Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.29, NO.3, Jun. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.3.541

인스턴트 메신저 말랑말랑 톡카페 애플리케이션 데이터베이스 복호화 방안 및 분석*

김 기 윤,[1†] 이 종 혁,[2] 신 수 민,[2] 김 종 성[1,2‡]
[1]국민대학교 금융정보보안학과, [2]국민대학교 정보보안암호수학과

Study on The Decryption Method and Analysis of MalangMalang Talkcafe Application Database*

Giyoon Kim,[1†] Jonghyeok Lee,[2] Sumin Shin,[2] Jongsung Kim[1,2‡]
[1]Dept. of Financial Information Security, Kookmin University
[2]Dept. of Information Security, Cryptology and Mathematics, Kookmin University

요 약

개인 정보 유출 사례가 빈번해짐에 따라 개인정보보호에 대한 관심이 증가하고 있다. 이러한 이유로 개인 정보를 수집해야 하는 대다수 애플리케이션은 민감한 정보를 암호화하여 저장하는 방식을 취하고 있다. 특히 사용자의 흔적이 가장 많이 기록되는 인스턴트 메신저는 대화 내용 등을 암호화하지 않는 경우를 찾는 것이 더 어렵다. 하지만 개인 정보 암호화는 디지털 포렌식 수사 관점에서 안티 포렌식에 해당하므로, 이를 증거로 활용하기 위해서 메신저 애플리케이션 데이터 복호화 연구는 선행되어야 한다. 본 논문에서는 인스턴트 메신저인 말랑말랑 톡카페 애플리케이션의 데이터베이스 암호화 과정을 분석하여 암호화 키 생성 과정에 존재하는 취약점을 밝혀낸다. 이는 암호화된 데이터베이스를 복호화해내는데 치명적으로 작용하여, 이를 토대로 실제 사용 가능한 복호화 방안을 제시한다. 또한, 복호화를 통해 얻은 데이터베이스에서 포렌식적으로 의미 있는 데이터를 분류한다.

ABSTRACT

As leakage cases of personal information increase, the concern of personal information protection is also increasing. As a result, most applications encrypt and store sensitive information such as personal information. Especially, in case of instant messengers, it is more difficult to find database where is not encrypted and stored. However, this kind of database encryption acts as anti-forensic from the point of view of digital forensic investigation. In this paper, we analyze database encryption process of MalangMalang Talkcafe application which is one of instant messenger. Based on our analysis, we propose a decryption method and explain the meaningful information collected in the database.

Keywords: Instant Messenger, SQLCipher, Decrypt, Database

Received(02. 11. 2019), Modified(04. 16. 2019), Accepted(04. 27. 2019)

* 본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

* 본 논문은 2018년도 한국정보보호학회 동계학술대회에 발표한 우수논문을 개선 및 확장한 것임

† 주저자, gi0412@kookmin.ac.kr
‡ 교신저자, jskim@kookmin.ac.kr(Corresponding author)

I. 서 론

IT 시대가 가속화됨에 따라 사람들은 다양한 디지털 기기와 연관되어 있다. 미국 리서치기관인 퓨(Pew)리서치에 의하면 우리나라는 인터넷과 스마트폰 보급률이 96% 이상으로 전 세계 1위를 기록하고 있다[1]. 이처럼 우리나라 사람들은 다양한 애플리케이션 사용 환경에 손쉽게 노출될 수 있음을 뜻하며, 다양하고 지속적인 데이터가 수집되고 있음을 나타낸다. 특히 인스턴트 메신저[1]는 사용자의 행위와 직접적으로 관련된 데이터가 다수 기록되어 있는 애플리케이션이다. 따라서 디지털 포렌식 수사 시 필수적인 조사대상이지만, 대부분 개인 정보를 암호화하여 보관하기 때문에 분석 시에 데이터 획득이 불가능한 경우가 발생한다. 이를 방지하기 위해서는 인스턴트 메신저의 데이터베이스 복호화 연구가 선행되어야 한다.

기존에 발표된 인스턴트 메신저 관련 연구 논문들은 주로 메신저 데이터 분석 및 데이터 복호화 내용이며, 본 논문에서 분석한 인스턴트 메신저 말랑말랑 톡카페와 유사한 기존 연구 논문 결과들은 다음과 같다. Sudozai 등은 무료 문자 및 전화 기능을 제공하는 IMO 애플리케이션을 분석하여 디지털 포렌식 관점에서 데이터 사용방법을 제시했다[2]. Anglano 등은 SQL Cipher 모듈을 사용하여 데이터베이스를 암호화 한 Chatsecure을 분석하였으며 초기 입력값을 기반으로 데이터를 암호화함을 밝혔다[3]. 그리고, Songyang Wu 등은 중국에서 많이 사용하는 인스턴트 메신저인 WeChat에 데이터베이스 복호화 방안과 아티팩트를 분석했다[4]. 분석된 WeChat 메신저는 스마트폰의 고유정보와 사용자의 고유정보를 이용하여 패스프레이즈(Passphrase)[2]를 생성하고, SQL Cipher 모듈을 사용하여 데이터베이스를 전체 암호화했다.

본 논문에서는 데이터베이스를 암호화하여 보관하는 인스턴트 메신저인 말랑말랑 톡카페 애플리케이션에 대한 분석 연구를 수행한다. 2장에서는 말랑말랑 톡카페 애플리케이션의 데이터 암호화 과정에 대한 분석과 복호화에 필요한 요소들을 설명한다. 3장에서는 실제 복호화된 데이터베이스에 저장되는 데이터의 의미를 설명하며, 마지막 4장에서 본 논문에 대한 결론으로 마무리한다.

1) 네트워크를 이용해 두 명 이상의 즉각적인(실시간) 텍스트 통신에 이용되는 애플리케이션이다.

2) 컴퓨터 시스템, 프로그램 또는 데이터에 대한 접근 권한을 제어하기 위한 단어들의 나열, 비밀번호(password)와 유사하다.

II. 말랑말랑 톡카페 애플리케이션 데이터베이스에 대한 암호화 과정 분석 및 복호화

본 장에서는 말랑말랑 톡카페 애플리케이션에 대한 정적분석을 통해 데이터 암호화 시 사용하는 암호 모듈을 밝혀내며, 암호키 생성 방안 및 복구 방법을 제시한다. 분석에는 JEB Decompiler를 사용하였다. JEB Decompiler는 기계어(바이트코드)로 작성되어 있는 애플리케이션을 java코드로 디컴파일 해주는 프로그램이다. 이를 통해 말랑말랑 톡카페 애플리케이션을 java코드로 변환하여 코드 분석을 진행하였다.

애플리케이션이 관리하는 데이터베이스는 'user.db'와 'talk.db'이며, 두 파일 모두 전체암호화가 되어 있다. 데이터베이스 암호화에는 외부라이브러리인 SQLCipher를 사용하는 것으로 확인하였다[Fig. 1].

SQLCipher모듈을 사용한 경우 데이터베이스에 접근하기 전, 패스프레이즈를 입력하여 데이터베이스를 사용 가능하게 변경한다. 본 장에서는 SQLCipher의 동작 과정과 소스코드 분석을 통해 밝혀낸 패스프레이즈 생성 과정에 대해 설명한다.

```
public static void loadLibs(Context arg2, File arg3) {
    Class v1 = SQLiteDatabase.class;
    __monitor_enter(v1);
    try {
        System.loadLibrary("sqlcipher");
    }
    catch(Throwable v0) {
        __monitor_exit(v1);
        throw v0;
    }

    __monitor_exit(v1);
}
```

Fig. 1. Loading SQLCipher Library in Application

2.1 SQLCipher

SQLCipher는 SQLite3 데이터베이스를 암호화하는 모듈로써 패스프레이즈를 입력받아 키를 생성하여 데이터베이스를 암호화한다[5]. 데이터 암호화는 AES-256-CBC 모드를 이용하며, SQLite3 데이터베이스 페이지의 크기가 128비트의 배수 비트이므로 패딩은 사용하지 않는다. 암호화키는 PBKDF2

(Password-Based Key Derivation Function 2)를 사용해서 생성하며, 생성방법은 수식(1)과 같다.

$$AESKey = PBKDF2(HMAC-SHA1, Passphrase, Salt, 64000, 256) \quad (1)$$

SQLCipher는 데이터베이스 암호화 시 이에 대한 HMAC 값을 생성하며, 이 값을 암호화된 데이터 뒷부분에 덧붙인다. HMAC 값은 복호화 시 올바른 키를 사용했는지 검증 용도로 사용된다. HMAC에 사용되는 키 생성 방법은 수식(2)와 같다.

$$HMACKey = PBKDF2(HMAC-SHA1, AESKey, Database, 2, 256) \quad (2)$$

SQLCipher로 암호화된 데이터베이스 파일의 구조는 Table 1과 같다. 이때 사용되는 salt와 IV (Initial Vector)는 각각 128비트의 난수이며, 평문형태로 저장되어있다.

Fig. 2는 SQLCipher 모듈에 의해 암호화된 데이터베이스의 복호화 과정이다[5].

복호화의 상세과정은 다음과 같다.

1. 수식(1)을 사용하여 수식(2)의 HMAC key 유도
2. 유도된 HMAC key를 사용하여 암호화된 데이터베이스의 HMAC-SHA1값 계산
3. 계산된 HMAC 값과 저장되어있는 HMAC 값 비교 후, 일치 시 옳은 패스프레이즈 검증 성공
4. 수식(1)에서 유도된 키를 토대로 데이터베이스 복호화

Table 1. Structure of Encrypted Database

Salt	Encrypted Database	HMAC Value	IV

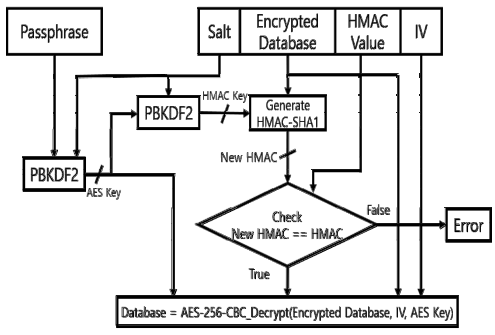

Fig. 2. Decryption Method of Encrypted Database

이때 SQLCipher에 사용되는 패스프레이즈는 데이터베이스마다 다르며 'talk.db'의 경우 특정 위치에 패스프레이즈를 저장하고 있으나, 'user.db'는 필요에 따라 매번 값을 유도하여 사용한다.

2.2 패스프레이즈 생성 과정

'user.db'를 암/복호화하기 위해 사용되는 패스프레이즈의 생성 과정은 다음과 같다[Fig. 3].

Fig. 3의 Android ID는 64비트의 난수값으로 Android Froyo 버전부터 안드로이드가 설치된 기기의 사용자 식별을 목적으로 생성된다. Serial Number는 기기의 일련번호를 의미한다. 이 두 가지 정보를 연접하여 SHA-256 해시 알고리즘으로 해싱한 값이 'user.db' 전용 패스프레이즈가 된다.

'talk.db'의 경우 애플리케이션 사용자의 고유 인덱스값인 useridx 값을 패스프레이즈로 사용하고 있으며, 해당 정보는 'user.db'내에 저장되어있다. 따라서 일반적인 방법으로 'talk.db'를 복호화 하기 위해서는 'user.db'의 복호화가 선행돼야 한다. 하지만, useridx 값은 0부터 시작하여 1씩 증가하기 때문에 전수조사를 통해 복구할 수 있다. 또는 다른 계정으로 친구추가를 진행한 뒤, 다른 계정의 'user.db'를 복호화함으로써 useridx값의 획득이 가능하다. 단, 'user.db'의 경우 Android ID가 64비트이고 PBKDF2의 반복횟수가 64,000회이므로 실질적으로 전수조사는 불가능하다.

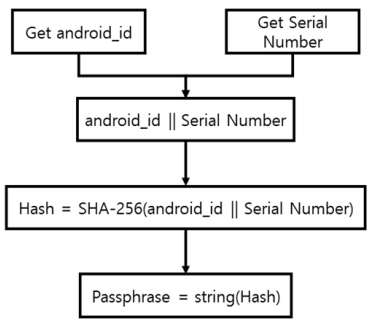

Fig. 3. Generating Passphrase of user.db

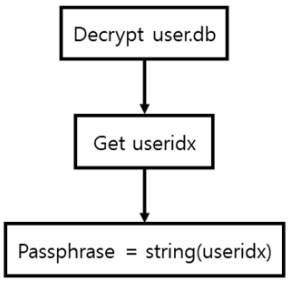

Fig. 4. Getting Passphrase of talk.db

2.3 Android ID 재현 가능성 및 획득 방법

Android ID는 Android Nougat 이하의 버전에서 기기 첫 부팅 시 혹은 공장 초기화 시 생성되는 값을 설치된 모든 애플리케이션이 공통으로 사용할 수 있었지만, Android Oreo 이상에서는 애플리케이션마다 생성해서 사용하도록 정책이 변경되었다[6]. 이때 Android ID는 SecureRandom()을 사용하여 생성되는데, 해당 함수는 안전성이 증명된 /dev/random 알고리즘을 이용하기 때문에 Android ID 재현은 불가능하다. 따라서 Android ID를 획득하기 위해서는 기기확보가 필수적이다.

Android ID 획득 방법은 다음과 같다. 애플리케이션별로 생성된 Android ID는 기기의 content내에 저장이 되어있으므로 ADB (Android Debug Bridge)[3]를 이용하거나 Setting.Secure.getString() 메소드를 사용하여 Android ID를 획득하는 애플리케이션을 제작하여 실행하는 것으로 가능하다. ADB를 사용하는 경우 명령어는 다음과 같다.

```
adb shell content query --uri content://settings/secure --where "name=\'android_id\'"
```

Fig. 5는 ADB를 통해 실제 Android ID를 획득하는 과정이다.

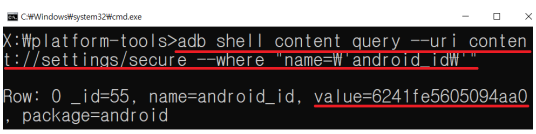

Fig. 5. Getting Android ID from ADB

3) 에뮬레이터 인스턴스나 연결된 Android 기기와 통신할 수 있는 다목적 명령줄 도구이다.

2.4 데이터베이스 복호화 결과

Fig. 6은 ADB를 통하여 Android ID 값을 획득한 후, 스마트폰의 시리얼 넘버와 연접한 값을 해싱하여 생성해낸 키를 사용하여 실제 암호화된 데이터베이스를 복호화한 결과이다.

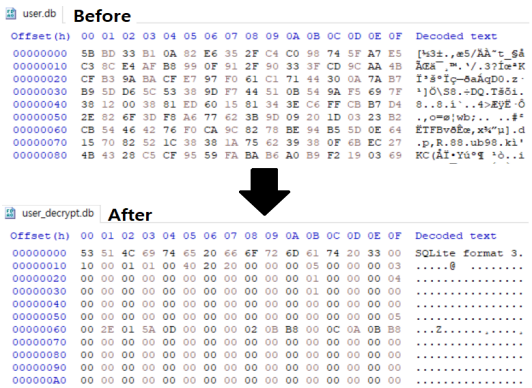

Fig. 6. Decrypt Result

III. 데이터베이스 분석 결과

본 장은 복호화된 'user.db'와 'talk.db'의 분석 결과를 설명한다. 두 데이터베이스에는 각각 사용자의 ID 및 프로필에 대한 정보와 채팅에 대한 정보를 담고 있다. 따라서 데이터 획득이 가능하다면 정황증거 또는 직접증거로써 활용할 수 있다.

'user.db'는 사용자의 개인 정보를 저장한 데이터베이스로 사용자의 프로필, 이름, 특정 유저 전용 이름, 친구 정보 등 다양한 정보를 저장하고 있다. 'talk.db'는 사용자들 간의 대화 내용을 저장하고, 메시지 수신/발신 시간, 메시지의 발신인 등을 포함하고 있다. 각각의 데이터베이스의 주요 데이터는 다음과 같다(Table 2,3).

Table 2. The Data of user.db

Table Name	Name of the Data	Data
user	useridx	The index of user
	userID	The ID of user
	username	The name of user

	profileMessage	The status message of user
	idType	The type of ID
	friends	The number of friends
	updateTime	The time of latest access
	secretProfileFlag	The existence of the profile which only friends can see who are designated by the user(T/F)
	secretUserName	The name of the user which only friends can see who are designated by the user

Table 3. The Data of talk.db

Table Name	Name of the Data	Data
chats	roomidx	The index of the chatroom
	type	The type of message(text, image, etc.)
	text	The message
	sender	The sender of the message
	time	The time when the message is received
rooms	roomidx	The index of the chatroom
	name	The name of the chatroom
	roomsLastIdx	The chatroom which is the user chatted lastly
	lastChatText	The message which is sended lastly
	lastChatType	The type of message(text, image, etc.)
	lastChatTime	The time when the user chatted lastly
	lastChatSender	The Sender who sended a message lastly
	mqttHost	The address of the host of the mqtt
	mqttPort	The port number of the mqtt (ex. 1883)

두 데이터베이스는 SQLite3의 Secure Delete[4] 옵션을 사용하기에 데이터가 삭제된 경우 복구가 어렵다. Journal mode[5] 역시 Memory mode[6]를 사용하기 때문에 데이터 삭제 후 장시간이 지났다면 메모리 분석을 통한 데이터 부분 복구 역시 불가능하다. 그러나 사용자의 기기에서 메시지를 삭제하여도 서버에 저장되는 데이터는 삭제되지 않음을 확인하였다. 따라서 서버에 데이터를 보관하고 있는 동안에는 애플리케이션 재설치 및 재접속을 통하여 부분적으로 데이터 복구가 가능하다.

IV. 결 론

애플리케이션 개발자들은 개인 정보를 안전하게 보관하기 위하여 다양한 암호화 기술을 적용한다. 하지만 이러한 기술들은 디지털 포렌식 수사 관점에서 안티 포렌식으로 행위이며, 일부 범죄자들은 이를 악용하기도 한다. 따라서 본 논문에서는 인스턴트 메신저 중 하나인 말랑말랑 톡카페 애플리케이션의 데이터베이스에 대한 복호화 방안 연구를 하였다. 정적분석을 통하여 복호화에 필요한 패스프레이즈 생성 방안을 연구한 결과 'user.db' 의 패스프레이즈 생성 과정에는 충분한 PBKDF2 반복횟수와 seed를 사용하였다. 이에 따라 'user.db'는 특정 정보 없이 전

4) 데이터 영역을 삭제 후 0으로 덮어씌우는 기법이다.
5) 원본 데이터를 별도의 파일에 저장하여, DB 혹은 서버에 오류 시에도 데이터 유실 및 파일 손상을 방지하고 문제 발생 시 원상복구를 하기 위한 방식으로 사용된다.
6) SQLite의 저널링 파일을 메모리에 유지하는 기법이다.

수조사로 복호화를 하기에는 어려움이 있었고, 'talk.db'는 짧은 길이의 패스프레이즈 사용, 손쉽게 획득할 수 있는 패스프레이즈를 사용하였기에 특정 정보 없이도 데이터 복호화가 가능했다. 이들을 기반으로 두 데이터베이스를 복호화하였으며 각각의 아티팩트에 대한 분석을 통해 디지털 포렌식 관점에서 의미있는 데이터들을 정리하였다. 따라서 본 논문에서 제시한 데이터 획득 방법과 분석된 아티팩트를 통해 효율적인 증거수집이 가능할 것으로 기대한다.

References

[1] Pew Research Center, "Mobile", http://www.pewresearch.org/topic/mobile/

[2] ANGLANO, Cosimo; CANONICO, Massimo; GUAZZONE, Marco. Forensic analysis of the chatsecure instant messaging application on android smartphones. Digital investigation, vol. 19, pp 4~59, Mar. 2016.

[3] SUDOZAI, M. A. K., et al. Forensics study of IMO call and chat app. Digital Investigation, vol. 25, pp. 5~23, Dec. 2018.

[4] WU, Songyang, et al. Forensic analysis of WeChat on Android smartphones. Digital investigation, vol. 21, pp. 3~10, Jun. 2017.

[5] Zentic "SQLCipher API", https://www.zetetic.net/sqlcipher/sqlcipher-api/#key

[6] Android Developers, "Provider Android_id", https://developer.android.com/reference/android/provider/Settings.Secure?hl=ko#ANDROID_ID

〈저자소개〉

김 기 윤 (Giyoon Kim) 학생회원
2019년 2월: 국민대학교 정보보안암호수학과 졸업
2019년 3월~현재: 국민대학교 금융정보보안학과 석사과정
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식

이 종 혁 (Jonghyeok Lee) 학생회원
2017년 3월~현재: 국민대학교 정보보안암호수학과 재학중
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식

신 수 민 (Sumin shin) 학생회원
2016년 3월~현재: 국민대학교 정보보안암호수학과 재학중
〈관심분야〉 정보보호, 디지털 포렌식

김 종 성 (Jongsung Kim) 종신회원
2000년 8월/2002년 8월: 고려대학교 수학 전공 학사/이학석사
2006년 11월: K.U.Leuven. ESAT/SCD-COSIC 정보보호 전공 공학박사
2007년 2월: 고려대학교 정보보호대학원 공학박사
2007년 3월~2009년 8월: 고려대학교 정보보호기술연구센터 연구교수
2009년 9월~2013년 2월: 경남대학교 e-비즈니스학과 조교수
2013년 3월~2017년 2월: 국민대학교 수학과 부교수
2014년 3월~현재: 국민대학교 일반대학원 금융정보보안학과 부교수
2017년 3월~현재: 국민대학교 정보보안암호수학과 부교수
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식.