리눅스와 안드로이드 시스템에서 타임스탬프 조작 식별을 위한 로그 분석 기법

Log Analysis Techniques for Identifying Timestamp Manipulation in Linux and Android Systems

이산, 조민혁, 정지헌, 조성제[1)]
San Lee, Min Hyuk Cho, Jiheun Jung, Seong-je Cho

(16890) 경기도 용인시 수지구 죽전로 152 단국대학교
산업보안학과, 모바일시스템공학과, 컴퓨터학과, 소프트웨어학과
{dltkshdkd915, cgumgek8, wlgjsjames7224, sjcho}@dankook.ac.kr

요 약

디지털 포렌식 분석에서 안티-포렌식 기법에 대응해 디지털 증거의 무결성을 유지하는 것이 중요하다. 타임스탬프가 변조되면 증거 신뢰성이 저하되고 사건 타임라인 구성이 어려워진다. 본 논문은 리눅스와 안드로이드 시스템에서 타임스탬프 변조가 의심될 때 로그 데이터를 수집·분석해 이를 탐지하는 효과적인 기법을 제안한다. 재부팅 시점에서 타임스탬프 변조 여부를 확인하는 새로운 방법도 제시하여 각 운영체제별 타임스탬프 조작 시점을 효과적으로 파악한다. Ubuntu와 CentOS 리눅스, 안드로이드 9와 14를 대상으로 실험하여 제안 기법의 유효성을 검증하였다. 본 기법은 여러 운영체제에서 발생할 수 있는 타임스탬프 조작 공격에 대응하고 디지털 증거의 정확성과 신뢰성을 높이는 데 기여할 수 있다.

Abstract

In digital forensic analysis, maintaining the integrity of digital evidence against anti-forensic techniques is crucial. If timestamps are tampered with, the reliability of the evidence diminishes, complicating the construction of an accurate incident timeline. This paper proposes an effective technique to detect timestamp manipulation by collecting and analyzing log data in cases of suspected tampering in Linux and Android systems. Additionally, we present a new method to verify timestamp manipulation at reboot, identifying tampering times for each operating system. Experiments on Ubuntu and CentOS Linux desktops, as well as Android 9 and 14 phones, demonstrate the technique's validity. This proposed method enhances responses to timestamp manipulation across various OSes, improving the accuracy and reliability of digital evidence.

1) 교신 저자

논문투고일자 : 2024. 08. 02 심사일자 : 2024. 08. 05 게재확정일자 : 2024. 08. 23 http://doi.org/10.23019/kingpc.20.4.202408.003

키워드: 디지털 포렌식, 안티 포렌식, 타임스탬프 조작, 로그 분석, 리눅스, 안드로이드
Keyword: Digital forensics, Anti forensics, Timestamp manipulation, Log analysis, Linux, Android

1. 서론

디지털 포렌식은 법적 수사 과정에서 디지털 기기에 저장된 데이터를 수집, 복구, 분석 및 보고하는 과학적 절차로, 법집행기관이 컴퓨터 기기를 압수 및 수색하여 잠재적 증거를 확보하는 중요한 역할을 한다[1][2]. 디지털 포렌식의 목표는 증거의 무결성과 신뢰성을 유지하면서 정확한 정보를 제공하는 것이다. 그러나 디지털 포렌식 분석을 방해하는 다양한 안티 포렌식(anti-forensic) 기법들이 존재한다. 잘 알려진 안티 포렌식 기법으로, 데이터를 암호화하거나 삭제하고, 타임스탬프(timestamp)를 조작하여 증거의 신뢰성을 손상시키는 행위 등이 있다[3]. 특히 타임스탬프 조작은 컴퓨팅 시스템의 시간을 왜곡하여 실제 사건 발생 시점을 혼란스럽게 만들어 포렌식 분석을 어렵게 한다[4].

기존 연구들은 다양한 운영체제와 환경에서의 디지털 포렌식과 안티 포렌식 기법에 대해 다루어 왔다. 예를 들어, 리눅스 기반 시스템에서의 타임스탬프 조작과 그에 따른 로그 파일 분석 방법에 대한 연구들이 진행되었으며[5], 안드로이드 시스템에서의 포렌식 분석 기법도 연구되어 왔다[6]. 본 연구진은 이전에 Ubuntu 리눅스 시스템과 안드로이드 9 시스템에서, 타임스탬프 조작이 디지털 포렌식에 미치는 영향을 분석했다[7].

본 논문은 기존 연구[7]를 확장한 버전이다. 즉, [7]에서 실험한 시스템들 외에, 추가로 CentOS 리눅스를 탑재한 데스크톱 컴퓨터와 안드로이드 14를 탑재한 스마트폰에서 타임스탬프 조작이 로깅 시스템에 미치는 영향을 분석하고, 타임스탬프 조작 여부를 탐지하는 기법을 제안한다. 특히, 안드로이드 기반 스마트폰의 재부팅 시점에서 시간 조작 여부를 확인할 수 있는 새로운 아티팩트(artifact)를 발견하고 이를 분석하였다. 이 새로운 아티팩트는 로그 데이터를 통해 시간 조작 시점을 보다 정확하게 파악할 수 있게 해 준다.

본 논문에서 제시한 기법은 디지털 포렌식 분석의 신뢰성을 높이고, 다양한 운영체제에서 발생할 수 있는 시간 조작 공격에 대한 대응력을 강화하는 데 기여할 수 있다. 제안 기법은 타임스탬프가 조작된 경우에도 디지털 포렌식 수사가 효과적으로 수행될 수 있게 하며, 이를 통해 디지털 증거 분석의 정확성과 신뢰성을 한층 높일 수 있을 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 설명하고, 3장에서는 모델 구성에 대해 기술한다. 4장에서는 실험 결과를 보이면서 제안 기법을 평가하고, 5장에서는 결론을 맺는다.

2. 관련 연구

Singh et al. [8]은 WSL(Windows Subsystem for Linux) 환경에서 NTFS 파일을 수정, 접근 등을 수행하는 다양한 유틸리티를 적용할 때의 타임스탬프 패턴을 분석하여 타임스탬프 위조를 감지하는 방법을 제시하였다. 파일 생성, 접근 수정, 이름 변경 및 복사에 대한 타임스탬프 규칙과 NTFS와 Ext4 간 파일 전송, 그리고 타임스탬핑 도구의 분석을 통해 타임스탬프의 변화를 체계적으로 연구하였다.

Song et al. [9]은 NTFS에서 저장장치의 성능을 활용하여 타임스탬프의 변조를 확인하는 기법을 설계하였다. 파일의 변경된 타임스탬프 시간 차이와 파일크기를 통해 초당 기록된 데이터양과 저장장치가 가진 최대 읽기, 쓰기 속도와 비교하여 타임스탬프의 변조 여부를 탐지하였다.

Gübel et al. [10]은 ext4 파일시스템에서 타임스탬프를 이용한 스테가노그래피(steganography) 채널로 데이터를 숨기는 방법을 제시하였으며 해당 기법의 비밀성과 사용성을 평가하였다. 이들은 Ext4 파일 시스템의 타임스탬프 구조와 사용법을

분석하여 데이터 은닉 가능성을 확인하였고 데이터 숨김 과정에서 암호화와 오류 수정 코드를 포함한 데이터 삽입 단계와 숨겨진 데이터를 복원하는 방법에 대해 설명하였다.

Oh et al. [11]은 기존의 타임스탬프 조작 탐지 방법의 특징과 한계를 비교 분석하고 새로운 탐지 알고리즘을 제안하였다. 제안된 탐지 알고리즘은 기존 NTFS 저널 기반 방법의 한계를 개선하여 탐지 정확도를 높였고, 실제 APT(Advanced Persistent Threat)공격에서 타임스탬프 조작을 탐지하는 사례를 보여주었다.

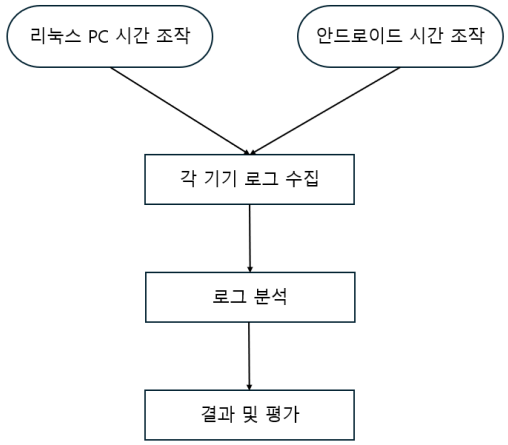
이상현. [12]은 IoS 운영체제를 기반으로 하는 디바이스 내에서 타임스탬프 위 · 변조 여부를 이벤트를 발생시켜 아티팩트 흔적이 남는 SMS, CallHistory, Photos.sqlite등을 확인하여 조작된 시스템 시간을 탐지하는 방법을 제안하였다.

기존 연구들[8, 9, 10, 11]에서는 파일의 메타 데이터인 타임스탬프의 조작을 탐지하였다. 기존 연구는 다양한 유틸리티를 통해 타임스탬프 변조 여부를 파악하거나 또는 저장장치의 성능을 분석하여 타임스탬프 조작 여부를 탐지하였다. 이러한 기존 연구는, 공격자에 의해 시스템의 시간이 조작된 이후에 생성된 파일과 파일 타임스탬프를 탐지하기 어렵다는 한계가 있다. 또한 기존 연구[7]에서는 안드로이드 9와 Ubuntu 리눅스를 대상으로 실험을 진행하였고, 본 논문에서는 [7]에서 실험한 시스템 뿐만 아니라 CentOS 리눅스와 안드로이드 14도 대상 시스템에 포함하여 실험한다.

본 논문에서는 파일의 메타 데이터 분석이 아닌 시스템의 시간 자체가 조작되었는지를 탐지하는 기법을 제안한다. 제안한 기법은 시스템의 시간 조작 이후에 생성된 파일 타임스탬프도 판별할 수 있다.

3. 타임스탬프 조작 및 로그 기반 탐지 기법

본 논문은 컴퓨팅 시스템의 타임스탬프가 조작된 경우에, 조작된 타임스탬프를 효과적으로 탐지하는 기법을 제안한다. 제안한 기법에 대한 전반적인 절차가 (그림 1)에 나타나 있다.



(그림 1) 타임스탬프 조작 및 탐지 프로세스

네트워크에 연결된 컴퓨팅 기기는 네트워크 시간을 동기화해서 타임스탬프를 관리한다. 본 논문에서는 제안 기법을 효율적으로 검증하기 위해, 네트워크 동기화를 일시적으로 해제하여 각 컴퓨팅 기기의 타임스탬프(시간)를 조작한다. 컴퓨팅 기기의 시간을 조작할 수 있는 경우의 수는 다양하며, 본 논문에서 고려하고 있는 타임스탬프 조작의 예가 <표 1>에 나타나 있다.

〈표 1〉 타임스탬프 조작 경우의 수

기기	리눅스		안드로이드	
시간 조작	현재	과거	현재	과거
		현재		현재
		미래		미래

타임스탬프 조작이 의심되는 경우, 타임스탬프 조작 여부를 판단하기 위해 시스템 로그를 수집하여 분석한다. 본 논문에서 고려하고 있는 시스템은 리눅스와 안드로이드이므로, 이들 로깅 기법을 분석한다.

리눅스의 경우 PC에서 /var/log 하위 디렉터리에 다양한 로그들이 생성된다. 그중 시스템에 직접

적인 이벤트 관련 로그가 저장된 /syslog를 수집한다. 특히 네트워크 시간 동기화와 관련 있는 NTP (Network Time Protocol) 위주로 로그를 수집한다. 안드로이드의 경우에는 ADB (Android Debug Bridge)를 통하여 로그를 수집한다. 로그 추출 명령어는 <표 2>와 같다.

〈표 2〉 안드로이드 로그 추출 명령어

명령어
logcat
bugreport

로그를 수집한 후 수집한 로그 데이터를 분석한다. 리눅스 기반 PC의 경우, /var/log/syslog 아래에 생성된 로그를 통해 특정 메시지를 담고 있는 로그와 시스템 시간이 변경되는 로그 위주로 분석한다. 안드로이드 스마트폰의 경우, 먼저, logcat을 통해 시간 조작 관련 로그 메시지를 확인한다. 기존 연구[7]에서는 logcat을 통해 시간 조작 시점을 확인 가능하였으나 휘발성 로그이기 때문에 재부팅시 확인할 수 없다는 한계점이 존재했다. 본 논문에서는 bugreport를 이용하여 추가로 분석한다. 마지막 단계에서는, 분석된 결과를 바탕으로 시스템의 타임스탬프가 조작되었는지 여부를 판단한다.

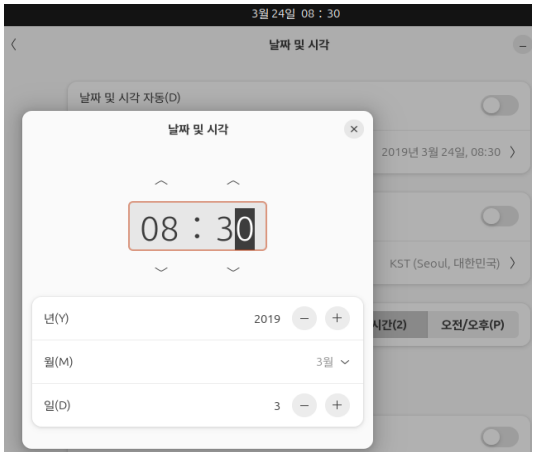
4. 실험 방법

4.1 대상 시스템

본 절에서는 데스크톱 리눅스(Ubuntu 24.04, CentOS Stream 9)와 안드로이드 스마트폰(안드로이드 버전 9, 14)을 대상으로 시간 정보를 변경한 후 로그 분석을 통해 시간 변경이 이루어진 시점을 파악한다.

4.2 PC용 리눅스

리눅스의 경우 설정 - 날짜 및 시각 자동(D) – 비활성화를 하고 (그림 2)와 같이 시간을 조작한다.



(그림 2) 네트워크 동기화 비활성화

4.2.1 Ubuntu 24.04

Ubuntu 24.04에 대해 <표 3>의 상태로 설정하여 시스템 시간을 조작한 후 /var/log/syslog 에 생성된 로그를 수집한다.

〈표 3〉 리눅스(Ubuntu) 실험 설정

Ubuntu 24.04 실험		
시간	과거	미래
실험 일시	2024.07.24, 15:18	2024.07.24, 15:29
변경 일시	2019.03.03, 08:30	2025.02.24, 08:30
실험 방법	시간 조작 후 syslog 확인	

터미널에서 date 명령어를 통해 변경된 날짜와 시간 정보를 확인할 수 있다(<표 4> 참조).

〈표 4〉 터미널에서 date 명령어로 확인

```
root@san-san2:/home/san# date
2019.03.24.(일) 08:31:15 KST
```

대부분의 로그 메시지는 syslog 파일에 저장된다. <표 5>와 같이 cat 명령어를 통해 syslog를 분석한 결과, 2024-07-24 15:18에 NTP(Network Time Protocol)가 비활성화된 메시지를 발견할 수

있었다.

〈표 5〉 /var/log/syslog 중 NTP 비활성화 내역

2024-07-24T15:18:55.372520+09:00 san-san2 systemd-timedated[3013] : **Set NTP to disabled.**

이후 생성된 로그 메시지는 "Change local time to-"라는 메시지가 생성되며, 바뀐 시간대로 로그가 저장되는 것을 확인할 수 있다. 로그 분석 결과, 네트워크 동기화를 비활성화한 시점인 NTP 비활성화 로그 메시지가 생성된 2024-07-24 15:18에 시간을 변경하였다는 것을 알 수 있다(<표 6> 참조).

〈표 6〉 syslog 중 시간 조작 내역

2019-07-23T08:30:48.002864+09:00 san-san2 systemd-resolved[565] : **Clock change detected.** Flushing caches. 2019-07-23T08.30:48.122107+09:00 san-san2 systemd-timedated[3013] : **Changed local time to** Tue 2019-07-23 08:30:48 KST

미래시간은 <표 7>의 date 명령어를 통해 변경된 날짜와 시간 정보를 확인할 수 있다.

〈표 7〉 터미널에서 date 명령어로 확인

root@san-san2:/home/san/바탕화면# **date** 2025. 02. 24. (월) 08:30:37 KST

<표 8>의 cat 명령어를 통해 syslog를 분석한 결과, 2024-07-24 15:29에 NTP가 비활성화된 메시지를 발견할 수 있었다.

〈표 8〉 /var/log/syslog 중 NTP 비활성화 내역

2024-07-24T15:29:26.589297+09:00 san-san2 systemd-timedated[3214] : **Set NTP to disabled.**

<표 9>를 통해 이후 생성된 로그 메시지는 "Change local time to-"라는 메시지가 생성되며, 바뀐 시간대로 로그가 저장되는 것을 확인할 수 있다. 로그 분석 결과, 네트워크 동기화를 비활성화한 시점인 NTP 비활성화 로그 메시지가 생성된 2024-07-24 15:29에 시간을 변경하였다는 것을 알 수 있다.

〈표 9〉 syslog 중 시간 조작 내역

2025-07-24T08:30:23.004872+09:00 san-san2 systemd-resolved[541] : **Clock change detected.** Flushing caches. 2025-07-24T08:30:23.306704+09:00 san-san2 systemd-timedated[3214] : **Changed local time to** Thu 2025-07-24 08:30:23 KST

4.2.2 CentOS stream 9

CentOS stream 9에 대해 <표 10>의 정보로 설정하여 시스템 시간을 조작한 후, /var/log/syslog에 생성된 로그를 수집한다.

〈표 10〉 리눅스(CentOS) 실험 설정

CentOS stream 9 실험		
시간	과거	미래
실험 일시	2024.06.24, 16:57	2024.06.24, 16:36
변경 일시	2019.03.03, 08:33	2025.02.01, 05:33
실험 방법	시간 조작 후 syslog 확인	

과거 시간은 <표 11>의 date 명령어를 통해 변경된 날짜와 시간 정보를 확인할 수 있다.

〈표 11〉 터미널에서 date 명령어로 확인

[leesan@localhost ~]$ **date** 2019.03.03.(일) 08:33:24 KST

<표 12>를 통해 6월 24일 16:57에 NTP가 비활성화되었다는 로그 메시지가 생성되었고, 이후

"Changed local time to-"라는 로그 메시지가 순차적으로 생성되는 것을 확인할 수 있다. 로그 분석 결과, 네트워크 동기화를 비활성화한 시점인 NTP 비활성화 로그 메시지가 생성된 2024-06-24 16:57에 시간을 변경하였다는 것을 알 수 있다.

〈표 12〉 CentOS의 /var/log/syslog 중 일부

6월 24일 16:57:11 localhost.localdomain systemd-timedated[8082] : chronyd.service :Disabling unit. 6월 24일 16:57:11 localhost.localdomain systemd-timedated[8082] : **Set NTP to disabled.** 6월 03 16:57:00 localhost.localdomain systemd-timedated[8082] : **Changed local time to** Sun 2018-06-03 16:57:00 KST 6월 03 16:57:00 localhost.localdomain systemd-timedated[8082] : **Changed local time to** Sun 201906-03 16:57:00 KST

<표 13>에 나타난 바와 같이 date 명령어를 통해 변경된 날짜와 시간 정보를 확인할 수 있다.

〈표 13〉 터미널에서 date 명령어로 확인

[root@localhost leesan]# **date** 2025.02.01.(토) 05:33:32 KST

<표 14>를 통해 6월 24일 16:36에 NTP가 비활성화 되었다는 로그 메시지가 생성되었고, 이후 "Changed local time to-"라는 로그 메시지가 순차적으로 생성되는 것을 확인할 수 있다. 로그 분석 결과, 네트워크 동기화를 비활성화한 시점인 NTP 비활성화 로그 메시지가 생성된 2024-06-24 16:36에 시간을 변경하였다는 것을 알 수 있다.

리눅스 PC의 경우 재부팅 시에도 syslog가 저장되어 있어 로그 분석을 통해 변경 시점을 확인할 수 있다(<표 15> 참조).

〈표 14〉 CentOS의 /var/log/syslog 중 일부

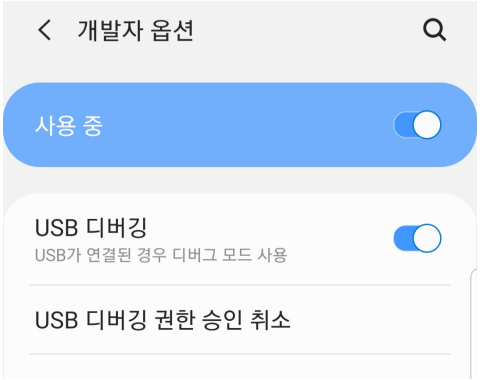
6월 24 16:35:00 localhost.localdomain systemd[1] : systemd-timedated.service : Deactivated successfully. 6월 24 16:36:11 localhost.localdomain systemd[1] : systemd-timedated.service : Deactivated successfully. 6월 24 16:36:15 localhost.localdomain systemd-timedate[6577] : chronyd.service : Disabling unit. 6월 24 16:36:15 localhost.localdomain systemd-timedate[6577] : **Set NTP to disabled.** 6월 24 16:36:00 localhost.localdomain systemd-timedate[6577] : **Changed local time to** Sat 2023-06-24 16:36:00 KST 6월 24 16:36:00 localhost.localdomain systemd-timedate[6577] : **Changed local time to** Sat 2022-06-24 16:36:00 KST

〈표 15〉 syslog 파일들

syslog syslog.1 syslog.2.gz syslog.3.gz syslog.4.gz

4.3 안드로이드

안드로이드의 경우 (그림 3)과 같이 설정 - 개발자 옵션 - USB 디버깅을 활성화시킨 뒤 리눅스와 같은 방법으로 시간을 조작한다.



(그림 3) USB 디버깅 활성화

시간 조작 여부를 파악하기 위해, 안드로이드에서는 'ADB(Android Debug Bridge)' 를 이용하여 로그를 추출한다. 로그를 추출하는 명령어는 <표 2>에 나타나 있다.

먼저, logcat의 경우에는 부팅 시점 이후로 log들이 누적되는 특성이 있다. 따라서 만약 범죄자가 시간을 조작한 후 현재시간으로 되돌려 재부팅을 한다면 logcat에서는 시간 조작에 대한 증거를 찾을 수 없다. 그러나 bugreport의 경우에는 logcat을 포함하여 상위 로그 파일들까지 추출이 가능하다. 따라서 기존 휘발성 로그를 누적하지 못하는 logcat의 단점을 bugreport를 통하여 극복 가능하다. 안드로이드의 경우의 실험에 대한 정보는 <표 16>과 같다.

〈표 16〉 안드로이드에서 시간 조작 실험 정보

안드로이드 실험 설정		
대상 기기	Galaxy S8, 안드로이드 9	Galaxy S21, 안드로이드 14
실험 일시	2024.07.21. 14:41	2024.07.21. 15:08
변경 일시	2024.07.14	2024.07.28
실험 방법	시간 조작 후 현재 시간으로 되돌려 재부팅 후 로그 확인	

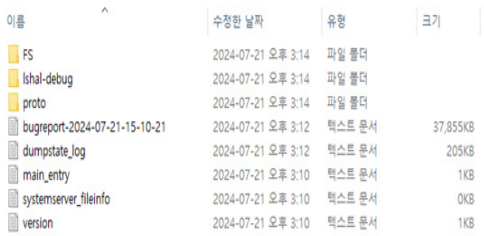
4.3.1 안드로이드 9

안드로이드 9에 대한 logcat 결과가 <표 17>에 나타나 있다. <표 17>을 분석해 보면 재부팅 시점 이후로 로그 데이터가 누적되는 걸 확인할 수 있다. 따라서 범죄자가 시간을 조작하고 현재시간으로 되돌린 후 재부팅을 하면 이전 로그들에 대해 확인이 불가능하다.

(그림 4)는 안드로이드 9에 대한 bugreport 결과이다. 안드로이드 9의 경우 bugreport 명령어 사용 시 'bugreport-2024-07-21-15-10-21.txt' 로 시스템 로그가 생성된다. 해당 파일을 분석한 결과는 (그림 4)와 같다.

〈표 17〉 안드로이드 9의 logcat 화면

```
PS C:\Users\cgumg> adb logcat
---------beginning of main
07-21 15:08:42.512 3212 3212 I SeLinux : SELinux : Loaded service_contexts from :
07-21 15:08:42.513 3212 3212 I SeLinux : /vndservice_contexts
07-21 15:08:42.602 3211 3211 I hwservicemanager : hwservicemanager is ready now.
07-21 15:08:42.605 3210 3210 I SeLinux : SeLinux : Loaded service_contexts from :
07-21 15:08:42.606 3210 3210 I SeLinux : /plat_service_contexts
07-21 15:08:42.606 3210 3210 I SeLinux : /vendor_service_contexts
07-21 15:08:42.619 3217 3217 D scs : bODMorProduct : false, bOmcSupport : true
07-21 15:08:42.619 3217 3217 D scs : OMC_B28_FILE : /system/omc/omc_b2b_list
07-21 15:08:42.619 3217 3217 D scs : getValue FromFile : Luc
```



(그림 4) bugreport 로그 파일

<표 18>에서 밑줄이 그어진 이탤릭체로 표시된 부분을 통해 시간 조작이 이루어졌고, 재부팅이 이루어졌다는 걸 알 수 있다.

〈표 18〉 안드로이드 bugerport 파일 분석 결과

+9d22h00m43s027ms(14) TIME : 2024-07-27-15-08-19
+9d22h00m43s160ms(2) c4190022 +fg=u0a53: "com.lguplus.appstore"
+9d22h00m43s235ms(2)023 c4190022 + tmpwhitelist =u0a187:"fg-service-launch"
+9d22h00m43s387ms(2)023 c4190022 +fg=u0a187:"com.samsung.android.net.wifi.wifiguider"
+9d22h00m45s252ms(2)023 c4190022 -fg=u0a53: "com.lguplus.appstore"
+9d22h00m46s762ms(3023 c4190022 current=281 ao temp=33 pst_temp=32 bat_temp=28 chg_temp =31 pa_temp=29
+9d22h00m48s460ms(3)023 c4190022
+9d22h00m48s469ms(14) *TIME : 2024-07-28-06-00-00*
+9d22h00m51s004ms(14) *TIME : 2024-07-21-15-08-26*
+9d22h00m58s571ms(12) SHUTDOWN
+9d22h00m58s582ms(12) START

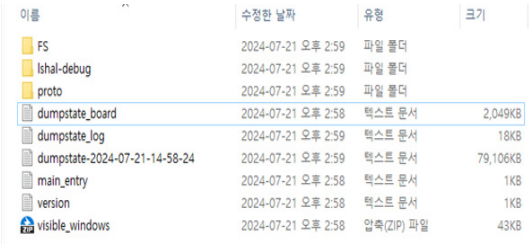
4.3.2 안드로이드 14

<표 19>는 안드로이드 14에 대한 logcat 결과이다. <표 19>를 분석한 결과, 안드로이드 9와 같이 재부팅 시점 이후로 로그 데이터가 누적되는 걸 확인할 수 있었다.

〈표 19〉 안드로이드 14의 logcat 화면

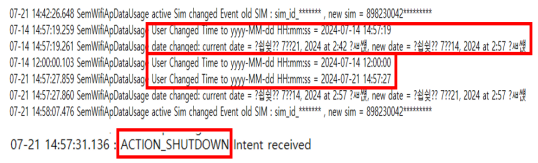
PS C:\Users\cgumg〉 **adb logcat**
---------beginning of main
07-27 14:41:37.880 626 626 W linker : Warining : failed to find generated linker configuration from "/linkerconfig/ ld.config.txt"
07-27 14:41:37.881 625 625 W linker :Warining : failed to find generated linker configuration from "/linkerconfig/ ld.config.txt"
07-27 14:41:37.884 624 624 D lmkd : fetech chimera enabled flag : [0] from property
07-27 14:41:37.884 624 624 I lmkd : Using psi monitors for memory pressure detection
07-27 14:41:37.884 624 624 I lmkd : lmkd enable_killbooster_all
07-27 14:41:37.884 624 624 I lmkd : set custom_sw_limit : 500
07-27 14:41:37.884 624 624 I lmkd : set upgrade_pressure : 80
07-27 14:41:37.884 624 624 I lmkd : set lmkd_freelimit_val : 13
07-27 14:41:37.884 624 624 I lmkd : enable_upgrade_ criadj : 0
07-27 14:41:37.884 624 624 I lmkd : Process polling is supported
07-27 14:41:37.885 624 624 I lmkd : Process reaping is not supported
07-27 14:41:37.886 624 629 W libprocessgroup : Controlier io is not found
07-27 14:41:37.887 624 629 W libprocessgroup : Controlier io is not found
07-27 14:41:37.887 624 629 W libprocessgroup : Controlier io is not found

(그림 5)는 안드로이드 14에 대한 bugreport 결과를 보여준다.



(그림 5) 안드로이드 14의 bugreport 결과

(그림 5)를 보면, 안드로이드 14는 안드로이드 9와는 달리 'dumpstate-2024-07-21-14-58-24.txt' 라는 로그 파일이 생성되는 것을 확인할 수 있다. 해당 파일을 분석한 결과를 (그림 6)에 제시하였다.



(그림 6) 안드로이드 bugreport 파일 분석 결과

(그림 6)을 분석하면 박스로 표시된 부분을 통해 먼저 시간 조작이 이루어지고 현재 시간으로 되돌린 후, 재부팅을 수행한 것을 알 수 있다.

따라서, 안드로이드 9와 안드로이드 14 두 버전 모두 logcat을 통하여 시간 조작 및 재부팅에 대한 로그 메시지를 확인할 수 없었지만, bugreport를 통하여 확인할 수 있었다. 이에 대한 결과를 정리하면 <표 20>와 같다.

〈표 20〉 안드로이드 실험 결과 정리

	안드로이드 9	안드로이드 14
logcat을 통한 시간 조작 증거 확인	X	X
bugreport를 통한 시간 조작 증거 확인	O	O

5. 논의

본 논문에서는 리눅스 기반 PC와 안드로이드 스마트폰에서 시간 정보를 조작한 후, 로그 분석을 통해 그 시점을 파악하는 방법을 제시하였다. 특히, 안티 포렌식 기법으로 타임스탬프를 조작하여 디지털 포렌식 분석을 방해하려고 할 때, 타임스탬프 조작 여부를 탐지하는 연구에 초점을 맞추었다.

리눅스 기반 OS(Ubuntu 24.04, CentOS)의 경우, 시간 조작 후 syslog를 통해 변경된 시점을 정확히 파악할 수 있었다. 실험 결과, NTP(Network Time Protocol)가 비활성화 되었다는 로그 메시지가 생성된 이후에 "Change local time to-"라는 메시지가 순차적으로 기록되는 것을 확인하였다. 이는 네트워크 동기화가 비활성화된 상태에서 사용자가 수동으로 날짜와 시간을 조작할 때 NTP 프로토콜이 비활성화 된다는 것을 의미한다. 본 논문은 선행연구[7]에 CentOS 리눅스와 안드로이드 14를 추가하여 실험하고 결과를 분석하였다. 또한, 시간 변경 관련 메시지 기반의 분석을 포함하여, NTP 프로토콜과의 연관성을 분석하여 타임스탬프 조작 시점을 명확히 파악할 수 있었다.

〈표 21〉 관련 연구와 본 논문과의 비교 분석

	[11]	[12]	[7]	본 논문
운영체제	Windows 10	iOS (7, 8)	안드로이드 9, Ubuntu 22.04	안드로이드 (9, 11), Ubuntu 24.04, CentOS stream 9
파일 시스템	NTFS (New Technology File System)	APFS (Apple File System)	Ext4	Ext4
탐지하려는 분석 행위	파일의 타임스탬프를 조작하는 행위	파일의 타임스탬프를 조작하는 행위	시스템의 시간을 조작하는 행위	시스템의 시간을 조작하는 행위
분석 파일	*$MFT, $LogFile, $UsnJrnl, $Prefetch, Registry, LNK Files, Event Log, Volume Shadow Copy*	*sms.db, Callhistory.storedata, Photos.sqlite, CurrentPowerLog.PLSQL, CurrentPowerLog.powerlog, general.log*	**안드로이드 9**	**안드로이드 9**
			logcat으로 추출한 파일	*bugreport-YYYY-MM-DD-HH-MM-SS.txt*
			Ubuntu 22.04	**안드로이드 11**
			/var/log/syslog	*dumpstate-YYYY-MM-DD-HH-MM-SS.txt*
				Ubuntu 24.04, CenOS stream 9
				/var/log/syslog

<표 21>은 본 논문과 기존 연구들과의 차이점을 보여준다. Oh et al.[11]은 Windows 10을 탑재한 데스크톱 컴퓨터에서 파일의 타임스탬프를 조작하는 행위를 탐지하였다. $MFT, $LogFile, $UsnJrnl, Prefetch, Registry, LNK 파일, 이벤트 로그, 볼륨 새도우 복사본과 같은 다양한 아티팩트들을 수집하고 분석하였다. 이상현 등[12]은 iOS를 탑재한 아이폰과 아이패드에서 파일의 타임스탬프를 조작하는 행위에 관한 흔적을 조사하였다. 이를 위해, 그들은 sms.db, Callhistory.storedata, Photos.sqlite, CurrentPowerLog.PLSQL,CurrentPowerLog.powerlog,general.log 등을 분석하였다.

본 연구진의 선행연구[7]는 Ubuntu 22.04와 안드로이드 9에서 시스템의 시간을 조작하는 행위가 로깅시스템에 미치는 영향을 분석하고, 시스템 시간 조작 여부를 탐지하기 위해 로그를 분석했다. 안드로이드 9에서는 logcat 명령어로 추출한 로그를 분석하였다. 본 논문은 [7]을 확장한 것으로, 추가적인 대상 OS로 안드로이드 14와 CentOS를 포함하여 실험하였고, 시간 동기화와 관련이 있는 NTP 프로토콜에 관한 로그와 비휘발성 로그인 bugreport를 통해 새로운 아티팩트를 수집 및 분석하였다.

안드로이드 기반 OS에서는 logcat 명령어를 사용하여 로그를 분석할 수 있다. 그러나 logcat을 이용해서 휘발성 로그만을 수집할 수 있어, 기기를 재부팅한 후에는 시간이 조작된 시점을 확인하기 어려웠다. 본 논문에서는 bugreport 명령어를 통해 비휘발성 파일을 수집하여 분석함으로써 시간이 조작된 시점을 효과적으로 파악할 수 있었다.

6. 결과

본 논문에서는 리눅스 기반 PC와 안드로이드 스마트폰에서 시간 조작을 통한 안티 포렌식 기법을 분석하고, 이러한 조작 행위를 탐지할 수 있는 방법을 제시하고 실험하였다. 논문에서 파악한 주요 결과는 다음과 같다. 첫째, 리눅스 시스템에서 NTP 비활성화 로그 메시지와 "Change local time to-" 메시지를 통해 시간 조작의 시점을 파악할 수 있다. 둘째, 안드로이드 시스템에서는 logcat과 bugreport를 활용하여 시간 조작의 시점을 파악할 수 있다. 셋째, 재부팅 후에도 syslog와 bugreport를 통해 시간 조작의 시점을 파악할 수 있다.

이러한 결과물들은 타임스탬프 조작을 통한 안티 포렌식 공격에 대응하기 위한 중요한 자료를 제공한다. 예로, 로그 파일을 통해 시간 조작의 시점을 파악하여 특정 사건과 관련된 아티팩트들에 대해 신뢰할 수 있는 타임라인을 구성할 수 있다. 향후에는 더 다양한 OS와 더 많은 안티 포렌식 기법을 고려하여 포괄적인 포렌식 분석을 수행할 필요가 있다.

Acknowledgements

이 연구는 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(no. 2021R1A2C2012574). 이 연구는 2022년도 정부(과학기술 정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No.1711170476, 이벤트 기반 실험시 스템구축을 통한 자동차 내·외부 아티팩트 수집 및 통합 분석 기술 개발). 이 연구는 2024년도 산업통상자원부 및 한국산업기술진흥원(KIAT) 연구비 지원에 의한 연구임('P0023522')

참고문헌

[1] Michael G. Nobleet, Mark M. Pollitt, and Lawrence A, Presley, "Recovering and Examining Computer Forensic Evidence", Forensic Science Communication Vol.2, No. 4, Oct. 2000.

[2] Carrie Morgan Whitcomb, "An Historical Perspective of Digital Evidence : A Forensic Scientist's View", International Journal of Digital Evidence, Vol. 1, No. 1, pp. 7-15, Spring 2002.

[3] Simon L. Garfinkel, "Anti-forensics: Techniques,

detection and countermeasures", INTERNATIONAL CONFERENCE ON INFORMATION WAREFARE AND SECURITY, Volume 2, No. 1, pp 77-84, 2007.

〔4〕 R. González Arias, J. Bermejo Higuera, J. J. Rainer Granados, J. R. Bermejo Higuera, and J. A. Sicilia Montalvo,"Systematic Review: Anti-Forensic Computer Techniques", Applied Sciences, Vol. 14, No. 12, 2024.

〔5〕 Byeongyeong Yoo, "Analysis of File Time Change by File Manipulation of Linux System", The Journal of the Institute of Internet, Broadcasting and Communication, Vol. 16, No. 3, pp. 21-28, 2016.

〔6〕 Murtaza Ahmed. & M.N.A Khan "A Review Of Forensic Analysis Techniques For Android Phones", Journal of Independent Studies and Research - Computing, Vol. 15, NO. 1, 23-30, 2017.

〔7〕 S. Lee, J. Jung, S. Ahn and S. Cho, "Analysis of the Impact of Time Information Manipulation on Logging in Linux and Android", Spring Conference of Korean Institute of Next Generation Computing, April 2024.

〔8〕 Bhupendra Singh and Gaurav Gupta, "Analyzing windows subsystem for linux metadata to detect timestamp forgery", Advances in Digital Forensics XV: 15th IFIP WG 11.9 International Conference, Orlando, FL, USA, January 28 - 29, 2019, Revised Selected Papers 15. Springer International Publishing, 2019.

〔9〕 Jong-Hwa Song and Hyun-Seob Lee, "A Design of Timestamp Manipulation Detection Method using Storage Performance in NTFS", Journal of Internet of Things and Convergence, Vol. 9, No. 6, pp.23-28, 2023.

〔10〕 Thomas Göbel and Harald Baier. "Anti-forensics in ext4: On secrecy and usability of timestamp-based data hiding", Digital Investigation, Vol. 24, pp. S111-S120, 2018.

〔11〕 J. Oh, S. Lee and H. Hwang, "Forensic Detection of Timestamp Manipulation for Digital Forensic Investigation", IEEE Access, vol. 12, pp. 72544-72565, 2024.

〔12〕 Sanghyun Lee, Yunho Lee, and Sangjin Lee. "A Study on the Evidence Investigation of Forged/Modulated Time-Stamp at iOS (iPhone, iPad)", KIPS Tr. Comp. and Comm. Sys, Vol.5, No.7, pp.173-180, 2016.

저자소개

◆ 이산

- 2019년 3월~현재 단국대학교 산업보안학과 학사과정
- 관심분야: 디지털 포렌식, 안티 포렌식, 정보보안

◆ 조민혁

- 2020년 3월~현재 단국대학교 모바일시스템공학과 학사과정
- 관심분야: 디지털 포렌식, 안드로이드, 시스템 보안

◆ 정지헌

- 2018년 3월~2024년 2월 단국대학교 소프트웨어학과 학사과정
- 2024년 2월~현재 단국대학교 컴퓨터학과 석사과정
- 관심분야: 디지털 포렌식, 정보보안, 차량 내부 네트워크 보안

◆ 조성제

- 1989년 2월 서울대학교 컴퓨터공학과 공학사
- 1991년 2월 서울대학교 컴퓨터공학과 공학석사
- 1996년 8월 서울대학교 컴퓨터공학과 공학박사
- 1997년 3월~현재 단국대학교 컴퓨터학과/소프트웨어학과 교수
- 관심분야: 디지털 포렌식, 시스템 보안 및 악성코드 분석, 인공지능 보안, 시스템 소프트웨어 등