안드로이드폰에서 타임스탬프 변경 이벤트의 실시간 탐지

안균승[1], 김선재[2], 조성제[3]
[1]단국대학교 소프트웨어학과 학부생
[2]단국대학교 컴퓨터학과 석사과정
[3]단국대학교 소프트웨어학과 교수
{staran1227, rlatjswo0824, sjcho}@dankook.ac.kr

Real-time Detection of Timestamp Modification Events in Android Phones

Gyun-Seung Ahn[1], Sun-Jae Kim[2], Seong-je Cho[1]
[1]Dept. of Software Science, Dankook University
[2]Dept. of Computer Science & Engineering, Dankook University

요 약

본 논문은 안드로이드폰에서 발생하는 안티포렌식(anti-forensics) 기법인 타임스탬프 변경 이벤트를 실시간으로 탐지하는 기법을 제안한다. 기존 안티포렌식 연구들은 주로 로그 기반 분석에 의존하였으나, 분석 과정이 오래 걸리고 로그 위변조 가능성을 완전히 배제할 수 없는 한계가 있다. 이를 해결하기 위해 본 논문은 리눅스 커널 수준에서 eBPF를 활용해 시스템 시간 변경을, inotify API를 통해 파일 수정시간 변경을 모니터링하여 타임스탬프 조작 여부를 판별한다. 실험 결과, 시스템 앱과 adb shell 명령어를 통한 모든 타임스탬프 조작을 탐지할 수 있음을 확인하였으며, 이를 통해 로그 기반 탐지 기법의 한계를 보완하고 포렌식 수사의 신뢰성을 높일 수 있음을 보인다.

1. 서론

안티포렌식(Anti-forensics)이란 포렌식 절차에서 증거의 가용성이나 유용성을 훼손하려는 모든 시도이다[1]. 이 중 타임스탬프 변경(timestamp tampering)은 파일이나 시스템 리소스의 생성, 수정, 접근 시간 등의 메타데이터를 조작하여 실제 이벤트 발생 시점을 감추는 대표적인 안티포렌식 기법이다[2]. 타임스탬프 변경 이벤트 감지는 보통 로그를 기반으로 이루어진다[3,4,5]. 하지만 이는 분석에 시간이 오래 걸리고 로그에 대한 위변조 가능성을 배제할 수 없다.

안드로이드는 리눅스 커널 기반의 오픈소스 운영체제로, 모바일, 차량의 IVI 시스템, TV 등 여러 임베디드 환경 등 광범위하게 사용된다. 하지만 타임스탬프 변경 탐지에 관한 연구는 부족한 실정이다.

본 논문에서는 이러한 문제를 해결하고자 안드로이드에서 eBPF (extended Berkeley Packet Filter)와 inotify (inode notify) API를 사용한다. eBPF는 리눅스 내에서 커널 모듈 수정 없이도 커널 동작에 훅을 걸어 사용자 정의 코드를 실행할 수 있는 기술로, 시간 조작 관련 시스템 콜을 커널 레벨에서 실시간 탐지할 수 있다. 특히 본 논문은 eBPF 기반의 커널 수준 모니터링과 더불어 inotify API를 도입하여 파일시스템 이벤트를 실시간으로 감시함으로써, 시스템 시간 조작과 파일 메타데이터 조작을 즉시 포착할 수 있다는 점에서 차별화된다. inotify API는 파일시스템 이벤트를 모니터링하는 기법을 제공하며, 개별 파일이나 디렉토리를 모니터링하는데 사용된다. inotify API는 inotify_init() 또는 inotify_init1() 시스템콜을 사용한다. 이를 이용하여 로그 기반의 타임스탬프 변경 이벤트 감지 대신 실시간으로 타임스탬프 변경 이벤트를 탐지하는 방법을 제안한다.

2. 관련 연구

Palmbach 등[3]은 NTFS 파일시스템의 여러 아티팩트 로그 파일들을 활용하여 타임스탬프 조작을 탐지하는 방법을 제안하였다. Oh et al.[4]는 NTFS 파일시스템에서 개별적으로 해석되던 다양한 로그·메타데이터 아티팩트를 통합하여 타임스탬프 조작을 식별하는 종합적 탐지 알고리즘을 제안하였다.

또한 이산 등[5]은 리눅스·안드로이드 환경에서 여러 시스템 로그를 활용해 과거 및 미래 시점의 타임스탬프 조작을 탐지하는 방법을 제시하였다. 안균승 등[6]은 eBPF를 활용하여 settimeofday 시스템콜을 모니터링하여, 시스템 날짜 및 시간 설정, date, toybox date, hwclock 명령어를 통한 날짜 및 시간 변경 행위를 탐지하였다.

이와 같이 선행 연구들은 주로 로그 기반의 파일 메타데이터 조작[3,4]과 시스템 시간 조작[5,6]을 별

개로 다루어 왔다. 반면 본 논문은 두 영역을 함께 고려하기 위해 eBPF와 inotify API를 이용하는 실시간 탐지 기법을 제안한다는 점에서 차별성을 갖는다.

3. 타임스탬프 변경 이벤트를 탐지하는 방법

안드로이드 환경에서 발생할 수 있는 타임스탬프 변경 이벤트를 시스템 시간 변경과 파일 수정시간 변경으로 구분하여 연구를 수행한다. 시스템 시간 변경은 settimeofday 시스템 콜을 통해 탐지한다. 파일 수정시간에 대한 변조 이벤트를 판단하기 위해, 해당 파일의 inode에 저장된 mtime (파일의 내용이 마지막으로 수정된 시간) 값과 시스템 시간을 활용한다. 즉, 파일의 mtime을 변경하려는 시도가 감지될 때, 변경될 시간이 현재 mtime보다 과거이면 “시간 조작”이라고 판단한다. 또한, 변경될 시간이 현재의 시스템 시간보다 미래일 경우에도 “시간 조작”이라고 판단한다 (그림 1 참조).

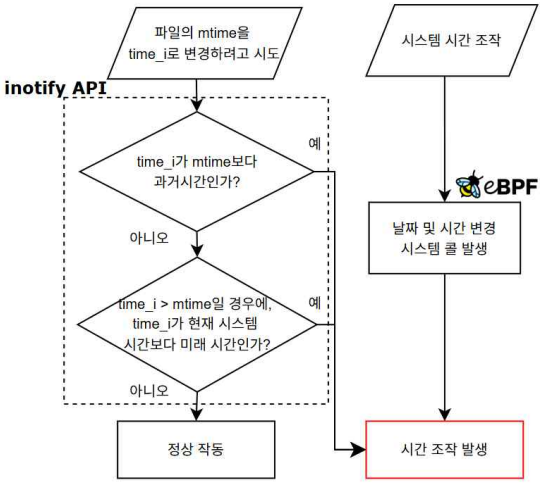

그림 1 타임스탬프 변경 탐지 방법

4. 실험 및 결과 분석

실험에 사용한 기기 및 도구 정보를 표 1에 요약하였다. 시스템 시간 변경 탐지는 eBPF 기반 도구인 bpftrace를 ExtendedAndroidTools를 통해 크로스 컴파일하여 사용하였다. 그리고, Ubuntu 22.04에서 inotify API를 이용한 C 프로그램을 작성하여 컴파일하였다.

표 1 실험 환경

실험기기	Galaxy A23N(pre-rooting)
운영체제	Android 14
커널버전	4.19.157
사용도구	bpftrace, ExtendedAndroidTools, inotify API
크로스 컴파일 환경	Ubuntu 22.04
디버깅도구	Android Debug Bridge 1.0.41

실험에 사용된 시간 변경 명령어와 탐지 방법이 표 2에 나타나 있다. 시스템 앱(네트워크 동기화/날짜·시간/지역 시간대)과 date, toybox date, hwclock(adb shell)로의 시스템 시간 변경, 그리고 touch [-m,-d,-t]로의 파일 수정시간 변경을 모두 실험했으며, 제안 기법은 과거·미래로의 모든 변조 사례에서 탐지에 성공하였다.

표 2 실시간 탐지 가능한 타임스탬프 변경 이벤트

시간변경 방법	설명
날짜 및 시간 설정	시스템 시간 변경 (settimeofday 시스템 콜)
date -s “YYYY-MM-DD HH:MM:SS”	
toybox date -s “YYYY-MM-DD HH:MM:SS”	
hwclock -s	
네트워크시간대로 설정	
지역 시간대로 설정	
touch [-m, -d, -t] “YYYY-MM-DD” <filename>	파일의 수정시간 변경 (inotify API)

5. 결론

본 논문에서 안드로이드 환경에서 발생하는 타임스탬프 변경을 eBPF 기반 bpftrace와 inotify API로 실시간 탐지하는 기법을 제안하고, 로그 기반 탐지 기법을 보완할 수 있음을 보였다. 향후, 제안 기법을 파일 삭제나 파일 이름 조작 등과 같은 다른 유형의 안티포렌식 행위를 탐지하고, 또한 악성 코드의 행위 분석에 활용하는 방안을 연구할 계획이다.

Acknowledgement

본 연구는 2024년 과학기술정통신부 및 정보통신기획평가원의 SW중심대학사업 지원을 받아 수행되었음(2024-0-00035) 또한 이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원-대학ICT연구센터(ITRC)의 지원을 받아 수행된 연구임(IITP-2025-RS-2023-00259967)

참고문헌

[1] Palmer, Gary. "A road map for digital forensic research." First digital forensic research workshop, utica, new york. 2001

[2] Zdzichowski, Patrycjusz, et al. "Anti-forensic study." NATO CCD COE, Tallinn, 2015.

[3] Palmbach, David, and Frank Breitinger. "Artifacts for detecting timestamp manipulation in NTFS on windows and their reliability." Forensic Science International: Digital Investigation, Vol. 32, 300920, 2020.

[4] Oh, Junghoon, Sangjin Lee, and Hyunuk Hwang. "Forensic detection of timestamp manipulation for digital forensic investigation." IEEE Access 12, pp.72544-72565, 2024.

[5] 이산, 조민혁, 정지헌, 조성제, “리눅스와 안드로이드 시스템에서 타임스탬프 조작 식별을 위한 로그 분석 기법”, 한국차세대컴퓨팅학회 논문지, 제20권, 제4호, pp. 34-45, 2024.

[6] 안균승, 김선재, 정연수, 김보겸, 조성제, “안드로이드에서 eBPF를 이용한 타임스탬프 변경 이벤트 감지”, 차세대컴퓨팅학회 하계워크숍, 제주도, 2025, pp. 1-4