301 – Find hidden information

Team Information

Team Name: kimbabasaksaksak

Team Member: Jaeheon Kim, Donghyun Kim, Soyoung Yoo, Minhee Lee

Email Address : uaaoong@gmail.com

Instructions

Description The police obtained information that the suspect was attempting to leak confidential information by hiding it on his personal smartphone. During the search and seizure process, no traces of classified information could be found. The police asked the suspect where he hid the confidential information on his smartphone, but he said he didn't know. It is known that the suspect has been interested in Android app development. Find confidential information hidden by the suspect.

Target	Hash (MD5)
APKS.zip	7d5de865c82cb5d08a39ae5b523904cf

Questions

Please solve all problems based on UTC+9 time zone.

1. What is the signature information of the APK file where confidential information is hidden? (60 points)

   (MD5, SHA1, SHA256 must all be obtained. 20 points each)

2. What is the confidential information decryption algorithm? (90 points)
3. What is the decrypted plaintext of encrypted confidential information? (150 points)

Teams must:

- Describe step-by-step processes for generating your solution.
- Specify any tools used for this problem.

Tools used:

Name:	jadx	Publisher:	skylot
Version:	1.4.7		
URL:	https://github.com/skylot/jadx/releases		

Name:	meld	Publisher:	meld
Version:	3.22.0		
URL:	https://meld.app/		

Name:	apktool.jar	Publisher:	apktool
Version:	2.7.0		
URL:	https://ibotpeaches.github.io/Apktool/		

Name:	JEB	Publisher:	PNF Software
Version:	JEB Demo 4.30.0.202304130349		
URL:	https://www.pnfsoftware.com/		

Name:	CyberChef	Publisher:	CyberChef
Version:	10.4.0		
URL:	https://gchq.github.io/CyberChef/		

Step-by-step methodology:

Q1. What is the signature information of the APK file where confidential information is hidden? (60 points)

(MD5, SHA1, SHA256 must all be obtained. 20 points each)

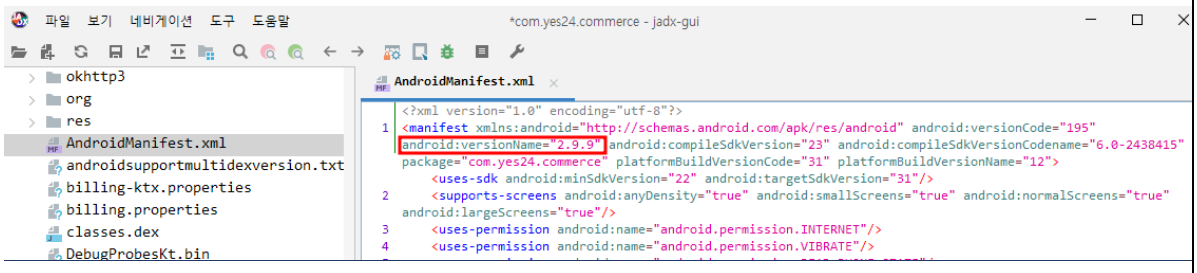
지문에서 제시한 정보를 통해 피의자가 안드로이드 앱 내부에 기밀 정보를 숨긴 후 리패키징을 수행하였을 것으로 추정했다. 리패키징 시 사이닝 과정이 필요하므로 앱 서명에 사용되는 도구인 Jarsigner를 사용하여 전체 APK 파일에 대한 서명과 인증서 정보를 출력하는 기능의 스크립트를 제작하여 주어진 앱 리스트의 사이닝 정보를 분석하였다.

Signer.bat

```
@echo off
for %%f in (*.apk) do (
    echo Verifying %%f...
    jarsigner -verify -verbose -certs "%%f" > "%%~nf.txt"
)
echo Verification complete.
pause
```

사이닝 정보 분석 결과, 50개의 앱 리스트 중 "com.yes24.commerce.apk" 앱만 유일하게 인증서 유효 시작 시간이 2023년인 것을 확인할 수 있었다. 해당 앱이 변조되었다는 것을 확인하기 위해 안드로이드 디컴파일 도구인 JADX로 해당 앱의 AndroidManifest.xml 파일 내용을 확인하여 앱의 버전 정보(2.9.9)를 확인하였다.

com.yes24.commerce.apk 파일의 버전 정보(2.9.9)



동일한 패키지 이름과 버전의 앱 파일을 다운로드하여 사이닝 정보를 비교해본 결과, 실제 배포중인 앱과 문제로 주어진 앱이 서로 다른 시그니처 정보를 가지고 있는 것을 확인하였다.

문제에서 주어진 APK 파일의 Signature 정보

```
APK signature verification result:

서명 검증 성공
유효한 APK 서명 v1을(를) 찾았습니다.

    서명자 DEVELOPE.RSA (META-INF/DEVELOPE.SF)

        유형: X.509
        버전: 3
        시리얼 번호: 0x42142261
        소유자: CN=developer, OU=yes24, O=yes24, L=seoul, ST=seoul, C=82
        유효 시작 시각: Tue Apr 25 10:55:40 KST 2023
        유효 종료 시각: Wed Apr 24 10:55:40 KST 2024

        공개키 타입: RSA
        지수: 65537
        모듈러스 크기 (비트): 1024
        모듈러스: 113317259106582802988180884935176398031891288852006382208768441166871546628508783976466635096088480791

        서명 유형: MD5withRSA
        서명 OID: 1.2.840.113549.1.1.4

        MD5 지문: 1F 81 4B 7C 3E A1 80 E2 29 31 C6 60 1F D7 75 1F
        SHA-1 지문: 9F 3E D0 61 54 83 64 B3 C9 4B 84 DD 2D 67 55 59 AF 34 49 27
        SHA-256 지문: 98 2B 7D 04 92 CD 43 42 58 D3 44 91 F9 F3 3C E6 32 2C 24 AE D6 72 7B B0 CC 8E 0D AB AF EE 53 F3
```

정상 APK 파일의 Signature 정보

```
APK signature verification result:

서명 검증 성공
유효한 APK 서명 v1을(를) 찾았습니다.

    서명자 CERT.RSA (META-INF/CERT.SF)

        유형: X.509
        버전: 3
        시리얼 번호: 0x4d192b2c
        소유자: CN=Woo Young Seo, O=annex institute, L=Seoul, ST=Yeongdeungpo-gu, C=KR
        유효 시작 시각: Tue Dec 28 09:11:24 KST 2010
        유효 종료 시각: Wed Dec 15 09:11:24 KST 2060

        공개키 타입: RSA
        지수: 65537
        모듈러스 크기 (비트): 1024
        모듈러스: 14818427526717213203567799645993022606603761164612968503779097722695436195495937725357431931291427083385

        서명 유형: SHA1withRSA
        서명 OID: 1.2.840.113549.1.1.5

        MD5 지문: 7E E8 3E EA 72 61 A6 5A FE 23 45 9B 8C 5F 5B D9
        SHA-1 지문: 79 D3 AC 94 C0 65 FC 7D 33 DD A8 B3 27 3B FB C0 08 75 7B D0
        SHA-256 지문: F9 E9 F9 4F B1 9B 79 3C F3 50 A5 47 9B 38 77 8A 1C 29 4C 13 CF 23 CA F9 83 B9 BB E4 5C 1D DB 81
```

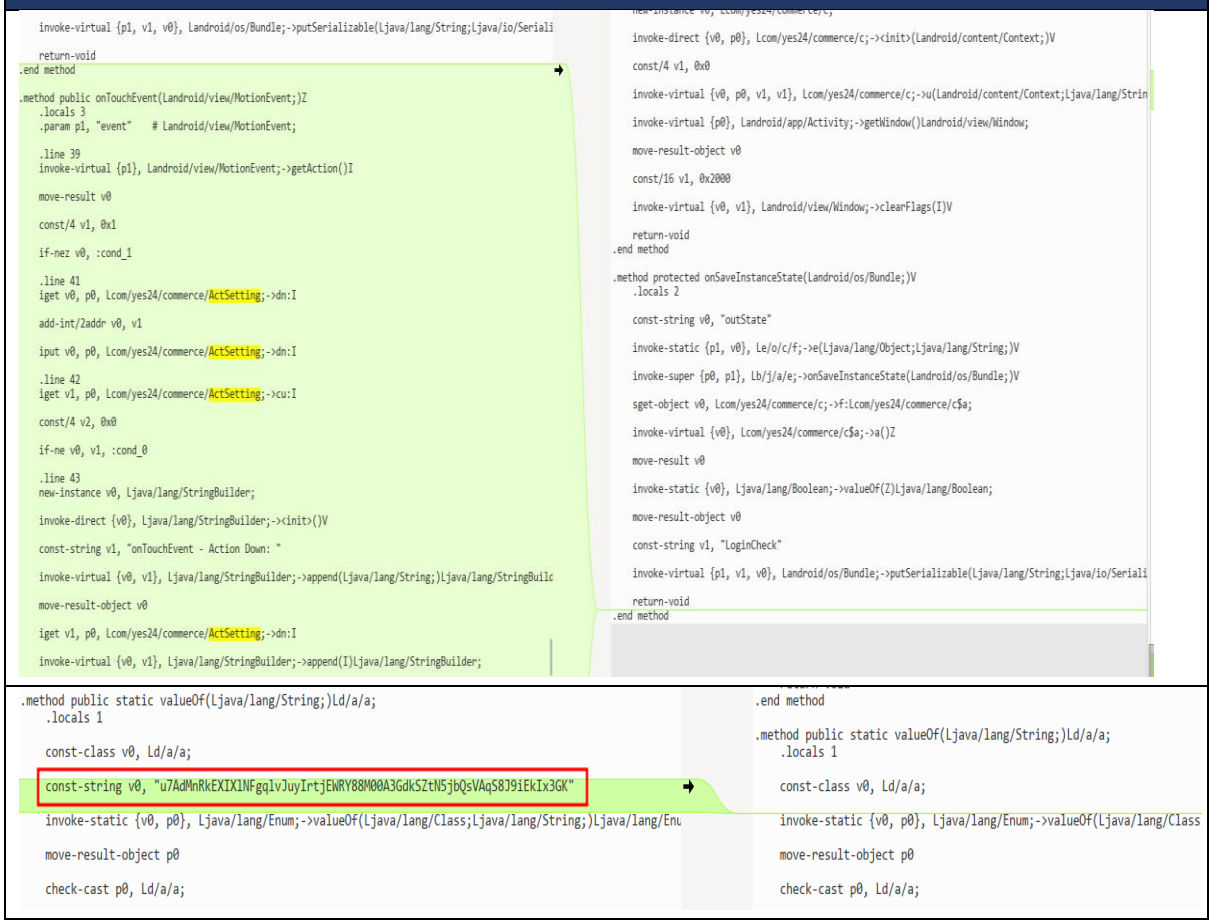
따라서 기밀 정보가 숨겨진 APK 파일의 signature 정보는 아래와 같다.

- MD5 : 1F 81 4B 7C 3E A1 80 E2 29 31 C6 60 1F D7 75 1F
- SHA1 : 9F 3E D0 61 54 83 64 B3 C9 4B 84 DD 2D 67 55 59 AF 34 49 27
- SHA256 : 98 2B 7D 04 92 CD 43 42 58 D3 44 91 F9 F3 3C E6 32 2C 24 AE D6 72 7B B0 CC 8E 0D AB AF EE 53 F3

Q2. What is the confidential information decryption algorithm? (90 points)

변조된 APK 파일과 정상 APK 파일을 안드로이드 앱 디컴파일 및 패키징 도구인 apktool.jar를 사용하여 앱의 리소스를 추출하였다. 이후 디컴파일된 코드를 코드 비교 도구인 Meld를 사용하여 정상 APK 파일에서 추가된 코드를 식별하였다. 코드를 비교 분석한 결과, ActSetting 클래스에 코드가 추가된 것과 암호화된 기밀 정보로 보이는 문자열을 확인하였다.

변조된 APK에 추가된 코드



Jadx를 통해 코드가 추가된 ActSetting 클래스를 분석한 결과, onTouchEvent – ACTION_DOWN 이벤트가 20번 발생했을 때 dd 메소드를 호출하며, 암호화 키 ky와 암호화된 기밀 정보를 인자로 하여 "c.c.d.u.b.dbk" 메소드를 호출한다.

- 암호화 키 : ActSettingCreate
- 암호화된 기밀 정보 : u7AdMnRkEXIXlNFgqlvJuyIrtjEWRY88M00A3GdkSZtN5jbQsVAqS8J9iEkIx3GK

ActSetting 클래스 주요 코드

```
/* loaded from: classes.dex */
public final class ActSetting extends r implements View.OnClickListener, com.yes24.commerce.control.d {
    private String u;
    public c v;
    public com.yes24.commerce.control.r w;
    private com.yes24.commerce.b0.i x;
    private String ky = "ActSettingCreate";
    private int dn = 0;
    private int cu = 20;

    @Override // android.app.Activity
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction() == 0) {
            int i2 = this.dn + 1;
            this.dn = i2;
            if (i2 == this.cu) {
                Log.e("developer", "onTouchEvent - Action Down: " + this.dn);
                this.dn = 0;
                Toast.makeText(this, dd(BuildConfig.FLAVOR), 0).show();
            }
            return false;
        }
        return true;
    }

    public String dd(String it) {
        String rv = BuildConfig.FLAVOR;
        Log.e("developer", "dd");
        try {
            String dt = c.c.d.u.b.dbk(this.ky, it);
            rv = dt;
            Log.e("developer", "t: " + rv);
            return rv;
        } catch (Exception e2) {
            e2.printStackTrace();
            return rv;
        }
    }
```

암호화된 데이터는 Base64로 디코딩 후 “ActSettingCreate”를 UTF8 포맷의 키 값으로, iv = {1, 2, 3, 4, 5, 6, 7, 8, 9, 16, 17, 18, 19, 20, 21, 22} 를 HEX 포맷으로 변환한 “01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16” 값을 IV 값으로 하여 AES-128 CBC모드 (PKCS5Padding) 암호화 알고리즘을 통해 복호화된다.

복호화 관련 코드

b

```
package c.c.d.u;

import android.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

/* loaded from: classes.dex */
28 public class b {
    public static byte[] iv = {1, 2, 3, 4, 5, 6, 7, 8, 9, 16, 17, 18, 19, 20, 21, 22};

29     public static String dbk(String k, String t) throws Exception {
30         return dd(k.getBytes(), Base64.decode(t, 0));
    }

34     private static String dd(byte[] k, byte[] t) throws Exception {
35         SecretKeySpec secretKeySpec = new SecretKeySpec(k, "AES");
36         Cipher c2 = Cipher.getInstance("AES/CBC/PKCS5Padding");
37         c2.init(2, secretKeySpec, new IvParameterSpec(iv));
38         byte[] sk = c2.doFinal(t);
39         return new String(sk);
    }
}
```

Q3. What is the decrypted plaintext of encrypted confidential information? (150 points)

2번 문제를 통해 Base64 인코딩 및 AES 암호화된 데이터 "u7AdMnRkEXIXlNFgqlvJuyIrtjEWRY88M00A3GdkSZtN5jbQsVAqS8J9iEkIx3GK", 16바이트의 AES 키 "ActSettingCreate", 16바이트의 헥스 값으로 변환한 IV "01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16" 값을 확보하였다. 복호화 정보를 통해 CyberChef를 사용하여 암호화된 데이터를 복호화한 결과 아래와 같은 평문을 획득할 수 있었다.

information was stored in my private cloud

복호화한 기밀 정보