KIPS Tr. Comp. and Comm. Sys.
Vol.5, No.5 pp.117~126 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2016.5.5.117

Digital Forensics Investigation of Redis Database

Choi Jae Mun† · Jeong Doo Won†† · Yoon Jong Seong†† · Lee Sang Jin†††

ABSTRACT

Recently, increasing utilization of Big Data or Social Network Service involves the increases in demand for NoSQL Database that overcomes the limitations of existing relational database. A forensic examination of Relational Database has steadily researched in terms of Digital Forensics. In contrast, the forensic examination of NoSQL Database is rarely studied. In this paper, We introduce Redis (which is) based on Key-Value Store NoSQL Database, and research the collection and analysis of forensic artifacts then propose recovery method of deleted data. Also we developed a recovery tool, it will be verified our recovery algorithm.

Keywords : NoSQL, Redis, Database Forensic, Digital Forensic

Redis 데이터베이스에 대한 디지털 포렌식 조사 기법 연구

최 재 문† · 정 두 원†† · 윤 종 성†† · 이 상 진†††

요 약

최근 빅 데이터나 소셜 네트워크 서비스의 활용도가 증가하면서 기존 관계형 데이터베이스의 한계를 극복한 NoSQL 데이터베이스의 수요가 꾸준히 증가하고 있다. 디지털 포렌식 관점에서 관계형 데이터베이스의 디지털 포렌식 조사 기법은 꾸준히 연구되어 왔으나 NoSQL 데이터베이스의 디지털 포렌식 조사 기법에 대한 연구는 거의 없는 실정이다. 본 논문에서는 메모리 기반의 Key-Value Store NoSQL 데이터베이스인 Redis를 소개하고 디지털 포렌식 관점에서 살펴보아야 할 아티팩트의 수집과 분석, 삭제된 데이터 복구 기법을 제안한다. 또한 제안된 데이터 복구 기법을 도구로 구현하여 복구 기법을 검증한다.

키워드 : NoSQL, Redis, 데이터베이스 포렌식, 디지털 포렌식

1. 서 론

소셜 네트워크 서비스와 사물인터넷의 활성화에 따라 데이터가 폭발적으로 증가하고 있다. 이에 대량화된 데이터를 효과적으로 처리하기 위해 다양한 기술들이 개발되고 있는데 NoSQL 데이터베이스가 대표적으로 활용되고 있다.

관계형 데이터베이스는 높은 안정성과 SQL 질의문을 통해 데이터를 효과적으로 관리할 수 있지만 대량화된 데이터의 입·출력, 스키마 변경, 데이터베이스 복제와 확장 등이 유연하게 이뤄질 수 없어 다양하고 대용량화된 데이터를 관리하기에 적합하지 않다. 따라서 최근 기업이나 기관에서는 다양하고 대용량화된 데이터를 처리하기 위해 NoSQL DBMS를 도입하는 추세이다[1, 2].

디지털 포렌식 조사과정에서 DBMS에 대한 분석기법의 수요가 높아 데이터베이스 조사기법, 삭제된 데이터 복구 방안이 꾸준히 연구되어 왔으나 대부분 Oracle, MSSQL, MySQL 등 관계형 데이터베이스를 대상으로 한 연구가 주를 이루었다. 최근 NoSQL 데이터베이스에 대한 시장의 수요 증대와 관련 솔루션이 다수 개발되고 있어 NoSQL 데이터베이스에 대한 디지털 포렌식 분석 기법 방안을 마련해야 한다.

본 논문에서는 Key-Value Store NoSQL 데이터베이스인 Redis를 소개하고 디지털 포렌식 관점에서 살펴보아야 할 아티팩트의 수집과 분석 방안, 삭제된 데이터 복구 기법을 제안한다. 또한 제안된 복구 기법을 도구로 구현하여 복구 기법을 검증한다. 이를 통해 Redis 데이터베이스에 대한 디지털 포렌식 조사 기법을 제안한다.

※ 이 논문은 2015년도 정부(미래창조과학부)의 재원으로 한국연구재단-공공복지안전사업의 지원을 받아 수행된 연구임(2012M3A2A1051106).
† 준 회 원 : 고려대학교 사이버국방학과 석사과정
†† 비 회 원 : 고려대학교 정보보호학과 석·박사통합과정
††† 종신회원 : 고려대학교 교수
Manuscript Received : March 10, 2016
First Revision : April 21, 2016
Accepted : May 17, 2016
* Corresponding Author : Lee Sang Jin(sangjin@korea.ac.kr)

2. 관련 연구

데이터베이스에 대한 디지털 포렌식 조사 기법 연구는 데이터 추출 기법, 로그를 통한 포렌식 조사 기법, 삭제된 데이터 복구 기법 등 크게 3가지 영역으로 구분돼 다양한 연구가 진행되고 있지만, 연구 내용은 관계형 데이터베이스에 대한 조사 기법 연구가 주를 이루었다.

데이터 추출 기법에 관한 연구는 2009년 임경수가 DBMS의 메타 정보를 수집하고 이를 토대로 테이블간의 연관 관계를 검색하고 조사 데이터를 분석하는 도구를 제작하였으며[3], 2012년 이동찬이 관계형 데이터베이스에서 DBMS의 스키마 정보와 정책 정보를 수집하여 조사 대상 테이블에 대한 참조 테이블을 검색하고 이를 연관 관계를 검증하여 데이터를 추출하는 방안을 연구하였다[4]. 또한 Paul Wright는 Oracle 데이터베이스의 Redo Log 파일을 분석하여 DBMS의 변경내용을 추적하는 도구를 개발하여 포렌식 조사에 활용할 수 있는 방안을 제안하였다[5]. 삭제된 데이터 복구 기법 연구는 Peter Fruhwir가 MySQL의 Inno DB 엔진에서 Redo Log를 통한 데이터 복구 방안을 제시하였으며[6], 최종현이 Oracle 데이터베이스에서 데이터 파일을 이용하여 삭제된 레코드를 복구하는 방안을 제시하였다[7]. 또한 James Wagner는 범용적인 관계형 데이터베이스 8종을 선정하여 데이터베이스 저장소의 원리를 분석하고 Page라는 기본 단위를 통해 각 DBMS에서 공통적으로 사용되는 파라미터들을 일반화하여 디스크 이미지나 메모리 스냅샷에서 단편화된 데이터를 복구하는 방안을 제시하고 도구로 구현하였다[8]. 위 논문들은 주로 특정 데이터베이스 조사 시 일부 절차에 해당되는 기술에만 초점을 맞추어 실질적으로 Redis 수집 및 분석에 이르는 조사 절차에 적용하는 데에는 한계가 있다.

NoSQL에 관련된 연구는 2014년 윤종성이 MongoDB에 대한 디지털 포렌식 조사 기법에 대한 연구를 진행하였다. 논문에는 Mongo DB의 특징과 운용방법을 서술하고 Mongo DB 특성에 따른 디지털 포렌식 절차가 논의되었으며, 메타 데이터를 활용하여 데이터 파일에서 삭제된 리스트를 복구하는 기법이 제시되었는데[9], 데이터베이스의 구조를 파악하고 데이터 및 로그 수집-분석하는 전체적인 조사 절차를 제시하였다. Mongo DB의 경우 Redis와 동일한 NoSQL DBMS이며, 복제를 지원하고 스키마가 없는 공통점을 가지므로 조사 절차는 참고할 수 있다. 그러나 Mongo DB는 Document DBMS로 각 레코드는 JSON 형태로 구성되어 있어 Redis의 Key-Value Store 모델과는 다른 저장방식을 가지고 있으며, 남겨지는 로그의 범위이나 내용도 상이하다. 또한 Mongo DB의 경우 타 데이터베이스와 동일한 디스크 기반의 DBMS이지만, Redis의 경우 메모리 기반의 DBMS로 데이터 추출 및 데이터 파일 구조가 다르기 때문에 기존의 조사기법을 Redis에 적용할 경우 많은 부분의 수정이 필요하다.

Redis에 대한 국외 연구로는 2014년 중국의 Ming Xu가 Redis에서 데이터가 저장되는 RDB 파일과 사용자가 입력한 쿼리가 저장되는 AOF 파일을 분석하여 저장된 데이터와 쿼리를 추출하는 기법이 연구하였다[10]. 하지만 Redis 데이터베이스를 운용하는 시스템의 디지털 포렌식 조사 절차나 삭제된 데이터를 복구하는 방안은 논의되지 않았다.

본 논문의 대상인 Redis는 메모리 기반의 NoSQL 데이터베이스로 데이터베이스의 테이블 구조가 정의되어 있지 않고, 다수의 복제 서버가 존재할 수 있으며, 데이터 파일의 구조가 다르기 때문에 기 연구된 스키마 정보를 통한 테이블 연관관계 분석이나 삭제된 데이터 복구 기법은 적용될 수 없다. 특히 기존의 데이터베이스와 다르게 메모리라는 저장매체의 특수성에 따라 Redis의 특성에 맞는 조사 방안이 마련되어야 한다.

따라서 본 논문에서는 Key-Value Store NoSQL 데이터베이스인 Redis를 소개하고 디지털 포렌식 관점에서 살펴보아야 할 아티팩트의 수집과 분석, 삭제된 데이터 복구 기법을 제안하고 데이터 복구 도구를 구현하였다. 이를 통해 Redis에 대한 디지털 포렌식 조사 절차와 방안을 제시한다.

3장에서는 Redis 데이터베이스를 소개하고 자료형과 데이터 저장 방식, 데이터 파일 구조를 설명한다. 4장에서는 Redis 데이터베이스의 조사를 위한 다양한 아티팩트 수집 및 분석 방안 그리고 데이터 파일 구조를 통해 삭제된 데이터 파일의 복구 방안을 제시하고 이를 도구로 구현하여 제안된 복구 기법을 검증한다.

3. Redis 데이터베이스

Redis(Remote Dictionary Server)는 2009년 Salvatore Sanfilippo가 개발한 BSD 라이센스 기반의 오픈소스 프로젝트로, Key를 사용한 완전 일치 검색을 통해 데이터를 조회하는 Key-Value Store 데이터베이스이다. DBMS의 인기도 순위를 평가하는 DB-Engines에 따르면 2016년 4월 등록된 292개의 DBMS 중 9위, Key-Value Store의 DBMS에서는 가장 높은 순위를 기록하고 있다[11].

Redis는 메모리 기반의 DBMS이기 때문에 일반 디스크 저장소를 사용하는 데이터베이스에 비해 빠른 속도를 보장하고 Hashes, List, Set 등 다양한 자료형을 지원하여 데이터를 유연하게 관리할 수 있기 때문에 많은 곳에서 사용되고 있다. 대표적으로 페이스북, 인스타그램, 텀블러, 카카오톡 등 대량의 메시지를 실시간으로 처리해야 하는 시스템에서 Result Cache와 데이터를 저장하는데 주로 사용되고 있으며, 소규모 서비스 및 빠른 응답 처리를 위한 시스템에서 저장소로 사용되고 있다.

Redis는 메모리 기반의 DBMS이기 때문에 시스템의 전원이 꺼지면 데이터가 모두 손실된다. 따라서 메모리상의 데이터베이스를 디스크에 파일로 저장하고, 서버 재시작시 파일에서 데이터를 읽어와 메모리상에 다시 로드하는 방법을 사용하여 데이터를 보존한다.

3.1 데이터베이스 구성

Redis는 Master-Slave 관계의 데이터 복제를 지원한다. 데

이터 복제는 데이터베이스의 성능 향상이나 데이터 백업을 위해 사용하는데 Redis는 싱글 Slave와 다중 Slave 구성을 지원한다. Redis는 메모리 기반의 데이터베이스이기 때문에 메모리상의 데이터를 디스크에 파일로 저장하는 기능을 지원한다. 하지만 이 방법을 이용하지 않고, 복제 서버를 구성하여 Master 노드의 데이터를 Slave 노드에 저장하는 경우도 있기 때문에 Redis를 운용하는 서버를 조사할 경우 데이터베이스의 운용 사항을 파악하여 복제 세트 구성을 확인해야 한다.

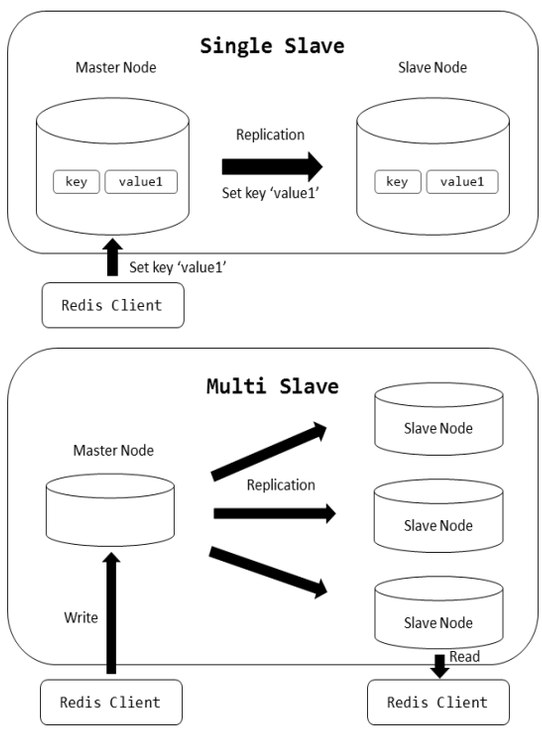

Fig. 1. Redis Database Replication

3.2 자료형

Redis는 NoSQL 데이터베이스 Store 중 가장 간단한 Key-Value Store 구조를 가지지만 다양한 자료형을 지원하여 데이터를 유연하게 관리할 수 있다. 자료형은 총 5가지이며 String, List, Set, Sorted Set, Hashes이다[12, 13].

1) String

String은 가장 간단한 형태의 자료형으로 문자열, 숫자를 저장할 수 있으며, 바이너리 파일도 저장 가능하다. 최대 길이는 512MB이며, Key에 하나의 데이터를 저장한다.

2) List

배열 형태의 자료형이며, PUSH와 POP 명령어를 통해 리스트의 좌, 우에 데이터를 삽입하거나 삭제할 수 있다.

Fig. 2. List Data Type

3) Set

Set은 List와 다르게 순서가 지정되지 않고, 중복된 데이터가 존재하지 않는 집합이다. List와 같은 PUSH, POP을 통해 데이터의 삽입·삭제는 할 수 없지만 Redis는 집합 연산을 지원하고 있기 때문에 Set 자료형 간의 집합 연산이 가능하다.

4) Sorted Set

Sorted Set은 Set과 유사하지만, 각 데이터는 Score라는 가중치를 가지고 있다. 이 가중치의 값을 통해 Sorted Set 내에서 오름차순으로 저장된다.

Score	Value1
Score	Value2
Score	Value3
Score	Value4

Fig. 3. Sorted Set Data Type

5) Hashes

Hashes는 “필드명”과 그에 대응되는 “필드값”의 연속으로 이루어져 있다. 이는 연상 배열과 같은 형식이다.

Field1	Field1 value
Field2	Field2 value
Field3	Field3 value
Field4	Field4 value

Fig. 4. Hashes Data Type

3.3 데이터 저장 방식

Redis 데이터베이스는 메모리에 로드되어 운용된다. 따라서 전원을 차단하거나 정전 시 데이터가 모두 사라지게 된다.

Redis에서는 데이터의 영속성을 위해 메모리에 로드된 데이터를 디스크에 일정 주기로 저장한다. 데이터 저장방식은 2가지로 구분되는데 Snapshotting(RDB) 방식과 Append of file(AOF) 방식이다.

1) Snapshotting(RDB)

Snapshotting 방식은 메모리상의 데이터를 디스크에 옮겨 담는 방식이다. 디스크에 저장하는 방식은 2가지로 구분되는데 SAVE 방식과 BGSAVE으로 구분된다. SAVE 방식은 Redis의 동작을 멈추고 데이터를 저장하는 방식이며, BGSAVE 방식은 Redis Client에서 ‘BGSAVE’ 명령어를 통해 Redis의 동작을 멈추지 않고 자식 프로세스를 생성 후 해당 시점의 데이터를 저장하는 방식이다. 서버 재시작 시 Redis는 RDB파일을 다시 읽어 메모리에 로드한다. Redis가 Snappshotting을 하는 시점은 환경 설정 파일에서 사용자가 정의 가능하다.

Table 1. Example of snapshot setting

Operation	Description
SAVE 900 10	if more than 10 key change occurs for 900 seconds then save

Table 1은 Snapshotting 시점 설정 방법이다. Snapshotting 방식을 통해 데이터를 저장하면 '/var/lib/redis' 경로에 'dump.rdb' 형태로 저장되며 저장되는 경로와 파일명은 환경 설정 파일에서 사용자가 정의할 수 있다.

2) Append of file(AOF)

Append of file 방식은 Redis에서 실행한 모든 명령어를 파일에 기록하는 방법이다. 서버 재시작시, 기록된 파일의 명령을 순차적으로 재실행하여 데이터를 복구한다. 모든 명령어를 기록하므로 AOF 파일이 사용자가 지정한 용량을 넘어서는 경우 자동으로 초기화를 하고 이후부터 명령어를 기록하는 설정을 할 수 있다.

Table 2. Example of AOF setting

Operation	Description
appendfsync always	Recorded AOF files each time the instruction is executed
appendfsync everysec	Recored AOF files in every seconds
appendfsync no	OS is in charge of recording AOF files

Table 2는 Append of file 시점 설정 방법이다. Append of file 방식을 통해 데이터를 저장하게 되면 '/var/lib/redis' 경로에 'appendonly.aof' 형태로 저장되며, 저장되는 경로와 파일명 또한 환경 설정 파일에서 사용자가 정의할 수 있다.

3.4 데이터 파일 구조

1) RDB 파일

RDB 파일은 메모리상의 모든 데이터가 사용자가 지정한 시점에 Snapshotting된 파일이며 구조는 다음과 같다[14].

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td colspan="5">Magic String</td><td colspan="3">RDB Versi-</td></tr>
<tr><td>on</td><td>DS</td><td colspan="2">DB Number</td><td colspan="2">Key Length</td><td colspan="2">Key</td></tr>
<tr><td colspan="2">Value Length</td><td colspan="4">Value</td><td colspan="2">Key Length</td></tr>
<tr><td colspan="2">Key</td><td colspan="2">Value Length</td><td colspan="4">Value</td></tr>
<tr><td colspan="7">...(Key-Value Pairs)</td><td>ER</td></tr>
<tr><td colspan="8">Checksum</td></tr>
</table>

- Magic String(5 Bytes) : Redis Signature
- RDB Version(4 Bytes) : RDB File Version
- DS(1 Byte) : Start of Database
- DB Number(2 Bytes) : Number of Database
- Key Length(Variable) : Key Length
- Key(Variable) : Key
- Value Length(Variable) : Value Length
- Value(Variable) : Value
- ER(1 Byte) : End of RDB File
- Checksum(8 Bytes) : CRC 64 Checksum Value

Fig. 5. Structure of RDB File

RDB 파일의 구조는 Fig. 5와 같은 구조를 가진다. Magic String부터 DB Number까지는 RDB 파일의 헤더이며, 이후에 Key-Value Pair가 저장된다. 다만 위의 구조는 Key의 Expire Time이 포함되어 있지 않은 구조이기 때문에, Expire Time-Key-Value 구조가 될 수도 있음에 유의하여야 한다. Fig. 6은 Redis 운용 후 추출한 RDB 파일을 Fig. 5의 구조로 해석한 것이다.

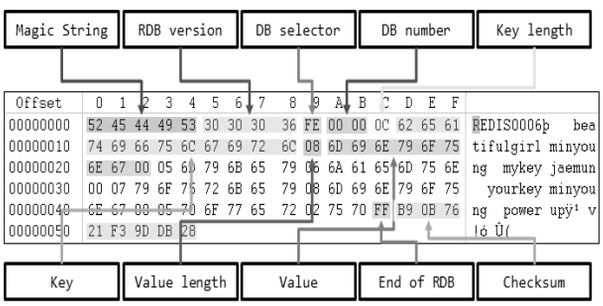

Fig. 6. Example of RDB File Structure

2) AOF 파일

AOF 파일은 Redis에서 데이터 생성·수정·삭제를 위해 실행된 모든 쿼리를 저장하는 파일이다.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>0x2A</td><td>NA</td><td colspan="2">0x0D0A</td><td>0x24</td><td>NAB</td><td colspan="2">0x0D0A</td></tr>
<tr><td colspan="8">Argument Data</td></tr>
<tr><td>0x24</td><td>NAB</td><td colspan="2">0x0D0A</td><td colspan="4">Argument Data</td></tr>
<tr><td>0x2A</td><td>NA</td><td colspan="2">0x0D0A</td><td>0x24</td><td>NAB</td><td colspan="2">0x0D0A</td></tr>
<tr><td colspan="8">Argument Data</td></tr>
<tr><td>0x24</td><td>NAB</td><td colspan="2">0x0D0A</td><td colspan="4">Argument Data</td></tr>
<tr><td colspan="8">...</td></tr>
</table>

- NA(Variable) : Number of Argument
- NAB(Variable) : Number of Argument of Bytes
- Argument Data(Variable) : Operation or Key or Value

Fig. 7. Structure of AOF File

AOF 파일의 구조는 Fig. 7과 같고, Fig. 8의 명령어 구조가 반복되어 저장된다. 제일 첫 바이트인 '*'은 Fig. 7의 '0x2A'와 같으며, 바로 뒤의 '2'는 NA 값으로 'SELECT', '0'의 2개 인자가 따라오는 것을 의미한다. 그 뒤의 문자는 '$'로 Fig. 7의 '0x24'와 같으며, 바로 뒤에 '6'은 NAB이다. 이 값은 뒤에 오는 Argument Data의 크기를 나타낸다.

AOF 파일은 이와 같은 구조로 하나의 명령어를 이루고 이 값이 반복되어 저장된다. 각 인자들 사이에는 '0x0D0A' 값을 가지는 CRLF 문자가 삽입되어 있는데, CRLF 문자를 구분자로 사용하여 쉽게 데이터를 파싱할 수 있다.

Fig. 8은 Redis 운용 후 추출한 AOF 파일을 Fig. 7의 구조로 해석한 것이다.

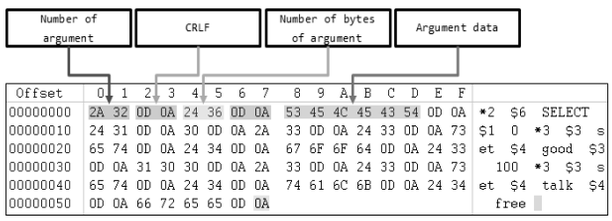

Fig. 8. Example of AOF File Structure

4. Redis 데이터베이스 디지털 포렌식 조사 기법

Redis 서버를 대상으로 디지털 포렌식 조사 시 데이터베이스는 메모리에 로드되어 있는 상태이며, NoSQL 특성상 다수의 서버에 복제되어 있고 비정형 스키마인 것을 고려해야한다. Master-Slave 관계의 데이터 복제를 지원하기 때문에, Master 혹은 Slave 서버의 환경 설정을 확인하여 연결된 데이터베이스 서버를 확인하여 분석해야 한다.

데이터베이스 분석을 위한 효율적인 방안은 운영 중 이었던 조사대상 데이터베이스의 환경, 추출한 환경 설정, 데이터베이스, 로그 파일을 통해 현장과 동일한 분석 환경을 구축하여 조사하는 것이다. 이를 통해 분석가는 원본 데이터를 훼손하지 않고 대상 데이터베이스에 대한 정밀한 분석이 가능하다.

4.1 운영 환경 및 구조 파악

데이터베이스 서버 조사 시 가장 먼저 데이터베이스의 운용 형태를 파악하는 것이 중요하다. 서버에 질의문, 환경 설정 파일의 구동 옵션을 통해 운용 형태와 데이터베이스 구조를 파악하여 대상 데이터베이스를 효과적으로 분석할 수 있도록 운영 정보를 수집하여야 한다.

1) 명령어를 통한 정보 수집

Redis에서는 다음과 같은 Info 명령어를 통해 서버의 정보를 획득할 수 있다.

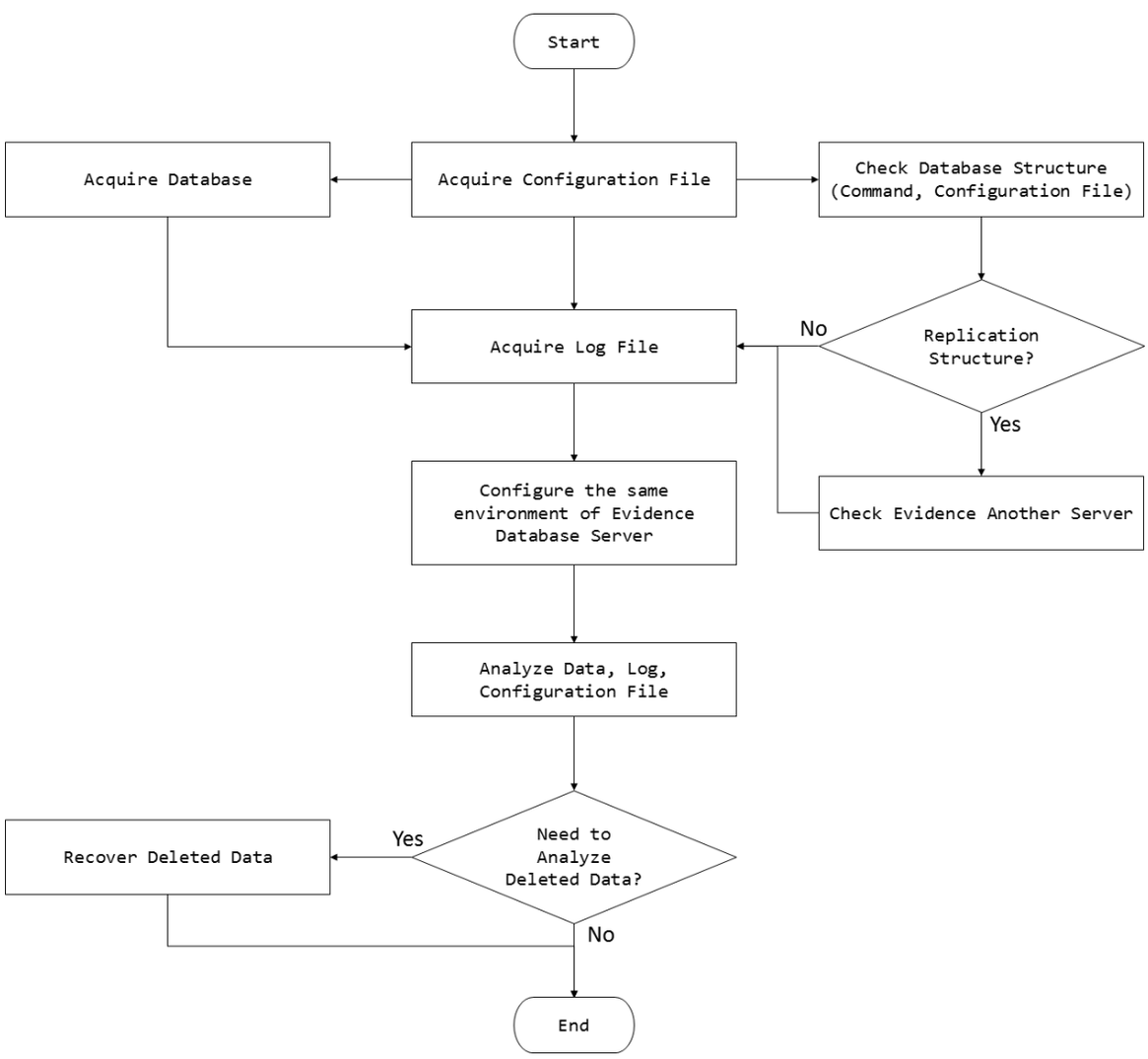

Fig. 9. The procedure for Redis Database forensic investigation

Fig. 10은 Table 3을 통해 얻을 수 있는 정보의 예시이다. Server의 정보를 통해 Redis의 버전이나 Port, 환경 설정 파일의 저장 경로를 획득할 수 있으며, Persistence 정보를 통해 데이터 백업 기능의 활성화 여부나 상태를 확인할 수 있다. Replication 정보의 경우 Master와 Slave로 구분되어 있는데 노드의 역할에 따라 정보가 구분되어 나타난다. 확인 가능한 정보는 노드의 역할, 연결된 Slave의 수, Slave 노드의 port, 상태 등을 확인할 수 있다. 요약된 정보 이외에도 분석에 활용될 수 있는 정보들이 다수 포함되어 있어 분석가는 Info 명령어를 통해 데이터베이스의 구조와 상태를 파악할 수 있다.

```
Info Server
{
   redis_version : 2.8.4
   OS : Linux 3.19.0-25-generic x86_64
   tcp_port : 6379
   config_file: /etc/redis/redis.conf
}
Info Persistence
{
   rdb_changes_since_last_save : 3
   rdb_last_save_time : 1438064972
   rdb_last_bgsave_status : ok
   aof_enabled : 1
   aof_last_write_status : ok
}
Info Replication(Master)
{
   role : master
   connected_slaves : 1
   slave0 : ip = 127.0.0.1, port = 5002, state = online, offset = 1818050955, lag = 0
   slave1
   …
   slaveN
}
Info Replication(Slave)
{
   master_host : 127.0.0.1
   master_port : 5001
   master_link_status : up
}
```

Fig. 10. Info instruction summary

Table 3. Info Command List

Operation	Description
Info server	General information of redis sever
Info client	Information of redis client
Info memory	Information of memory usage
Info persistence	Information of RDB, AOF
Info stat	Information of connection, command processing
Info replication	Information of Master/Slave replication
Info cpu	Statistics of CPU usage
Info commandstats	Statistics of command usage of redis
Info cluster	Information of cluster
Info keyspace	Information of keyspace

2) 환경 설정 파일을 통한 정보 수집

데이터베이스 운용에 관한 전반적인 정보는 환경 설정 파일에서 확인할 수 있다. 명령어를 통해서도 데이터베이스 운용에 관한 정보를 수집할 수 있지만, 환경 설정 파일에서는 명령어를 통해 얻을 수 없는 정보와 세부적인 옵션을 파악할 수 있다. Table 4는 데이터베이스 구조 파악을 위한 주요 구동 옵션과 세부 옵션을 요약한 것이다. General 옵션은 DB의 기본적인 운용정보를 포함하고 있다. 세부 옵션의 logfile과 loglevel은 Redis 서버의 상태를 나타내는 시스템 로그로 데이터베이스 운용에 관한 사항이 기록되는데 loglevel을 통해 남겨지는 로그의 범위를 조절할 수 있다. Snapshotting 옵션은 RDB 데이터 백업의 방식이나 경로, 파일명 등을 설정할 수 있다. 다음으로 Replication 옵션의 slaveof는 Master 노드의 IP, Port 정보를 획득할 수 있으며 Master 노드에서 데이터 동기화 시 Slave 노드의 인증을 위한 인증 패스워드를 masterauth 옵션에서 확인할 수 있다. 또한 Security 옵션의 requirepass는 Client가 Server의 인증을 위한 Password이다. Slow log의 slowlog-log-slower-than 옵션은 slow log에 기록될 쿼리의 기준 시간을 설정하는 옵션으로 설정된 실행 시간을 초과하는 쿼리들이 slow log에 기록되게 된다.

Table 4. Database Configuration

Main option	option	Description
General	Port	Service port
	ip	Bind IP address
	loglevel [level]	Define Log Level
	logfile [path]	Define Logfile path
	databases [number]	Database Count
Snapshotting	dbfilename [filename]	Define RDB filename
	dir [directory path]	Define RDB save path
Replication	slaveof [master ip] [master port]	Master node setting
	masterauth [password]	marset node password
Security	requirepass [password]	server authentication password
Limits	maxmemory [bytes]	limits memory
Append only mode	appendfilename [filename]	Define AOF filename
Slow log	slowlog-log-slower-than [microseconds]	Define slow execution time

4.2 데이터 수집 및 분석

1) 데이터 파일 수집

Redis는 데이터베이스를 메모리에 로드하여 데이터를 처리하기 때문에 빠른 속도를 보장한다. 따라서 모든 데이터는 메모리상에 존재하기 때문에 메모리에 존재하는 데이터베이스를 획득하여야 한다.

메모리에 로드된 데이터와 디스크에 파일로 저장한 형태의 데이터가 다를 수 있는데, 이는 데이터 저장을 위해 사용되는 RDB 방식과 AOF 방식의 시점 설정 차이에 의해 메모리에 갱신된 내용이 디스크에 반영되지 않아 발생한 것이다. 이러한 경우에는 현재 메모리에 로드된 데이터의 수집을 통해 수집된 데이터와 디스크에 저장된 데이터를 비교하여 생성·변경·삭제된 데이터를 파악할 수 있다.

데이터 파일 수집은 물리 메모리에 존재하는 데이터베이스 수집과 디스크에 저장된 데이터 파일 수집으로 구분된

다. 물리 메모리에 존재하는 데이터 수집은 Redis Client 명령어인 'BGSAVE' 명령어를 통해 데이터를 수집할 수 있다. 이 명령어는 현재 시점에 존재하는 메모리상의 데이터를 RDB 파일로 저장하는 기능을 수행한다. 물리 메모리 덤프를 수행하지 않고, Redis Client 명령어를 사용하여 데이터를 수집하는 이유는 리눅스의 경우 배포판 또는 커널 버전에 따라 메모리 구조가 상이하고 조사 대상 PC의 별도의 프로그램을 설치하여 실행할 경우 대상 메모리에 저장된 데이터베이스의 무결성을 해칠 수 있기 때문이다.

데이터 파일은 4.1 운용 환경 파악의 환경 설정 파일을 통한 정보 수집 절에서 저장경로를 확인하여 데이터가 저장되는 RDB 파일과 AOF 파일을 획득할 수 있다.

Fig. 11은 Redis의 특성을 반영한 데이터 수집 방안이다. 이는 전체적인 조사 절차의 Fig. 9의 Acquire Database 부분에 해당하며, 디지털 포렌식 조사 시 데이터 파일이 필요하지 않은 경우 생략 가능하다. 데이터 파일의 경우 파일 시스템의 백업 경로에 존재하기 때문에 복사를 통해 파일을 획득할 수 있다. 만약 사용자가 RDB, AOF 방식을 통해 데이터 백업 기능을 사용하지 않고, 데이터 파일이 백업 경로에 존재하지 않는 경우, BGSAVE 명령어를 통해 메모리에 있는 데이터를 디스크에 파일로 저장하거나 환경 설정 파일에서 RDB 옵션을 활성화시키는 방법을 이용해 데이터를 수집할 수 있다.

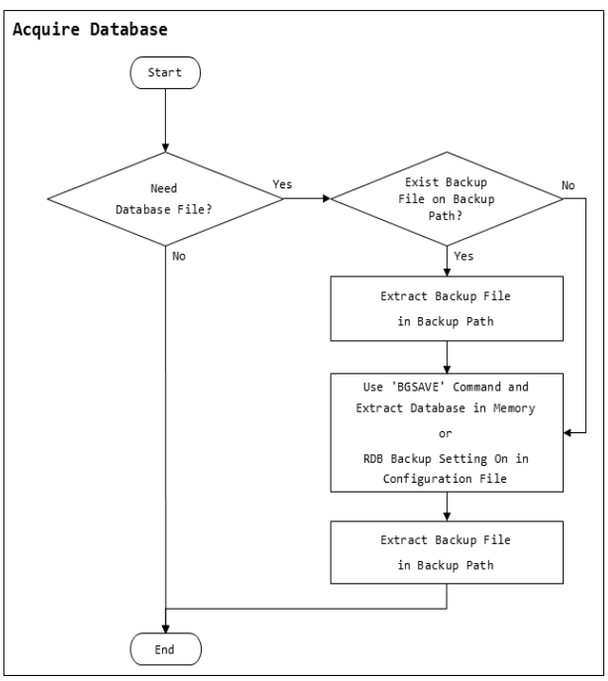

Fig. 11. Data Collection Method

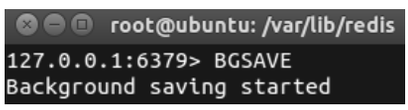

Fig. 12. Command : BGSAVE

2) 환경 설정 파일 수집

환경 설정 파일의 경로는 4.1 운용 환경 파악의 명령어를 통한 정보 수집 절에서 저장경로를 확인할 수 있다. 기본 경로는 '/etc/redis'이며 'redis.conf' 파일명으로 저장되어 있다. 또한 데이터 파일과 마찬가지로 복사를 통해 파일을 획득할 수 있다. 환경 설정 파일을 수집하여 조사 대상의 환경과 동일한 환경을 가지는 분석용 Redis 서버를 제작할 수 있으며, 이를 통해 상세분석이 가능하다.

4.3 로그 파일 수집 및 분석

Redis에서는 데이터베이스의 운용 상태를 메시지 형태로 기록하는 syslog와 설정된 실행 시간을 초과하는 쿼리를 기록하는 slow log가 존재한다.

1) syslog

syslog는 리눅스의 시스템 로그 파일명과 동일하지만 다른 Redis 만의 독자 로그 파일이다. syslog는 5초 마다 로그가 기록되는데 기본 저장 경로는 '/var/log/redis/'이며 'redis-server.log' 파일명으로 저장된다. 이는 환경 설정 파일에서 재설정할 수 있다. syslog는 환경 설정 파일에서 구동 옵션을 조절하여 남기는 로그의 범위를 지정할 수 있는데 Table 5와 같이 Level에 따라 남겨지는 정보가 상이하다.

Table 5. Syslog Level

Log Level	Description
Warning	Server start·end, Critical message
Notice(Default)	(Include Warning), Server State, Backup time
Verbose	(Include Notice)
Debug	(Include Verbose) client, slave node connection info, Database Key count in Database, Expire Key Count

Warning Level은 서버의 시작·종료, 서버 위험 메시지와 같은 최소한의 정보가 기록되며 Notice Level은 Log Level의 기본 값으로 Warning Level이 남기는 정보를 포함하고 서버의 상태나 데이터 백업의 시점과 상태를 추가로 기록한다. Verbose Level의 경우 Notice Level의 정보를 포함하여 추가적인 정보를 남긴다고 알려져 있지만 실제 Redis에서 Log Level을 Verbose Level로 지정할 경우 Redis는 동작하지 않으며, 다른 Level로 지정하여 시작하라는 메시지를 보여준다. Debug Level의 경우 Client나 Slave 노드의 연결 정보, 데이터베이스에 존재하는 키 개수, 사용 용량, 삭제될 키의 개수 등 데이터베이스 데이터 입·출력 정보 등이 자세하게 기록된다. 이는 데이터베이스 개발 및 테스트 시 사용되는 Level로 가장 많은 정보를 남긴다.

Fig. 13은 Notice Level syslog의 예시로 서버시작, DB 로드 정보, RDB 백업 상황을 확인할 수 있다. 이와 같이 조사자는 syslog를 통해 DB 운용 상황을 파악할 수 있다.

```
[1318] 26 Jan 15:45:53.903 # Server started, Redis version 2.8.4
[1318] 26 Jan 15:45:53.904 # WARNING overcommit_memory is set to 0! Background save may fail
under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to
/etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this
to take effect.
[1318] 26 Jan 15:45:53.946 * DB loaded from disk: 0.042 seconds
[1318] 26 Jan 15:45:53.946 * The server is now ready to accept connections on port 6379
[1318] 27 Jan 09:04:50.135 * 1 changes in 900 seconds. Saving...
[1318] 27 Jan 09:04:50.165 * Background saving started by pid 4495
[4495] 27 Jan 09:04:50.167 * DB saved on disk
[4495] 27 Jan 09:04:50.167 * RDB: 6 MB of memory used by copy-on-write
[1318] 27 Jan 09:04:50.266 * Background saving terminated with success
[1318] 27 Jan 09:19:51.041 * 1 changes in 900 seconds. Saving...
[1318] 27 Jan 09:19:51.226 * Background saving started by pid 4858
[4858] 27 Jan 09:19:51.348 * DB saved on disk
[4858] 27 Jan 09:19:51.349 * RDB: 6 MB of memory used by copy-on-write
[1318] 27 Jan 09:19:51.444 * Background saving terminated with success
[1318 | signal handler] (1453939506) Received SIGTERM, scheduling shutdown...
[1318] 28 Jan 09:05:06.662 # User requested shutdown...
[1318] 28 Jan 09:05:06.662 * Saving the final RDB snapshot before exiting.
[1318] 28 Jan 09:05:06.667 * DB saved on disk
```

Fig. 13. Example of syslog Level : Notice

2) slow log

slow log는 환경설정 파일의 slowlog-log-slower-than 옵션의 지정시간보다 실행시간이 길어질 경우 기록되는 로그로 Redis 서버에서 slowlog get 명령어를 통해 쿼리 실행 시간을 체크할 수 있어 데이터베이스 개발 및 테스트에 활용되고 있다.

Fig. 14의 Log Identifer는 저장된 쿼리의 고유 식별자이며 Unix Timestamp는 쿼리가 저장된 시각을 나타낸다. 이후 쿼리 실행에 소요된 시간, 실행된 쿼리가 Array 형태로 나타나게 된다.

```
127.0.0.1:6379 : slowlog get
1)(integer) Log Identifier
2)(integer) Unix Timestamp
3)(integer) Query Execution Time
4) Saved Query
```

Fig. 14. slowlog

4.4 삭제된 데이터 복구

데이터베이스에서 삭제된 데이터를 복구하는 것은 중요하다. 삭제된 데이터 복구를 통해 손실된 데이터를 복구할 수 있으며, 용의자의 고의적 데이터 삭제에 대응할 수 있기 때문이다.

RDB 파일의 경우 사용자가 지정한 시점마다 이전의 RDB 파일은 삭제하고 새로운 RDB 파일을 생성하여 현재 시점의 메모리의 데이터를 저장하기 때문에 삭제된 데이터의 복구 가능성이 낮다. 따라서 본 논문에서는 AOF 파일을 이용하여 삭제된 데이터를 복구하는 방안을 제시한다.

AOF 파일에는 Redis 운용 간 사용했던 명령어가 모두 기록돼 있기 때문에, 명령어를 순차적으로 실행하여 삭제된 데이터를 판별할 수 있다. Fig. 15는 AOF 파일에 기록된 쿼리를 통해 데이터를 재구성하는 모습을 보여준다. 데이터의 삽입이나 변경 사항을 모두 기록하고 삭제된 명령어가 나타나면 해당 Key-Value Pair는 삭제된 데이터로 판별할 수 있다.

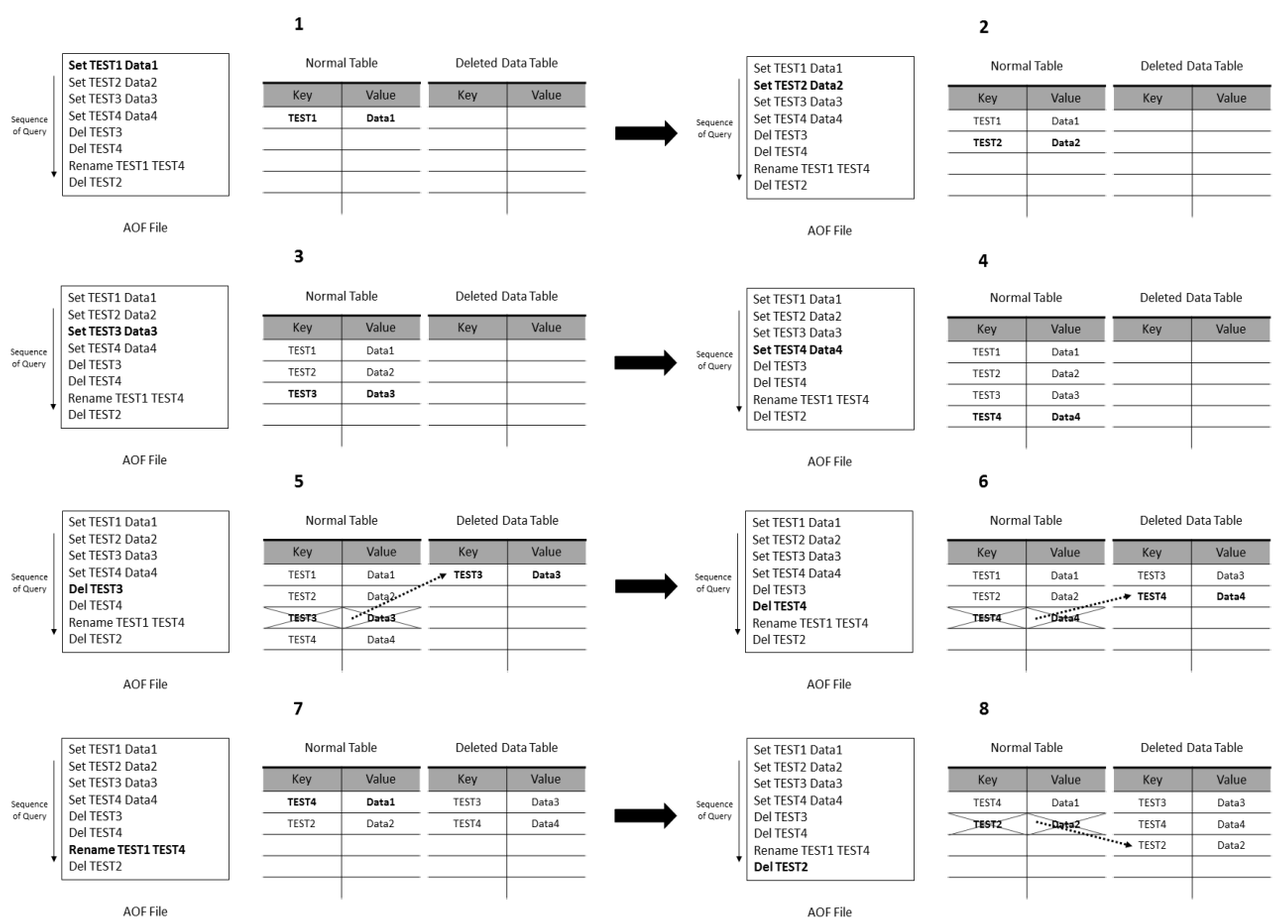

Fig. 15. Data Reconstruction through AOF File

Fig. 16의 알고리즘은 AOF 파일 구조를 해석하여 각 명령어를 추출하고, 명령어의 의미를 해석하여 정상 테이블과 삭제된 데이터 테이블에 분류하는 방안이다. 제시된 알고리즘을 통해 삭제된 데이터를 판별하여 복구할 수 있다.

제시된 알고리즘의 복구 가능성을 검증하기 위해 도구를 구현하였다. 도구는 AOF 파일을 입력 받아 String 자료형의 삭제된 데이터를 복구하고 SQLite 데이터베이스 형태로 출력하도록 구현하였다. Redis Client에서 Table 6의 String 자료형 명령어를 사용하여 테스트 세트를 제작하였다. 총 200개의 데이터를 생성하고 수정하였으며, 그중 25%에 해당하는 50개의 데이터를 삭제하였다.

Table 6. Command List for Experiment

Operation	Description
SET	Data Insert, Update
DEL	Data Delete
RENAME	Key Rename, Data Insert
APPEND	Data Insert, Data Append
DECR	Int type data -1
DECRBY [INT]	Deduct user defined number from Int type data
INCR	Int type data +1
INCRBY [INT]	Add user defined number to Int type data

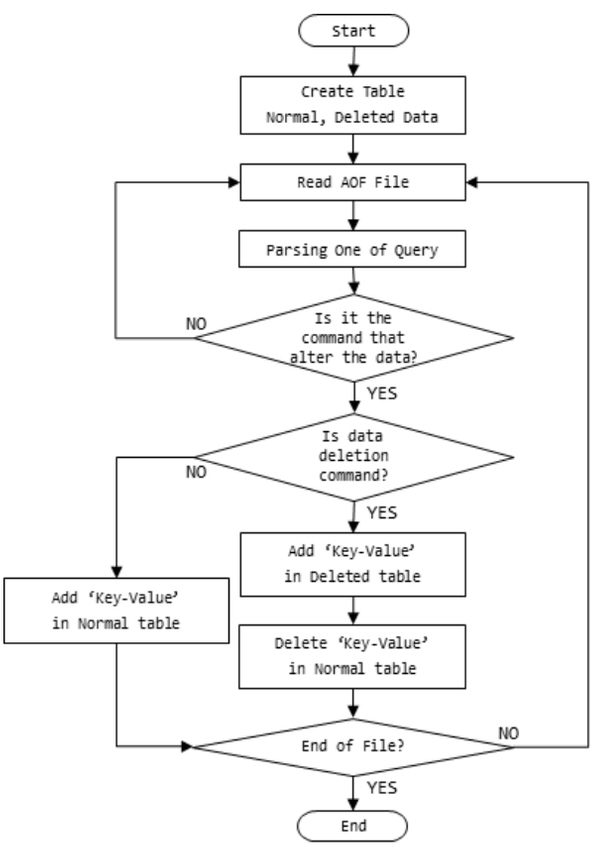

Fig. 16. Data Recovery Method

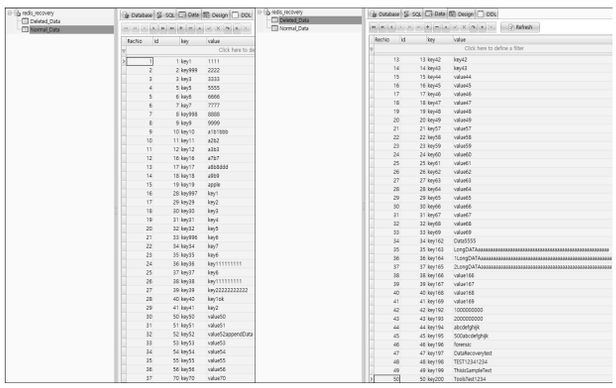

Fig. 17. Tools Result : SQLite DB

```
        1-th Argument : del
        2-th Argument : key193
Argument Count : 2
        1-th Argument : del
        2-th Argument : key194
Argument Count : 2
        1-th Argument : del
        2-th Argument : key195
Argument Count : 2
        1-th Argument : del
        2-th Argument : key196
Argument Count : 2
        1-th Argument : del
        2-th Argument : key197
Argument Count : 2
        1-th Argument : del
        2-th Argument : key198
Argument Count : 2
        1-th Argument : del
        2-th Argument : key199
Argument Count : 2
        1-th Argument : del
        2-th Argument : key200


 RART(RedisAOFfileRecoveryTools) Result
 Deleted Data Count : 50
Recovery Data Count : 50
     Recovery Ratio : 100%
```

Fig. 18. Data Recovery Result

Fig. 18과 같이 삭제된 데이터 복구 결과 총 50개의 데이터가 모두 복구되는 것을 확인할 수 있었으며, 이외에 다수의 반복 실험에서도 DEL 명령어를 통해 삭제된 Key가 모두 복구되는 것을 확인하였다. 도구의 최종 출력물은 SQLite DB파일이다. DB 파일은 일반 데이터 테이블과 삭제된 데이터 테이블로 구성되어 있으며 Fig. 17과 같이 삭제된 데이터 테이블에서 데이터를 확인할 수 있으며 SQL 질의문을 통해서도 데이터 검색이 가능하다.

5. 결론 및 향후계획

본 논문에서는 메모리 기반의 NoSQL DBMS인 Redis를 소개하고 디지털 포렌식 관점에서 살펴보아야 할 아티팩트를 수집, 분석 방안, 삭제된 데이터 복구 기법을 제안하였으며, 이를 토대로 Redis 데이터베이스 디지털 포렌식 조사 기법을 제안하였다. 제안된 복구 기법을 통해 구현된 도구는 삭제된 데이터를 모두 복구하는 것을 확인하였으며, 이는 추후 Redis 데이터베이스 조사 시 분석관에게 데이터 분석에 많은 도움이 될 것이다.

Redis는 데이터가 저장되는 방식이나 데이터베이스가 동작하는 방식은 관계형 데이터베이스에 비해 직관적이며, 단순한 형태의 구조를 가지고 있다. 다만 다수의 복제서버가 존재할 수 있고, 일반적인 NoSQL DBMS와 다르게 저장매체의 특수성에 따라 메모리 기반의 DBMS이기 때문에 분석 시 메모리에 로드된 데이터와 디스크에 저장된 데이터를 모두 수집하는 방안이 고려되어야 한다.

향후에는 스키마 구조가 없는 Key-Value Store의 NoSQL 데이터베이스에서 의미 있는 데이터 추출 및 분석 방안을 연구하고 논문에서 구현된 도구의 자료형 추가 후 오픈소스로 공개할 예정이다.

References

[1] National Information Society Agency, "*Big Data era which opens a new future*," revised ed., Seoul: NIA, 2013.

[2] J. H. Kwon, "The latest trend of NoSQL database," *The KIPS*, Vol.22, No.4, pp.35-47, 2015.

[3] K. S. Lim, D. C. Lee, J. H. Park, and S. J. Lee, "A Novel Database Forensic Technique Using Table Relationship Analysis," *Korea Multimedia Society Fall Conference Proceedings*, pp.65-68, 2009.

[4] D. C. Lee and S. J. Lee, "Research of Organized Data Extraction Method for Digital Investigation in Relational Database System," *Journal of The Korea Institute of Information Security and Cryptology*, Vol.22, No.3, pp.565-573, 2012.

[5] Paul M. Wright, "Oracle database forensics using LogMiner," in *Proceedings of the GIAC SANS Institute*, Ed., 2005.

[6] P. Fruhwirt, P. Kieseberg, S. Schrittwieser, M. Huber, and E. Weippl, "InnoDB Database Forensics : Reconstructing Data Manipulation Queries from Redo Logs," in *Proceedings of the IEEE Availability, Reliability and Security*, Prague, 2012, pp.625-633.

[7] J. H. Choi, D. W. Jung, and S. J. Lee, "The Method of Recovery for Deleted record in Oracle Database," *Journal of The Korea Institute of Information Security and Cryptology*, Vol.23, No.5, pp.947-955, 2013.

[8] James Wagner, Alexander Rasin, and Jonathan Grier, "Database forensic analysis through internal structure carving," in *Proceedings of the Fifteenth Annual DFRWS Conference*, Philadelphia, 2015, pp.S106-S115.

[9] J. S. Yoon, D. W. Jung, C. H. Kang, and S. J. Lee, "Digital Forensic Investigation of MongoDB," *Journal of The Korea Institute of Information Security and Cryptology*, Vol.24, No.1, pp.123-134, 2014.

[10] Ming Xu, Xiaowei Xu, Jian Xu, Yizhi Ren, Haiping Zheng, and Ning Zheng, "A Forensic Analysis Method for Redis Database Based on RDB and AOF File," *Journal of Computers*, Vol.9, No.11, pp.2538-2544, 2014.

[11] DB-ENGINES Ranking [Internet], http://www.http://db-engines.com/en/ranking.

[12] Eric Redmond and Jim R. Wilson, "Seven Databases in Seven Weeks," ed., Dallas, Texas·Raleigh, North Carolina: The Pragmatic Bookshelf, 2012

[13] Redisgate [internet], http://www.redisgate.com/.

[14] Redis-rdb-tools [internet], https://github.com/sripathikrishnan/redis-rdb-tools/wiki/Redis-RDB-Dump-File-Format.

최 재 문

e-mail : jaemunks@korea.ac.kr
2015년 호서대학교 정보보호학과(학사)
2015년~현 재 고려대학교 사이버국방학과 석사과정
관심분야 : Digital Forensic, Reverse Engineering

정 두 원

e-mail : dwjung77@korea.ac.kr
2011년 고려대학교 산업경영공학과(학사)
2011년~현 재 고려대학교 정보보호학과 석·박사통합과정
관심분야 : Digital Forensic, Information Security, Big Data Analysis

윤 종 성

e-mail : yoonjs53@naver.com
2005년 공군사관학교 전산과학과(학사)
2013년~현 재 고려대학교 정보보호학과 석·박사통합과정
관심분야 : Digital Forensic, Information Security

이 상 진

e-mail : sangjin@korea.ac.kr
1987년 고려대학교 수학과(학사)
1989년 고려대학교 수학과(석사)
1994년 고려대학교 수학과(박사)
1989년~1999년 ETRI 선임연구원
1999년~현 재 고려대학교 교수
2008년~현 재 고려대학교 디지털포렌식연구센터 센터장
관심분야 : Digital Forensic, Steganography, Hash Function