

PC 환경에서 보안 메신저 Wire 앱의 사용자 데이터 캐싱 메커니즘 연구

조민지*, 정병찬**, 이상진***, 박정흠****
고려대학교 일반대학원 정보보안학과 (대학원생)*,
고려대학교 정보보호대학원 (대학원생)**, (교수)*** (조교수)****

A Research on User Data Caching Mechanism of Wire Messenger in PC

Minji Cho*, Byeongchan Jeong**, Jungheum Park***, Sangjin Lee***

Dept. of Information Security, Graduate School, Korea University(Graduate Student)*
School of Cybersecurity, Korea University(Graduate Student)**, (Assistant Professor)***, (Professor)****

요약

오늘날 텔레그램(Telegram), 와이어(Wire), 디스코드(Discord), 위커(Wickr) 등 보안 메신저가 많은 범죄 사건에서 범죄의 수단으로 악용되고 있다. 사용자들은 개인정보보호의 이슈로 인해 조금이라도 더 안전한 메신저를 찾고 있다. 이러한 메신저들에 강력한 보안 기술이 적용됨으로써 디지털 포렌식 수사는 더욱 어려워지고 있다.

하지만, 메신저의 데이터는 증거로써 중요한 가치를 가지고 있어 이에 관련된 연구가 필수적이다. 본 논문에서는 PC 환경에서 메신저 와이어의 캐싱 메커니즘을 분석하여 사용자 데이터를 획득하는 방법을 제안한다. 또한, 본 연구의 방법론이 적용된 도구를 개발하였으며 이를 통해 실제 범죄 수사 과정에 도움을 줄 것으로 기대한다.

주제어 : 디지털 포렌식, 보안 메신저, Electron Framework, Chrome Web Cache, LevelDB

ABSTRACT

Today, secure messengers such as Telegram, Wire, and Discord have been used as a means of crime in many criminal cases, and the need for related research has been raised. Most users are looking for a safer messenger because of the issue of privacy problem. As strong security technology is applied to these messengers, digital forensic investigations are becoming more difficult.

However, messenger data has important value as evidence, so research related to it is essential. In this paper, we propose an investigation methodology for acquire user data by analyzing the caching mechanism generated by Messenger Wire. Then, we developed a tool which the methodology of this study is applied and it is expected that this will help an efficient investigation methodology to identify related crimes.

Key Words : Digital Forensics, Secure Messenger, Electron Framework, Chrome Web Cache, LevelDB

1. 서론

오늘날 메신저는 사람들 간의 관계를 유지하고 소통하는 필수 수단이다. 메신저를 통해 메시지, 통화, 파일 공유 등 다양한 기능을 사용하기 때문에 메신저 내 데이터에 사용자의 정보들이 많이 저장되어 있다. 최근 사용자의 개인정보보호에 대한 인식이 제고됨에 따라 보안 기능이 강화된 메신저에 대한 수요가 증가하고 있다.

따라서, 보안 기능이 강화된 메신저들을 악용한 피싱, 디지털 성범죄 등과 같은 범죄가 빈번하게 발생하고

※ 이 성과는 과학기술정보통신부·경찰청이 공동 지원한 '폴리스랩2.0 사업(www.kipot.or.kr)'의 지원을 받아 수행된 연구결과임
(과제명: 안티-포렌식 기술 대응을 위한 데이터 획득 및 분석 기술 연구 / 과제번호: 210121M07)

Received 28 February 2022, Revised 24 March 2022, Accepted 05 June 2022

제1저자(First Author) : Minji Cho (Email : cmj101804@korea.ac.kr)

교신저자(Corresponding Author) : Jungheum Park (Email : jungheumpark@korea.ac.kr)

있다. 2020년 발생한 n번방 사건에서도 텔레그램뿐만 아니라 와이어, 디스코드, 위커와 같은 보안 메신저도 함께 이용되었다. 와이어 메신저는 100만 명 이상의 사용자가 이용하는 오픈소스 메신저로 PC(Windows, macOS, Linux)와 모바일(Android, iOS), 웹(Web) 환경 모두를 지원한다. 또한, 와이어 메신저는 텔레그램보다 보안성이 뛰어난 메신저로 평가받아 더욱 범죄에 악용되고 있다[1][2].

와이어 메신저에 관한 기존 연구에서는 메신저 내 채팅 데이터의 일부만 다루고 있으며 로컬 PC 내 저장된 전체 대화 내용과 첨부파일은 다루지 않았다[3]. 이러한 메신저의 대화 내용과 암호화된 첨부파일에 관한 연구는 기업 기밀 유출 사건, 디지털 성범죄 등과 같은 범죄 수사 과정 중 중요한 단서로 활용될 수 있으므로 선제적으로 대응할 수 있는 연구가 필수적이다.

이에 본 연구에서는 로컬 PC에 남아있는 와이어 메신저 내 저장된 모든 데이터를 분석하여 대화 내용을 재구성하고 암호화된 첨부파일에 대한 암호 알고리즘을 분석하여 복호화 하는 연구를 수행하였으며 이를 자동화 하는 도구를 개발하였다.

본 논문의 2장에서는 관련된 연구와 배경지식에 대하여 설명한다. 3장에서는 와이어 메신저 내 존재하는 아티팩트 및 데이터 분석을 하고 4장에서는 이를 활용하여 메신저 내 전체 대화 내용을 재구성하고 암호화된 첨부파일을 복호화 하는 방법을 다룬다. 마지막 5장에서는 결론 및 향후 연구 계획, 한계점으로 본 논문을 마무리한다.

II. 관련 연구 및 배경지식

2.1 배경지식

2.1.1 Electron Framework

Electron Framework은 Chromium 엔진과 Node.js 그리고 CSS, HTML, JavaScript 언어를 이용하여 Windows, MacOS, Linux 환경에서 애플리케이션을 개발 작업을 지원하는 크로스 플랫폼 프레임워크이다. Electron Framework를 이용하면 Node.js로 Local 시스템에 접근하여 Chromium으로 화면을 구성하며 각 OS 버전에 따라 애플리케이션을 패키징 할 수 있다. 해당 프레임워크로 만들어진 애플리케이션으로는 디스코드(Discord), 팀즈(teams), 슬랙(slack), 스카이프(skype) 등이 있으며 본 논문에서 다룰 와이어도 이에 해당한다. 최근 많은 메신저가 해당 프레임워크를 사용하고 있고, 그에 관한 연구가 활발히 이루어지고 있다. [3][4][5][6]

2.1.2 LevelDB

구글(Google)에서 개발한 LevelDB는 Key-Value 데이터베이스 시스템이며, 키를 이용하여 매핑된 데이터를 읽어올 수 있다. 빠른 속도로 많은 데이터를 저장할 수 있으나 탐색이 느리다는 특징이 있다. LevelDB는 메모리 영역에 데이터를 존재하며, 이를 레벨 단위로 관리한다. 각 레벨의 저장 용량이 임계치를 넘게 되면 하위 레벨로 데이터를 정렬하여 저장한다. 이 과정에서 발생할 수 있는 위험 상황을 대비하기 위해 안전하게 복구할 수 있는 임시 저장영역이 존재하며, LevelDB에 반영되기 전에 log 파일에 먼저 데이터를 기록한다.

2.2 관련 연구

Electron Framework를 이용해 패키징 된 다양한 PC 환경의 메신저에 대해서 아티팩트를 분석하고 이를 이용해 사용자의 행위를 특징하는 관련 연구가 수행되고 있다. SuminShin은 Windows 환경에서 와이어 메신저를 대상으로 LevelDB의 log 파일을 분석하여, 대화 내역을 확인하였으며, 행위별로 생성되는 로컬 데이터의 분석결과를 제시하였다[3]. YoungChan Kim은 와이어 메신저에서 메시지를 주고받는 과정에서 사용하는 알고리즘, 키 분배 방식, 사용자 등록 과정, 대화를 위한 세션 형성과 같은 프로토콜을 분석하는 방법을 제시하였다[7]. JiSu Lee은 컴퓨터·스마트폰에서 디지털 성 착취물 시청을 입증하여 텔레그램, 와이어, 디스코드 등 모바일 메신저에서 디지털 성 착취물을 시청하였을 때 어떠한 데이터가 남는지 연구한 관련 연구가 존재한다[8].

Sumin Shin은 슬랙과 디스코드 애플리케이션을 대상으로 모바일과 PC 환경에서 메시지 수/발신, 공유한 파일, 메시지 채팅방, 사용자 계정 정보 등과 같은 주요 아티팩트의 저장 위치를 파악한 연구가 진행되었다[4].

앞서 언급한 바와 같이 다양한 보안 메신저와 관련 데이터를 분석하는 다양한 연구들이 존재하지만 본 연구와 비교하여 몇 가지 차이점이 존재한다. Sumin Shin은 분석의 범위가 LevelDB의 log 파일에 국한되었다. 그로 인해 사용자가 로그아웃할 경우 파일 log 파일 내 모든 데이터가 ldb 파일에 commit 되고, 데이터가 삭제되어 log 파일보다 많은 양의 데이터가 저장된 ldb 파일을 분석할 수 없다.

JiSu Lee는 cache 파일만을 분석하는 데 그쳐 메신저를 통해 주고받은 첨부파일을 저장, 시청하는 행위만을 파악할 뿐 실제 주고받은 데이터를 파악하려는 시도는 없었다[8].

이와 같이 LevelDB에 데이터를 저장하는 보안 메신저의 아티팩트를 종합적으로 분석하여 사용자 행위를 특정하려는 연구는 아직 진행된 바가 없다. 이에 본 연구에서는 로컬 PC 내 와이어 메신저의 모든 데이터를 분석하여 메신저상에서 사용자의 행위를 재연하는 방안을 연구하고, 이를 토대로 아티팩트 분석 도구를 개발하였다.

III. 와이어 메신저 아티팩트 분석

3.1 데이터 저장 경로 및 데이터 종류

운영체제별로 데이터가 저장된 경로는 다음과 같다.

〈Table 1〉 Wire Messenger Data storage path

OS	path
Windows(client version)	%USERPROFILE%\AppData\Roaming\Wire
macOS	/Users/{User_Name}/Library/Containers/com.wearezeta.zclient.mac/Data/Library/Application/Support/Wire
Web Browser	[Web Browser Path]\User Data\{ProfileName}\IndexedDB\https_app.wire.com_0.indexeddb.leveldb

3.2 채팅 데이터

와이어 메신저를 통해 주고받은 데이터를 확인하기 위해 Process Monitor[9]를 통해 데이터들을 확인했다. 데이터가 생성되는 주요 경로는 [표 1]과 같다. LevelDB 구조를 분석하기 위해 CCL Solutions Group에서 제공하는 ccl_chrome_indexeddb 오픈소스를 활용하여 분석을 진행했다[10]. 해당 코드를 통해 log 파일뿐 아니라, ldb 파일 분석이 가능하여, 사용자가 로그아웃하더라도 대화 내역 확인이 가능했다. 또한, 대상 기기에서 로그인한 시점 이후 주고받은 데이터들은 모두 확인할 수 있다. ldb 파일은 총 10개의 테이블로 구성되어 있으며, 본 논문에서 다룰 주요 데이터는 기기 정보를 담고 있는 clients, 사용자 정보를 담고 있는 conversations, 메신저 내에서 주고받은 내용을 담고 있는 events 테이블이 있다. 다음 내용은 [표 2], [표 3], [표 4]와 같다.

3.2.1 clients

〈Table 2〉 clients table data

Type	Description	Example
class	사용된 기기의 종류	desktop, phone
id	사용자 id	e1f30718fd378362
time	최초 로그인 시각(Unix time + Z: utc +0)	2021-09-25T 06:42:08.490Z
label	세부 환경	Windows 10 64 bit 10
location	사용자 위치(latitude, longitude)	(37.4897, 127.0639)
model	사용 환경	Wire Windows, Chrome

3.2.2 conversations

〈Table 3〉 conversations table data

Type	Description	Example
id	사용자 id	-
name	사용자 이름	-

3.2.3 events

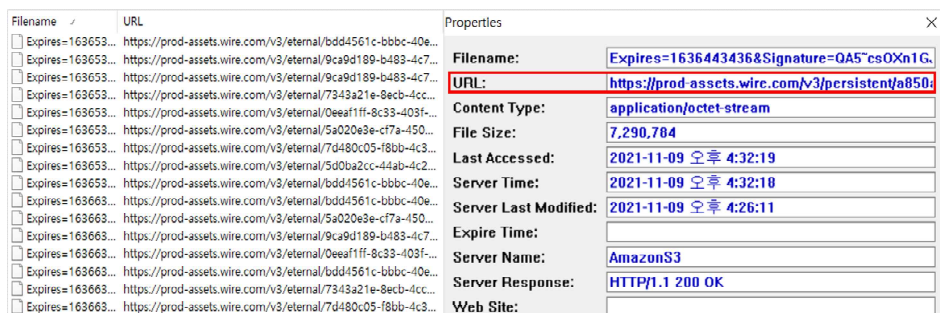
〈Table 4〉 events table data

Type	Description	Example
from	발신자 id	-
time	메시지 발신 시각	Unix time (utc +0)
type	데이터 유형	convconversation.message-add conversation.asset-add conversation.delete-everywhere conversation.voice-channel-(de)activate conversation.one2one-creation conversation.message-add conversation.asset-add conversation.delete-everywhere conversation.voice-channel-(de)activate conversation.one2one-creation
content	메시지 내역	-
ephemeral_expires	메시지 만료 시각(Unix time + Z: utc +0)	2021-09-14T 08:52:01.950Z
content_type	메시지 유형	text/plain, image/png, image/jpeg, application/pdf, application/haansofthwp, application/vnd.openxmlformats-officedocument.presentationml.presentation
content_length	첨부파일 크기	byte
name	첨부파일 이름	-
status	첨부파일 유무	NULL, uploaded
otr_key	첨부파일 복호화 키	32 bytes

3.3 첨부파일

LevelDB 등의 로컬 아티팩트에서 첨부파일과 관련된 정보를 찾고자 하였으나 로컬 아티팩트 내에는 해당 데이터가 존재하지 않았다. 파일을 주고받은 뒤, Process monitor 도구를 활용하여 와이어 메신저의 시스템 활동을 모니터링했다[9]. 사진 파일을 불러올 때, %USERPROFILE\AppData\Roaming\Wire\Cache 경로에 캐시 되는 것을 확인했다.

NirSoft 사에서 개발한 ChromeCacheView[11] 도구를 통해 cache 폴더에 저장된 파일 목록들을 확인하였다.



〈Figure 1〉 Wire cache folder using ChromeCacheView tool

[그림 1]과 같이 첨부파일을 주고받을 때 로드된 파일의 열람한 시간과 대응되는 'Expires=##(TimeStamp)&Signature=##' 형식의 cache 데이터 파일이 생성되었다. URL 주소가 CDN(Content Delivery Network)의 주소와 유사하여, 해당 cache 데이터가 와이어 서버를 통해 내려받은 첨부파일임을 알 수 있었다. 해당 cache 파일을 hex editor로 열람했을 때는 암호화가 되어 있어 유의미한 정보를 획득할 수는 없었다. 원본 파일 [그림 2]와 cache 파일 [그림 3]을 비교 분석했을 때, 파일의 크기의 차이가 16 bytes 이상 32 bytes 이내였음을 확인했다. 와이어 메시저의 경우 오픈소스이기 때문에 코드 분석을 통해 암호화 알고리즘을 분석했다.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	FF	D8	FF	E0	00	10	4A	44	49	46	00	01	01	00	00	48	00000000
00000010	00	48	00	00	FF	E1	00	58	45	78	69	66	00	00	4D	4D	.H..yá.XExif..MM
00000020	00	2A	00	00	00	08	00	02	01	12	00	03	00	00	00	01	.*.....
00000030	00	01	00	00	87	69	00	04	00	00	00	01	00	00	00	26#i.....&
00000040	00	00	00	00	00	03	A0	01	00	03	00	00	00	01	00	01
00000050	00	00	A0	02	00	04	00	00	00	01	00	00	05	48	A0	03H .
00000060	00	04	00	00	00	01	00	00	07	0B	00	00	00	00	FF	EDyi
00000070	00	38	50	68	6F	74	6F	73	68	6F	70	20	33	2E	30	00	.8Photoshop 3.0.

〈Figure 2〉 Original data

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	71	E5	94	71	07	52	DB	C8	EC	76	A7	35	84	C8	8C	04	qâ"q.RÜëiv\$S„EQ.
00000010	C5	DA	1B	2E	7B	F5	B8	3E	BE	B5	CD	90	5E	4E	F6	0C	ÄÜ..{ö,>*qi.°Nö.
00000020	2E	17	32	5A	45	3B	B3	97	CB	64	73	0A	8D	EA	4E	43	..2ZE;°-Èds..èNC
00000030	0F	49	D9	FE	EB	14	E9	35	61	8E	1F	C8	DE	95	C7	51	.IÜpë.é5až.Èp•ÇQ
00000040	5F	D5	C7	77	9F	31	39	97	E3	56	C4	F6	BF	87	8D	47	_ÖÇwŸ19-äVÄöç+.G
00000050	2D	B4	41	0E	3F	4C	D7	FB	A2	07	1B	93	9F	39	EF	06	-°A.?L×ûc...°Y9i.
00000060	0A	70	56	44	CD	D1	18	B5	61	4E	B9	85	24	3F	C3	FC	.pVDİÑ.µaN°...\$?Äü
00000070	FC	CB	8C	27	82	CE	B9	7C	CD	98	45	04	6E	6E	29	50	üEÖ',î: í°E.nn)P

〈Figure 3〉 Cache data

[그림 4]와 같이 코드 분석을 통해 cache 파일은 AES-CBC 알고리즘을 통해 암호화하고, cache 데이터의 최상위 16byte 가 초기화 벡터로 사용되는 것을 확인했다. 그리고 역공학을 진행한 결과 LevelDB events 테이블 안에 있는 otr_key 데이터가 암호화 키값으로 사용되었다. [그림 5]와 같이 otr_key 라는 문자열이 LevelDB의 키값으로 들어가 있고, 뒤에 32 bytes는 복호화 키 데이터가 들어있다. otr_key는 첨부 파일 별로 생성되며, 다른 기기에서 로드한 경우에도 동일한 otr_key 값을 가지고 있었다. 위의 과정을 통해 복호화를 진행한 결과 주고받은 첨부파일로 변환할 수 있었다.

```
const aesKey = await window.crypto.subtle.importKey('raw', keyBytes, 'AES-CBC', false, ['decrypt']);
const initializationVector = cipherText.slice(0, 16);
const assetCipherText = cipherText.slice(16);
return window.crypto.subtle.decrypt({iv: initializationVector, name: 'AES-CBC'}, aesKey, assetCipherText);

export const encryptAesAsset = async (plaintext: ArrayBuffer): Promise<EncryptedAsset> => {
  const initializationVector = generateRandomBytes(16);
  const rawKeyBytes = generateRandomBytes(32);
  const key = await window.crypto.subtle.importKey('raw', rawKeyBytes.buffer, 'AES-CBC', true, ['encrypt']);
  const cipherText = await window.crypto.subtle.encrypt(
    {iv: initializationVector.buffer, name: 'AES-CBC'},
    key,
    plaintext,
  );
};
```

〈Figure 4〉 Wire AssetCrypto.ts code

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00014870	34	61	30	32	64	33	34	22	07	6F	74	72	5F	6B	65	79	4a02d34".otr_key
00014880	42	20	60	4E	6A	CB	78	AF	41	9F	A9	8E	44	D6	3D	35	B `NjËx`Aÿ@ZDÖ=5
00014890	4A	2A	2B	81	72	6B	43	AB	B8	89	E1	DD	8C	51	B2	89	J*+.rkC«,%áŸEQ°%
000148A0	DB	3B	56	42	00	20	22	06	73	68	61	32	35	36	42	20	Û;VB. ".sha256B

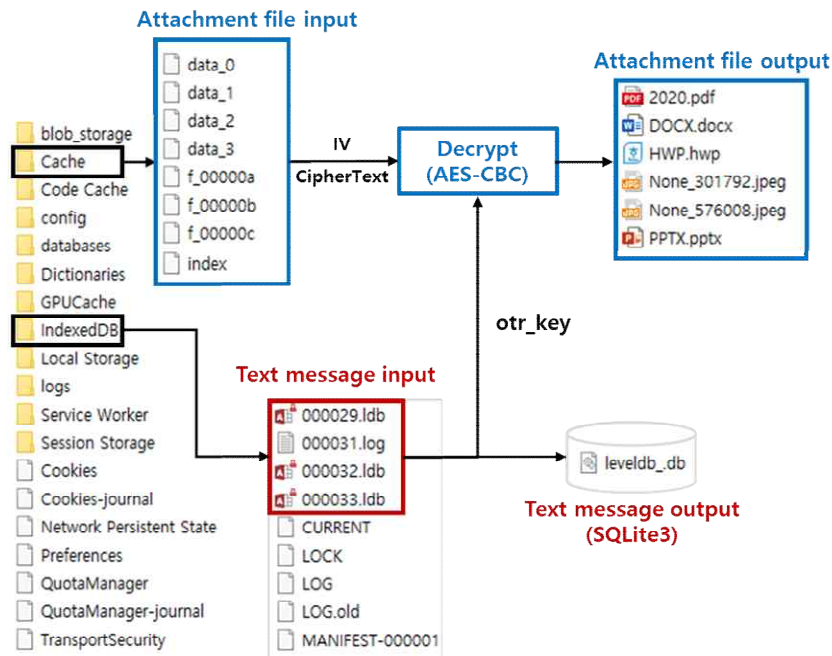
: key name length
 : key name
 : key data length
 : key data

〈Figure 5〉 Decryption key(otr_key) in ldb

IV. 분석 도구 개발

4.1 Credentials 활용한 데이터 획득

PC와 모바일 메신저의 데이터는 동기화되어 대화 내역이 동일하나, 새로운 기기에서 로그인을 한 경우 로그인한 시점 이후 생성된 데이터만을 확인할 수 있어 이전 데이터는 확인할 수 없다. 로컬 클라이언트 환경에서 로그인한 경우, 로그인 과정에서 Cookies 파일과 ldb 파일을 참조한다. 따라서 Cookies 파일과 ldb를 획득한 경우 해당 기기에서 로그인한 이력이 없더라도 정상적으로 계정을 활성화할 수 있다. 그리고 로그아웃을 한 뒤, 다른 계정에 로그인하더라도 이전에 생성된 데이터들은 log 파일과 ldb 파일에 남아있어 해당 데이터 확인이 가능하다[3]. 본 논문에서는 송, 수신된 메시지 내역과 첨부파일을 추출해 주는 도구를 개발하였으며, 도구의 전체 동작 과정은 [그림 6]과 같다.



〈Figure 6〉 Architecture of Automation Frameworks

4.2 수집 가능한 데이터 유형 및 분석 도구 개발

4.2.1 대화 내역, 관련 데이터 및 분석 도구 개발

송, 수신 메시지의 경우 ldb 파일과 log 파일에서 획득할 수 있으며, log 파일에서는 삭제된 메시지, 수정 이전의 메시지와 수정 이후 메시지 모두 획득이 가능하다. ldb의 경우 삭제된 데이터는 획득할 수 없지만 삭제된 흔적과 그 시각을 알 수 있다. 삭제된 데이터는 event 테이블에서 type 값이 conversation.delete-everywhere인 경우 메시지가 삭제되었음을 알 수 있고, deleted_time을 통해 삭제 시각을 알 수 있다[3].

4.1에서 언급한 내용 기반으로 타임라인 순으로 데이터를 추출하는 도구를 개발했다. 도구를 통해 사용자의 정보, 주고받은 메시지 그리고 메시지를 주고받은 사람들의 목록을 확인할 수 있다. 최종 결과물은 SQLite3 데이터베이스 형태로 출력된다. [그림 7]과 같이 client 테이블에서는 사용된 기기 종류와 최초 로그인 시각, 그리고 사용자의 위치 데이터 정보를 획득할 수 있다. [그림 8]은 conversations 테이블에서는 메시지를 주고받은 사람들의 목록, [그림 9]은 events 테이블에서 주고받은 메시지와 첨부파일에 대한 정보를 획득할 수 있다.

	device	id	Time	Label	Latitude	Longitude	Model
1	desktop	f005241c36b7ca97	2021-11-09T07:21:48.737Z	Windows 10 64-bit 10	37.6034	127.0065	Wire Windows
2	desktop	417b52663269bb10	2021-11-09T02:10:53.833Z	Windows 10 64-bit 10	37.6034	127.0065	Wire Windows
3	phone	6d7af92a1daa8115	2021-08-05T11:54:17.132Z	의 iPhone	37.5355	126.9766	iPhone12,8
4	desktop	bd4f9e4ad8d007dc	2021-10-11T09:09:19.465Z	Windows 10 64-bit 10	37.6034	127.0065	Wire Windows
5	desktop	ccb91da9994ba52a	2021-10-11T08:30:06.089Z	Windows 10 64-bit 10	37.6034	127.0065	Chrome
6	desktop	d2b8448c85aabe6f6	2021-09-14T07:48:43.816Z	Windows 10 64-bit 10	37.6034	127.0065	Wire Windows
7	phone	e6950dcf3a65375	2021-08-04T04:28:22.750Z	Galaxy S8	37.6023	127.0041	samsung SM-G950N

〈Figure 7〉 client table

	name_id	name__
1	004bb6a9-fc12-42f9-9480-2e847040490b	Sueun
2	2efb5cc1-e922-4d4e-874f-5913b18834ff	minzi
3	2f29cf69-21b5-4697-a1a5-ffee72c5a39	Haeun Kim
4	4e6fc1ee-359f-44d8-9cc9-d0d38950da49	minzi
5	5f3e61b6-f7ff-48f7-9181-d2258dc0b8ad	dfrc

〈Figure 8〉 conversations table

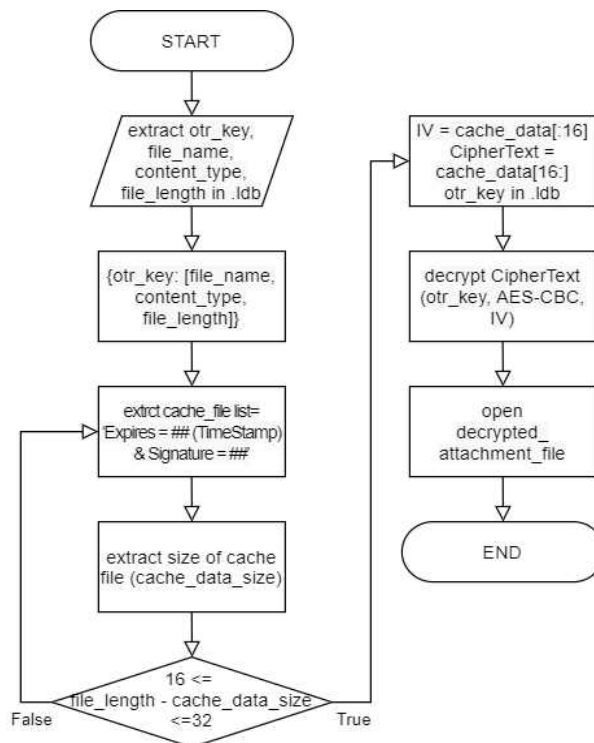
from_	create_time	type	content	expire_time	content_type	content_length	content_name	status	OTR_KEY
826820c8-af42-4b48...	2021-09-30T09:59:44.943Z	conversation.asset-add	NULL	NULL	image/png	222991	NULL	uploaded	44bdc69aede64810a7b33b623c1bb2c0795c6...
826820c8-af42-4b48...	2021-09-30T09:59:56.880Z	conversation.asset-add	NULL	NULL	image/peg	192956	NULL	uploaded	5024428367d573532c6e48581b5b9773ea3560...
826820c8-af42-4b48...	2021-09-30T10:07:48.508Z	conversation.asset-add	NULL	NULL	application/pdf	668352	Apple Watch ...	uploaded	fc278a1d644659b1a7d0b2a1f916ae722376e42...
826820c8-af42-4b48...	2021-09-30T10:08:32.248Z	conversation.asset-add	NULL	NULL	application/pdf	4908731	신입생안내사...	uploaded	f2cbc7e2ad09c4412c29b05e7900bc096540e3...
826820c8-af42-4b48...	2021-09-30T10:08:34.402Z	conversation.asset-add	NULL	NULL	application/pdf	67936	신입생등록안...	uploaded	c052846477ee47b15400122836ff782b4c8390d...
acc1aeal-7273-4b4f...	2021-10-01T05:41:52.875Z	conversation.message-add	W	NULL	NULL	NULL	NULL	NULL	NULL
acc1aeal-7273-4b4f...	2021-11-10T07:13:52.241Z	conversation.message-add	test11	NULL	NULL	NULL	NULL	NULL	NULL
acc1aeal-7273-4b4f...	2021-11-10T07:13:54.634Z	conversation.message-add	test22	NULL	NULL	NULL	NULL	NULL	NULL
acc1aeal-7273-4b4f...	2021-11-10T07:13:55.764Z	conversation.message-add	test33	NULL	NULL	NULL	NULL	NULL	NULL
acc1aeal-7273-4b4f...	2021-11-10T07:13:57.508Z	conversation.message-add	test 44	NULL	NULL	NULL	NULL	NULL	NULL
acc1aeal-7273-4b4f...	2021-11-10T07:13:59.220Z	conversation.message-add	test 55	NULL	NULL	NULL	NULL	NULL	NULL
acc1aeal-7273-4b4f...	2021-11-10T07:14:05.537Z	conversation.message-add	test 66	NULL	NULL	NULL	NULL	NULL	NULL
acc1aeal-7273-4b4f...	2021-11-10T07:13:54.634Z	conversation.message-add	test22	NULL	NULL	NULL	NULL	NULL	NULL
acc1aeal-7273-4b4f...	2021-11-10T07:13:55.764Z	conversation.message-add	test33	NULL	NULL	NULL	NULL	NULL	NULL
acc1aeal-7273-4b4f...	2021-11-10T07:13:57.508Z	conversation.message-add	test 44	NULL	NULL	NULL	NULL	NULL	NULL
acc1aeal-7273-4b4f...	2021-11-10T07:13:59.220Z	conversation.message-add	test 55	NULL	NULL	NULL	NULL	NULL	NULL
acc1aeal-7273-4b4f...	2021-11-10T07:14:05.537Z	conversation.message-add	test 66	NULL	NULL	NULL	NULL	NULL	NULL

〈Figure 9〉 events table

4.2.2 첨부파일 및 분석 도구 개발

와이어 메신저를 통해 기밀 유출 파일 혹은 음란물 등의 첨부파일을 주고받고, 해당 첨부파일을 내려받으면 cache 폴더에 데이터가 캐시 되는 것을 3.3에서 언급하였다. 특히 사진을 주고받는 경우, 별도의 파일을 내려받는 과정 없이 사진 파일의 전송만으로 캐시가 생성된다. 해당 cache 데이터는 암호화되어 별도의 복호화 과정이 필요하며, 이를 통해 첨부파일을 파일을 내려받은 사용자의 행위를 입증할 수 있다.

암호화된 파일을 복호화 하여 주고받은 첨부파일을 획득하기 위한 도구를 개발했다. [그림 10]은 첨부파일을 복호화 해주는 자동화 도구의 순서도이고, 해당 과정을 반복하여 cache 폴더에 존재하는 첨부파일의 데이터를 획득한다. ldb 파일에서 otr_key, 파일 이름, 파일 타입 그리고 파일 크기를 추출한다. 그다음 cache 폴더에 존재하는 'Expires = ## (TimeStamp) & Signature = ##' 형태의 암호화된 cache 파일들을 획득한다. 암호화 과정에서 초기화 벡터로 사용된 16 bytes와 16 bytes 단위로 패딩 하기 위해 추가로 생성된 데이터로 인해 암호화된 파일과 원본 파일의 크기 차이가 발생한다. 따라서 ldb 레코드에 기록된 첨부파일의 크기와 암호화된 파일 크기의 차이가 16 bytes 이상 32 bytes 이하인 파일을 매칭 시킨다. 그 이후에 [그림 11]에서 얻은 otr_key 값과 [그림 12]의 초기화 벡터 값을 이용하여 암호문을 복호화 시킨다. [그림 13]의 데이터가 위와 같은 과정을 반복하여 [그림 14]와 같이 성공적으로 복호화 되었음을 확인했다.





〈Figure 10〉 Analysis of Attachment file Frameworks

content_type	content_length	content_name	status	OTR_KEY **
application/vnd.openxmlformats-officedocument.presentationml.presentation	7290755	PPTX.pptx	uploaded	198ca15961a878944c6bc7b6c9bb74d7068b56...
application/vnd.openxmlformats-officedocument.presentationml.presentation	7290755	PPTX.pptx	uploaded	198ca15961a878944c6bc7b6c9bb74d7068b56...
application/vnd.openxmlformats-officedocument.presentationml.presentation	7290755	PPTX.pptx	uploaded	198ca15961a878944c6bc7b6c9bb74d7068b56...
application/vnd.openxmlformats-officedocument.wordprocessingml.document	25313	DOCX.docx	uploaded	cfaf085b10664d0dfadd24717443cdf1a29eb553...
application/vnd.openxmlformats-officedocument.wordprocessingml.document	25313	DOCX.docx	uploaded	cfaf085b10664d0dfadd24717443cdf1a29eb553...
image/jpeg	301792		uploaded	9d1297a1b9b86d031d390c1023c1399214dd67...
application/haansothwp	3472384	HWP.hwp	uploaded	6931e248b066074b1e987288170d4a45834fbac...
application/haansothwp	3472384	HWP.hwp	uploaded	6931e248b066074b1e987288170d4a45834fbac...
application/haansothwp	3472384	HWP.hwp	uploaded	6931e248b066074b1e987288170d4a45834fbac...

〈Figure 11〉 Attachment file acquired from ldb

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	5F	A0	A8	4A	A4	BB	C6	F1	23	92	F7	BF	F6	0A	D6	F8	
00000010	49	F0	71	EA	2B	83	BC	AB	46	10	2E	57	CD	A0	C6	B7	
00000020	52	D7	17	69	3D	C5	02	0C	AD	28	BD	8B	5D	B7	A1	42	
00000030	A6	6D	5B	50	B8	ED	BA	5F	52	77	CB	09	DB	E1	4F	DD	
00000040	74	ED	9E	F2	42	43	2B	80	7B	70	EB	20	77	88	48	87	
00000050	8A	8E	D4	20	D6	BD	50	AE	3E	9A	B0	FA	45	66	7B	C1	
00000060	B5	E7	FE	A4	16	58	18	1E	5E	B0	03	B9	E1	34	3E	92	
00000070	8B	3A	7D	4D	1A	0C	4C	02	4C	74	DC	60	91	C5	4D	7D	
00000080	65	91	E1	49	08	70	34	54	2F	D2	97	D5	67	24	B9	2B	

 : IV(Initial Vector)

 : Cipher Text

〈Figure 12〉 Encrypted initialization vector and Cipher Text

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	F8	0A	D4	0D	B7	A0	9E	8B	0B	93	55	7E	31	D1	51	B6	ø.Ô.·ž<."U~lÑQq
00000010	AE	CE	78	7C	49	EE	EA	B6	EC	26	57	FD	F9	AA	77	1B	0ix IiêQi&Wýù*w.
00000020	33	1B	06	15	3D	4B	80	15	82	4A	07	5A	6C	CA	0B	69	3...=K€,.,J.ZlÊ.i
00000030	5C	15	F6	9F	3A	CD	5E	71	42	4E	66	6E	03	9A	ED	91	\.öÿ:í^qBNfn.šì`
00000040	E7	06	9A	B1	F4	88	45	B7	97	3B	A8	09	05	64	83	D8	ç.šîô*E-;".dfø

〈Figure 13〉 Encrypted attachment file data

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	00	00	48	y0Yà..JFIF.....H
00000010	00	48	00	00	FF	E1	00	58	45	78	69	66	00	00	4D	4D	.H..Yá.XExif..MM
00000020	00	2A	00	00	00	08	00	02	01	12	00	03	00	00	00	01	.*.....
00000030	00	01	00	00	87	69	00	04	00	00	00	01	00	00	00	26#i.....&
00000040	00	00	00	00	00	03	A0	01	00	03	00	00	00	01	00	01

〈Figure 14〉 Decrypted attachment file data

V. 결 론

본 논문에서는 로컬 PC에 저장된 와이어 메신저 데이터를 분석하여 사용자의 전체 대화 내용을 재구성하였고 암호화된 첨부파일을 복호화 하는 연구를 수행하였다. 또한, 연구 내용을 기반으로 이를 자동화하는 도구를 개발하였다.

기존 연구의 분석 대상이었던 LevelDB의 log 파일은 사용자가 로그아웃할 경우 파일 내 모든 데이터가 ldb 파일에 commit 되고 삭제된다. 따라서 사용자의 행위에 따라 대화 내용이 분석되지 않을 가능성이 존재했다. 하지만 본 연구에서는 log 파일뿐 아니라 로그아웃 후 commit 된 데이터가 저장된 ldb 파일까지 분석의 범위를 확장했다. ldb 파일과 log 파일에 존재하는 데이터를 분석하여 로컬 PC에 저장된 전체 대화 내용과 암호화된 캐시 데이터의 복호화 키를 획득하였다. 획득한 복호화 키를 이용해 암호화된 cache 데이터를 복호화 하여 메신저상에서 주고받은 첨부파일 데이터를 획득하였다.

사진 같은 경우 스크롤 하는 행위만으로도 cache 폴더에 캐시가 되었지만, 사진을 제외한 멀티미디어 파일과 문서 파일의 경우 다운로드를 해야 캐시가 된다. 그러므로 다운로드를 하지 않으면 전송 여부만 알 수 있을 뿐 실제 데이터는 획득할 수 없다는 한계점이 존재한다. 그러나 이를 통해 해당 파일의 다운로드 여부를 파악할 수 있으므로 수사에 유의미한 자료가 될 수 있음을 나타낸다.

보안 메신저의 특성상, 범죄의 수단으로 활용될 수 있으므로 본 논문에서는 포렌식 수사 시 획득 가능한 데이터를 파악하는 데 중점을 두었다. 이를 통해 사용자 행위를 특정하고, 자동화 도구를 이용하여 데이터를 효율적으로 분석하는 데 도움을 줄 것으로 기대한다.

참 고 문 헌 (References)

- [1] kaspersky, "Messaging app security: Which are the best apps for privacy?", <https://www.kaspersky.com/resource-center/preemptive-safety/messaging-app-security>, Feb. 2022.
- [2] SecurityTech, "The most secure collaboration platform", <https://securitytech.org/secure-encrypted-messaging-app/wire/>, Feb. 2022.
- [3] Sumin Shin, et al. "Acquiring Credential and Analyzing Artifacts of Wire Messenger on Window." Journal of The Korea Institute of Information Security & Cryptology 31.1, 2021.
- [4] Sumin Shin, et al. "Artifacts Analysis of Slack and Discord Messenger in Digital Forensic." Journal of Digital Contents Society 21.4, pp.799-809. 2020.
- [5] Iqbal, Farkhund, Michał Motyliński, and Áine MacDermott. "Discord Server Forensics: Analysis and Extraction of Digital Evidence." 2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS). IEEE, 2021.
- [6] Younghoon Kim, and Taekyoung Kwon. "On Artifact Analysis for User Behaviors in Collaboration Tools - Using Differential Forensics for Distinct Operating Environments.." Journal of The Korea Institute of Information Security and Cryptology (JKIISC), 31.3, pp. 353-363, 2021.
- [7] YoungChan Kim. "Wire Messenger Application Protocol Analysis." 한국정보처리학회 학술대회논문집 27.2, pp. 387-389, 2020.
- [8] JiSu Lee, YeonJu Lee, and GiBum Kim. "A Study on the Digital Forensic of Viewing Digital Sexual Exploitation Material in Mobile Messenger." The Journal of Police Science 20.4, pp. 227-253, 2020.
- [9] <https://docs.microsoft.com/en-us/sysinternals/downloads/procmon>, June. 2022.
- [10] https://github.com/cclgrouppltd/ccl_chrome_indexeddb, Nov. 2021
- [11] https://www.nirsoft.net/utils/chrome_cache_view.html, Nov. 2021

저 자 소 개



조 민 지 (Minji Cho)
준회원

2020년 2월 : 서울여자대학교 정보보호학과 졸업
2022년 3월 ~ 현재 : 고려대학교 정보보호대학원 석사과정
관심분야 : 디지털포렌식, 사이버 범죄 대응



정 병 찬 (Byeongchan Jeong)
준회원

2017년 2월 : 고려대학교 정보보호대학원 공학석사
2021년 2월 ~ 현재 : 고려대학교 정보보호대학원 박사수료
관심분야 : 디지털 포렌식, 모바일 포렌식, 침해사고 조사



박 정 흠 (Jungheum Park)
준회원

2014년 2월 : 고려대학교 정보보호대학원 공학박사
2015년 1월 ~ 2019년 2월 : 미국 국립표준기술연구원(NIST) 방문연구원
2019년 3월 ~ 2021년 8월 : 고려대학교 정보보호연구원 연구교수
2021년 9월 ~ 현재 : 고려대학교 정보보호대학원 조교수
관심 분야 : 디지털포렌식, 사이버범죄대응, 침해사고대응



이 상 진 (Sangjin Lee)
평생회원

1989년 10월 ~ 1999년 2월 : 한국전자통신연구원 선임연구원
1999년 3월 ~ 2001년 8월 : 고려대학교 자연과학대학 조교수
2001년 9월 ~ 현재 : 고려대학교 정보보호대학원 교수
2017년 3월 ~ 현재 : 고려대학교 정보보호대학원 원장
관심 분야 : 대칭 키 암호, 정보은닉 이론, 디지털포렌식