정 보 보 호 학 회 지
제 32 권 제6호, 2022. 12

안드로이드 데이터 암호화 앱 동향 및 분석

이 성 원*, 김 청 운**, 김 도 현***

요 약

현대 사회에서 스마트폰이 일상생활에 밀접하게 사용됨에 따라 스마트폰 내부에는 사용자가 사용한 다양한 앱 데이터가 저장되고 있고 이 중에는 민감한 개인정보도 포함된다. 따라서 스마트폰을 분실하거나 스마트폰이 악성앱에 공격당하는 경우 개인정보가 유출될 수 있기 때문에, 이를 대응하기 위해 스마트폰 내부 데이터를 암호화 저장하는 다양한 앱들이 출시되고 있다. 우리는 총 12개의 데이터 암호화 앱들에 대한 기존 연구 결과를 통해 데이터 암호화 앱에 대한 동향을 살펴보고, 안드로이드 앱 마켓에서 전 세계적으로 10,000,000회 이상 다운로드되어 널리 사용되고 있는 5개의 추가적인 데이터 암호화 앱을 분석했다. 그 중 특히 LOCKit 앱을 자세히 분석하여 암호 알고리즘에 대한 취약점을 밝혀내 데이터 복호화 방법과 취약점 보완을 위한 방안을 제시했다.

Ⅰ. 서 론

스마트폰 분실 및 악성앱으로 인한 스마트폰 내부 데이터의 유출로 인해 개인정보가 침해될 수 있다. 이에 따라 사용자가 특정 데이터를 안전하게 암호화하여 보관할 수 있게 해주는 앱들이 출시되고 있다. 이러한 앱들은 일반적으로 사용자가 선택한 파일뿐만 아니라 그 내역이 저장된 로그인 DB파일도 암호화 한다.

본 고는 이러한 암호 기술이 적용된 앱들의 암호화 방법에 대한 동향을 살펴보고, 역공학을 통해 실제로 이러한 앱이 사용자의 민감한 개인정보를 잘 보호하고 있는지에 대한 취약점을 분석한다. 2장은 암호 기술을 사용한 대상으로 한 관련 연구들을 소개하고 각 앱들이 사용한 암호화 알고리즘과 키를 관리하는 방식에 대해 정리했다. 3장은 10,000,000회 이상 다운로드되어 널리 사용되고 있지만 아직 적용된 암호 방법에 대해 연구되지 않는 앱을 분석했고, 각 앱에 적용된 암호화 알고리즘 종류와 복호화 방안을 제시한다. 4장은 3절에서 분석한 앱 중 LOCKit 앱에 적용된 암호 기능을 자세히 분석하고 그 취약점을 분석한다. 5장에서는 본 고에서 분석한 내용을 정리하고 암호 기술을 사용한 앱들의 취약점 보완을 위한 방안을 제시한다.

Ⅱ. 관련 연구

암호 기술을 사용하는 다양한 앱들을 역공학하여 해당 앱들이 사용하는 암호화 알고리즘과 키 관리 방식을 분석한 연구들이 있었다. 대부분의 암호 기술이 적용된 앱 분석 연구는 데이터 은닉 기능인 Vault 기능이 있는 앱들을 분석하여 암호화 알고리즘과 복호화 방안을 제시했다. 박상호 외 3명은 암호 기술이 사용된 앱 Encypt File Free, CJ one Card, Gallery lock lite, SSE_File의 암호화 알고리즘과 취약점을 연구했다[1]. 이세훈 외 4명은 KakaoTalke 내부 대화 내역 DB에 저장하는 값과 네트워크로 전송되는 정보에 암호 기술이 사용됨과 복호화가 가능함을 확인했다[3]. Xiaolu Zhang는 구글 플레이에서 다운로드한 18개의 Vault 앱을 역공학하여 암호화 알고리즘을 밝혀냈다[4]. 홍표길은 Vault 앱인 AppLock의 암호화 알고리즘과 키 관리 방식 취약점을 분석했다[5]. 이정우 외 2명은 계산기로 위장한 Vault 앱 Calculator - photo vault을 정적 분석하여 암호화 방식과 고정된 salt 값을 확인했다[6]. 김성진은 은 암호 기술이 적용된 앱들의 취약점을 분석하여 각 앱이 사용한 알고리즘과 유출될 수 있는 개인정보를 분석했다[7]. 김정섭은 랜섬웨어에서 사용하는 암호화 방식과 키 관리 방안을 확

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2021R1F1A1061926).
* 부산가톨릭대학교 컴퓨터공학과 (대학원생, glick925@gmail.com)
** 부산가톨릭대학교 컴퓨터공학과 (학부생, kim789561@naver.com)
*** 부산가톨릭대학교 컴퓨터정보공학과 (교수, dohyun@cup.ac.kr)

〔표 1〕 데이터 암호화 앱의 암호 기술 동향

앱이름 (버전) [참고문헌번호]	패키지이름	접근 제어	파일 암호화	암호화 알고리즘	암호키	
					관리방식	salt 사용유무
Encrypt File Free(1.0.6) [1]	com.acr.encryptfilefree&hl=ko&gl=US	O	O	내부의 독자적인 알고리즘 사용	Login 클래스에서 MD5되어databases/optionscrypt 파일과 암호화된 파일의첫 부분에 존재	고정값 사용
CJ One Card(2.7.0) [1]	kr.co.ivlog.mobile.app.cjonecard&hl=ko&gl=US	O	O	AES	SetCardPW 클래스에서 사용자로부터 입력받은 패스워드와 salt값을 AES 클래스의 메쏘드를 호출하여 암호화해 존재	''cjonemobilecard1'' 고정값 사용
사진/동영상 숨기기, 잠금 - Gallery lock lite (4.6) [1]	com.morrision.gallerylocklite	O	X	3-DES	shared_prefs/com.morrison.gallerylock - lite_preferneces. xml 파일에 앱 실행 비밀 번호가 암호화 되지 않고 존재, 난독화	자바 API를 통해 랜덤 생성
SSE_File/Text Encryption & Password Vault (1.4.7) [1]	com.paranoiaworks.unicus	O	O	AES ,RC6, Serpent, Blowfish, Twofish, GOST+	사용자로부터 입력받아 복호화 패스워드를 저장하지 않고 사용자에게 물어보는 방식. 정적분석으로 복호화 불가능	고정값 사용
LockMyPix(4.2.6-b) [2]	com.fourchars.lmpfree	O	O	AES128	PIN 번호 및 패스워드는 암호화되어 '.ini.keyfile.ctr' 파일로 저장 사용자가 설정한 복구 이메일을 암호화해 '.ini.keyfile3.cmp' 파일로 존재	고정값 사용
KakaoTalk(3.6.5) [3]	com.kakao.talk	O	O	PBEwithSHA1andAES256	PBE (Password-Based Encryption)을 통해 암호키 생성 및 데이터 암호화를 수행 /data/data/com.kakao.yellowid/databases에 존재	"kevin" 문자열과 plus_friend.db 내 chat_log 테이블에 존재하는 profile_id 값을 연접한 후 상위 16 바이트를 사용
Valut - 사진 숨기기, 앱 잠금, 클라우드 백업 (6.4.22.22) [4]	com.netqin.ps	O	O	Base64		
Keepsafe 개인적인 사진 비디오 숨기는 잠금 앨범 (7.3.1) [4]	com.kii.safe	O	O	IV(Initialization Vector)	공유 prefs 폴더에 XML파일 com.ki.safe_preferences.xml의 태그 master-password 값에 존재	X
사진 및 비디오 숨기기 - HIde it Pro (5.4) [4]	com.hideitpro	O	X			
Gallery Vault - 사진 및 비디오 숨기기 (2.9.1) [4]	com.thinkyeah.galleryvault	O	X			
AppLock(3.3.2, 5.3.7) [5]	com.domobile.applockwatcher	O	O	AES/CBC/PKCS7Padding	'/data/data/com.domobile.applockwatcher/shared_pref/com.domobile.applockwatcher_preferences.xml'의 'image_lock_pattern' 값에 존재	'2011071120170711' 고정값 사용
Calculator - photo vault & Video Vault Hide Phothos (10.0.7) [6]	com.hld.anzenbokusucal	O	O	AES/CBC/PKCS5Padding	Shared Preference의 Share_privacy_safe.xml 파일에 AES/CBC/PKCS5Padding 암호화 알고리즘을 통해 암호화되어 존재	'Rny48Ni8aPjYCnUI' 고정값 사용

인하였다[8]. 그 외 vault 앱을 정적 분석하는 방법에 대하여 연구가 있다[9-10].

2.1. Encrypt File Free(1.0.6)

Encrypt File Free는 100,000 이상 다운로드를 보이며 리뷰 375개 평점 3.5인 안드로이드 애플리케이션이다. 이 앱은 이용자가 암호화를 원하는 앱을 선택해 암호화하는 앱으로 접근제어 기능과 파일 암호화 기능을 보유한 앱이다. 암호화, 복호화 시에는 Password를 요청하지 않고 앱 구동 시에만 Password를 요청한다. Password를 이용한 접근제어와 파일 암호화를 제공하며, Password는 databases/optionscrypt 파일과 암호화된 파일의 첫 부분에 MD5 되어 저장한다. 암호화는 내부의 독자적인 알고리즘을 사용해 진행하며 공개 암호화 알고리즘들을 일절 사용하지 않는다. 그래서 Main 클래스와 Crypt 클래스를 분석해 복호화 알고리즘을 만들었고 이를 이용해 원본을 획득 가능하다.

2.2. CJ One Card(2.7.0)

CJ One Card는 10,000,000 이상 다운로드를 보이며, 리뷰는 25,200개 이상, 평점은 3.7인 안드로이드 애플리케이션이다. 이 앱은 콘텐츠나 제품 구매 시 쿠폰 발행 및 포인트 적립을 수행하는 앱으로 접근제어 기능과 파일 암호화 기능을 보유하고 있다. 앱의 AES 클래스를 분석한 결과, “cjonemobilecard1”이라는 고정된 암호 키를 SetCardPW 클래스에서 사용자에게서 입력받은 Password와 함께 AES 알고리즘을 이용해 암호화해 저장한다. 저장된 값이 있는 파일은 루팅 된 단말기에서 접근이 가능하며 shared_prefs/CjOneCarePref.xml의 autoLoginUserPass 값에 저장한다.

2.3. Gallery Lock Lite (4.6)

Gallery Lock Lite는 10,000,000 이상 다운로드를 보이며 428,000개 이상의 리뷰, 평점 4.1점의 안드로이드 애플리케이션이다. 이 앱의 기능은 사진과 동영상을 숨겨주는 앱으로 접근제어 기능을 수행한다. 앱 구동 시 Password를 입력해야 하고, 앱에서 사진 또는 동영상 파일을 선택하면 갤러리에서는 해당 파일이 더 이상 보이지 않는다. 앱의 Password는 shared_prefs/com.morrison.gallery의 lock-lite_preferences.xml 파일에 암호화되지 않고 그대로 저장되어 있다. 암호 키는 자바 API를 통해 랜덤하게 생성해 사용한다. 앱에 암호 기능은 구현되어 있었지만 파일이 실제로 암호화되지는 않고 확장자만 변경해 숨겨진 폴더로 이동되기만 한다.

2.4. LockMyPix(4.2.6-b)

LockMyPix는 10,000,000 이상 다운로드를 보이며 리뷰 274,000개 이상, 평점 4.6점 안드로이드 애플리케이션이다. 사진과 동영상을 암호화해 보관하는 기능을 하며 PIN과 Password를 이용해 접근제어를 한다. 이는 암호화되어 ‘.ini.keyfile.ctr’파일로 저장되는데 그 과정은 다음과 같다. 먼저 PIN과 Password를 SHA-1을 이용해 해싱한다. 해싱한 160bit의 결과에서 상위 128bit를 추출해 AES128-CTR의 Key, IV 값에 동일하게 사용해 암호화한다.

이 앱은 첫 실행 시 Shared storage에 ‘LockMyPix’라는 숨김 디렉토리를 생성하고, PIN 설정, 복구 e-mail 설정 등 초기 설정을 진행해 원본 경로 정보, 썸네일, 데이터 등을 암호화해 저장한다. LockMyPix 앱을 통해 암호화 저장할 때 원본 데이터를 복사해 저장한다. 이 때 원본 데이터는 삭제되지 않지만 사용자의 갤러리에서는 보이지 않게 된다. 사진과 동영상의 암호화 과정은 앞서 말했던 PIN과 Password 암호화 과정에서 추출한 Key, IV 값을 이용해 원본 사진 파일을 AES128-CTR을 이용해 암호화한다.

2.5. KaKaoTalk(3.6.5)

KakaoTalk은 100,000,000 이상 다운로드를 보이며 리뷰 3,140,000개 이상의 리뷰, 평점 4.3점의 한국 사람들이 가장 많이 사용하고, 친숙한 애플리케이션이다. 카카오톡은 본래 기능이었던 채팅 기능 외에도 현재 앱 내부에 여러 가지 기능들을 추가해 없어서는 안 될 앱으로 자리 잡았다. 이 중 카카오톡 채널관리자는 PBE(Password-Based Encryption)를 이용해 암호 키를 생성하고 데이터 암호화를 수행한다. PBE는 PRF와 Cipher로 표현되고, PRF(Pseudo Random Function)을 이용해 암호 키를 생성하고, 이를 기반으로 암호 알고리즘을 이용해 데이터를 암호화, 복호화

즉, Cipher 과정을 수행한다. 여기서 PRF로 사용된 해시 함수는 SHA1이고, Cipher는 AES256/CBC/PKCS #7Padding이 사용되었다. 암호 키 생성에는 Password 값, Salt 값, Iteration 값이 사용되고 이를 통해 Cipher의 블록 알고리즘에 사용되는 암호 키 길이인 32Byte 만큼 생성된다. Password는 코드 내 존재하는 고정 값이고, Salt는 plus_friend.db의 chat_log 테이블에 있는 profile_id 값과 “kevin” 문자열을 연접하고 그 값의 상위 16바이트로 구성된다. 만약 그 값이 16바이트 미만이라면 나머지 부분을 0으로 패딩 해 16바이트로 만든다. Iteration은 반복 횟수를 의미하며 2회 반복한다.

2.6. SSE-File/Text Encryption & Password Vault(1.4.7)

SSE-File/Text Encryptor & Password Vault는 500,000 이상 다운로드를 보이며 리뷰 3,530개 이상, 평점 4.4인 안드로이드 애플리케이션이다. 이 앱은 파일과 문자열 암호화를 주 기능으로 한다. Password는 사용자로부터 입력받으며, 암호화 시 사용한다. 앱은 난독화는 되어있지 않고 GOST 256bit+, Twofish 256bit, Blowfist 448bit, Serpent 256bit, RC6 256bit, AES(Rijndael) 256bit를 이용해 파일을 암호화한다. F5 알고리즘을 이용한 스테가노그래피 기능도 있다는 것이 특징이다. PRO 버전은 이에 추가적으로 Paranoia C4 Pro Version 2048bit, SHACAL-2 512bit, Threefish 1024bit를 제공한다.

2.7. Vault(6.4.22.22)

Vault는 2011년 12월 16일에 출시된 앱으로 100,000,000 이상 다운로드 수와 1,260,000 이상 리뷰, 그리고 4.5점의 평점을 갖고 있다. 이 앱은 휴대전화에서 개인의 사진과 비디오를 숨기고, 앱 잠금, 앨범 잠금, 전화 알림, 비공개 브라우저, 클라우드 백업의 여러 가지 기능을 가지고 있다. 이 앱은 암호화된 콘텐츠를 .video 나 .image 라는 두 하위 폴더가 포함되어 있는 /sdcard/SystemAndroid/Data/MTC1MDU4OQ== 이라는 디렉토리에 저장한다. 소스코드 내에서는 B라고 불리는 반복 단일 바이트를 이용해 각 미디어 파일의 상위 80Byte만 암호화함을 알았다. 예를 들어 JPEG의 경우에는 0xFFD8이고 B=0x3D가 된다. 이는 이 미디어를 복호화 하기 위해서는 상위 80Byte에 대해 XOR 0x3D를 한다. 그리고 이 앱은 Password를 틀릴 때마다 사진을 찍어서 저장한다.

2.8. Keepsafe(7.3.1)

Keepsafe는 2011년 6월 3일에 출시된 앱으로 총 다운로드 수 50,000,000 이상, 리뷰 1,800,000만 개 이상, 평점 3.9점의 안드로이드 애플리케이션이다. 이 앱은 개인의 사진과 비디오 등을 지문인증과 PIN 보호 등 여러 암호화 기술로 보호하는 기능을 가지고 있다. Keepsafe 앱은 Password를 Shared prefs 폴더의 com.ki.safe_preferences.xml 파일에 master-password 값에 암호화 없이 저장한다. 사진과 비디오 등 미디어 파일은 이름을 해시값으로 변경했고,/sdcard/.keepsafe/manifests/primary 의 하위 폴더에 암호화된 데이터들이 저장되어 있었다. 각 하위 폴더들은 파일의 앞 두 글자를 사용해서 명명되었다. 암호화, 복호화 함수는 앱의 /lib/armeabi-v7a/libcrypt_user.so에 있는 네이티브 라이브러리에서 발견되었다. 암호화는 다음과 같은 세 단계로 진행한다. 1단계 : 미디어 파일을 16,384Byte의 데이터 블록으로 나눈다. 2단계 : 모든 블록을 IV(Initialization Vector)와 암호를 이용해 암호화한다. 이 때, 크기는 그대로이다. 3단계 : 모든 블록이 암호화되면 파일의 시작 부분에 3,728Byte의 일정한 크기를 갖는 초기화 블록인 block zero를 추가한다. 이 블록은 Keepsafe의 로고 3,711Byte, IV 16Byte, Terminator 1Byte로 구성된다. 사용자가 암호화된 파일을 누르면 Keepsafe의 로고가 PNG 로고 파일이라고 착각하게 된다. 복호화는 Key와 IV가 필요한 네이티브 라이브러리에서 processBlock() 함수를 사용해서 수행한다. 키가 block zero(초기화 블록)에 저장되는 동안에 32Byte의 Key는 shared_prefs에 있는 com.kii.secmanager.xml 파일의 taga5a0a33e25e4c9108939f6d6292c4507b045eAe에서 발견할 수 있다. 모든 블록의 암호를 복호화 하면 암호화된 미디어 파일에서 Initialization Block이 최종적으로 제거되어 원래 상태로 복구할 수 있다.

2.9. Hide it Pro(5.4)

Hide it pro는 장치에서 오디오 관리자인 Audio

Manager 앱으로 위장해서 사진과 비디오를 숨긴다. 이 앱은 50,000,000 이상 다운로드와 654,000 이상의 리뷰, 평점 4.4점을 보이고 있다. 앱 내부에 들어가서 화면 상단의 로고를 클릭하면 앱이 실행된다. 사진과 비디오 등 미디어 파일은 /sdcard/ProgramData/Android/Language/.fr/의 하위 폴더에 암호화되지 않은 채로 따로 저장한다. 비밀번호는 shared prefs의 com.hideitpro_preferences.xml파일에 그대로 존재한다. New Album은 숨긴 파일을 저장하는 앨범의 기본 이름이고 사용자가 이름을 바꿀 수 있다.

2.10. GalleryVault(2.9.1)

GalleryVault는 10,000,000 이상 다운로드 수 5,850,000개 이상 리뷰, 평점 4.0점의 안드로이드 애플리케이션이다. 앱의 기능은 사진과 동영상 및 기타 파일들을 숨기고 암호화하는 것이다. 사진과 동영상을 숨길 때 /sdcard/.galleryvault_DoNotdelete_1466617307/file/ 하위에 상위 10Byte는 0x00으로 덮어쓰고, 원래 10Byte는 데이터베이스에 배치한다. 파일 재구성을 위해 데이터베이스를 쿼리하고 파일을 식별한 후 헤더를 다음 단계에 맞춰 재배치한다.

1단계 : ‘SELECT name, path, org file header blob FROM file’으로 데이터베이스 파일 /sdcard/galleryvault_DoNotDelete_1446610307/backup/galleryvault.db를 쿼리 해 원래의 헤더 문자열을 검색한다. 이렇게 하면 미디어 파일의 원래 이름, 분해된 미디어 파일의 경로 및 각 파일에 대한 지워진 10Byte의 데이터가 반환된다.

2단계 : 이름과 경로를 사용해 파일을 식별한 후 파일의 처음 10Byte를 원래 헤더로 대체한다. 단말기가 활성화되어 있을 경우 MD5나 SHA1값에 대해 swap attack을 수행할 수 있다. 이 값은 shared_prefs 내 LockPin의 Kidd.xml에 존재한다.

2.11. AppLock(3.3.2 5.3.7)

AppLock은 100,000,000 이상 다운로드 수 와 1,030,000 이상의 리뷰, 별점 4.3점을 보이고 있고, 접근 제어 기능과 Vault(금고) 기능, 그리고 정보 은닉 기능을 가지고 있는 안드로이드 애플리케이션이다. 이 앱은 첫 구동 시 Pattern을 설정한다. 이 Pattern으로 접근 제어 기능을 수행한다 이는 AES/CBC/PKCS7 Padding 암호화 알고리즘을 사용하고, MD5, SHA1, SHA256을 해시 알고리즘으로 사용한다. 암호키는 랜덤 값을 생성해 사용하고, Salt 값은 ‘2011071120170711’의 고정값을 사용한다. 이렇게 생성된 암호키는 ‘/data/data/com.domobile.applockwatcher/shared_pref/com.domobile.applockwatcher_preferences.xml’의 ‘image_lock_pattern’ 값에 저장한다. AppLock은 APK, 파일, 오디오, 비디오, 이미지 등 각 파일 형식에 맞춰 Vault 기능을 수행한다. Vault 기능을 수행할 때는 파일 이름을 변경하고 확장자를 제거해 인식하기 어려운 경로를 만들어서 저장한다. 이때 파일은 암호화되지 않고 인식하기 어려운 경로를 통해 이동한다. 저장된 파일들은 AppLock의 특정 로그파일에 저장된다. 이 경로는 기본적으로 /storage/emulated/0/.dom0o7b1i1le/dnt_remove/로 설정되어 있다. 이 디렉토리 아래에 0~99까지의 숫자를 MD5로 해시해 100개의 디렉토리를 만들고 아래에 파일 형식에 맞춰 디렉토리를 추가 생성한다. 이 형식은 ‘.thumb’, ‘.file’, ‘video’, ‘image’ 등이 있다. 이후, 파일의 이름을 이동한 시점의 UNIX TIME으로 변경해 확장자를 제거하고 이동한다. 이 앱은 암호화와 복호화와 관련된 데이터들이 모두 특정 xm l파일에 적혀있고 암호 키의 값 또한 Plaintext(평문)으로 적혀있어 취약점이 있다.

2.12. Calculator – photo vault(10.0.7)

Calculator는 10,000,000 이상 다운로드 수와 리뷰 3,560,000개 이상, 그리고 평점 4.8의 높은 평점을 기록하고 있는 안드로이드 애플리케이션이다. 이 앱은 계산기로 위장되어 있으며, 접근 제어 기능과 파일 암호, 은닉을 수행하는 개인정보보호 앱이다. 처음 앱 구동 시 PIN 번호를 설정하며 동시에 비밀번호 복구 질문과 답, E-Mail을 설정한다. 이를 통해 접근제어 기능을 수행한다. 관련 정보들은 Shared Preference의 Share_privacy_safe.xml 파일에 AES/CBC/PKCS5Padding 암호화 알고리즘을 통해 암호화되어 저장한다. 이때 사용하는 암호키는 3개의 고정된 문자열들을 조합해 사용한다. 먼저 두 개의 문자열은 앱 내부에 존재하고, “CnUI”와 “Rny4”다. 나머지 문자열은 “8Ni8aPjY”로 라이브러리 libkey.so에 존재한다. 이 세 개의 문자열을 조합해 “Rny48Ni8aPjYCnUI”라는 암호 키를 생

성한다. 파일 암호화에서도 위의 접근 제어를 수행할 때 사용한 암호 알고리즘과 암호 키를 동일하게 사용한다. 파일을 암호화한 후에 각 파일은 각각의 파일 형식에 맞춰 /storage/emulator/0/.privacy_safe 경로에 저장된다. 디렉토리명은 'db', 'other', 'audio', 'video', 'picture' 등으로 나타났다. 파일명의 첫 글자는 파일을 암호화했는지 은닉했는지에 따라 암호화를 수행했으면 'e', 은닉했으면 'h'로 나타나고 다음은 Java의 randomUUID()를 이용해 랜덤 한 값을 생성해서 사용한다. 그 뒤에는 원본 파일의 확장자를 특정한 값으로 치환하여 사용한다. 이 파일들은 암호화에 사용된 암호 알고리즘과 암호 키를 이용해 복호화 프로그램을 제작해 복호화 할 수 있다.

Ⅲ. 암호기능 적용 앱 분석

암호 앱 분석을 위해 구글 플레이 기준 10,000,000회 이상 다운로드와 최근 업데이트된 앱 5개를 다운로드하여 분석을 진행했다. 분석 대상 앱 리스트는 [표 2]와 같다. 해당 앱들의 암호화 기능을 확인하기 위해 정적 분석 도구인 JADX-GUI를 활용하였다. 분석 결과 LOCKit 앱 외의 다른 앱들은 구글 플레이에 설명된 기능과는 달리 개인정보를 제대로 보호하고 있지 않았다. 각 앱의 취약점을 분석한 결과는 다음과 같다.

3.1. Photo Lock App Hide Pictures & Videos

Photo Lock App Hide Pictures & Videos는 50,000,000건 이상 다운로드된 앱이며 주요 기능은 PIN을 이용한 접근제어 기능과 사진 및 동영상 파일을 앱 내부에 암호화하여 저장한 뒤 원본은 삭제하는 사진, 동영상 금고 기능이 있다.

이 앱을 처음 실행하면 4~12자리의 숫자 비밀번호 설정 및 구글 계정 로그인을 진행한다. 등록한 정보 [그림 1]와 같이 data/data/vault.gallery.lock/shared_prefs/vault.gallery.lock_preferences.xml의 password 항목의 값으로 저장돼있으며 암호화 없이 그대로 저장함을 확인할 수 있다.

사진, 동영상 금고 기능은 사진, 동영상을 선택하여 원본 파일을 삭제하고 앱 내부 저장소에 저장하는 기능이다. 해당 사진 파일은 /data/data/vault.gallery.lock.files/lockerVault/Images1769/Pictures/경로에 암호화 없이 저장되어 있음을 확인했고, 동영상 또한 암호화 없이 /data/data/vault.gallery.lock.files/lockerVault/Videos1769/Videos/경로에 저장됨을 확인하였다. 이 앱은 접근제어, 금고 기능에 암호화를 사용한다고 설명하고 있지만 실제로는 password와 은닉된 파일이 저장되는 경로와 파일 모두 암호화되어 있지 않다.

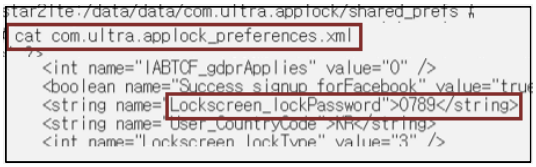
〔그림 1〕 Photo Lock App Hide Pictures

〔표 2〕 분석 대상 앱 리스트

앱이름	패키지이름	다운로드 수	마켓 출시일	최근 업데이트일	앱 기능
Photo Lock App - Hide Pictures & Videos	vault.gallery.lock	1000만+	2016.01.29	2022.05.14	앱접근제어, 사진동영상금고
앱 잠금 - Ultra Applock	com.ultra.applock	1000만+	2019.07.06	2022.07.25	앱접근제어
앱 잠금 (Smart App Protector)	com.sp.protector.free	5000만+	2010.11.24	2022.06.24	앱접근제어
어플 잠금 (Perfect AppLock)	com.morrison.applocklite	1000만+	2010.10.22	2020.10.12	앱접근제어
LOCKit - App Lock, Photos, Vault, Fingerprint Lock	com.ushareit.lockit	1000만+	2016.02.03	2021.09.28	앱접근제어, 사진동영상금고

3.2. 앱 잠금 - Ultra Applock

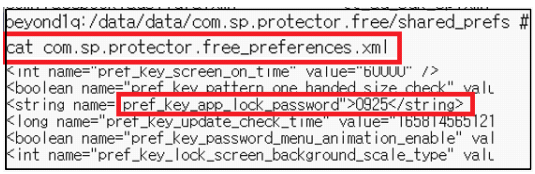
Ultra Applock는 10,000,000건 이상 다운로드된 앱이며 주요 기능은 앱 실행 시 접근제어 기능을 통한 앱 잠금 기능이다. 이 앱을 처음 실행하면 숫자 비밀번호를 설정, 비밀번호 찾기 기능을 위한 질문과 답을 입력한다. 사용자가 입력한 정보는 [그림 2]와 같이 /data/data/com.ultra.applock/shared_prefs/com.ultra.applock_preferences.xml의 Lockscreen_lockPassword 항목에 평문으로 저장된다. 이 앱은 접근제어에 암호화를 사용한다고 설명하지만 실제로는 평문으로 저장됨을 알 수 있었다.



〔그림 2〕 Ultra Applock

3.3. 앱 잠금 (Smart App Protector)

Smart App Protector는 50,000,000건 이상 다운로드된 앱이며 주요 기능은 Ultra Applock과 유사하게 앱 실행 시 접근제어 기능을 통한 잠금 기능이다. 이 앱을 처음 실행하면 사용자에게 숫자 포맷의 비밀번호를 요구한다. 사용자가 입력한 비밀번호는 [그림 3]과 같이 /data/data/com.sp.protector.free/shared_prefs/com.sop.protector.free_preferences.xml의 pref_key_app_lock_password 항목에 평문으로 저장된다. 이 앱도 Ultra Applock과 마찬가지로 접근제어에 암호화를 사용한다고 설명하지만 실제로는 평문으로 저장하는 것을 알 수 있다.



〔그림 3〕 Smart App Protector

3.4. 어플 잠금 (Perfect AppLock)

Perfect AppLock는 10,000,000건 이상 다운로드된 앱이며 주요 기능은 앞의 Ultra Applock, Smart App Protector와 마찬가지로 앱 실행 시 접근제어 기능을 통해 앱을 보호한다. 이 앱을 처음 실행하면 초기 비밀번호가 7777로 설정되어 있으며 이후 사용자가 비밀번호를 변경할 수 있다. 변경된 비밀번호는 [그림 4]와 같이 /data/data/commorrision.applockite/shared_prefs/com.morrision.applocklite_preferences.xml의 pwd 항목에 평문으로 저장된다. 이 앱 또는 앞의 2가지 앱처럼 사용자의 비밀번호를 평문으로 저장한다.

〔그림 4〕 Perfect AppLock

3.5. LOCKit

해당 앱은 2016년 6월에 출시한 앱으로 110,000,000건 이상 다운로드 및 리뷰 560,000 이상이 작성된 앱이다. 주요 기능으로는 특정 앱에 잠금을 걸어 실행 시 PIN 번호를 입력하게 하는 접근제어기능과 핸드폰 내부에 저장된 이미지, 동영상 파일을 암호화하여 해당 앱을 통해서만 열람 할 수 있게 하는 잠금 기능 또한 존재한다. 분석 결과 파일을 암호화하는 기능과 입력한 PIN을 암호화하여 저장하는 기능이 존재하는 것으로 확인되어 4장에서 해당 앱에 적용된 암호 기능과 복호화 방안을 기술한다.

Ⅳ. LOCKit의 암호 기술 분석

4.1. 접근제어 기술 분석

해당 앱을 처음 실행하면 접근제어에 사용할 숫자로만 구성된 최대 10자리의 PIN 번호와 비밀번호 복구 시에 사용될 질문과 답변을 설정하게 된다. 등록한 정보는 [그림 5]와 같이 data/data/com.ushareit.lockit/shared_prefes/Settings.xml 파일의 KEY_LOCK_PIN_PASSWORD 항목의 값으로 저장되며 이때의 값은 암

〔그림 5〕 LOCKit

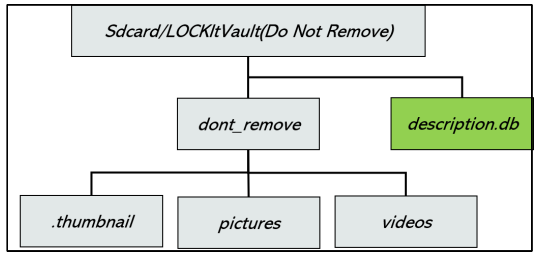
호화되어 있다.

이 값을 복호화하기 위해서는 해당 xml 파일에 저장될 때 동작하는 코드를 확인해야 하며 정적 분석 도구인 JADX를 활용하였다. 해당 코드를 확인한 결과 PIN 번호가 입력되면 해당 PIN 번호를 AES-128/CBC/PKCSSPadding을 사용하여 암호화를 진행하며 이때의 Key 값 alockit_password을 사용하고 해당 Key 값은 소스 코드에 하드코딩 되어있다. KEY_LOCK_PIN_PASSWORD의 암호화된 값을 저장하기 전 Base64로 인코딩하여 저장한다. 해당 과정을 역순으로 진행하면 data/data/com.ushareit.lockit/shared_prefes/Settings.xml 파일의 KEY_LOCK_PIN_PASSWORD 값을 참조하여 PIN 번호를 복호화 한다. 암호화 알고리즘은 다음 수식과 같다.

$$\begin{aligned} P &= PIN(10) \\ Key &= alockit\_password \\ IV &= alockit\_password \\ Encryption\ Key &= \\ Base64(AES\_CBC(P,\ Key,\ IV)) & \end{aligned} \quad (1)$$

4.2. 사진, 동영상 금고 기능 분석

LOCKit 앱 기능 중에는 PIN 번호를 이용한 접근제어 기능뿐만이 아닌 앱 내부에 사진, 동영상 파일을 암호화하여 저장하는 기능이 존재한다. 해당 기능을 통해 원하는 사진, 동영상 파일을 은닉할 수 있으며 이때 원본 파일은 삭제되고 앱을 통해서만 암호화된 파일을 복호화 하여 원래의 사진, 동영상 형태로 확인할 수 있다. 암호화하여 은닉된 사진, 동영상 파일은 SDcard/.LOCKitVault(Do Not Remove) 숨김 폴더 내부에 저장되어 있다. 해당 폴더의 구조는 다음 [그림 6]과 같다. 하위 폴더 .thumbnamil, pictures, videos에 사진, 동영상 파일을 암호화하여 저장한다.description.db 파일은 DB 파일이며 열람 시 3개의 테이블로 구성되어 있고 그중 android_metadata, sqlite_sequence 테이블은 사용하지 않고, [표 3]의 스키마를 갖는 File_Vault



〔그림 6〕 LOCKit의 Vault 폴더 구조

〔표 3〕 File Vault 테이블

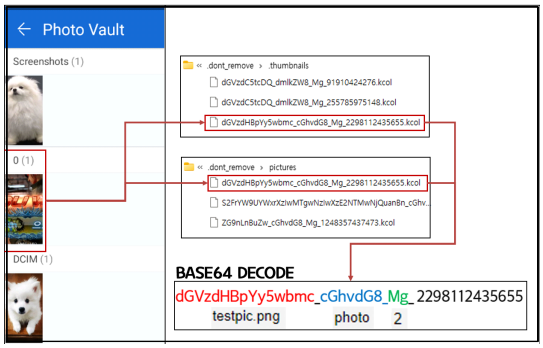
컬럼명	해석
ow_id	저장된 순번
file_type	사진, 동영상 구분
dst_path	해당 파일이 저장된 위치
content_description	해당 파일 특징기록(암호화)

테이블만 사용한다.

content_description 항목은 AES-128/CBC/PKCSSSPadding을 사용하였고 키값과 Initial Vector값인 alockit_password을 사용하여 암호화한 뒤 Base64로 인코딩한 것을 확인할 수 있었다. 해당 과정을 역순으로 하여 복호화를 진행할 수 있었고 [그림 7]과 같은 JSON 형식 데이터임을 확인했다. 해당 JSON에는 원본 파일의 경로, 원본 파일 이름, 암호화된 썸네일 파일 경로, 암호화된 사진, 동영상 파일 경로, 해당 앱에서 복호화 할 때 사용하는 인자 값이 저장되어있다. 복호화 한 description.db의 내용을 통해 [그림 8]의 해당 사진이 SDCard/.LOCKitVault(Do Not Remove)/.dont_remove/.thumbnails/dGVzdHBpYy5wbmc_cGhvdG

{"type":"photo","id":"974","ver":"","name":"testpic","has_thumbnail":true,"photoid":974,"showname":"testpic","filepath":"₩/storage₩/emulated₩/0₩/testpic.png","rawfilename":"testpic.png","filesize":1049581,"datemodified":1658714650000,"thumbnailpath":"₩/storage₩/emulated₩/0₩/.LOCKitVault(Do Not Remove)₩/.dont_remove₩/.thumbnails₩/dGVzdHBpYy5wbmc_cGhvdG8_Mg_2298112435655.kcol","format":"png","albumid":1389444597,"albumname":"0","orientation":0,"dst_path":"₩/storage₩/emulated₩/0₩/.LOCKitVault(Do Not Remove)₩/.dont_remove₩/pictures₩/dGVzdHBpYy5wbmc_cGhvdG8_Mg_2298112435655.kcol","encrypt_version":2}

〔그림 7〕 복호화한 content_descruption



〔그림 8〕 암호화된 썸네일과 사진파일

8_MG_2298112435655.kcol 과 SDCard/.LOCKitVault (Do Not Remove)/.dont_remove/pictures/dGVzdHBpYy5wbmc_cGhvdG8_MG_2298112435655.kcol 경로의 썸네일과 , 사진파일인 것을 확인했으며,

해당 파일의 이름은 파일명을 생성하는 함수를 정적 분석하여 Base64로 인코딩 되어 있음을 확인했다. 디코딩 한 결과 dGVzdHBpYy5wbmc_cGhvdG8_MG_2298112435655은 “testpic.png_photo_2_2298112435655”로 “파일명.확장자_타입_암호화 버전_나노 타임” 형식임을 확인했다. 암호화 버전은 해당 앱에서 사용하는 암호화 함수의 인자값이고, 나노 타임은 JAVA에서 특정 메서드가 실행될 때 걸리는 시간을 측정한 값이다. 은닉된 사진 파일들의 Hex 값을 분석한 결과 최초의 1,024바이트만 암호화함을 확인했다. 복호화는 다음 수식과 같다.

$$
\begin{aligned}
&Bytearray[] = read(1024)\\
&Shift\,Value = Bytearray[0] \veebar 121\\
&For\ a \in Bytearray\\
&\quad \text{if } a \geq 65 \text{ and } a \leq 122\\
&\qquad \text{if } a > 97\\
&\qquad\quad a = ((52 + (a - 74))\%52)\\
&\qquad else\\
&\qquad\quad a = ((52 + (a - 68))\%52)\\
&\qquad \text{if } a < 26\\
&\qquad\quad a = (a + 65) \veebar Shift\,Value\\
&\qquad else\\
&\qquad\quad a = (a + 71) \veebar Shift\,Value\\
&\quad else\\
&\qquad a = a \veebar Shift\,Value
\end{aligned}
\tag{2}
$$

4.3. LOCkit 앱 분석 정리

해당 앱은 접근제어, 사진 동영상 금고 기능이 존재하고 접근제어에는 입력받은 PIN 번호를 AES-128/CBC/PKCSSPadding 암호화하였고, 이때의 Key 값과 Initial Vector 값은 alockit_password이다. 이후 Base64 인코딩한 값을 data/com.ushareit.lockit/shared_prefes/Settings.xml 파일의 KEY_LOCK_PIN_PASSWORD의 값으로 저장하여 PIN번호를 관리한다.

사진 동영상 금고 기능을 사용하면 해당 파일은 최초 1,024바이트만 암호화하고 SDcard/.LOCKitVault (Do Not Remove) 숨김 폴더 하위에 경로에 파일명을 “파일명.확장자_타입_암호화 버전_나노 타임” 형태로 수정한 뒤 Base64로 인코딩하여 저장한다. 그 후 원본 파일은 삭제한다. 이렇게 금고에 저장한 파일들의 경로는 암호키와 Initail Vector 값을 alockit_password로 하는 AES-128/CBC/PKCSSPadding을 사용하여 암호화 후 Base64로 인코딩한 후 description.db 파일 filevault 테이블의 content_description 컬럼에 저장한다.

Ⅴ. 결 론

본 논문은 기존 논문에서 분석하였던 암호 기능을 사용한 안드로이드 앱들의 암호화 알고리즘과 키 관리 방식에 대한 동향을 조사했다. 또한, 기존 논문에서 다루지 않은 Google Play 기준 다운로드 10,000,000건 이상의 암호화 앱인 Photo Lock App, Ultra Applock, Smart App Protector, Perfect AppLock, LOCKit을 분석했다. 이러한 앱들은 Google Play는 접근 제어 및 파일 암호화를 통해 파일들을 보고한다고 앱을 소개하고 있다. 하지만 실제로 파일 암호화와 접근제어 Password 암호화를 진행하는 앱은 LOCKit이 유일했다.

악성앱으로 인한 스마트폰의 개인정보가 유출되는 사건이 많이 발생함에 따라 스마트폰의 파일들을 암호화 보관하는 앱의 수요가 높아지고 있고, 이를 지원하는 다양한 앱들이 출시되고 있다. 하지만 대부분의 앱들은 실제로는 암호화를 하지 않거나 암호화를 하더라도 복호화에 필요한 암호 키와 Initial Vector값을 앱 내부에 저장함으로써 역공학을 통해 쉽게 복호화가 가능한 것을 알 수 있었다. 심지어 본 논문에서 분석한 LOCkit의 경우는 파일 암호화 시 독자적인 알고리즘을 사용함으로써 보안 강도가 매우 낮음을 알 수 있었다.

이러한 문제점들을 해결하기 위해서는 공개키 암호 알고리즘을 사용하여 파일의 암호키를 관리하는 것이 바람직하고, 그렇지 않다면 암호화 알고리즘을 안드로이드 Java가 아닌 라이브러리로 구현하여 암호화키와 Initial Vector 값의 유출을 최대한 방지해야 한다. 또

한, LOCKit처럼 독자적인 암호화 알고리즘을 개발해서 사용하는 것이 아닌 알려진 강력한 암호 알고리즘을 사용해야 한다.

참 고 문 헌

[1] 박상호, 김현진, 권태경, "안드로이드 스마트폰 암호 사용 앱 보안 분석 및 대응." 정보보호학회논문지, 23(6), 1049-1055, 2013

[2] 박진성, 서승희, 김역, 이창훈, "포렌식 분석을 위한 LockMyPix 의 미디어 파일 복호화 방안 연구." 디지털포렌식연구14.3, 269-278, 2020

[3] 이세훈, 박명서, 김기윤, 허욱, 김종성, "암호화된 SNS 애플리케이션 데이터 복호화 방안 연구: 카카오톡 채널 관리자, Purple, TongTong." 디지털포렌식연구, 14(1), 87-96, 2020

[4] Zhang, X., Baggili, I., & Breitinger, F. "Breaking into the vault: Privacy, security and forensic analysis of Android vault applications." Computers & Security, 70, 516-531, 2017

[5] 홍표길, 김도현*, "모바일 정보 은닉 앱에 대한 취약점 분석", 한국정보보호학회 동계학술대회 (CISC-W'21), 2021.11.27.

[6] 이정우, 홍표길, 김도현. "개인정보보호 안드로이드 앱에 대한 취약점 분석." 한국정보통신학회 종합학술대회 논문집 26.1 , 184-186. 2022

[7] 김성진, 허준범, "모바일 어플리케이션 개인정보 유출탐지 및 보안강화 연구." 정보보호학회논문지 29.1 , 195-203, 2019

[8] Kim Jun-Sub, "블록암호 기반 랜섬웨어에 대한 분석 사례 동향." Review of KIISC, 32(3), 41-47, 2022

[9] 강수진, 신수민, 김기윤, 김종성, "디지털 포렌식 관점에서의 사진 잠금, 사진/비디오 숨기기, Safe Gallery/Camera 애플리케이션 분석." 디지털포렌식연구14.2, 125-137, 2020

[10] 김기윤, 장성우, 김종성. "LG 갤러리 애플리케이션 잠금 파일 복호화 연구." 디지털포렌식연구 12.2, 31-38, 2018

〈저 자 소 개〉

이 성 원 (Sungwon Lee)
정회원
2022년 2월 : 부산가톨릭대학교 컴퓨터공학과 졸업
2022년 2월~현재 : 부산가톨릭대학교 컴퓨터공학과 석사과정
<관심분야> 디지털 포렌식, 머신러닝

김 청 운 (Chungwoon Kim)
2017년 3월~현재 : 부산가톨릭대학교 컴퓨터공학과 학부과정
<관심분야> 디지털 포렌식, 취약점 분석, 인공지능

김 도 현 (Dohyun Kim)
종신회원
2019년 8월 : 고려대학교 정보보호대학원 공학박사
2017년 7월~2019년 8월 : 한국전자통신연구원 정보보호연구본부 연구원
2019년 9월~2020년 3월 : 고려대학교 정보보호연구원 연구교수
2020년 4월~2022년 2월 : 부산가톨릭대학교 컴퓨터공학과 조교수
2022년 3월~현재 : 부산가톨릭대학교 컴퓨터정보공학과 조교수
<관심분야> 디지털 포렌식, 취약점 분석, 사이버보안 등