구성된다. 정상적인 상황의 자원 정보는 3일간, 10초 단위로 수집되었으며, 총 25920개의 시간 정보로 구성된다. 비정상 어플리케이션이 실행된 기기의 자원 정보는 5일간에 데이터 수집이 이루어졌으며, 각 일별로 수행된 악성코드가 상이하다. 본 논문에서 활용된 악성코드의 경우 Anserver, Angry Birds Rio Unlocker, DroidKungFu, GoldDream, DroidDeluxe, 총 5개로 구성되어 있다. 이는 Ahnlab에서 발표한 리포트[13]에 소개되었으며, 국내 및 국외에 많은 피해를 입힌 악성코드들이다.

### 4.2 자기회귀 이동평균 모델링 및 데이터 추출 결과

본 절에서는 3장에서 제시한 자원 정보를 바탕으로 자기회귀 이동평균 모델링을 수행한 결과를 나타내고, Z-Score를 이용하여 악성코드가 실행되었을 가능성이 있는 시간 정보를 추출한 결과에 대해 서술한다. 자기회귀 이동평균 모델링을 수행한 데이터는 정상적인 상황에서의 안드로이드 기기 자원 정보를 활용한다. 표 2는 각 자원별 자기회귀 이동평균 모델링을 수행한 결과, 모델의 파라미터 값을 나타낸다.

Z-Score를 이용하여 후보군 데이터를 추출하는 것은 비정상 어플리케이션이 실행되었을 경우 각 자원 정보가 평균값과 상이하게 나타낼 것이라는 가정 하에 이루어진다. 그림 6은 본 논문에서 활용한 악성코드 중 하나인 Anserver 악성코드가 실행된 시계열 데이터 중 Total_CPU 측면의 데이터를 활용하여 시간 정보를 추출한 결과이다. 이와 동일한 방법으로 다른 자원 에서도 시간 정보를 추출한다.

Table 2. Modeling results using auto-regressive moving-average

| Feature information | The result of modeling |
|---|---|
| Total_CPU | ARIMA(7, 1, 2) |
| User_CPU | ARIMA(5, 1, 2) |
| Rxbytes | ARIMA(8, 1, 2) |
| Txbytes | ARIMA(8, 1, 1) |
| Usage_Memory | ARIMA(1, 1, 0) |

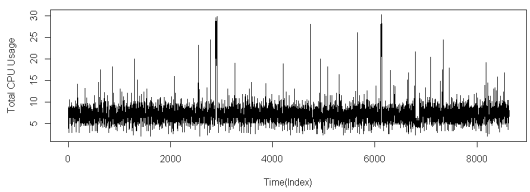


Fig. 6. Total CPU use rate of Anserver's implemented data

### 4.3 악성코드 탐지 결과

본 절에서는 자기회귀 이동평균 모형으로 모델링된 데이터와 Z-Score를 바탕으로 얻어진 시계열 데이터의 비교를 통한 악성코드 탐지 결과를 서술한다. 탐지 성능은 TPR(True Positive Rate), FPR(False Positive Rate), Precision, Accuracy를 활용한다. 표 3은 각 악성코드별 탐지 결과를 나타낸다.

Anserver 악성코드의 경우 사용 중인 메모리 정보를 바탕으로 탐지하였을 때 TPR이 97.8%, FPR이 0.0%로 가장 좋은 성능을 나타냈으며, CPU와 관련된 정보를 바탕으로 탐지하였을 때도 비교적 좋은 탐지 성능을 나타냈다. 하지만 네트워크와 관련된 정보에서는 FPR이 100.0%로 탐지가 불가능했다. Angry Birds Rio Unlocker 악성코드의 경우 네트워크와 관련된 정보를 바탕으로 탐지하였을 때 TPR이 99.8%, FPR이 3.3%로 가장 좋은 성능을 나타냈지만, 사용 중인 메모리 정보 및 CPU와 관련된 정보로는 악성코드 탐지가 불가능했다. DroidKungFu 악성코드의 경우 사용 중인 메모리 정보를 바탕으로 탐지하였을 때 TPR이 97.2%, FPR이 0.0%으로 가장 좋은 성능을 나타냈으며, CPU와 관련된 정보로 탐지하였을 때도 비교적 좋은 탐지 성능을 나타냈다. 하지만 네트워크와 관련된 정보에서는 FPR이 100.0%로 탐지가 불가능했다. GoldDream 악성코드의 경우도 Anserver, DroidKungFu와 마찬가지로 사용 중인 메모리 정보를

Table 3. Malware detection result

| Malware | Resource information | TPR | FPR | Pre. | Acc. |
|---|---|---|---|---|---|
| Anserver | Total_CPU | 88.4% | 0.0% | 100.0% | 88.5% |
| | User_CPU | 90.3% | 0.0% | 100.0% | 90.3% |
| | Rxbytes | 99.2% | 100.0% | 99.3% | 98.5% |
| | Txbytes | 99.2% | 100.0% | 99.3% | 98.5% |
| | Usage_Memory | 97.8% | 0.0% | 100.0% | 97.8% |
| Angry Birds Rio Unlocker | Total_CPU | 72.9% | 33.3% | 99.6% | 72.9% |
| | User_CPU | 77.4% | 30.0% | 99.7% | 77.4% |
| | Rxbytes | 99.8% | 3.3% | 99.9% | 99.8% |
| | Txbytes | 99.8% | 3.3% | 99.9% | 99.8% |
| | Usage_Memory | 99.6% | 0.0% | 100.0% | 99.6% |
| Droid KungFu | Total_CPU | 87.1% | 0.0% | 100.0% | 87.2% |
| | User_CPU | 89.3% | 0.0% | 100.0% | 89.4% |
| | Rxbytes | 99.2% | 100.0% | 99.3% | 98.5% |
| | Txbytes | 99.2% | 100.0% | 99.3% | 98.5% |
| | Usage_Memory | 97.2% | 0.0% | 100.0% | 97.2% |
| Gold Dream | Total_CPU | 86.1% | 0.0% | 100.0% | 86.2% |
| | User_CPU | 89.0% | 0.0% | 100.0% | 89.1% |
| | Rxbytes | 99.3% | 100.0% | 99.3% | 98.6% |
| | Txbytes | 99.3% | 100.0% | 99.3% | 98.6% |
| | Usage_Memory | 98.8% | 0.0% | 100.0% | 98.8% |
| Droid Deluxe | Total_CPU | 96.8% | 0.0% | 100.0% | 96.9% |
| | User_CPU | 96.8% | 0.0% | 100.0% | 96.9% |
| | Rxbytes | 100.0% | 0.0% | 100.0% | 100.0% |
| | Txbytes | 100.0% | 0.0% | 100.0% | 100.0% |
| | Usage_Memory | 99.7% | 100.0% | 99.2% | 99.0% |

바탕으로 탐지 시에 TPR이 99.9%, FPR이 0.0%로 가장 좋은 성능을 나타냈으며, 네트워크 관련 정보로는 탐지가 불가능했다. 마지막으로 DroidDeluxe 악성코드의 경우 네트워크 관련 정보로 탐지 시 TPR이 100.0%, FPR이 0.0%로 정확한 탐지율을 보였으며, CPU 관련 정보 또한 좋은 성능을 나타냈다. 하지만 사용 중인 메모리 측면에서의 탐지는 불가능했다.

이 실험결과 악성코드가 메모리나 CPU 자원을 많이 소비하는 것과 네트워크 자원을 많이 소비하는 특징으로 나뉨을 볼 수 있다. 대부분의 악성코드들이 메모리, CPU, 주고받는 데이터양을 통해 탐지됨을 실험결과를 통해 알 수 있다. 실험을 통해 본 논문에서 제안하는 시계열 기반 분석 방법이 최근에 발생한 안드로이드 기반 악성코드 탐지에 적용 가능함을 보였다.

## V. 결 론

본 연구에서는 통계 기반 분석 방법 중 하나인 자기회귀 이동평균 시계열 모델을 활용하여 모바일 악성코드를 탐지하는 방안을 제안하였다. 기존 시계열 기반 비정상 트래픽 탐지 연구에서의 한계점은 첫 번째, 대부분이 유선 네트워크 환경의 비정상만을 고려하였으며, 일부 모바일 환경의 비정상을 고려한 논문에서는 네트워크 공격 탐지에만 초점을 맞추었다. 두 번째, 비정상 탐지 성능을 검증하기 위해 사용한 데이터 셋이 1990년대의 자료로 최근 동향을 반영하지 못한다. 세 번째, 비정상 탐지를 위해 활용되는 정보가 네트워크 기반의 정보만으로 구성되어 있다. 본 논문에서는 이를 해결하기 위해 먼저 모바일 환경 중에서도 악성코드 발생 빈도가 가장 높은 안드로이드 환경에서의 악성코드 탐지 방안을 제안하였다. 두 번째로, 최근 국내에서 발생한 악성코드를 데이터 셋으로 활용하여 최근의 모바일 악성코드 탐지를 수행하였다. 세 번째로, 네트워크 공격 외에 트로이목마(Trojan), 루트 권한 획득을 목적으로 하는 악성코드(Exploit)를 탐지하기 위해 네트워크 관련 정보 이외에 CPU, 메모리 측면의 정보를 함께 고려하여 탐지하는 방안을 제안하였다. 실험의 결과를 보면 제안하는 시계열 기법이 대부분의 악성코드 탐지에 있어서 높은 탐지율과 정확성을 보이고 있다. 이는 추후 스마트 기기의 보안을 강화하는 첫 단계로 악성코드 탐지에 적용될 수 있다고 사료된다.

향후 연구로는 자원 정보간의 상관관계를 고려한 다변량 시계열 모델을 구축하여 탐지 시스템의 성능을 높이는 연구를 진행할 것이다. 또한 기계학습 기법과 시계열기반 기법을 혼합하여 탐지 성능을 더욱더 향상 시킬 계획이다. 마지막으로 악성코드의 패턴을 정확하게 파악하고, 이를 바탕으로 악성코드가 실행되는 시점의 예측하고, 악성코드를 예방하는 방안까지 확대할 계획이다.

## References


[1] M. Chandramohan and Hee Beng Kuan Tan, "Detection of mobile malware in the wild," *Computer,* vol. 45, no. 9, pp. 65-71, 2012.

[2] F-secure, *2013 mobile threat report*, 2013.

[3] F-secure, *Mobile threat report Q1 2014*, 2014.

[4] H.-S. Ham, H.-H. Kim, M.-S. Kim, and M.-J. Choi, "Linear SVM-based android malware detection," *Frontier and innovation in future computing and communications*, vol. 301, pp. 575-585, Apr. 2014.

[5] I. D. Corporation, *Worldwide quarterly mobile phone tracker 3Q13*, Nov. 2013.

[6] S.-H. Yoon and M.-S. Kim, "Behavior based signature extraction method for internet application traffic identification," *J. KICS*, vol. 38B, no. 5, pp. 368-376, 2013.

[7] K. Kim and M. Choi, "Linear SVM-based android malware detection and feature selection for performance improvement," *J. KICS*, vol. 39C, no. 8, pp. 738-745, 2014.

[8] S. Arzt, et al., "FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," in *Proc. 35th ACM SIGPLAN Conf. Programing Language Design and Implementation*, pp. 259-269, Edinburgh, UK, Jun. 2014.

[9] K. Woo and C. Kim, "Internet worm propagation modeling using a statistical method," *J. KICS*, vol. 37B, no. 3, pp. 212-218, 2012.

[10] H. Akaike, "Maximum likelihood identification of gaussian autoregressive moving average models," *Biometrika*, vol. 60, no. 2, pp. 255-265, 1973.

[11] A. M. Bahaa-Eldin, "Time series analysis based models for network abnormal traffic detection," *Int. Conf. Computer Engineering &*

*Systems(ICCES)*, pp. 64-70, Cairo, 2011.
[12] Y. Lai, et al., "On monitoring and predicting mobile network traffic abnormality," *Simulation Modeling Practice and Theory*, vol. 50, pp. 176-188, 2014.
[13] Ahnlab, *Ahnlab ASEC Report 2013*, 2013.
[14] Bell Lab, CRAN R, Retrieved Nov. 12, 2014, from http://www.r-project.org.

**김 환 희** (Hwan-Hee Kim)

2013년 2월 : 강원대학교 컴퓨터과학과 학사
2015년 2월 : 강원대학교 컴퓨터과학과 석사
<관심분야> 네트워크 관리, 정보보안

**최 미 정** (Mi-Jung Choi)

1998년 : 이화여자대학교 컴퓨터공학과 학사
2000년 : 포항공과대학교 컴퓨터공학과 석사
2004년 : 포항공과대학교 컴퓨터공학과 박사
2004년~2005년 : 프랑스 INRIA 연구소 박사후 연구원
2005년~2006년 : 캐나다 워터루대학 컴퓨터과학부 박사후 연구원
2006년~2008년 : 포항공대 컴퓨터공학과 연구 조교수
2008년~현재 : 강원대학교 컴퓨터과학과 부교수
<관심분야> 네트워크 관리, 정보보안, SDN/NFV 관리