JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 24, No. 2, pp. 335-342, Feb. 2023

Check for updates

IoT 플랫폼에 탑재되는 안드로이드 및 리눅스 기반 파일시스템 포렌식

이 진 오[1] · 손 태 식[2*]
[1]아주대학교 AI융합네트워크학과 석사과정
[2*]아주대학교 사이버보안학과 교수

Forensics for Android and Linux-based file system on IoT platform

Jino Lee[1] · Tae-Shik Shon[2*]

[1]Master's Course, Department of AI Convergence Network, Ajou University, Suwon, Gyeonggi-do, Korea
[2*]Professor, Department of Cyber Security, Ajou University, Suwon, Gyeonggi-do, Korea

[요 약]

기술의 발전으로 IoT 기기들의 보급률이 증가하고 있으며, 일상생활의 많은 부분에 IoT가 사용되고 있다. 이에 따라 IoT 기기 내부에 저장된 다양한 데이터는 범죄 발생 시 증거로 사용될 수 있다. 그러나 IoT 기기는 기능 및 종류가 매우 다양하므로 데이터 획득을 위한 지속적인 연구가 필요하다. 따라서 본 논문에서는 리눅스 기반으로 제작된 파일시스템인 ext4와 JFFS2 및 UBIFS를 선정하여 파일시스템 단계에서의 분석을 진행하였다. 각 파일시스템의 메타데이터를 분석하여 저장구조를 파악하고, 각 구조에 따른 파일의 복구 가능성을 분석하였다. 본 연구를 통해 리눅스 기반의 환경에서 사용되는 플랫폼의 특성과 각 플랫폼에 저장된 데이터의 특성을 설명하고, 데이터의 저장 구조를 파악하여 기기 및 플랫폼의 보안성 향상에 기여한다. 또한 고안한 파일시스템 포렌식 분석 방법을 통해 다른 기기에도 적용할 수 있도록 한다.

[Abstract]

With the development of technology, the penetration rate of IoT devices is increasing, and IoT is used in many parts of daily life. Accordingly, various data stored inside the IoT device may be used as evidence in the event of a crime. However, IoT devices have a wide variety of functions and types, so continuous research is needed to acquire data. Therefore, in this paper, ext4, JFFS2, and UBIFS, which are filesystems produced based on Linux, were selected and analyzed at the filesystem stage. By analyzing the metadata of each filesystem, the storage structure was identified, and the possibility of file recovery according to each structure was analyzed. Through this study, the characteristics of the platform used in the Linux-based environment and the characteristics of the data stored on each platform are explained, and the storage structure of the data is identified to contribute to improving the security of devices and platforms. It also allows it to be applied to other devices through the devised filesystem forensic analysis method.

색인어 : 디지털 포렌식, 파일시스템, 사물인터넷, 리눅스, 메타데이터

Keyword : Digital Forensics, Filesystem, Internet of Things, Linux, Metadata

http://dx.doi.org/10.9728/dcs.2023.24.2.335

Received 04 November 2022; **Revised** 01 December 2022
Accepted 05 December 2022

***Corresponding Author; Tae-Shik Shon**

Tel: +82-31-219-3321
E-mail: tsshon@ajou.ac.kr

Ⅰ. 서 론

현재 IoT(Internet of Things) 기술은 우리의 삶에 점점 많은 부분을 차지하고 있다. 기술이 발전하고 시간이 지날수록 IoT 기기의 사용량이 증가하고 있으며, 대다수의 기기는 가정 및 우리의 일상에 보급되어 사용되고 있다. IoT 기기들은 우리의 일상과 함께하면서 편의성을 제공하고 있다. 그러나 보안적 측면에서 보면 편의성이 발전할수록, 보안성을 같이 발전시키기에는 많은 어려움이 따른다. 현재는 그러한 보안성이 갖춰지지 않고 편의성만 발달한 IoT 기기들이 많이 존재한다. IoT 기기에는 사용자의 계정 및 신상 정보부터, 습관, 사용 시간 기록 등 다양한 민감한 정보들이 존재한다. 따라서 현재의 미흡한 보안성으로 인해 해킹과 같은 보안사고 발생 시, 개인정보가 유출될 위험뿐만 아니라, 사용 기록을 통한 주거 침입 등의 2차 피해가 발생할 우려가 있다[1]. 이에 따라 IoT 기기의 보안 수준을 분석하고, 보다 강화해야 할 필요가 있다.

IoT 기기 내부의 데이터들은 범죄 발생 시, 증거 및 알리바이로 사용될 수 있다. 실제로 2018년 미국 플로리다 주에서 살인 혐의로 기소된 한 남성의 Amazon echo에 녹음된 음성을 통해, 알리바이를 증명한 사건이 존재한다[2]. 이처럼 보안성 분석 및 증거를 위한 아티팩트 획득을 위해 현재 사용되고 있는 IoT 기기의 분석이 필요하다. 그러나 기기 자체를 분석할 경우, 삭제되거나 의도적으로 숨긴 데이터를 획득하기 어려울 수 있다[3]. 이에 따라 파일시스템 단계에서 분석을 진행하여 삭제된 파일까지 획득할 수 있도록 연구가 필요하다.

대다수의 IoT 기기들은 리눅스를 기반으로 IoT 기능에 맞춰 개발한 파일시스템을 사용하지만, IoT 기기는 소형 제품부터 차량까지 크기 및 기능이 다르기 때문에 유사한 리눅스 기반의 파일시스템일 경우에도 파일 저장 방식, 저장되는 데이터 종류 등이 상이할 수 있다. 또한 IoT 기기의 발전에 따라 분석되지 않은 플랫폼을 사용하는 제품들이 계속해서 출시되고 있다. 따라서 본 논문에서는 분석되지 않았거나, 미래에 사용될 수 있는 IoT 기기들을 파일시스템 단계에서의 분석을 진행하였다. 해당 연구는 산업용 전력 DCU에 탑재되는 리눅스 기반의 파일시스템인 JFFS2 및 UBIFS와 차량에 탑재되는 Android Automotive의 ext4 파일시스템을 대상으로 진행하였다. 해당 연구를 통해 IoT 기기들의 파일시스템 단계에서 저장되는 메타데이터를 분석하였으며, 데이터 추출 및 삭제된 데이터 복구가 가능함을 확인하였다.

본 논문은 2장에서는 관련 연구에 관해서 설명하고, 3장에서는 해당 플랫폼에 대해 설명한다. 4장에서는 메타데이터를 분석하고 데이터 추출방안을 설명하고, 5장에서는 삭제된 데이터 복구 방안에 대해서 설명한다. 마지막으로 6장에서는 결론에 대해 서술하였다.

Ⅱ. 관련 연구

IoT 기기를 분석하는 연구는 지속적으로 수행되어 왔다. 그러나 IoT 기기의 특성 상, 데이터 획득 방법, 탑재되는 파일시스템 등 그 수가 매우 다양하기 때문에 기기가 출시됨에 따라 꾸준한 연구를 필요로 한다. 본 절에서는 IoT 기기를 대상으로 한 포렌식 연구 및 IoT 파일시스템 분석 관련 연구를 소개한다.

2019년 Nemayire 등은 삼성 스마트 TV를 대상으로 포렌식 연구를 진행했다[4]. 해당 연구는 삼성 스마트 TV의 데이터를 획득하여 Magnet Axiom과 MD-Red 프로그램을 통해 아티팩트 획득을 진행하였다. 해당 연구를 통해 TV에 저장될 수 있는 다양한 아티팩트를 알 수 있었다. 그러나 아티팩트의 조사 단계에서 연구가 마무리되었기 때문에, 삭제된 파일 및 파일시스템 분석은 진행되지 않았다.

2019년, 조우연 등은 AI 스피커를 분석 및 내부 데이터 획득과 관련된 연구를 진행했다[5]. 해당 연구는 4 종류의 AI 스피커를 대상으로 데이터 획득 방안을 고안하였다. 또한 AI 스피커에서 획득할 수 있는 아티팩트들을 조사하고, 파일시스템을 분석하여 파일 추출 및 삭제된 파일과 관련된 연구를 진행하였다. Shancang Li 등은 Amazon echo를 대상으로 포렌식 연구를 진행했다[6]. 해당 연구는 기기 자체, 연동된 앱, 펌웨어 분석 등 다양한 방법을 통해 아티팩트 획득을 시도하였다. 또한 수사 중심의 연구를 진행하여, 실제 범죄 발생 시 적용할 수 있는 포렌식 모델, 실제 상황에서의 데이터 획득 방법 등을 조사하였다.

2021년 MattJarrett은 Google에서 개발 중이며, Google Nest Hub에 시범 탑재된 Fuchsia OS를 대상으로 분석 연구를 진행했다[7]. 해당 연구는 가상머신을 통해 오픈소스인 fuchsia OS를 설치하여, 메타 데이터의 분석 연구를 진행하였다. 해당 연구는 추후 탑재될 수 있는 플랫폼을 미리 분석함으로써, 탑재 기기의 데이터 획득 방안 연구에 기여하였다.

2022년 김형찬 등은 안드로이드 9, 10을 사용하는 스마트폰을 대상으로 개인정보와 관련된 포렌식 연구를 진행했다[8]. 해당 연구에서는 스마트폰에 저장될 수 있는 개인정보를 시나리오를 통해 획득을 시도하였으며, 파일시스템 단계의 분석을 통해 삭제된 개인정보를 복구함으로써 보안의 필요성을 상기시켰다.

해당 연구들은 해당 IoT 기기를 대상으로 연구를 진행하여 다양한 포렌식 방안을 제안하고, IoT 기기에 탑재되는 파일시스템을 분석하였다. 그러나 같은 종류의 IoT 기기들도 계속해서 출시되고 있으며, 새로운 종류의 기기에 IoT 기술이 결합되고 있기 때문에, 꾸준한 연구를 필요로 하고 있다. 이에 따라 연구가 진행되지 않았거나, 미래에 사용될 수 있는 IoT를 대상으로 연구를 진행하였다.

Ⅲ. IoT 플랫폼

본 장에서는 대상 IoT 기기에 탑재되는 플랫폼 및 파일시스템에 대해서 설명한다. 현재 IoT 기기의 수가 다양한 만큼 플랫폼 및 파일시스템의 종류 또한 다양하다. 그러나 대부분의 파일시스템은 리눅스를 기반으로 제작되었으며, 본 연구에 사용된 AAOS(Android Automotive OS)는 리눅스 기반 플랫폼이며, DCU 또한 리눅스 플랫폼을 사용한다. 이처럼 리눅스 기반 플랫폼이 다수 사용됨에 따라 포렌식 진행을 위해 다양한 플랫폼 분석의 필요성을 확인했다. 이에 따라 연구 진행을 위해 플랫폼 별 사용되는 파일시스템의 특징을 확인한다.

3-1 Android Automotive (ext4)

Android Automotive는 기존의 Google Android를 개량하여 차량의 대시보드에 탑재되도록 개발한 차량용 OS이다. AAOS는 기존의 Android auto나 Apple CarPlay와 달리, 모바일 기기를 통해 차량과 연결하는 것이 아닌 차량 자체에 탑재되는 OS이다. AAOS는 Android OS를 변형하여 개발하였기 때문에, 기존에 사용하던 ext4 파일시스템을 동일하게 사용한다. Ext4는 리눅스 기반의 파일시스템으로 스마트폰이나 다양한 IoT 기기에 탑재된다. 이에 따라 ext4 파일시스템은 현재까지 다양한 연구가 진행되어 왔으나, 탑재되는 기기에 따라 일부 차이점이 존재했다[9]. AAOS 또한 ext4가 탑재되었으나, 차량 전용으로 개발되었기 때문에, 기존의 파일시스템과의 차이점이 존재할 수 있다. 또한 AAOS는 현재 일부 차량에 탑재되기 시작하고 있기 때문에, 추후 다양한 차량에 탑재되었을 시를 위한 분석 연구가 필요한 상황이다. 따라서 AAOS 플랫폼에 사용된 ext4 파일시스템이 사용됨을 확인함으로써, 메타 데이터 분석 및 파일 추출에 대한 연구의 필요성을 확인했다.

3-2 Linux (JFFS2/UBIFS)

Linux는 1991년 개발된 오픈소스 플랫폼으로 다양한 기기 및 PC에 탑재된다. Linux 플랫폼에 사용되는 linux OS에는 다양한 파일시스템이 사용될 수 있으나, 해당 실험에서 사용하는 파일시스템은 JFFS2 와 UBIFS이다. JFFS2 와 UBIFS는 플래시 메모리 장치에 쓰이는 linux 기반 파일시스템이다. JFFS2와 UBIFS는 로그 형태로 데이터를 저장하는 저널링 형태의 파일시스템이다. UBIFS는 JFFS2를 개선하여 개발된 파일시스템으로, JFFS2와 달리 전체 미디어를 스캔하지 않아 큰 용량의 Nand flash에 적합하다. JFFS2는 MTD(Memory Technology Device) 장치 위에서 작동하며, UBIFS는 일반 MTD에서는 바로 작동할 수 없으며 UBI 볼륨 위에서만 작동한다. UBI 볼륨은 MTD 장치에 접근할 수 있기 때문에 UBIFS는 UBI 볼륨을 사용하여 MTD에 접근한다. 해당 파일시스템들은 임베디드 시스템에서 주로 사용되며, 개발 후 많은 기간이 지났으나 여전히 일부 IoT 모델에서는 사용된다. 이처럼 IoT 기기의 분석을 위한, 해당 파일시스템에 대한 메타 데이터 분석 및 파일 추출에 대한 연구의 필요성을 확인했다.

Ⅳ. 파일시스템 메타데이터 분석 및 데이터 추출

4-1 Android Automotive (ext4)

Android Automotive는 차량 전용 플랫폼으로 일반적인 Android와 커널을 제외하고는 동일한 ext4 파일시스템을 사용하고 있다. 해당 플랫폼의 분석을 위해 오픈소스로 공개된 Android Automotive 코드를 사용해 빌드를 진행하였으며, 라즈베리파이 4B를 사용하여 OS를 실행하였다. SD 카드를 이용해 빌드를 진행하였기 때문에, SD카드를 PC에 연결하여 리눅스의 dd if 명령어를 통해 파일시스템 데이터를 획득할 수 있었다.

Android Automotive의 분석을 위해 데이터 파티션을 추출하여 파일시스템 분석을 진행하였다. 데이터 파티션은 리눅스의 dd 명령어를 통해 덤프를 진행하였으며, 해당 이미지를 분석하였다. 분석 결과 그림 1와 같이 superblock을 확인할 수 있었으며, magic number인 0xEF53 값을 통해 ext4 파일시스템을 사용하는 것을 알 수 있었다. Ext4에서의 아티팩트 획득을 위해, 메타데이터를 분석하여 파일 데이터가 저장된 위치를 확인하고, 다른 기기에 사용된 ext4파일시스템과의 차이점이 존재하는지 확인한다. 그림 1의 superblock을 통해 해당 파티션의 블록 크기, 그룹 당 inode 개수 등 메타데이터 분석에 필요한 정보를 확인할 수 있었다.

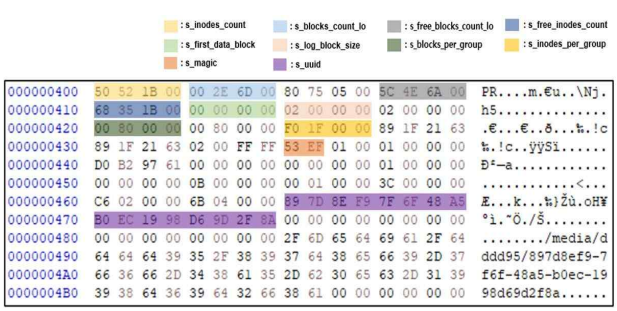

그림 1. Ex4의 슈퍼블록 메타데이터
Fig. 1. Ext4 superblock metadata

Superblock의 분석 후, inode table의 주소를 저장하고 있는 group descriptor의 분석을 진행하였다. Group descriptor는 그림 2와 같이 0x1000 주소에 위치하며, block bitmap, inode bitmap, inode table의 주소를 저장하고 있다. 해당 정보들 중 inode table은 데이터를 저장하는 inode들이 저장된 블록으로, 해당 주소로 이동하여 기기에 저장된 데이터를 확인할 수 있다.

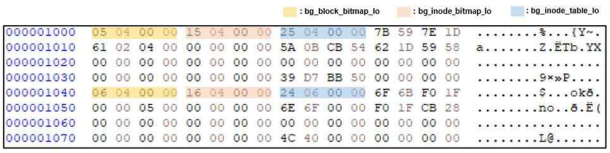

그림 2. Ext4의 그룹 디스크립터 메타데이터
Fig. 2. Ext4 group descriptor metadata

Inode table로 이동 시, inode들이 존재함을 확인할 수 있다. Ext4 파일시스템은 1번~10번의 inode가 이미 예약되어 있으며, 2번 inode는 root 디렉토리 정보를, 8번 inode는 저널영역의 정보가 저장되어 있다. 그림 3은 2번 inode로 root 디렉토리에 대한 메타데이터를 저장하고 있다. Inode에는 데이터 정보, timestamp 등 기본적인 정보와 데이터가 담겨있는 extents 정보를 저장하고 있다. 해당 inode가 폴더 데이터일 경우 extents는 폴더 내부에 존재하는 데이터 리스트 정보가 담긴 directory entry의 주소를 저장한다. 해당 inode가 파일 데이터일 경우 extents는 파일 데이터가 저장된 블록 주소를 저장한다. 그림 3의 inode는 root 디렉토리 정보를 저장하므로 extents는 directory entry를 나타낸다.

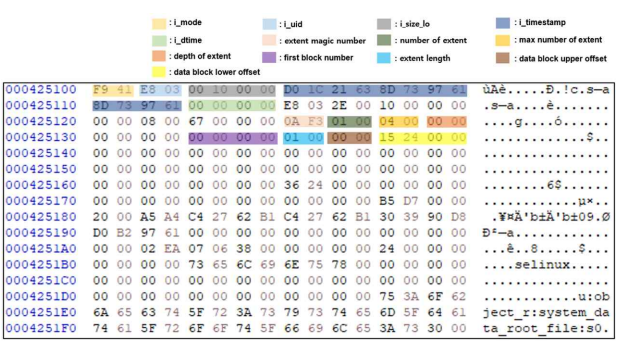

그림 3. 루트 디렉토리의 아이노드
Fig. 3. Root directory inode

그림 4는 root 디렉토리의 directory entry를 나타내며, inode 번호, 데이터 타입, 데이터 이름 등의 정보가 저장되어 있다. 해당 데이터의 inode 번호를 확인 시, 앞서 분석을 진행한 group descriptor로 다시 이동하여, 분석한 방식대로 다시 진행 시 inode를 찾을 수 있다.

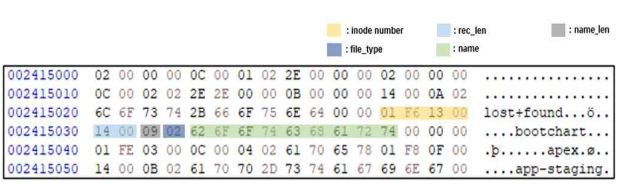

그림 4. 디렉토리 엔트리
Fig. 4. Directory entry

그림 5는 분석을 위해 기기에 저장한 test image.jpg 파일을 가리키는 directory entry와 inode를 나타낸다. Directory entry에서 test image.jpg 파일의 inode 번호가 0x15F8EF 임을 확인하여 해당 inode로 이동할 수 있다. 그림 5의 inode는 test image.jpg인 파일을 나타내는 inode이므로 extents의 주소가 파일의 데이터를 가리킨다. 따라서 extents의 값을 통해 test_image.jpg의 데이터가 저장되어 있음을 알 수 있으며, 실제로 이동 시에 그림 6과 같이 test image.jpg 파일의 데이터를 확인할 수 있다. 해당 데이터는 파일 카빙을 통해 추출이 가능하다.

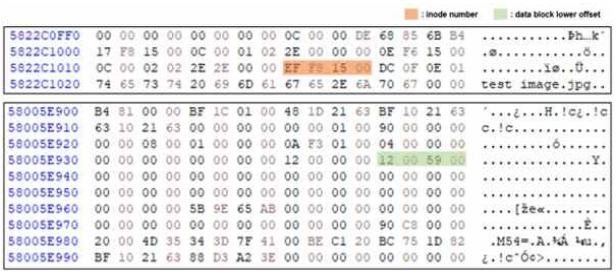

그림 5. Test image.jpg 디렉토리 엔트리(상)/ 아이노드(하)
Fig. 5. Test image.jpg directory entry(top)/ node(bottom)

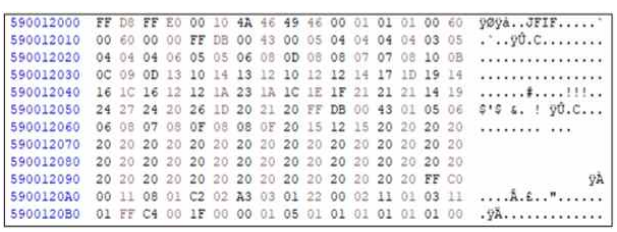

그림 6. test image.jpg 파일 데이터
Fig. 6. test image.jpg file data

4-2 Linux (JFFS2 / UBIFS)

Linux 플랫폼의 분석을 위해 전주에 설치되는 산업용 전력기기 제품인 DCU(Data Central Unit)를 대상으로 연구를 진행한다. DCU는 Linux를 사용하며 JFFS2 및 UBIFS를 데이터 파티션으로 사용한다. 해당 파일시스템들의 분석을 위해 파티션 데이터의 이미지가 필요하기 때문에, 이미지 획득을 위해 PCB 분석을 진행하였다. 분석을 통해 PCB 상에 RS-232 포트가 존재하는 것을 확인했으며, RS-232 포트를 통해 시리얼 통신을 진행했다. 시리얼 통신을 통해 root 계정에 접근하여 확인한 결과 3개의 MTD를 확인할 수 있었다. MTD의 추출을 위해 ssh 서버를 만든 후, 기기 상에서 ssh를 통해 파일을 전송하는 명령어인 scp 명령어를 통해 MTD의 덤프를 진행했다. MTD를 분석한 결과 각각 커널, JFFS2, UBIFS가 탑재되어 있음을 확인했으며, JFFS2 및 UBIFS의 분석을 진행했다.

1) JFFS2

JFFS2는 '0x1985' 라는 매직넘버를 노드 맨 앞에 가지며, 사용하지 않는 영역은 0xFF로 채워져 있다. JFFS2는 여러 종류의 node로 구성되어 있다. 그림 7은 test3.txt 파일의

dirent node를 나타낸다. Dirent node의 node type은 '0xE001'이며 node의 버전 정보, 파일명, 수정시간 등에 대한 정보가 저장되어 있으며, 해당 노드의 inode number를 검색을 통해 일치하는 inode를 찾을 수 있다.

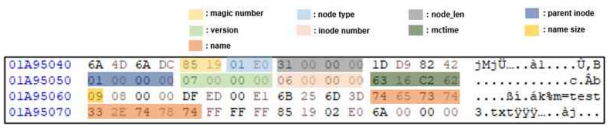

그림 7. 'test3.txt' 파일의 다이런트 노드
Fig. 7. 'test3.txt' file's dirent node

그림 8은 test3.txt 파일의 inode를 나타낸다. Inode에는 파일명이 존재하지 않으므로, dirent node를 통해 파일 및 폴더명을 확인 후 inode number의 비교를 통해 해당 데이터의 inode를 찾을 수 있다. 또한 JFFS2는 효율적인 데이터 관리를 위해 데이터를 압축해서 저장하며, 사용되는 압축 알고리즘은 zlib, rubin, rtime, LZO 중 한 가지가 사용된다. 따라서 데이터의 추출을 위해서는 inode에 담겨있는 압축 타입에 대해서 확인 후, 해당 알고리즘을 사용하여 압축 해제를 진행해야 한다.

그림 8. 'test3.txt' 파일의 아이노드
Fig. 8. 'test3.txt' file's inode

2) UBIFS

UBIFS는 UBI Volume을 통해 작동하기 때문에, MTD device의 데이터를 획득한 경우 UBI Volume에서 UBIFS 영역을 추출해야 한다. UBIFS 이미지 추출은 Github의 ubi reader를 통해 가능하다. 해당 연구에서는 Linux 환경에서 ubi reader v0.8.0를 사용해 UBIFS 이미지를 추출하였다. UBIFS는 node 형태로 데이터를 저장하며, 그림 9는 superblock node를 보여준다. UBIFS의 모든 node들은 앞에 '0x06101831'을 magic number로 갖는다. Superblock 노드에는 UBIFS의 LEB 크기, LEB 개와 같은 기본적인 정보를 알 수 있으며, LEB 크기 데이터는 메타데이터 분석 시에 중요한 정보가 된다.

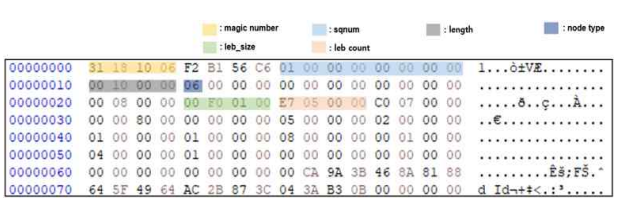

그림 9. UBIFS 슈퍼블록 노드
Fig. 9. UBIFS superblock node

두 번째의 LEB에는 master node가 존재한다. 그림 10은 Master node의 구조를 나타내며, 해당 node는 superblock node와 유사하게 기본적인 정보가 저장되어 있으며, 사용가능한 영역의 크기, 사용된 LEB의 수와 같이 해당 볼륨에 대한 정보들이 저장된다. Master node에는 root의 index node의 위치에 대한 정보를 저장하고 있으므로, 해당 정보를 이용해 root의 index node로 이동할 수 있다.

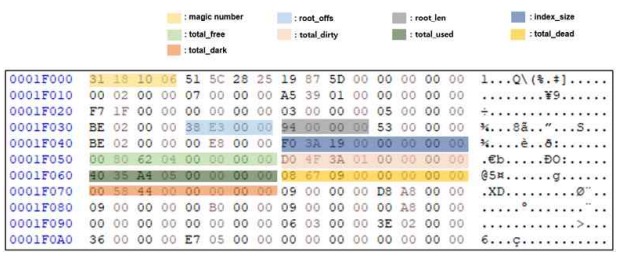

그림 10. UBIFS 마스터 노드
Fig. 10. UBIFS master node

Index node는 LEB 번호 및 node offset에 대한 정보가 존재하며, node가 가리키는 offset을 따라 이동할 시, inode, data node, directory entry node 등 파일에 대한 다양한 정보가 저장되어 있는 node들에 접근할 수 있다. Directory node는 파일 및 폴더의 이름 및 inode 번호를 갖고 있으며, inode에는 파일에 대한 정보를, data node에는 파일의 데이터가 존재한다. 파일의 추출을 위해서 directory entry에서의 파일명 확인 및 inode 번호를 통한 data node로의 이동이 필요하다. 이 때, data node의 inode number 맨 뒤의 값은 0x20으로 변경된다. 그림 11은 dcu에 존재하는 k2logd.sh 파일에 대한 directory entry와 data node를 나타낸다.

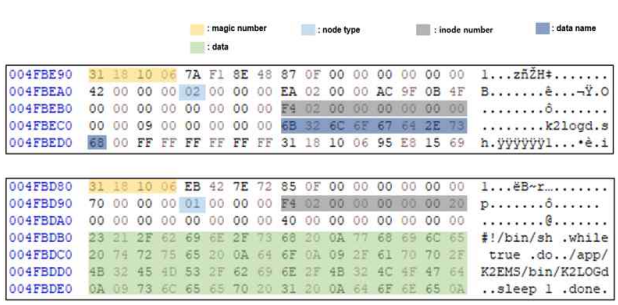

그림 11. 'K2logd.sh' 파일 디렉토리 엔트리(상)/ 데이터 노드(하)
Fig. 11. 'K2logd.sh' file's directory entry(top)/ data node(bottom)

UBIFS는 data node의 접근을 통해 제한적인 데이터의 추출이 가능하다. UBIFS는 파일을 저장할 때 암호화 및 압축을 진행하기 때문에 일정 용량 이상의 파일들은 압축 및 암호화가 진행되어서 추출을 위해서는 복호화 과정을 필요로 한다. 따라서 현재 추출이 가능한 데이터는 압축이 및 암호화가 진행되지 않은 작은 크기의 파일들만 가능하다. 그림 12는 용량

이 큰 dcu.tar 파일의 data node와 실제 데이터를 비교한 그림으로, 암호화 및 압축을 통해 기존의 데이터와 차이가 존재함을 알 수 있다.

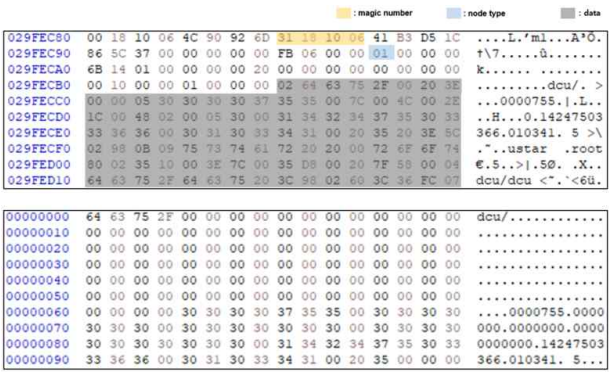

그림 12. 'dcu.tar' 파일의 데이터 노드(상)/ 실제 데이터(하)
Fig. 12. 'dcu.tar' file's data node(top)/ real data(bottom)

V. 파일 복구

5-1 Android Automotive

Android Automotive의 경우 ext4 파일시스템을 사용하며, ext4에는 삭제된 데이터가 남아있는 저널 영역이 존재한다. 따라서 파일 삭제 시, 기존의 inode의 extents 데이터는 삭제되며 저널 영역에 남아있게 된다. 따라서 저널 영역에서 복구를 원하는 파일의 inode를 찾아 파일 카빙을 진행하거나, 기존의 inode에 백업을 진행하는 방식을 통해 삭제된 파일의 복구가 가능하다.

그러나 AAOS의 경우 파일을 삭제하여도 directory entry에서도 삭제되지 않으며, 해당 파일의 inode의 extents 또한 삭제되지 않고 유지되는 것을 확인했다. 그림 13과 같이 autopsy 프로그램을 통해 /media/0/Download 내부를 확인한 결과, test image.jpg 파일이 삭제되었음을 알 수 있었다. 그러나 파일시스템 단계에서 test image.jpg의 inode가 위치한 0x58005E900으로 이동 시, 그림 14와 같이 기존의 ext4의 파일 삭제와 달리 extents 정보가 삭제되지 않고 존재하는 것을 확인했다.

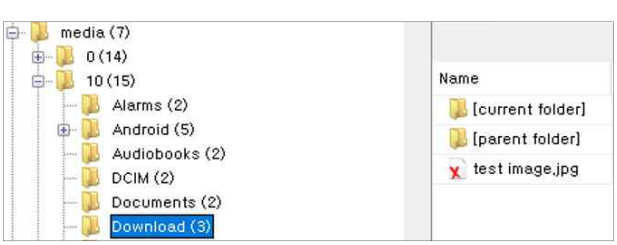

그림 13. 삭제된 'test image.jpg' 파일
Fig. 13. Deleted file 'test image.jpg'

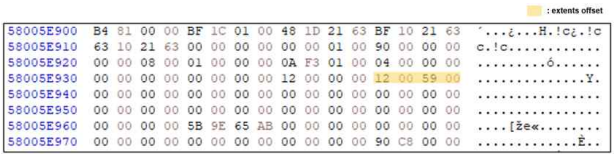

그림 14. 'test image.jpg' 파일 삭제 후 inode
Fig. 14. Inode of 'test image.jpg' file after delete

Ext4의 경우 저널 영역을 사용하기 때문에 기존의 inode의 정보는 삭제된다. 그러나 AAOS의 삭제된 파일 복구 실험을 진행한 결과 기존의 inode의 정보가 유지되는 것을 확인하였다. 그러나 분석 결과 저널 영역이 해당 파티션에 존재하고 있기 때문에, 해당 상황이 AAOS의 특징 때문인지, 라즈베리파이를 통한 실행으로 인해 발생한 상황인지에 대한 확인을 위한 추가적인 연구가 필요하다.

5-2 Linux

1) JFFS2

JFFS2는 저널링 파일시스템이기 때문에 삭제된 파일의 복구 또한 가능하다. 삭제된 파일의 복구를 진행하기 위해 test.txt 파일을 삭제 후, 파티션 덤프를 통해 분석을 진행하였다. 그림 15는 파일 test.txt의 inode 및 dirent inode를 나타낸다. 그림 15의 상단 사진은 파일이 삭제되기 전의 데이터로 version이 0x04이며, 하단은 파일이 삭제된 후의 데이터로 version이 0x05로 변경되었음을 확인할 수 있다. 따라서 JFFS2는 파일 삭제 시 그림 15의 하단 사진과 같이 dirent node만 존재하고 있으며, 해당 파일의 복구는 inode 번호 및 파일 이름의 검색을 통해 파일이 삭제되기 전의 버전의 inode를 찾아 삭제된 파일을 복구할 수 있다.

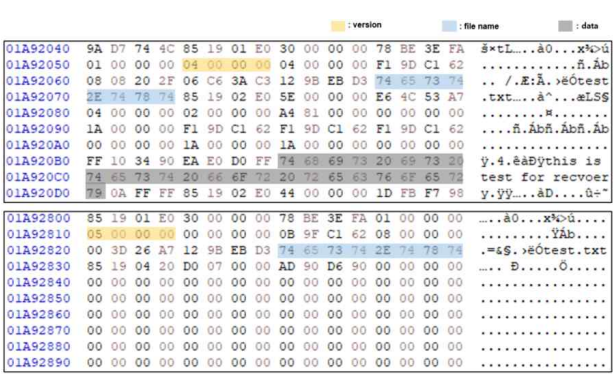

그림 15. 삭제 전 test.txt 파일의 아이노드와 다이런트 노드(상)/ 삭제 후 다이런트 노드(하)
Fig. 15. 'Inode and dirent node of test.txt' file before delete(top)/ dirent node of 'test.txt' file after delete(bottom)

2) UBIFS

UBIFS는 저널링 시스템을 사용하기 때문에 삭제된 파일의 경우에도 데이터가 남아있다. 그러나 UBIFS는 파일 저장

시에 압축 및 암호화를 진행하기 때문에, 데이터를 찾더라도 복호화 및 압축 해제를 진행해야 한다. 그림 16은 dcu.tar 파일의 data node의 압축 및 암호화된 데이터 영역과 실제 dcu 파일의 데이터를 비교하여 압축 및 암호화를 확인할 수 있다. 따라서 데이터의 복호화를 진행하지 않으면 삭제된 파일의 복구는 어렵다고 판단된다.

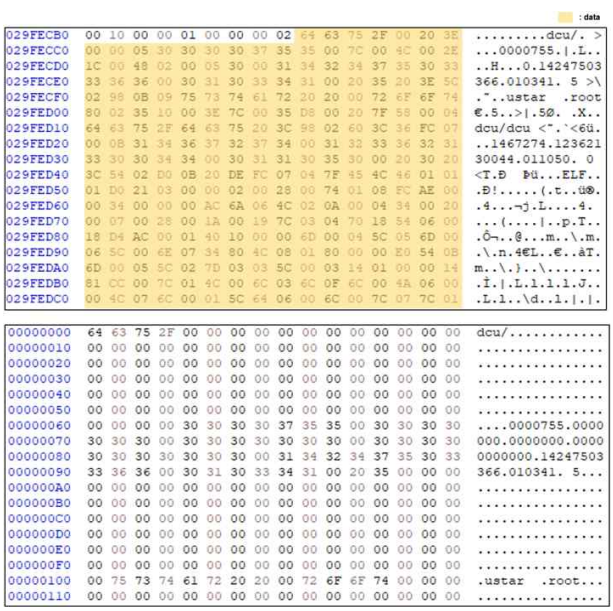

그림 16. 'dcu.tar' 파일의 데이터 노드(상) / 'dcu.tar' 파일의 실제 데이터

Fig. 16. data node of dcu.tar' file(top)/ real data of 'dcu.tar' file

V. 결 론

현재 다양한 리눅스 기반 IoT 플랫폼이 개발되고 사용되고 있기 때문에, 리눅스 기반의 IoT 플랫폼의 분석 연구는 포렌식 관점에서 매우 중요하다. 그러나 단순 기기의 아티팩트 추출의 경우 획득할 수 있는 아티팩트의 종류가 제한될 수 있다. 이에 따라 본 연구에서는 리눅스 기반으로 제작된 파일시스템인 JFFS2 및 UBIFS와 ext4를 대상으로 파일시스템 단계에서의 분석을 진행하였다.

본 연구에서는 현재까지 분석되지 않았거나, 개발 단계에 사용될 수 있는 기기의 파일시스템을 대상으로 보안성 향상과, 범죄 수사 시 도움이 될 수 있도록 파일시스템의 메타데이터의 분석을 진행하였다. 분석한 결과를 통해 파일시스템에서 데이터가 저장되는 구조를 확인하고, 이를 통해 저장된 데이터의 추출 방안을 고안하였다. 뿐만 아니라 수정 및 삭제된 파일의 저널링, 로그 등의 저장 방식을 확인하여 파일시스템별 삭제된 파일의 복구 가능성을 확인하였다. 해당 연구는 리눅스 기반의 파일시스템이 탑재된 IoT 기기에서의 아티팩트의 효율적인 획득을 가능하도록 하였으며, 범죄 발생 시 증거 및 알리바이의 확보에 기여할 것으로 기대된다. 그러나 AAOS의 경우 기존의 ext4와 달리 삭제된 파일의 메타데이터가 남아있는 현상이 존재했다. 또한 UBIFS의 경우 파일 추출 시, 암호화로 인해 크기가 큰 파일의 추출 및 복구를 진행하지 못하였다. 따라서 오픈소스로 제공된 코드 분석이나 실제 AAOS가 탑재되어 출시된 기기와의 비교 분석을 통해 원인을 파악하고, 본 논문에서 제시한 데이터 추출 방안을 적용하는 연구 및 UBIFS의 암호화에 대한 연구를 진행하여, 복호화를 통해 데이터 추출 및 복구에 대한 연구가 필요하다. 이러한 연구를 통해 다양한 파일시스템에 대해 보다 효율적인 데이터 획득이 가능할 것으로 기대되며, 추후 출시될 기기의 보안성 및 기술 향상에 도움이 될 것으로 기대된다.

참고문헌

[1] H. W. Lee "Intrusion Artifact Acquisition Method based on IoT Botnet Malware" *Journal of Internet of Things and Convergence,* Vol. 7, No. 3, pp. 1–8, Sep. 2021. https://doi.org/10.20465/KIOTS.2021.7.3.001

[2] Nbcnews. Amazon's Alexa may have witnessed alleged Florida murder, authorities say[Internet. Available: https://www.nbcnews.com/news/us-news/amazon-s-alexa-may-have-witnessed-alleged-florida-murder-authorities-n1075621

[3] S. Neuner, A. G.Voyiatzis, M. Schmiedecker, S. Brunthaler, S. Katzenbeisser, and E. R.Weippl, “Time is on my side: Steganography in filesystem metadata”, *Digital Investigation*, Vol. 18, pp S76-S86, Aug, 2016, https://doi.org/10.1016/j.diin.2016.04.010

[4] T. Nemayire, A. Ogbole, S. Park, K. Kim, Y. Jeong and Y. Jang, "A 2018 Samsung Smart TV Data Acquisition Method Analysis". *Journal of Digital Forensics,* Vol. 13, No 3, pp. 205-218. Sep. 2019. http://dx.doi.org/10.22798/kdfs.2019.13.3.205

[5] A. Hashmi, H. Berry, O. Temam, and M. Lipasti, "Automatic abstraction and fault tolerance in cortical microachitectures." *ACM SIGARCH computer architecture news*, Vol 39(3), pp. 1-10. June, 2011, https://doi.org/10.1145/2024723.2000066

[6] S. Li, K-K. R.Choo, Q. Sun, W. J. Buchanan and J. Cao, "IoT Forensics: Amazon Echo as a Use Case", *in IEEE Internet of Things Journal,* Vol. 6, no. 4, pp. 6487-6497, Aug. 2019, https://doi.org/10.1109/JIOT.2019.2906946

[7] M. Jarrett and S. Morris. "Purple dawn: Dead disk forensics on Google's Fuchsia operating system." *Forensic Science International: Digital Investigation*, Vol 39, 301269, Sep, 2021, https://doi.org/10.1016/j.fsidi.2021.301269

[8] H. Kim, Y. Shin, S. Kim, W. Jo, M. Kim and T. Shon

"Digital Forensic Analysis to Improve User Privacy on Android." *Sensors,* Vol 22(11), 3971. May, 2022, https://doi.org/10.3390/s22113971

[9] D. Kim, J. Park, Kg. Lee and S. Lee, "Forensic Analysis of Android Phone Using Ext4 File System Journal Log". *in Future Information Technology, Application, and Service. Lecture Notes in Electrical Engineering*, Springer, Vol 164. pp. 435-446, June. 2012, https://doi.org/10.1007/978-94-007-4516-2_44

이진오(Jino Lee)

2021년 : 아주대학교 정보통신대학 사이버보안학과 (공학사)
2021년~현 재 : 아주대학교 대학원 AI융합네트워크학과 석사과정

※관심분야 : 사이버보안, 디지털 포렌식, 파일시스템 등

손태식(Tae-Shik shon)

2000년 : 아주대학교 정보및컴퓨터공학부 졸업 (학사)
2002년 : 아주대학교 정보통신전문대학원 졸업 (석사)
2005년 : 고려대학교 정보보호대학원 졸업 (박사)

2004년~2005년: University of Minnesota 방문연구원
2005년~2011년: 삼성전자 통신·DMC 연구소 책임연구원
2017년~2018년: Illinois Insitute of Technology 방문교수
2011년~현 재: 아주대학교 정보통신대학 사이버보안학과 교수
※관심분야 : Digital Forensics, ICS/Automotive Security