리눅스 mlocate.db 파일 포맷 분석 및 활용 방안

김 재 청*, 이 상 진**
서울지방경찰청 (수사관)*, 고려대학교 정보보호대학원 (교수)**

The analysis of Linux mlocate.db and its application

Jaecheong Kim,* Sangjin Lee**
Seoul Metropolitan Police Agency (Detective)*
Graduate School of Information Security, Korea University (Professor)**

요 약

mlocate는 리눅스 운영체제에서 파일과 디렉터리 목록을 주기적으로 데이터베이스에 저장하고, 편리하게 검색할 수 있도록 도와주는 유틸리티이다. 파일검색 용도로 주로 사용되는 find, which, whereis 명령어보다 사용방법이 간단하고, 빠른 속도로 검색할 수 있어 주요 리눅스 배포본에 기본적으로 제공된다. 파일과 디렉터리 목록과 함께 시간 정보를 포함하고 있어 디지털 포렌식 관점에서도 유용한 아티팩트이다. 업데이트할 때 데이터베이스 파일을 삭제하고, 새로운 파일을 생성하는 특징이 있어 복구를 통해 과거 생성·삭제된 파일 및 디렉터리의 흔적과 시간을 특정할 수 있다. 본 논문에서는 분석 도구를 개발하여 디지털 포렌식 업무에 적용할 수 있도록 제안하였고, 사례를 통해 개발 도구를 실무에 적용하고 효용성을 검증하였다.

주제어 : 디지털 포렌식 조사, 디지털 증거, 리눅스 포렌식, 파일 시스템

ABSTRACT

The mlocate is a utility that helps Linux operating systems store file and directory lists in databases on a regular basis and search conveniently. The method of use is simpler than the find, which, and whereis commands are used primarily for file retrieval purposes, and can be retrieved at a faster rate, which is standard for major Linux distributions. It contains time information along with a list of files and directories, making it a useful artifact from a digital forensics perspective. With features that delete database files and create new files when updated, recovery can specify traces and times of files and directories that have been created and deleted in the past. In this paper, the analysis tools were developed and proposed to be applicable to digital forensics work, and the development tools were applied in practice through the case and the utility was verified.

Keywords : Digital Forensics Investigation, Digital Evidence, Linux Forensics, Filesystem

Ⅰ. 서 론

사물인터넷(IoT), 클라우드 등 신기술이 확산되면서 오픈소스 운영체제(OS)인 '리눅스'가 마이크로소프트(MS) OS '윈도'의 아성을 넘어서고 있다[1]. 리눅스 재단에 따르면 퍼블릭 클라우드 컴퓨팅 워크로드의 90%, 세계 스마트폰의 75.16%, 임베디드 기기의 62%, 슈퍼컴퓨터 시장의 99%가 리눅스로 작동한다. 상당수의 웹서버와 모바일 장치를 구동하는 운영체제다. 데스크톱 시장 또한 성장하고 있는 추세이다[2]. 행정안전부는 2019. 5. 15. 행정·공공기관이 사용하는 인터넷망 PC에 대해 리눅스 기반 운영체제를 2021년부터 단계적으로 도입, 확산할 계획이고, 중장기적으로는, 다양한 SW가 설치·운영되고 있는 업무망 PC로 개방형 OS를 확대해 나갈 계획이라고 밝혔다[3].

마이크로소프트사 윈도우즈 운영체제의 포렌식 아티팩트에 대한 다양한 연구가 활발하게 진행되고 있지만, 리눅스 운영체

▪ Received 28 November 2019, Revised 03 December 2019, Accepted 17 December 2019
▪ 제1저자(First Author) : Jaecheong Kim (Email: evefear@gmail.com)
▪ 교신저자(Corresponding Author) : Sangjin Lee (Email: sangjin@korea.ac.kr)

제의 포렌식 아티팩트에 대한 연구는 상대적으로 부족한 편이다. 윈도우즈 운영체제에는 registry, lnk, jumplist, shellbag, prefetch, windows.edb, iconcache.db, event logs 등 다양한 아티팩트들이 존재한다. 그러나 리눅스 운영체제는 윈도우즈 운영체제와 같이 사용자의 행위를 추정할 수 있는 아티팩트가 부족한 실정이고, 주로 운영체제에서 저장하는 /var/log 하위에 로그들이 전부이다. [표 2]와 같이 클라이언트를 대상으로 하는 운영체제 특성상 운영과 서비스에 관련된 로그들로 구성되어 있어 침해사고대응 또는 장애진단 용도로 주로 사용되었고, 사용자 행위를 판단하는 요소는 매우 부족하다.

본 논문에서는 [표 1]과 같이 리눅스 운영체제 배포본에 기본으로 제공되고 있는 mlocate.db의 구조를 분석하고, 주요 포렌식 항목을 도출하고 실제 사건 사례를 통해 유효성을 검증했다.

표 1. mlocate 패키지 지원 현황 pkgs.org 자료[4]
Table 1. mlocate package support status pkgs.org resources[4]

Linux Name	mlocate package
ALT Linux Sisyphus	mlocate-0.26-alt2.x86_64.rpm
Arch Linux	mlocate-0.26.git.20170220-2-x86_64.pkg.tar.xz
CentOS 8	mlocate-0.26-20.el8.x86_64.rpm
Debian 10	mlocate_0.26-3_amd64.deb
Fedora 31	mlocate-0.26-24.fc31.x86_64.rpm
Mageia 7	mlocate-0.26-13.mga7.x86_64.rpm
OpenMandriva Lx 4.0	mlocate-0.26-24-omv4000.x86_64.rpm
openSUSE Leap 15.1	mlocate-0.26-lp151.5.1.x86_64.rpm
PCLinuxOS	mlocate-0.22.2-1pclos2011.x86_64.rpm
RedHat	mlocate-0.26-20.el8.x86_64.rpm
ROSA 2016.1	mlocate-0.24-14-rosa2016.1.x86_64.rpm
Slackware	mlocate-0.26-x86_64-2.txz
Ubuntu 19.10	mlocate_0.26-3ubuntu3_amd64.deb

Ⅱ. 관련 연구

앞서 언급한 바와 같이 윈도우즈 운영체제의 포렌식 아티팩트에 대한 다양한 연구가 활발하게 진행되고 있지만, 리눅스 운영체제의 포렌식 아티팩트에 대한 연구는 상대적으로 부족하다.

현재까지 mlocate.db에 관한 연구는 Linux.com에서 Chris Binnie가 Finding Files With Mlocate: Part 1~3 연재하면서 mlocate에 관해 소개한 것이 유일하다[5].

리눅스 운영체제에서 저장하는 로그는 [표 2]와 같이 매우 제한적이고 로그의 형태는 대부분 텍스트 형태로 저장되어 있다. 이번 연구에서 다루게 될 mlocate.db는 텍스트 형태로 저장되지 않아 정확한 분석을 위해서는 구문분석이 필요하다.

리눅스는 오픈소스 정책으로 관련 패키지들의 소스 코드가 공개되어 있고 리눅스 운영체제 매뉴얼에서 제공하는 설명 정보를 통해 mlocate.db 특성에 관해 확인할 수 있다.

기존 분석방법은 분석대상 컴퓨터에서 mlocate.db에서 strings 프로그램으로 문자열을 추출하거나 mlocate -d 〈path〉〈mlocate.db〉 옵션을 이용하여 분석대상 mlocate.db의 경로를 수동으로 지정하고, 찾는 문자열을 검색하는 방식이었다. 위와 같은 방법으로 분석할 경우 해당 파일이나 디렉터리가 존재했었는지만 알 수 있었을 뿐, 시간 정보는 확인할 수 없다.

mlocate.db 파일의 구체적인 파일 구조를 비롯하여 파일의 이름, 경로 등의 정보가 어떻게 저장되는지 구체적인 연구가 부족하고, mlocate.db 파일이 디지털 포렌식 분석대상으로 활용되기 위해서는 추가적인 연구가 필요하다.

표 2. 리눅스 운영체제 로그
Table 2. Linux operating system logs

Log	Content
/var/log/messages	General message and system related stuff
/var/log/auth.log	Authenication logs
/var/log/kern.log	Kernel logs
/var/log/cron.log	Crond logs (cron job)
/var/log/maillog	Mail server logs
/var/log/httpd/apache.log or error.log	Apache access and error logs directory
/var/log/boot.log	System boot log
/var/log/mysqld.log	MySQL database server log file
/var/log/secure or /var/log/auth.log	Authentication log
/var/log/utmp or /var/log/wtmp	Login records file

Ⅲ. mlocate.db 파일 구조

3.1 파일 경로

mlocate 패키지의 기본저장 경로와 파일명은 〔표 3〕과 같다.

표 3. mlocate 경로 정보
Table 3. mlocate path info

File Name	Path	Content
mlocate.db	/var/lib/mlocate/mlocate.db	Database File
updatedb.conf	/etc/updatedb.conf	updatedb Configuration File
updatedb	/usr/bin/updatedb	mlocate.db builder program
mlocate	/usr/bin/mlocate	find program

3.2 전체 구조

mlocate.db 파일의 전체 구조는 〔그림 1〕과 같이 헤더 정보와 디렉터리 엔트리 정보로 구분되어 있다. 헤더 정보에는 mlocate.db 파일의 시그니쳐 0\mlocate가 존재하고, 이후 데이터베이스 업데이트와 관련된 설정(updatedb.conf) 내용이 저장된다.

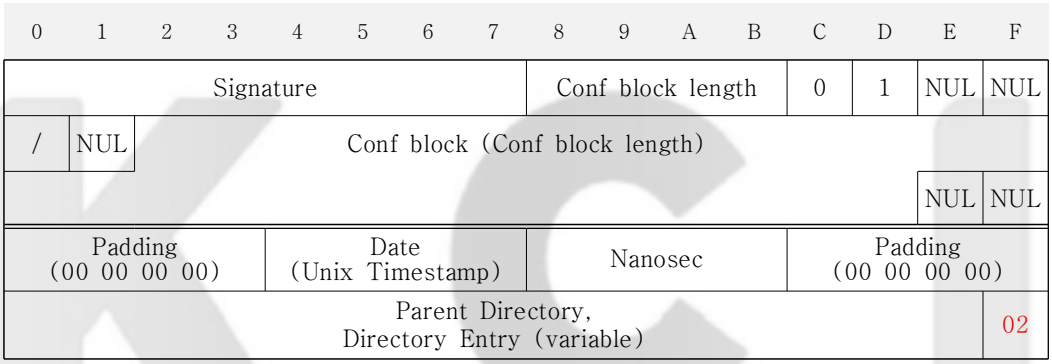
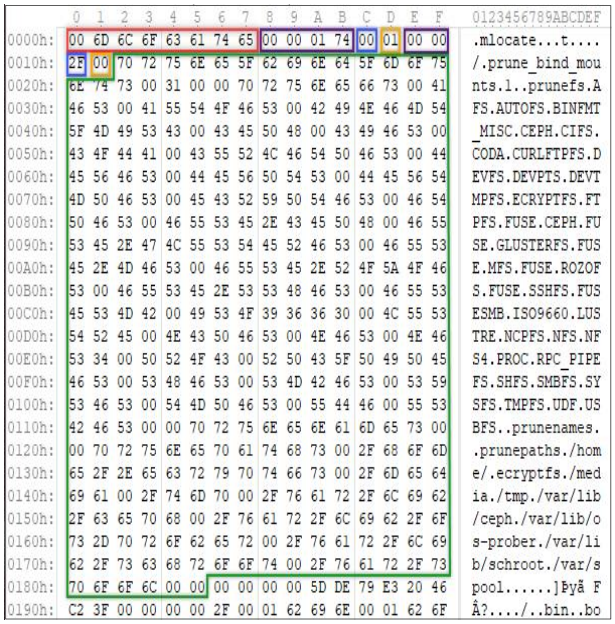


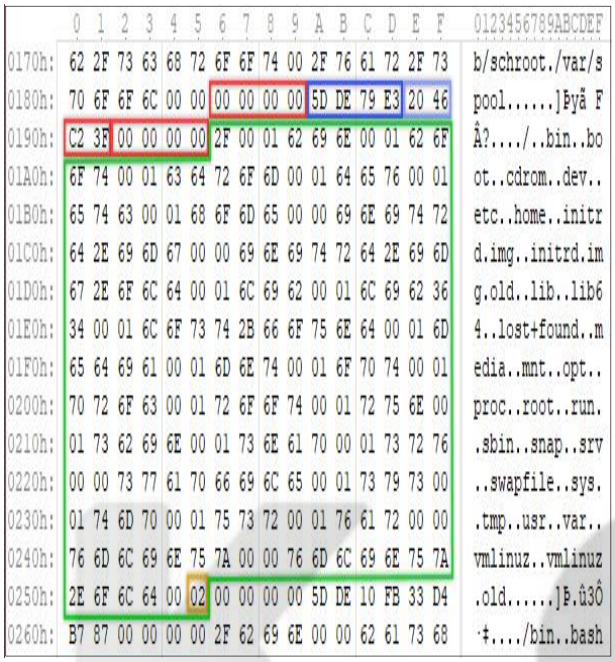
그림 1 mlocate.db 구조
Figure 1. mlocate.db Structure



Range	Size	Content
0x00 ~ 0x07	8	Signature(mlocate)
0x08 ~ 0x11	4	Configuration Block Length
0x12	1	File format version (0) Default 0
0x13	1	Visibility flag (0,1) Default 1
0x14 ~ 0x15	2	Padding
0x16	1	Root of the directory
0x17	1	NUL
0x18 ~	var	updatedb.conf file content
~ NUL NUL	2	End of header

그림 2. mlocate.db Header Structure
Figure 2. mlocate.db Header Structure

헤더에는 대표적으로 mlocate 시그니쳐와 데이터베이스 생성 시 참조한 환경설정 값을 저장되어 있다. mlocate.db 파일의 헤더는 [그림 2]와 같이 9개의 섹션으로 구분되어 저장된다. 첫 번째는 섹션은 0\mlocate 시그니쳐로 시작하고, 이후 두 번째 섹션은 데이터베이스 업데이트 시 참조하는 설정(updatedb.conf)블록의 길이 0x00000174 => 372)를 빅엔디안 형식으로 저장한다. 세 번째 섹션은 파일 포맷 버전이 저장된다. 네 번째 섹션은 require visibility 사용자에게 접근 권한이 없는 디렉터리의 검색 시 보여줄 것인지를 설정하는 옵션으로 기본적으로 보여주지 않는다. 다섯 번째 섹션은 패딩 0x0000이 저장된다. 이후 여섯 번째 섹션은 최상위 디렉터리 정보 "/"가 표시되고, 일곱 번째 섹션은 NUL값이 있고 난 뒤, 여덟 번째 섹션에는 데이터베이스 업데이트 시 참조하였던 updatedb.conf의 설정 내용이 저장된다. 마지막 아홉 번째는 NUL 문자로 종료한다.



Range	Size	Content
0x00 ~ 0x03	4	Padding
0x04 ~ 0x07	4	Unix TimeStamp
0x08 ~ 0x11	4	Nanosec
0x12 ~ 0x15	4	Padding
0x16 ~	var	Parent Directory (various)
	var	Type, Sub directory, File Name (various)
	1	End of Entry (0x02)

그림 3. 디렉터리 엔트리 구조
Figure 3. Directory Entry Structure

디렉터리 엔트리는 [그림 3]과 같이 7개의 섹션으로 구분되어 저장된다. 첫 번째 섹션은 4byte 패딩 0x00000000으로 시작하고, 두 번째 섹션은 디렉터리의 시간 정보가 빅엔디안 형식의 유닉스 타임스탬프로 저장된다. 세 번째 섹션은 시간 정보가 빅엔디안 형식의 나노 세컨드로 저장된다. 네 번째 섹션은 4byte 패딩 0x00000000, 이후에는 다섯 번째 섹션은 부모 디렉터리(Parent directory) 정보가 저장된다. 여섯 번째 섹션은 Type정보와 함께 디렉터리(Sub directory) 또는 파일명이 저장된다. 마지막으로 일곱 번째 섹션에서 0x02가 저장되어 있으면 해당 엔트리의 끝이다. 여기에서 디렉터리와 파일은 이름순으로 정렬되어 순서대로 저장된다. 두 번째 섹션의 유닉스 타임스탬프는 디렉터리가 마지막으로 수정된 시간을 저장한다. 디렉터리의 방문만으로는 타임스탬프가 변경되지는 않지만, 디렉터리에 저장된 파일을 열람하거나 수정, 삭제 시 타임스탬프가 변경된다. [그림 4]와 같이 두 번째 섹션의 4byte 값 0x5DDE79E3을 10진수로 변환한 값은 유닉스 타임스탬프값 1574861283 이고, 이를 DCode[6]를 사용하여 변환한 값은 2019. 11. 27. 22:28:03이다.

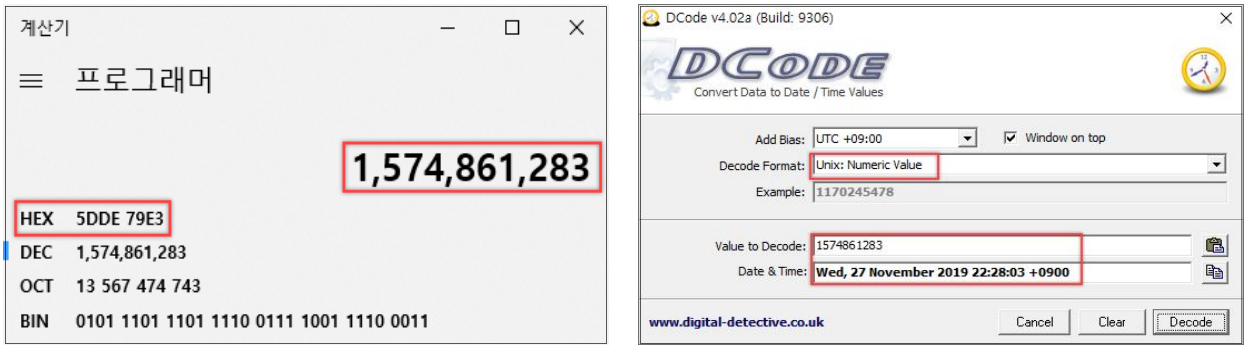


그림 4. 타임스탬프 변환
Figure 4. Timestamp Convert

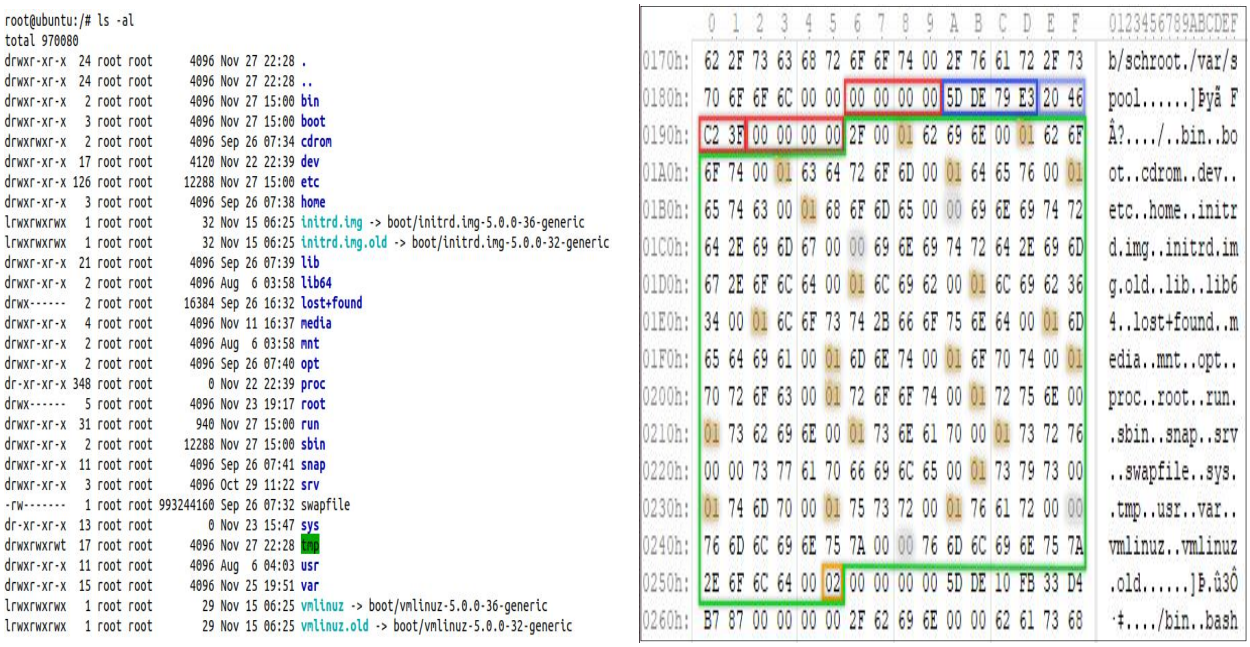


그림 5. 최상위 디렉터리 엔트리 구조
Figure 5. root directory entry structure

〔그림 5〕의 우측과 같이 최상위 디렉터리와 파일의 정보를 포함하고 있다. 0x00은 파일, 0x01은 디렉터리, 0x02는 해당 엔트리의 끝을 의미한다. 〔그림 5〕의 좌측과 같이 최상위 디렉터리 '/'는 22개의 디렉터리와 5개의 파일로 구성되어 있다.

표 4. 파일 타입 정보

Type	Size	Content
00	1	File
01	1	Directory
02	1	End of Entry

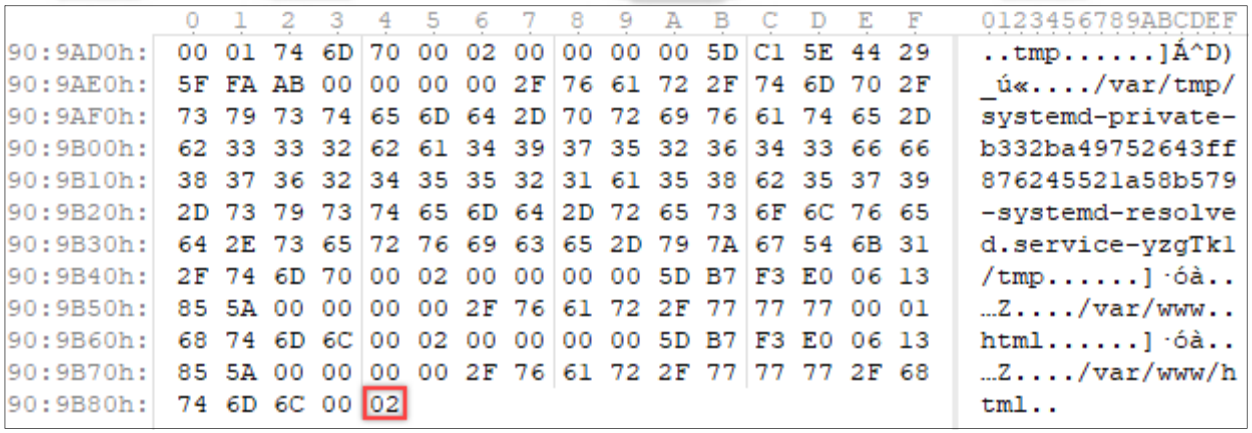


그림 6. mlocate.db 파일의 끝부분
Figure 6. mlocate.db file end

mlocate.db는 헤더(Header) 시그니처가 존재하지만 〔그림 6〕과 같이 풋터(Footer) 시그니처는 따로 존재하지 않고, 마지막 디렉터리 엔트리의 끝을 표시하는 0x02로 종결된다.

Ⅳ. mlocate.db 파일의 특성

4.1 mlocate.db 파일의 생성

리눅스 운영체제는 mlocate.db 데이터베이스를 최신상태로 유지하기 위해 〔그림 7〕과 같이 데이터베이스 업데이트 프

로그램인 /usr/bin/updatedb를 스케줄링 데몬인 crond에 의해 cron.daily에 등록하고 설정된 시간에 업데이트를 수행한다. 시스템의 부하를 증가시킬 수 있으므로 주로 사용량이 많지 않은 시간대에 실행된다.

```
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

그림 7. crontab 설정 화면
Figure 7. crontab view

mlocate.db 파일 업데이트 프로그램인 updatedb는 mlocate.db 파일생성에 필요한 환경설정 파일인 /etc/updatedb.conf를 참조하고 프로그램을 실행한다. 〔그림 8〕에는 업데이트하지 않는 확장자, 경로, 파일 시스템 등이 PRUNE으로 저장된다.

```
PRUNE_BIND_MOUNTS="yes"
# PRUNENAMES=".git .bzr .hg .svn"
PRUNEPATHS="/tmp /var/spool /media /var/lib/os-prober /var/lib/ceph /home/.ecryptfs /var/lib/schroot"
PRUNEFS="NFS nfs nfs4 rpc_pipefs afs binfmt_misc proc smbfs autofs iso9660 ncpfs coda devpts ftpfs devfs devtmpfs
 fuse.mfs shfs sysfs cifs lustre tmpfs usbfs udf fuse.glusterfs fuse.sshfs curlftpfs ceph fuse.ceph fuse.rozofs e
cryptfs fusesmb"
```

그림 8. updatedb.conf 화면
Figure 8. updatedb.conf view

4.2 mlocate.db 파일의 업데이트

updatedb는 데이터베이스와 파일 시스템의 디렉터리 최종 수정시간이 모두 일치하면 원래 데이터베이스를 사용하고, 일치하지 않으면 새로운 데이터베이스를 생성한다. updatedb가 디렉터리의 내용을 다시 생성할지를 업데이트 프로세스 중에 결정하기 위해 타임스탬프가 저장된 것이다.

```
root@ubuntu:/var/lib/mlocate# stat mlocate.db
  File: mlocate.db
  Size: 9512360         Blocks: 18584      IO Block: 4096   regular file
Device: 801h/2049d      Inode: 262185      Links: 1
Access: (0640/-rw-r-----)  Uid: (    0/    root)   Gid: (  109/ mlocate)
Access: 2019-11-29 00:05:36.291547602 +0900
Modify: 2019-11-29 00:05:41.507519506 +0900
Change: 2019-11-29 00:05:41.511519484 +0900
 Birth: -
```

그림 9. 업데이트 명령 실행 전 mlocate.db inode 값
Figure 9. mlocate.db inode value before executing update command

```
root@ubuntu:/var/lib/mlocate# updatedb
```

그림 10. updatedb 명령 실행
Figure 10. updatedb execute

```
root@ubuntu:/var/lib/mlocate# stat mlocate.db
  File: mlocate.db
  Size: 9512387         Blocks: 18584      IO Block: 4096   regular file
Device: 801h/2049d      Inode: 262363      Links: 1
Access: (0640/-rw-r-----)  Uid: (    0/    root)   Gid: (  109/ mlocate)
Access: 2019-11-29 07:43:53.396257639 +0900
Modify: 2019-11-29 07:43:53.564256638 +0900
Change: 2019-11-29 07:43:53.580256543 +0900
 Birth: -
```

그림 11. 업데이트 명령 실행 후 mlocate.db inode 값
Figure 11. mlocate.db inode value after running update command

〔그림 10〕과 같이 updatedb 명령 실행 후, mlocate.db의 inode 값이 〔그림 9〕와 같이 262185에서 〔그림 11〕과 같이 262363으로 변경되었다는 것은 새로운 inode가 생성되었음을 의미하고, 삭제된 inode에 할당되었던 파일은 미할당 영

역에 존재할 수 있으므로 복구 가능성이 열려 있다. 실제 운영되는 시스템에서 전체 디렉터리의 타임스탬프 변경 요소인 파일의 생성·수정·삭제·접근시간의 변경이 없을 가능성은 존재하기 어려우므로 타임스탬프가 변경되지 않고 그대로 유지되고 있을 가능성은 0에 가깝다. 데이터베이스 업데이트가 실행되면 기존 데이터베이스는 삭제되고, 새로운 데이터베이스가 생성된다고 봐도 무방하다. updatedb의 소스 코드(updatedb.c)상에도 unlink() 함수를 사용하여 파일을 삭제하고 새로 생성한다.

Ⅴ. mlocate.db 파일 활용 방안

mlocate.db는 리눅스 운영체제에서 파일과 디렉터리 목록을 주기적으로 데이테베이스에 저장한다. 데이터베이스 업데이트 시 파일을 삭제하고, 새로운 파일을 생성하기 때문에 데이터베이스 파일복구를 통해 과거 생성·수정·삭제·접근된 파일과 디렉터리 흔적을 확인할 수 있다. 여기서 삭제된 mlocate.db 파일을 효과적으로 수집하고 분석하는 방법을 제시하고자 한다.

5.1 현재 mlocate.db 분석

앞서 mlocate.db 파일의 구조에서 확인하였듯이 파일명, 디렉터리명 외에 타임스탬프(나노 세컨드 포함) 정보가 저장되어 있어 해당 디렉터리의 생성, 수정, 삭제, 접근 시점을 알 수 있다. 파일의 헤더 부분을 제외하고 디렉터리 엔트리 부분은 정규표현식 등을 사용하여 0x20으로 구분하고, 타임스탬프 정보와 디렉터리, 파일명 정보를 출력할 수 있다.

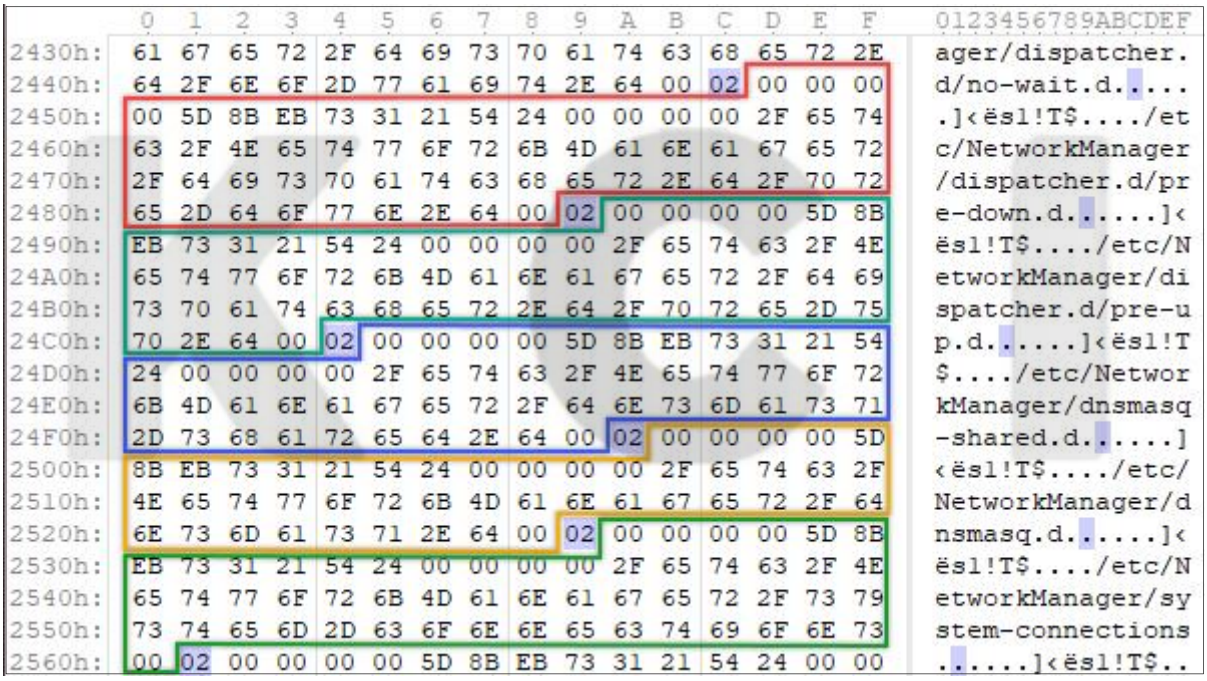


그림 12. 0x20으로 구분되는 디렉터리 엔트리
Figure 12. Directory Entry Separated by 0x20

5.2. 삭제된 mlocate.db 파일복구

앞서 〔표 1〕에서 언급한 리눅스 운영체제들 대부분은 mlocate.db 파일의 업데이트를 매일 수행하도록 설정되어 있다. 1년이라면 약 364개의 삭제된 mlocate.db 파일이 미할당 영역에 존재할 것으로 예상한다. 일부는 미할당 영역에서 덮어써 져 완전한 형태의 파일을 복구할 수 없을 수 있지만 그래도 많은 흔적이 있을 것이다.

본 논문에서는 mlocate.db 파일복구 실험을 위해 Ubuntu 18.04.3(64bit)〔7〕 버전을 VMware〔8〕 가상머신 환경에 ext4 파일 시스템에 설치하고, 오픈소스 디지털 포렌식 도구인 Sleuth kit 4.4.2-3〔9〕 버전을 사용하여 실험했다. 실험 과정은 먼저 바이너리 파일의 시그니쳐를 찾는 프로그램인 sigfind[1]를 사용하여 mlocate.db 파일의 시그니쳐 값(\0mlocate)에서 4byte 값 0x006D6C6F로 mlocate.db 파일이 저장되어 있던 /dev/sda1 파티션에 대해 조사하였다. 〔그림 13〕과 같이 시그니쳐를 포함하는 블록들이 다수 발견되었고, 시그니쳐가 검출된 블록들은 할당 상태와 미할당 상태가 존재했다. 미할당 상태의 블록을 조사한 결과, 전체 mlocate.db의 형태를 한 블록도 있었고, 다른 데이터들이 섞여 있는 블록도 존재하였다.

1) sigfind는 데이터에서 이진 서명을 찾는 데 사용된다. (4byte까지만 검색 가능)

```
root@ubuntu:~# sigfind -b 4096 006D6C6F /dev/sda1
Block size: 4096  Offset: 0  Signature: 6D6C6F
Block: 1148154 (-)
Block: 1149100 (+946)
Block: 1389454 (+240354)
Block: 1400832 (+11378)
Block: 1402498 (+1666)
```

그림 13. sigfind 실행 결과
Figure 13. execution of sigfind

〔그림 14〕와 같이 blkstat[2] 명령을 사용해 〔그림 13〕의 mlocate.db 시그니처가 확인된 블록들의 할당 상태를 확인하였다.

```
================================================
Block: 1400832
Fragment: 1400832
Not Allocated
Group: 42
0       006d6c6f 63617465 00000174 00010000     .mlo cate ...t ....
================================================
Block: 1402498
Fragment: 1402498
Allocated
Group: 42
0       006d6c6f 63617465 00000174 00010000     .mlo cate ...t ....
================================================
```

```
Block: 1148154
Fragment: 1148154
Not Allocated
Group: 35
0       006d6c6f 63617465 00000174 00010000     .mlo cate ...t ....
================================================
Block: 1149100
Fragment: 1149100
Not Allocated
Group: 35
0       006d6c6f 63617465 00000174 00010000     .mlo cate ...t ....
================================================
Block: 1389454
Fragment: 1389454
Not Allocated
Group: 42
0       006d6c6f 63617465 00000174 00010000     .mlo cate ...t ....
================================================
```

그림 14. sigfind 검색 결과 블록들의 할당 상태 확인
Figure 14. Assignment Status of Blocks

```
root@ubuntu:~# blkcat -h /dev/sda1 1402498
```

그림 15. sigfind 검색 결과 블록 추출
Figure 15. Extract sigfind search result blocks

미할당 영역에서 mlocate.db 시그니처가 발견된 블록에 대해 〔그림 15〕와 같이 blkcat[3] 명령을 사용하여 구문 분석이 가능한지 확인하였다.

```
0       006d6c6f 63617465 00000174 00010000     .mlo cate ...t ....
16      2f007072 756e655f 62696e64 5f6d6f75     /.pr une_ bind _mou
32      6e747300 31000070 72756e65 66730041     nts. 1..p rune fs.A
48      46530041 55544f46 53004249 4e464d54     FS.A UTOF S.BI NFMT
64      5f4d4953 43004345 50480043 49465300     _MIS C.CE PH.C IFS.
80      434f4441 00435552 4c465450 46530044     CODA .CUR LFTP FS.D
96      45564653 00444556 50545300 44455654     EVFS .DEV PTS. DEVT
112     4d504653 00454352 59505446 53004654     MPFS .ECR YPTF S.FT
128     50465300 46555345 2e434550 48004655     PFS. FUSE .CEP H.FU
144     53452e47 4c555354 45524653 00465553     SE.G LUST ERFS .FUS
160     452e4d46 53004655 53452e52 4f5a4f46     E.MF S.FU SE.R OZOF
176     53004655 53452e53 53484653 00465553     S.FU SE.S SHFS .FUS
192     45534d42 0049534f 39363630 004c5553     ESMB .ISO 9660 .LUS
208     54524500 4e435046 53004e46 53004e46     TRE. NCPF S.NF S.NF
224     53340050 524f4300 5250435f 50495045     S4.P ROC. RPC_ PIPE
240     46530053 48465300 534d4246 53005359     FS.S HFS. SMBF S.SY
256     53465300 544d5046 53005544 46005553     SFS. TMPF S.UD F.US
272     42465300 00707275 6e656e61 6d657300     BFS. .pru nena mes.
288     00707275 6e657061 74687300 2f686f6d     .pru nepa ths. /hom
304     652f2e65 63727970 74667300 2f6d6564     e/.e cryp tfs. /med
320     6961002f 746d7000 2f766172 2f6c6962     ia./ tmp. /var /lib
336     2f636570 68002f76 61722f6c 69622f6f     /cep h./v ar/l ib/o
352     732d7072 6f626572 002f7661 722f6c69     s-pr ober ./va r/li
368     622f7363 68726f6f 74002f76 61722f73     b/sc hroo t./v ar/s
384     706f6f6c 00000000 00005dcd c66424d4     pool .... ..]. .d$.
400     b5fd0000 00002f00 0162696e 0001626f     .... ../. .bin ..bo
```

그림 16. 헤더 정보가 있는 mlocate.db (allocated area)
Figure 16. mlocate.db with header information (allocated area)

〔그림 16〕과 같이 1402498 블록을 덤프한 파일을 16진수 형태로 보여주는 xxd[4]로 보면, 현재 컴퓨터에서 사용하고 있는 mlocate.db 임을 알 수 있다.

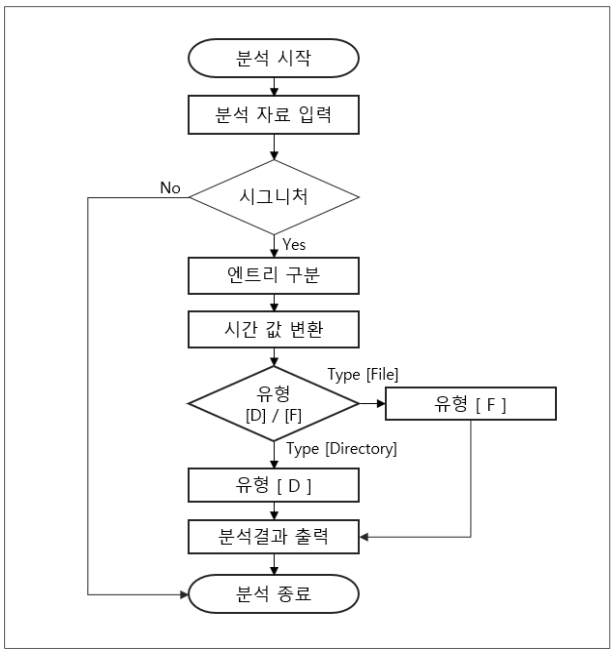
2) blkstat는 주어진 파일 시스템 데이터 단위의 할당 상태를 표시한다.
3) blkcat는 지정 블록의 데이터를 출력해 주는 도구로써 ASCII, hexdump, html 형태로 출력한다.
4) xxd는 주어진 파일이나 standard input으로 들어온 문자들에 대해서 hex dump를 만들어 준다.

그림 17. 미할당 영역(unallocated area)에서 발견된 mlocate.db 디렉터리 엔트리
Figure 17. mlocate.db directory entry found in unallocated area

그러나 실험을 통해 〔그림 17〕과 같이 mlocate.db 시그니처 4byte 값 0x006D6C6F를 포함하지 않는 블록도 다수 발견되었다. ext4 파일 시스템은 extents 구조체로부터 삭제된 파일의 데이터 블록 정보가 초기화되기 때문에〔10〕 mlocate.db를 발견하기는 쉽지 않은 일이었다. 포렌식 관점에서 보면 시그니처가 있는 mlocate.db 파일의 헤더 정보보다는 각 디렉터리 엔트리의 타임스탬프와 디렉터리, 파일명이 더 중요하고, 각 디렉터리 엔트리는 0x00000000으로 시작해서 0x02로 끝나기 때문에 전체 디스크 파일 시스템 각 블록에 대해 정규표현식을 이용하여 카빙해 내면 타임스탬프 정보와 디렉터리 엔트리의 정보를 확보할 수 있다.

5.3. mlocate.db 분석 도구

현재까지 분석된 mlocate.db 파일 구조를 바탕으로 리눅스 환경에서 실행할 수 있는 분석 도구를 개발하였다. 본 도구는 할당 상태의 정상적인 파일의 구문분석 기능과 미할당 영역의 디스크 블록 파일로부터 구문 분석을 수행한다. 〔그림 18〕과 같이 출력은 디렉터리 최종 수정시간을 표시하고, 디렉터리는 〔D〕, 파일은 〔F〕로 표시한다.



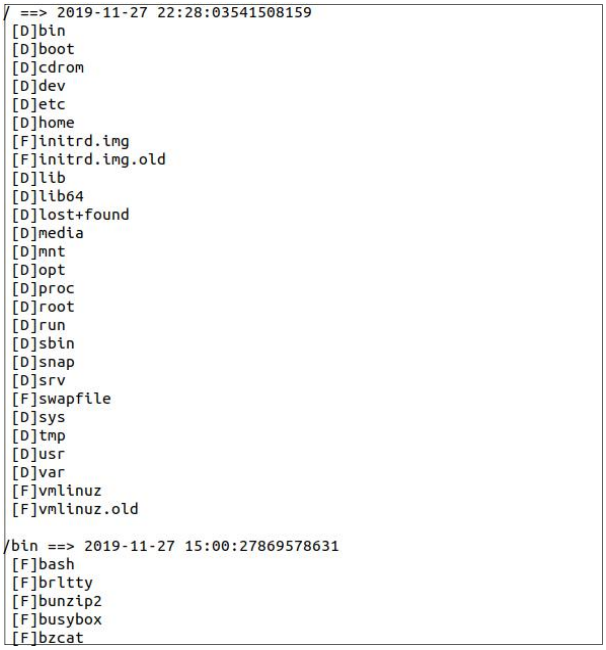


그림 18. mlocate.db viewer 동작 개요도와 실행 화면
Figure 18. mlocate.db viewer operation overview and execution screen

5.4. mlocate.db 분석 도구 활용 사례

산업기술 유출 사고를 겪은 기업 10곳 중 4곳 가량은 사고 발생 후 3개월을 넘어서야 기술 유출을 감지하는 것으로 조사됐다. 특히 기술 유출 후 1년 이상이 지난 한참 뒤에야 보안사고를 감지하는 기업은 17.7%에 달하는 것으로 나타났다. 1년 이내~6개월 이상은 14.7%, 6개월 이내~수일 이상은 5.9%였다. 수일 이내에 보안사고를 감지하는 경우는 26.5%였다[11].

본 사례는 범행 발생 약 6개월이 지난 후에 기술 유출이 발견되었으나, 미할당 영역 잔류정보에서 발견된 mlocate.db 디렉터리 엔트리 흔적을 복원하고, 범죄 혐의를 입증할 수 있었던 사건으로써, 연구의 결과를 실제 사건에 적용한 사례이다.

피해 회사에서 소프트웨어 개발을 담당하던 직원이 경쟁사로 이직하면서 영업비밀 자료를 무단 반출한 사건으로 이미 대상자가 퇴사한 지 6개월가량 지난 후에 경쟁사에서 복제한 제품을 출시한 것을 확인하고서야 인지하였다. 전형적인 퇴사자에 의한 영업비밀 유출 사건이었다. 통상적으로 제품을 출시했다는 것은 개발이 완료되었다는 것을 의미하므로, 유출한 영업비밀 자료는 이미 삭제하거나 은닉했을 가능성이 커 증거를 발견하기 어려운 상황이었다.

압수·수색 현장에서 피해 회사에서 유출한 소스 코드와 동일한 파일은 발견되지 않았다. 이미 소스 코드의 변경작업을 완료한 후였기 때문에 삭제했거나 은닉한 것이었다. 리눅스 서버 전체 저장매체 영역을 키워드 검색한 결과, 고소인 회사의 소스 코드와 동일한 파일명이 미할당 영역 잔류정보에서 발견되었고, mlocate.db 디렉터리 엔트리 구조 분석을 통해 피해 회사의 소스 코드와 동일한 파일임을 확인할 수 있었다. 이번 연구를 통해 개발한 도구[5]를 사용하여 유출한 자료가 저장된 시점까지 특정할 수 있는 증거를 확보할 수 있었다.

Ⅵ. 결 론

리눅스 운영체제는 사용자의 행위를 특정할 수 있는 아티팩트가 다양하지 않아 분석에 어려움을 겪는 경우가 많다. 분석 대상이 리눅스 운영체제일 때 통상적으로 사건이 발생한 시점 전, 후의 로그 기록을 분석하고, 이후 파일 시스템 전체 영역을 전문 포렌식 도구[12][13]을 이용하여 키워드 검색을 하고, 검색된 자료들을 분석하게 된다. 키워드 검색에서 매칭이 되는 자료를 찾는다고 하더라도 시간 정보가 없고, 인과관계가 증명되지 않는다면 증거로써 사용하기에는 충분하지 않다.

본 논문에서는 리눅스 운영체제에서 사용하는 파일검색 유틸리티인 locate(mlocate)의 데이터베이스 mlocate.db 파일의 포맷 분석 및 활용방안에 관한 연구를 하였다. mlocate.db에는 시스템에 존재했던 디렉터리와 파일명을 비롯하여 시간 정보를 저장하고 있어 사용자가 언제 디렉터리나 파일을 생성·수정·삭제·접근했는지 알 수 있으므로 디지털 포렌식 관점에서 사용자의 행위를 분석할 수 있고, 안티포렌식 행위가 언제 있었는지 특정할 수 있는 유용한 아티팩트란 사실을 확인했다.

mlocate.db 파일의 업데이트는 스케쥴러에 의해 최소 하루에 한 번은 실행되고, 기존 데이터베이스 파일을 삭제하고 새로운 파일을 생성하는 방식을 취하고 있어 미할당 영역 잔류정보에 분포하고 있는 mlocate.db 파일의 복구를 통해 과거 시점의 디렉터리 구조를 확인할 수 있다. 그러나 리눅스 운영체제에서 주로 사용하는 ext4 파일 시스템은 extents 구조체로부터 삭제된 파일의 데이터 블록 정보가 초기화되기 때문에 전체 파일을 복원하기 어렵고 이와 관련한 연구는 계속 진행중이다[2]. 미할당 영역에서 완전한 mlocate.db파일의 구조를 지닌 파일을 복구하지 못한다 해도 mlocate.db 파일의 구성요소인 디렉터리 엔트리를 포함하고 있는 블록들이 존재하면 분석할 수 있다.

이번 연구결과를 바탕으로 분석 도구를 개발하여 디지털 포렌식 업무에 적용할 수 있도록 제안하였고, 사례를 통해 개발 도구를 실무에 적용하고 효용성을 검증하였다.

5) https://github.com/mlocate1212/mlocate-viewer

참 고 문 헌 (References)

〔1〕 The Digital Times, Linux beyond the Windows stronghold...IoT market's 'Great' 2017. 7. 3 http://www.dt.co.kr/contents.html?article_no=2017070402101560041001

〔2〕 Jonathan Corbet, LWN.net, Greg Kroah-Hartman, The Linux Foundation "Linux Kernel Development Report" 2017. ,https://www.linuxfoundation.org/projects/linux/

〔3〕 Information Resource Policy and press release of the Ministry of Public Administration and Security, "Promoting the introduction of open OS, signaling a new change in the government's PC environment." 2019.05.15.,https://www.mois.go.kr/frt/bbs/type010/commonSelectBoardArticle.do?bbsId=BBSMSTR_000000000008&nttId=70679

〔4〕 pkgs.org, "Packages Search", https://pkgs.org/

〔5〕 Chris Binnie, "Finding Files With Mlocate" 2017. 11. , https://www.linux.com/tutorials/finding-files-mlocate/

〔6〕 Digital Detective, https://www.digital-detective.net/dcode/

〔7〕 Canonical Ltd, https://ubuntu.com/

〔8〕 VMware, Inc, https://www.vmware.com/

〔9〕 Brian Carrier, https://www.sleuthkit.org/

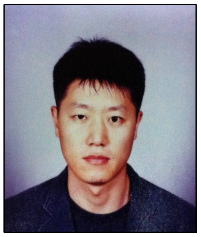
〔10〕 Kevin D. Fairbanks, "An analysis of Ext4 for digital forensics", Digital Investigation, Vol. 9, pp. 118-130, 2012.

〔11〕 Two out of 10 companies that have been involved in technology leaks are dark even after a year,http://www.sisajournal-e.com/news/articleView.html?idxno=171199

〔12〕 Guidance Software, http://www.guidancesoftware.com.

〔13〕 X-Ways Software Technology AG, http://www.x-ways.net/forensics/

저 자 소 개



김 재 청 (Jaecheong Kim)

1998년 2월 : 강남대학교 전자공학과 졸업
2013년 2월 ~ 현재 : 서울지방경찰청 국제범죄수사대 산업기술보호수사팀
2018년 9월 ~ 현재 : 고려대학교 정보보호대학원 석사과정
관심분야 : 디지털 포렌식, 산업보안, 정보보호

이 상 진 (Sangjin Lee)

1989년 10월 ~ 1999년 2월 : ETRI 선임 연구원
1999년 3월 ~ 2001년 8월 : 고려대학교 자연과학대학 조교수
2001년 9월 ~ 현재 : 고려대학교 정보보호대학원 교수
2008년 3월 ~ 현재 : 고려대학교 디지털포렌식연구센터 센터장
관심분야 : 디지털 포렌식, 심층암호, 해시함수