Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.1, Feb. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.1.123

MongoDB에 대한 디지털 포렌식 조사 기법 연구*

윤 종 성,[1†] 정 두 원,[1] 강 철 훈,[2] 이 상 진[1‡]
[1]고려대학교 정보보호대학원, [2]대검찰청 디지털수사담당관실

Digital Forensic Investigation of MongoDB*

Jong-Seong Yoon,[1†] Doo-Won Jung,[1] Chul-hoon Kang,[2] Sang-Jin Lee[1‡]
[1]Center for Information Security Technologies(CIST), Korea University,
[2]Digital Forensic Center, Supreme Prosecutors' Office

요 약

최근 데이터의 대용량화로 인해 관계형 데이터베이스 관리 시스템(RDBMS)과 빅데이터 처리를 위한 NoSQL DBMS에 대한 수요가 꾸준히 늘고 있다. 관계형 DBMS에 대한 디지털 포렌식 조사 기법은 활발히 연구되어 왔으나 최근 사용이 급증하고 있는 NoSQL DBMS에 대한 포렌식 조사 기법에 대한 연구는 거의 없는 실정이다. 본 논문에서는 NoSQL DBMS 중 가장 많이 사용되고 있는 MongoDB에 대한 디지털 포렌식 조사 절차와 기법을 제안한다.

ABSTRACT

As the data gets bigger recently, the demand for relational database management system (RDBMS) and NoSQL DBMS to process big data has been increased consistently. The digital forensic investigation method for RDBMS has been studied actively, but that for NoSQL DBMS, which is popularly used nowadays, has almost no research. This paper proposes the digital forensic investigation process and method for MongoDB, the most popularly used among NoSQL DBMS.

Keywords: MongoDB, NoSQL Database Forensics, Database Forensics, Digital Forensics

I. 서 론

스마트폰, 스마트패드, 노트북 등 휴대성이 뛰어난 전자기기들이 보편화되면서 사용자들은 소셜네트워크에 쉽게 접근할 수 있게 되었으며, 더불어 영화, TV 컨텐츠, 음악 등 다수의 대용량 멀티미디어 컨텐츠를 쉽게 업로드 및 다운로드할 수 있게 되었다. 그 결과 서버 시스템에서 관리해야 하는 데이터의 양은 기하급수적으로 증가하게 되었고, 발생되는 로그의 크기도 하루에 수백 기가바이트에서 많게는 수십 테라바이트를 넘어서게 되었다. 이에 폭발적으로 증가하는 데이터를 효율적으로 저장하고 빠르게 처리하여 가치 있는 정보를 도출할 수 있는 새로운 기술들이 등장하고 있다.

빅데이터 기술은 데이터의 크기가 대용량이고, 빠르게 크기가 증가하며 데이터 형태가 비정형인 대규모의 데이터를 저비용으로 빠르게 수집, 저장, 분석하여 가치 있는 정보를 추출하기 위해 등장한 기술이다. 빅데이터 처리를 위한 소프트웨어는 대표적으로 데이터 저장 소프트웨어인 NoSQL DBMS, 분산 처리, 분산 파일시스템인 Hadoop, 분석・시각화 소프트웨어인 R 등이 있다[1].

본 논문에서는 빅데이터 저장기술인 NoSQL DBMS 중 가장 많이 사용되고 있는 MongoDB에

접수일(2013년 10월 30일), 수정일(2014년 1월 9일), 게재확정일(2014년 2월 5일)
* 이 논문은 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단-공공복지안전사업의 지원을 받아 수행된 연구임(2012M3A2A1051106)
† 주저자, yoonjs53@korea.ac.kr
‡ 교신저자, sangjin@korea.ac.kr (Corresponding author)

대한 디지털 포렌식 조사 절차와 기법을 제안하고자 한다.

II. 관련연구

현재 Oracle, MySQL, MSSQL 등 관계형 DBMS에 대한 디지털 포렌식 조사 기법에 대한 연구는 조사 절차 및 방법, 아티팩트, 삭제된 데이터 복구 등 여러 방면에서 활발히 이루어지고 있다.

Khanuja 등[2]은 오픈소스 관계형 DBMS인 Mysql에 대한 내부 동작 방식 및 구조 분석, 아티팩트, 분석 도구 등을 설명하고 포렌식 프레임워크를 제시하였으며 Fruhwirt 등[3]은 Mysql 데이터 파일 및 레코드에 대한 세부 포맷에 대해 연구하였다. Paul M. Wright[4]은 Oracle DBMS의 RedoLog 파일을 분석하여 Oracle DBMS의 변경내역을 추적하는 도구를 개발하여 공개하였고 Pavlou 등[5]은 관계형 DBMS의 데이터 변조를 탐지할 수 있는 알고리즘을 제시하였다. 또한 Jonghyun Choi 등[6]은 Oracle DBMS에서 삭제된 레코드를 복구하는 기법을 제안하였다.

NoSQL DBMS은 관계형 DBMS와 개념 및 구조가 상이하므로 기존 관계형 DBMS 포렌식 연구를 그대로 적용하기 어렵다. 전체적인 흐름은 유사할 수 있으나 데이터베이스에 스키마가 없고 다수의 물리적인 서버에서 운영된다는 NoSQL의 특징을 반영한 포렌식 조사 방법이 필요하다. 관계형 DBMS와 대조적으로 NoSQL DBMS에 대한 디지털 포렌식 관점의 연구는 거의 없는 실정이다. NoSQL에 대한 관련 연구는 기존 관계형 DBMS와의 차이점, NoSQL DBMS의 종류와 특징, 성능 분석, 새로운 분야에 적용 방법 등이 주를 이루고 있다[7][8].

대량의 데이터를 저장하기 위해 NoSQL DBMS의 사용이 점점 늘어나고 있으며, 이에 따라 NoSQL DBMS에 대한 디지털 포렌식 연구의 필요성이 증가하고 있다. 본 논문에서는 대표적인 NoSQL DBMS 중 하나인 MongoDB에 대한 포렌식 조사 절차와 기법을 제안한다. 3장에서는 MongoDB에 대한 소개를, 4장에서는 MongoDB 포렌식 조사를 위한 내부 운영 구조, 데이터 포맷 등을 설명하고 MongoDB에 대한 포렌식 조사 절차 및 기법에 대해 제시한다.

III. MongoDB 소개

MongoDB는 10gen이라는 미국 회사에서 개발한 오픈소스 DBMS이며, NoSQL 중 가장 많이 활용되는 Document DBMS이다[9]. DBMS의 순위를 평가하는 DB-ENGINES[10]에 따르면 MongoDB는 전체 DBMS중 6위로(1~5위는 Oracle, Mysql, MSSQL, PostgreSQL, DB2로 관계형 DBMS임) NoSQL DBMS 중에는 가장 높은 순위를 유지하고 있다. 위치기반 SNS인 Foursquare, 음악 채널인 MTV, 미국의 경제지인 Forbes 등에서 대량의 트랜잭션을 처리하기 위해 사용되고 있다. 주 업무 뿐만 아니라 각 기업의 사내 시스템에도 MongoDB 사용이 늘어나고 있으며 서버로그, 웹로그 등 로그 분석·처리 시스템을 MongoDB 기반으로 구축하는 사례도 늘어나고 있다.

MongoDB는 대규모 데이터를 분산, 병렬 처리할 수 있고 하드웨어를 수평적으로 확장하는데 용이하다는 특징이 있다. 또한 기존 관계형 DBMS에서 SQL 쿼리로 할 수 있는 대부분의 기능을 지원하면서 Document 기반의 유연한 데이터모델을 지원하고 있다.

MongoDB를 Document DBMS라고 하는 이유는 정형화된 스키마가 없고 각 레코드는 JSON 문서 형태를 가지기 때문이다.

```
{
    "Name": "Kim",
    "address": {
    "streetAddress": "1st Street",
    "city": "Seoul",
    "postalCode": 1112
    },
    "phoneNumbers": [
    "111-222-333",
    "555-666-777"
    ]
}
```

Fig.1. An Example of JSON Document

JSON(JavaScript Object notation)은 자바스크립트 문법을 기반으로 데이터를 교환하기 위해 만들어진 개방형 표준이다[11]. JSON 포맷에서 중괄호 "{,}"는 하나의 Object를 표현하고, Object 안에는 Key : Value 형태로 표현된다. Value에는 다시 Object 형태가 들어갈 수도 있으며, 대괄호 "[,]"를

사용하여 배열 형태를 Value로 지정할 수도 있다. Fig. 1.은 개인 정보를 JSON 형태로 표현한 예이다. MongoDB는 사전에 정의된 스키마와 관계없이 임의의 JSON Document 형태로 레코드를 DB에 삽입할 수 있어, 데이터를 모델링하고 모델을 변경하기 용이하다.

MongoDB의 운영 형태는 크게 단일구성, 복제셋(Replica Set) 구성, 샤딩(Sharding) 구성으로 나눌 수 있다. 단일구성은 한 개의 mongod 프로세스를 운영하는 것으로 실제 서비스에는 거의 사용되지 않으나 개발, 연구 등의 용도로 사용된다.

복제셋 구성은 여러 개의 mongod 프로세스 또는 서버로 다중화한 구성으로 Primary와 Secondary 간에 자동으로 데이터 동기화가 이루어지며 장애가 발생할 경우에 자동 Fail Over를 통해 서비스 지속성을 보장한다. 복제셋 구성요소 중 Arbiter는 데이터는 저장하지 않으나 장애 시 Primary를 재지정하는 과정에 사용된다. Fig. 2.는 MongoDB의 복제셋 구성 예를 보여주고 있으며 mongod 구동시 --replSet 옵션으로 복제셋을 구성할 수 있다.

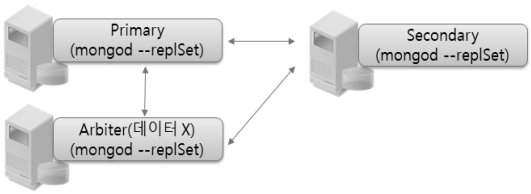

Fig.2. MongoDB Replication

샤딩 구성은 데이터를 분산처리하기 위한 구성이며 MongoDB를 샤딩 형태로 구성하면 데이터를 청크라는 단위로 분할하여 여러 서버에 나누어서 저장한다. 또한 데이터 크기에 따라 자동으로 서버별로 데이터를 균등하게 이동하여 성능을 높여준다. Fig. 3.은 MongoDB의 샤딩 구성을 보여준다. 샤딩 구성에서는 샤드라는 단위로 서버가 나누어지며 샤드는 다수로 구성될 수 있다. 분산된 데이터에 대한 일관된 접근을 위해 mongos라는 프로세스를 통해 접속해야 하며 설정서버(Config Server)에는 각 샤드의 현재 상태와 메타데이터가 저장된다[12].

MongoDB는 관계형 DBMS에서 사용하는 용어와 유사하지만 대응되는 용어 중에 일부 다른 것이 있으므로 MongoDB 서버를 조사할 경우 사전에 인지하고 있어야 한다. Table 1.은 관계형 DBMS와

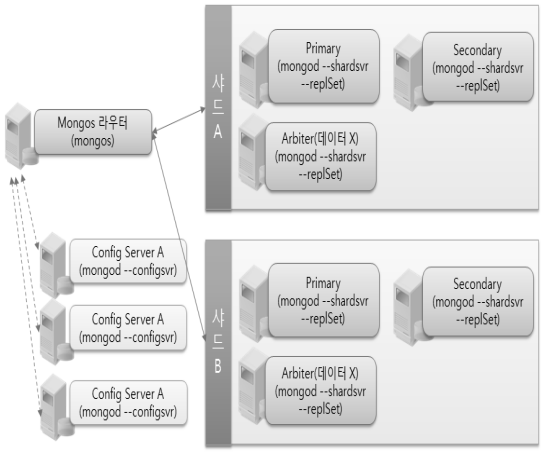

Fig.3. MongoDB Sharding

MongoDB 간의 용어 차이를 보여준다. 관계형 DBMS의 Table과 대응되는 용어는 Collection이고, Column이나 Field와 대응되는 용어는 Key이다.

Table 1. The term difference between RDBMS and MongoDB

관계형 DBMS	MongoDB
DB/Schema	DB
Table	Collection
Row/Tuple/Record	Record/Document
Column/Field	Key

IV. MongoDB 디지털 포렌식 조사 기법

MongoDB 서버를 대상으로 디지털 포렌식 조사를 할 경우에는 대상 데이터가 대용량이고 다수 서버에 나누어져 있으며 스키마가 없는 형태임을 고려해야 한다. 대용량의 데이터에서 증거를 수집하기 위해서는 DB 구조와 데이터의 위치를 파악하여 선별적으로 데이터를 수집하는 것이 효율적이다. 또한, 수집한 데이터를 직접 분석하기 보다는 분석을 위한 MongoDB 서버 환경을 미리 구축한 후, 수집한 데이터를 분석용 MongoDB에 입력하여 분석하는 것이 효과적이다.

본 논문에서는 MongoDB의 특성을 반영하여 Fig. 4.와 같이 MongoDB 디지털 포렌식 조사 절차를 제안한다.

MongoDB가 다양한 환경으로 운영될 수 있기 때문에 본격적인 포렌식 조사 전에 운영환경과 구조를

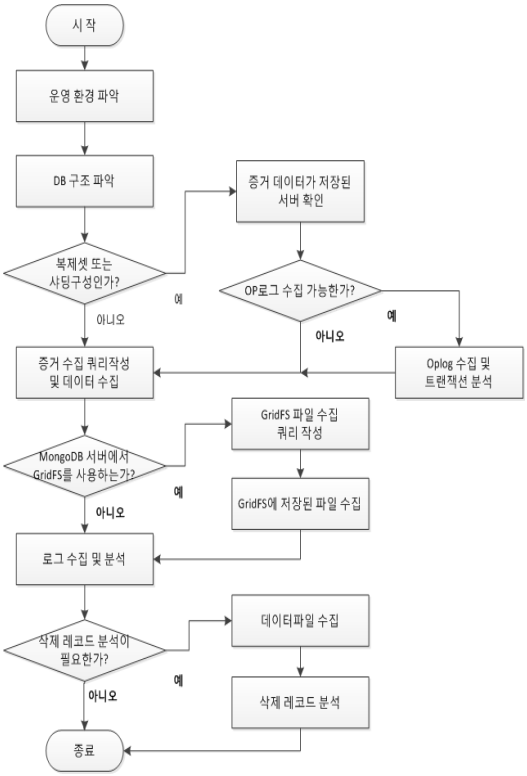

Fig.4. The procedure for MongoDB forensic investigation

파악하는 것이 중요하다. 구동옵션 등을 통해서 서버의 운영형태를 확인하고 DB의 구조를 파악하여 조사 대상 증거를 효과적으로 분석할 수 있도록 사전 정보를 수집한다. MongoDB 서버가 다수 서버에서 운영되는 복제셋 또는 샤딩 구성일 경우 OP로그를 수집하여 최근 DB의 변화와 트랜잭션을 세부적으로 분석할 수 있다.

기본 정보를 파악하였다면 증거를 수집하기 위해 중요 키워드를 포함한 쿼리를 작성하고 실행하여 증거 데이터를 수집한다. 이때 MongoDB 서버에서 GridFS 기능을 사용한다면 MongoDB에 저장되어 있는 파일 목록에서 사건과 관련된 파일을 검색하고 수집할 수 있다.

마지막으로 증거 데이터를 찾기 어렵거나 인위적으로 삭제된 흔적이 있다면 데이터파일을 수집하여 삭제된 레코드를 분석한다. 이때 OP로그가 확보 되었다면 세부적인 분석이 가능하다. 일부 절차는 사건에 따라 생략할 수 있다.

4.1 운영 환경 파악

4.1.1 구동옵션 확인

MongoDB는 MongoDB 서버 프로세스인 mongod를 구동할 때 적용하는 옵션에 따라 데이터, 로그파일의 위치, 운영 방식 등이 결정되므로 본격적인 조사에 앞서 구동 옵션을 파악해야 한다. 구동 옵션 중에 포렌식 관점에서 확인해야 하는 옵션은 데이터와 로그파일 위치, 복제셋 및 샤드서버 여부 등이다. Table 2.는 MongoDB의 주요 구동옵션을 나타낸다[13].

Table 2. The main executing options of MongoDB

옵 션	설 명
--dbpath "경로"	데이터 파일 위치 (Default : /data/db)
--port 포트번호	서비스 포트 (Default : 27017)
--logpath "경로"	로그파일 위치 (Default : 화면출력)
--logappend	서버 재구동시 로그파일을 덮어쓰지 않고 이어서 씀
--replSet "복제셋 이름"	복제셋 서버로 동작
--shrdsvr	샤드 서버로 동작
--configsvr	설정서버로 동작

구동옵션을 확인하는 방법에는 MongoDB 서버의 local DB에서 startup_log Collection을 확인하는 방법과 OS에서 프로세스 리스트를 확인하는 방법이 있다. Fig. 5.는 startup_log Collection에서 구동옵션을 확인하는 방법을 보여준다.

```
shard-a:PRIMARY> use local
switched to db local
shard-a:PRIMARY> db.startup_log.findOne({},{cmdLine:1})
{
        "_id" : "ServerA-1372812799893",
        "cmdLine" : {
                "dbpath" : "/data/shard-a-p",
                "logappend" : true,
                "logpath" : "/data/shard-a-p.log",
                "port" : 30000,
                "replSet" : "shard-a",
                "shardsvr" : true
        }
}
```

Fig.5. The executing options in startup_log

4.1.2 복제셋 구성 확인

MongoDB의 복제셋 구성은 rs.status() 명령으

로 확인한다. Fig. 6.은 rs.status() 명령으로 복제셋 구성을 확인한 결과인데 "members" 키에 복제셋을 구성하는 서버의 정보가 배열형태로 저장되어 있음을 알 수 있다. 또한 "name" 키에서 서버의 호스트명 또는 IP 주소와 포트를 확인할 수 있으며, 각 서버의 상태와 시작시간 등을 확인할 수 있다.


```
"set" : "shard-a",
"date" : ISODate("2013-10-18T00:45:45Z"),
"myState" : 1,
"members" : [
        {
                "_id" : 0,
                "name" : "ServerA:30000",
                "health" : 1,
                "state" : 1,
                "stateStr" : "PRIMARY",
                "uptime" : 1628,
                "optime" : {
                        "t" : 1375942989,
                        "i" : 1
                },
                "optimeDate" : ISODate("2013-08-08T06:23:09Z"),
                "self" : true
        },
        {
                "_id" : 1,
                "name" : "ServerC:30001",
                "health" : 1,
                "state" : 2,
                "stateStr" : "SECONDARY",
                "uptime" : 1382,
                "optime" : {
                        "t" : 1375942989,
                        "i" : 1
                },
                "optimeDate" : ISODate("2013-08-08T06:23:09Z"),
                "lastHeartbeat" : ISODate("2013-10-18T00:45:44Z"),
                "lastHeartbeatRecv" : ISODate("2013-10-18T00:45:44Z"),
                "pingMs" : 0,
                "syncingTo" : "ServerA:30000"
        },
        {
                "_id" : 2,
                "name" : "ServerB:30002",
                "health" : 1,
                "state" : 7,
                "stateStr" : "ARBITER",
                "uptime" : 1624,
                "lastHeartbeat" : ISODate("2013-10-18T00:45:44Z"),
                "lastHeartbeatRecv" : ISODate("2013-10-18T00:45:44Z"),
                "pingMs" : 0
        }
]
```


Fig.6. Checking the configuration of Replica Set by rs.status() command

4.1.3 샤딩 구성 확인

MongoDB가 샤딩형태로 운영될 경우 mogos를 통해 접속하게 된다. config DB의 shards Collection을 조회하면 샤드의 이름과 샤드 멤버의 호스트명 또는 IP 주소와 포트를 확인할 수 있다. Fig. 7.은 config DB에서 shards Collection을 조회하여 샤딩 구성을 확인한 결과이다.

```
mongos> use config
switched to db config
mongos> db.shards.find()
{ "_id" : "shard-a", "host" : "shard-a/ServerA:30000,ServerC:30001" }
{ "_id" : "shard-b", "host" : "shard-b/ServerB:30100,ServerC:30101" }
```

Fig.7. The configuration of sharding in shards Collection

4.2 DB 구조 파악

증거 데이터 수집을 위해서는 먼저 DB의 구조와 원하는 데이터가 포함된 Collection을 찾아야 한다. MongoDB에 존재하는 DB의 목록은 Fig. 8.과 같이 show dbs 명령으로 확인한다. DB 이름과 데이터파일의 용량을 확인할 수 있다. 특정 DB의 Collection 목록은 show collections 명령으로 확인한다.

```
mongos> show dbs
admin   (empty)
cloud   0.203125GB
cloud-docs      9.9033203125GB
config  0.046875GB
onpart  0.203125GB
test    0.203125GB
yjstest 0.203125GB
```

Fig.8. Show dbs command

MongoDB는 미리 정해진 스키마가 없기 때문에 Collection에 임의의 형태로 키와 값을 입력할 수 있다. 특히 값에 또 다른 JSON 형태의 레코드가 들어갈 수 있어 중첩된 형태로 키, 값 쌍이 저장될 수 있다. 따라서 Collection에 포함된 모든 키를 확인하기 어렵다. 다만 응용프로그램에서 데이터를 저장할 때는 일정한 형태를 가지기 때문에 Collection의 일부 Record를 출력하면 주요 키를 확인할 수 있다. Collection에서 일부 Record를 출력할 때는 db.[Collection 이름].find().limit([출력할 Record 개수]) 질의를 사용하여 JSON Document의 형태와 키를 확인한다. 그러나 이 방법은 전체 키를 확인할 수 없기 때문에 제한적이다.

다른 방법으로는 오픈소스 MongoDB 스키마 분석도구인 variety.js[14]를 사용하는 방법이 있다. variety.js 파일을 mongo 실행파일이 있는 위치에 저장한 후 다음과 같은 명령으로 스키마를 파악할 수 있다.

```
mongo 서버IP:포트/DB명 --eval "var collection = 'Collection 이름', limit = 숫자, maxDepth = 숫자" variety.js
```

여기서 limit 옵션은 스키마 분석에 사용할 레코드의 개수이고 maxDepth 옵션은 중첩된 하위키의 분

석 깊이를 지정한다. 옵션을 생략하면 기본값으로 limit 옵션은 69,837, maxDepth는 99 값을 갖는다.

조금 더 간단하게 DB 구조를 파악하고 Collections의 키를 확인하는 방법은 상용 또는 무료 MongoDB 클라이언트 S/W를 사용하는 방법이다. Fig. 9.는 무료 MongoDB 클라이언트 S/W인 Tadpole DB Hub에서 DB 구조 및 Record를 확인하는 화면이다[15].

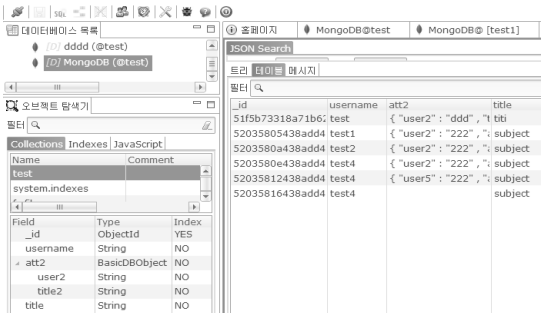

Fig.9. Tadpole DB Hub tool

4.3 증거 데이터 수집

4.3.1 데이터 전체 수집

MongoDB의 데이터 파일 경로에 있는 데이터 파일을 복사하여 사본을 생성하면 분석 시스템에서 별도의 mongod 프로세스를 구동하여 분석할 수 있다. 복사한 데이터 파일을 특정 경로에 두고 mongod 구동 시 --dbpath 옵션 뒤에 데이터 파일 사본 경로를 지정해주면 MongoDB 명령으로 데이터를 분석할 수 있다. 특정 DB만 필요하다면 DB명으로 시작하는 파일만 복사하여 사본을 생성하면 해당 DB만 수집할 수 있다. 그러나 샤딩 구성으로 운영 중인 시스템의 경우 여러 대의 서버에서 물리적으로 파일을 수집하는데 한계가 있다. 따라서 삭제된 레코드를 복원해야 할 필요성이 없을 경우에는 데이터 파일 자체를 수집하는 것은 비효율적이다.

4.3.2 데이터 선별 수집

MongoDB에서 데이터 선별 수집은 DB 단위, Collection 단위, Key 단위, 특정 키워드나 쿼리 결과 단위로 가능하다. 선별수집에는 MongoDB에서 기본적으로 제공하는 유틸리티인 mongodump와 mongoexport를 활용한다.

mongodump는 DB와 Collection 단위로 BSON(JSON의 바이너리 형태) 포맷의 파일로 데이터를 추출한다. mongodump의 사용방법은 다음과 같다.

```
mongodump -h 서버명 또는 IP:포트 -d DB 이름
-c Collection 이름 -o 결과파일을 저장할 경로
```

mongoexport는 DB, Collection, 키 단위 또는 특정 질의결과를 JSON, CSV 포맷의 파일로 데이터를 추출한다. mongoexport의 사용방법은 다음과 같다.

```
mongoexport -h 서버명 또는 IP:포트 -d DB 이름
-c Collection 이름 -f 키 이름(','로 분리하여 여러개의 키 입력 가능) -q {'키':'값','키':/값(정규표현식)/}
-o 결과 파일 이름 --csv(csv 형태 결과를 원할 경우)
```

MongoDB에 접속하여 데이터를 확인할 경우에는 find() 명령어를 사용한다. 사용방법은 db.Collection.find({'키':'값'}) 이며 이때 '값'에 검색하고자 하는 정확한 검색어를 입력하거나 외따옴표 대신 /를 사용할 경우 정규표현식 형태로 검색어를 질의할 수 있다. find 명령어로 특정 데이터를 찾는 질의 방법은 매우 다양하여 본 논문에서는 기본적인 사용법만 제시하였다.

MongoDB가 샤딩 구성으로 운영되고 있고 특정 데이터와 관련하여 삭제된 레코드를 분석하거나 서버의 변동사항을 파악하기 위해 세부로그를 수집해야 한다면 대상 데이터가 저장되어 있는 서버를 찾아야 한다. 특정 데이터가 저장되어 있는 서버를 찾기 위해서는 sh.status("verbose") 명령을 사용할 수 있다. sh.status("verbose") 명령을 사용하면 데이터가 나누어진 Collection 이름과 샤드키의 범위가 표시되며 해당 범위의 데이터가 어떤 서버에 저장되어 있는지 파악할 수 있다.

4.4 GridFS 데이터 수집

GridFS는 대용량 파일을 MongoDB에 입·출력할 수 있는 기능으로 MongoDB를 활용하는 시스템

에서 자주 활용된다. 따라서 조사 대상 MongoDB 서버에서 GridFS에 저장되어 있는 파일 목록을 확인하고 필요시 파일을 수집해야 한다.

GridFS를 통해 파일을 저장할 경우에는 Fig. 10.과 같이 MongoDB에 fs.files와 fs.chunks라는 2개의 Collection이 생성된다. fs.files Collection에는 파일에 대한 메타데이터가 저장되며, 저장되는 내용은 파일이름, Chunk 크기(파일이 분할된 크기), 업로드된 시간, 파일의 해시값(MD5), 파일의 크기이다. fs.chunks Collection에는 파일을 256KB 크기의 Chunk로 분할하여 바이너리 데이터를 저장한다.

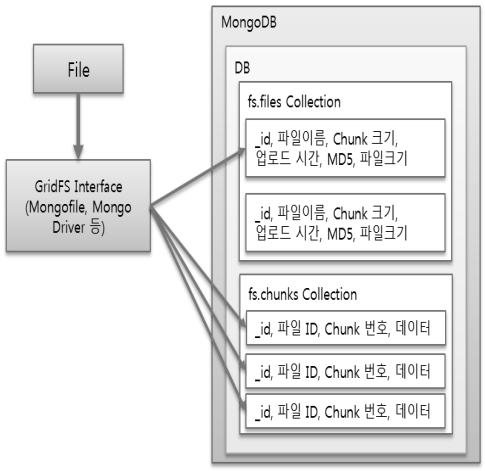

Fig.10. The architecture of GridFS

MongoDB에 GridFS로 저장된 파일 조사는 크게 세 가지 방법으로 가능하다.

첫 번째는 MongoDB에 직접 접속하여 각 DB에 fs.files Collection과 fs.Files Collection의 존재 여부를 확인한 다음 직접 조회하는 방법이다. 만약 특정 DB의 GridFS 파일 목록을 조사하고 싶다면 MongoDB Shell에서 Find 명령어를 통해 특정 파일명, 특정 시간에 업로드된 파일, 파일 크기 등으로 검색할 수 있다. MongoDB Shell에서 파일의 메타정보를 통해 조사대상 파일을 검색하는 것은 용이하나, 파일자체를 다운로드하는 것은 불가능하다. 다음은 Find 명령어를 통해 GridFS로 저장된 파일 목록을 검색하는 예이다.

- 전체 목록 조회 : db.fs.files.find()
- 특정 키워드가 포함된 파일명 조회 : db.fs.files.find({filename:/키워드/})
- 특정 시점 이후 업로드된 파일 조회 : db.fs.files.find({uploadDate:{$gte: new Date ("년-월-일T시:분:초+09:00")}})

두 번째는 MongoDB에서 제공하는 유틸리티인 mongofiles를 이용하는 방법이다. mongofiles를 이용하여 DB에 있는 파일목록을 조회할 경우 파일명과 크기 이외의 메타정보가 표시되지 않는다. 다음은 mongofiles를 이용하여 GridFS로 저장된 파일 목록을 확인하고 특정파일을 수집하는 예이다.

- 전체 목록 조회 : mongofiles -h 서버명 또는 IP:포트 -d DB 이름 list
- 특정 파일 수집 : mongofiles -h 서버명 또는 IP:포트 -d DB이름 get 파일이름

세 번째는 MongoDB Driver를 사용하여 자체적으로 도구를 개발하거나 MongoDB 클라이언트 S/W를 이용하는 방법이다.

GridFS 데이터를 수집하기 위해 현재까지 가장 효과적인 방법은 MongoDB Shell에서 메타데이터 검색을 통해 수집할 파일을 검색한 후 mongofiles로 파일을 수집하는 방법이다.

4.5 Data File 구조 분석 및 삭제레코드 분석

MongoDB의 데이터 파일 Default 경로는 "/data/db/"이나 MongoDB 구동시 --dbpath 옵션으로 변경 가능하다. 데이터 파일은 DB의 메타정보를 저장하는 파일과 데이터를 저장하는 파일로 구분된다. 메타정보를 저장하는 파일은 Namespace 파일이라고 하는데 "{DB이름}.ns" 형태의 파일명으로 생성된다. 데이터를 저장하는 파일은 "{DB이름.숫자}" 형태의 파일명으로 저장되며, 데이터의 용량이 증가하여 파일 하나의 크기가 2GB를 넘을 경우 파일명의 숫자가 0부터 1씩 증가하면서 새로운 파일이 생성된다. Fig. 11.을 보면 기본적으로 생성되는 DB인 local DB와 추가로 생성한 cloud, cloud-docs라는 DB를 확인할 수 있다.

MongoDB의 데이터 구조는 크게 Namespace-Details, Extent, Record로 구성된다. Name-

Fig.11. An example of the generated data files

spaceDetails 구조체는 "DB이름. Collections 이름" 형태의 Namespace가 사용하는 Extent의 위치, 삭제된 레코드 리스트, 레코드 크기, 개수, 인덱스 관련정보 등 메타정보를 저장하고 있다[16]. Namespace 파일의 NamespaceDetails 구조체를 통해 Extent와 Record를 따라가면서 파싱하면 Namespace 파일과 데이터 파일에서 데이터를 추출할 수 있다. Fig. 12.는 MongoDB의 전체적인 데이터 구조를 보여준다.

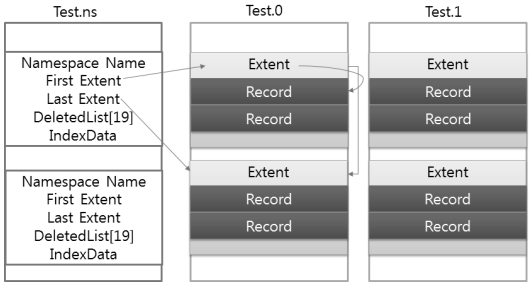

Fig.12. The data file structure of MongoDB

Extent는 연속된 블록의 집합으로, 같은 Namespace에 속한 Extent는 이중 링크드리스트로 연결되어 있고 각 Extent에는 Record가 속해 있다. Record는 BSON Document나 B-tree 형태의 데이터를 저장하고 있으며 관련된 Record끼리 이중 링크드리스트로 연결되어 있다. Fig. 13.은 Extent와 Record의 구조를 보여준다.

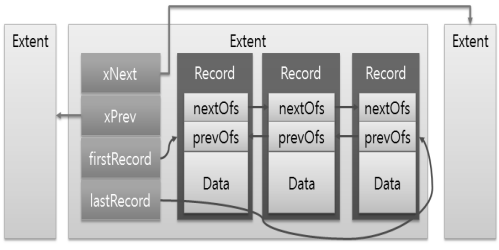

Fig.13. The structure of Extent and Record

4.5.1 NamespaceDetails 구조체

MongoDB는 Hashtable을 이용하여 Namespace 파일에서 NamespaceDetails 구조체를 찾는다. NamespaceDetails 구조체의 첫 4bytes는 해시값이 나타나고, 다음 128bytes는 "DB이름. Collections 이름" 형태의 Namespace가 기록된다. 이후 각 8bytes는 첫 번째와 마지막 Extent의 위치를 나타낸다. 위치를 나타내는 8bytes의 처음 4bytes는 데이터파일의 번호("DB명.숫자"에서 뒤의 숫자)를 나타내고, 다음 4bytes는 파일 내에서의 오프셋 값을 나타낸다. 그 다음은 삭제된 레코드를 배열 형태로 기록하고 있으며, 다음 8bytes씩 데이터의 크기와 레코드 개수가 나타난다.

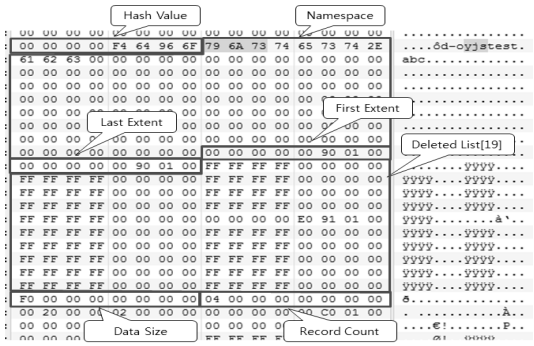

Fig.14. The format of NamespaceDetails struct

Namespace 파일에서 NamespaceDetails 구조체를 통해 Extent의 위치와 삭제된 레코드의 위치를 파악할 수 있다.

4.5.2 Extent 구조체

NamespaceDetails 구조체에서 Extent의 위치를 확인하고 이동하면 Signature 값인 "0x44434241"을 확인할 수 있다. 이후 8bytes씩 해당 Extent의 위치, 다음 Extent의 위치, 이전 Extent의 위치가 나타난다. 다음 128bytes는 Namespace가 나오고 이후 4bytes는 Extent의 크기를 표시한다. 이후 각 8bytes씩 첫 Record의 위치와 마지막 Record의 위

치가 기록된다. Extent 구조체의 Record 위치를 확인하여 Record를 탐색할 수 있다.

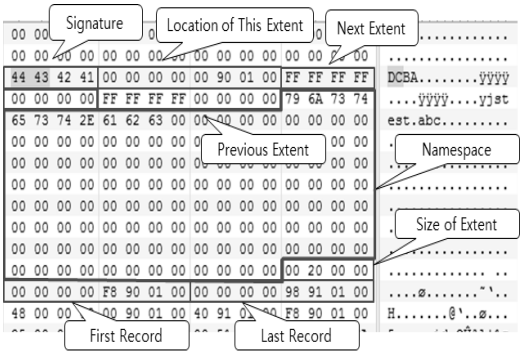

Fig.15. The format of Extent struct

4.5.3 Record 구조체

Extent 구조체에서 Record의 위치를 확인하고 이동하면 각 4bytes씩 Record의 크기, 해당 Record가 속한 Extent의 Offset, 다음 Record의 Offset, 이전 Record의 Offset이 나타난다. 이후 BSON 이나 B-Tree 형태의 Data를 확인할 수 있다.

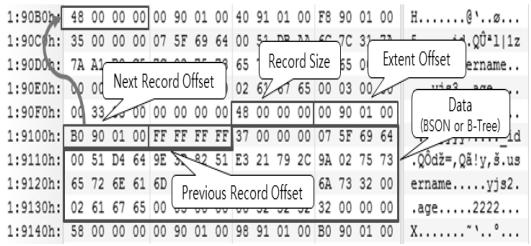

Fig.16. The format of Record struct

4.5.4 삭제된 Record 분석

MongoDB는 관리자가 관리 명령어인 repairDatabase 또는 Collection Compact를 실행하기 전까지 NamespaceDetails 구조체의 Deleted List에 삭제된 Record의 위치를 기록하고 실제 데이터를 삭제하지 않는다. 특정 Record 삭제 후 변화를 보면 Fig. 17.처럼 삭제된 Record와 관련된 Offset 값이 변경되고 삭제된 Record의 Data의 첫 번째 4bytes의 값이 "0xEEEEEEEE"로 변경된다.

삭제된 Record를 복구하기 위해서는 Name-

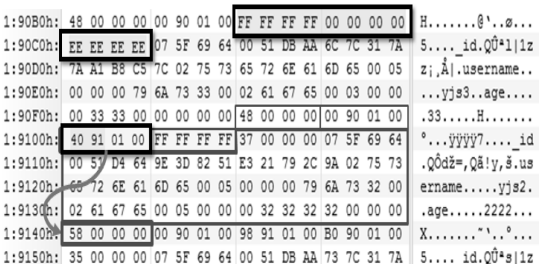

Fig.17. The changes after deleting record

space 파일이 존재하는 경우 DeletedList 확인 후 삭제된 Record의 위치를 탐색하여 복구가 가능하며, Namespace 파일이 유실된 경우에는 데이터 파일에서 삭제된 Record의 시그니처인 "0xEEEEEEEE"을 탐색하여 복구할 수 있다.

4.6 로그 파일 분석

관계형 DBMS와 마찬가지로 MongoDB도 다양한 로그를 남긴다. MongoDB 서버의 로그를 분석하여 최근 접속자 목록이나 DB의 변경 이력, 행위 등을 분석할 수 있다. MongoDB 로그는 일반 Log, Oplog, startup Log, Diagnostic Log가 있다. 각 로그는 MongoDB의 운영 형태에 따라 생성될 수도 있고 존재하지 않을 수도 있으므로 로그 수집·분석 이전에 조사대상 서버의 운영환경을 파악해야 한다.

4.6.1 일반로그

MongoDB의 일반 로그는 Stdout, 즉 화면으로 보여주는 것이 기본설정이다. MongoDB를 서비스에 이용한다면 일반적으로 로그파일을 지정하여 저장한다. 로그파일의 위치는 MongoDB 구동 시 --logpath 옵션으로 지정한다. 유닉스 계열의 OS는 syslog 형태로 로그를 저장할 수도 있다. 일반 로그는 로그를 기록하는 내용의 세밀함에 따라 Log Level을 기본에서 -vvvv(v 개수가 많을수록 세밀한 로그)까지 조정할 수 있다. 여기서는 기본 Log Level에서 획득할 수 있는 정보를 다룰 것이다.

일반로그는 Fig. 18.과 같이 이벤트 발생일시, 이벤트 생성자, 이벤트 내용 형태로 기록된다. 일반로그에서 확인할 수 있는 정보는 서버의 구동/정지, 클라이언트 연결, DB 생성/삭제, Collection 생성/삭제 등이다.

```
Fri Oct 18 13:53:17.635 [initandlisten] waiting for connections on port 27017
Fri Oct 18 13:53:17.651 [websvr] admin web console waiting for connections on po
rt 28017
Fri Oct 18 13:53:42.486 [initandlisten] connection accepted from 127.0.0.1:54799
 #1 (1 connection now open)
Fri Oct 18 13:55:56.054 [FileAllocator] allocating new datafile /data/test\test1
.ns, filling with zeroes...
Fri Oct 18 13:55:56.194 [FileAllocator] done allocating datafile /data/test\test
1.ns, size: 16MB,  took 0.142 secs
Fri Oct 18 13:55:56.210 [FileAllocator] allocating new datafile /data/test\test1
.0, filling with zeroes...
Fri Oct 18 13:55:56.802 [FileAllocator] done allocating datafile /data/test\test
1.0, size: 64MB,  took 0.6 secs
Fri Oct 18 13:55:56.818 [conn1] build index test1.test { _id: 1 }
Fri Oct 18 13:55:56.818 [conn1] build index done.  scanned 0 total records. 0.00
2 secs
Fri Oct 18 13:55:56.818 [conn1] insert test1.test ninserted:1 keyUpdates:0 locks
(micros) w:768002 768ms
Fri Oct 18 13:55:56.818 [FileAllocator] allocating new datafile /data/test\test1
.1, filling with zeroes...
Fri Oct 18 13:55:57.707 [FileAllocator] done allocating datafile /data/test\test
1.1, size: 128MB,  took 0.876 secs
```

Fig.18. An Example of MongoDB log

조사 대상 MongoDB 서버에서 일반로그를 저장하고 있다면 세부적인 데이터 변경사항까지 알기는 어렵지만 서버의 운영 상태와 DB의 구조변경 시간과 내용을 분석할 수 있다.

4.6.2 Oplog

Oplog는 복제셋 형태로 운영되는 MongoDB 서버 간의 동기화를 위해 DB의 변경사항을 저장하는 로그이다. Oplog는 local DB 내에 oplog.rs Collection에 저장된다. Oplog는 MongoDB 운영 환경에 따라 기록되어 있는 용량이 다르며 32bit 서버 에서는 50M, 64bit 서버에서는 미사용중인 디스크 용량의 5%를 사용한다. MongoDB 구동시 --oplogSize 옵션을 통해 크기를 변경할 수 있다. Oplog의 크기가 서버마다 다를 수 있기 때문에 정확하게 어느 시점까지 DB의 변화를 분석할 수 있을지 특정하기 어렵지만 최근 DB의 변화를 데이터 수준까지 세부적인 부분을 알 수 있다는 점에서 의미가 있다. 다음은 Oplog의 형태와 각 키의 의미에 대한 설명이다.

```
{
  "ts" : { "t" : 1286821993000, "i" : 1 },
  "h" : NumberLong("5491114356580488109"),
  "op" : "i",
  "ns" : "test.foo",
  "o"          :          {          "_id"          :
ObjectId("4cb3586928ce78a2245fbd57"),
   "x" : 2, "y" : 1 }
 }
```

- ts : 해당 연산이 발생한 타임스탬프(t)와 연산을 처리하는데 소요된 시간(i)
- h : 각 연산의 식별 ID
- op : 연산의 종류, i는 삽입, u는 수정, d는 삭제, n은 연산이 아닌 단순 메시지를 나타냄
- ns : 해당 연산에 의해 영향을 받는 DB와 Collection
- o : 연산의 내용(데이터)

조사 대상이 복제셋이나 샤딩으로 구성되어 있을 경우 local DB의 oplog.rs에서 특정 시점의 DB 변경 내용을 분석하거나 데이터의 삽입, 수정, 삭제 별로 변화를 분석할 수 있다.

V. 결 론

본 논문에서는 대표적인 NoSQL DBMS인 MongoDB에 대한 디지털 포렌식 절차와 아티팩트 그리고 세부 조사 방법을 제시하였다. 기존 관계형 DBMS와는 다르게 대용량, 다수의 서버, 비정형 데이터를 가지고 있기 때문에 실제 디지털 포렌식 조사를 수행할 때 고려해야 하는 사항들이 많아진다. 그러나 데이터 파일의 포맷이나 서버가 동작하는 방식이 상당히 직관적이어서 관련된 명령어와 내부적인 동작 방식을 이해한다면 관계형 DBMS에 비해 기술적 난이도가 높은 편은 아닌 것으로 판단된다.

향후에는 효과적으로 MongoDB를 분석할 수 있는 디지털 포렌식 조사 도구를 개발할 계획이며, MongoDB 이외에도 HBase, Cassandra, CouchDB 등 다양한 NoSQL DBMS에 대한 포렌식 조사 기법 연구를 진행할 예정이다.

References

[1] Philip Russom, "Big data analytics," TDWI Best Practices Report, Sep. 2011.

[2] Harmett Kaur Khanuja and D.S.Adane, "A Framework For Database Forensic Analysis," Computer Science & Engineering : An International Journal(CSEIJ), vol. 2, No. 3, pp. 27-41, Jun. 2012.

[3] Peter Fruhwirt, Markus Huber, Martin Mulazzani, and Edgar R. Weippl, "InnoDB database forensics," 2010 24th

IEEE International Conference on Advanced Information Networking and Applications, pp. 1028-1036, Apr. 2010.
〔4〕 Paul M. Wright, "Oracle Database Forensics using LogMiner," NGSSoftware, 2005.
〔5〕 Kyriacos E.Pavlou and Richard T. Snodgrass, "Forensic analysis of database tampering," ACM Transactions on Database Systems (TODS), vol. 33, no. 30, Nov. 2008.
〔6〕 Jonghyun Choi, DooWon Jeong, and Sangjin Lee, "The method of recovery for deleted record in Oracle Database," Journal of the Korea Institute of Information Security and Cryptology, 23(5), pp947-955, Oct. 2013.
〔7〕 Michael Stonebraker, "SQL databases v. NoSQL databases," Communications of the ACM, vol 53, issue 4, pp. 10-11, Apr. 2010.
〔8〕 Yishan Lie and Sathiamoorthy Manoharan, "A performance comparison of SQL and NoSQL databases," In Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on. IEEE, pp. 15-19, Aug. 2013.
〔9〕 MongoDB Homepage, http://www.mongodb.com
〔10〕 DB-ENGINES Homepage, http://www.db-engines.com/en/ranking
〔11〕 JSON Homepage, http://www.json.org
〔12〕 Kyle Banker, MongoDB in action, Manning Publications Co., ,2011.
〔13〕 MongoDB manual, http://docs.mongodb.org/manual
〔14〕 Open source tool for MongoDB schema analysis, https://github.com/variety/variety
〔15〕 MongoDB client tool, https://sites.google.com/site/tadpolefordb
〔16〕 MongoDB source code, https://github.com/mongodb/mongo

〈저자소개〉

윤 종 성 (Jong-Seong Yoon) 정회원
2005년 3월: 공군사관학교 전산과학과 이학사
2013년 3월~현재: 고려대학교 정보보호대학원 정보보호학과 석·박사통합과정
〈관심분야〉 디지털 포렌식, 정보보호

정 두 원 (Doo-won Jeong) 정회원
2011년 8월: 고려대학교 공과대학 산업경영공학과 공학사
2011년 9월~현재: 고려대학교 정보보호대학원 정보보호학과 석·박사통합과정
〈관심분야〉 디지털 포렌식, 정보보호, 빅데이터 분석

강 철 훈 (Chul-Hoon Kang) 정회원
2006년 12월~현재: 대검찰청 디지털포렌식센터 데이터베이스포렌식 팀장
〈관심분야〉 Database Forensic, Accounting Forensic

이 상 진 (Sang-jin Lee) 종신회원
1989년 2월~1999년 2월: 한국전자통신연구원 선임 연구원
1999년 2월~2001년 8월: 고려대학교 자연과학대학 조교수
2001년 9월~현재: 고려대학교 정보보호대학원 교수
〈관심분야〉 대칭키 암호, 정보은닉이론, 디지털 포렌식