Research on Note-Taking Apps with Security Features

융합보안학과 윤세영
유투브 주소: https://youtu.be/vj_N95AxS84

한성대학교 HANSUNG UNIVERSITY

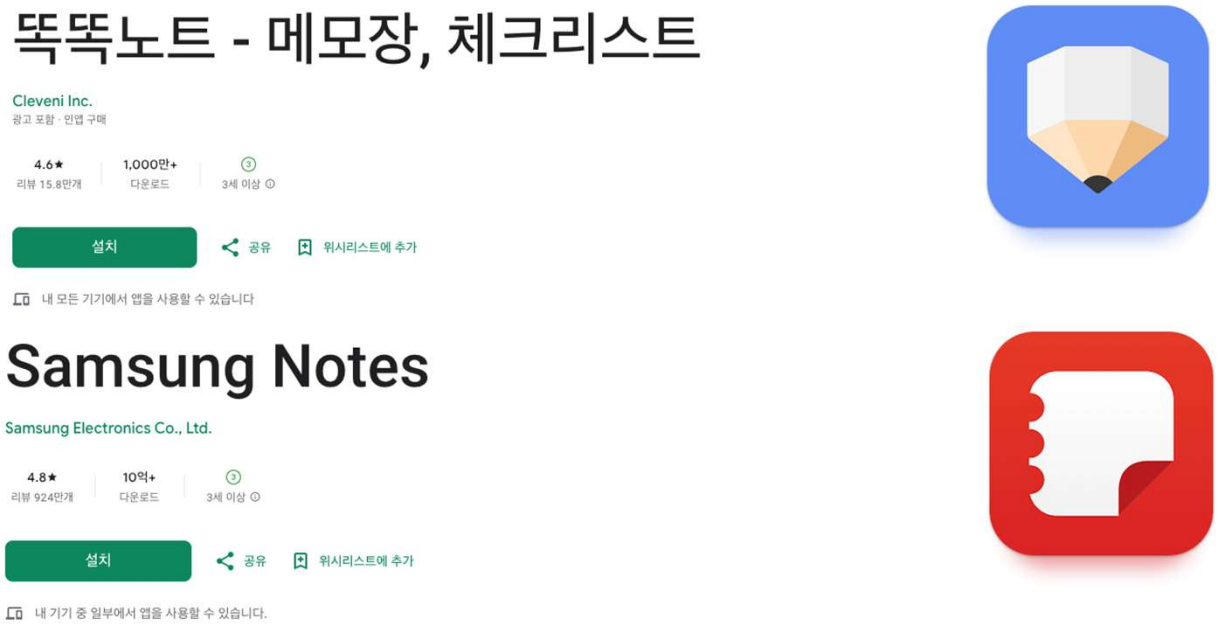
CryptoCraft LAB

목차

Research on Note-Taking Apps with Security Features

'ClevNote', 'Samsung Notes'라는 메모 애플리케이션을 대상으로 보안 기능을 분석한 논문

JEB Decompiler와 IDA Pro를 가지고 정적, 동적 분석 시도



Apk 파일을 JEB 디컴파일러로 확인하면 다음과 같이 코드가 난독화(자바 객체, 함수, 변수가 어떤 역할을 하는지 알아볼 수 없도록 변경되어 있다는 것.) 예시 사진에서 보면 변수명이나 함수가 a, b 등 알파벳으로 써있음.

c/Source

```
public class c {
    public static boolean a(Context context0, String s, boolean z) {
        String s1 = c.g(context0, z);
        String s2 = c.i(context0, z);
        e e0 = f.c();
        e0.r(s);
        e0.o(a.e);
        if(s1 == null) {
            b b0 = com.socialnmobile.colornote.b.l(context0).h().d();
            if(b0 != null) {
                sm.r4.c.l().i("CHECK MASTER PASSWORD : SIGNED IN").q().m("org:" + s2 + ":" + b0.d + ":"
                return false;
            }

            sm.r4.c.l().i("CHECK MASTER PASSWORD").m("org:" + s2 + ":" + z).q().o();
            return false;
        }

        try {
            if(e0.d(s1).equals(s2)) {
                if(z) {
                    return true;
                }
```

그러나 라이브러리에서 호출되는 함수의 이름은 유지됨.

(애플리케이션이 라이브러리를 통해 특정 기능을 수행하기 때문에, 호출면에서 오류가 발생하지 않으려면 이름을 유지해야 함.)

```
public static void Q(Context context0, Uri uri0, boolean z) throws sm.M3.a {
    long v = System.currentTimeMillis();
    Cursor cursor0 = context0.getContentResolver().query(uri0, null, null, null, null);
    if(cursor0.moveToNext()) {
        h h0 = new h();
        h0.F(cursor0);
        ArrayList arrayList0 = sm.h4.a.i(h0.i(context0, false));
        for(Object object0: arrayList0) {
            ((c)object0).h(z);
        }

        String s = sm.h4.a.j(arrayList0);
        ContentValues contentValues0 = new ContentValues();
        if(h0.B()) {
            e e0 = f.d(context0);
            contentValues0.put("note", e0.g(s));
            contentValues0.put("encrypted", Integer.valueOf(e0.m()));
        }
        else {
            contentValues0.put("note", s);
        }

        contentValues0.put("status", Integer.valueOf((z ? g.S(h0.k(), 16, 16) : g.S(h0.k(), 0, 16))));
        context0.getContentResolver().update(uri0, contentValues0, null, null);
        g.j(context0);
        x.B(context0, v);
        sm.U3.c.s(context0, uri0);
    }

    cursor0.close();
}
```

ClevNote

ClevNote는 비밀번호 4자리를 사용하여 접근을 제어함

비밀번호 입력 시도 횟수 제한 없음

비밀번호와 관련된 정보는 com.dencreak.esmemo preference.xml 파일에 저장됨

Preference.xml 파일 내에 ispassword 값이 ture로 설정되면 잠금이 완료된 것.

Figure 3: Information related to the password for service lock on ClevNote

ClevNote

잠금 설정된 메모를 보호하기 위해 (메모 데이터를) 암호화함.

설정된 비밀번호를 암호화하여 EncryptMainPass 라는 속성에 저장해둠.

메모의 내용은 esmemo.db 파일에 암호화 되어 저장됨.

Table 2는 ClevNote의 데이터베이스 구조를 나타냄.

Table 2: Classification of encrypted column app data on ClevNote

Package name	File path	Table name	Column name
com.dencreak.esmemo	databases\esmemo.db	accountdatum	a_name, a_bank, a_number, a_holder, a_memo, a_protect
		birthdaydatum	b_name, b_memo, b_protect
		cartdatum	c_name, c_memo, c_protect
		folderdatum	f_name, f_protect
		siteiddatum	s_name, s_address, s_siteid, s_memo, s_protect
		textmemodatum	t_subject, t_body, t_protect

ClevNote

데이터 추출시 Android Backup 기능을 사용하여 추출 가능.

비밀번호 복호화 과정: $P = \text{AES256-CBC-DECRYPT}(C, MK, IV).$

(C는 '암호화된' 비밀번호, MK는 32바이트의 고정 값으로 소스 코드에 FK1으로 하드 코딩 됨,
IV는 16바이트의 초기화 벡터로 소스 코드에 f IV1로 하드코딩 되어 있음)

(=> 하드 코딩 되어 있다?: 고정된 값이 코드 내에 직접적으로 입력되어 있다는 것. 동적으로 생성되는 값이 아니라 개발자가 값을 직접 넣어뒀다는 뜻~ 보안상 취약점이 될 수 있음)

위 식을 통해 암호화된 비밀번호를 복호화하여 사용자 입력 비밀번호와 비교하는 방식으로
비밀번호를 검증함.

ClevNote

메모 내용을 복호화하기 위해서는 데이터베이스에서 protect라는 컬럼을 복호화하여 복호화 키(DK)를 먼저 얻어야 함. 사용되는 복호화 알고리즘: P = AES256-CBC-DECRYPT(C, MK, IV)
여기서 C는 protect 컬럼의 값, MK와 IV는 이전과 같음.

배열 arrd의 값에 따라 R 값 결정됨.
키 후보는 Base64로 인코딩된 상태로 저장되어
이 값을 디코딩한 후 AES256-CBC 알고리즘을 통해
최종 복호화 키를 얻게 됨.

Algorithm 1: ClevNote decryption key acquisition algorithm

```
1  Function getKey (arr);
   Input  : Array arr of integers of size 12
   Output: Decryption key DK
2  if arr[3] mod 2 !=0 then
3      if arr[11] mod 2 == 0 then
4          R = arr[6];
5      else
6          R = arr[5];
7      end
8  else if arr[11] mod 2 == 0 then
9      R=arr[9];
10 else
11     R=arr[7];
12 end
13 EncryptedDK = Base64Decoding(KeyArr[R]);
14 DK = AES256-CBC-Decrypt(C(=EncryptedDK), MK(=FK2), IV(=fIV1))
15 Return(DK);
```

ClevNote

ClevNote 정리:

복호화 대상 – 잠금 비밀번호

복호화 알고리즘 – 1. Base64로 디코딩 2. AES256-CBC-DECRYPT(C, MK, IV) 알고리즘 사용

복호화 대상 – protect 컬럼의 데이터 (메모 내용)

복호화 알고리즘 – AES256-CBC-DECRYPT(C, MK, IV) 알고리즘 사용

MK와 IV의 값이 하드 코딩 되어있어서 복호화하기 비교적 쉬워보임. (보안 문제)

Samsung Notes

삼성 계정으로 로그인한 경우에 노트를 잠글 수 있는 기능 사용 가능
잠금 비밀번호는 4~16자리이며, 숫자, 문자, 특수 문자 포함 가능
잠금 비밀번호는 사용자 인증을 위해 사용되며 데이터 암호화와는 별개의 기능
(즉, 비밀번호는 데이터 보호가 아닌 노트 잠금을 위한 인증용으로만 사용됨)

안드로이드 백업을 지원하지 않아서 데이터 추출에 한계가 있었음 (루팅을 통한 접근을 해볼 수 있지만 포렌식 관점으로 보았을 때, 데이터를 변형하지 않고 무결성을 유지하는 것이 중요하므로 루팅하지 않고 데이터를 추출할 수 있는 방법을 사용해야 함)
-> 루팅 없이 데이터를 추출할 수 있는 방법으로 Samsung Smart Switch (삼성 백업 서비스)를 사용하여 무결성을 유지하면서 데이터를 추출함. (그러나 이 방법도 모든 데이터에 접근할 수 있는 것은 아님)

Samsung Notes

비밀번호를 설정하면 비밀번호와 솔트 값이 com.samsung.android.app.notes preferences.xml과 UserAuthInfo.xml 파일에 저장됨.
키 생성시 PBKDF2withHMACSHA1 알고리즘이 사용되며 반복 횟수는 4000이고 키 길이는 256비트임.
Android Keystore는 RSA/ECB/OAEPWithSHA-256AndMGF1Padding 을 사용하여 키를 관리함.

비밀번호 검증 절차:
우선 비밀번호해시값과 솔트값을 복호화함.
사용자가 입력한 비밀번호와 위에서 복호화된 솔트값을 사용하여 PBEkey를 생성함.
비밀번호해시값과 PBEkey가 일치하면 비밀번호 검증이 완료됨.
(논문에서는 솔트값과 PBEkey가 일치하면 검증이 완료된다고 써있는데 아무리봐도 오타같아요..)

Samsung Notes

데이터 복호화 방법:

암호화된 노트 데이터의 복호화 방법: P= AES256-CBC-DECRYPT(C, MK, IV). – ClevNote와 동일

그러나 IV 값은 0으로 고정되어있고, MK(master key)는 특정 정보를 통해 생성됨.

MK 생성 방법:

/shared_prefs/ 폴더에 저장된 NotesDeviceInfo.xml 파일의 세 가지 정보를 조합하여 생성

NotesDeviceID#1: 앱에서 랜덤하게 생성된 UUID (Universally Unique Identifier).

NotesDeviceID#2: 기기의 고유 식별자인 Android ID.

NotesDeviceID#3: 랜덤하게 생성된 16바이트 길이의 문자열.

➔ 결론적으로 MK는 동적으로 생성된다는 것.

➔ NotesDeviceInfo.xml은 얻을 수 없음. (결론에서 설명)

Samsung Notes

Samsung Notes 정리:

복호화 대상 – 메모 내용

복호화 알고리즘 – AES256-CBC-DECRYPT(C, MK, IV)

복호화 대상 – 암호화된 비밀번호 해시 값 / 암호화된 솔트 값 (두 개를 이용하여 사용자 입력 비밀번호와 대조 후 검증)

복호화 알고리즘 – RSA/ECB/OAEPWithSHA-256AndMGF1Padding(C,MK)

// 여기서 MK는 Android KeyStore에서 관리하는 개인 키. 위랑 다름.

결론

스마트폰 메모 애플리케이션에 중요한 정보가 담겨있을 수 있으므로,
이러한 분석은 수사관들이 암호화된 데이터를 쉽게 복호화할 수 있도록 도움을 줌.
(많은 사람들이 사용하고 있는 메모 애플리케이션의 경우 자체적으로 접근 제어 및 데이터 암호화 같은 보안 기능이 구현되어 있기 때문임.)

ClevNote – 안드로이드 백업을 사용하여 모든 앱 데이터 추출 가능
Samsung Notes – 삼성 스마트 스위치를 사용하여 일부 앱 데이터 추출 가능
"" NotesDeviceInfo.xml "" 파일 데이터는 추출할 수 없으므로 MK 값을 얻어낼 수 없고, 결론적으로 암호화된 데이터(메모)를 복호화하기 어려움.

=> 이처럼 안드로이드 백업을 사용하지 않는, 사용할 수 없는 애플리케이션에 대한 연구가 진행되어야할 것임.

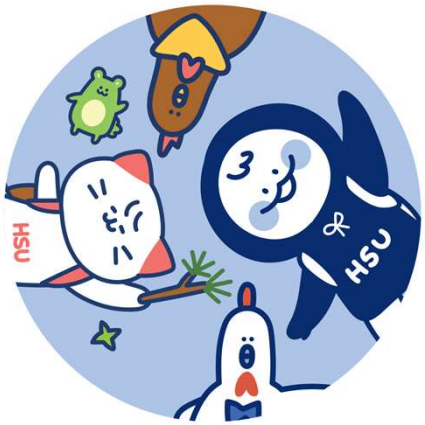
현재 연구 진행중인 부분

컬러노트를 대상으로 보안 기능을 분석하고 있음. (JEB 디컴파일러 사용, 필요하다면 IDA pro를 사용하여 동적 분석도 해볼 예정임)

오늘 소개한 논문은 2020년에 나온 것이므로, 4년간 ClevNote와 Samsung Notes의 업데이트가 있었을 것임.

따라서 현재 분석하고 있는 컬러노트를 중심으로 ClevNote와 Samsung Notes 최신 버전을 함께 분석하여 비교(?)해보는 것도 좋다고 생각함.

(논문으로 써볼 수도 있지 않을까요?... 부족하겠지만...)



감사합니다