ISSN : 1976－5304 / (Online)2713－931X
디지털포렌식연구
제19권 제4호(2025.08) 59－80쪽
(사단법인)한국디지털포렌식학회

DOI : 10.22798 / kdfs.2025.19.4.59
Journal of Digital Forensics
Vol.19 No.4(Aug 2025) pp.59－80
Korean Digital Forensics Society

# 비휘발성 가상메모리 파일 분석을 통한 카카오톡 PC 메신저 데이터 복구 방안

박 동 현*, 이 상 은**, 김 기 범***
성균관대학교 디지털포렌식학과(대학원생)*, 성균관대학교 과학수사학과(대학원생)**, (교수)***

# Recovery of KakaoTalk PC Messenger Data from Non－Volatile Windows Virtual Memory Files

Dong－Hyun Park*, Sang－eun Lee**, Gi－Bum Kim**
Dept. of Digital Forensics, Sungkyunkwan University(Graduate Student)*,
Dept. of Forensic Science, Sungkyunkwan University(Graduate Student)**, (Professor)***

■ 요약 ■

메신저는 다수의 단서를 포함하고 있어 수사상 핵심적인 증거로서 가치가 크다. 메신저 데이터 확보와 삭제된 데이터 복구는 주로 저장매체의 DB 분석에 의존하여 메시지 삭제나 애플리케이션 제거 시 증거 확보가 어렵다. 본 연구는 카카오톡 PC 버전의 물리 메모리(RAM)에서 메시지 시그니처를 식별하고, Windows 가상메모리 파일에서 삭제된 메시지를 복구하는 기법을 제안한다. 메시지 구조와 시그니처를 식별한 뒤, 가상메모리를 대상으로 삭제된 메시지 복구 실험을 수행하였다. 제안 기법을 적용한 실험 결과, 삭제된 메시지에 대해 hiberfil.sys는 평균 62.3%, pagefile.sys는 30%의 복구율을 나타냈다. 가상메모리를 활용한 메시지 복구 방법을 제시하였으며, 범죄수사 과정에서 삭제 메시지 복구의 새로운 접근 방안을 제시한다.

● 주제어 : 메모리 포렌식, pagefile.sys, hiberfil.sys, PC 메신저 애플리케이션, 안티포렌식

■ ABSTRACT ■

Messenger applications contain numerous investigative clues and serve as critical evidence in criminal investigations. However, the acquisition and recovery of deleted messenger data have mainly relied on database analysis of storage media, making it difficult to secure evidence once messages are deleted or the application is removed. This study proposes a technique to identify message signatures in the physical memory (RAM) of the KakaoTalk PC version and to recover deleted messages from Windows virtual memory files. After analyzing the message structure and identifying signatures, recovery experiments were conducted on virtual memory. The experimental results showed that the proposed method achieved average recovery rates of 62.3% for hiberfil.sys and 30% for pagefile.sys. This study presents a message recovery approach using virtual memory and provides a new direction for recovering deleted messages in criminal investigations.

● Key Words : Memory forensics, pagefile.sys, hiberfil.sys, PC messenger, Anti－Forensics

■ 투고일 : 2025.07.25. 심사게시일 : 2025.07.25. 게재확정일 : 2025.08.31.
■ 제1저자(First Author) : Donghyun Park (Email : park44177@gmail.com)
■ 교신저자(Corresponding Author) : GiBum Kim (Email : freekgb02@gmail.com)

# Ⅰ. 서 론

사이버범죄 수사에서 메신저 대화 기록은 핵심적인 증거이나, 삭제가 일반화됨에 따라 확보가 어려워지고 있다. 현재 사용되는 저장매체 내 DB 파일 분석은 SQLite 데이터베이스와 WAL(Write－Ahead Logging) 파일을 분석하여 메시지를 복구하는 방식으로, DB 파일이 삭제되거나 SSD의 TRIM 명령으로 데이터가 초기화되면 복구가 어렵다. 특히 메신저 애플리케이션 제거나, 애플리케이션에서 메시지 삭제 및 초기화는 저장매체 기반 분석을 무력화하고 있다.

운영체제의 가상메모리 영역 분석은 저장매체 기반 분석의 한계를 극복할 수 있다. Windows 운영체제는 물리 메모리(RAM) 공간이 부족하면 일부 메모리 페이지를 pagefile.sys에 저장하고, 절전 모드 전환 시 전체 메모리 페이지를 hiberfil.sys에 기록한다. 메신저 애플리케이션은 실행 중 메시지와 메타데이터를 물리 메모리에 저장하고, 일부 페이지는 삭제 이후에도 가상메모리 파일에 저장될 수 있다. 그러나 메신저의 메모리 저장 구조를 체계적으로 분석하고 가상메모리에서 삭제 메시지를 복구한 연구는 다소 미흡하다.

본 연구는 Windows의 pagefile.sys와 hiberfil.sys에서 삭제된 카카오톡 메시지를 복구하는 기법을 제안한다. 카카오톡 메시지가 메모리에 기록되는 구조와 시그니처를 규명하고, 가상메모리에서 복구 가능성을 실험하여 정량화하였다. 또한 가상메모리 기반 메신저 대화내역 탐지 및 복구 스크립트를 개발하여 수사 현장에서 활용할 수 있게 하였다.

본 논문은 제2장에서 가상메모리 기반 데이터 복구 연구와 카카오톡 삭제 방식별 복구 가능성에 대한 선행연구를 검토하고, 제3장에서 카카오톡 실행 시 메모리에 저장되는 메시지 및 친구 정보 구조를 분석한다. 제4장에서는 가상메모리 영역에서 삭제 메시지를 식별·복구하는 실험을 수행하고, 제5장에서 메시지 삭제 유형과 가상메모리 종류별 복구 가능성을 비교·분석한다.

# Ⅱ. 배경지식 및 선행연구

## 2.1. 메신저 데이터 삭제 및 복구

메신저에 대한 삭제 데이터 연구는 다양하다. Lee 등(2014)은 WhatsApp, 카카오톡 등에서 SQLite 기반 메시지 삭제 방식을 분석하고, 논리 삭제는 복구가 가능하나 Secure Delete가 적용되면 복구가 불가능함을 확인하였다. WAL과 Journal 파일을 통해 일부 데이터의 복구 가능성을 확인하였다[1]. Heath 등(2023)은 다양한 메신저 플랫폼을 대상으로 메시지 삭제 후에도 메시지 본문, 첨부파일, 메타데이터, 송수신 기록 등이 잔류하는 것을 발견하였다. 데이터는 주로 기기의 파일 시스템과 애플리케이션 저장 영역 등 비휘발성 저장소에 남았고, 일정 시간이 지나거나 안티포렌식으로 삭제하면 복구가 불가능하다고 주장하였다[2].

정병찬 등(2018)은 모바일 환경의 카카오톡 메시지 데이터베이스인 KakaoTalk.db의 SQLite 저널 파일

으로부터 채팅방 나가기 기능으로 삭제된 메시지를 복구하였다. 개별적으로 삭제된 단일 메시지는 복구되지 않았다[3]. 김민동 등(2021)은 사용자의 USIM을 다른 기기에 삽입하여 동기화된 크리덴셜으로 카카오톡의 최근 3일간의 채팅 내역을 획득하였다. '이 기기에서 삭제' 옵션으로 삭제한 카카오톡 메시지를 복구하였으나 '모든 대화 상대에게서 삭제'옵션으로 삭제한 메시지는 복구하지 못했다[4]. 조민욱 등(2023)은 Windows 환경에서 카카오톡 메시지의 암호화 구조를 분석하였지만, '이 기기에서 삭제' 기능이 수행된 메시지의 경우 WAL 파일이 체크포인트 처리되어 데이터가 제거되므로 복구가 불가능하다는 것을 확인하였다[5].

## 2.2. 가상메모리 기반 데이터 복구

Shin 등(2021)은 SNS, 메신저, 금융 서비스 이용 시 개인정보가 물리 메모리에 평문으로 저장되어 노출될 수 있음을 확인하였다[6]. 윤병철 등(2020)은 Windows 환경에서 말랑말랑 톡카페 메신저의 데이터베이스 암호화 구조를 분석하고, 메모리 기반 삭제 메시지의 복구를 시도하였으나, 복구 범위는 메시지 본문에 한정되었다. 메시지 ID, 전송시간, 발신자 등 구조화된 대화 정보는 복구하지 못했다[7]. Rawashdeh 등(2024)은 pagefile.sys를 수집·분석하여 비휘발성 저장소에 남은 메모리 기반 증거 확보 절차를 제시하였고[8], Ghafarian 등(2020)은 Windows 10의 hiberfil.sys를 분석하여 메신저 메시지, 연락처, 사용자 ID 복구 가능성을 입증하였다[9]. Hwang 등(2015)은 물리 메모리에서 네이트온 및 카카오톡 사용자 정보와 파일 전송 내역을 추출하는 기법을 제시하였으나, 대화 내용 자체의 추출이나 비휘발성 가상메모리 분석은 수행하지 않았다[10].

## 2.3. 스와핑 발생 조건에 관한 선행연구

Sylve 등(2016)은 Windows 환경에서 인위적인 메모리 압박과 I/O 부하를 병행하여 pagefile.sys 갱신 패턴과 교체율을 실험적으로 분석하였다[11]. Ahmad 등(2022)은 고해상도 영상 스트리밍이 Android 기반 애플리케이션의 메모리·스토리지에 대규모 아티팩트를 생성한다는 점을 포렌식 분석으로 확인하였다[12]. Fernández-Fuentes 등(2022)은 다중 웹 브라우저 세션과 캐시 동작이 메모리에 잔존 데이터를 남겨 스와핑 가능성을 높인다고 주장하였다[13]. 일반적인 스와핑은 물리 메모리 부족, 백그라운드 전환, 시스템 유휴 상태 등의 상황에서 발생한다. Russinovich 등(2017)은 Windows 10 환경에서 메모리 사용률이 70~80%를 지속적으로 초과할 경우 비활성 페이지가 pagefile.sys로 이동하는 현상을 설명하였다[15]. 선행 연구들은 고해상도 스트리밍·대용량 압축·다중 브라우저 실행 등과 같은 자원 집중형 작업이 실제 환경에서도 수 시간 내에 스와핑을 유발할 수 있음을 확인하였다.

## 2.4. 카카오톡 메신저의 데이터 삭제 방식 및 복구

카카오톡은 로컬 SQLite 데이터베이스를 이용해 메시지를 저장·관리하며, 삭제 유형에 따라 서로 다른 처리 구조가 적용된다. 본 연구에서는 기술적 근거를 보완하기 위해 국내 포렌식 기업의 개발자를 대

상으로 2025년 7월 21일 인터뷰를 실시하였다. 인터뷰 대상자는 포렌식 경력 10년과 개발자 경력 8년을 보유한 전문가로, 카카오톡 메신저의 데이터 처리 방식과 한계점을 확인하기 위해 ① 삭제 방식, ② DB 복구 방식, ③ 삭제 유형별 복구 가능성에 관한 질문을 진행하여 카카오톡 메신저 삭제 방식과 복구 가능성을 정리하였다.

첫째, '모든 대화상대에서 삭제' 기능은 논리적 삭제(Logical Deletion) 방식으로, 기존 메시지 레코드의 type 값이 '16385'로 변경되고 삭제 대상 ID를 참조하는 신규 레코드가 생성된다. 원본 메시지 본문은 데이터베이스에 잔존하며, 이 특성으로 인해 메인 DB를 통해 복구가 가능하다.

둘째, '이 기기에서만 삭제' 기능은 보안 삭제(Secure Delete) 방식으로, 메시지 본문이 0x00으로 덮어쓰여 물리적으로 제거된다. 삭제 직후 WAL 파일에 데이터가 일시 저장되지만, 체크포인트 병합이 완료되면 WAL 파일에서도 해당 데이터가 제거된다.

셋째, '채팅방 나가기'는 채팅방 UUID가 제거되어 기존 메시지와의 연결이 단절되며, 관련 레코드가 삭제된다. 삭제 직후 WAL 파일에 일부 데이터가 잔존할 수 있으나, 병합 이후에는 복구가 제한된다.

넷째, '애플리케이션 삭제'는 데이터베이스, WAL, 캐시 등을 일괄 제거한다. 확보 시점이 지연되면 TRIM 명령이 작동하여 삭제 영역이 초기화되므로 물리적 복구가 불가능하다.

이러한 삭제 방식과 복구 가능성의 차이는 <표 1>에 정리하였으며, 본 연구에서는 복구 난이도가 높은 '이 기기에서만 삭제', '채팅방 나가기', '애플리케이션 삭제' 방식을 중심으로 실험을 수행하였다.

〈표 1〉 카카오톡 삭제 방식별 처리구조와 복구 가능성

| 연번 | 삭제 방식 | 삭제 대상 | 처리 방식 | WAL 잔존 가능성 | 복구 가능성 | 특징 요약 |
|---|---|---|---|---|---|---|
| 1 | 모든 대화상대에서 삭제 | 발신자 및 수신자 기기 | 삭제 레코드 변경 (논리적 삭제, 원본 유지) | 없음 | 높음 | DB에 원본 데이터 잔존, 비교적 용이한 복구 가능 |
| 2 | 이 기기에서만 삭제 | 발신자 기기 | 본문 0x00으로 덮어쓰기 (Secure Delete) | 있음 (일시적) | 낮음 | WAL에 일시적 잔존 가능, 체크포인트 병합 후 삭제 |
| 3 | 채팅방 나가기 | 발신자 기기 | 채팅방 UUID 변경 및 로컬 DB 일부 삭제 | 있음 (일시적) | 낮음 | UUID 변경으로 논리적 연결 단절, WAL 병합 후 복구 어려움 |
| 4 | 애플리케이션 삭제 | 발신자 기기 | SQLite DB, WAL, 캐시 등 전체 삭제 | 없음 | 없음 | DB파일 자체 삭제, Trim으로 복구 불가 |

## Ⅲ. 메모리에서 카카오톡 메신저의 대화내용 구조 분석

### 3.1. 실험설계

본 실험은 카카오톡 메신저 실행 중 전송된 메시지와 사용자 정보가 물리 메모리(RAM)에 저장되는 구

조를 분석하는 것을 목적으로 한다. 데이터의 구조적 패턴과 시그니처를 식별하여, 가상메모리 기반 삭제 메시지 복구 기법 설계에 활용하기 위한 기초 실험으로 수행되었다.

메모리 구조 분석은 Windows 10 Pro(64bit), 8GB RAM 환경의 데스크톱에서 진행하였다. 분석 대상 애플리케이션은 국내에서 점유율이 높은 카카오톡 PC 버전(v4.3.5.4323)으로 선정하였다. 실험용 메시지를 전송한 후 물리 메모리에서 분석한 데이터 검증을 위해 Android 15(Samsung Galaxy S25) 환경에서 동일 버전의 모바일 애플리케이션(v11.4.2) 데이터를 확보하여 수행하였다. 물리 메모리 덤프 수집에는 Procdump와 FTK Imager를 사용하였다. 데이터의 바이너리 구조 분석과 시그니처 추출은 WinHex로 진행하였고, 탐지 스크립트는 Python 기반으로 구현하였다. 제안 기법의 결과는 GMD SOFT에서 개발한 상용 도구인 MD-RED의 분석 결과와 비교하였다. 실험에 사용된 주요 시스템과 도구는 <표 2>에 제시하였다.

〈표 2〉 실험 대상 시스템 및 사용 도구 목록

| Type | Experimental Environment and Tools | |
|---|---|---|
| Device/OS | Desktop: Windows 10 Pro (19045.3537), RAM 8GB<br>Notebook: windows 11 Pro(22631.5335), RAM 16GB<br>Mobile: Samsung Galaxy S25 (SM-S931N) / Android 15 | |
| Application Version | KakaoTalk | PC: v.4.3.5.4323 (Released 02/12/2025)<br>Mobile: v11.4.2 (Released 02/10/2025) |
| Analysis Tool | Acquisition | FTK Imager v3.1.1.8, Procdump v11.0 |
| | Scripting | Python v3.10.1 |
| | Hex Analysis | WinHex v19.6 |
| | Mobile Acquisition | MD-NEXT v2.1.16.2533 |
| | Mobile Analysis | MD-RED v3.12.1.3235 |

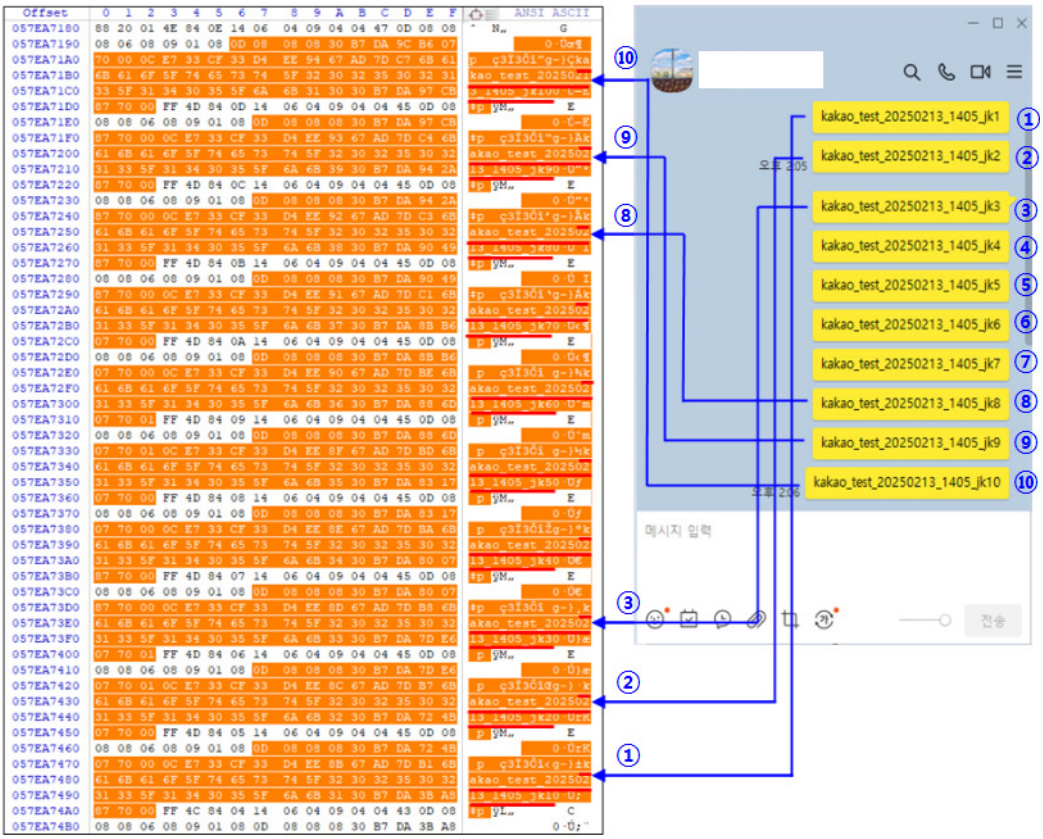
카카오톡 PC 메신저 실행 상태에서 메시지 10건을 순차 전송한 뒤, 즉시 프로세스 메모리 덤프(.dmp)와 전체 물리 메모리 덤프를 수집하였다. 검증 메시지는 “kakao_test_20250213_1405_jk1”부터 “_jk10”까지 시간·내용·순번만 변경하여 전송하였고, WinHex와 Python 기반 정규표현식 분석 도구로 HEX 데이터를 분석하여 메시지 본문, 전송 시각, 송신자 ID, 메시지 ID의 저장 형식과 반복 패턴을 확인하고, 시그니처의 구조를 식별하였다.

## 3.2. 메시지 저장 구조 분석

### 3.2.1. 메시지 구조

카카오톡 메시지는 물리 메모리에서 일정한 구조를 가지며, 하위 주소에서 상위 주소 방향으로 순차 저장된다. 각 블록에는 메시지 본문과 함께 메시지 ID, 전송 시각, 송신자 ID가 포함된다. 전송한 메시지(kakao_test_20250213_...)는 카카오톡 화면의 실제 전송 내용과 일치하며, 메시지가 물리 메모리에 일관된 형태로 저장됨을 확인할 수 있다<그림 1>.

〈그림 1〉 물리 메모리 덤프 내 카카오톡 메시지 구조와 실제 대화 내용 비교




각 메시지의 Hex 데이터를 시각화하여 병렬 배열로 비교 분석하였다<그림 2>. 각 항목은 오프셋(offset) 기준으로 순차적으로 증가하거나 동일한 데이터가 반복적으로 나타났다.

〈그림 2〉 물리 메모리에 저장된 카카오톡 메시지 16진수(Hex) 구조 해석

| | 메시지 구분자 | 메시지 ID | 전송자 ID | 메시지 순번 | 전송 시간 | 메시지 내용 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Offset | 00 | 04 | 08 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 | 44 | 48 | 52 |
| 000057EA7196 | 0D080808 | 30B7DA9C | B6077000 | 0CE733CF | 33D4EE94 | 67AD7DC7 | 6B616B61 | 6F5F7465 | 73745F32 | 30323530 | 3231335F | 31343035 | 5F6A6B31 | 3030B7DA |
| 000057EA71E7 | 0D080808 | 30B7DA97 | CB877000 | 0CE733CF | 33D4EE93 | 67AD7DC4 | 6B616B61 | 6F5F7465 | 73745F32 | 30323530 | 3231335F | 31343035 | 5F6A6B39 | 30B7DA94 |
| 000057EA7237 | 0D080808 | 30B7DA94 | 2A877000 | 0CE733CF | 33D4EE92 | 67AD7DC3 | 6B616B61 | 6F5F7465 | 73745F32 | 30323530 | 3231335F | 31343035 | 5F6A6B38 | 30B7DA90 |
| 000057EA7287 | 0D080808 | 30B7DA90 | 49877000 | 0CE733CF | 33D4EE91 | 67AD7DC1 | 6B616B61 | 6F5F7465 | 73745F32 | 30323530 | 3231335F | 31343035 | 5F6A6B37 | 30B7DA8B |
| 000057EA72D7 | 0D080808 | 30B7DA8B | B6077000 | 0CE733CF | 33D4EE90 | 67AD7DBE | 6B616B61 | 6F5F7465 | 73745F32 | 30323530 | 3231335F | 31343035 | 5F6A6B36 | 30B7DA88 |
| 000057EA7327 | 0D080808 | 30B7DA88 | 6D077001 | 0CE733CF | 33D4EE8F | 67AD7DBD | 6B616B61 | 6F5F7465 | 73745F32 | 30323530 | 3231335F | 31343035 | 5F6A6B35 | 30B7DA83 |
| 000057EA7377 | 0D080808 | 30B7DA83 | 17077000 | 0CE733CF | 33D4EE8E | 67AD7DBA | 6B616B61 | 6F5F7465 | 73745F32 | 30323530 | 3231335F | 31343035 | 5F6A6B34 | 30B7DA80 |
| 000057EA73C7 | 0D080808 | 30B7DA80 | 07877000 | 0CE733CF | 33D4EE8D | 67AD7DB8 | 6B616B61 | 6F5F7465 | 73745F32 | 30323530 | 3231335F | 31343035 | 5F6A6B33 | 30B7DA7D |
| 000057EA7417 | 0D080808 | 30B7DA7D | E6077001 | 0CE733CF | 33D4EE8C | 67AD7DB7 | 6B616B61 | 6F5F7465 | 73745F32 | 30323530 | 3231335F | 31343035 | 5F6A6B32 | 30B7DA72 |
| 000057EA7467 | 0D080808 | 30B7DA72 | 4B077000 | 0CE733CF | 33D4EE8B | 67AD7DB1 | 6B616B61 | 6F5F7465 | 73745F32 | 30323530 | 3231335F | 31343035 | 5F6A6B31 | 30B7DA3B |

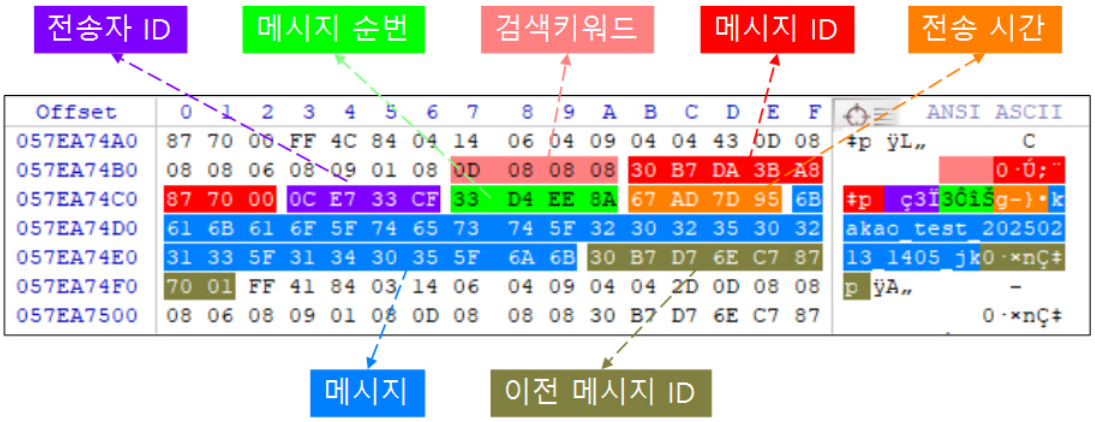
반복적으로 나타나는 패턴은 전송자 식별자(ID)로 확인되었고, 점진적으로 변화하는 값은 메시지의 전송 시각과 순번으로 확인하였다. 예를 들어, Offset 0x57EA7470에서 확인된 0C E7 33 CF는 전송자 ID로, 10진수로 변환하면 216478671이며, MD-RED 프로그램 분석 결과에서 확인된 계정 식별자 박○현(216478671)과 일치하였다<그림 3>.

〈그림 3〉 MD-RED 프로그램 분석 결과(연속된 카카오톡 메시지)

| 날짜 | 본문 | 메시지 ID | 채팅방 | 계정 |
|---|---|---|---|---|
| 생성 일시 : 2025-02-13 14:05:53 | 내용 : kakao_test_20250213_1405_jk1 | 3510514618978496512 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:05:59 | 내용 : kakao_test_20250213_1405_jk2 | 3510514668823605249 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:00 | 내용 : kakao_test_20250213_1405_jk3 | 3510514677975576576 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:02 | 내용 : kakao_test_20250213_1405_jk4 | 3510514691120525312 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:05 | 내용 : kakao_test_20250213_1405_jk5 | 3510514714038202369 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:06 | 내용 : kakao_test_20250213_1405_jk6 | 3510514728147841024 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:09 | 내용 : kakao_test_20250213_1405_jk7 | 3510514747802349568 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:11 | 내용 : kakao_test_20250213_1405_jk8 | 3510514764462125056 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:12 | 내용 : kakao_test_20250213_1405_jk9 | 3510514780048158720 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:15 | 내용 : kakao_test_20250213_1405_jk10 | 3510514801162285056 | 144164716725612 | 박동현 (216478671) |

카카오톡 메시지는 16진수(Hex) 값으로 인코딩된 상태로 물리 메모리에 저장된다. 전송자 ID, 메시지 순번, 메시지 ID, 전송 시간 등은 특정 바이트 배열을 통해 구조화된 형태로 기록되며, 메시지 내용은 ASCII 범위 내에서 평문으로 기록되었다<그림 4>.

〈그림 4〉 물리 메모리에 저장된 카카오톡 메시지 구조




각 위치별 Hex 값을 분석한 결과, offset에는 0D080808 값이 존재하며, 메시지 구분자로 사용되는 시그니처로 0D는 ASCII 코드상 Carriage Return(CR) 제어 문자로, 줄 시작 위치로 커서를 이동시키는 기능을 갖는다. 구조를 기반으로 메시지 구분 탐지를 위한 정규표현식을 pattern = re.compile(rb'\x0D\x08\x08\x08[\x30\x32\x06]')으로 작성하였다<표 3>.

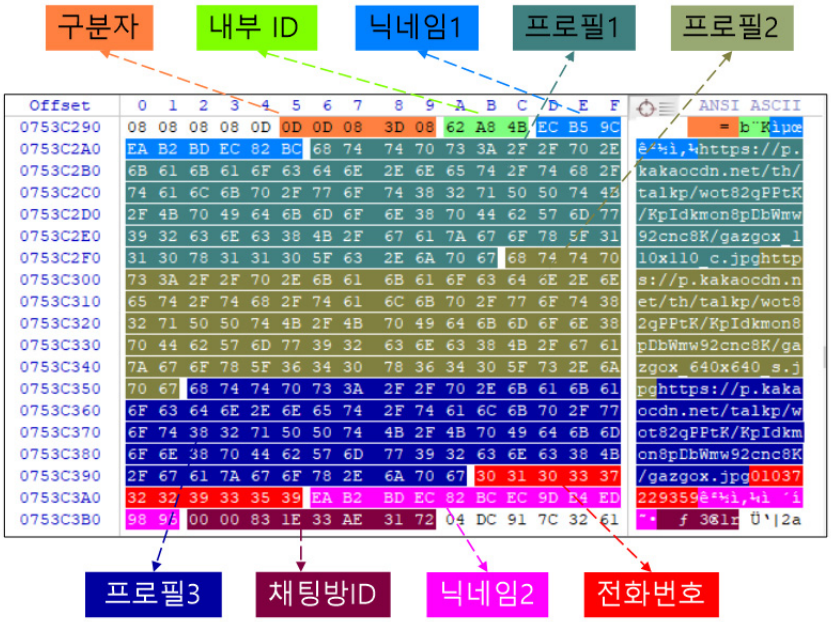
〈표 3〉 물리 메모리 내 카카오톡 대화 내용 Hex 해석

| 필드명 | 필드타입 | 길이 | 의미 | 위치 | 값 | 해석 |
|---|---|---|---|---|---|---|
| signature | uint8_t | 4 | 검색키워드 | 0x57EA74B7 | 0D 08 08 08 | 데이터 시그니처 (Signature) |
| message_id | uint8_t | 8 | 메시지 ID | 0x57EA74BB | 30 B7 DA 3B A8 87 70 00 | 3510514384323964928 (UUID 방식, 10진수 변환) |
| sender_id | uint8_t | 3~4 | 전송자 ID | 0x57EA74C6 | 0C E7 33 CF | 216478671 (10진수 변환) |
| msg_seq | uint8_t | 4 | 메시지 순번 | 0x57EA74C7 | 33 D4 EE 8A | 869,592,714 (10진수 변환) |
| timestamp | uint8_t | 4 | 전송 시간 (Big-Endian) | 0x57EA74CB | 67 AD 7D 95 | 2025-02-13 14:05:25 (Big-Endian, UTC+9) |
| msg_body | uint8_t | 가변 | 메시지 본문 | 0x57EA74CF | 6B616B616F5F746573745F323032353032313 35F313430355F6A6B | kakao_test_20250213_1405_jk (UTF-8 변환) |
| prev_msg_id | uint8_t | 8 | 이전 메시지 ID | 0x57EA74EA | 30 B7 D7 6E C7 87 70 01 | 3510511305352507393 (UUID 방식, 10진수 변환) |

### 3.2.2. 친구 정보 구조

카카오톡 메신저 친구 정보는 사용자 식별, 닉네임, 전화번호, 채팅방 ID 등의 정보를 포함한다. 데이터 구조는 메시지 구분을 위한 시그니처 값(0D0D083D08)으로 시작하고, 내부 ID(62A84B)는 UUID 방식으로 저장되어 사용자 또는 메시지를 식별하는 역할을 한다<그림 5>.

〈그림 5〉 프로세스 덤프 내 친구 정보 구조




닉네임1(EAB2BDEC82...)은 UTF-8 디코딩을 통해 '최*삼'으로 변환되고, 프로필 이미지 URL(68747470733A...)은 UTF-8 문자열로 변환되며, 프로필 사진은 썸네일(110x110px), 고해상도(640x640px), 원본 세 가지 해상

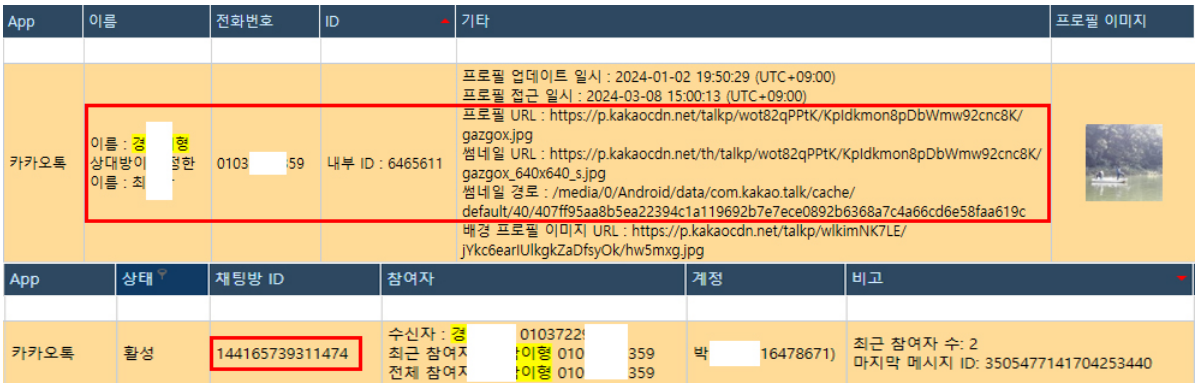
도로 저장된다. 닉네임2(EAB2BDEC82BCEC9DB4ED9895)는 UTF-8 디코딩 후 '경*이형'으로 복구되며, 사용자가 설정한 상대방의 별칭이다. 전화번호(3031303337.....)는 ASCII 변환을 통해 '0103****59'로 해석되고, 카카오톡 메신저에서 저장된 정보와 일치한다. 채팅방 ID(0000831E33AE3172)는 10진수로 변환하면 '144165739311474'로 채팅방 식별 값이다<표 4>.

〈표 4〉 프로세스 덤프 내 친구 정보 Hex 해석

| 위치 | 값 | 해석 | 의미 |
|---|---|---|---|
| 0x753C295 | 0D0D083D08 | 데이터 시그니처 (Signature) | 검색키워드 |
| 0x753C29A | 62A84B | 6465611 (UUID 방식) | 내부 ID |
| 0x753C29D | ECB59CEAB2BDEC82BC | 최*삼 (UTF-8 디코딩 변환) | 닉네임1 (상대방이 설정한 이름) |
| 0x753C2A6 | 68747470733A..... | KakaoTalk 프로필 이미지 URL (110x110px) | 프로필 사진 (썸네일) |
| 0x753C2FC | 68747470733A.... | KakaoTalk 프로필 이미지 URL (640x640px) | 프로필 사진 (고해상도) |
| 0x753C352 | 68747470733A.... | KakaoTalk 프로필 이미지 원본 URL | 프로필 사진 (원본) |
| 0x7544348 | EAB2BDEC82BCEC9DB4..... | 경*이형 (UTF-8 디코딩 변환) | 닉네임2 (내가 설정한 이름) |
| 0x753C39B | 3031303337323239333539 | 01037***59 (ASCII 변환) | 전화번호 |
| 0x754435B | 0000831E33AE3172 | 144165739311474 (10진수 변환) | 채팅방 ID |

프로세스 메모리 덤프에서 확인된 카카오톡 친구 정보의 구조는 모바일에서 수집한 데이터를 MD-RED 도구로 분석한 결과와 일치하였고, 내부 식별자(ID) 6465611에 대한 해석도 정확하였다. 상대방이 설정한 이름(닉네임1)은 최*삼이고, 사용자가 설정한 별칭(닉네임2)은 경*이형이다. 전화번호는 ASCII 변환을 통해 01037**359으로 채팅방 ID의 10진수 변환 결과도 144165739311474로 MD-RED 분석결과와 동일하게 나타났다. 물리 메모리 덤프에서 추출한 카카오톡 사용자 정보의 분석 결과가 MD-RED를 이용한 데이터와 일치하였다<그림 6>.

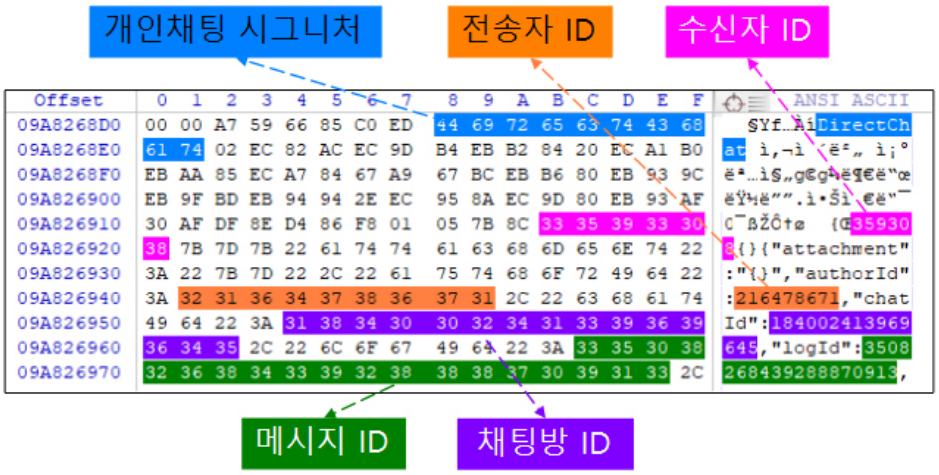
〈그림 6〉 MD-RED 분석 결과



### 3.2.3. 채팅방 정보 구조

채팅방 정보는 물리 메모리에 구조화된 형태로 저장되고, 'DirectChat' 시그니처를 기준으로 연속 기록된다. 데이터는 전송자 ID, 수신자 ID, 메시지 ID, 채팅방 ID 순으로 저장되고, 각 값은 Hex를 ASCII로 변환해 확인할 수 있다. 예를 들어 전송자 ID 3231363437383637 31은 ASCII 디코딩 시 216478671로, 수신자 ID 3335393038은 359308로 변환된다. 고정된 순서와 규칙성 덕분에 시그니처 기반 탐색을 통해 전송자·수신자, 채팅방 식별자, 메시지 ID 등 주요 데이터를 추출할 수 있다<그림 7>.

〈그림 7〉 물리 메모리 덤프 내 카카오톡 채팅방 정보 구조




Hex 데이터를 기반으로, 물리 메모리 내 채팅방 관련 정보 항목들을 정리하여 각 필드의 시작 주소와 저장된 값, 디코딩 결과, 의미를 포함하여 표를 작성하였다<표 5>.

〈표 5〉 물리 메모리 내 카카오톡 채팅방 정보 Hex 해석

| 위치 | 값 | 해석 | 의미 |
|---|---|---|---|
| 0x9A8268D8 | 4469726563<br>7443686174 | DirectChat<br>(Signature) | 검색키워드 |
| 0x9A82691B | 333539333038 | 359308<br>UTF-8 Decoding | 수신자 ID |
| 0x9A826941 | 3231363437383637 31 | 216478671<br>UTF-8 Decoding | 전송자 ID |
| 0x9A826954 | 31383430303234<br>3133393639363435 | 184002413969645 | 채팅방 ID |
| 0x9A82696C | 33353038323633834333932<br>3838383730393133 | 3508268439288870913 | 메시지ID |

## 3.3 스크립트 개발 및 검증

### 3.3.1. 메시지 검색 결과

16.9GB 크기의 물리 메모리 덤프를 대상으로 Hex 패턴 기반 탐색과 정규표현식을 적용한 카카오톡 메시지 수집용 Python 스크립트를 실행한 결과, 총 3,112건의 메시지 패턴이 탐지되고 분석에는 약 2분이 소요되었다. 먼저 0D080808 시그니처를 검색하여 메시지 블록의 시작 오프셋을 식별한 후, 해당 위치에서 메시지 ID(8바이트), 송신자 ID(3~4바이트), 전송 시각(4바이트), 메시지 본문을 순차적으로 추출하였다. 메시지 본문은 null 문자(0x00)나 특정 기호({,}) 또는 다음 메시지 시작 시그니처(0D080808)와 같은 종료 마커를 기준으로 분리하였다. 메시지 ID의 연속성을 비교해 중복 항목을 제거하고, 본문이 비어 있거나 오류가 포함된 데이터는 제외하는 전처리를 수행하였다.

탐색 과정은 물리 메모리 덤프를 100MB 단위로 분할해 순차 처리함으로써 대용량 파일 전체를 한 번에 메모리에 적재할 때 발생하는 I/O 지연과 메모리 부족을 방지하였다. 각 청크는 Hex 패턴 검색과 메시지 파싱 후 즉시 해제해 불필요한 메모리 점유를 줄였으며, 시그니처 절단 방지를 위해 청크 간 1KB 오버랩 영역을 두어 누락 없는 탐색을 구현하였다<그림 8>.

〈그림 8〉 카카오톡 메시지 수집 알고리즘 순서도(청크=100MB, 오버랩=1KB)


이 방식은 중복 검출을 방지하고 불필요한 스캔 범위를 줄여 분석 속도를 향상시켰다. 최종적으로 추출된 메시지와 메타데이터는 SQLite 데이터베이스(DB)에 저장하였다<그림 9>.

〈그림 9〉 자동수집 스크립트 실행결과 DB파일 내역

| offset_hex | message_id_hex | message_id_dec | sender_id_hex | sender_id_dec | timestamp_hex | timestamp | message_content |
|---|---|---|---|---|---|---|---|
| 필터 | 필터 | 필터 | 필터 | 필터 | 필터 | 필터 | jk |
| 0x57EA74B7 | 30B7DA3BA8877000 | 3510514384323964928 | 0CE733CF | 216478671 | 67AD7D95 | 2025-02-13 14:05:25 | kakao_test_20250213_1405_jk |
| 0x57EA7467 | 30B7DA724B077000 | 3510514618978496512 | 0CE733CF | 216478671 | 67AD7DB1 | 2025-02-13 14:05:53 | kakao_test_20250213_1405_jk1 |
| 0x57EA7417 | 30B7DA7DE6077001 | 3510514668823605249 | 0CE733CF | 216478671 | 67AD7DB7 | 2025-02-13 14:05:59 | kakao_test_20250213_1405_jk2 |
| 0x57EA73C7 | 30B7DA8007877000 | 3510514677975576576 | 0CE733CF | 216478671 | 67AD7DB8 | 2025-02-13 14:06:00 | kakao_test_20250213_1405_jk3 |
| 0x57EA7377 | 30B7DA8317077000 | 3510514691120525312 | 0CE733CF | 216478671 | 67AD7DBA | 2025-02-13 14:06:02 | kakao_test_20250213_1405_jk4 |
| 0x57EA7327 | 30B7DA886D077001 | 3510514714038202369 | 0CE733CF | 216478671 | 67AD7DBD | 2025-02-13 14:06:05 | kakao_test_20250213_1405_jk5 |
| 0x57EA72D7 | 30B7DA8BB6077000 | 3510514728147841024 | 0CE733CF | 216478671 | 67AD7DBE | 2025-02-13 14:06:06 | kakao_test_20250213_1405_jk6 |
| 0x57EA7287 | 30B7DA9049877000 | 3510514747802349568 | 0CE733CF | 216478671 | 67AD7DC1 | 2025-02-13 14:06:09 | kakao_test_20250213_1405_jk7 |
| 0x57EA7237 | 30B7DA942A877000 | 3510514764462125056 | 0CE733CF | 216478671 | 67AD7DC3 | 2025-02-13 14:06:11 | kakao_test_20250213_1405_jk8 |
| 0x57EA71E7 | 30B7DA97CB877000 | 3510514780048158720 | 0CE733CF | 216478671 | 67AD7DC4 | 2025-02-13 14:06:12 | kakao_test_20250213_1405_jk9 |
| 0x57EA7196 | 30B7DA9CB6077000 | 3510514801162285056 | 0CE733CF | 216478671 | 67AD7DC7 | 2025-02-13 14:06:15 | kakao_test_20250213_1405_jk10 |

메시지 구조를 기반으로 분석 스크립트를 작성하여 물리 메모리 덤프를 분석한 결과, 총 10건의 카카오톡 메시지가 모두 추출되었다. 이 결과는 동일 기기의 휴대폰을 획득하여 MD-RED로 분석한 값과 메시지 ID, 송신자 ID, 전송 시각, 메시지 본문 등 모든 항목이 일치하였다<그림 10>.

〈그림 10〉 MD-RED 프로그램 분석 결과

| 날짜 | 본문 | 메시지 ID | 채팅방 | 계정 |
|---|---|---|---|---|
| 생성 일시 : 2025-02-13 14:05:53 | 내용 : kakao_test_20250213_1405_jk1 | 3510514618978496512 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:05:59 | 내용 : kakao_test_20250213_1405_jk2 | 3510514668823605249 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:00 | 내용 : kakao_test_20250213_1405_jk3 | 3510514677975576576 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:02 | 내용 : kakao_test_20250213_1405_jk4 | 3510514691120525312 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:05 | 내용 : kakao_test_20250213_1405_jk5 | 3510514714038202369 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:06 | 내용 : kakao_test_20250213_1405_jk6 | 3510514728147841024 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:09 | 내용 : kakao_test_20250213_1405_jk7 | 3510514747802349568 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:11 | 내용 : kakao_test_20250213_1405_jk8 | 3510514764462125056 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:12 | 내용 : kakao_test_20250213_1405_jk9 | 3510514780048158720 | 144164716725612 | 박동현 (216478671) |
| 생성 일시 : 2025-02-13 14:06:15 | 내용 : kakao_test_20250213_1405_jk10 | 3510514801162285056 | 144164716725612 | 박동현 (216478671) |

### 3.3.2. 친구 정보 검색 결과

물리 메모리 덤프를 대상으로 <그림 5>의 구조 해석에 따라 고정된 Hex 시그니처 0D0D083D08을 기준으로 탐색한 결과, MD-RED 도구에서 확인된 총 248건의 친구 정보 중 245건이 추출되어 수집률은 약 98.7%로 나타났다. 분석 시간은 약 1분 40초였다. 추출된 245건의 친구 정보 블록에서는 닉네임, 전화번호, 프로필 이미지 URL, 채팅방 ID, 내부 ID를 각각 분리하여 파싱하였다<그림 11>,<그림 12>.

〈그림 11〉 물리 메모리에서 수집한 친구 정보(건수)

| RecNo | nickname | phone_number | profile_pic_url | chatroom_id_dec | _internal_id_dec |
|---|---|---|---|---|---|
| | | Click here to define a filter | | | |
| 241 | 박 | 01 7 | NO DATA | NO DATA | 270076805 |
| 242 | 서 님 | 01 8 | https://p.kakao XhAIS1r/j3K2TL3Xkel5KyEvivjpN1/x3f55s.jpg | 0 | 180102808 |
| 243 | 사 | 01 0 | https://p.kakao WGv2Eo/Cjsis1ISsTCCuykBeYTEP1/igfcqw.jpg | NO DATA | 265993134 |
| 244 | 소 | 01 7 | https://p.kakao rPocvnv/GeEhZ96jcE2sdKlf25dy4K/imcvkj.jpg | NO DATA | 424139604 |
| 245 | 전 토자 | 01 0 | https://p.kakao bFzAkC/H9oeGQ3y6kuSbJHZGeuWg1/ycmatp.jpg | NO DATA | 272141862 |

〈그림 12〉 카카오톡 메시지 수집 알고리즘 순서도(청크=100MB, 오버랩=1KB)

| 순번 | App | 이름 | 전화번호 | 기타 | ID |
|---|---|---|---|---|---|
| | | | 010 | | |
| 242 | 카카오톡 | 이름 상대 | 0102 :7 | 프로필 이미지 URL : http asDe/76hhiLP4GLgEKktK3royFk/4zlqqc_640x640_s.jpg | 내부 ID : 3675574 |
| 243 | 카카오톡 | 이름 상대 님 나 | 0102 :2 | 프로필 이미지 URL : http OMRJc/hImVu1qMAidloRScodbvtk/wrfwc8_640x640_s.jpg 배경 프로필 이미지 URL : b8ujxn/YnrJVOk5E50PAZw7eqvCN1/a6o96h.jpg | 내부 ID : 129232826 |
| 244 | 카카오톡 | 이름 상대 완 | 0102 :7 | | 내부 ID : 270076805 |
| 245 | 카카오톡 | 이름 상대 준 | 0102 :0 | 프로필 이미지 URL : http iv2Eo/Cjsis1ISsTCCuykBeYTEP1/igfcqw_640x640_s.jpg 배경 프로필 이미지 URL : mMaBRI/CL8ALu23o2IEK5DPg7iCnK/sduejg.jpg | 내부 ID : 265993134 |
| 246 | 카카오톡 | 이름 상대 이 | 0102 7 | 디데이 : 나야 (날짜: 2024 디데이 : 에이티즈 입덕 (날 디데이 : 개학 (날짜: 2025 디데이 : 후추랑ㅁ (날짜: 2 디데이 : 내가 태어난지 (날 디데이 : 아이즈ⓞ (날짜: 디데이 : 도은 (날짜: 2025 배너 : "0" 프로필 이미지 URL : http Csye/rQxv2ljONwl768wD5hbDb0/6jksw9_640x640_s.jpg 배경 프로필 이미지 URL : pefng8b/Q7xaRFlPDrS6MquezljDOk/kq0ywz.jpg 프로필 유직 : Cyberpunk | 내부 ID : 424139604 |
| 247 | 카카오톡 | 이름 상대 종 | 0102 4 | | 내부 ID : 273434 |
| 248 | 카카오톡 | 이름 상대 자 | 0102 0 | 프로필 이미지 URL : http zAkC/H9oeGQ3y6kuSbJHZGeuWg1/ycmatp_640x640_s.jpg 배경 프로필 이미지 URL : vtFL0/Q9FabbjBMFEJA8RlizGvv1/trz76s.jpg | 내부 ID : 272141862 |

MD-RED를 이용한 친구 정보 분석 결과, 시그니처 기반 탐색 기법을 통해 저장매체 내 DB 파일 분석 방식과 구조적 동일한 결과가 확인되었다<그림 13>,<그림 14>.

〈그림 13〉 MD-RED 카카오톡 연락처 분석결과

| App | 이름 | 전화번호 | ID | 기타 | 프로필 이미지 |
|---|---|---|---|---|---|
| 카카오톡 | 이름 : 경 형<br>상대방이 정한<br>이름 : 최 | 0103 59 | 내부 ID : 6465611 | 프로필 업데이트 일시 : 2024-01-02 19:50:29 (UTC+09:00)<br>프로필 접근 일시 : 2024-03-08 15:00:13 (UTC+09:00)<br>프로필 URL : https:// et/talkp/wot82qPPtK/KpIdkmon8pDbWmw92cnc8K/ gazgox.jpg<br>썸네일 URL : https:// et/th/talkp/wot82qPPtK/KpIdkmon8pDbWmw92cnc8K/ gazgox_640x640_s.jp<br>썸네일 경로 : /media lata/com.kakao.talk/cache/ default/40/407ff95aa :1a119692b7e7ece0892b6368a7c4a66cd6e58faa619c<br>배경 프로필 이미지 U .kakaocdn.net/talkp/wlkimNK7LE/ jYkc6earIUlkgkZaDfsy .jpg |  |

〈그림 14〉 물리 메모리에서 추출된 친구 정보의 데이터베이스 저장 결과

| RecNo | min(offset) | nickname | phone_number | profile_pic_url | chatroom_id_dec | _internal_id_dec |
|---|---|---|---|---|---|---|
| | | Click here to define a filter | | | | |
| 1 | 0x75441A5 | 서 | 0106 50 | https://p.ka ıl2jKla/t3w3y9qXP9AkZRsGNOAfgK/talkp.jpg | 144164716725612 | 406257 |
| 2 | 0x753BF60 | 경 | 0103 59 | https://p.ka 2qPPtK/KpIdkmon8pDbWmw92cnc8K/gazgox.jpg | 144165739311474 | 6465611 |
| 3 | 0x752ED46 | 장 | 0106 07 | https://p.ka mTcQlg/krQ0r6eyKZo4pujvayKo11/yjs3fz.jpg | 148874694241352 | 63809480 |
| 4 | 0xBCC293A | 인 | 0102 22 | https://p.ka 7k0y0H/DakoSCpmLFX9LKbgnorKt1/7agacz.jpg | 149575159714819 | 181244032 |
| 5 | 0xBCEEA8F | 김 배님 | 0106 25 | https://p.ka nYc2Kqm/oAmcGJkYrbwRCWWDijn4tk/xcjhr5.jpg | 155357022621157 | 207547713 |
| 6 | 0x753C22B | 토 니어 | 0104 27 | NO DATA | 1554829090689 | 1655769390 |
| 7 | 0xBCE0DD2 | 어 | 0104 86 | https://p.ka JwuIXc/5uP8k0MuDN8NIlzYPVwh01/s6pcl9.jpg | 178080616026343 | 249729883 |
| 8 | 0x754432C | 사 명진 | 0103 76 | NO DATA | 184002413969645 | 359308 |
| 9 | 0x753779D | 송 | 0109 97 | https://p.ka xgXgaj/ddQf4tnvPn5w7AKfGCxRE0/ug86ok.jpg | 185929036242011 | 11776540 |

### 3.4. 시사점 및 한계

카카오톡 메신저 실행 중 생성된 물리 메모리 영역에서 메시지, 친구 정보, 채팅방 정보 등 다양한 데이터가 일정한 구조로 남아 있었다. 특히 삭제된 메시지도 평문 문자열 형태로 존재했으며, 고정된 Hex 시그니처와 반복 패턴을 기반으로 정규표현식을 적용하면 자동 추출이 가능했다. 기존 저장매체 내 DB 파일 분석 방식으로 식별되지 않던 데이터를 확보할 수 있는 새로운 방법이다. 상용 도구(MD-RED)의 결과와 비교한 결과, 모든 분석 항목이 일치하여 구조적 동일성이 확인되었다. 다만 본 실험은 특정 시점의 물리 메모리 덤프를 대상으로 수행되었기 때문에 운영체제의 메모리 관리 방식, 페이지 교체 정책, 시스템 부하 등 환경 변수에 따라 데이터의 잔존 여부와 위치가 달라질 수 있다. 아울러 메시지와 친구 정보의 시그니처 기반 탐색은 메신저 애플리케이션 업데이트나 내부 저장 구조 변경 시 일부 변형 구조에 대응이 어려운 한계를 가진다. 향후 다양한 시스템 설정, 버전, 시간 경과 조건에 따른 데이터 잔존 여부에 대한 추가 분석이 필요하다.

## Ⅳ. 가상메모리에서 카카오톡 메신저의 대화내용 구조 분석

### 4.1. 실험설계

본 연구는 Windows 가상메모리 영역(pagefile.sys, hiberfil.sys)에 잔존하는 카카오톡 PC 메신저의 삭제 메시지를 추출하여 획득하는 기법을 제시하는 것을 목적으로 한다. 삭제 방식과 시스템 상태를 달리하여

반복적으로 실험을 수행하였다.

수집 대상은 Windows 10 Pro 데스크톱(8GB RAM)에서 pagefile.sys, Windows 11 Pro 노트북(16GB RAM)에서 hiberfil.sys를 확보하였다. 데이터 비교 검증을 위해 Android 15 기반 Samsung Galaxy S25(SM-S931N)에서 카카오톡 모바일 버전(2025년 최신)을 사용하였다.

실험 내용은 메시지 삭제 방식(나에게서 삭제, 채팅방 나가기, 프로그램 제거 등)과 시스템 상태(재부팅, 절전 모드, 활성 상태)를 달리하여 반복 시나리오를 구성하고, 각 조건에서 메시지 잔존 여부와 복구 가능성을 확인하는 것이다.

실험 도구는 FTK Imager와 Procdump를 활용하여 가상메모리 파일을 수집하였고, Hibr2Bin을 사용하여 hiberfil.sys를 RAW 포맷으로 변환하였다. 분석은 WinHex를 통한 Hex 구조 확인과 Python 기반 정규표현식 스크립트로 수행하였으며 복구 결과의 신뢰성을 검증하기 위해 MD-RED 상용 도구와 비교하였다.

실험 절차는 ① 검증 메시지 전송, ② 사용자 행위 수행(삭제, 채팅방 퇴장), ③ 메모리 수집, ④ 시그니처 기반 탐색, ⑤ 기존 도구와의 결과 비교의 순서로 진행하였다. 메시지는 1초 간격으로 100건씩 자동 전송하였고, 각 메시지는 KAKAO-{NML,DEL,EXT}-MSG-YYYYMMDD-HHMMSS-SEQ-END 형식으로 구분하였다. 실험 시간 단축을 위해 CPU와 메모리에 부하를 가하여 스와핑을 유도하였다. pagefile.sys는 모든 시스템에서 활성화되며, hiberfil.sys는 노트북에서만 기본적으로 활성화되고 데스크톱에서는  비활성화 상태다. hiberfil.sys는 절전 모드나 하이브리드 절전이 설정된 경우에만 생성된다<표 6>.

〈표 6〉 Windows 10과 Windows 11 환경에서 pagefile.sys 및 hiberfil.sys 활성 여부

| OS / 장치 | | pagefile.sys | hiberfil.sys |
|---|---|---|---|
| Win 10 | 데스크탑 | O | X |
| | 노트북 | O | O |
| Win 11 | 데스크탑 | O | X |
| | 노트북 | O | O |

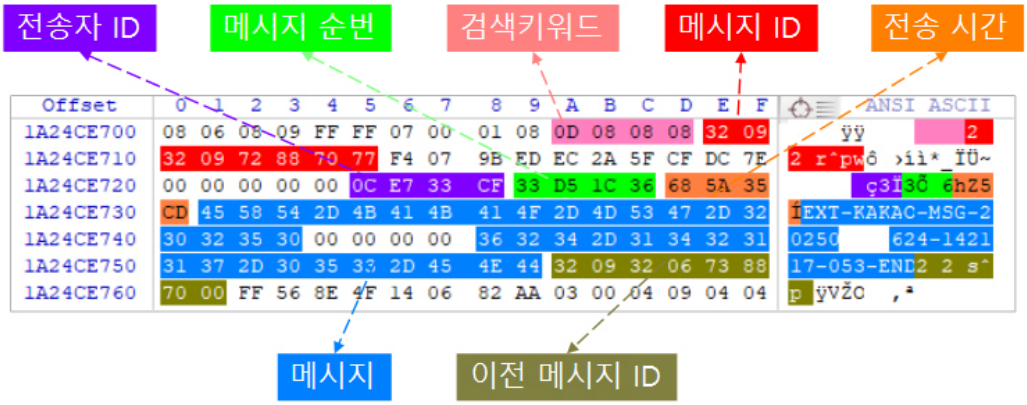
실험 시간을 단축하고 특정 시점의 데이터를 확보하기 위해 CPU와 메모리에 높은 부하를 주어 스와핑이 발생하도록 유도하였다. CPU 부하는 논리 프로세서 수에 맞춰 다중 스레드를 생성하고, 각 스레드에서 곱셈·모듈로 연산을 무한 반복하여 GIL(Global Interpreter Lock)을 우회하며 지속적으로 점유하도록 하였다. 물리 메모리 부하는 100MB 크기의 bytearray 객체를 0.1초 간격으로 연속 동적 할당하여, 시스템 RAM 전체 용량(8GB 기준 약 80회)에 도달할 때까지 반복되도록 구성하였다. 이 조건은 메모리 점유율을 100%까지 끌어올려 운영체제가 강제로 스와핑을 수행하도록 만든다. 결과적으로 자연 환경에서는 수십 분에서 수 시간 이상 소요될 수 있는 스와핑 현상을 실험 환경에서 수분 이내에 재현할 수 있도록 강제 스와핑 조건을 설계하였다.

## 4.2. 가상메모리 저장 구조 분석

### 4.2.1. pagefile.sys 기반 실험

pagefile.sys에 저장된 메시지 구조는 기존 메모리(RAM) 구조와 항목 배치 순서 면에서 유사하나, 항목 간 불필요한 값이 삽입되거나 공백 바이트(0x00)가 포함되어 차이가 있다. 스와핑 과정에서 메모리 페이지가 디스크에 기록될 때 페이지 프레임 단위로 재배치되거나 패딩이 삽입되는 Windows 메모리 관리자 동작의 결과로 해석된다<그림 15>.

〈그림 15〉 pagefile에 저장된 카카오톡 메시지 구조




총 100건의 메시지 중 전송 후 별도의 사용자 조작 없이 유지된 메시지(NML)는 41건, '나에게서 삭제'(DEL)는 28건, '채팅방 나가기'(EXT)는 33건이 복구되었다<표 7>. 세 유형 모두 동일한 시스템 환경에서 전송 후 12시간이 경과한 시점에 pagefile.sys를 수집하여 분석하였으며, 실험 동안 메신저는 활성 상태를 유지하고 타 사용자와의 대화도 지속하였다.

삭제 방식별 복구율 차이는 메모리 페이지의 참조 유지·해제 방식 차이로 설명할 수 있다. '채팅방 나가기'는 UI 구성 요소 제거와 리스트 갱신 과정에서 해당 객체에 대한 참조가 해제된다. Chan 등[13]은 참조 비트가 클리어된 페이지가 LRU 기반 페이지 교체 정책에 따라 빠르게 Standby List 또는 Free List로 이동한다고 보고하였다. Sylve 등[11] 역시 이와 유사하게 클리어된 페이지가 스와핑 대상으로 지정될 가능성이 높다고 설명하였다. 반면 '나에게서 삭제'는 플래그 변경에 국한되어 주변 페이지가 활성 상태를 유지하고 참조 비트도 유지되기 때문에, Russinovich 등[15]이 설명한 참조 유지 페이지의 스와핑 우선순위 하락 현상에 따라 pagefile.sys에 포함되지 않는다.

### 4.2.2. hiberfil.sys 기반 실험

총 100건의 메시지를 대상으로 1시간 경과 후 hiberfil.sys 복구를 수행한 결과, 정상 메시지(NML)는 65건, 나에게서 삭제(DEL)는 40건, 채팅방 나가기(EXT)는 37건이 복구되었다. 복구율 차이는 각 메시지 객체의 메모리 참조 상태에 따라 발생하였다. 정상 메시지는 UI와 백그라운드 프로세스에 의해 지속 참조되어 하이버네이션 시점까지 유지되었다.

나에게서 삭제'는 본문이 0x00으로 덮여도, 해당 페이지가 UI 스레드·캐시 매니저에 의해 참조되어 작업셋(Active) 상태로 평균 46시간 유지되었다. Russinovich 등은 Windows 메모리 관리자가 참조 비트를 유지하는 페이지를 Working Set 트리밍이나 페이지 폴트 처리 시에도 우선적으로 Standby List를 거쳐 스와핑한다고 설명하였는데[15] 본 실험에서 해당 페이지가 장기간 활성 상태로 유지된 결과와 부합한다. Chan 등(2011)이 보고한 바와 같이, 해제된 데이터 구조가 새로운 데이터로 덮어쓰기되기 전까지 메모리에 잔류하는 데이터 잔류현상(data remanence)이 확인되었다[15].

'채팅방 나가기' 조건에서는 UUID 제거 이후 약 1~2분 내에 참조 비트가 클리어(clear)되어 페이지가 비활성 상태로 전환되었고, LRU 기반 페이지 교체 정책에 따라 Standby List 또는 Free List로 신속하게 이동하였다. Sylve 등(2016)의 pool tag 스캐닝 기법 분석 결과에서도, 참조가 제거된 객체는 메모리에서 빠르게 회수되어 하이버네이션 파일(hiberfil.sys)에 포함될 가능성이 낮다고 보고되었는데[11], 본 실험 결과와 일치한다.

'프로그램 제거' 조건에서는 프로세스 종료 직후에도 일부 페이지가 Standby List에 잔존하거나 Zero Page Thread에 의해 지연 초기화되어 메모리 데이터가 단기적으로 유지되었다. Chan 등(2011)의 연구에서도 이와 유사하게, 프로세스 종료 이후에도 메모리 페이지가 일정 기간 덮어쓰기되지 않고 유지될 수 있음이 확인되었다[13].

재부팅 후 hiberfil.sys에서는 기존 데이터가 식별되지 않았다. 그 이유는 Windows가 부팅 시 커널 초기화 단계에서 하이버네이션 파일을 검사하고, 최대절전 모드 복원이 아닌 경우 해당 파일을 무효화하기 때문이다. 이 과정에서 파일의 헤더 시그니처(HIBR → WAKE)가 변경되거나 내부 페이지가 0x00 패딩으로 덮어써져 초기화된다. Windows 전원 관리 정책상 최대절전 모드는 메모리를 hiberfil.sys에 저장한 뒤 전원을 차단하고, 재개 시 이를 기반으로 시스템 상태를 복원한다. 반면 일반적인 재부팅은 해당 절차를 수행하지 않으며 파일의 무결성을 보장하기 위해 기존 데이터가 제거된다.

메시지 전송 후 삭제 방식에 따른 복구 건수를 분석한 결과, 메모리 잔존 데이터 수는 행위 유형에 따라 유의한 차이를 보였다<표 7>.

〈표 7〉 가상메모리 별 카카오톡 메시지 복구 건수(단위: 건, 총 100건 기준)

| 대상 구분 | 메시지 전송 | 나에게서 삭제 | 채팅방 나가기 | 프로그램 제거 |
|---|---|---|---|---|
| pagefile.sys | 41 | 28 | 33 | 영향 없음 |
| hiberfil.sys | 65 | 40 | 37 | 영향 없음 |

#### 4.2.3. MD-RED의 데이터 추출 방식과 비교

MD-RED는 모바일 단말의 애플리케이션 아티팩트를 기반으로 메시지 데이터를 분석하는 상용 포렌식 도구로, 대상 단말에서 확보한 SQLite 데이터베이스(main.db), WAL/Journal 파일, 캐시, 설정 파일 등을 스키마 기반 파서를 이용하여 해석한다. 해당 과정에서 메시지, 대화방, 계정, 미디어 메타데이터를 추출하며, 암호화된 필드는 애플리케이션 내부 키 또는 키체인 정보를 활용하여 복호화한다. 논리 삭제된 레코드나 WAL/Journal 병합 전 페이지에 남아 있는 삭제 전 데이터 일부를 복구할 수 있다. 다만, 애플리케

이션 제거, TRIM 명령 실행 이후에는 복구가 불가능하다.

본 연구의 데이터 분석 절차는 MD-RED와 달리 비휘발성 가상메모리(pagefile.sys, hiberfil.sys)를 대상으로 RAW 단위의 시그니처 기반 탐색을 수행한다. RAM 분석을 통해 규명한 메시지 구조를 토대로 정규표현식 및 패턴 매칭을 적용하여 메시지 블록을 카빙한 후, 바이트열을 정수·문자열로 디코딩하여 message_id, sender_id, room_id, timestamp, body 등의 필드를 재구성한다. MD-RED로 분석한 결과와 메시지 ID, 전송 시각, 송신자 ID, 본문 데이터의 일치 여부를 비교하여 정합성을 검증하였다.

두 방식 모두 메시지 메타데이터와 본문을 구조화하여 ID, 시간, 본문을 기준으로 결과를 제공한다는 공통점이 있다. 그러나 MD-RED는 애플리케이션의 데이터베이스와 WAL/Journal 파일 등 비휘발성 저장소를 기반으로 스키마 파싱을 수행하는 반면, 본 연구는 데이터베이스 파일이 없어도 가상메모리 파일에 남은 데이터를 직접 카빙하여 복구할 수 있다는 차이가 있다. MD-RED는 체크포인트 병합과 SSD TRIM 동작 이후에는 복구가 불가능하지만, 본 연구는 RAM(물리 메모리)이 가상메모리 파일로 스와핑된 시점의 상태를 활용하여 재부팅 이후에도 잔존 데이터의 복구가 가능하다. 다만 데이터베이스 파일이 존재하는 경우에는 MD-RED에 비해 복구 범위가 제한되고, 모든 대화를 완전히 추출할 수 없다는 한계가 있다.

#### 4.2.4. 일반 사용 환경 복구 사례

일반 사용자 환경에서 OS의 기본 메모리 관리 정책에 따라 자연적으로 스와핑이 발생한 시스템을 대상으로 가상메모리 파일 확보 및 복구를 수행하였다. 분석 대상은 실제 분석 의뢰물 PC였으며, 현장에서 복제 장비를 사용하여 하드디스크 전체를 증거이미지(E01)로 생성한 후, 이미지 파일에서 pagefile.sys와 hiberfil.sys를 추출하였다. 스와핑 발생 시점은 확인할 수 없었으나, 추출한 가상메모리 파일에서 카카오톡 메신저 대화 데이터 일부가 복구되었다. 특히, 분석 의뢰물 중 메신저 애플리케이션이 이미 삭제되어 데이터베이스 기반 분석이 불가능한 사례에서도 가상메모리 분석을 통해 메시지 복구가 가능함을 확인하였다.

Windows 10 Pro 데스크톱 시스템에서 확보한 pagefile.sys의 분석 결과와 Windows 11 Pro 노트북에서 확보한 hiberfil.sys의 분석 결과를 보면, 모두 메신저 애플리케이션이 삭제되어 데이터베이스 기반 분석이 불가능한 상황에서 복구가 이루어졌다는 것을 알 수 있다<그림 16>, <그림 17>.

〈그림 16〉 pagefile.sys를 분석한 결과

| 전송일자 | 발신자 | 메시지ID | 수신자 | | offset |
|---|---|---|---|---|---|
| 2025-06-10 ... | 420111211 | 3595490765377898... | | 응응 알 | 0x43E2B9A |
| 2025-06-10 ... | 10241371 | 3595490599123810... | | 난 10시 | 0x43E2BE0 |
| 2025-06-10 ... | 420111211 | 3595490423131080... | | 커플인증 | 0x43E2C6B |
| 2025-06-10 ... | 420111211 | 3595490385843718... | | 회원가입 | 0x43E2CC0 |
| 2025-06-10 ... | 420111211 | 3595490325168916... | | 커플이벤 | 0x43E2D0D |
| 2025-06-10 ... | 420111211 | 3595490185230157... | | 진언니가 | 0x43E2D71 |
| 2025-06-10 ... | 420111211 | 3595490101830617... | | 아니앙~ | 0x43E2DCF |
| 2025-06-10 ... | 10241371 | 3595489970666076... | | 근데~~ | 0x43E2E0F |
| 2025-06-10 ... | 420111211 | 3595485683424954... | | 다 러뷰 | 0x43E2EA7 |
| 2025-06-10 ... | 420111211 | 3595485637446993... | | 요즘 안 | 0x43E2EF5 |
| 2025-06-10 ... | 420111211 | 3595485515787012... | | 응응 나 | 0x43E2F57 |
| 2025-06-10 ... | 10241371 | 3595478503262779... | | 그런것도 | 0x43E2FA9 |
| 2025-06-10 ... | 420111211 | 3595469135142455... | | 그걸로도 | 0x43E2FF6 |

〈그림 17〉 hiberfil.sys의 분석 결과

| 전송일자 | 발신자 | 메시지ID | 수신자 | | offset |
|---|---|---|---|---|---|
| 2025-07-27 ... | 293858915 | 3629205643406743... | | [톡 | 0x3E86AD6 |
| 2025-07-27 ... | 210171848 | 3629426978749646... | | 우 | 0x34D4E8E |
| 2025-07-27 ... | 36488214 | 3629557918167855... | | ht | 0x206F3C2 |
| 2025-07-27 ... | 266806891 | 3629377872870600... | | 전 | 0x52CC673 |
| 2025-07-27 ... | 150137528 | 3629373918546503... | | 네 | 0x52CCA66 |
| 2025-07-27 ... | 241000755 | 3629403445516840... | | ht | 0x49CD7C8 |
| 2025-07-27 ... | 210171848 | 3629424781823543... | | 지 | 0x1DD3374 |
| 2025-07-27 ... | 50889240 | 3629420066745344... | | ■ | 0x1DD380D |
| 2025-07-27 ... | 150137528 | 3629289032218439... | | ● | 0x35BFDCA |
| 2025-07-27 ... | 417058569 | 3629288962291027... | | 지 | 0x35BFEE4 |
| 2025-07-26 ... | 417031757 | 3628899109802895... | | l... | 0x4DA03E |
| 2025-07-26 ... | 179320661 | 3628951722464413... | | 타 | 0x4DA312 |
| 2025-07-26 ... | 407677559 | 3628943707066957... | | 명 | 0x4DA465 |

## 4.3. 요약 및 분석절차 제언

### 4.3.1. 실험 결과

기존 저장매체 기반 DB 분석에서는 ‘나에게서 삭제’와 ‘채팅방 나가기’ 메시지가 모두 복구되지 않았다. 삭제 직후와 6시간 경과 후 결과가 동일했으며, SSD의 TRIM 동작과 DB의 WAL/JOURNAL 병합 처리로 인해 삭제 직후부터 복구 가능성이 급격히 낮아진 기존 연구와 동일한 결과다.

반면 가상메모리 기반 분석에서는 삭제된 메시지 복구가 가능했다. pagefile.sys에서는 ‘정상 전송’ 41건, ‘나에게서 삭제’ 28건, ‘채팅방 나가기’ 33건이 복구되었고, hiberfil.sys에서는 각각 65건, 40건, 37건이 복구되어 더 높은 성능을 보였다. 이는 hiberfil.sys가 시스템 전체 메모리를 스냅샷 형태로 저장하는 구조적 특성 때문이며, 운영체제의 메모리 관리 과정에서 삭제 데이터가 일정 시간 후에도 디스크에 잔존할 수 있기 때문이다.

시간 경과에 따른 복구율 비교 결과, pagefile.sys의 복구 건수는 삭제 직후 12건, 6시간 후 33건, 12시간 후 45건으로 변동 폭이 컸다. 작업셋 트리밍과 메모리 압력에 따라 유휴 페이지가 점진적으로 스와핑되면서 일정 시간 후 복구 기회가 발생하는 메커니즘 때문이다. hiberfil.sys는 12시간 경과 후 63건을 유지하여 상대적으로 안정적인 복구율을 보였다. 시스템이 최대 절전 모드로 전환될 때 전체 메모리 상태가 hiberfil.sys에 스냅샷 형태로 온전히 보존되는 구조적 특성 때문이다.

삭제 방식별 비교에서는 ‘채팅방 나가기’가 UI 정리와 리스트 삭제 등 부가 연산을 동반하여 메모리 교란 수준이 높았고, 복구율이 상승했다. 반면 ‘나에게서 삭제’는 구조 변화가 적어 복구율이 낮았다. 이러한 차이는 저장소 기반 복구가 삭제 직후부터 회수 가능성이 사라지는 것과 달리, 가상메모리 기반 복구율은 확보 시점과 시스템 상태(메모리 압력, 하이버네이션 여부)에 따라 변동한다<표 8>.

〈표 8〉 삭제 후 경과 시간에 따른 가상메모리 및 기존 DB 분석 방식별 메시지 복구 건수

(단위: 건, 총 100건 기준)

| 대상 | 행위 직후 | 6시간 후 | 12시간 후 | 평균 복원율(%) |
|---|---|---|---|---|
| pagefile.sys | 12 | 33 | 45 | 30 |
| hiberfil.sys | 65 | 59 | 63 | 62.3 |
| 기존 DB분석 방식 | 0 | 0 | 0 | 0 |

본 연구의 복구율 측정은 Windows 10(64bit), 동일 하드웨어 사양, 8GB RAM 환경을 표준 조건으로 수행하였다. 4GB, 8GB, 16GB 환경의 예비 실험 결과, Russinovich 등[15]이 설명한 바와 같이 동일 OS에서는 메모리 크기와 무관하게 사용률이 약 70~80%를 초과하면 동일한 스와핑 메커니즘이 작동함을 확인하였다. 16GB 이상 환경은 페이지 파일이 커져 반복 실험에 시간이 과도하게 소요되어 제외하고, 반복성과 장기 관찰(100회 이상, 최대 7일)을 위해 8GB 환경을 기준으로 수치를 산출하였다.

메시지 삭제 방식에 따른 복구 가능 여부를 기존 도구(MD-RED, 저장매체 내 DB 파일 분석 방식)와 가상메모리 기반 분석 기법으로 비교하였다. 그 결과, 기존 방식은 ‘모든 대화상대에서 삭제’를 제외한 나머

지 방식에서는 모두 복구가 불가능하였다. 반면, 가상메모리 기반 분석은 삭제 방식과 무관하게 hiberfil.sys 기준 평균 62.3%의 복구율을 보였다<표 9>.

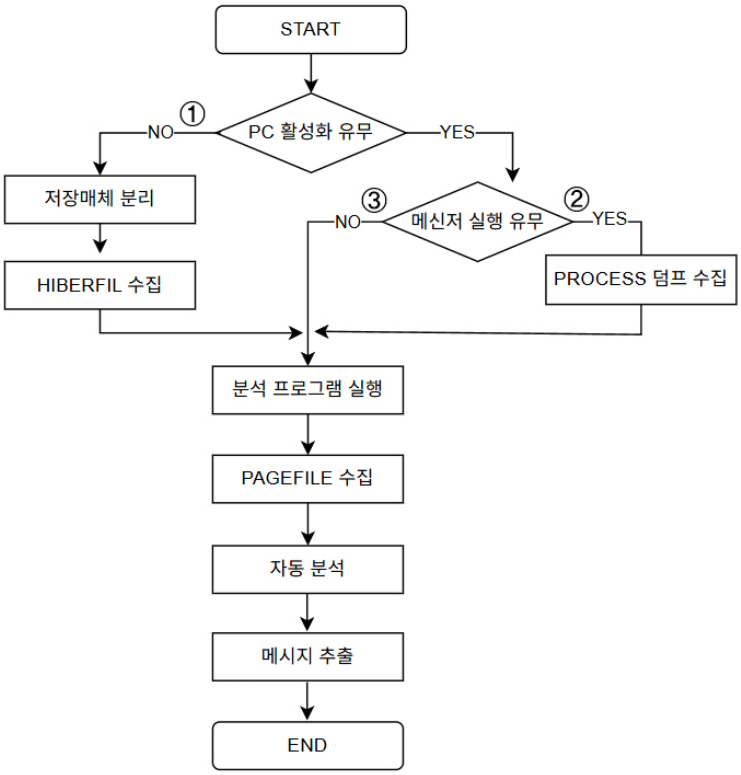
〈표 9〉 삭제 방식별 분석 도구별 메시지 복구 가능 여부

| 실험 항목<br>(삭제 방식) | 모바일 분석<br>(MD－RED) | 디스크 분석<br>(DB기반 기존 분석 방식) | 가상메모리 분석 |
|---|---|---|---|
| 모든 대화상대에서 삭제 | 0 | 0 | 0 |
| 이 기기에서만 삭제 | X | X | 0 |
| 채팅방 나가기 | X | X | 0 |
| 프로그램(애플리케이션) 삭제 | X | X | 0 |

### 4.3.2. 분석절차 제언

Windows 가상메모리 영역에 잔존하는 메신저 데이터를 효과적으로 수집 및 분석하기 위한 절차를 제안한다.<그림 18> 시스템의 상태에 따라 최적의 분석 경로를 선택하도록 하였다. ① 시스템 전원이 꺼진 경우에는 hiberfil.sys 파일을 우선적으로 확보한다. 시스템 종료 직전의 메모리 상태를 가장 온전하게 담고 있을 가능성이 높기 때문이다. 단, 시스템 전원이 켜진 상태에서는 hiberfil.sys 파일이 운영체제에 의해 초기화되므로, 해당 파일을 별도로 수집할 필요는 없다. ② 분석 대상 메신저가 실행 중인 경우,

〈그림 18〉 가상메모리 디지털증거 수집 절차



해당 프로세스의 메모리를 직접 덤프하여 분석한다. 이 방식은 스와핑되지 않은 실시간 데이터를 확보하는 데 가장 효과적이다. ③ 메신저가 종료되었으나 시스템은 계속 구동된 경우에는 pagefile.sys를 분석하여 스와핑된 데이터 흔적을 탐색한다. 제안된 절차는 기존 저장매체 분석의 한계를 보완하는 실질적인 분석 수단이 될 수 있다. 다만, 가상메모리 기능이 비활성화되었거나 절전 모드를 사용하지 않는 등, 시스템 설정에 따라 관련 파일이 생성되지 않을 수 있으므로 압수수색 현장에서는 사전 설정 확인이 반드시 필요하다<그림 18>.

## Ⅴ. 결론

본 연구는 저장매체 내 DB 파일 기반 분석으로는 복구가 어려운 메신저 삭제 메시지를, 가상메모리 영역에서 탐지·복구할 수 있는 기법을 제안하였다. 제2장에서는 메신저 삭제 방식과 기존 복구 기법의 한계를 검토하였고, 제3장에서는 카카오톡 PC 메신저 실행 중 메모리에 저장되는 메시지와 사용자 정보의 구조를 분석하여 시그니처 기반 탐지를 위한 패턴을 정리하였다. 제4장에서는 pagefile.sys와 hiberfil.sys를 대상으로 실험을 수행하고, 삭제 메시지를 정규표현식 기반으로 자동 탐지·복구하는 스크립트를 구현하였다. 실험 절차는 ① 검증 메시지 전송, ② 사용자 행위 수행, ③ 메모리 수집, ④ 시그니처 기반 탐색, ⑤ 결과 비교의 단계로 진행하였다.

실험 결과, 평균 복구율은 hiberfil.sys에서 62.3%, pagefile.sys에서 30%로 나타났으며, DB 파일 기반 분석으로 확보할 수 없는 삭제 메시지까지 복구 가능함을 확인하였다. 특히 제안된 기법은 메시지 본문뿐 아니라 송신자와 전송 시간 등 구조화된 대화 정보를 함께 복구할 수 있어 수사 실무에서의 활용 가능성이 높다. 연구의 한계점은 Windows 운영체제와 카카오톡에 국한되며, 시스템 부하와 시간 경과에 따라 복구율이 달라질 수 있다는 점이다. 스와핑 시점을 단축하기 위해 CPU와 메모리에 고부하를 주는 강제적 조건을 적용하였으나, 선행 연구[14][15]와 같이 실제 환경에서도 발생 가능한 수준으로 복구율 또한 자연 발생 스와핑과 유사한 결과를 보였다. 이로써 실제 수사 환경에서의 적용 가능성을 확인하였다.

향후 다양한 운영체제와 메신저를 반영하고, 구조 변화에 유연하게 대응할 수 있는 탐지 모델을 고도화할 필요가 있다. 제안된 기법은 자동화 프로그램으로 구현되어, 메시지 삭제와 애플리케이션 제거 등 안티포렌식 환경에서도 효과적인 수사 지원 도구로 활용될 수 있다.

## 참 고 문 헌 (References)

[1] Lee, H., Chung, S., and Lim, J. "Forensic analysis of mobile instant messenger applications on An droid smartphones", Digital Investigation, 11(3), pp.197－213, 2014.

[2] H. Heath, Á. MacDermott, and A. Akinbi, “Forensic analysis of ephemeral messaging application s: Disappearing messages or evidential data?”, Forensic Science International: Digital Investigati on, 46, 2023.

[3] B. Jung, J. Han, H. Choi, and S. Lee, “A study on the possibility of recovering deleted data thro ugh analysis of SQLite journal in messenger application”, Journal of Digital Forensics, 12(2), pp.11-20, Sep. 2018.

[4] M.－D. Kim, H.－J. Lee, S. Lee, Y.－J. Lee, and G.－B. Kim, “A study on the permissibility of search and seizure of mobile messenger data using a USIM”, Journal of The Korea Institute of Informati on Security & Cryptology, 31(2), pp.197-209, Apr. 2021.

[5] M. Cho and N. Jang, “Study on the data decryption and artifacts analysis of KakaoTalk in Wind ows environment”, Journal of The Korea Institute of Information Security & Cryptology, 33(1), pp.51－61, 2023.

[6] Shin, D., and Shon, T. “Analysis of personal identification information exposure using memory dum p－based forensics”, Journal of Digital Forensics, 15(2), pp.35-49, 2021.

[7] B.－C. Yoon, S.－R. Kim, and J.－S. Kim, “A study on database encryption process analysis and del eted message recovery in MalangMalang Tok－Cafe on Windows”, Journal of The Korea Institute o f Information Security & Cryptology, 30(3), pp.397-403, Jun. 2020.

[8] M. Rawashdeh, N. Alzahrani, M. T. Alshamrani, and Y. Al－Hadhrami, “On the forensics of pagefi le.sys in Windows systems”, International Journal of Computer Applications, 187(3), pp.15-23, 2024.

[9] Ghafarian, A., and Keskin, D. “Windows 10 hibernation File Forensics”, In Communications in Com puter and Information Science, 1231(), pp.367-378, 2020. https://doi.org/10.1007/978－3－030－5 2243－8_31.

[10] Hwang, H. “An improved extraction technique of messenger user information in physical memory”, Master’s thesis, Chonnam National University, 2015.

[11] Sylve, J., Marziale, V., and Richard, G.G., “Pool tag quick scanning for Windows memory a nalysis”. Digital Investigation, 16, pp.25-32, 2016.

[12] Ahmad, A., and Hussain, M., “A forensic analysis of video streaming activities on Android applications”, Mobile and Forensics, 4(1), pp.44-52, 2022.

[13] E. Chan, S. Venkataraman, N. Tkach, K. Larson, A. Gutierrez and R. H. Campbell, "Characterizing Data Structures for Volatile Forensics“, 2011 Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering, (Oakland, CA, USA,) pp.1－9, 2011.

[14] Fernández－Fuentes, X., Nieto, A., Rivas, J., and Ribagorda, A., “Digital forensic analysis methodology for private Browse: Firefox and Chrome on Linux as a case study”, Computers & Security, 115, 2022.

[15] Russinovich, M., Solomon, D., and Ionescu, A. “Windows Internals, Part 1: System archite cture, processes, threads, memory management, and more(7th ed.):, Microsoft Press, 2017.

[16] GMD Soft. MD－RED Product Overview. [Online]. Available: https://www.gmdsoft.com/ko/ir/press/md－red－mobile－forensic－software－for－data－analysis/ 2025.08.30. confirmed.

# 저 자 소 개

박 동 현 (Donghyun Park)
준회원

2021년 2월~현재 : 서울경찰청 사이버수사과 디지털포렌식계 분석관
2024년 9월~현재 : 성균대학교 디지털포렌식학과 석사과정
관심분야 : 디지털포렌식, 사이버범죄추적

이 상 은 (Sangeun Lee)
준회원

2017년 8월 : 상명대학교(서울) 컴퓨터과학과 이학사
2022년 8월 : 성균관대학교 과학수사학과 석사
2022년 9월~현재 : 성균관대학교 과학수사학과(디지털포렌식) 박사수료
관심분야 : 디지털포렌식, 사이버범죄 수사, 형사소송법 등

김 기 범 (Gibum Kim)
평생회원

2017년 2월 : 고려대학교 정보보호대학원 박사
2014년 2월~2020년 2월 : 경찰대학 경찰학과 교수요원/국제사이버범죄연구센터장
2020년 3월~현재 : 성균관대학교 과학수사학과(디지털포렌식) 부교수
관심분야 : 디지털포렌식, 사이버범죄, 범죄수사, 과학기술치안, 국제개발협력 등