

Memcached 데이터베이스 디지털 포렌식 조사 기법 연구: 안티-포렌식 대응방안을 중점으로

윤 호*, 김 준 호**, 김 성 배***, 양 양 한***, 이 상 진****
고려대학교 일반대학원 정보보안학과 (대학원생)*,
고려대학교 정보보호대학원 정보보호학과 (대학원생)**, (교수)****
대검찰청 디지털수사과 (수사관)***

Digital Forensic Investigation of Memcached Database: Countering Anti-Forensics

Ho Yoon*, Junho Kim**, Sungbae Kim***, Yanghan Yang***, Sangjin Lee****
Dept. of Information Security, Korea University (Graduate Student)*,
School of Cybersecurity, Korea University (Graduate Student)**, (Professor)****
Digital Forensic Division, Supreme Prosecutors' Office (Investigator)***

요 약

비관계형 데이터베이스는 빅데이터 기반 서비스를 제공하는 기업들에 의해 많이 도입되어 사용되고 있다. 하지만, 사용량이 상대적으로 미비한 일부 비관계형 데이터베이스에 대한 디지털 포렌식 조사 기법이 부족한 상황이다. 이에, 본 논문에서는 데이터베이스를 위한 인메모리 솔루션이자 비관계형 Key-Value 모델인 Memcached의 디지털 포렌식 조사 기법을 제안한다. Memcached의 운영 환경 정보 수집 방법, 데이터 수집 및 분석 방법을 살펴보고, Memcached에 대한 안티-포렌식 대응 방안으로 로그 분석을 통해 안티-포렌식 행위를 탐지하고 메모리로부터 삭제 및 변경된 원본데이터를 확인하는 기법을 제안한다.

주제어 : Memcached, Key-value Database, 비관계형 데이터베이스, 데이터베이스 포렌식, 디지털 포렌식

ABSTRACT

Non-relational databases(NoSQL) are widely used by companies providing big data-oriented services. Nevertheless, only a few studies have been placed on non-relational databases from the digital forensics perspective. This paper proposes a digital forensics investigation technique of Memcached, the key-value model of NoSQL in-memory solution for a database. Techniques for data collection, anti-forensic detection using logs, and discovering original data through memory analysis are covered.

Key Words : Memcached, Key-value Database, NoSQL, Database Forensics, Digital Forensics

1. 서 론

인터넷 기술이 발전함에 따라 하루에 생성되는 디지털 데이터의 양은 폭발적으로 증가하고 있다. 이에, 정부, 기업, 기관은 방대한 양의 디지털 데이터를 보다 효과적으로 저장 및 관리할 필요성이 증가하였다. 그리고 새로운 서비스 및 애플리케이션이 생겨남에 따라 처리할 데이터의 형태도 다양화되고 있다. 하지만, 기존 관계형 데이터베이스(RDB: Relational Database)는 다양한 형태의 데이터를 처리하는데 한계가 있고 확장성이 떨어지며, 방대한 양의 데이터를 처리 시 속도가 저하되는 문제가 존재한다. 이에 대한 대안으로 빅데이터를 처리하는데 적합한 비관계형 데이터베이스(NoSQL: Not Only SQL)가 빠른 속도로 도입되고 있다 [1].

• Received 26 February 2022, Revised 30 May 2022, Accepted 27 May 2022
• 제1저자(First Author) : Ho Yoon (Email : ymaul2@korea.ac.kr)
• 교신저자(Corresponding Author) : Sangjin Lee (Email : sangjin@korea.ac.kr)

한국데이터산업진흥원의 “2020 데이터산업현황조사 보고서”[2]에 따르면 국내 데이터 산업 규모가 가파르게 성장하고 있음을 엿볼 수 있다. 2010년 8조 6374억원 규모에서 매년 성장하여 2019년에는 2배에 근접한 16조 8,582억원을 기록하였으며, 2018년~2020년 3개년 평균 증가율은 11.3%를 기록하였다. 이처럼, 국내의 데이터 산업이 성장함에 따라 처리되는 누적 데이터량은 점차 증가하여 빅데이터 관리에 보다 효율적인 비관계형 데이터베이스의 사용량 또한 증가할 것으로 예상된다.

이러한 수요에 따라 비관계형 데이터베이스는 현재 그 종류가 수백가지에 이르며, Key-Value 모델의 경우 2021년 11월 기준 64개가 존재한다 [3]. 하지만, 각 모델별 사용량이 가장 많은 MongoDB, Redis, Cassandra 등을 제외한 비관계형 데이터베이스에 대한 디지털 포렌식 연구는 미비한 상태이다 [4].

본 논문에서는 인메모리 저장소이자 비관계형 Key-Value 모델인 Memcached의 디지털 포렌식 조사 기법을 제안한다. 구체적으로, 업데이트된 1.4.31 버전부터 새롭게 추가된 명령어를 활용하여 활성 상태에서 데이터를 수집하는 방식을 제안한다. 또한, 의도적인 시스템 종료 및 데이터 삭제·변조와 같은 안티-포렌식 행위가 발생했을 때 로그를 통해 사용자 행위와 삭제된 데이터를 탐지하는 방법을 제시하며, 메모리를 수집 및 분석함으로써 로그 파일에서 확인한 Key를 토대로 삭제·변조된 데이터에 접근하여 각 Key에 상응하는 원본 값(Value)을 추적·확인하는 기법을 제안한다.

II. 관련 연구

Min Xu et al[5]는 비관계형 데이터베이스 중 인메모리 데이터베이스이자 Key-Value 모델인 Redis의 백업파일을 활용한 데이터 분석 기법을 제시하였다. 저자는 메모리상에 존재하는 데이터를 추출하여 분석하는 방식과 달리 백업파일을 수집하여 분석하였을 때 보다 간편하고 효과적으로 데이터를 분석할 수 있다고 서술하였다. 가령 데이터가 삭제된 경우에도 RDB(Redis Database Backup File) 및 AOF(Append Only File) 파일과 같은 백업 파일에 데이터가 남아있을 가능성과 이를 활용한 삭제된 데이터 복구 가능성을 제시하였다. 한편, Fmem 도구를 이용하여 메모리를 분석한 결과 메모리 포맷이 RDB 백업 파일의 구조와 일치한다는 점을 서술하였다. 본 연구에서는 메모리 덤프 도구인 “ProcDump”를 사용하여 Memcached 프로세스를 덤프한 후 삭제·변조된 원본 데이터 확인을 시도하였다.

Choi Jaemun et al[6]은 Redis의 디지털 포렌식 조사 기법에 대한 연구를 진행하였다. 운영 환경 및 구조 파악, 데이터 수집 및 분석, 로그 파일 수집 및 분석, 삭제된 데이터 복구 등 총 4차례에 걸친 조사 절차를 제시하였다 [표 1]. 각 과정을 Memcached와 대입·비교하면 [표 2]와 같다.

〈Table 1〉 The procedure for Redis database forensic investigation

목표	설명
운영 환경 및 구조 파악	명령어를 통한 정보 수집 환경 설정 파일을 통한 정보 수집
데이터 수집 및 분석	데이터 파일 수집 환경 설정 파일 수집
로그 파일 수집 및 분석	syslog, slow log 파일 분석
삭제된 데이터 복구	AOF 파일을 활용한 복구 방안

〈Table 2〉 Applying redis database forensic investigation procedure into Memcached

목표	Memcached	Redis
운영 환경 및 구조 파악	-명령어를 통한 정보 수집 가능 -환경 설정 파일을 통한 정보 수집 가능	-명령어를 통한 정보 수집 가능 -환경 설정 파일을 통한 정보 수집 가능
데이터 수집 및 분석	-백업 기능 및 데이터 파일 존재하지 않음	-백업 기능 및 데이터 파일 존재
로그 파일 수집 및 분석	로그 파일이 수행된 명령어 기록 (Log Level “vv” 기준)	로그 파일이 시스템 운용 관련 사항만 기록
삭제된 데이터 복구	인메모리 솔루션 특성상 삭제된 데이터 복구가 어려움	백업 파일을 이용한 복구 가능

첫 번째 과정은 운영 환경 및 구조 파악 단계이다. 클라이언트에서 명령어를 기반으로 한 정보 수집 방식과 환경 설정 파일을 직접 열람하는 정보 수집 방식을 제시하고 있다. 저자는 Redis에서 각각의 방식을 통해 획득할 수 있는 정보가 상이하다고 서술했다. Memcached의 경우에도 환경 설정 파일을 통한 정보 수집 방식과 Telnet에 접속하여 명령어로 정보를 수집하는 방식이 각각 획득 가능한 정보가 상이하다는 공통점이 있다.

두 번째 과정은 데이터 수집 및 분석 단계이다. Redis는 메모리에 존재하는 데이터를 디스크에 백업하는 기능이 존재한다. 반면, Memcached는 Redis와 달리 백업 기능이 존재하지 않기 때문에 백업 파일이나 데이터 파일이 존재하지 않는다.

세 번째 과정은 로그 파일 수집 및 분석 단계이다. Redis와 Memcached는 모두 로그 파일이 존재한다. 하지만 Redis는 시스템 운용과 관련한 로그가 주로 기록되며, 사용자가 수행한 명령어는 백업 기능 중 하나인 AOF가 사실상 그 기능을 대체하여 기록하고 있다. 반면, Memcached에서는 Log Level에 관계 없이 Memcached 시작 및 종료와 같은 시스템 로그가 기록되며, Log Level이 가장 높은 단계로 설정된 경우에는 사용자가 수행한 명령어가 기록된다는 특징이 있다.

마지막 과정은 삭제된 데이터 복구 단계이다. Redis는 AOF 파일을 이용한 복구 방법을 제시하고 있다. 반면, Memcached는 데이터를 메모리에 저장하는 특성상 휘발성 성격을 가지고 있으며 별도의 디스크 쓰기 방식의 백업 기능이 존재하지 않는다. 따라서, 한 번 삭제된 데이터를 복구하는데 어려움이 존재한다.

Choi Jaemun[7]은 Memcached에 대한 디지털 포렌식 조사 기법을 제시하였다. 첫째로 명령어와 환경 설정 파일을 수집·분석하여 운영 환경을 파악하는 방식을 서술하였으며, 둘째로 “cachedump” 명령어와 코드를 통한 자동화 방식으로 데이터를 선별 수집·분석하는 방식을 제시하였다. 셋째로, Memcached 메모리 덤프 도구를 활용하여 메모리 내 존재하는 데이터 전체를 수집하는 방식을 제시하였다. 하지만, “cachedump” 명령어를 통한 데이터 수집 방식은 시스템에 영향을 줄 수 있으며, 조회 가능한 Key의 개수가 한정되어 있다는 한계점이 있다 [8]. 따라서 본 논문에서는 상기 문제점을 개선한 Memcached 1.4.31 버전부터 추가된 명령어를 활용한 데이터 수집 방식을 새롭게 제안한다.

Han Kyusik et al[9]은 인메모리 솔루션에 메모리 포렌식 기법을 적용한 데이터 접근·분석 기법을 제시하였다. 분석 방식에는 메모리 분석 도구인 Volatility가 활용되었으며, 문자열 검색 방식으로 데이터를 분석하였다. 본 연구에서는 Memcached에 데이터 삭제·변조와 같은 안티-포렌식 행위가 발생했을 때 로그와 메모리 포렌식 기법을 통해 안티-포렌식 행위를 탐지하고 원본 데이터 값(Value)을 확인하는 기법을 제안한다.

III. Memcached

Memcached는 오픈 소스 기반의 Key-Value 인메모리 솔루션으로 2003년 Brad Fitzpatrick에 의해 개발되었다. Linux와 BSD 계열의 운영 체제를 지원하며, 데이터는 문자열만 지원한다 [10]. 인메모리 저장소의 장점으로 디스크 저장소 대비 데이터 처리 속도가 빠른 반면 데이터가 메모리에 저장되기 때문에, 시스템 전원이 종료되었을 경우 데이터가 손실될 수 있다. Redis와 Memcached의 특성을 비교하면 [표 3]과 같다.

〈Table 3〉 Memcached and Redis comparison

항목	Memcached	Redis
지원하는 데이터 타입	String	String, Set, Sorted Set, Hash, List
데이터 저장	메모리에만 저장	메모리와 디스크에 저장
데이터 지속성 및 복구	지원하지 않음	RDB 스냅샷 및 AOF 로그를 통해 데이터의 지속성 및 백업을 지원
클러스터(샤딩)	지원하지 않음	지원
복제	지원하지 않음	지원
로그파일	수행된 명령어 기록 (Log Level “vv” 기준)	시스템 운용 관련 사항 기록

3.1 Memcached 주요 파일 및 아티팩트 소개

3.1.1 환경 설정 파일

“/etc/memcached.conf”는 Memcached의 환경 설정 파일로 로그, 네트워크와 같이 여러 설정 정보들을 확인할 수 있다. 현재 Memcached 운영 관련 현황을 나타내주는 “stats” 명령어를 통해서도 확인할 수 없는 정보들을 저장하고 있다는 점에서 우선적으로 수집되어야 하는 파일이다.

3.1.1.1 Log Level

memcached.conf 파일에서 로그 관련 설정 정보를 확인할 수 있는 영역이다. 사용자는 로깅의 강도 (strength)를 설정할 수 있으며 이렇게 생성된 로그는 “/var/log/syslog” 파일에 기록된다 [그림 1]. 설정된 Log Level에 따라 얼마나 상세한 정보를 기록하는지는 [표 4]와 같다.

```
# Be verbose
#-v

# Be even more verbose (print client commands as well)
#-vv
```

〈Figure 1〉 Log level of Memcached

〈Table 4〉 Logged information per log level

Log Level	설명
0(Default)	Memcached 시작 및 종료
v	0 단계의 로깅 + Memcached 운영상 발생한 오류 관련 로깅
vv	v 단계의 로깅 + 클라이언트에서 수행한 명령어 로깅

Log Level은 memcached.conf 파일을 수기로 임의 수정하여 주석을 추가하거나 삭제하는 방식으로 설정한다. [그림 1]의 경우 “v” 및 “vv”가 모두 주석 처리되어 default 값인 “0”으로 설정된 모습이다. [그림 2]의 경우 “vv” 앞에 주석이 해제됨에 따라서 Log Level이 “vv”로 설정되었음을 알 수 있다.

```
# Be verbose
#-v

# Be even more verbose (print client commands as well)
-vv
```

〈Figure 2〉 “vv” setting of log level

3.1.1.2 포트

Memcached가 사용하는 포트 정보를 확인할 수 있는 영역이다. Memcached는 default 값으로 11211번 포트를 사용하나 이는 사용자가 변경이 가능하다 [그림 3]. 따라서 Memcached에 직접 접속이 필요한 경우 사전에 포트 정보를 필히 확인하여야 한다.

```
# Default connection port is 11211
-p 11211
```

〈Figure 3〉 Connection port information

3.1.1.3 IP

Memcached는 외부에서 접속 가능한 IP를 설정하는 영역이 존재한다. default 값으로 로컬호스트를 지칭하는 “127.0.0.1”이 기재되어 있다 [그림 4].

```
# Specify which IP address to listen on. The default is to listen on all IP addresses
# This parameter is one of the only security measures that memcached has, so make sure
# it's listening on a firewalled interface.
-l 127.0.0.1
```

〈Figure 4〉 Default IP setting of Memcached

3.1.2 syslog

“/var/log/syslog”는 Memcached의 로그가 기록되는 파일이다 [그림 5]. 로그는 기본적으로 시간 정보와 함께 [표 4]에서 서술한 Log Level 설정값에 따라 기록된다. [그림 2]와 같이 Log Level이 가장 높은 단계인 “vv”로 설정되었을 경우 사용자가 수행한 명령어가 Key와 함께 화면에 출력되는 형태로 로그에 남기 때문에 이를 토대로 사용자 행위를 분석할 수 있다 [그림 6].

```
kyleyon@ubuntu:/var/log$ ls
alternatives.log      dist-upgrade          hp
alternatives.log.1    dmesg                 installer
alternatives.log.2.gz dmesg.0               journal
apport.log            dmesg.1.gz           kern.log
apt                   dmesg.2.gz           kern.log.1
auth.log              dmesg.3.gz           kern.log.2.gz
auth.log.1           dmesg.4.gz           kern.log.3.gz
auth.log.2.gz        dmesg.log            kern.log.4.gz
auth.log.3.gz        dpkg.log.1           lastlog
auth.log.4.gz        dpkg.log.2.gz        memcached.log
bootstrap.log        faillog              openvpn
btmtp                fontconfig.log        private
btmtp.1              gdm3                 speech-dispatcher
cups                  gpu-manager.log      syslog
```

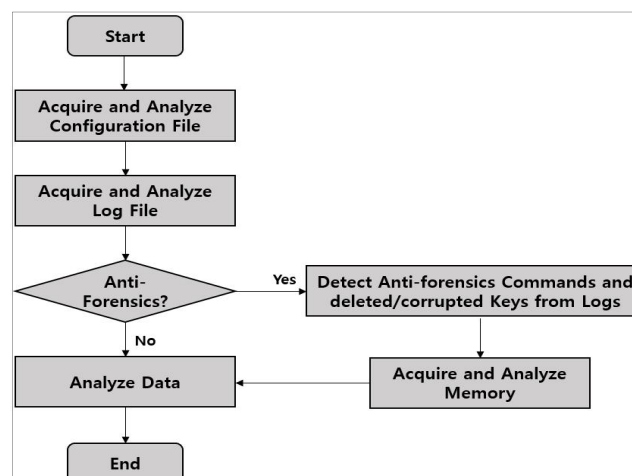
〈Figure 5〉 Syslog file located in /var/log

```
Nov 09 22:53:48 ubuntu systemd[1]: Started memcached daemon.
Nov 09 22:54:08 ubuntu systemd-memcached-wrapper[26766]: <27 set key_1 1 3600 7
Nov 09 22:54:09 ubuntu systemd-memcached-wrapper[26766]: >27 STORED
Nov 09 22:54:14 ubuntu systemd-memcached-wrapper[26766]: <27 delete key_1
Nov 09 22:54:14 ubuntu systemd-memcached-wrapper[26766]: >27 DELETED
Nov 09 22:54:17 ubuntu systemd-memcached-wrapper[26766]: <27 quit
Nov 09 22:54:17 ubuntu systemd-memcached-wrapper[26766]: <27 connection closed.
Nov 09 22:54:21 ubuntu systemd[1]: Stopping memcached daemon...
Nov 09 22:54:21 ubuntu systemd-memcached-wrapper[26766]: Signal handled: Terminated.
Nov 09 22:54:21 ubuntu systemd[1]: memcached.service: Succeeded.
Nov 09 22:54:21 ubuntu systemd[1]: Stopped memcached daemon.
```

〈Figure 6〉 Memcached logs from syslog file

IV. Memcached 디지털 포렌식 조사 기법

Memcached의 디지털 포렌식 조사 절차는 [그림 7]과 같다. 운영 정보 수집·분석, 로그 선별 수집·분석을 차례대로 수행한다. 수집된 로그 파일 분석 과정에서 의도적인 시스템 종료 행위가 파악된 경우 안티-포렌식 정황으로 의심할 수 있다. 만일, 특정 데이터가 삭제·변조된 행위가 파악된 경우 메모리로부터 Memcached 프로세스를 수집·분석하여 안티-포렌식이 적용된 데이터의 원본값 확인을 시도한다. 마지막으로 일반 데이터에 대한 분석을 진행한다. 각 과정의 자세한 설명은 다음과 같다.



〈Figure 7〉 The procedure for Memcached forensic investigation

4.1 운영 정보 수집·분석

Memcached의 운영 정보는 Memcached에 접속한 상태에서 “stats” 명령어를 통해 확인할 수 있다. [그림 8]은 Telnet을 통해 Memcached가 사용중인 포트(11211)에 접속하여 “stats” 명령어를 실행한 화면의 일부이다. “stats” 명령어로 확인할 수 있는 주요 정보를 정리하면 [표 5]와 같다.

```
kyleyon@ubuntu:~$ telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
stats
STAT pid 882
STAT uptime 877757
STAT time 1632732679
STAT version 1.5.22
```

〈Figure 8〉 Result of “stats” command in Telnet

〈Table 5〉 List of key findings of “stats” command

주요 요소	설명
pid	Memcached 프로세스 ID
uptime	Memcached가 가장 마지막 시작된 이후 경과된 시간(초)
version	Memcached 버전
cmd_flush	“flush all” 명령어가 수행된 횟수
curr_itmes	현재 Cache에 남아있는 유효한 items 개수
total_itmes	curr_items + 과거 삭제된 items 개수

4.2 로그 선별 수집·분석

본 절에서는 syslog에서 Memcached 로그를 선별 수집하는 과정을 소개한다. 기본적으로 syslog에는 Memcached 이외에 운영체제, 프로세스, 애플리케이션 등에서 생성된 다양한 로그들이 함께 존재한다 [11]. 이에, 터미널에서 “journalctl -u memcached.service” 명령어로 Memcached에 관한 로그만 선별적으로 조회가 가능하다.

이를 응용하여 “journalctl -u memcached.service > (filename)” 명령으로 현재 경로상에 존재하는 텍스트 파일에 Memcached 로그를 선별하여 저장할 수 있다. [그림 9]는 “memcached_logfile”이라는 이름을 가진 텍스트 파일을 생성하여, syslog에 존재하는 Memcached 로그들을 선별하여 저장한 모습이다.

```
kyleyon@ubuntu:~/memcached_log$ journalctl -u memcached.service > memcached_logfile
kyleyon@ubuntu:~/memcached_log$ ls
memcached_logfile
```

〈Figure 9〉 “memcached_logfile” containing memcached logs

로그가 수집된 파일을 열람하여 분석하면 시간 정보와 함께 수행된 명령어를 바탕으로 사용자 행위를 추적할 수 있다. 이때, 안티-포렌식 행위로 사용될 수 있는 데이터 삭제 및 변경 명령어를 정리하면 [표 6]과 같다.

〈Table 6〉 List of data deletion and modification commands

구분	명령	설명
Deletion	delete	단일 키 삭제
	flush_all	모든 키 삭제
Alteration	set	키 생성 및 기존 키의 값 변경
	replace	기존 키의 값 변경
	append	기존 키의 값 뒤에 값 추가
	prepend	기존 키의 값 앞에 값 추가
	incr/decr	기존 키의 (정수)값에 증가/감소
	cas	기존 키의 값 변경

4.3 안티-포렌식 행위 탐지

Memcached의 데이터 삭제·변조와 같은 안티-포렌식 행위 대응 방안으로 Memcached에서 지원하는 로그 기능을 활용할 수 있다. Log Level이 “0(Default)” 또는 “v”로 설정된 경우에는 Memcached 시작 및 종료 행위를 확인할 수 있으므로, 메모리에 저장되는 데이터의 성격상 휘발성이 강한 특징을 활용하여 고의적인 시스템 종료 행위를 추적할 수 있다. Log Level이 “vv”로 설정되었을 경우, 사용자가 특정 데이터를 대상으로 수행한 명령어가 시간정보와 함께 로그로 기록되어 보다 정확한 안티-포렌식 행위를 파악할 수 있다.

4.4 데이터 조회·분석

Memcached 1.4.31 버전부터 새로 업데이트된 기능(명령어)인 “lru_crawler metadump all”은 현재 메모리에 존재하는 Key 일체를 메타데이터와 함께 나타낸다 (그림 10). 각 메타데이터 요소가 지칭하는 바는 [표 7]과 같다. “exp(expiration)”는 데이터의 만료 시간, “la(last accessed)”는 데이터의 가장 최근 접근 시간을 의미하며, 데이터에 “get”, “replace”와 같이 명령어가 수행될 경우 해당 시간으로 동기화된다. 각 시간은 Unix Seconds로 표시된다.

```
kyleyoona@ubuntu:~$ telnet localhost 11211
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
lru_crawler metadump all
key=firstkey exp=1636535073 la=1636531474 cas=1002 fetch=no cls=1 size=75
key=secondkey exp=1636535078 la=1636531480 cas=1003 fetch=no cls=1 size=77
END
```

〈Figure 10〉 Result of “lru_crawler metadump all” command

〈Table 7〉 Explanation of each factor of the metadata

요소	설명
key	키 이름
exp	해당 Key 만료일
la(Last Accessed)	해당 Key 최근 접근 시간
cas(Check and set)	한쌍의 “Key-value” 데이터가 가진 고유 CAS ID
fetch	과거에 fetch* 되었는지 유무
cls	slab class id
size	크기(bytes)

조회한 Key에 상응하는 값(Value)을 조회하고자 할 경우 단일 데이터쌍은 “get [key]”, 복수의 데이터쌍은 “gets [key 1] [key 2] ...” 명령어로 확인이 가능하다. [그림 11]의 경우 “gets firstkey secondkey” 명령어 결과로 Key 값을 “firstkey”, “secondkey”로 가지는 Key에 상응하는 Value 값이 각각 “dfrc”, “korea”로 조회된 모습이다. 더불어, 조회한 Key의 Key 값과 메타데이터가 같이 나타나는데 메타데이터는 각각 데이터의 플래그, 크기, CAS ID 이다.

```
kyleyoona@ubuntu:~$ telnet localhost 11211
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
gets firstkey secondkey
VALUE firstkey 1 4 1
dfrc
VALUE secondkey 2 5 2
korea
END
```

〈Figure 11〉 Result of “gets” command

V. Memcached 안티-포렌식 행위 대응 방안 시나리오

본 절에서는 안티-포렌식 행위가 적용된 Memcached를 가정하여, 로그 파일을 통한 안티-포렌식 행위 탐지와 삭제·변조된 원본 데이터를 확인하는 방안을 제시한다. 본 실험의 환경은 [표 8]과 같다.

〈Table 8〉 Test environment

실험 환경	설명
가상 환경 도구	VMware workstation 16 PRO
분석 대상 환경	ubuntu-20.04.2.0
메모리 수집 도구	ProcDump v1.0.1
메모리 분석 도구	HxD

5.1 안티-포렌식 행위 탐지

Log Level이 “vv”로 설정되고, 일반적으로 사용되고 있는 활성상태의 Memcached에 존재하는 데이터 3쌍을 대상으로 안티-포렌식 행위 탐지 실험을 진행하였다. 안티-포렌식 행위로써 단일 데이터 변조(“set”), 단일 데이터 삭제(“delete”), 데이터 일괄 삭제(“flush_all”) 명령어가 수행된 후 종료되었다고 가정하기 위해, ①Memcached 실행(“sudo service memcached start”), ②데이터 생성(“set name_1(생략), name_2(생략), name_3(생략)”), ③데이터 조회(“gets name_1 name_2 name_3”), ④데이터 수정(“set name_1(생략)”), ⑤데이터 삭제(“delete name_2”), ⑥데이터 일괄 삭제(“flush_all”), ⑦텔넷(Telnet) 종료(“quit”) ⑧Memcached 종료(“sudo service memcached stop”) 과정을 차례대로 수행하였다 [그림 12].

```
kyleyoona@ubuntu:~$ sudo service memcached start
kyleyoona@ubuntu:~$ telnet localhost 11211
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
set name_1 1 3600 4
dfrc
STORED
set name_2 2 3600 9
forensics
STORED
set name_3 3 3600 5
korea
STORED
gets name_1 name_2 name_3
VALUE name_1 1 4 1
dfrc
VALUE name_2 2 9 2
forensics
VALUE name_3 3 5 3
korea
END
set name_1 1 3600 5
dfrcs
STORED
delete name_2
DELETED
flush_all
OK
quit
Connection closed by foreign host.
kyleyoona@ubuntu:~$ sudo service memcached stop
```

〈Figure 12〉 An example of Memcached with Anti-Forensics being taken

Syslog에 기록된 Memcached 로그의 수집 및 분석을 위해 현재 경로에 “log_collection”이라는 별도의 폴더를 생성, 해당 폴더로 이동한 후, “journalctl -u memcached.service >> memcached_logs” 명령어를 통해 “/var/log/syslog”상에 존재하는 Memcached 로그들을 선별하여 “memcached_logs” 파일에 텍스트 형태로 저장하였다 [그림 13].


```

kyleyoona@ubuntu:~$ mkdir log_collection
kyleyoona@ubuntu:~$ cd log_collection
kyleyoona@ubuntu:~/log_collection$ journalctl -u memcached.service >> memcached_logs
kyleyoona@ubuntu:~/log_collection$ ls
memcached_logs

```

〈Figure 13〉 A process of memcached logs being written and saved in a text file

이후, “memcached_logs” 파일을 열람한 결과 [그림 12]의 과정이 [그림 14]와 같이 시간정보와 함께 로그형태로 기록된 것을 확인할 수 있었다. 안티-포렌식 행위(④~⑥)도 상세히 기록되었는데 단일 데이터 삭제 및 변조 명령어가 행해진 Key(“name_1”, “name_2”)의 값을 로그에서 확인할 수 있었다. 데이터 일괄 삭제 명령어 “flush_all” 또한 시간정보와 함께 기록되었으며, Memcached 시작 및 종료 시간도 기록되었다.

```

Nov 04 19:13:54 ubuntu systemd[1]: Started memcached daemon.
Nov 04 19:13:54 ubuntu systemd-memcached-wrapper[12006]: <26 server listening (auto-negotiate)
Nov 04 19:13:58 ubuntu systemd-memcached-wrapper[12006]: <27 new auto-negotiating client connection
Nov 04 19:14:04 ubuntu systemd-memcached-wrapper[12006]: 27: Client using the ascii protocol
Nov 04 19:14:04 ubuntu systemd-memcached-wrapper[12006]: <27 set name_1 1 3600 4
Nov 04 19:14:04 ubuntu systemd-memcached-wrapper[12006]: >27 STORED
Nov 04 19:14:29 ubuntu systemd-memcached-wrapper[12006]: <27 set name_2 2 3600 9
Nov 04 19:14:31 ubuntu systemd-memcached-wrapper[12006]: >27 STORED
Nov 04 19:14:57 ubuntu systemd-memcached-wrapper[12006]: <27 set name_3 3 3600 5
Nov 04 19:14:58 ubuntu systemd-memcached-wrapper[12006]: >27 STORED
Nov 04 19:15:08 ubuntu systemd-memcached-wrapper[12006]: <27 gets name_1 name_2 name_3
Nov 04 19:15:08 ubuntu systemd-memcached-wrapper[12006]: >27 sending key name_1
Nov 04 19:15:08 ubuntu systemd-memcached-wrapper[12006]: >27 sending key name_2
Nov 04 19:15:08 ubuntu systemd-memcached-wrapper[12006]: >27 sending key name_3
Nov 04 19:15:08 ubuntu systemd-memcached-wrapper[12006]: >27 END
Nov 04 19:15:18 ubuntu systemd-memcached-wrapper[12006]: <27 set name_1 1 3600 5
Nov 04 19:15:19 ubuntu systemd-memcached-wrapper[12006]: >27 STORED
Nov 04 19:15:26 ubuntu systemd-memcached-wrapper[12006]: <27 delete name_2
Nov 04 19:15:26 ubuntu systemd-memcached-wrapper[12006]: >27 DELETED
Nov 04 19:15:29 ubuntu systemd-memcached-wrapper[12006]: <27 flush_all
Nov 04 19:15:29 ubuntu systemd-memcached-wrapper[12006]: >27 OK
Nov 04 19:15:37 ubuntu systemd-memcached-wrapper[12006]: <27 quit
Nov 04 19:15:37 ubuntu systemd-memcached-wrapper[12006]: <27 connection closed.
Nov 04 19:15:41 ubuntu systemd-memcached-wrapper[12006]: Signal handled: Terminated.
Nov 04 19:15:41 ubuntu systemd[1]: Stopping memcached daemon...
Nov 04 19:15:41 ubuntu systemd[1]: memcached.service: Succeeded.
Nov 04 19:15:41 ubuntu systemd[1]: Stopped memcached daemon.

```

〈Figure 14〉 Memcached logs in “memcached_logs” file

5.2 메모리 포렌식 기법을 활용한 삭제·변조된 데이터 원본값 확인

본 절에서는 1000개의 데이터셋을 가지며, Log Level이 “vv”로 설정된 활성 상태의 Memcached를 대상으로 안티-포렌식 행위를 수행하였다. 이후, 로그와 메모리를 수집·분석하여 안티-포렌식 행위를 탐지하고 삭제·변조된 데이터의 원본값을 확인하는 실험을 진행하였다.

5.2.1 테스트 데이터 생성

key[1~1000], value[1~1000]의 값을 각각 가지는 1000개의 Key-Value 쌍을 생성하였다. [그림 15]와 같이 “stats” 명령어를 입력한 결과 현재 1000개의 데이터가 존재하는 것을 확인하였다.

```

kyleyoona@ubuntu: ~/test
STAT curr_items 1000

```

〈Figure 15〉 Result of “stats” command

5.2.2 데이터 삭제·변조

1000개의 데이터셋 중 임의로 3개의 데이터를 삭제하고, 1개의 데이터를 변조하였다 [그림 16]. 삭제·변조된 데이터는 [표 9]와 같다. 안티-포렌식 행위가 정상적으로 반영되었는지 확인하기 위해 “stats” 명령을 재 입력한 결과 3개의 데이터가 삭제되었으므로 현재 유효한 데이터를 나타내는 “STAT curr_items”의 값이 기존의 “1000”에서 “997”로 변경된 것으로 확인하였다 [그림 17].



```
kyleyon@ubuntu: ~/test
delete key100
DELETED
delete key200
DELETED
delete key300
DELETED
set key999 999 3600 5
dummy
STORED
```

〈Figure 16〉 Results of “delete” and “set” commands on Terminal

〈Table 9〉 Data with Anti-Forensics applied

Key	Value	Anti-Forensics Applied
key100	value100	데이터 삭제
key200	value200	데이터 삭제
key300	value300	데이터 삭제
key999	value999	데이터 변조 (value999 → dummy)

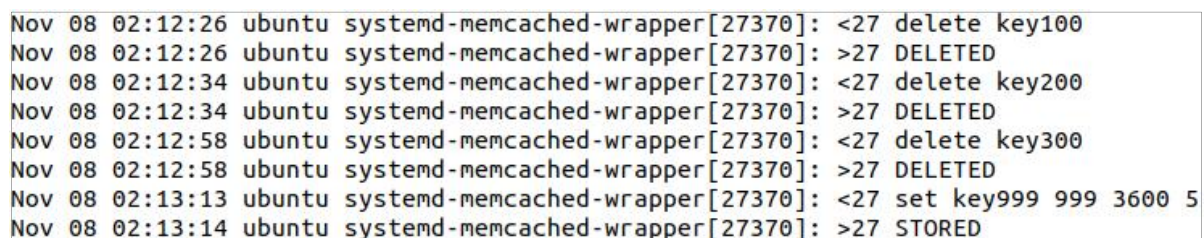


```
kyleyon@ubuntu: ~/test
STAT curr_items 997
```

〈Figure 17〉 Result of “stats” command on Terminal

5.2.3 로그파일을 통한 안티-포렌식 행위 탐지

[그림 16]과 같이 행해진 안티-포렌식 행위를 로그파일에서 확인하기 위해 “journalctl -u memcached.service” 명령을 통해 syslog 파일에서 Memcached 관련 로그를 선별하여 조회하였다. 그 결과, 로그에서 수행되었던 안티-포렌식 관련 명령어와 각 상응하는 Key 값을 확인할 수 있었다 [그림 18]. 이를 토대로, “key100”, “key200”, “key300”이 시간정보와 함께 삭제되었다는 것을 유추할 수 있으며, “key999”의 경우 시간정보와 함께 5 byte의 값을 가지는 값으로 수정(변조)되었다는 것을 확인할 수 있다. 이렇게 삭제·변조된 데이터의 원본값은 Memcached에서 삭제되었거나 원래의 값이 다른 값으로 변조되었으므로 확인할 수 없다. 따라서, 로그를 통해 4개의 데이터가 삭제·변조되었다는 정황을 토대로 메모리를 수집·분석하여 데이터 원본값을 확인하는 실험을 진행하였다.

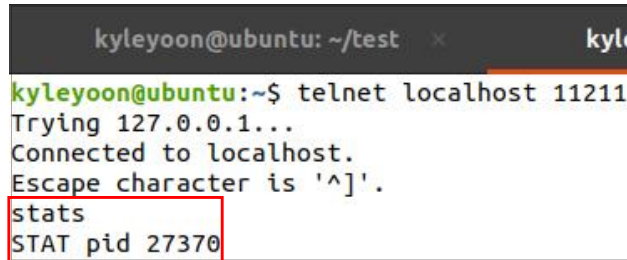


```
Nov 08 02:12:26 ubuntu systemd-memcached-wrapper[27370]: <27 delete key100
Nov 08 02:12:26 ubuntu systemd-memcached-wrapper[27370]: >27 DELETED
Nov 08 02:12:34 ubuntu systemd-memcached-wrapper[27370]: <27 delete key200
Nov 08 02:12:34 ubuntu systemd-memcached-wrapper[27370]: >27 DELETED
Nov 08 02:12:58 ubuntu systemd-memcached-wrapper[27370]: <27 delete key300
Nov 08 02:12:58 ubuntu systemd-memcached-wrapper[27370]: >27 DELETED
Nov 08 02:12:13 ubuntu systemd-memcached-wrapper[27370]: <27 set key999 999 3600 5
Nov 08 02:13:14 ubuntu systemd-memcached-wrapper[27370]: >27 STORED
```

〈Figure 18〉 Memcached logs

5.2.4 메모리 수집 분석

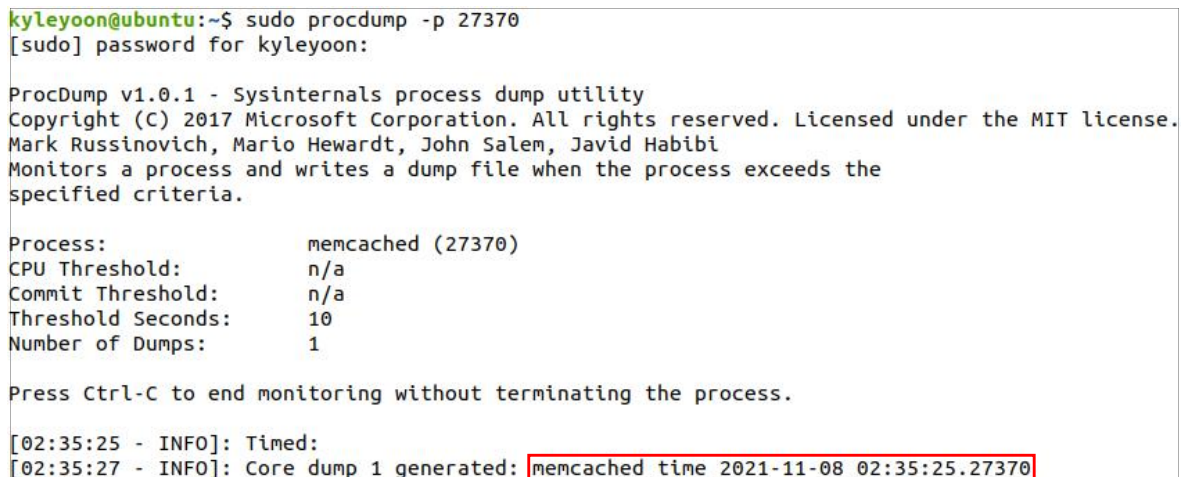
메모리를 수집하기 위해 Memcached의 “Process ID”를 조회하고자 “stats” 명령어를 입력하여 확인하였다 [그림 19].



```
kyleyon@ubuntu: ~/test x kyle
kyleyon@ubuntu:~$ telnet localhost 11211
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
stats
STAT pid 27370
```

〈Figure 19〉 Process ID of Memcached

이후, 메모리 덤프 도구인 “ProcDump”를 사용하여 Memcached 프로세스를 덤프하였다 [그림 20].



```
kyleyon@ubuntu:~$ sudo procdump -p 27370
[sudo] password for kyleyon:

ProcDump v1.0.1 - Sysinternals process dump utility
Copyright (C) 2017 Microsoft Corporation. All rights reserved. Licensed under the MIT license.
Mark Russinovich, Mario Hewardt, John Salem, Javid Habibi
Monitors a process and writes a dump file when the process exceeds the
specified criteria.

Process:                memcached (27370)
CPU Threshold:          n/a
Commit Threshold:       n/a
Threshold Seconds:      10
Number of Dumps:        1

Press Ctrl-C to end monitoring without terminating the process.

[02:35:25 - INFO]: Timed:
[02:35:27 - INFO]: Core dump 1 generated: memcached_time_2021-11-08_02:35:25.27370
```

〈Figure 20〉 Memory dump of Memcached process

획득한 메모리 파일은 HxD 도구를 사용하여 로그 파일에서 안티-포렌식 행위가 적용된 것으로 추정되는 Key 값(“key100”, “key200”, “key300”, “key999”)을 각각의 문자열로 검색하는 방식으로 분석하였다. 그 결과, 삭제된 데이터 3쌍 모두 각 Key에 해당되는 원본 값을 각기 다른 오프셋 주소에서 확인할 수 있었다. 데이터가 변조된 것으로 추정되는 “key999”의 경우 각기 다른 두 개의 오프셋 주소에서 변조 이전·이후로 추정되는 Value 값을 확인할 수 있었다. Key 값으로 “key999”를 가지는 데이터는 [그림 18]과 같이 로그 파일을 분석한 결과로 Value 값이 5 byte로 변경되었다는 것을 확인하였으므로, 5 byte의 값을 가지는 “dummy”로 변조되었다는 것을 확인할 수 있다 [표 10] [그림 21]. 만약 byte 값이 변조 이전·이후 동일한 경우, 변조 이후(현재)의 값은 활성 상태의 Memcached에서 명령어를 통해 확인할 수 있으므로 이를 교차 확인하여 변조되기 이전의 원본 값을 확인하는 방식을 취할 수 있다.

〈Table 10〉 Analysis result of memory dump file of Memcached process

실험 데이터	메모리 주소 오프셋	비고
〈key 100, value 100〉	15941D60	원본 데이터
〈key 200, value 200〉	1593F360	원본 데이터
〈key 300, value 300〉	1593A020	원본 데이터
〈key 999, value 999〉	1592A780	원본 데이터
〈key 999, dummy〉	1593C9C0	변조 데이터

memcached_time_2021-11-08_02^%35^%25.27370	
Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
15941D60	6B 65 79 31 30 30 00 76 61 6C 75 65 5F 31 30 30 key100.value_100
memcached_time_2021-11-08_02^%35^%25.27370	
Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
1593F360	6B 65 79 32 30 30 00 76 61 6C 75 65 5F 32 30 30 key200.value_200
memcached_time_2021-11-08_02^%35^%25.27370	
Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
1593A020	6B 65 79 33 30 30 00 76 61 6C 75 65 5F 33 30 30 key300.value_300
memcached_time_2021-11-08_02^%35^%25.27370	
Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
1592A780	6B 65 79 39 39 39 00 76 61 6C 75 65 5F 39 39 39 key999.value_999
memcached_time_2021-11-08_02^%35^%25.27370	
Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
1593C9C0	6B 65 79 39 39 39 00 E7 03 00 00 64 75 6D 6D 79 key999.ç...dummy

〈Figure 21〉 Analysis result of memory dump file of Memcached process on HxD

VI. 결론

본 논문에서는 Key-Value 모델의 인메모리 솔루션인 Memcached의 데이터 수집·분석 방법, 로그를 활용한 안티-포렌식 행위 탐지 방법, 메모리 포렌식 기법을 통한 삭제·변조된 데이터의 원본 값을 확인하는 방법을 연구하였다. 활성상태의 Memcached에 존재하는 데이터를 효과적으로 수집하는 방안으로, 새롭게 추가된 명령어를 활용하여 Memcached에 존재하는 모든 Key를 메타데이터 정보와 함께 조회가 가능하였다. 안티-포렌식 행위를 탐지하기 위한 방안으로 Log Level이 “vv”로 설정되어 있는 경우 syslog에 기록되는 Memcached 로그를 수집·분석하여 사용된 명령어와 안티-포렌식 명령어가 실행된 데이터의 Key 값을 추적하는 방법을 제안하였다. 마지막으로, Memcached 프로세스의 메모리를 수집하여 로그에서 확인한 안티-포렌식 적용된 데이터의 Key 값을 문자열로 조회하였을 때 일치하는 원본 Value 값을 확인할 수 있었다.

Memcached는 별도의 데이터 백업 기능이나 영속성을 보장하는 기능을 지원하지 않으며, 인메모리 저장소 특성상 데이터가 쉽게 손실되는 휘발성 성격을 가진다. 이에, 로그와 메모리 포렌식 기법을 활용하면 안티-포렌식 행위 탐지와 삭제·변조된 원본 데이터를 확인함에 있어 많은 기여를 할 수 있을 것이다.

참 고 문 헌 (References)

- [1] Houcine Matallah, Ghalem Belalem, Karim Bouamrane, "Evaluation of NoSQL Database: MongoDB, Cassandra, HBase, Redis, Couchbase, OrientDB", International Journal of Software Science and Computational Intelligence 12(4), Dec. 2020, pp. 71-91.
- [2] Korea Data Agency, "2020 Data Industry Survey Report", Available: https://www.kdata.or.kr/info/info_01_view.html?dbnum=297, 2021.11.08. confirmed.
- [3] Solid IT, "DB-Engines Ranking of Key-value stores", Available: <https://db-engines.com/en/ranking/key-value+store>, 2021.11.08. confirmed.
- [4] Werner K. Hauger and Martin S. Olivier, "Forensic attribution in NoSQL databses", SAIEE Africa Research Journal 109(2), June. 2018, pp. 119-132
- [5] Ming Xu, Xiaowei Xu, Jian Xu, Yizhi Ren, Haiping Zhang, and Ning Zheng, "A Forensic Analysis Method for Redis Database Based on RDB and AOF File", Journal of Computers 9(11), Nov. 2014, pp. 2538 - 2544
- [6] Choi Jaemun, Jeong Doowon, Yoon Jongseong, and Lee Sangjin, "Digital Forensics Investigation of Redis Database", KIPS Transactions on Computer and Communication Systems 5(5), May. 2016, pp. 117-126
- [7] Choi Jaemun, "Digital Forensic Investgiation of NoSQL Key-Value model Database", Master's thesis, Korea University, 2011.
- [8] "Memcached 1.4.31 Release Notes", Available: <https://github.com/memcached/memcached/wiki/ReleaseNotes1431>, 2021.11.08. confirmed.
- [9] Han Kyusik and Nah Yunmook, "In-Memory Database Data Access using Memory Forensics Techniques", The Journal of Korean Institute of Next Generation Computing 12(5), Oct. 2016, pp. 23-30
- [10] "About Memcached", Available: <https://memcached.org/about>, 2021.11.10. confirmed.
- [11] Hiroshi Tsunoda and Glenn Mansfield Keeni, "Managing Syslog", The 16th Asia-Pacific Network Operations and Management Symposium, (Taiwan,) 2014

저 자 소 개



윤 호 (Ho Yoon)

준회원

2015년 5월 : 영국 Royal Holloway, University of London 국제경영학 졸업

2021년 3월 ~ 현재: 고려대학교 일반대학원 정보보안학과 석사과정 재학

관심분야 : 디지털 포렌식, e-Discovery



김 준 호 (Junho Kim)

준회원

2020년 2월 : 고려대학교 정보보호대학원 공학석사

2020년 3월 ~ 현재: 고려대학교 정보보호대학원 박사과정

관심분야 : 디지털 포렌식, 침해사고대응, 인공지능



김 성 배 (Sungbae Kim)

준회원

2019년 8월 ~ 현재 : 대검찰청 디지털수사과

관심분야 : 디지털 포렌식



양 양 한 (Yanghan Yang)

준회원

2019년 2월 : 서울대학교 융합과학기술대학원 이학석사

2021년 8월 ~ 현재: 대검찰청 디지털수사과

관심분야 : 디지털 포렌식



이 상 진 (Sangjin Lee)

평생회원

1989년 10월 ~ 1999년 2월 : 한국전자통신연구원 선임 연구원

1999년 3월 ~ 2001년 8월 : 고려대학교 자연과학대학 조교수

2001년 9월 ~ 현재 : 고려대학교 정보보호대학원 교수

2017년 3월 ~ 현재 : 고려대학교 정보보호대학원 원장

관심분야 : 디지털 포렌식, 정보윤리 이론, 대칭키 암호