무작위 터치 발생 탐지를 이용한 안드로이드 앱 자동 분석 회피에 관한 연구

윤한재* · 이만희*

요 약

급속하게 늘어나는 악성앱을 효과적으로 분석하기 위해 악성앱 자동 분석 시스템이 구축·활용되고 있다. 악성앱의 행위를 보다 많이 활성화시키기 위해 무작위 터치를 발생시키는 자동 터치 모듈을 추가하는 연구가 진행되고 있다. 본 연구에서는 실제 사람의 터치와 자동으로 발생되는 무작위 터치와의 차이를 구별할 수 있는 방안을 제시한다. 실험을 바탕으로 사람들은 한번 터치 후 다음 터치와의 거리가 자동화 모듈보다 짧은 경향이 있으며, 손가락 움직임으로 터치할 수 있는 빠르기도 한계가 있고, 사람은 일반적으로 스마트폰의 최 외곽 지역은 잘 터치하지 않는다는 사실을 알게 되었다. 본 연구에서는 실험을 통해 얻은 통계적 수치를 이용하여 스마트폰의 터치에서 사람인지 자동화 모듈인지 판단할 수 있는 알고리즘을 개발하였다. 본 연구는 궁극적으로 자동 터치 발생을 통한 자동 분석 시스템 고도화에 기여할 것으로 예상된다.

Avoiding Automatic Android App Analysis by Detecting Random Touch Generation

Yun Han Jae* · Lee Man Hee*

ABSTRACT

As the number of malicious Android applications increases rapidly, many automatic analysis systems are proposed. Hoping to trigger as many malicious behaviors as possible, the automatic analysis systems are adopting random touch generation modules. In this paper, we propose how to differentiate real human touches and randomly generated touches. Through experiments, we figured out that the distance between two consecutive human touches is shorter than that of random generation module. Also we found that the touch speed of human is also limited. In addition, humans rarely touch the outer area of smartphone screen. By using statistics of human smartphone touch, we developed an algorithm to differentiate between human touches and randomly generated touches. We hope this research will help enhance automatic Android app analysis systems.

Key words : Android, Random touch generation, automatic analysis avoidance

접수일(2015년 12월 1일), 수정일(1차: 2015년 12월 10일,) 게재확정일(2015년 12월 18일)

* 한남대학교/컴퓨터공학과

1. 서 론

안드로이드 운영체제가 오픈 소스로 공개되었다는 점을 악용하여 악성앱은 공식 마켓에서 정상적인 앱으로 위장하거나 공식 마켓 이외의 서드파티 마켓에서 인증을 거치지 않은 상태로 스마트폰에 침투하고 침투한 악성앱은 일반 스마트폰 사용자는 알아챌 수 없는 방법으로 스마트폰 내부에서 악성 행위를 수행하는데 비정상적인 방법으로 요금을 부과하거나 스마트폰 내부에 저장되어있는 개인정보, 고유정보, 민감정보를 유출함으로써 악성앱에 대한 보안 대책을 마련하는 것이 시급하게 되었다.

악성으로 의심되는 신규 앱을 분석하는 경우, 앱을 실행하지 않고 실행코드를 분석하는 정적분석과 앱을 실제 혹은 가상환경에서 실행하여 그 행위를 분석하는 동적분석 기법이 있다. 정적기법은 모든 코드를 분석할 수는 있지만 난독화 되어 있거나 외부 접속을 통한 행위 변경 등이 일어날 경우 분석이 불가능하다. 반면에 동적분석은 분석 대상 앱의 모든 행위를 일어나게 하기는 힘들지만 행위 변경 등의 동작 모니터링이 가능하므로 최근 급속히 증가하는 앱을 분석하기 위해서 자동분석에 관한 연구와 개발이 많이 진행되고 있다. 자동 동적분석의 전통적인 방법은 악성앱을 가상 안드로이드 환경에 설치, 구동, 행위 분석, 종료 순으로 진행된다. 자동으로 분석을 진행해야 하므로 앱의 특정 기능을 구동하기보다 앱을 설치한 후 실행하며 자동으로 구동되는 행동의 분석에 초점이 맞추어져 있었다.

이 접근법의 문제점은 앱의 자동 구동되는 부분 이외의 행동을 모니터링 할 수 없다는 것이다. 이를 해결하기 위해 안드로이드 시스템에 임의의 이벤트를 생성할 수 있는 MonkyRunner와 같은 도구들이 개발되었고 이를 이용하여 앱의 동적 분석을 더욱 용이하게 하는 연구가 진행되었다[2][3][4][5][6][7][8][9][10][11][12].

본 연구에서는 자동 이벤트 생성을 이용한 분석 기법 회피 방안을 제안한다. 악성앱 제작자의 입장에서 이 분석기법을 회피하기 위해 어떤 방법을 사용할 수 있을지 생각해봄으로써 자동 이벤트 생성을 이용한 분석 기법이 다시 고도화되기를 기대한다. 본 연구에서는 특히 MonkeyRunner를 이용해 자동 분석할 때 기본으로 사용되는 무작위 이벤트 발생 기법을 회피하는 방안을 제안한다. 특별한 알고리즘을 추가로 개발하지 않는 한 MonkeyRunner의 무작위 이벤트 발생을 사용하므로 이 방법만 회피하더라도 상당한 분석 시스템을 회피할 것으로 기대된다.

회피 방안의 핵심 아이디어는 MonkeyRunner가 사람이라면 발생시키지 않았을 이벤트를 발생시킨다는 가정에서 출발하였다. 이 가정이 맞는다면 앱을 개발할 때 일반적인 사람의 행위 패턴에서 벗어나는 이벤트가 감지될 경우, MonkeyRunner가 분석한다는 의미이므로 앱을 종료하거나 악성행위가 아닌 정상행위만 수행하여 자동분석 시스템을 회피할 수 있다.

본 아이디어의 검증을 위해 실제 사용자들로부터 이벤트 발생을 중심으로 행위 데이터를 수집하였다. 분석 결과, MonkeyRunner의 무작위 이벤트 발생과 실제 사람의 행위를 구별할 수 있는 4가지 카테고리에서 판단 기준을 찾을 수 있었다. 이 기준을 이용하여 무작위 이벤트 발생을 이용하는 자동 분석 시스템 탐지앱을 개발하였다.

2. 관련 연구

악성앱 제작자들은 무작위 이벤트 발생을 통한 분석을 회피하기 위한 방안으로써 사용자에게 입력을 요구한 후 적당한 입력이 이루어졌을 때만 악성 행위를 하도록 하고, 적절한 입력이 이루어지지 않을 때는 악성 행위를 숨겨 분석을 회피하는 방법을 사용해 왔다. 이 분석회피 기술을 다시 방해하기 위한 연구들은 악성앱을 분석할 때 무작위 이벤트가 아닌 앱에서 요구하는 적절한 이벤트를 발생시키고자 하였다.

· 앱의 행위에 따라 발생하는 system event를 감지하여 적절한 입력 이벤트 전송[2]

- 터치 횟수 또는 제스처 보다는 버튼 터치에 초점을 두어 터치 이벤트 전송[3]
- button-mask를 이용하여 button-mask에 일치하는 이벤트 전송[4]
- 모든 이벤트에 대해 관심있는 이벤트를 선택적으로 이벤트를 전송[5]
- 전송할 수 있는 여러 가지 이벤트를 미리 세 가지 범주로 구분한 후 분석하는 앱 상태에 따라 세 가지 범주 중에 하나를 선택하여 이벤트 전송[6]
- 랜덤 이벤트를 전송하기 위해 monkey를 확장하여 자체 개발한 이벤트 발생 모듈을 이용하여 랜덤 이벤트 전송[7]
- 자체 개발한 알고리즘에 따라 분석하는 앱 상태에 따라 알맞은 이벤트 전송[8]
- 동적 분석을 위해 Monkey를 확장하여 자체 개발한 툴을 이용해 가능한 많은 기능을 동작시키도록 앱의 UI를 제어[9]
- 자체 개발한 분석 도구와 Monkey 등의 다른 분석 도구를 비교하기 위해 whitebox, blackbox test를 수행[10]
- Monkey의 무작위 터치를 이용하여 앱의 대부분의 기능과 문제점들을 탐색[11]
- 자체개발한 툴에서 무작위 이벤트를 전송하기 위해 Monkey를 이용[12]

위와 같은 방법으로 적절한 이벤트를 발생 시킨 후 분석한 비교 결과에 따르면 이벤트 발생 모듈 없이 분석을 진행했을 때보다 이벤트 발생 모듈을 같이 사용하여 분석했을 때 데이터 유출 행위를 더욱 많이 발현 시켰으며[2], 파일 입출력, 네트워크 통신 동의 행위도 더욱 많이 발현 시켰고[3][8], 전체적인 앱의 행위 또한 크게 증가하였다[4][5][6][7].

기존 연구들은 악성앱을 좀 더 잘 분석하기 위해서 적절한 이벤트를 발생시키는 것에 초점이 있다면, 본 연구는 그러한 자동 이벤트 발생 기법을 이용한 분석 기법을 회피하는데 초점이 있다는 것이 다르다. 특히, 무작위 이벤트 발생 회피 기법은 아직 제안되지 않은 방법이므로 본 연구의 결과 자체로 큰 의미가 있을 것으로 생각된다.

3. 터치 이벤트 자동 발생 탐지 방안

3.1 구분 기준

실제 사용자와 터치 이벤트 발생 모듈을 구분하기 위한 기준은 일반적인 사람들의 평균 행동에 벗어날 경우, 사람이 아닌 자동화 모듈이라고 판단한다. 이 가정을 바탕으로 세운 구분 기준은 다음 표 1과 같다.

<표 1> 실제 사용자와 터치 이벤트 발생 모듈 구분 기준

순서	기준	가정
1	외곽지역 터치	사람들은 스마트폰의 외곽 지역을 잘 터치하지 않을 것이다.
2	터치지점 사이의 거리	사람들은 한번 터치한 지점으로부터 멀지 않은 곳에 다음 터치를 할 것이다.
3	터치 사이의 시간	사람들은 터치 간에 적절한 시간 간격을 둘 것이다.
4	전체 소요시간	사람들이 터치할 수 있는 속도에는 한계가 있을 것이다.

상기 가정을 검증하기 위해 다음과 같은 방법으로 수집한다. 먼저 100번의 터치 입력을 받는 안드로이드 앱을 개발하였다. 터치 정보 수집앱은 터치 이벤트가 발생한 x, y 좌표, 발생 시간, 전체 소요 시간을 수집한다. 이 앱을 일반 학생 30명의 스마트폰에 설치하여 실행하여 학생별로 스마트폰 화면을 100번 터치하는 정보를 데이터베이스에 저장하였다.

실제 사용자와 자동 터치 이벤트와의 비교를 위해 Google에서 제공하는 MonkeyRunner를 사용해 터치 이벤트 자동 발생 모듈을 개발하였다. 터치 이벤트 자동 발생 모듈은 100회의 터치 이벤트 실험을 200회 수행한다. 일반적으로 악성코드 분석에 사용되는 터치 발생 방법은 랜덤 위치에 최대한 빠른 시간에 많은 이벤트를 발생시킨다. 그 이유는 앱의 사용자 인터페이스가 어떻게 구성되어 있는지 전혀 모르기 때문에

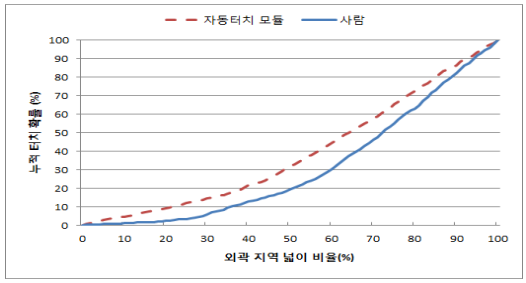
최대한 많은 지역에 많은 터치 이벤트를 발생시키는 것이 가장 효과적이다. 즉, 짧은 시간에 최대한 많은 위치에 터치 이벤트를 발생시키는 것이 분석대상 앱의 가능한 많은 행위를 발현시키는 것이다. 따라서 자동 발생 모듈에서 생성하는 터치 이벤트의 위치는 랜덤하게 생성된다.

3.2 외곽 지역 터치 분석

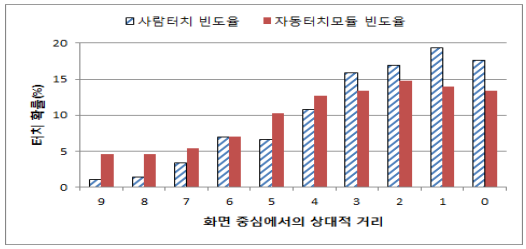
본 논문에서의 외곽 지역이란 전체 스마트폰 화면의 넓이를 100%로 생각했을 때 바깥쪽 테두리를 기준으로 바깥쪽으로부터 몇 %의 공간을 의미한다. 예를 들면 본 연구에서 기준으로 사용하는 1080*1920 화면의 경우 x좌표가 0<=x<=2 또는 1078<=x<=1080 이고 y좌표가 0<=y<=4 또는 1926<=y<=1920 에 포함되는 화면의 가장 바깥쪽 지역 픽셀의 개수는 전체 픽셀 개수의 약 1%에 해당하여 이를 본 논문에서는 외곽 지역 넓이 비율이 1%인 지역 혹은 외곽 1% 지역이라고 한다. 그림 1은 사람과 자동 생성 모듈이 터치한 위치를 누적 확률로 나타내었다.

그림 1의 사람의 터치를 살펴보면 외곽 20% 지역까지는 터치가 잘 이루어지지 않다가 외곽 30% 지역부터 터치가 늘어나며 자동 터치 모듈과 같이 꾸준히 증가하다가 외곽 80% 지역, 즉 중심부 지역으로 가까이 갈수록 급격히 터치 빈도가 늘어나는 것을 알 수 있다. 그림 2의 자동터치 모듈의 터치도 사람의 터치와 전체적으로 비슷한 모양을 띄기는 하지만 비교적 외곽 지역에서도 어느 정도의 이벤트가 발생하는 것을 알 수 있다.

그림 2는 전체 누적확률이 아닌 외곽 10% 지역부터 중심부 쪽으로 10%씩 넓이를 늘여갈 때 해당 지역에 얼마나 높은 비율로 터치를 하는지 살펴본 것이다. 즉, 전체 넓이의 10%에 해당하는 창틀 모양의 공간에 터치한 확률을 표현한다. x축은 각 창틀 모양의 중심부로부터 거리를 상대적으로 표현한 것이다. 앞서 설명한 것과 마찬가지로 외곽 20% 지역까지 사람들은 거의 터치를 하지 않았고 중심부로 이동할수록 무작위 터치보다 훨씬 더 많이 터치하는 것을 알 수 있다. 이 분석을 통해 외곽 20% 지역 터치 비율을 이용하면 사람과 자동터치 모듈을 판단할 수 있다는 것을 알 수 있다.



(그림 1) 외곽 지역 터치 누적확률 비교



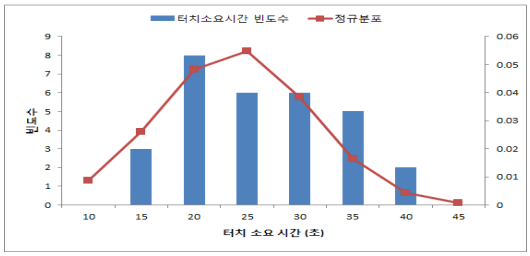
(그림 2) 외곽 지역 터치 빈도율 비교

3.3 터치 소요 시간 분석

터치 소요시간은 실제 사용자가 무작위 100회의 터치 이벤트를 수행할 때 소요된 시간을 측정하였다. 가장 빠르게 터치한 사람이 10.69초, 가능 느리게 터치한 사람이 38.32초, 평균 23.81초, 표준편차는 7.18초이다. 그림 3은 터치소요 시간의 분포를 정규분포와 함께 나타낸다. 계급구간 간격은 5초이며 10초~15초에 완료한 사람이 3명이며 10초미만으로 완료한 사람은 없었다.

실험대상자가 30명이어서 정규분포와 약간 다르게 나타나지만 사람의 일반적 신체적 능력과 관련된 내용이므로 대상자가 많으면 정규분포를 따를 것으로 쉽게 예상할 수 있다. 이 데이터가 정규분포를 따른다고 가정할 때, 전체 터치 평균 시간에 대한 99% 신뢰

구간은 5.31초~42.3초이다. 본 실험결과에 의해 5.31초보다 빠르게 100회의 터치를 종료한다면 사람이라기보다 프로그래밍에 의한 터치라고 생각할 수 있다.



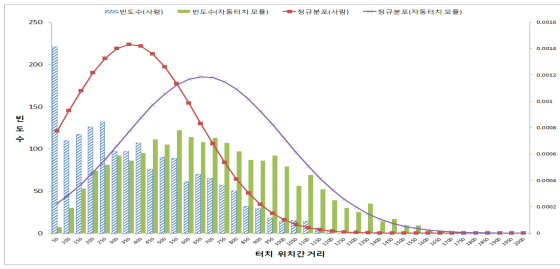
(그림 4) 사람의 터치 소요 시간 분석

3.4 터치 위치간 거리 분석

터치 위치간 거리는 실제 사용자가 무작위 터치 이벤트를 수행할 때 이전에 터치한 위치와 다음에 터치한 위치 사이의 거리를 의미한다. 본 데이터 분석 이유는 실제 사용자라면 손가락 움직임이 한계로 인해 연속된 터치 사이에 거리가 제한적일 것으로 예상되어 완전히 랜덤하게 터치하는 자동 터치 모듈과 차이가 있을 것이라는 가정을 하였기 때문이다.

먼저 사람의 평균 거리는 369.14이고 표준편차는 275.36이다. 자동 터치모듈의 평균거리는 662.45이고 표준편차는 335.94이다. 전반적인 분포를 비교하기위해 터치 위치간 거리의 빈도와 분포도를 그림 4에 나타내었다. 앞서 가정하였던 것과 마찬가지로 사람이 수행한 터치 사이의 거리가 비교적 짧았고 자동터치 모듈의 터치 사이 거리가 비교적 큰 것을 알 수 있다.

두 데이터의 평균의 차이를 이용하여 추후 사람과 자동터치 모듈을 구별할 수 있는지 확인하기 위하여 통계적 검정을 실시하였다. 먼저 두 데이터가 분산이 같은지 확인하기 위해 F-검정을 실시한 결과, F 비(0.6718)가 F 기각치 (0.9267)보다 작아 두 데이터의 분산이 같음을 알 수 있었다 ($p>.05$). 등분산 내에서의 t-검정을 통해 p는 1.1786E-167로써 유의수준 $\alpha = 0.05$ 보다 작으므로 두 평균이 같다는 가설을 기각할 수 있다. 사람의 터치 위치간 거리와 자동터치 모듈의 터치 위치간 거리에는 유의미한 차이가 있으므로 이를 이용해 자동터치 모듈 판단 알고리즘에 적용한다. 사람의 터치 위치간 거리에 대한 95% 신뢰구간은 0~1026.588이다. 본 실험결과에 의해 터치 위치간 거리가 1027보다 크다면 프로그래밍에 의한 터치라고 의심할 수 있다.



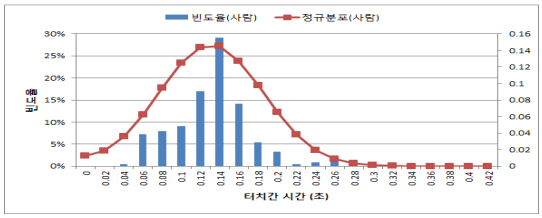
(그림 3) 터치 위치간 거리 분석

3.5 터치 사이 시간 분석

터치 사이 시간은 실제 사용자가 무작위 100회의 터치 이벤트를 수행할 때 터치사이에 소요된 시간을 측정하였다. 이 실험은 앞서 수행한 시험과 독립적으로 수행하였으며 실험 참가자에게 위치와 관계없이 최대한 빨리 100회를 터치하도록 요구하였으며 스마트폰의 일반적인 사용 환경과 비슷하게 하기 위해 두 손가락을 이용하도록 하였다. 실험 목적은 사람은 얼마나 빨리 터치할 수 있는가에 초점이 맞추어져있다. 사람보다 빨리 터치한다면 자동 터치 모듈이라고 판단하기 위함이다.

평균 터치 사이 시간은 0.1291초, 표준편차는 0.0543이다. 그림 5는 터치 사이 시간의 분포를 정규분포와 함께 나타낸다. 터치 사이 시간 분석은 앞서 수집한 데이터 분석과 비교해서 봐야하는 포인트가 있다. 먼저 전체 소요시간을 살펴보면 최대한 빠르게 터치하라고 요구했을 때 터치 사이 평균 시간이 0.1291초이므로 100회 터치 시간이 약 12.9초 정도 될 것으로 예상된다. 하지만 3.3절의 터치 소요시간 분석에서 가장 빠르게 터치한 사람이 10.69초였고 평균이 23.81초인 것과 비교해 보면 10.69초가 소요되었던 실

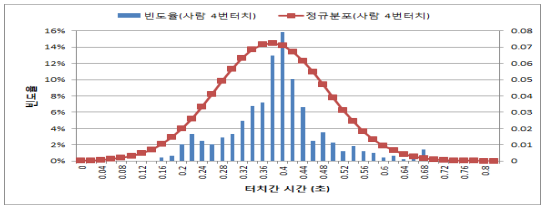
험자는 가장 빠르게 터치하려고 노력하였고 일반적인 사용자 터치는 최대 속도보다 약 2배 느리게 터치한다는 것을 알 수 있다.



(그림 5) 터치 사이 시간 분석

한 가지 더 분석해야할 내용은 실험 시 두 손가락을 사용했다는 점이다. 대부분의 실험자는 스마트폰을 두 손으로 잡고 엄지손가락 두 개를 이용하여 최대한 빠르게 터치하므로 자연스럽게 두 손가락이 비슷한 횟수로 터치를 하게 된다. 이 의미는 하나의 엄지손가락으로 최대한 빠르게 터치한다면 두 손가락 이용 시 평균값의 두 배인 평균 0.2582초라고 예상할 수 있다. 그렇다면 한 손가락으로 가장 빠르게 터치할 수 있는 시간을 유추하기 위해 우리는 4번의 터치를 하는데 소요되는 시간을 사용하였다. 그 이유는 2번의 연속된 터치는 두 엄지손가락이 거의 동시에 터치를 할 수 있고 세 번의 연속된 터치 또한 비슷한 이유로 정확하게 한 손가락의 속도를 알기 어렵다. 하지만 연속된 4번의 터치는 두 손가락을 사용하는 이상 한 손가락이 두 번 이상 터치를 해야 하고 가장 빠르게 4번의 터치를 하는 경우는 양 엄지손가락을 거의 같은 타이밍에 두 번 터치를 하는 것이다.

그림 6은 연속된 4번 터치에 소요된 시간에 대한 그래프이다. 평균 0.3874초이고 최솟값이 0.15초이다. 이는 한 손가락으로 최대한 빠르게 터치하는 또 다른 실험에서 한 손가락 터치 사이 시간의 평균이 0.1112초, 표준편차가 0.0215초인 것과 비교하면 연속된 4번 터치에 소요된 시간이 한 손가락 최고 속도 터치보다 조금 더 걸린다는 것을 알 수 있다. 이 실험결과에 의해 4번의 연속된 터치에 소요된 시간의 95% 신뢰구간인 0.1718초~0.6030초보다 작다면 사람이라기보다 프로그래밍에 의한 터치라고 생각할 수 있다.



(그림 6) 4번 터치 소요 시간 분석

4. 판단 기준 효용성 평가

본 연구를 위해 개발한 자동 터치 탐지앱은 앞서 설명한 실제 사용자 데이터 분석 결과를 바탕으로 사람이 아닌 자동 터치 모듈의 터치 패턴을 탐지하여 앱을 조기 종료시킨다. 이 자동 터치 탐지 앱의 효용성 판단 기준은 오탐율, 오탐시까지 소요 시간 및 종료 전까지 수행한 터치의 개수, 정탐율, 정탐시 종료 전까지 소요 시간 및 종료 전까지 수행한 터치의 개수 등이 될 수 있다. 오탐율은 사람의 터치를 자동 터치로 오인하여 앱을 종료시키는 확률을 의미한다. 정탐율은 자동 터치 모듈의 터치를 정확히 탐지하는 확률을 의미한다. 정탐으로 판단할 때까지의 소요시간과 이때까지 수행한 터치의 개수는 자동 터치 탐지앱이 최소한으로 허용하는 자동 분석 시간 및 이벤트라고 할 수 있다. 낮은 오탐률과 높은 정탐율, 그리고 정탐까지의 터치 이벤트가 적을수록 좋은 판단 기준이라 할 수 있다. 어떤 탐지 기준이 자동 터치 탐지앱을 효과적으로 탐지하는지 알아보기 위해 각 판단기준을 단독으로 이용해서 그 효용성을 검증하고, 마지막으로 이를 적절하게 융합한 탐지 결과를 제시한다.

4.1 외곽지역 판단

스마트폰의 외곽 10% 부분을 연속해서 2회 또는 3회 이상 터치할 때로 설정하여 실제 사람의 터치 데이터와 자동 터치 모듈의 데이터를 입력한 결과, 2회

와 3회 터치의 경우, 사람의 터치를 자동화 터치로 잘못 인식하는 확률이 각각 13.33%와 3.33%였다. 흥미있는 결과는 정탐율이 2회 연속에서는 63.33%였지만 3회 연속에서는 3.33%로 크게 떨어졌다. 이는 자동 터치 모듈의 무작위 생성 포인트 역시 3회 연속 외곽 지역을 선택하는 것이 확률적으로 어렵다는 것을 알 수 있다.

4.2 터치 소요 시간 판단

100회 터치에 소요시간이 5.31초 미만이면 자동 터치 모듈일 확률이 높고, 또 100회까지 터치를 허용하지 않게 하기 위해 10회 터치 시간이 0.53초 미만인 이벤트 또는 20회 터치 시간이 1.06초 미만인 이벤트로 설정하여 실제 사람의 터치 데이터와 자동 터치 모듈의 데이터를 입력한 결과 표 2에 나타냈다. 정탐율을 나타내지 않은 이유는 자동 터치 모듈의 터치 속도를 조절하여 이보다 빠르게 터치할 경우 100% 정탐이 될 것이고 반대로 사람과 비슷하거나 느리게 터치할 경우는 100% 미탐이 되므로 큰 의미가 없기 때문이다. 하지만 대부분의 앱 분석 시스템은 분석 효율성을 고려하여 사람보다 훨씬 빠른 터치를 하는 것이 일반적이므로 터치 소요 시간은 분석 시스템의 자동 분석을 방해하는데 효과적으로 사용될 것이다.

<표 2> 터치 소요 시간 판단 기준 적용 결과

항목	10회 터치 0.53초 미만	20회 터치 1.06초 미만
오탐율	6.67%	3.33%
오탐전 터치 개수	52.5개	34개

4.3 터치 위치간 거리 판단

터치위치간 거리가 1027보다 큰 터치가 연속해서 2회 또는 3회 계속될 때로 설정하여 실제 사람의 터치 데이터와 자동 터치 모듈의 데이터를 입력한 결과, 사람은 1027보다 큰 터치를 2회 연속한 사람이 3명, 3회 연속한 사람이 2명으로 각각 오탐율 10.00%, 6.67%을 보였다. 자동 터치 모듈은 터치 위치간 거리 판단 알고리즘에 의하여 각각 96.67%, 53.33%의 정탐율을 보였다.

4.4 터치 사이 소요 시간 판단

4번의 연속된 터치에 소요된 시간이 0.17초미만 또는 0.15초 미만인 경우를 설정하여 실제 사람의 터치 데이터와 자동 터치 모듈의 데이터를 입력한 결과, 017초미만의 오탐율은 13.33%, 오탐전 터치개수는 46.75개 였으며 0.15초 미만의 오탐율은 6.67%, 오탐전 터치개수는 64개였다. 터치 소요 시간 판단과 같은 이유로 정탐율을 나타내지 않았다.

4.5 판단 기준 종합 분석

현재까지 가장 효과적인 판단기준은 터치 위치간 거리로써 2회 연속 1027보다 먼 거리를 터치하는 기준으로 탐지율은 96.67%, 오탐율은 10%이다. 외곽 지역 판단기준은 2회 연속 외곽 10% 지역을 탐지하는 기준으로 탐지율은 63.33%, 오탐율은 13.33%이다. 이 두 가지 기준을 동시에 적용하면 탐지율은 96.67%로 같은 대신에 오탐율은 23.33%로 늘어난다.

위 두 가지 판단기준에 10회 터치 소요 시간이 0.53초 미만 기준과 4회 터치 소요 시간이 0.17초 미만 기준을 적용하면 최종 오탐율은 33.33% 될 수 있다. 하지만 현재 판단 기준은 통계에 의해 확률이 적은 값을 택해 실험해 본 내용이므로 이 기준값을 조금만 줄이면 오탐율 0%이 된다.

악성앱 제작자의 입장에서 앱 분석을 효과적으로 방해하면서도 사용자들의 이용에는 불편이 없도록 하는 것이 최대 목표지만 어떤 곳에 중점을 두는 가에 따라 판단 기준은 다르게 적용할 수 있다. 예를 들면, 사용자 불편을 조금 초래하더라도 분석을 적극적으로 방해하기 위해서 본 논문에서 제시한 기준 중에서 정탐율을 낮추는 기준들을 중복해서 사용하면 된다. 반대로 분석 방해보다는 사용자 불편을 없애기 위해서는 시간 기준을 짧게 잡고 위치 기준 또한 오탐율이

거의 나오지 않도록 정할 수 있다. 이런 조건도 최대한 빠르게 터치를 수행하는 대부분의 악성코드 분석 시스템의 패턴을 고려할 때 상당한 분석 방해 요인이 될 것으로 생각된다.

5. 결 론

본 논문은 최근 악성앱의 효율적 분석을 위해 사용되고 있는 자동 터치 기능을 우회할 수 있는 방안을 제안하였다. 실제 사람의 터치를 분석하여 외과지역 터치, 터치 사이 거리, 터치 소요 시간, 터치 사이 소요 시간을 사람과 자동 터치와의 차이를 구별할 수 있는 기준을 제시하였다. 외곽 지역 및 터치 위치간 거리의 경우 비교적 높은 탐지율과 낮은 오탐율로 판단 기준을 사용할 수 있음을 보였다. 총 터치 소요 시간 및 터치 사이 소요시간 분석을 통해 사람이 수행할 수 있는 능력에 대한 수치를 얻을 수 있었다. 이 데이터는 자동 터치 모듈을 판단할 수 있는 임계치로 사용될 수 있을 것이다.

향후 연구로써 화면 터치 외에 사람의 행위를 판단할 수 있는 다양한 기준을 추가하여 자동 분석 시스템을 보다 효과적으로 탐지하는 방안을 제공하고자 한다. 이와 동시에 본 연구에서 제안한 방법으로 악성앱 자동 분석을 방해하고자하는 앱의 판단하는 연구를 진행할 예정이다.

참고문헌

[1] http://developer.android.com/tools/help/monkeyrunner_concepts.html

[2] Ying-Chih Shen, Roger Chien, Shih-Hao Hung, Toward Efficient Dynamic Analysis and Testing for Android Malware, IT Convergence Practice, Vol.2, No.3, 2014

[3] Lukas Weichselbaum, Matthias Neugschwandtner, Martina Lindorfer, Yanick Fratantonio, Voctor van der Veen, Christian Platzer, ANDRUBIS: Android Malware Under The Magnifying Glass, Technical Report TR-ISECLAB-0414-001, 2014.

[4] Andrea Gianazza, PuppetDroid: A Remote execution environment and UI exerciser for android malware analysis, politesi, 2013

[5] Kimberly Tam, Salahuddin J. Khan, Aristide Fattoriy, and Lorenzo Cavallaro, CopperDroid: Automatic Reconstruction of Android Malware Behaviors, NDSS symposium, 2015.

[6] Aravind Machiry, Rohan Tahiliani, Mayur Naik, Dynodroid: An Input Generation System for Android Apps, ESEC/FSE, 2013.

[7] Jan Van Eijck, Evaluating the Efficiency of GUI Ripping for Automated Testing of Android Applications

[8] Shuai Hao, Bin Liu, Suman Nath, William G.J. Halfond, Ramesh Govindan, PUMA: Programmable UI-Automation for Large-Scale Dynamic Analysis of Mobile Apps

[9] Machael Bierma, Eric Gustafson, Jeremy Erickson, David Fritz, Yung Ryn Choe, Andlantis: Large-scale Android Dynamic Analysis.

[10] Riyadh Mahmood, Nariman Mirzaei, Sam Malek. EvoDroid: Segmented Evolutionary Testing of Android Apps

[11] Tahir Javaid, Testing of Android Testing Tools: Development of a Benchmark for the Evaluation. Universidad Politecnica de Madrid 2015.

[12] Christian Rossow, Dynamic Analysis of Android Malware. University Amsterdam. 2013.

[저 자 소 개]

윤 한 재 (Han-jae Yun)

2016년 2월 한남대학교 컴퓨터공학과
학사 (예정)

email : hanjae.karoha@gmail.com

이 만 희 (Man-hee Lee)

1995년 경북대학교 컴퓨터공학과
공학사
1997년 경북대학교 공학석사
2008년 Texas A&M 대학교
컴퓨터공학과 공학박사
1997년~2003년 한국과학기술정보연구원
2008년~2009년 Cisco Systems
2010년~2011년 국가보안기술연구소
2012년~현재 한남대학교 조교수

email : manheelee@hnu.kr