정 보 보 호 학 회 지
제 29 권 제6호, 2019. 12

포렌식 관점에서 차세대 파일시스템 연구 동향

황 현 욱*, 오 정 훈*, 이 승 용*, 김 기 범*, 손 기 욱*

요 약

기존의 NTFS, HFS+, Ext4와 같은 전통적인 파일시스템들은 디스크 사용, 공간 관리, 데이터 암호화 등 여러 측면에서 한계점을 가지고 있었다. 특히 디스크 사용 측면에서 기본적으로 단일 디스크 안에서 동작하도록 설계되었기 때문에 여러 개의 디스크에서 동작하도록 하려면 RAID와 같은 별도의 구성이 필요했다. 이에 따라 주요 운영체제들은 위와 같은 기존 파일시스템들의 한계점들을 극복하도록 설계된 Pooled Storage 파일시스템들을 공개하였다. Pooled Storage 파일시스템에 관한 연구는 2017년 여름 미국 오스틴에서 열린 DFRWS 학회에서 독일의 Jan-Niclas Hilgert에 의해 발표된 이후 디지털 포렌식 학계 및 산업계에서 집중적인 연구개발이 진행되고 있다. 2017년 Hilgert는 ZFS 파일시스템에 대한 분석기능을 공개소프트웨어인 SleuthKit에 추가한 기술을 발표하였고, 2018년 DFRWS에서는 BtrFS 파일시스템에 대한 분석기능을 공개하였다. BlackBag Technologies의 Joe Syle은 APFS 파일시스템에 대한 분석기능을 SleuthKit에 추가한 결과를 DFRWS 2018에서 발표하였다. 노르웨이의 Rune Nordvik은 2019년 DFRWS에서 REFS를 역공학을 통하여 분석한 결과를 공개하였다. 국내에서는 고려대학교를 중심으로 ReFS에 대한 연구가 진행 중이다. 본 논문에서는 주요 운영체제들이 공개한 Pooled Storage 파일시스템 형태의 차세대 파일시스템인 ReFS, APFS, BtrFS를 소개하고 각 파일시스템의 특징과 주요 기능들을 설명한다.

Ⅰ. 서 론

기존의 전통적인 파일시스템인 NTFS, HFS+, Ext4는 디스크 사용, 공간 관리, 데이터 암호화 등 여러 측면에서 한계점을 가지고 있다. 디스크 사용 측면에서는 기본적으로 단일 디스크 안에서 동작하도록 설계되었기 때문에 여러 개의 디스크에서 동작하도록 하려면 RAID와 같은 별도의 구성이 필요하다. 공간 관리 측면에서도 한 개의 고정된 크기의 볼륨을 사용하였기 때문에 여러 볼륨을 사용할 경우, 효율적으로 디스크 공간을 사용하지 못하는 한계점이 있다. 데이터 암호화 측면에서는 자체적인 암호화 기능을 가지고 있지 않은 경우도 존재하여 데이터 암호화를 위해서는 추가적인 운영체제 기능이 필요하다. 예를 들어 HFS+의 경우, 파일시스템 자체적인 데이터 암호화 기능이 없기 때문에 macOS에서 제공하는 Core Storage 기능을 통해 데이터를 암호화해야 한다.

기존 파일시스템들의 이러한 한계점들을 극복하고자 주요 운영체제에서는 차세대 파일시스템들을 공개하였다. Windows 운영체제에서는 ReFS(Resilient File System)을 Windows Server 2012부터 공개하였고 macOS에서는 APFS(Apple File System)를 macOS 10.13 High Sierra부터 공개하였다. 그리고 Linux에서는 BtrFS(B-Tree File System)를 SUSE Linux Enterprise Server12 버전부터 기본 파일시스템으로 사용하기 시작하였다. 이들 차세대 파일시스템들은 현재까지 버전 업그레이드가 계속되고 있으며 여러 기능들도 지속적으로 추가되고 있다.

파일시스템은 데이터를 저장 매체에 데이터를 쓰고 읽는 방법을 정하는 것이다. 디지털포렌식 기술 개발 및 증거 분석을 위해 파일시스템에 대한 이해는 가장 기본이 되는 요소이다. 이에 따라, Brain Carrier가 저술한 File System Forensic Analysis 혹은 주필환님의 번역서가 널리 읽히고 있다[1, 2]. 이 책에는 현재 컴퓨터에서 많이 사용되는 FAT, NTFS, EXT2/3, UFS 파일시스템에 대한 상세한 분석을 다루고 있다. 이들 파일시스템은 저장매체 하나를 여러 개의 볼륨으로 나누고 각각의 볼륨에 특정한 파일시스템이 존재하는 구조를 갖고 있다.

* ETRI부설연구소 (hhu@nsr.re.kr, blueangel@nsr.re.kr, sylee@nsr.re.kr, kibom@nsr.re.kr, kiwook@nsr.re.kr)

독일의 Fraunhofer 연구소의 Jan-Niclas Hilgert는 DFRWS 2017 학회에서 Brain이 제시한 파일시스템 분석 모델이 최근에 개발된 Pooled Storage 파일시스템에는 적합하지 않기 때문에 이를 해결하기 위한 방안을 제시하여 최우수 논문상을 수상하였다[3]. DFRWS 2018에서는 기존 연구결과를 토대로 BtrFS 파일시스템에 대한 분석기능을 공개하였다[4].

본 논문에서는 위에서 언급한 ReFS, APFS, BtrFS를 소개하고 각 파일시스템의 특징과 주요 기능들을 설명한다.

Ⅱ. ReFS 파일시스템

2.1. ReFS 개요

ReFS(Resilient Filesystem)는 Microsoft가 비교적 최근에 개발한 파일시스템으로서 무결성 보장, 복원력 강화, 대용량 저장장치 활용, 데이터 가용성을 극대화하기 위하여 개발되어 Windows Server 2012, Windows 8.1 운영체제에 새롭게 추가되었다. ReFS 파일시스템은 그림 1과 같이 초기 버전인 v1.1에서부터 시작하여 2019년 11월 현재 v3.4 버전이 Windows Server 2019, Windows 10 운영체제에서 사용 가능하다[5]. Windows 10의 경우 Windows 10의 업데이트 버전에 따라서 기본으로 지원되는 ReFS 버전이 다르고 Windows 10 업데이트 버전 1803 이후부터 2019년 11월 현재 업데이트 버전인 1909까지는 v3.4를 지원하고 있다. Windows 10의 모든 종류의 운영체제가 ReFS를 지원하는 것은 아니다. Windows Home과 Windows Pro에서는 ReFS가 지원되지 않고, Windows Pro for Work stations, Windows 10 Enterprise 버전에서만 ReFS가 지원된다[6].

2.2. ReFS 특징

ReFS의 장점은 무결성 스트림, 저장소 공간 통합, 데이터 복구, 사전 오류 수정 기능을 통하여 복원력 향상에 장점이 있으며, 미러가속 패리티, VM 작업 가속화, 가변 클러스터 크기 등의 기능을 이용하여 파일시스템 성능을 향상하였다. ReFS는 다음과 같은 특징을 가지고 있다[7].

- **복원력**

무결성 스트림, 저장소 공간 통합, 데이터 복구, 사전 오류 수정 등의 기능을 도입하여 온라인 상태에서 손상을 정확하게 탐지하여 수정함으로써 데이터의 무결성 및 가용성을 향상 시킨다.

- **성능**

데이터에 대한 고성능 및 용량 효율적인 저장소 관리를 위한 미러가속 패리티 기능, 블록 복제 기능 및 Sparse VDL 기능으로 가상화 작업의 성능 개선하며, 가변 클러스터 크기 기능을 제공하여 대규모의 순차적 IO 작업에도 좋은 성능을 유지한다.

- **확장성**

ReFS는 매우 큰 데이터 집합을 지원하도록 설계되어 성능을 저하 시키지 않으면서 더 큰 확장성을 제공한다.

ReFS	Windows Server 2012	Windows 8.1, Server 2012 R2	Windows 10 v1507 – v1607	Windows Server 2016 TP2, TP3	Windows Server 2016 TP4, TP5	Windows Server 2016 RTM	Windows 10 v1703	Windows 10 v1709, Windows Server 1709[5]	Windows 10 v1803 – v1809, Windows Server 2019, 1803 – 1809[5]
1.1	**Default**	Yes[1]	Yes[1]	Yes[1]	Yes[1]	Yes[1]	Yes[1]	?	?
1.2	Yes	**Default**	**Default**	Yes	Yes	Yes	Yes	Yes	Yes
2.0	No	No	No	No	**Default**	No	No	No	No
3.0	No	No	No	No	No	Yes[2]	Yes[3]	Yes[4]	Yes[6]
3.1	No	No	No	No	No	**Default**	Yes[3]	Yes[4]	Yes[6]
3.2	No	No	No	No	No	No	**Default**	Yes[4]	Yes[6]
3.3	No	No	No	No	No	No	No	**Default**	Yes[6]
3.4	No	No	No	No	No	No	No	No	**Default**

(그림 1) Windows OS별 지원되는 ReFS

2.3. ReFS 사용 방법

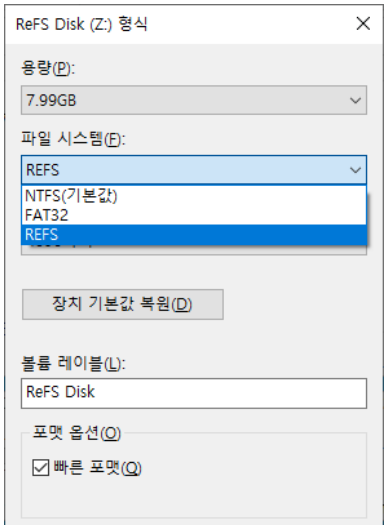
Windows Server 2012 ~ 2019의 서버 운영체제에서는 특별한 설정을 변경하지 않고 ReFS 사용이 가능하지만, 기본으로 설치된 Windows 8.1에서는 ReFS 사용이 불가능하게 되어 있다. 이 운영체제에서 ReFS를 사용하기 위해서는 다음과 같이 특정 레지스트리 키 위치에 MiniNT 키를 생성 후 DWORD 형태의 AllowRefsFormatOverNonmirrorVolume 값을 생성하고 이 값의 데이터를 0x01로 설정해 주어야 한다.

```
Key Path: HKEY_LOCAL_MACHINE\SYSTEM
    \CurrentControlSet\Control\MiniNT

Value Name: AllowRefsFormatOverNonMirrorV
            olume
Value Type: DWORD
Value Data: 1
```

이 레지스트리 값을 설정하면 그림 2와 같이 Windows 8.1에서 파일시스템을 초기화할 때 REFS 형식을 선택할 수 있는 파일시스템이 추가로 표시되어 ReFS 형식으로 포맷할 수 있다.

Windows 10 Pro for Workstations, Enterprise의 경우는 레지스트리 수정 없이 기본으로 설치된 환경에서 ReFS로 포맷이 가능하다.



(그림 2) REFS 포맷 형식 선택

2.4. ReFS와 NTFS 파일시스템과 비교

ReFS와 NTFS에서 파일 이름과 최대 크기, 볼륨의 최대 크기는 표 1과 같이 파일 이름, 경로의 최대 길이는 255자, 32 * 1024 문자로서 동일하지만 단위 파일 최대 크기와 볼륨의 최대 크기는 ReFS 파일시스템이 NTFS보다 140배 정도로 크게 설정이 가능하다[7].

〔표 1〕 ReFS와 NTFS의 최대 한계 비교

Feature	ReFS	NTFS
Maximum file name length in Unicode	255	255
Maximum path name length in Unicode	32 KB	32 KB
Maximum file size	35 PB	256 TB
Maximum volume size	35 PB	256 TB

표 2와 같이 ReFS 파일시스템은 BitLocker 암호화, USN 저널, 접근제어 목록 등의 NTFS에 지원되는 기능을 많이 포함하기도 하지만, 파일 수준 암호화(EFS), 파일 압축, 하드링크, 객체 ID, 짧은 이름 지정 등의 NTFS에서 지원되는 기능들이 ReFS에서 지원되는 않는 기능들이 존재한다.

〔표 2〕 ReFS와 NTFS의 기능 비교

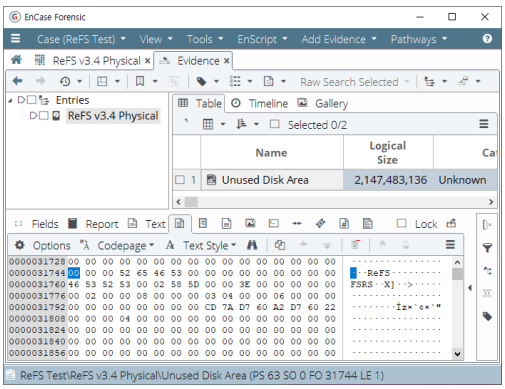
Functionality	ReFS	NTFS
BitLocker encryption	○	○
Data Deduplication	○	○
CSV(Cluster Shared Volume) support	○	○
Soft link	○	○
Failover cluster support	○	○
Access-control lists	○	○
USN journal	○	○
Changes notifications	○	○
Junction points	○	○
Mount points	○	○
Reparse points	○	○
Volume snapshots	○	○
File IDs	○	○
Oplocks	○	○
Sparse files	○	○
Named streams	○	○

Functionality	ReFS	NTFS
Thin Provisioning	○	○
Trim/Unmap	○	○
Block clone	○	X
Sparse VDL	○	X
Mirror-accelerated parity	○	X
Volume Shadow Copy	X	○
File system compression	X	○
File system encryption	X	○
Transactions	X	○
Hard links	X	○
Object IDs	X	○
Offloaded Data Transfer (ODX)	X	○
Short names	X	○
Extended attributes	X	○
Disk quotas	X	○
Bootable	X	○
Supported on removable media	X	○

2.5. ReFS 포렌식 분석 도구와 분석 라이브러리

현재 최신 버전인 EnCase 8.10 버전에서 ReFS v1.2는 분석 가능하지만 삭제된 파일의 파일 복구는 지원되지 않고 있으며, 그림 3과 같이 ReFS v3.4는 분석 자체가 되지 않는다. 그림에서 해당 증거이미지는 물리적 디스크 이미지인데 63번째 섹터가 ReFS Boot Record이며 ReFS 파일시스템임을 확인할 수 있다. X-Ways, AXIOM, FTK Imager에서도 v3.4의 ReFS 파일시스템이 분석되지 않는다.

ReFS 분석 라이브러리와 관련하여 구글의 Joachim



(그림 3) EnCase v8.10의 ReFS 분석 결과

Metz가 GitHub의 libfsrefs 프로젝트에 ReFS 파일시스템 분석 라이브러리 소스코드와 파일 구조 문서를 공개하고 있지만, libfsrefs 라이브러리는 ReFS v1.2에 대한 버전만 지원하고 Windows 10의 v3.4 버전은 지원하지 않고 있다. ReFS 파일시스템 구조 분석 문서에도 v1.2에 대한 구조 설명만 기술하고 있다[8].

Ⅲ. APFS 파일시스템

3.1. APFS 개요

APFS(Apple File System)는 2017년 공개된 macOS 10.13 High Sierra부터 기본 탑재되어 현재까지 macOS의 기본 파일시스템으로 사용되고 있다. macOS 10.13 보다 이전 버전에서 10.13 이후 버전으로 업그레이드할 경우, 자동으로 파일시스템이 HFS+에서 APFS로 업그레이드되기 때문에 macOS 10.13 버전 이후의 운영체제에서는 모두 APFS를 파일시스템을 사용한다.

APFS는 기본적으로 64bit 파일시스템으로 주요 파일시스템 정보들을 64bit 구조에 저장한다. 특히 Inode 번호를 64bit에 저장하여 이론적으로 2^{63}개의 파일을 APFS 파일시스템에 저장할 수 있다. 그리고 Flash/SSD에 최적화된 파일시스템으로 TRIM 명령을 지원하고 운영체제 설치 및 데이터 저장 용도로 모두 사용할 수 있다. 디스크 사용 측면에서는 기본적으로 다중 디스크에서 동작하도록 설계되어 Fusion Drive와 같이 여러 개의 디스크를 하나의 디스크처럼 사용하는 기능을 파일시스템 자체적으로 지원한다. 시간 정보의 경우, Nanosecond Timestamp를 사용하여 1970년 1월 1일부터 현재까지 경과된 시간을 Little Endian 형식으로 저장한다[9].

3.2. APFS 특징과 주요 기능

3.2.1. APFS Container

APFS를 이해하기 위해서는 먼저 Container라는 개념을 알아야 한다. APFS에서 Container란 파일시스템의 메타데이터와 파일/폴더 데이터가 저장되는 논리적인 공간을 말한다. 한 개의 디스크에 여러 개의

Container를 만들 수 있고 반대로 여러 개의 디스크에 걸쳐 한 개의 Container를 만들 수 있다. APFS는 이러한 Container 안에 파일시스템의 모든 데이터를 저장하기 때문에 APFS가 곧 Container라고 볼 수 있다.

Volume은 이러한 Container 내부에 존재하고 하나의 Container 안에 여러 개의 Volume 이 존재할 수 있다. 기존의 전통적인 파일시스템은 각 Volume마다 파일시스템이 존재하여 Volume이 곧 파일시스템이라는 개념이었으나 APFS에서는 반대로 파일시스템 안에 Volume이 존재한다.

디스크 하나를 사용하는 일반적인 경우, 디스크의 GPT 파티션 중 두 번째 파티션에 APFS Container가 위치한다. 그림 4는 macOS shell에서 “diskutil list” 명령으로 디스크 정보를 조회한 결과이다. 먼저 물리적인 장치 파일인 /dev/disk0 의 GPT 파티션 중 두 번째 파티션(2)이 APFS Container임을 알 수 있다. 그리고 APFS Container를 맵핑한 장치파일인 /dev/disk1 의 정보에서 APFS 내 위치한 각 Volume 들의 정보를 확인할 수 있다.

```
Sarahs-MBP-6:~ oompa$ diskutil list
/dev/disk0 (internal):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      GUID_partition_scheme                        1.0 TB     disk0
   1:                        EFI EFI                     314.6 MB   disk0s1
   2:                 Apple_APFS Container disk1         1.0 TB     disk0s2

/dev/disk1 (synthesized):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      APFS Container Scheme -                      +1.0 TB    disk1
                                 Physical Store disk0s2
   1:                APFS Volume HighSierra              650.4 GB   disk1s1
   2:                APFS Volume Preboot                 27.4 MB    disk1s2
   3:                APFS Volume Recovery                517.8 MB   disk1s3
   4:                APFS Volume VM                      4.3 GB     disk1s4
```

(그림 4) APFS 파티션 및 Container 구성

```
Sarahs-MBP-6:~ oompa$ diskutil ap list
APFS Containers (3 found)
|
+-- Container disk1 7AB09481-C4B7-4A1D-B551-6ADAA5D8ED24
    ==========================================================
    APFS Container Reference:     disk1
    Size (Capacity Ceiling):      1000240963584 B (1.0 TB)
    Minimum Size:                 1000655269888 B (1.0 TB)
    Capacity In Use By Volumes:   656089354240 B (656.1 GB) (65.6% used)
    Capacity Not Allocated:       344151609344 B (344.2 GB) (34.4% free)
    |
    +-< Physical Store disk0s2 FC80FCEE-E44F-4F25-BA57-F303E87FB108
    |   ------------------------------------------------------------
    |   APFS Physical Store Disk:   disk0s2
    |   Size:                       1000240963584 B (1.0 TB)
    |
    +-> Volume disk1s1 1D19162C-518C-3A34-A02C-2D428A4BC44E
    |   ------------------------------------------------------------
    |   APFS Volume Disk (Role):   disk1s1 (No specific role)
    |   Name:                      HighSierra (Case-insensitive)
    |   Mount Point:               /
    |   Capacity Consumed:         651046686720 B (651.0 GB)
    |   FileVault:                 Yes (Unlocked)
    |
    +-> Volume disk1s2 670CBB6D-1FE7-446D-A76A-C549A159F34F
    |   ------------------------------------------------------------
    |   APFS Volume Disk (Role):   disk1s2 (Preboot)
    |   Name:                      Preboot (Case-insensitive)
    |   Mount Point:               Not Mounted
    |   Capacity Consumed:         27398144 B (27.4 MB)
    |   FileVault:                 No
    |
    +-> Volume disk1s3 30817573-A0CE-4CF7-AC2B-D7C7E9218424
    |   ------------------------------------------------------------
    |   APFS Volume Disk (Role):   disk1s3 (Recovery)
    |   Name:                      Recovery (Case-insensitive)
    |   Mount Point:               Not Mounted
    |   Capacity Consumed:         517754880 B (517.8 MB)
    |   FileVault:                 No
    |
    +-> Volume disk1s4 38B6B6EE-C683-4FF1-8100-EE6C1A551668
    |   ------------------------------------------------------------
    |   APFS Volume Disk (Role):   disk1s4 (VM)
    |   Name:                      VM (Case-insensitive)
    |   Mount Point:               /private/var/vm
    |   Capacity Consumed:         4295012352 B (4.3 GB)
    |   FileVault:                 No
```

(그림 5) APFS Container 상세 정보

APFS Container에 대한 좀 더 자세한 정보를 확인하기 위해서는 “diskutil ap list” 명령을 사용하여 그림 5와 같이 APFS Container 내 각 Volume 들의 자세한 정보를 확인할 수 있다.

3.2.2. APFS 이미징

디스크를 하나 사용하는 일반적인 경우, APFS 이미징을 수행하기 위해서는 APFS Container와 맵핑되어 있는 /dev/disk1 장치 파일을 입력으로 dd 명령을 실행하면 된다. 만약 디스크를 여러 개 사용할 경우에는 앞에서 언급한 “diskutil list” 명령으로 Container가 맵핑된 장치 파일을 찾은 후, 해당 장치 파일을 입력으로 dd 명령을 수행해야 한다.

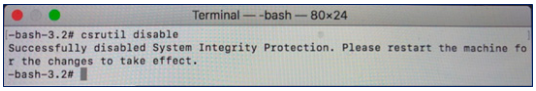
APFS 이미징을 하기 전에 한 가지 주의할 사항은 macOS 10.11부터 지원하는 시스템 보호 기능인 SIP(System Integrity Protection)의 활성화 여부이다. 실제로 dd 명령을 통해 /dev/disk1 장치 파일에 대해 이미지를 생성하려고 하면 그림 6과 같이 root 권한으로도 이미지를 생성할 수 없는 것을 알 수 있다.

```
MacBook-Pro:~ oompa$ sudo dd if=/dev/disk1 of=/tmp/test.dd
dd: /dev/disk1: Operation not permitted
```

(그림 6) /dev/disk1 이미징 실패

위와 같이 dd 명령이 실패하는 이유는 SIP 가 활성화되어 시스템을 보호하기 때문이다. SIP 는 주요 시스템 파일과 디렉터리에 대한 보호 기능으로 root 권한을 가지고도 해당 파일과 디렉터리에 접근할 수 없도록 만든다. 이러한 개념은 rootless라고도 불린다.

따라서 APFS 이미지를 생성하기 위해서는 SIP를 해제하여야 한다. SIP를 해제하기 위해서는 먼저 재부팅 후, “Command-R”을 누르고 있는 상태에서 Recovery Mode로 진입해야 한다. Recovery Mode에서 그림 7과 같이 Recovery Mode 터미널에서 “csrutil disable” 명령을 실행하여 SIP를 해제한 후, 다시 재부팅하면 된다.

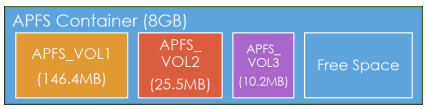


(그림 7) Recovery Mode에서 SIP 해제

APFS Container 이미지를 생성한 후, 다시 SIP를 활성화하려면 동일한 방법으로 Recovery Mode로 진입 후, "csrutil enable" 명령을 실행하고 재부팅하면 된다.

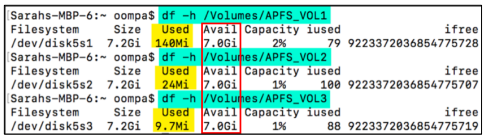
3.2.3. Space Sharing

APFS에서 Container 내 여러 Volume 들이 있을 경우, 각 Volume은 Container 내 남은 공간(Free Space)을 공유한다. 즉 Container 내의 Volume은 기존 파일시스템과 같이 고정된 크기를 갖지 않으며 Container 공간이 남아 있는 한 자유롭게 크기를 늘리고 줄일 수 있다. 이를 통해 APFS는 디스크 공간을 효율적으로 사용할 수 있게 된다.



(그림 8) APFS Space Sharing

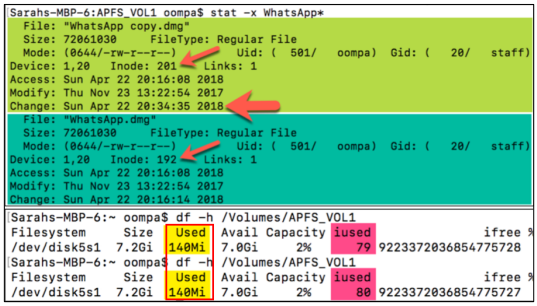
실제로 그림 9와 같이 "df -h" 명령을 통해 Container 내의 Volume들의 크기 정보를 확인해 보면 사용 가능한 공간 크기(Avail)가 모두 동일함을 알 수 있다. 즉, 세 개의 Volume 모두가 남은 공간 전체를 함께 공유한다는 의미이다.



(그림 9) 각 Volume에 대한 df 명령 결과

3.2.4. Clone

APFS에서는 디스크 공간을 효율적으로 쓰는 또 다른 방법으로 Clone 기능을 지원한다. Clone 기능은 파일/디렉터리 복사 시 디스크 공간을 추가 할당하지 않고 파일 메타데이터만 생성하는 기능을 말한다.



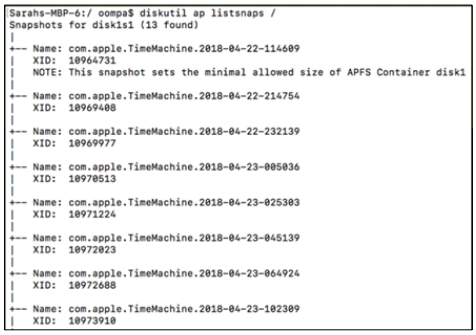
(그림 10) Clone 기능

그림 10은 "WhatsApp.dmg" 파일을 복사하여 "WhatsApp copy.dmg" 파일을 만든 경우로 Clone 기능을 통해 inode 개수(iused)만 증가하고 디스크 사용량(Used)은 증가하지 않는 것을 알 수 있다.

만약 복사된 파일의 데이터가 변경되면 새로운 디스크 공간을 할당하여 변경된 데이터를 저장한다.

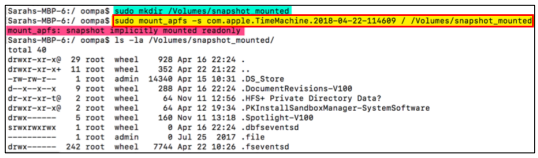
3.2.5. Snapshot

Snapshot 기능은 파일시스템의 특정 시점을 읽기 전용으로 저장하는 기능으로 백업 수단으로 사용된다. Snapshot 백업은 일반적으로 한 시간에 한 번씩 자동 생성되거나 macOS 업데이트 전에 자동 생성되며 생성 후, 24시간 동안 유지된다. 사용자가 직접 snapshot을 생성하고 싶을 경우, shell에서 "tmutil localsnapshot" 명령을 사용하면 된다. 생성된 Snapshot 백업 정보를 확인하는 방법은 그림 11과 같이 "diskutil ap listsnaps" 명령을 사용하거나 "tmutil listlocalsnapshots" 명령을 사용하는 방법이 있다.



(그림 11) Snapshot 정보 확인

생성된 snapshot 백업 데이터는 macOS 10.12 버전부터 mount_apfs 명령과 조합하여 특정 지점에 마운트 시킬 수 있으며 이러한 기능은 특정 시점의 포렌식 분석용 이미지를 만드는데 사용될 수 있다. 그림 12는 mount_apfs 명령을 통해 특정 시점의 snapshot 백업 데이터를 마운트 시킨 예이다.

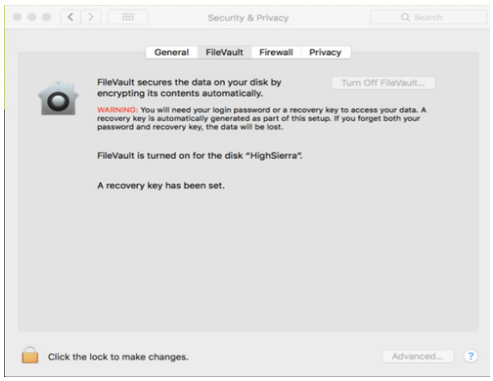


(그림 12) mount_apfs 명령을 통한 shapshot 마운트

3.2.6. Native Encryption

기존의 HFS+ 파일시스템에서는 데이터 암호화를 위해 운영체제의 Core Storage 기능이 필요하다. 이와 다르게 APFS에서는 파일시스템 자체적으로 데이터 암호화 기능을 지원한다. 따라서 macOS 10.13 이전 버전에서 데이터 암호화 기능인 FileVault를 사용하려면 파일시스템을 Core Storage로 구성해야 하지만 10.13 버전부터는 별도의 구성 필요 없이 바로 FileVault 기능을 사용할 수 있다.

데이터 암호화는 볼륨 별로 하나의 Key를 통해 암호화하는 방법과 볼륨 내 메타데이터, 파일 메타데이터, 파일 데이터를 각각 서로 다른 Key로 암호화하는 방법이 있다. 암호화 알고리즘은 AES_XTS, AES_CBC를 지원하며 이는 하드웨어에 따라 사용하는 알고리즘이 달라진다.



(그림 13) FileVault 기능 활성화

3.3. APFS 포렌식 분석

APFS를 분석하는 대표적인 도구로는 BlackBag의 MacQuisition이 있다. 해당 제품은 macOS 시스템의 라이브 상태에서 실행되는 것이 아니라, 시스템 재부팅 후, USB 부팅 형태로 실행되는 제품으로 이러한 방식을 사용하는 이유는 macOS 운영체제의 SIP(System Integrity Protection) 기능으로 인해 라이브 상태에서는 디스크에 대한 직접적인 접근이 불가능하기 때문이다. USB에 저장된 MacQuisition 프로그램은 USB 부팅 후 실행되어 디스크에 직접 접근하여 APFS를 분석한다. 지원되는 기능으로는 디스크 이미지 생성, 파일/디렉터리 분석, 삭제된 파일 복구, Snapshot 분석, FileVault 복호화 작업 등이 있다.

이외에 APFS를 분석해주는 도구로는 EnCase, AXIOM, FTK, X-Ways가 있다. 위 도구들은 디스크 이미지 생성 기능을 지원하지 않으며 생성된 이미지에 대한 분석만 지원해준다. 그리고 해당 도구들은 기본적인 파일/디렉터리 분석은 모두 지원하지만, 삭제 파일 복구, Snapshot 분석, FileVault 복호화 기능 등은 도구에 따라 아직 지원하지 않는 경우도 있다.

Ⅳ. BtrFS 파일시스템

4.1. BtrFS 개요 및 특징

BtrFS 파일시스템은 B-tree File System의 약어로 IBM 연구원 Ohad Rodeh에 의해 USENIX 2007에서 제안된 Copy-On-Write(이하 COW) B-tree 개념[10]을 기반으로 오라클의 Chris Mason에 의해 구현이 시작되었다. BtrFS는 2008년 1.0이 개발된 이후 2009년에는 Linux의 커널에 포함되었으며, 최근에는 리눅스뿐만 아니라 Synology, 넷기어 등의 NAS 장비에도 포함되어 활용 중이다. BtrFS의 주요 특징은 다음과 같다.

- **스냅샷**

BtrFS는 자신이 원하는 시점에 스냅샷으로 저장해놓은 서브볼륨 구조를 이용해 파일을 예전 상태로 돌릴 수 있는 기능을 제공하며, 디렉터리 전체를 되돌릴 수 있을 뿐 아니라 특정 파일만 되돌리는 것도 가능하다. 이는 파일의 변경 이력을 저장하는 방식을 사용하고 있어 저장 공간도 적고 디스크 부하도 적은 장점을 지닌다. 스

냅샷 저장 주기를 적절히 설정하면 백업뿐만 아니라 악성 코드나 랜섬웨어의 감염에도 대응할 수 있다.

- **클로닝**

BtrFS는 파일의 COW 스냅샷을 단위작업(atomic)으로 생성하는 clone 연산자를 제공한다. 파일을 복사하는 경우, 새로운 inode를 생성하지만, 실제 데이터 블록은 복제되지 않으며, 복제된 파일의 수정사항은 원본 파일에서 확인할 수 없으며 반대의 경우에도 동일하게 작용한다.

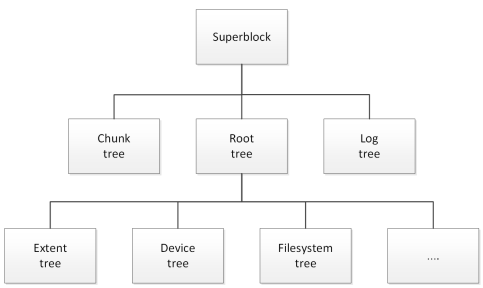
- **ext2/3/4 및 ReiserFS에서 내부 변환**

고정된 위치에 고정된 메타데이터가 거의 없기 때문에 BtrFS는 btrfs-convert 도구를 이용하여 ext2/3/4 및 ReiserFS 파일시스템의 수정되지 않은 복사본을 적절하게 포함 할 수 있다[11]. BtrFS 파일은 기존 파일시스템의 메타데이터 사본을 작성하지만, 데이터의 경우 동일한 블록을 가리키도록 한다.

이와 같은 특징 이외에도 zlib, LZO 압축기능 제공과 SSD 드라이브에 최적화되어 있어 “압축+SSD”를 함께 사용할 때 읽기/쓰기 성능이 효율적으로 작동할 수 있다. 또한, 실시간 오류 정정 기능과 스냅샷을 이용하여 볼륨 복원이 가능한 장애 복원성이 좋지만, 스냅샷의 이미지 저장을 위해 디스크 공간을 추가로 사용하여 디스크 사용량 예측이 어려운 단점도 존재한다.

4.2. BtrFS 기본 구조

BtrFS 파일시스템은 리눅스 기반의 포맷 방식을 사용하는 파일시스템으로 모든 정보를 트리로 관리한다. BtrFS 파일시스템의 기본 구조는 그림 14와 같다.



(그림 14) BtrFS 파일시스템 구조

BtrFS에서 하나의 서브볼륨/스냅샷은 하나의 파일시스템 트리와 대응되어 정보를 유지하기 때문에 BtrFS 파일시스템에는 여러 개의 파일시스템 트리가 존재할 수 있다. 루트 트리(Root tree)는 파일 시스템 트리를 포함하는 모든 트리의 루트에 대한 참조를 트리로 구성하여 기록한다.

- **Superblock**

파일시스템의 핵심정보를 기록하는 Superblock은 64KB 위치에 있고 사본은 유효한 위치인 경우 64MB, 256GB, 1PB에 존재한다. Root Tree, Chunk Tree, Log Tree의 논리적 주소값을 가지고 있고, 섹터 크기, 노드 크기, 모든 시스템 Chunk의 쌍에 관한 정보를 가지고 있다.

- **Root Tree**

모든 트리의 루트에 대한 정보를 저장하는 트리를 Root Tree라고 한다. Root Tree의 Leaf 노드의 아이템은 Key의 아이템 형식에 따라서 Inode item, Inode reference item, Root item, Root reference item을 저장한다.

- **Chunk Tree**

BtrFS 파일시스템은 블록 장치를 256MB 또는 1GB 크기의 Chunk라는 단위로 분할하여 관리한다. 이 Chunk들은 논리적 주소 공간을 통해 실제 블록 장치의 주소로 대응된다. Chunk Tree는 주어진 논리 주소에 해당하는 장치 및 실제 주소를 물리적인 주소로 연결하는 정보를 가지고 있다.

Chunk Tree의 Leaf 노드의 아이템은 Key의 아이템 형식에 따라서 Device Item과 Chunk Item을 가지고 있다.

- **Extent Tree**

BtrFS 파일시스템에서는 공간 할당 단위로 4KB의 Extent를 사용한다. Extent는 Block Group Extent, Data Extent, Metadata Extent로 나눌 수 있는데 이러한 Extent들을 관리하기 위한 트리가 Extent Tree이다. 블록 그룹은 파일시스템의 가상 주소 공간을 구성하는 기본 할당 단위로 블록 장치의 개수에 따라 여러 개의 Chunk로 구성된다. Extent는 파일시스템 할당자에 의해 Block Group을 나누어 할당한 공간으로 Data

Extent와 Metadata Extent로 나뉜다.

Extent Tree의 Leaf 노드의 아이템은 Key의 아이템 형식에 따라서 Block group extent Item, Extent Item, Metadata extent Item을 포함한다.

- File System Tree

File System Tree는 파일/디렉터리 개체들의 정보를 저장하는 트리이다. File System Tree의 Leaf 노드의 아이템은 Key의 아이템 형식에 따라서 Inode item, Directory Item, File extent item을 포함한다.

4.3. BtrFS 파일시스템 및 Raid 구조 분석

BtrFS 파일 시스템의 분석 과정은 다음의 순서로 분석이 가능하다.

(1) Superblock의 식별자(magic) 필드를 확인하여 BtrFS 파일 시스템 여부를 확인한다.
(2) Superblock 구조체를 분석한다.
(3) Superblock에서 Chunk Tree의 루트 노드의 주소를 확인하여 Chunk Tree를 분석한다.
(4) Superblock에서 분석된 Root Tree의 루트 노드 위치를 이용해 Root Tree를 분석한다.
(5) Root Tree에서 분석된 Extent Tree의 루트 노드 위치를 이용해 Extent Tree를 분석한다.
(6) Root Tree에서 분석된 File System Tree의 루트 노드 위치를 이용해 File System 트리를 분석한다.
(7) Root Tree에서 서브볼륨의 개수와 그 위치를 분석하여 서브볼륨의 File System Tree를 분석한다.

Root Tree에서 알아낸 각각 Tree의 루트 노드의 위치는 논리적 위치이므로 실제 데이터가 있는 물리적 위치로 변환하기 위해서 Chunk 트리를 이용하면 된다.

BtrFS 파일시스템은 자체적으로 RAID를 지원한다. 따라서 BtrFS 파일 시스템은 하나의 볼륨은 여러 개의 디바이스로 구성될 수 있다. 여러 개의 디바이스를 입력받으면 같은 볼륨에 묶이는 디바이스를 분류하고, 같은 볼륨에 묶이는 디바이스들은 SuperBlock의 fsid가 동일하므로 각 디바이스의 fsid를 확인하여 분류한다. 이때 분류된 디바이스의 개수가 SuperBlock의 num_devices와 같지 않으면 RAID를 이루는 모든 디바이스가 입력되지 않은 것이다.

모든 디바이스가 입력된 경우 볼륨 분석 순서는 다음과 같은 순서로 분석이 가능하다.

(1) 입력된 디바이스들로 부터 가장 높은 가장 높은 생성번호를 가지고 있는 Superblock을 선택한다. Superblock은 디바이스마다 64KB, 64MB, 256GB, 1PB의 위치에 있으므로 읽어드린 모든 Superblock을 비교하여 선택한다.
(2) Superblock로부터 Chunk Tree의 주소를 알아내고, Superblock의 KeyChunkItem을 이용하여 Chunk Tree를 분석한다.
(3) 분석된 Chunk Tree와 Superblock으로부터 Root Tree의 주소를 이용해 Root Tree를 분석한다.
(4) 분석된 Chunk Tree와 Root Tree를 이용해 나머지 Device Tree, Extent Tree, File System Tree를 분석한다.
(5) Tree 읽기를 실패한 경우 이전에 읽었던 Superblock의 생성번호보다 낮은 생성번호 가진 Superblock 중 가장 높은 생성번호를 기준으로 Tree를 분석한다.
(6) 분석된 Tree를 이용하여 파일 디렉터리 구조를 생성한다.
(7) Root Tree에 SubVolume의 File System 트리가 있는지 확인하여 서브 볼륨을 분석한다.

4.4. BtrFS 포렌식 분석

BtrFS 파일시스템의 삭제된 파일을 복구하기 위해서는, 파일 및 디렉터리에 할당된 메타데이터만 트리에 저장하므로 최근 트리로부터 삭제 파일 복구를 위해 할당되지 않은 메타데이터를 검색해도 삭제된 데이터를 찾을 수 없으며, Copy-On-Write 원칙으로 각 트랜잭션은 새로운 Root Tree를 생성하고 새로운 생성 번호를 생성하므로, 이전 Root Tree를 분석하여 삭제된 파일을 복구할 수 있다. 이전 Root Tree 정보는 Superblock의 root-backup의 정보를 통해 접근이 가능하다.

현재 시점에서 BtrFS 파일시스템 분석 및 삭제 파일 복구는 상용도구인 최신 EnCase 8.10 버전에서 분석을 지원하지는 않으며, 또한 X-Ways, AXIOM, FTK

Imager에서도 BtrFS 파일시스템의 분석 및 삭제 파일 복구를 지원하지 않는다. BtrFS 파일시스템 분석은 SleuthKit에 포함된 모듈을 이용하여 분석이 가능하다.

Ⅴ. 결 론

주요 운영체제들은 기존 전통적인 파일시스템의 한계점을 극복하기 위해 여러 차세대 파일시스템들을 공개하였다. 차세대 파일시스템들은 기본적으로 전통적인 파일시스템과는 전혀 다른 구조로 설계되었으며 기존에 없던 여러 기능들을 가지고 있다. 따라서 디지털 수사 과정에서 해당 파일시스템들을 분석하기 위해서는 파일시스템의 특징과 기능을 정확하게 이해하여야 한다. 이를 위해 본 논문에서는 주요 운영체제의 대표적인 차세대 파일시스템인 ReFS, APFS, BtrFS의 구조와 특징 그리고 주요 기능들을 설명하였다. 포렌식 관점에서 각각의 파일시스템들의 기본적인 디렉터리와 파일의 트리 구조 형태로 분석이 가능해야 하며, 삭제된 파일 복구, 저널링 복원, 파일시스템 내부의 주요 아티팩트, RAID 적용시 다중 디스크 처리 등에 대해 포렌식 도구에서 지원이 필요하다.

참 고 문 헌

[1] Brain Carrier, File System Forensic Analysis, *Addison-Wesley*, 2005

[2] 주필환, 파일시스템 포렌식 분석, 케이앤피IT, 2010

[3] Jan-Niclas Hilgert et al., “Extending The Sleuth Kit and its underlying model for pooled storage file system forensic analysis”, *DFRWS USA 2017*

[4] Jan-Niclas Hilgert et al.,, “Forensic Analysis of Multiple BTRFS Configurations using the Sleuth Kit”, *DFRWS USA 2018*

[5] Wikipedia, “ReFS”, https://en.wikipedia.org/wiki/ReFS

[6] Microsoft, “Compare Windows 10 editions”, https://www.microsoft.com/en-us/windowsforbusiness/compare

[7] Microsoft, “Resilient File System (ReFS) overview”, https://docs.microsoft.com/en-us/windows-server/storage/refs/refs-overview?redirectedfrom=MSDN#supported-deployments

[8] Joachim Metz, “Library and tools to access the Resilient File System(ReFS)”, http://github.com/libyal/libfsrefs

[9] Sarah Edwards, “Getting Saucy with APFS!”, *BsidesCharm 2018*

[10] Rodeh Ohad, “B-trees shadowing, and clones”, *USENIX 2008*

[11] Btrfs Wiki, https://wiki.archilinux.org/index.php/Btrfs, https://en.wikipedia.org/wiki/Btrfs

〈저 자 소 개〉

황 현 욱 (Hyunuk Hwang)

2004년 8월 : 전남대학교 일반대학원 정보보호협동과정 이학박사
2014년 12월~2015년 12월 : Arizona State University 교환연구원
2004년 9월 ~ 현재 : ETRI부설연구소 책임연구원(실장)
<관심분야> 디지털 포렌식, 취약점 검증, 악성코드, AI

오 정 훈 (Junghoon Oh)

2012년 2월 : 고려대학교 정보보호대학원 석사
2012년 1월~2015년 6월 : 안랩 A-FIRST 주임연구원
2015년 7월 ~ 현재 : ETRI부설연구소 선임연구원
<관심분야> 디지털 포렌식, 파일시스템, 침해사고대응, AI

이 승 용 (Seungyong Lee)

2004년 8월 : 전남대학교 일반대학원 정보보호협동과정 이학박사
2008년 5월 ~ 현재 : ETRI부설연구소 책임연구원
<관심분야> 디지털 포렌식, 침해사고 대응, 포렌식 도구 검증

김 기 범 (Kibom Kim)

2001년 2월 : 고려대학교 전산과학과 박사
2001년 1월~2004년 7월 : ㈜이씨오 개발 부장
2004년 8월 ~ 현재 : ETRI부설연구소 책임연구원(실장)
<관심분야> 디지털 포렌식, 사이버보안

손 기 욱 (Kiwook Sohn)

2002년 8월 : 성균관대학교 전기전자컴퓨터공학부 공학박사
1992년 1월~1999년 12월 : 한국전자통신연구원 부호기술부 선임연구원
2000년 1월 ~ 현재 : ETRI부설연구소 책임연구원(본부장)
<관심분야> 디지털 포렌식, 사이버보안