국방모바일보안 SECURITY AND PRIVACY REVIEW

By Ovi Liber

한국어 번역본

APRIL 26th, 2023

디지털안전연구소
INTERLAB

Contents

인터랩 소개

인터랩은 시민사회 조직을 대상으로 디지털 안전 컨설팅, 교육, 침해 대응, 교육, 위협 진단 지원을 제공하는 비영리 단체입니다.

인터랩의 목표는 시민 사회 단체들이 디지털 공간에서 안전하게 활동하면서 소셜 임팩트와 세상을 변화하는 일을 지속적으로 실현시킬 수 있는 세상을 만드는 것입니다.

인터랩은 시민사회 조직들이 정보를 안전하게 지키는데 많은 어려움이 있다는 사실을 알고 있습니다. 저희는 시민사회 조직들이 지속가능성과 회복 탄력성을 지켜낼 수 있게끔 헌신하겠습니다. 인터랩은 복잡한 디지털 공간에서 자신 있게 활동할 수 있게 도와드리며, 시민사회 조직들이 세상에 긍정적인 변화를 만든다는 핵심 목표만을 집중할 수 있도록 도와드립니다. 인터랩은 디지털 보안에 대한 전문적인 지원을 통해 시민 사회 단체가 궁극적인 목표를 달성하며 사회를 발전시킬 수 있게 돕는다고 믿습니다.

Find us at: interlab.or.kr

Reach us at: contact@interlab.or.kr

1_ KEY FINDINGS

- 국방모바일보안 개발자는 이 애플리케이션이 연락처, 비디오, 사진, SMS 데이터 등의 개인 정보를 수집하지 않고 있다고 명시하고 있습니다. 그러나 분석 결과 애플리케이션이 위치 정보 및 이에 해당하는 시간정보를 포함한 **민감한 개인정보를 기록하고 있음**을 확인하였습니다. 이러한 분석 결과는 민감한 데이터의 보호에 실패하는 것뿐만 아니라, 사용자 및 국방부 전체의 프라이버시 및 보안 위협이 애플리케이션으로부터 발생함을 의미합니다.
- 특히 간부 및 외부인 전용 애플리케이션 버전들의 경우, **공격자가 아무런 권한의 요청없이 개인정보를 추출할 수 있는 2 개의 취약점들을 발견하였습니다.** 만약 공격자가 기기에 접근이 가능할 시, GPS 주소와 관련 시간정보 등이 기록되어 있는 로그파일을 얼마든지 빼 갈 수 있는 상황임을 확인하였습니다.
- 분석결과 애플리케이션은 또한 필요 없는 소스코드와 기능들을 남겨둠으로써, 추가적인 프라이버시 및 보안 결함들을 내포하고 있음을 확인하였습니다

2_ INTRODUCTION

국방모바일보안은 국방부에서 개발과 배포를 담당하고 있는 스마트폰 앱입니다. 모든 병사, 직원 및 외부 출입자들은 보안 상 의무적으로 설치해야 하는 앱입니다. 앱은 군사 기밀 유출을 방지하기 위해 위치 기반 카메라 차단 기능을 제공합니다. 앱의 대표적인 기능은 군사 기밀 유출 방지를 위한 휴대전화의 카메라를 제어하는 기능입니다. 휴대전화를 군사시설 내의 NFC 기기에 접촉할 시 카메라를 비활성화합니다. 카메라 허용 기능은 위병소에 위치한 블루투스 비콘을 인식해 카메라 기능을 다시 사용할 수 있도록 허가합니다. 비콘으로 카메라를 활성화하지 못했다면, 앱에 사전 등록된 공공기관 등의 주소로 찾아가 위치 기반으로 차단을 해제해야 합니다.

이 앱은 설치 과정에서 지나치게 많은 권한을 요청합니다. 국방부는 이 앱이 사용자의 개인정보를 수집하거나 이용하지 않으며, 앱에 부여된 권한은 정보통신망 이용촉진 및 정보보호 등에 관한 법률 제 22 조의 2 를 준수하기 위함이라 주장합니다. 더 나아가, 국방부는 이 앱이 무서버 기반으로 동작한다고 설명합니다.

2019 년 11 월 26 일 출시 후 동년 12 월부터 이듬해 3 월까지 앱은 구글 플레이 스토어 기준 500,000 만 회 이상 설치되었으며, 평점은 5 점 만점 중 1 점 [1]. 에 수렴하고 있습니다. 35 억의 예산을 들여 개발된 앱은 많은 논란을 야기했고, 수많은 악평이 여러 앱 스토어에서 쏟아지고 있습니다. [2]. 앱 스토어의 리뷰 페이지에서는 사용자들로부터 앱의 오작동으로 인한 불편 호소와 보안 및 프라이버시 침해에 대한 우려와 의문이 제기되고 있습니다 [3].

국방모바일보안 앱은 MarkAny AegisSAFER (https://www.samsungknox.com/en/partner-solutions/markany-aegissafer)에서 개발된 정황을 여럿 보이고 있습니다. 개발 주체가 국방부인지 해당 단체인지 명확히 구분되지 않았고, 보고서의 내용이 국방부보다는 해당 단체에서 개발한 코드와 기능에 관한 내용을 담고 있을 가능성도 있습니다.

인터랩은 앱에 보안 문제나 프라이버시 침해 가능성이 있는지 검토하고, 이후 사용자들에게 앱의 작동 방식을 명확히 전달하기 위해 안드로이드 버전의 앱을 분석하기로 결정했습니다. 저희는 몇 달 동안 리버스 엔지니어링을 하며 정적 및 동적방식으로 연구했습니다.

다음의 안드로이드 앱들을 대상으로 분석을 진행했습니다:

- *국방모바일보안(병사)* - kr.go.mnd.mmsa: https://play.google.com/store/apps/details?id=kr.go.mnd.mmsa
- *국방모바일보안(직원)* - kr.go.mnd.mmsa.of: https://play.google.com/store/apps/details?id=kr.go.mnd.mmsa.of
- *국방모바일보안(외부인)* - kr.go.mnd.mmsa.vt https://play.google.com/store/apps/details?id=kr.go.mnd.mmsa.vt

연구 결과 앱의 세 가지 버전 모두 날짜, 시간과 대략적인 위치정보에 대한 상세한 로그를 남기고 있다는 사실을 알 수 있었습니다. 로그 기능은 앱이 삭제되기 전까지 기록되며 사용자들의 사생활의 자유를 중대하게 침해하고 있습니다. 위치정보의 보호 및 이용 등에 관한 법률(이하 '위치정보법')에 따르면, 위치정보는 다른 정보와 용이하게 결합하여 특정 개인의 위치를 알 수 있는 정보에 속합니다. 앱의 개발자들은 앱이 개인정보를 수집하지 않는다는 안내로 사용자들을 오도하고

있습니다. 나아가 앱이 위치정보를 수집, 저장, 이용하는 과정을 분석하며 수 차례의 위치정보법 위반 사례를 발견할 수 있었습니다.

그밖에도 두 버전의 앱(직원, 외부인)에서 공격자가 루트 권한 및 어떠한 액세스 권한이 없어도 대략적인 GPS 위치 정보를 포함한 앱의 모든 로그 정보를 탈취할 수 있는 보안 취약점을 발견했습니다. 인터랩은 '책임있는 공개'(Responsible Disclosure) 절차에 따라 국방부에 2023 년 3 월 10 일에 이러한 취약점 및 해결 방안을 고지 했습니다. 어떻게 공격자가 영향을 끼칠 수 있는지와 해결 방안을 상세 설명했습니다. 이 리포트를 공개하는 시점 까지도 국방부는 인터랩에 연락하지 않았으며, 이 취약점은 최신 버전 앱에도 포함됩니다

이 보고서는 다음 장으로 구성됩니다:

Methodology

이 장은 인터랩이 어떻게 안드로이드 애플리케이션을 분석했는지에 대한 방법을 앱 버전과 함께 설명합니다. 이를 통해 우리의 분석을 재현할 수 있도록 합니다.

Privacy

이 장은 세가지 앱 버전 모두(병사, 직원, 외부인)를 개인정보 보호와 관련된 문제들의 분석을 다룹니다. 서버 상호 작용을 이해하기 위한 네트워크 트래픽 분석을 포함합니다. 인터랩은 해당 애플리케이션이 실제로 무서버 기반이며 어떠한 외부 네트워크 트래픽 수신 능력이 없다는 것을 확인했습니다. 개인 정보 수집 및 저장 또한 살펴보았으며, 유저가 기기의 카메라를 공공 장소에서 잠금 해제할 때만 대략적인 GPS 위지 정보를 수집한다는 것을 확인했습니다. 애플리케이션 코드에 대한 정적 분석 또한 진행하면서 원격으로 장치의 기능을 비활성화 할 수 있는 기능 또한 확인했습니다.

Security

이 장에서는 애플리케이션의 보안 분석을 다룹니다. 이 장은 공격자가 사용할 수 있으며 사용자에게 위협이 될 수 있는 보안 취약점 분석을 포함합니다. 두가지 버전의 앱(직원용, 외부인)에서 공격자가 대략적인 GPS 위치를 포함한 애플리케이션의 로그 데이터를 추출할 수 있게 하는 현저한 보안 취약점을 발견했습니다.

3_ METHODOLOGY

이 장에서는 앱들을 분석한 방식의 개요를 살펴보도록 하겠습니다.

3.1_ Limitations

앱들을 동적으로 분석하는 과정에서 군 기지 내의 NFC 또는 비콘 장치에 접근할 수 없어 원격으로 앱을 분석해야 했습니다. 이 때문에, 분석을 효율적으로 진행하기 위해 안드로이드 에뮬레이터를 사용했습니다. 분석 과정에서 사용한 에뮬레이터에는 블루투스와 NFC 기능이 없으며, 런타임 후킹으로 여러 사용자 검증 절차를 우회하여 분석 결과를 얻을 수 있었다는 점이 감안되어야 합니다. 다만, 블루투스와 NFC 를 통한 잠금과 해제 기능이 데이터 수집 또는 취약점 분석에 미치는 영향은 없었습니다. 앱의 개인정보 수집과 보안 관련 문제를 연구하며 에뮬레이터 환경의 제한이 문제가 되지는 않았습니다. 연구 과정에서 있었던 제한사항들에 대해서 더 알고 싶으시다면 2 페이지의 안내사항도 참고하시길 바랍니다. 저희는 다방면으로 많은 결과가 도출되기를 바라며, 업계의 후속 연구를 환영합니다.

3.2_ Process

다음과 같은 대략적인 절차로 애플리케이션 정적 및 동적 분석을 진행했습니다:

1. 앱이 정상적으로 작동할 때 앱의 모든 기능을 테스트하며 발생한 네트워크 트래픽을 수집했습니다.
    a. 외부 서버와의 통신으로 해석될 수 있는 모든 네트워크 트래픽을 파악하였습니다
2. 동적으로 클래스와 메서드의 값을 후킹해 AES 키 등을 얻어내고, 저장된 파일과 인코딩된 데이터를 복호화 했습니다.
3. 동적으로 액티비티와 주요 클래스, 메서드를 후킹해 인자, 백트레이스, 반환값을 확인하고 수정했습니다. 저희는 앱 전체를 분석하기 위해서, 주로 참/거짓 반환값과 문자열값을 임의로 수정하며 앱의 특정 액티비티들을 우회했습니다.
4. 소스 코드를 정적으로 역분석해 앱에서 데이터가 수집되고 사용되는 방식을 연구했습니다.

3.3_ Application versions

- 국방모바일보안(병사) App 2.1.05
- 국방모바일보안(외부인) 2.1.18
- 국방모바일보안(직원) 2.1.25

3.4_ Testing Environment

- Android 스튜디오 가상 에뮬레이터 기기
    - Android 8.0 Google API
    - Android-26 x86 시스템 이미지
- System locale 을 한국어(ko_KR)으로 설정
- GPS 위치를 서울 중앙으로 설정
- gsm.operator.iso-country 를 KR 으로 설정

3.5_ Dynamic Analysis & Traffic Interception Tools Setup

- *libfrida-gadget* 을 포함하도록 Objection(1.11.0 버전) 및 APKTool(2.7.0 버전)으로 재컴파일된 앱
    - Gadget 버전: *Frida-gadget-16.0.9-android-x86*
- 트래픽 검사 도구:
    - Burp Suite Community Edition
    - 시스템 공간에 Burp CA (Certificate Authority) 인증서 설치
    - 앱 SSL Unpinning
    - Android 스튜디오 수동 프록시 설정
- Frida Javascript 스크립팅 기능을 이용해 기기 제조사를 삼성으로 설정
    -

```
Java.perform(function () {

    var buildProps = Java.use('android.os.Build');
    console.log(JSON.stringify(buildProps.MANUFACTURER))
    buildProps.MANUFACTURER.value = "Samsung";
    console.log(JSON.stringify(buildProps.MANUFACTURER))
});
```

- 루트 권한 탐지, 무결성 탐지 및 기타 검증을 우회하기 위해, 저희는 Frida 런타임 툴 킷의 여러 후킹 스크립트를 사용했습니다. 앱의 보안과 국방부 직원 분들을 고려해, 세부적인 내용은 보고서에 담지 않기로 결정했습니다.
- Drozer 보안 테스팅 프레임워크
    - 2.4.4 버전 Docker 빌드: https://hub.docker.com/r/fsecurelabs/drozer

3.6_ Static Analysis Tools Setup

- APKTool (Version 2.7.0)
- Jadx (Version 1.4.5)
- Dex2jar (Version 2.1)

4_ FUNCTIONALITY, PRIVACY & DATA COLLECTION

세 버전의 앱 모두 특정 기능을 제외하면 유사한 기능을 가집니다. 예를 들면, 직원과 방문자용 버전은 오직 한 가지 기능을 제외하면 동일한 앱입니다. 보고서에 나오는 "앱"은 세 가지 버전을 통칭합니다. 앱의 한 가지 버전만을 특정할 때에는 해당 버전을 따로 언급하도록 하겠습니다.

4.1_ Network connectivity

분석 중 앱이 무서버 기반으로 작동한다는 사실을 확인할 수 있었습니다. 서버와 테스트 기기간에 통신을 주고받는 일이 발견된 적은 없으며, 디컴파일하여 분석한 소스 코드에서도 서버와의 활성화된 통신은 발견되지 않았습니다.

병사 버전의 앱에서는 AegisSAFER 가 소유의 엔드포인트와 하나의 통신이 있었지만 확인 결과 단순히 버전 관련 정보를 받아오는 통신임을 알 수 있었습니다. 이 통신은 앱의 *직원* 버전과 *관리자* 버전에서는 발견되지 않았습니다. 이외에 발견된 유일한 통신으로는 사용자가 설명서를 다운받을 때 이루어지는 AegisSAFER 엔드포인트와의 통신이 있었습니다.

```java
package kr.go.mnd.mmsa;

import java.util.concurrent.Callable;
import org.jsoup.Jsoup;

/* compiled from: lambda */
/* renamed from: kr.go.mnd.mmsa.id */
/* loaded from: classes.dex */
public final /* synthetic */ class jsoupConnectVisitor implements Callable {
    public static final /* synthetic */ jsoupConnectVisitor a = new jsoupConnectVisitor();

    private /* synthetic */ jsoupConnectVisitor() {
    }

    @Override // java.util.concurrent.Callable
    public final Object call() {
        String node;
        node = Jsoup.connect("https://[redacted]:38448/visitor/resources/MMSA/version.html/").timeout(30000).get().getAllElements().get(4).childNodes().get(0).toString();
        return node;
    }
}
```

자료 1 AegisSAFER 엔트포인트와 통신하는 클래스

앱에서 사용자의 상호작용 없이 이루어진 네트워크 통신은 상술한 버전 확인용 통신이 유일합니다. 그러나 앱의 소스 코드에는 외부 서버와 통신할 수 있는 기능이 포함되어 있었습니다. 저희가 테스트한 버전에는 해당 기능이 구현되어 있지 않았지만, 이전 버전에서 사용되었을 가능성을 부정할 수는 없습니다. (이후 버전들 역시 마찬가지입니다).

이 앱은 디스크에 저장된 설정 파일과 함께 컴파일 됩니다. 이 설정 파일은 자료 1 과 같이 앱별 저장소에 저장됩니다. 이 파일은 AES/CBC/PKCS5Padding 방식으로 암호화되고 Base64 인코딩을 거친 JSON 파일입니다.

```
generic_x86:/data/user/0/kr.go.mnd.mmsa.vt/files/MobileSticker/task # ls -al
total 16
drwx------ 2 u0_a92 u0_a92 4096 2023-03-12 08:02 .
drwx------ 4 u0_a92 u0_a92 4096 2023-03-12 07:49 ..
-rw------- 1 u0_a92 u0_a92 7404 2023-03-12 08:02 ConfigMndMDM.json
```

자료 2 설정 파일 위치

세 가지 버전 앱의 설정 파일 모두 앱 버전 구분과 관련된 여러 색인이 포함되어 있었습니다. 색인에는 앱이 동작하는 방식을 정의하는 정보가 담겨 있었습니다. 자료 3 에서 예시를 확인하실 수 있습니다. 서버 주소와 연락처 정보는 국방부 자산의 보호를 위해 생략하였습니다.


```
},
{
        "index":"9005",
        "code":"mndmdm1",
        "name":"국방부",
        "description":"군장병 휴대전화 보안",
        "licenseExpired":"2020-12-31",
        "deviceEnv":
        {
                "forgery":"1", "debugger":"0", "validation":"1", "update":"0", "network":"0", "autoTime":"0", "deletePopup":"1"
        },
        "serverUrl":
        {
                "check":"ht                    :39440/enter-logs/insert",
                "debug":"",
                "logo":"http:/                 :38088/visitor/resources/MobileSticker/logo/mnd.png"
        },
        "checkIn":
        {
                "qr" : "1", "self" : "1", "nfc": "1"
        },
        "checkOut":
        {
                "sound" : "1", "gps": "1", "nfc": "1", "beacon": "1", "sms" : "1"
        },
        "policy":
        {
                "camera":"1", "wifi":"0", "bluetooth":"0", "tethering":"0", "mic":"1", "usb":"0", "sdcard":"0", "nfc":"0", "gps":"0", "res
et":"1"
        },
        "infoVoc":
        {
                "tel":"          ", "email":"          @gmail.com"
        },
        "infoGps":
        [
                { "name" : "국방부", "type":"out", "mode":"whitelist", "latitude":"38.563074", "longitude":"128.003582", "radius":"150" }
        ]
},
```


자료 3 복호화된 설정 파일 ConfigMndMDM.json

자료 3 에서 URI "/enter-logs/insert"가 포함된 URL 이 담겨있는 "check" 값이 "serverURL" 안에 있음을 확인할 수 있습니다. 분석 결과 저희가 사용한 버전에서는 이 URL 이 사용되지 않았습니다.

이 설정 파일은 AegisSAFER 측에서 앱이 포함할 기능의 구성을 선택하기 위해 구현한 것으로 보입니다. 사용될 기능들은 앱을 등록할 때 사용된 인증 코드를 바탕으로 결정됩니다. 구체적으로 어떠한 인증 코드가 누구에게 제공되는지는

확인할 수 없었습니다. 이 인증 코드는 국방부에서 직원들에게 직접 제공하는 것으로, 보안 유지를 위해 공개하지 않기로 결정했습니다. 이것의 구현은 자료 4 의 소스 코드에서 확인하실 수 있습니다.

```
352     public void m1351m() {
353         CompanyInfoN companyInfoN;
354         String m421s = this.f2027a.m421s("CompanyInfo", "company_info_auth_code", "0");
355         try {
356             if (getPackageName().equals("kr.go.mnd.mmsa.of")) {
357                 if (!ActivityC0302de.m2412j(m421s, 0).equals("")) {
358                     if (!m421s.equals("EaVdqsUlA5/1KGzYcv+Rtw==") && !m421s.equals("12GxUc7S/QvMRHfXU50mCg==")) {
359                         if (!m421s.equals("GsbJVE7AI0g7NrcEVtnmlg==") && !m421s.equals("NgSVE0zVG7VviB5589xvkw==")) {
360                             if (!m421s.equals("bUWtIPY5QS0DHtEPYg5Tfg==") && !m421s.equals("f054uIAhw5BBxisZ5KkxTw==")) {
361                                 if (!m421s.equals("ZuYm+QE9Dan5nyc6VfI+Yw==") && !m421s.equals("/bGIcr35yLhvA4BkFNSoJQ==")) {
362                                     if (m421s.equals("O1ATswTHCiKTBtAzP4SdRg==") || m421s.equals("T/ybotmUBSzrRPy5OoeOng==")) {
363                                     }
364                                 }
365                                 companyInfoN = C0950zd.parseConfigMain(this, "mndmdm2");
366                             }
367                             companyInfoN = C0950zd.parseConfigMain(this, "mndmdm3");
368                         }
369                         companyInfoN = C0950zd.parseConfigMain(this, "mndmdm1");
370                     }
371                     companyInfoN = C0950zd.parseConfigMain(this, "mndmdm");
372                 }
373                 companyInfoN = null;
374             } else {
375                 if (getPackageName().equals("kr.go.mnd.mmsa.vt") && !ActivityC0302de.m2412j(m421s, 0).equals("")) {
376                     if (!m421s.equals("Zn8bhFsk/+xh/pThclDRXg==") && !m421s.equals("2P9633939Rqvl+tNgq+QTg==")) {
377                         if (!m421s.equals("yJrhaTr7Ey7OTWKQh26l7A==") && !m421s.equals("ENpcaihT4KVLZWwpf9GfDA==")) {
378                             if (!m421s.equals("fH81vfMNrl/wo7O2/bSQzA==") && !m421s.equals("p1mfy3XaVikRuZiH+8Z9iA==")) {
379                                 if (!m421s.equals("JuEWinamPu2gsIWNhGK33w==") && !m421s.equals("YKRk2eWRtzOzlLzZkKrLDA==")) {
380                                     if (m421s.equals("HGYlxwPI78ekBDQrlY6Lxw==") || m421s.equals("0UG/TeZglIbj36RQsyKqEw==")) {
381                                         if (ActivityC0302de.m2415g(this)) {
382                                         }
383                                     }
384                                 }
385                                 companyInfoN = C0950zd.parseConfigMain(this, "mndmdm2");
386                             }
387                             companyInfoN = C0950zd.parseConfigMain(this, "mndmdm3");
388                         }
389                         companyInfoN = C0950zd.parseConfigMain(this, "mndmdm1");
390                     }
391                     companyInfoN = C0950zd.parseConfigMain(this, "mndmdm");
392                 }
393                 companyInfoN = null;
```

자료 4 AES 암호화와 base64 인코딩을 거친 인증 코드를 토대로 설정 파일을 결정하는 소스 코드

아래의 표 1 에서 인증 코드와 관련된 기능 설정을 확인하실 수 있습니다:

		Mndmdm	**mndmdm1**	**mndmdm2**	**mndmdm3**	**mndmdm4**
Device Environment	Forgery	1	1	1	1	1
	Debugger	0	0	0	0	0
	Validation	1	1	1	1	1
	Update	0	0	0	0	0
	Network	0	0	0	0	0
	AutoTime	0	0	0	0	0
	DeletePopup	1	1	1	1	1
Check In Functionality	QR	1	1	1	1	1
	Self	1	1	1	1	1
	NFC	1	1	1	1	1
Check Out Functionality	Sound	1	1	1	1	1
	GPS	1	1	1	1	1
	NFC	1	1	1	1	1
	Beacon	1	1	1	1	1
	SMS	1	1	1	1	1
Access Policy	Camera	1	1	1	1	1
	Wifi	0	0	1	0	0
	Bluetooth	0	0	0	0	0
	Tethering	0	0	1	1	1
	Mic	0	1	1	1	1
	USB	0	0	1	1	1

	SDCard	0	0	0	0	0
	NFC	0	0	0	0	0
	GPS	0	0	0	0	0
	Reset	1	1	1	1	1
	Iris	1	1	1	1	1

Table 1 인증 코드에 기반한 설정 목록

"Device Environment", "Check In Functionality"와 "Check Out Functionality"는 인증코드에 관계없이 같은 값을 가집니다. 그러나 "Access Policy"는 인증 코드에 따라 조금씩 달라짐을 확인할 수 있습니다. 이는 국방부의 어떤 조직에 속하는지에 따라 다른 접근 정책이 적용됨을 의미합니다. Mndmdm2 설정은 유일하게 Wifi 접근 정책을 True 로 설정합니다.

4.2_ Services

앱의 각 버전마다 서비스에 약간의 차이가 있습니다. 이 장에서는 각 앱별로 나타나는 서비스들에 대해서 분석하겠습니다. 분석을 시작하며 서비스의 기능이 무엇인지, 어떤 조건에서 서비스가 시작되는지에 대해 다루는 것을 목표로 잡았습니다. 다양한 서비스가 존재했기 때문에, 각 서비스가 어떤 앱에서 발견되는지 확인했습니다. 서비스들 중 외부로 내보내진 서비스는 없었고, 앱 외부에서 서비스를 직접 만들기 위한 접근을 할 수 없었습니다. (다음 장에서 다루어질 내보내진 브로드캐스트 리시버 중 서비스를 시작할 수 있는 것들도 예외는 아닙니다).

4.2.1_ ServiceAEGIS – Soldier, Staff & External

이 서비스는 메시지 핸들러를 통해 서비스가 시작될 때 OS 메시지의 값을 확인합니다. 해당 내용은 자료 5 에서 확인하실 수 있습니다. 메시지 값에 따라, 반환된 문자열은 기기 로그 파일과 SQL 데이터베이스에 기록됩니다. 그러나, 이 값 중 대부분이 앱에서 사용되지 않는 것을 확인할 수 있었습니다. 따라서 해당 기능은 이전 버전의 잔재이거나 사용되지 않은 소스 코드로 보였습니다.

```
5     public static String a(int i) {
6         if (i != 1001) {
7             if (i != 1002) {
8                 if (i != 1004) {
9                     if (i != 1005) {
10                        if (i != 1007) {
11                            if (i != 2010) {
12                                if (i != 2012) {
13                                    switch (i) {
14                                        case 1024:
15                                            return "appDownload";
16                                        case 1025:
17                                            return "devicePower";
18                                        case 1026:
19                                            return "lockDevice";
20                                        case 1027:
21                                            return "unregisterDevice";
22                                        default:
23                                            switch (i) {
24                                                case 2001:
25                                                    return "getContentManual";
26                                                case 2002:
27                                                    return "getContentConfig";
28                                                case 2003:
29                                                    return "getContentIcon";
30                                                default:
31                                                    return "unknown";
32                                            }
33                                    }
34                                }
35                                return "CheckOut";
36                            }
37                            return "CheckIn";
38                        }
39                        return "message";
40                    }
41                    return "token";
42                }
43                return "update";
44            }
45            return "enrollment";
46        }
47        return "enrollmentrequest";
48    }
```

자료 5 Value case return for message handler

이 서비스는 오직 로컬 앱으로부터 만 인텐트를 수신하지만 (https://developer.android.com/guide/components/intents-filters), 보안 장에서 다루어질 하나의 인텐트는 앱의 외부에서 수신할 수 있습니다. 앱은 다음 인텐트 필터에 따라 필요한 동작을 시작합니다. (이에 대한 소스 코드는 자료 6 에서 확인하실 수 있습니다.):

- com.markany.aegis.[app version code].MSTICKER_SERVICE
    - 시스템 버전 및 관련 권한 확인
- android.intent.action.BOOT_COMPLETED
    - 시스템 버전 및 관련 권한 확인
- com.markany.aegis.gate.AEGIS_GATE_ACTION_AGENT_RELEASE
    - 아래에 설명될 여러 서비스를 확인한 뒤 앱을 종료하는 "destroy" 명령을 포함하도록 설정 변경.
- com.markany.aegis.AEGIS_ACTION_ADMIN_REQUEST
    - 2 가지 기능을 하는 인텐트(자료 7 에서 설명):
        - 로컬 앱 로그 파일을 기기의 storage 디렉토리로 추출하기
        - GateKeeper 비활성화 및 사용자의 애플리케이션 제거 허용

```
342    public synchronized void onHandleIntent(Intent intent) {
343        String str;
344        String action;
345        try {
346            try {
347                try {
348                    action = intent.getAction();
349                } catch (Exception e) {
350                    e = e;
351                    str = a;
352                    ae.e(str, e);
353                }
354            } catch (NullPointerException e2) {
355                e = e2;
356                str = a;
357                ae.e(str, e);
358            }
359            if (action == null) {
360                return;
361            }
362            String str2 = a;
363            ae.g(str2, "<<<<< RECEIVE INTENT: " + action);
364            if (!"com.markany.aegis.gate.AEGIS_GATE_ACTION_DEVICE_LOCK".equals(action) && !"com.markany.aegis.gate.AEGIS_GATE_ACTION_SERVICE_DEVICE_LOCK".equals(action)) {
365                if ("com.markany.aegis.vt.MSTICKER_SERVICE".equals(action)) {
366                    appcheckH(intent);
367                } else if ("android.intent.action.BOOT_COMPLETED".equals(action)) {
368                    appcheckF(intent);
369                } else if ("com.markany.aegis.gate.AEGIS_GATE_ACTION_AGENT_RELEASE".equals(action)) {
370                    destroy();
371                } else if ("com.markany.aegis.AEGIS_ACTION_ADMIN_REQUEST".equals(action)) {
372                    releaseOrExportIntent(intent);
373                }
374            }
375        } catch (Throwable th) {
376            throw th;
377        }
378    }
```

자료 6 서비스의 handleIntent 필터 소스 코드

```java
    /* renamed from: e */
    public final void releaseOrExportIntent(Intent intent) {
        String stringExtra = intent.getStringExtra("action_admin");
        Intent intent2 = new Intent();
        intent2.setAction("com.markany.aegis.AEGIS_ACTION_ADMIN_RESPONSE");
        intent2.addFlags(268435456);
        intent2.putExtra("extra_result", "1");
        if ("action_admin_release".equals(stringExtra)) {
            destroy();
        } else if ("action_admin_exportLog".equals(stringExtra)) {
            String f = wd.f(this, "");
            if (f == null) {
                return;
            }
            if (!wd.d(f + "/Aegis")) {
                wd.c(f + "/Aegis");
            }
            File[] g = wd.g(wd.h(this, "/MobileSticker/log/"));
            if (g != null) {
                for (File file : g) {
                    wd.b(file.getPath(), f + "/Aegis/exportLog_" + file.getName());
                }
                de.X(this, "로그파일을 추출했습니다.\n경로-" + f);
            }
        } else {
            intent2.putExtra("extra_result", "0");
        }
        sendBroadcast(intent2);
    }
```

자료 7 ServiceAEGIS 서비스의 Release or Export 메서드

4.2.2_ ServiceAlert – Soldier, Staff & External

ServiceAlert 서비스는 앱의 기능을 바탕으로 사용자 및 앱에 대한 텍스트 음성 변환과 메시지 알림을 생성합니다.

4.2.3_ ServiceCall – *Staff*

ServiceCall 서비스는 직원 버전의 앱에만 구현되어 있습니다. 이 서비스는 *'BroadcastReceiverCall'* 브로드캐스트 리시버가(4.3.5 장) "android.intent.action.NEW_OUTGOING_CALL" 또는 "android.intent.action.PHONE_STATE" 브로드캐스트 인텐트를 수신할 때 시작됩니다. 이는 자료 8 에서 확인하실 수 있습니다. 사용자가 전화를 받거나 걸 때 서비스가 시작되고, 서비스 생성과 수신한 인텐트의 세부 정보는 자료 9 에서와 같이 앱의 로그 파일에 기록됩니다. 이 서비스는 전화번호를 포함한 통화의 데이터를 파싱하고 4.5 장에서 다루게 될 앱의 SQL 데이터베이스에 데이터를 기록합니다. 이 서비스가 생성되기 위해서는 기기의 언락 여부를 검사하고, 그에 따라 실행 여부를 결정합니다. - 자료 10 참조.

```
309     @Override // android.app.Service
310     public void onCreate() {
311         super.onCreate();
312         ae.d(f1194a, "onCreate: 호출");
313         try {
314             if (sd.f(this, getPackageName()) < 26 || Build.VERSION.SDK_INT < 26) {
315                 return;
316             }
317             be.f(this);
318             startForeground(1, be.e(this, getResources().getString(cd.krgomndmmsavtCheck(getPackageName())), getResources().getString(R.string.
    mndmdm_common_noti_service), cd.k()));
319         } catch (NullPointerException | Exception e) {
320             ae.e(f1194a, e);
321         }
322     }
323
324     @Override // android.app.Service
325     public int onStartCommand(Intent intent, int i, int i2) {
326         ae.d(f1194a, "onStartCommand: START");
327         if (intent == null) {
328             ae.d(f1194a, "onStartCommand: intent is NULL");
329             broadcastreceiverregist();
330             return 1;
331         }
332         String action = intent.getAction();
333         if (action != null) {
334             String str = f1194a;
335             ae.d(str, "onStartCommand: action is = " + action);
336             if ("android.intent.action.BOOT_COMPLETED".equals(action)) {
337                 broadcastreceiverregist();
338                 return 1;
339             } else if ("android.intent.action.PHONE_STATE".equals(action)) {
340                 checkDeviceLockStatusListen(this);
341             }
342         } else {
343             ae.d(f1194a, "onStartCommand: intent action is NULL");
344         }
345         return super.onStartCommand(intent, i, i2);
346     }
347 }
```

자료 8 ServiceCall 서비스의 소스 코드.

```
[D]     [16:28:59]      ServiceCall:    onStartCommand: START
[D]     [16:28:59]      ServiceCall:    onStartCommand: action is = android.intent.action.PHONE_STATE
[D]     [16:28:59]      ServiceCall:    onCallStateChanged: tel state RINGING
[D]     [16:29:00]      ServiceCall:    ServiceCall: 생성자 호출
[D]     [16:29:00]      ServiceCall:    onCreate: 호출
[D]     [16:29:00]      ServiceCall:    onStartCommand: START
[D]     [16:29:00]      ServiceCall:    onStartCommand: action is = android.intent.action.PHONE_STATE
[D]     [16:29:51]      ServiceCall:    ServiceCall: 생성자 호출
[D]     [16:29:51]      ServiceCall:    onCreate: 호출
[D]     [16:29:51]      ServiceCall:    onStartCommand: START
[D]     [16:29:51]      ServiceCall:    onStartCommand: action is = android.intent.action.PHONE_STATE
[D]     [16:29:51]      ServiceCall:    onCallStateChanged: tel state RINGING
[D]     [16:29:51]      ServiceCall:    ServiceCall: 생성자 호출
[D]     [16:29:51]      ServiceCall:    onCreate: 호출
[D]     [16:29:51]      ServiceCall:    onStartCommand: START
[D]     [16:29:51]      ServiceCall:    onStartCommand: action is = android.intent.action.PHONE_STATE
```

자료 9 서비스 동작을 보여주는 앱의 로그 파일

```
public static boolean f(Context context) {
    if (context == null) {
        return false;
    }
    ud n = ud.n(context);
    String s = n.s("CompanyInfo", "device_lock_status", o("off", 0));
    if (s.length() == 2 || s.length() == 3) {
        s = o(s, 0);
        n.z("CompanyInfo", "device_lock_status", s);
    }
    return !"off".equals(j(s, 0));
}
```

자료 10 DeviceLockStatus 검사

4.2.4_ ServicePolicy – Soldier, Staff & External

ServicePolicy 서비스는 앞의 장에서 다룬 설정 파일의 변경을 수행합니다.
4.1 장에서 구체적으로 다루었듯이, 이 서비스는 받은 인증 코드에 따라 설정 파일을 변경합니다. 자료 11 의 예시에서 메서드가 인증 코드를 검사한 뒤 설정 파일을 변경하는 것을 보실 수 있습니다. 이러한 변경 사항은 앱의 SQL 데이터베이스에도 기록됩니다.

```
public final void e(Context context) {
    try {
        ud n = ud.n(context);
        String j = ud.j(context);
        Intent intent = new Intent("android.intent.action.MAIN");
        String s = f1203a.s("CompanyInfo", "company_info_sub_code", "");
        if (j.equals("msGoLhcVq0dT7rv3dI+HDw==") || j.equals("Zn8bhFsk/+xh/pThclDRXg==")) {
            intent.putExtra("mmsa_policy", "mmsa_in_policy");
            n.z("ConfigInfo", "camera", "on");
        }
        if (j.equals("EaVdqsUlA5/1KGzYcv+Rtw==")) {
            intent.putExtra("mmsa_policy", "mmsa_of_a_policy");
            n.z("ConfigInfo", "camera", "on");
            n.z("ConfigInfo", "mic", "on");
        }
        if (j.equals("GsbJVE7AI0g7NrcEVtnmlg==")) {
            if (!s.equals("") && s.length() > 0) {
                n.z("ConfigInfo", "manager", "on");
            }
            intent.putExtra("mmsa_policy", "mmsa_of_b_policy");
            n.z("ConfigInfo", "camera", "on");
            n.z("ConfigInfo", "mic", "on");
            n.z("ConfigInfo", "usb", "on");
            n.z("ConfigInfo", "wifi", "on");
            n.z("ConfigInfo", "tethering", "on");
        }
        intent.putExtra("kr.go.mnd.mmsa.MANAGING_COMMAND", "start_manage");
        intent.setComponent(new ComponentName("kr.go.mnd.mmsa.mf", "kr.go.mnd.mmsa.mf.manager.ServicePolicy"));
        if (sd.f(context, context.getPackageName()) < 26 || Build.VERSION.SDK_INT < 26) {
            context.startService(intent);
        } else {
            context.startForegroundService(intent);
        }
    } catch (Exception e) {
        ae.e(f1201a, e);
    }
}
```

자료 11 ServicePolicy 서비스에서 이루어지는 설정 변경의 예시

4.2.5_ ServiceMFHW – *External*

이 서비스는 화웨이 기기에 맞는 정책 설정을 수행합니다. 기능은 이전 서비스와 거의 동일합니다.

4.2.6_ ServiceMFLG – Soldier, Staff & External

이 서비스는 LG 기기에만 해당된다는 점을 제외하면 4.2.5 와 동일합니다.

4.2.7_ ServiceMFSS – Soldier, Staff & External

이 서비스는 삼성 기기에만 해당된다는 점을 제외하면 4.2.5 와 동일합니다.

4.2.8_ ServiceDeviceAdmin – Soldier, Staff & External

이 서비스는 안드로이드 기기의 관리자 권한을 설정합니다

https://developer.android.com/guide/topics/admin/device-admin

4.2.9_ ServiceLocation – Soldier, Staff & External

이 서비스는 안드로이드 기기의 위치정보 제공자 변경을 감지하고 이에 따라 앱의 동작을 변경합니다.

4.3_ Broadcast receivers

이 장에서는 앱의 브로드캐스트 리시버와 그 기능에 관한 내용을 다루겠습니다. 우선, 엡에서 내보내지는(외부에서 호출 가능한) 유일한 브로드캐스트 리시버가 "BroadcastReceiverExternal"임을 강조하고 싶습니다. 브로드캐스트 리시버는 앱의 kr.go.mnd.mmsa.of(직원) 버전과 kr.go.mnd.mmsa.vt(외부인) 버전에서 내보내집니다. 브로드캐스트 리시버가 내 보내진다는 것은 누구든 기기에서 그 리시버를 호출할 수 있다는 것을 의미합니다. 또한 이 리시버에는 어떠한 권한도 설정되어 있지 않습니다. 자료 12 에 나오는 외부인 버전 앱의 AndroidManifest 에서 내보내진 브로드캐스트 리시버를 확인하실 수 있습니다. 자료 13 에서 역시 권한 관련 문제를 확인할 수 있습니다. 5 장에서는 공격자가 내보내진 브로드캐스트 리시버를 악용하여 앱에서 정보를 추출하는 방식에 대해 다루겠습니다.

AndroidManifest.xml

Find: broadcast

```
337                 <action android:name="edm.intent.action.device.outside"/>
338                 <action android:name="edm.intent.action.device.location.unavailable"/>
339                 <action android:name="com.samsung.edm.intent.action.CERTIFICATE_REMOVED"/>
340                 <action android:name="edm.intent.certificate.action.certificate.failure"/>
341                 <action android:name="com.samsung.edm.intent.action.APPLICATION_FOCUS_CHANGE"/>
342                 <action android:name="edm.intent.action.EMAIL_ACCOUNT_ADD_RESULT"/>
343                 <action android:name="edm.intent.action.EMAIL_ACCOUNT_DELETE_RESULT"/>
344                 <action android:name="com.sec.enterprise.intent.action.BLOCKED_DOMAIN"/>
345                 <action android:name="com.sec.enterprise.intent.action.UPDATE_FOTA_VERSION_RESULT"/>
346                 <action android:name="com.samsung.edm.intent.action.EXCHANGE_CBA_INSTALL_STATUS"/>
347                 <action android:name="android.intent.action.sec.CBA_INSTALL_STATUS"/>
348                 <action android:name="edm.intent.action.EXCHANGE_ACCOUNT_ADD_RESULT"/>
349                 <action android:name="edm.intent.action.EXCHANGE_ACCOUNT_DELETE_RESULT"/>
350                 <action android:name="com.samsung.edm.intent.action.ENFORCE_SMIME_ALIAS_RESULT"/>
351                 <action android:name="edm.intent.action.knox_license.status"/>
352                 <action android:name="edm.intent.action.license.status"/>
353                 <action android:name="com.samsung.edm.intent.event.NTP_SERVER_UNREACHABLE"/>
354                 <action android:name="edm.intent.action.enable.kiosk.mode.result"/>
355                 <action android:name="edm.intent.action.disable.kiosk.mode.result"/>
356                 <action android:name="edm.intent.action.unexpected.kiosk.behavior"/>
357                 <action android:name="com.samsung.edm.intent.action.SIM_CARD_CHANGED"/>
358                 <action android:name="android.intent.action.sec.SIM_CARD_CHANGED"/>
359                 <action android:name="com.samsung.action.knox.certenroll.CEP_CERT_ENROLL_STATUS"/>
360                 <action android:name="com.samsung.action.knox.certenroll.CEP_SERVICE_DISCONNECTED"/>
361                 <action android:name="com.sec.enterprise.knox.intent.action.KNOX_ATTESTATION_RESULT"/>
362                 <action android:name="com.sec.action.NO_USER_ACTIVITY"/>
363                 <action android:name="com.sec.action.USER_ACTIVITY"/>
364                 <action android:name="com.samsung.android.mdm.VPN_BIND_RESULT"/>
332             </intent-filter>
329         </receiver>
369         <receiver android:name="kr.go.mnd.mmsa.vt.br.BroadcastReceiverExternal" android:exported="true">
372             <intent-filter>
373                 <action android:name="com.markany.aegis.AEGIS_ACTION_ADMIN_REQUEST"/>
372             </intent-filter>
```

자료 12 내보내진 브로드캐스트 리시버를 보여주는 kr.go.mnd.mmsa.of(직원)과 kr.go.mnd.mmsa.vt (외부인)의 Android manifest.

```
dz> run app.broadcast.info -a kr.go.mnd.mmsa.of
Package: kr.go.mnd.mmsa.of
  kr.go.mnd.mmsa.of.br.BroadcastReceiverExternal
    Permission: null

dz>
```

자료 13 Permissions requirements

4.3.1_ BroadcastReceiverExternal – Soldier, Staff & External

이 브로드캐스트 리시버는 "com.markany.aegis.AEGIS_ACTION_ADMIN_REQUEST" 인텐트 필터를 가지며, 이 인텐트의 브로드캐스트를 수신할 때 ServiceAEGIS(4.2.1 참고)를 생성하며, 브로드캐스트에서 "action_admin" & "action_admin_exportLog" or "action_admin_release"와 같은 추가적인 문자열 키 값을 확인합니다. 앞서 4.2.1 장에서 다루었듯이 이 값에 따라 두 함수 중 하나가 실행됩니다:

- 기기의 앱 로그 파일을 기기의 공유 저장소에 추출하기
- 게이트키퍼(GateKeeper)를 비활성화하고 사용자의 앱 제거를 허용

4.3.2_ BroadcastReceiverGDA_LG – Soldier, Staff & External

이 리시버는 LG 기종 기기의 관리자 설정과 관련된 인텐트를 수신한 뒤, 인텐트에 따라서 몇 가지 설정을 수행하고 4.2.8 장에서 다룬 ServiceDeviceAdmin 서비스를 시작합니다.

4.3.3_ BroadcastReceiverRestartAegisSAFER – Soldier, Staff & External

이 브로드캐스트 리시버는 "com.markany.aegis.gate.AEGIS_GATE_SERVICE_DEVICE_ADMIN"와 같이 Aegis GATE 에 관련된 필터에 해당하는 인텐트를 수신하는 것으로 보입니다. 이 리시버는 해당 인텐트를 수신한 뒤 ServiceDeviceAdmin 서비스(4.2.8)를 시작합니다.

4.3.4_ BroadcastReceiverSystem – Soldier, Staff & External

이 리시버는 기기의 전원이 켜졌을 때 ServiceAEGIS(4.2.1)를 시작합니다.

4.3.5_ BroadcastReceiverCall - Staff

이 브로드캐스트 리시버는 전화 발신 또는 기기 상태의 변화와 관련된 인텐트를 수신했을 때 4.2.3 장에서 설명된 ServiceCall 서비스를 시작합니다. 서비스는 DeviceLockStatus 설정을 검사한 뒤 시작됩니다. 이는 자료 14 에서 확인하실 수

있습니다.

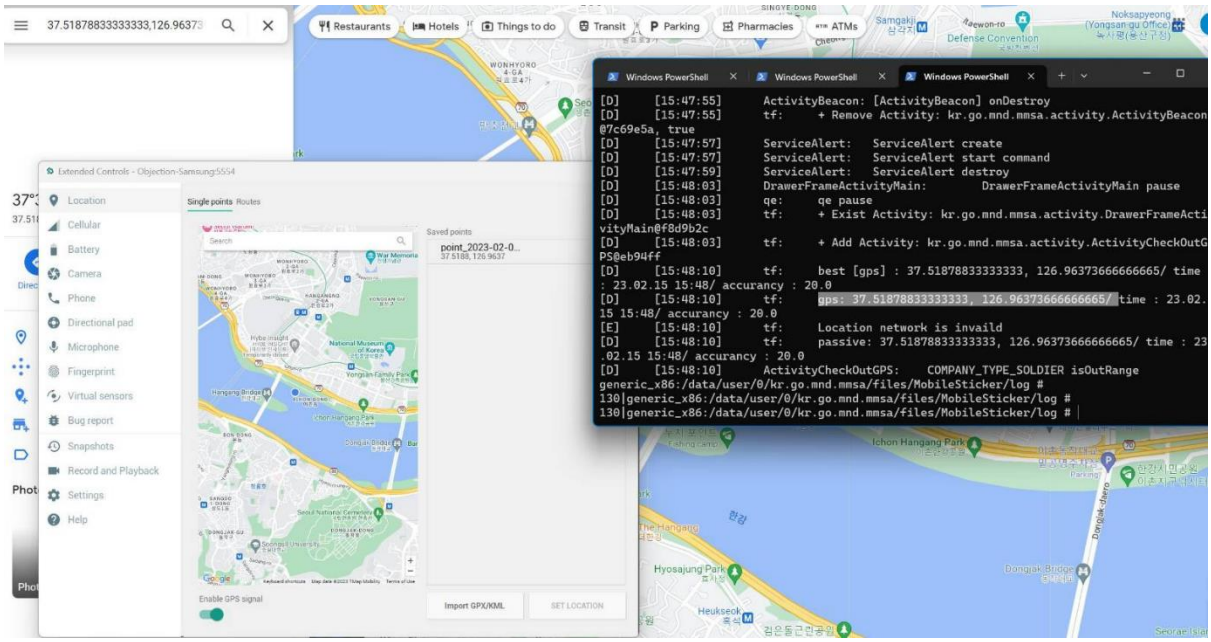
```java
/* loaded from: classes.dex */
public class BroadcastReceiverCall extends BroadcastReceiver {
    public static final String a = BroadcastReceiverCall.class.getSimpleName();

    @Override // android.content.BroadcastReceiver
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (action == null) {
            return;
        }
        String str = a;
        ae.addToLog(str, "<<<<< RECEIVE INTENT: " + action);
        ud n = ud.n(context);
        if (de.checkDeviceLockStatus(context)) {
            if ("android.intent.action.NEW_OUTGOING_CALL".equals(action)) {
                n.z("ConfigInfo", "outGoingNumber", intent.getExtras().getString("android.intent.extra.PHONE_NUMBER"));
            } else if ("android.intent.action.PHONE_STATE".equals(action)) {
                Intent intent2 = new Intent(context, ServiceCall.class);
                intent2.setAction("android.intent.action.PHONE_STATE");
                if (Build.VERSION.SDK_INT >= 26) {
                    context.startForegroundService(intent2);
                } else {
                    context.startService(intent2);
                }
            }
        }
    }
}
```

자료 14 BroadcastReceiverCall 소스코드

4.4_ Personal data

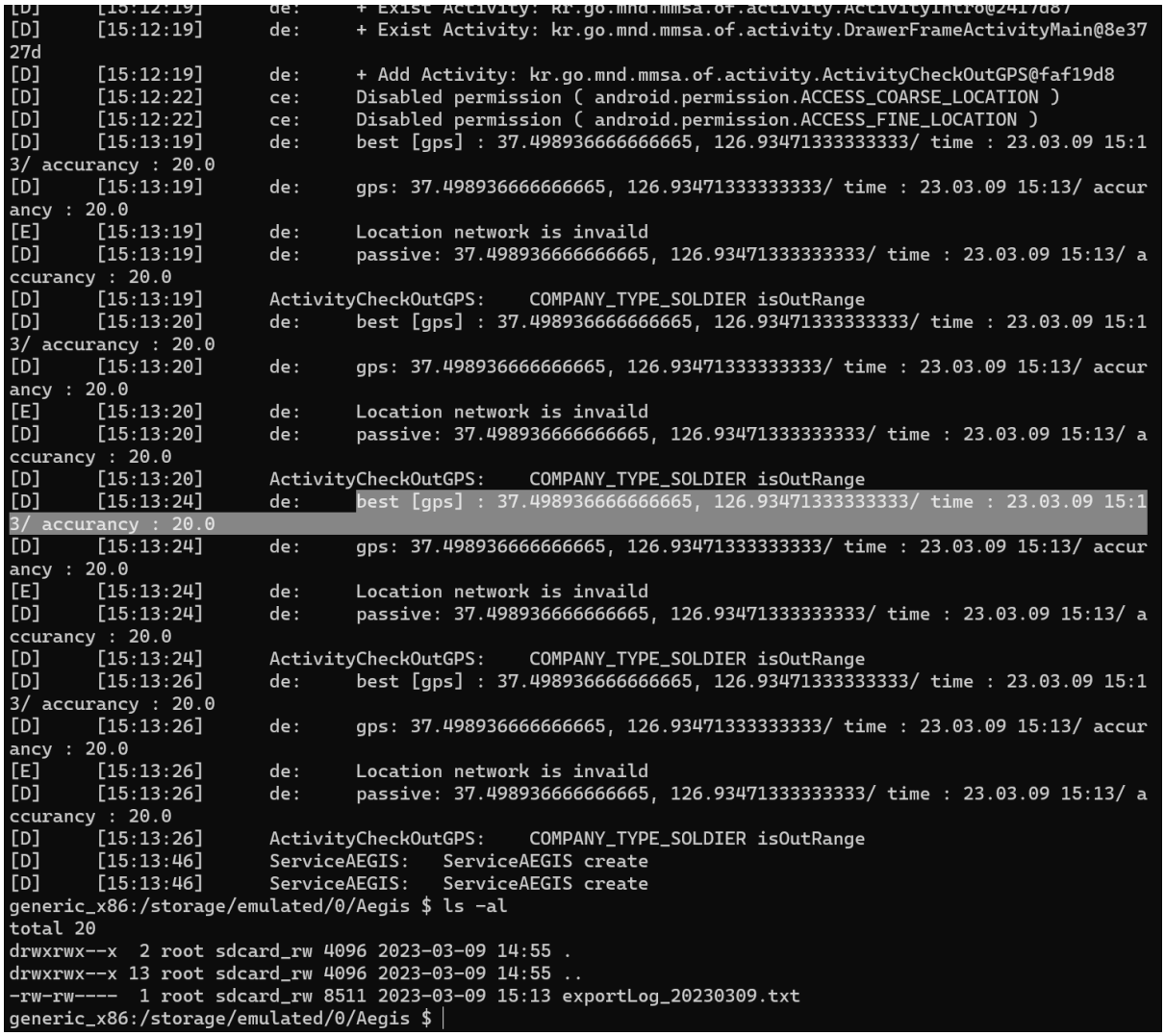
kr.go.mnd.mmsa (병사), , **kr.go.mnd.mmsa** (직원), **kr.go.mnd.mmsa.vt** (외부인) 세 버전의 앱 모두 전부 GPS 위치정보와 시간, 날짜를 /data/user/0/kr.go.mnd.mmsa/files/MobileSticker/log 위치에 있는 일일 로그 파일에 작성합니다. 관련 내용은 자료 15 에서 확인하실 수 있습니다. 로그 작성 동작은 자료 16(이 자료는 **kr.go.mnd.mmsa.vt** 버전 앱의 특정 클래스를 나타내며, 클래스 이름은 난독화 과정에서 앱의 버전별로 달라집니다. 유사한 동작을 하는 클래스의 이름은 각 버전의 앱에서 얻을 수 있습니다.)에서 나타나듯 클래스를 작동시키는 *ActivityCheckOutGPS* 액티비티에서 이루어집니다. 로그는 사용자가 위치 기반으로 기기 잠금 해제를 시도하는 등의 *ActivityCheckOutGPS* 액티비티에 관련된 GPS 기능을 사용할 때 작성됩니다. 아래의 자료 17 에서 예시를 참고하실 수 있습니다.



자료 15 data/user/0/kr.go.mnd.mmsa/files/MobileSticker/log 디렉터리의 로그 파일에 저장된 GPS 위치정보 예시

```
@SuppressLint({"MissingPermission"})
public static Location u(Context context) {
    Location location;
    if (context == null) {
        return null;
    }
    ArrayList arrayList = new ArrayList();
    try {
        LocationManager locationManager = (LocationManager) context.getSystemService("location");
        String bestProvider = locationManager.getBestProvider(new Criteria(), true);
        if (bestProvider != null) {
            location = locationManager.getLastKnownLocation(bestProvider);
            if (location != null) {
                yd.d(f510a, "best [" + location.getProvider() + "] : " + location.getLatitude() + ", " + location.getLongitude() + "/ time : " + y(Long.valueOf(location.getTi
            } else {
                yd.f(f510a, "Location best is invaild");
            }
        } else {
            location = null;
        }
        Location lastKnownLocation = locationManager.getLastKnownLocation("gps");
        if (lastKnownLocation != null) {
            arrayList.add(lastKnownLocation);
            yd.d(f510a, "gps: " + lastKnownLocation.getLatitude() + ", " + lastKnownLocation.getLongitude() + " / time : " + y(Long.valueOf(lastKnownLocation.getTime())) + "/
        } else {
            yd.f(f510a, "Location gps is invaild");
        }
        Location lastKnownLocation2 = locationManager.getLastKnownLocation("network");
        if (lastKnownLocation2 != null) {
            arrayList.add(lastKnownLocation2);
            yd.d(f510a, "network: " + lastKnownLocation2.getLatitude() + ", " + lastKnownLocation2.getLongitude() + " / time : " + y(Long.valueOf(lastKnownLocation2.getTime())
        } else {
            yd.f(f510a, "Location network is invaild");
        }
        Location lastKnownLocation3 = locationManager.getLastKnownLocation("passive");
        if (lastKnownLocation3 != null) {
            arrayList.add(lastKnownLocation3);
            yd.d(f510a, "passive: " + lastKnownLocation3.getLatitude() + ", " + lastKnownLocation3.getLongitude() + "/ time : " + y(Long.valueOf(lastKnownLocation3.getTime())
        } else {
            yd.f(f510a, "Location passive is invaild");
        }
        if (location == null && lastKnownLocation == null && lastKnownLocation2 == null && lastKnownLocation3 == null) {
            return null;
        }
        if (location == null || System.currentTimeMillis() - location.getTime() >= f508a) {
            Iterator it = arrayList.iterator();
            Location location2 = null;
            while (it.hasNext()) {
                Location location3 = (Location) it.next();
                if (location2 == null) {
                    location2 = location3;
                }
                if (location2.getTime() < location3.getTime()) {
                    location2 = location3;
                }
```

자료 16 GPS 위치정보를 로그로 남기는 kr.go.mnd.mmsa.vt 앱의 kr.go.mnd.mmsa.be 클래스



자료 17 애플리케이션 로그 파일에 기록된 GPS 정보 예시

https://cwe.mitre.org/data/definitions/532.html. MITRE's description of CWE-532 state GPS 위치정보를 앱의 로그 파일에 저장하는 일은 CWE(Common Weakness Enumeration) ID 532 (로그 파일에 민감 정보 삽입) [4]에 위배됩니다. MITRE 는 CWE-532 에 대해 다음과 같이 설명합니다:

"로그 파일에 저장된 정보는 민감한 정보일 수 있으며 공격자에게 유용하게 이용되거나 민감한 사용자 데이터를 노출시킬 수 있다. 모든 정보를 로그에 남기는 것이 개발 단계에는 도움이 될 수 있으나, 제품을 배포하기 전에 로그 레벨을 적절히 설정하여 민감한 사용자 데이터와 시스템 정보가 잠재적인 공격자에게 실수로 노출되지 않도록 하는 것이 중요하다 [역주: 영어 원문을 한국어로 번역]"

또한 MITRE 는 CWE-532 설명 문서에서 사용자의 현재 위치정보에 대한 부적절한 로그 동작에 대한 예시를 제공하고 있습니다.

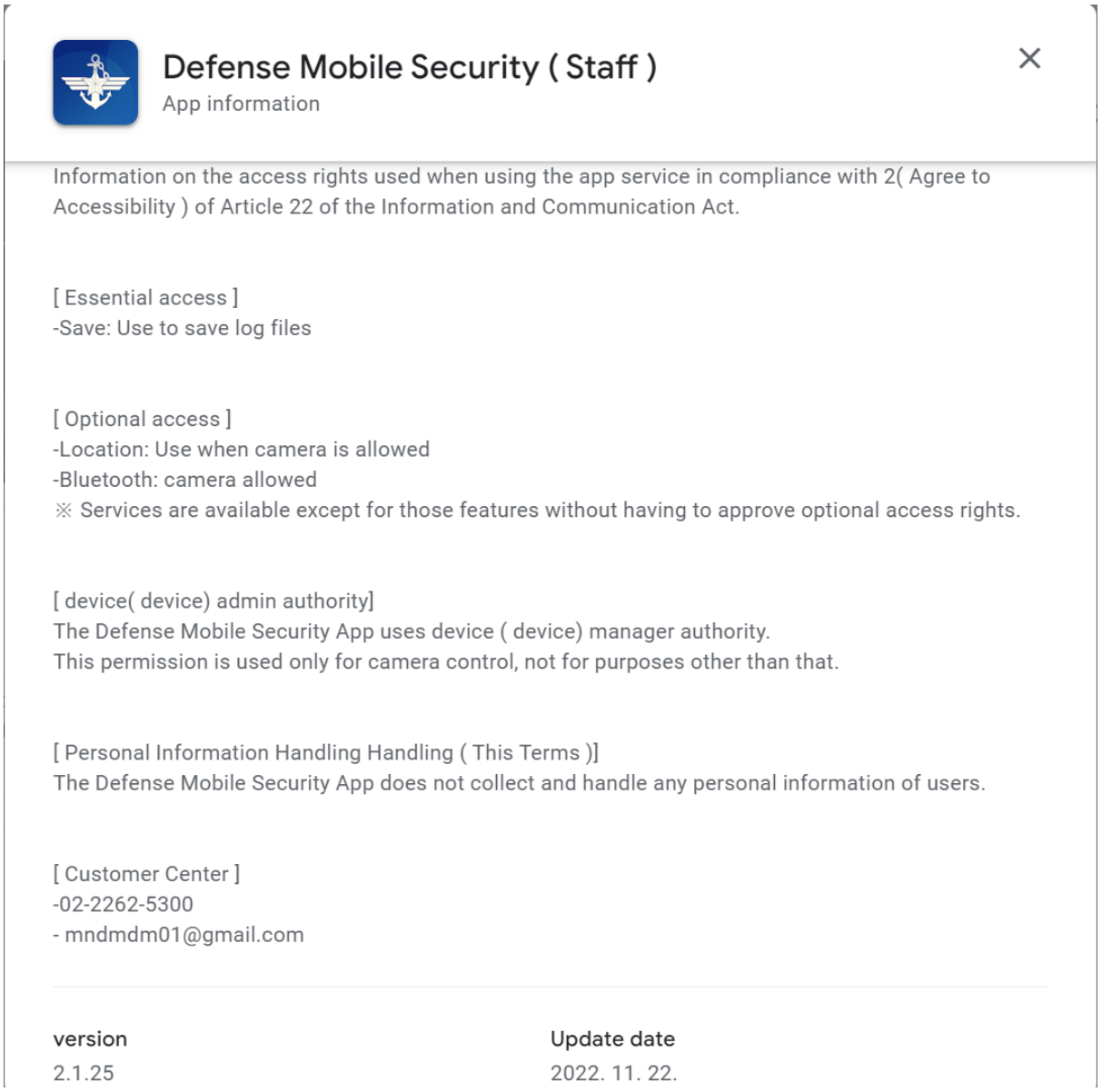
앱이 사용자의 대략적인 위치정보를 기록하고 있기에, 이는 사용자들에게 중대한 위협이 될 수 있습니다. 여러 가지 요소를 감안할 때, 이는 심각한 사생활 침해의 위험이 있습니다:

1. 자료 18 의 앱 정보란에서 국방부는 "... 사용자의 개인정보 일체를 수집 및 취급하지 않습니다."라 말합니다. 위치정보의 보호 및 이용 등에 관한 법률(이하 '위치정보법') [5] 에서 개인 위치정보 데이터는 개인 식별에 사용될 수 있음이 명시되어 있습니다. ("개인위치정보"라 함은 특정 개인의 위치정보(위치정보만으로는 특정 개인의 위치를 알 수 없는 경우에도 다른 정보와 용이하게 결합하여 특정 개인의 위치를 알 수 있는 것을 포함한다)를 말한다.) 이에 근거하여 위치정보는 개인정보 중 하나로 취급됩니다. 그러나, kr.go.mnd.mmsa(병사), kr.go.mnd.mmsa.of (직원), kr.go.mnd.mmsa.vt (외부인) 세 버전 모두 로그 파일에 사용자의 GPS 위치정보 기록을 저장합니다. **더 나아가, kr.go.mnd.mmsa.of (Staff) & kr.go.mnd.mmsa.vt (External)의 로그 파일은 보호되지 않아 기기에 접근하는 누구든지 이 파일을 추출할 수 있습니다. (이에 관해서는 5 장에서 다루고 있습니다.)**
2. 위치정보법에 따르면 이용약관에 명시하지 않았거나 개인의 동의를 받지 않았을 때에는 위치정보를 수집할 수 없습니다. 앱의 사용자가 받는 유일한 위치정보 수집에 관한 안내는 기기의 위치정보 액세스 권한 요청뿐입니다. 그러나 앱과 구글 플레이 스토어 항목(자료 19 확인) 어디에도 GPS 위치정보를 수집하고 로그 파일에 저장한다는 안내는 나와 있지 않습니다. 이는 앱이 개인정보의 수집 기준을 위반하는 것으로 볼 수 있습니다.
3. 위치정보법에 따르면 개인 위치정보의 수집, 이용 또는 제공목적을 달성한 때에는 파기되어야 합니다. 앱은 GPS 위치정보를 기록한 후 기기에 JSON 파일로 저장되어 있는 허가된 위치와의 비교를 수행합니다. 확인 작업이 끝난 뒤에는, GPS 위치는 더 이상 필요하지 않습니다. 따라서, 사용 후 개인 위치정보 파기의 미흡은 법률에 위반되는 사항입니다.

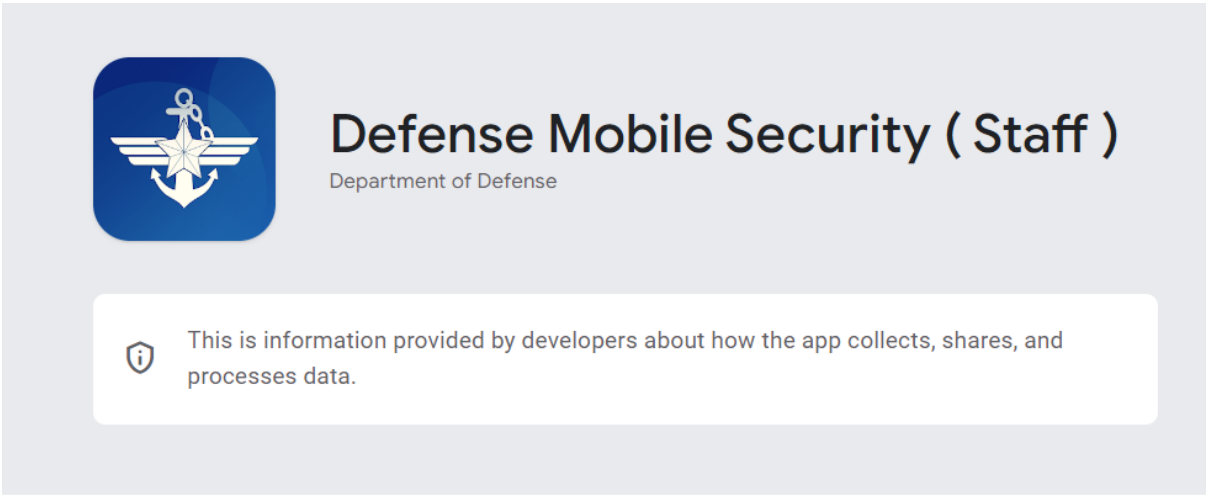
4. 위치정보법에 따르면 개인 위치정보를 사용하는 위치정보사업자 등은 의무적으로 "위치정보의 누출, 변조, 훼손 등을 방지하기 위하여 위치정보의 취급・관리 지침을 제정하거나 접근권한자를 지정하는 등의 관리적 조치와 방화벽의 설치나 암호화 소프트웨어의 활용 등의 기술적 조치를 하여야 한다. 이 경우 관리적 조치와 기술적 조치의 구체적 내용은 대통령령으로 정한다."의 규정에 따라야 할 의무가 있습니다. 따라서, 국방부는 개인 위치정보 데이터의 오남용을 방지할 책임이 있습니다. **5 장**에서 다루게 될 취약점을 볼 때, 앱의 로그 작성은 이 규정의 위반으로 볼 수 있습니다.

위 내용에 덧붙여 국방부는 앱이 GPS 위치 데이터를 수집하고 있음에도 불구하고 앱의 배포 과정에서 수차례 개인 정보를 수집하지 않는다고 말합니다:

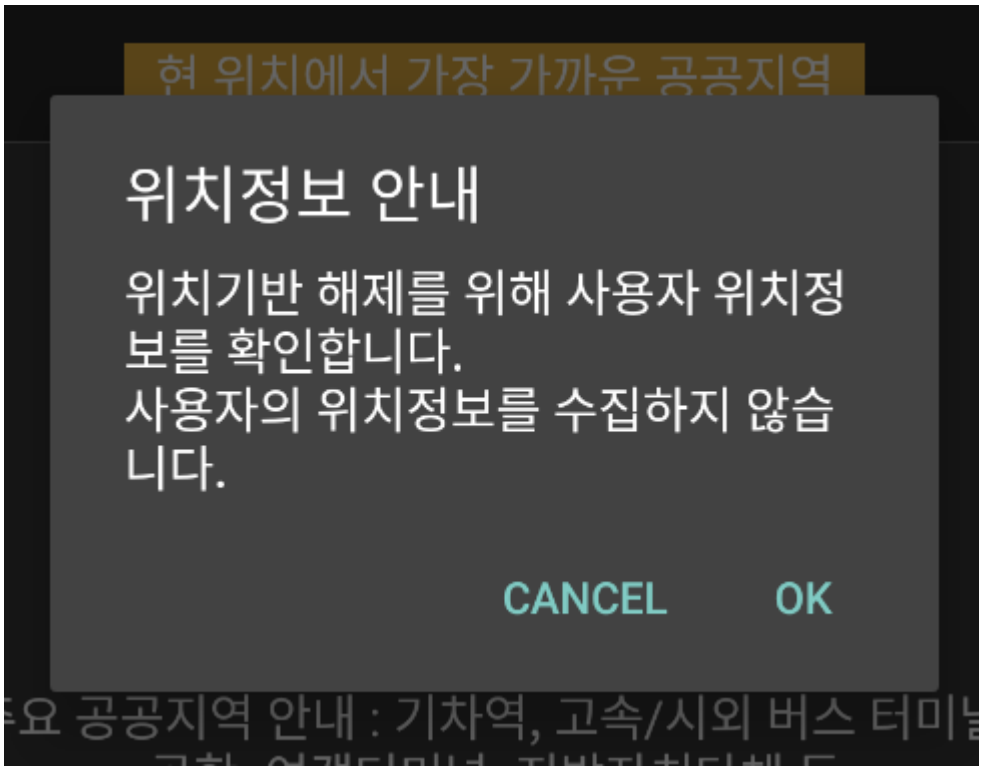
1. 자료 18 에서 국방부는 앱의 세 버전 모두에 반복하여 사용자의 개인 정보를 수집하지 않는다는 점에 대해 설명하고 있습니다. 위치정보법에 따르면 위치정보는 다른 정보와 용이하게 결합하여 특정 개인의 위치를 알 수 있는 특성을 가지고 있습니다. 위치정보는 개인정보의 범위 안에 포함됩니다.
2. 자료 19 를 보면 국방부는 구글 플레이 스토어에서 어떤 데이터도 수집되지 않는다고 말합니다. 기기의 로그 파일에 GPS 위치정보가 기록되고 있으므로 이는 사실이 아닙니다. 이 로그 파일은 기기에 접근(직접, 원격 모두 포함)할 수 있는 누구든 추출할 수 있습니다. 관련 내용은 5 장에서 살펴보실 수 있습니다.
3. 자료 20 과 자료 21 에서는 앱이 개인정보를 수집하지 않는다고 안내하는 두 별개의 화면 사진을 확인하실 수 있습니다.


Defense Mobile Security (Staff)
App information
Information on the access rights used when using the app service in compliance with 2(Agree to Accessibility) of Article 22 of the Information and Communication Act.
[Essential access]
-Save: Use to save log files
[Optional access]
-Location: Use when camera is allowed
-Bluetooth: camera allowed
※ Services are available except for those features without having to approve optional access rights.
[device(device) admin authority]
The Defense Mobile Security App uses device (device) manager authority.
This permission is used only for camera control, not for purposes other than that.
[Personal Information Handling Handling (This Terms)]
The Defense Mobile Security App does not collect and handle any personal information of users.
[Customer Center]
-02-2262-5300
- mndmdm01@gmail.com
version
2.1.25
Update date
2022. 11. 22.

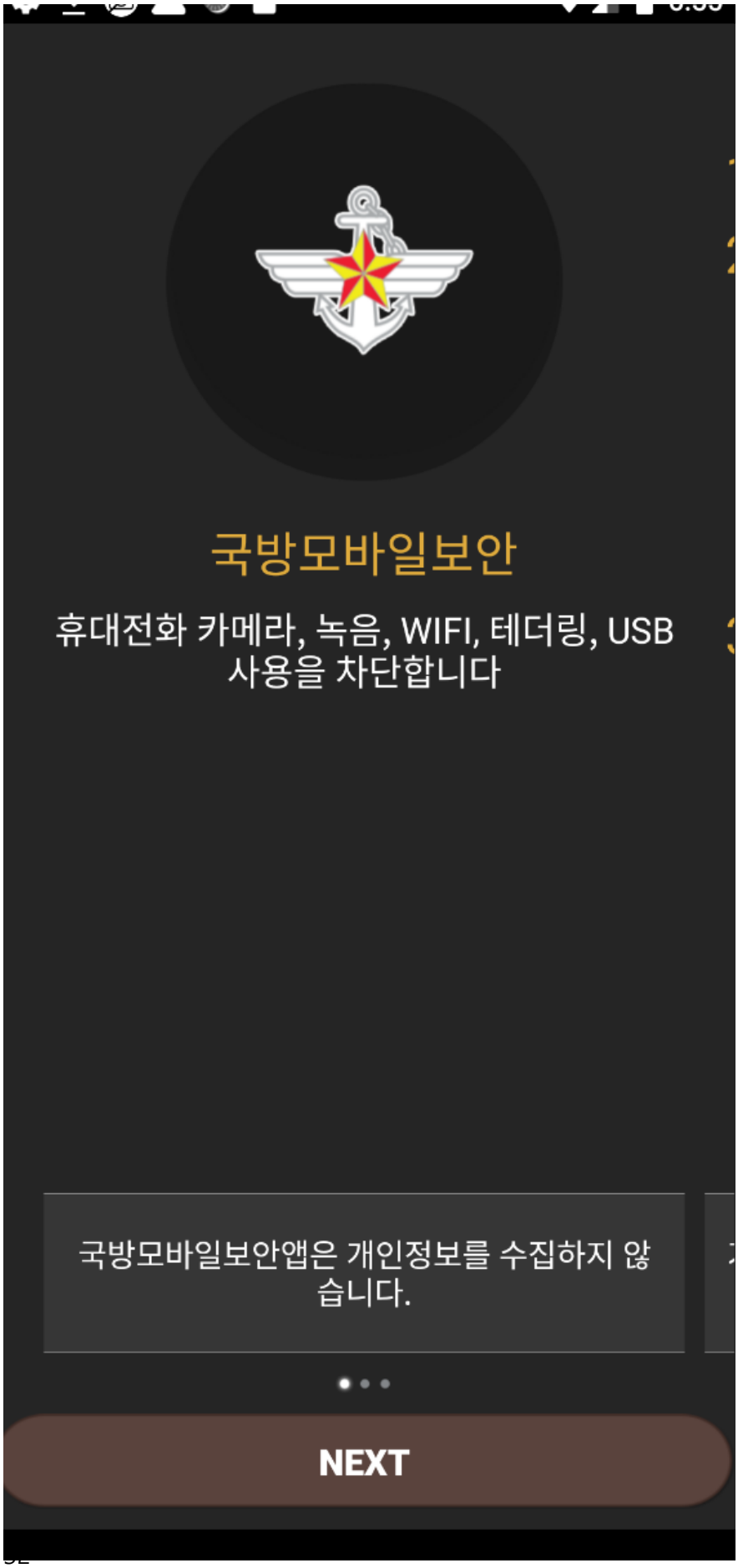
자료 18 앱 정보



자료 19 구글 플레이 스토어 - 데이터 보안 페이지



자료 20 앱이 개인 정보를 수집하지 않는다고 명시한 안내 메시지


국방모바일보안
휴대전화 카메라, 녹음, WIFI, 테더링, USB
사용을 차단합니다
국방모바일보안앱은 개인정보를 수집하지 않
습니다.
NEXT

자료 21 앱이 개인정보를 수집하지 않는다는 매뉴얼 페이지

4.5_ Local SQL Database

앱에는 AegisGate.db 라는 이름의 로컬 SQL 데이터베이스가 포함되어 있습니다.

이 데이터베이스는 기기의 정보를 기록하고 저장하는 역할을 합니다. 그 중 관리자 전화 정보, 기기 정보, 설정 정보 등을 자료 22 에서 확인하실 수 있습니다. 또한 기기 언락 여부 등의 정보도 저장하여 앱이 특정 기능을 수행할 때 사용됩니다. 이외의 몇 가지 경우는 이미 앞선 장에서 설명되었습니다.

Name	Type	Schema
Tables (10)		
AdminLogInfo		CREATE TABLE AdminLogInfo(id INTEGER PRIMARY KEY AUTOINCREMENT,value TEXT,time INTEGER DEFAULT 0)
AdminTelInfo		CREATE TABLE AdminTelInfo(id INTEGER PRIMARY KEY AUTOINCREMENT,value TEXT,time INTEGER DEFAULT 0)
AgentInfo		CREATE TABLE AgentInfo(id INTEGER PRIMARY KEY,name TEXT,value TEXT,time INTEGER DEFAULT 0)
AgentK		CREATE TABLE AgentK(id INTEGER PRIMARY KEY AUTOINCREMENT,name TEXT,value TEXT)
CheckInfo		CREATE TABLE CheckInfo(id INTEGER PRIMARY KEY,value TEXT,value1 TEXT,value2 TEXT,time INTEGER DEFAULT 0)
CompanyInfo		CREATE TABLE CompanyInfo(id INTEGER PRIMARY KEY,name TEXT,value TEXT,time INTEGER DEFAULT 0)
ConfigInfo		CREATE TABLE ConfigInfo(id INTEGER PRIMARY KEY,name TEXT,value TEXT,time INTEGER DEFAULT 0)
DeviceInfo		CREATE TABLE DeviceInfo(id INTEGER PRIMARY KEY,name TEXT,value TEXT,time INTEGER DEFAULT 0)
android_metadata		CREATE TABLE android_metadata (locale TEXT)
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)

자료 22 AegisGate.db tables

이 데이터베이스에서 가장 중요한 부분은 다음 테이블입니다:

- AdminTelInfo
- ConfigInfo
- AgentK

외부인 버전의 앱에서는 사용자로부터 AdminTelephone 번호를 입력 받습니다. 입력 받은 전화번호는 AdminTellInfo 테이블에 기록됩니다.

직원 버전의 앱에서는 자료 23 에서 보다시피 ServiceCall 클래스(4.2.3 장)와 BroadcastReceiverCall 클래스(4.3.5 장)가 담겨있고, 자료 23 에 나오듯 수신 전화와 발신 전화의 전화번호가 ConfigInfo 테이블에 기록됩니다.

	id	name	value	time
	Filter	Filter	Filter	Filter
1	1	config_uc_f_count	eYKlUjbrIoOg7Sv4kBhwKQ==	1679584430780
2	2	user_guide	jg5BKzlNG6paTxzu45egMQ==	1679583719562
3	3	incomingNumber	Lo4jxy578Oz6P5IX0VANRQ==	1679588992072
4	4	emergency	sqJKoXnqDM06Mayh12CrWQ==	1679588991949
5	5	outGoingNumber	1VpqKlaN5M+S75+Tarne3A==	1679588992015

자료 23 직원 버전 앱의 ConfigInfo 테이블

관리자 전화번호와 수신/발신 번호의 저장은 개인정보 문제가 아닐 수 있지만, 앱이 저장된 데이터를 사용하지는 않았기에 이 번호가 저장되어야 할 이유는 알 수 없었습니다. 더 이해할 수 없었던 점은, 데이터의 인코딩과 암호화입니다.

데이터베이스에 포함된 모든 값들은 AES/CBC/PKCS5Padding 으로 암호화되고 base64 로 인코딩이 되어있습니다. 그러나, 기기의 다른 암호화 방식과는 달리 AES 키는 타임스탬프를 기반으로 하여(자료 23 에서 예시를 보실 수 있습니다.) 생성됩니다. 이 타임스탬프는 SHA256 인코딩을 거쳐 32 바이트 값의 AES 암호화 키로 사용됩니다. 전화번호는 타임스탬프 기반 키와 앱에서 생성된 IV(초기화 벡터)값을 이용해 암호화를 거친 뒤 base64 로 인코딩 됩니다(자료 24 에 이 과정이 나와 있습니다).

```
/* renamed from: m */
public static String encodePhoneNumber(String phonenumber, long datetime) {
    try {
        byte[] bArr = new byte[16];
        System.arraycopy(new ConfigData().getInv(), 0, bArr, 0, 16);
        SecretKeySpec secretKeySpec = new SecretKeySpec(SHA256HashOfDateTime(datetime), "AES");
        Cipher cipher = Cipher.getInstance(b);
        cipher.init(1, secretKeySpec, new IvParameterSpec(bArr, 0, 16));
        return new String(Base64.encode(cipher.doFinal(phonenumber.getBytes(DataUtil.defaultCharset)), 2));
    } catch (Exception e) {
        ae.e(f556a, e);
        return null;
    }
}
```

자료 24 Encryption method for phone numbers within the database

여기서 가장 주목할 점은 이 메서드에서 타임스탬프를 기반으로 생성된 키가 데이터베이스 파일에 저장된다는 사실입니다. 특정 타임스탬프를 기반으로 AES 키가 생성될 때마다 키의 값은 모두 AgentK 테이블에 저장됩니다.

직원 버전의 앱에서 전화를 받을 경우를 예시로 들어 간단히 요약해 보겠습니다:

1. 수신 전화를 받았을 때, 걸려온 전화번호와 타임스탬프를 가져옵니다.
2. 타임스탬프는 SHA256 방식으로 AES 키를 만드는데 사용되며, 앱에서 생성된 IV 와 함께 전화번호를 AES 방식으로 암호화하는데 사용됩니다.
3. 자료 23 에서 나오듯이 암호화된 전화번호는 base64 인코딩을 거쳐 ConfigInfo 테이블에 저장됩니다.
4. 자료 25 에 나오듯이 2 번 과정에서 사용된 타임스탬프와 그를 이용해 생성된 AES 키는 AgentK 테이블에 저장됩니다.

	id	name	value
	Filter	Filter	Filter
1	1	1679583605085	59048296036eebf6ac0f3d8d1eef7a7f8ee4262f098b3c0d9787024e80b...
2	2	1679583605129	b0f52c6c24be07595d62601ab1724ec415fab5dab11fbcdd03a00f0a5e8...
3	3	1679583605322	ccf06369e379669b1abff6dcd2fd42411ce4faf92871036573633bd685d6...
4	4	1679583697942	ae8d197632056687ce62fff13fea3a074515611750a9c14d86cdb398a51...
5	5	1679583719562	ae58998762f330e5c708bf9ca4aed59810641a4dc1107ff18a7c2ecd3e6...
6	6	1679584111001	93eb50c004aeeaa3a1db17c59c66c4c2b7486511a6052ef5a42c96e461...
7	7	1679584111033	ebbb0157d8d753b778dfc749baccda7d3760f008468a737ab53143fdae9...
8	8	1679584338812	ad3c36396a8546714502c1e6f6d0be67f6897ee2cab711ac49ec9e355d...
9	9	1679584430780	6a6b76ee0205147d5726f4734d0b8e23fd6d80c2aef93cb7bf7065c429d...
10	10	1679585206593	10ab9a50f268ad118432b21a3aa775ea82b4b72c84fe1f0244aedf35ee3...
11	11	1679585208223	ebf6637660a553ef5fc8ea190795c08bcccc3d373f6d674ce7e93fd113c6...
12	12	1679585208276	6e0f936089d9c5acd75e3f74e99fc140b6241c060c570f8a935fb7b656a5...
13	13	1679585208320	4ca7c070388cf5a4ad25eafd892a6df989a6a2ef3f4e40102e783ebb7de...
14	14	1679585208407	880cea53aeabc0b8ed9d96cbe7cbe10f562ae4ee4a9520bfb9f06a1edbe...
15	15	1679585208488	681d9a0facbf20d6db8862addd94ff33ed923cdca3a28e85aff775da7767...
16	16	1679585208535	17cbf6fdd715769b7de0be554f28057b4fc04aab38f980807278836c0ce0...
17	17	1679585426087	67305ebe616a209fbcd61717e3ec5502a56bbb42fd506a17bc60ba2a7a...

자료 25 타임스탬프와 이를 바탕으로 생성된 AES 키를 담고 있는 AgentK 테이블

과정에 따르면, 데이터베이스와 AES 의 IV 값에 접근할 수 있을 때 데이터베이스에 저장된 어떤 데이터 든 복호화 할 수 있다는 결론에 다다르게 됩니다.

이 과정의 예시를 보여드리겠습니다:

1. ConfigInfo 테이블의 ID 4 의 값을 복호화 및 디코딩하려 할 때, 저장되어 있는 타임스탬프 값을 가져옵니다(자료 26).
2. 가져온 타임스탬프 값을 가져와 AgentK 테이블의 name 컬럼에서 검색합니다. 이 때 32 바이트 길이의 AES 키 값을 얻을 수 있습니다.
3. base64 값을 디코딩하고 이전 과정에서 얻은 AES 키와 IV(민감한 내용일 수 있어 보고서에서는 다루지 않겠습니다.)를 이용해 복호화 하여 원래 값을 얻을 수 있습니다.

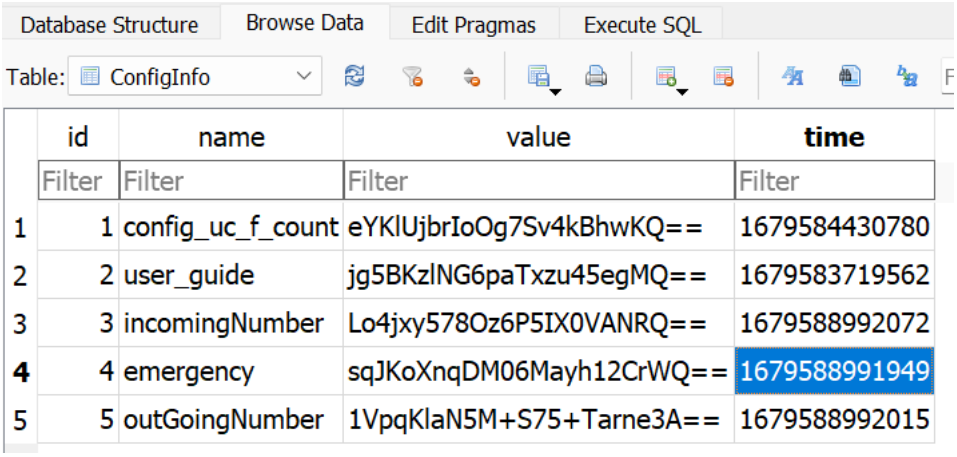
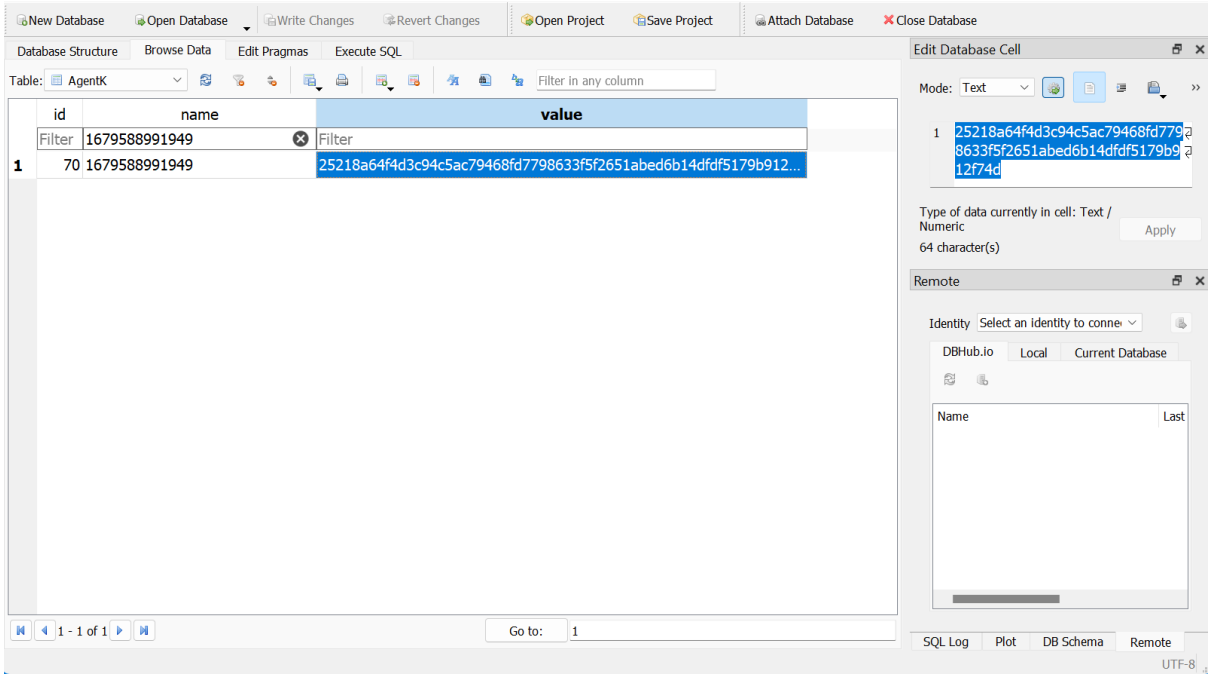

Database Structure | Browse Data | Edit Pragmas | Execute SQL

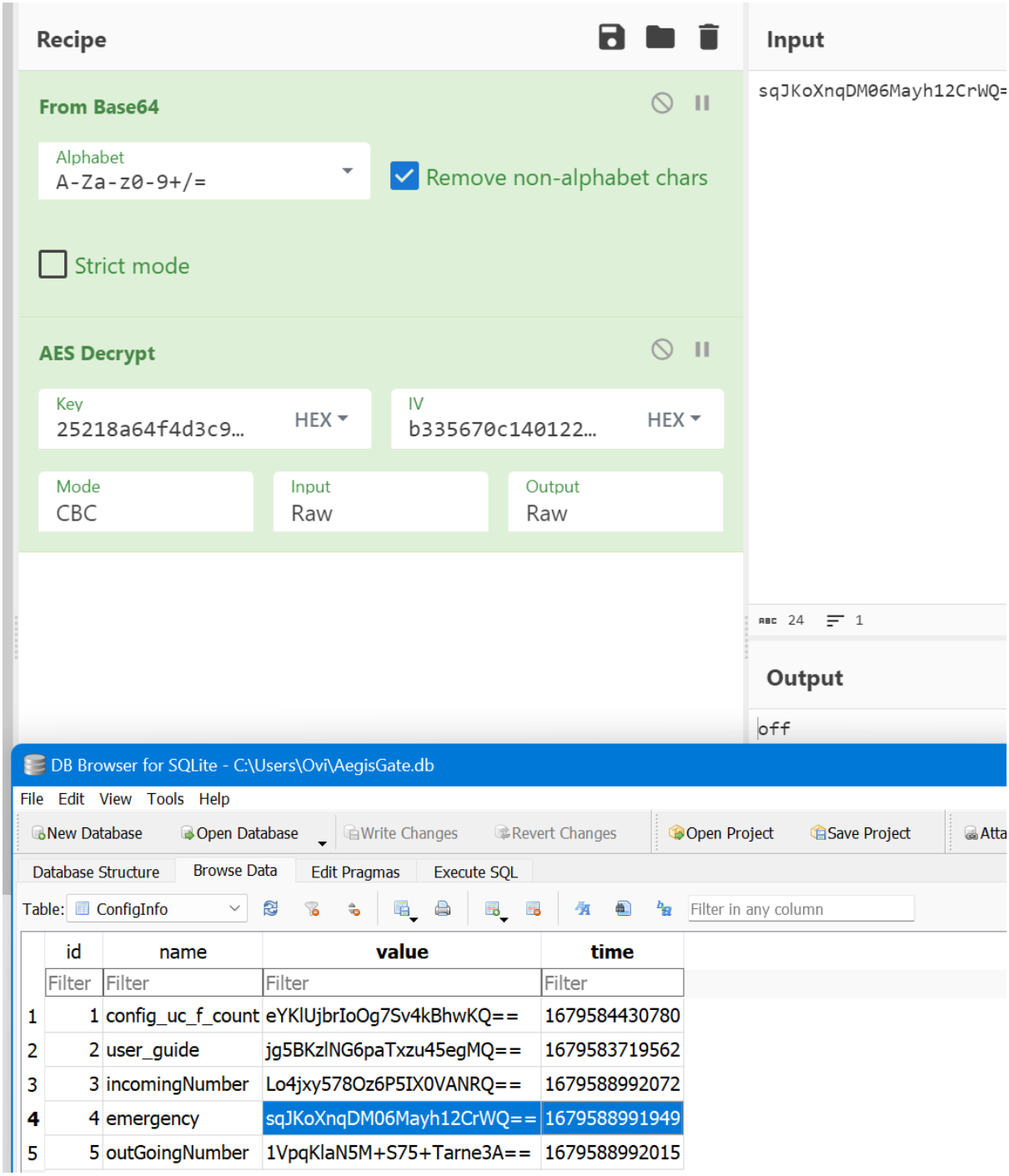
Table: ConfigInfo

	id	name	value	time
	Filter	Filter	Filter	Filter
1	1	config_uc_f_count	eYKlUjbrIoOg7Sv4kBhwKQ==	1679584430780
2	2	user_guide	jg5BKzlNG6paTxzu45egMQ==	1679583719562
3	3	incomingNumber	Lo4jxy578Oz6P5IX0VANRQ==	1679588992072
4	4	emergency	sqJKoXnqDM06Mayh12CrWQ==	1679588991949
5	5	outGoingNumber	1VpqKlaN5M+S75+Tarne3A==	1679588992015

자료 26 Emergency 값에 대한 ConfigInfo 데이터 타임스탬프



자료 27 타임스탬프 값을 이용한 AES 키 획득



자료 28 AgentK 에 저장된 키를 사용한 emergency 값의 복호화 과정

데이터베이스에 저장된 데이터가 꼭 민감한 정보는 아닐 수 있어도, 그 중 일부는 (관리자 전화번호, 수신/발신 전화번호 등) 충분히 민감한 내용을 담고 있었습니다. 이 앱의 소스 코드 리뷰 및 분석 과정에서는 이 기능이 구현된 이유를 찾을 수 없었기 때문에, 이 내용을 보고서에 포함하기로 결정했습니다. 앱 자체에는 데이터베이스 파일에 저장된 데이터를 사용하는 기능이 없었기 때문에, 이러한

데이터 저장은 개발자들의 사용을 위해 구현된 것으로 보입니다. 데이터를 복호화 할 때 사용될 수 있는 AES 키가 데이터베이스에 저장되는 것 역시 이 이러한 추측에 힘을 싣고 있습니다. 앱의 런타임 중 복호화 할 일이 없다면, AES 키가 데이터베이스에 저장되는 이유에 의문이 제기될 수 있습니다. 이 때문에, 보고서에 담긴 내용의 중요성은 독자 분들의 판단에 맡기도록 하겠습니다.

5_ SECURITY

전반적인 앱의 보안은 좋은 편입니다. 보안 취약점 테스트를 진행하며 사용자의 기기에 위험을 초래하는 심각한 문제는 찾을 수 없었습니다. **그러나, 앱의 직원 버전과 외부인 버전에서 공격자가 기기에 접근할 수 있을 때 개인정보 유출을 초래하는 두 보안 취약점을 찾을 수 있었습니다.** 공격자는 해당 취약점을 이용해 앱의 로그 정보를 추출하고 4.4 장에서 언급된 GPS 위치정보를 조회할 수 있습니다. 인터랩은 2023 년 3 월 10 일 국방부에 위 취약점을 알리고, 이 보고서를 통해 취약점을 공개하기까지 30 일의 기간(수정이 간단한 문제였기 때문입니다.)을 두었습니다. 아직 국방부로부터의 답은 없었으며, 이 보고서를 공개하는 시점에도 해당 취약점은 여전히 존재합니다

.

5.1_ Vulnerable broadcast receivers

이 취약점은 다음 버전에 존재합니다:

- 국방모바일보안(외부인) 2.1.17
- 국방모바일보안(직원) 2.1.25

자료 12 에서 보셨듯, kr.go.mnd.mmsa.of (직원)과 kr.go.mnd.mmsa.vt (외부인)에는 "BroadcastReceiverExternal" 브로드캐스트 리시버가 외부로 내 보내집니다. Drozer 를 사용했던 자료 13 에서 보였듯이 브로드 캐스터 리시버는 누구든 기기에 접근할 수 있다면 호출할 수 있고, 권한을 요구하지 않습니다. 'android:exported="true"' 값을 "false"로 변경하여 이 취약점을 고칠 수 있습니다. 이 브로드캐스트 리시버는 “com.markany.aegis.AEGIS_ACTION_ADMIN_REQUEST” 인텐트 필터를 가지고 있으며, 이 인텐트가 포함된 브로드캐스트를 받을 때 브로드캐스트에 문자열 값 ”action_admin”과 ”action_admin_exportLog”이

포함되는지 확인합니다. 두 문자열이 모두 포함될 경우, 이 서비스는 앱이 기록한 모든 로그를 /storage/emulated/0/Aegis/ 경로로 추출합니다. 위 내용은 다음의 자료 29 에서도 확인하실 수 있습니다.

```
public final void e(Intent intent) {
    String stringExtra = intent.getStringExtra("action_admin");
    Intent intent2 = new Intent();
    intent2.setAction("com.markany.aegis.AEGIS_ACTION_ADMIN_RESPONSE");
    intent2.addFlags(268435456);
    intent2.putExtra("extra_result", "1");
    if ("action_admin_release".equals(stringExtra)) {
        g();
    } else if ("action_admin_exportLog".equals(stringExtra)) {
        String f = ud.f(this, "");
        if (f == null) {
            return;
        }
        if (!ud.d(f + "/Aegis")) {
            ud.c(f + "/Aegis");
        }
        File[] g = ud.g(ud.h(this, "/MobileSticker/log/"));
        if (g != null) {
            for (File file : g) {
                ud.b(file.getPath(), f + "/Aegis/exportLog_" + file.getName());
            }
            be.X(this, "로그파일을 추출했습니다.\n경로-" + f);
        }
    } else {
        intent2.putExtra("extra_result", "0");
    }
    sendBroadcast(intent2);
}
```

자료 29 Broadcast Receiver intent filter method

액션 "com.markany.aegis.AEGIS_ACTION_ADMIN_REQUEST"와 문자열 "action_admin", "action_admin_exportLog"를 모두 포함하는 브로드캐스트 메시지를 만들었을 때, 4.2.1 에서 다룬 ServiceAEGIS 가 활성화되어, 모든 앱의 로그를 /storage/emulated/0/Aegis 로 추출하게 됩니다.

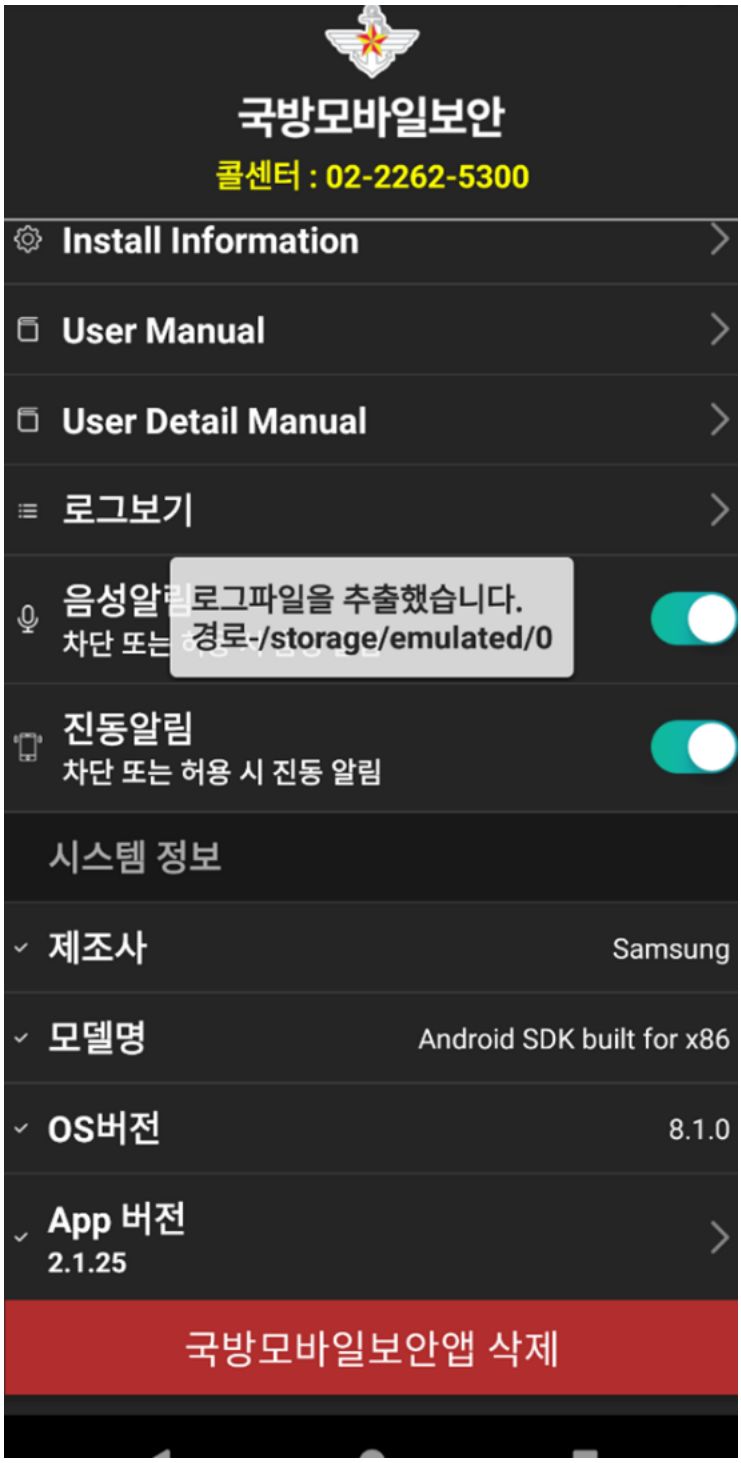
Drozer 에서 다음과 같은 간단한 명령어로 작동시킬 수 있습니다:

```
run app.broadcast.send --action
com.markany.aegis.AEGIS_ACTION_ADMIN_REQUEST --component
kr.go.mnd.mmsa.of kr.go.mnd.mmsa.of.br.BroadcastReceiverExternal --extra string
action_admin action_admin_exportLog
```

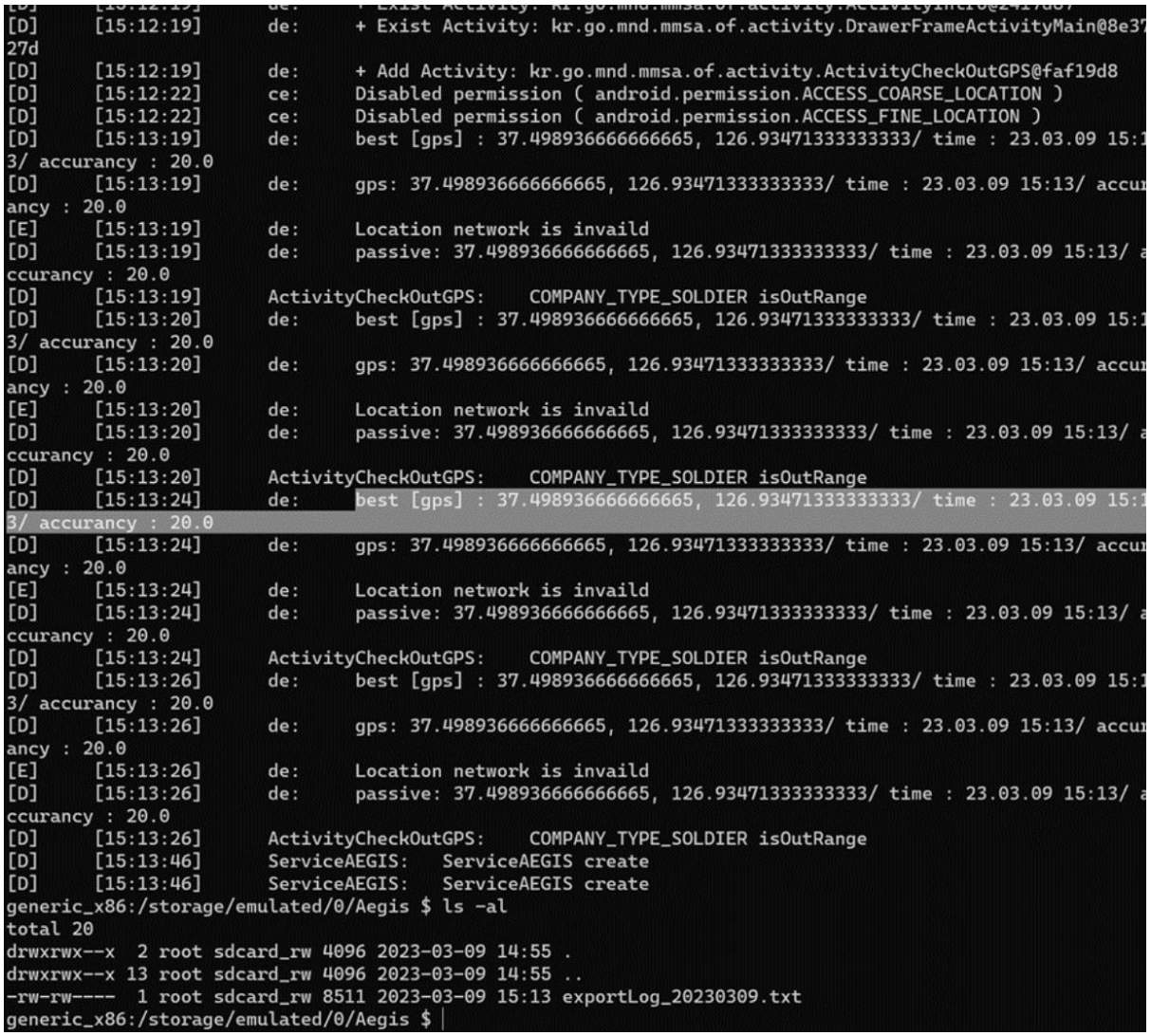
위 명령의 결과로 로그가 기기에 물리적으로, 또는 원격으로 접근할 수 있는 누구든 관리자 권한 없이 사용할 수 있는 저장소 디렉토리에 내 보내지게 됩니다. 이 때 자료 30 에서와 같이 앱 자체적으로 사용자에게 로그 파일이 내 보내졌다는 알림을 띄우게 됩니다.

저장소 디렉토리로 추출된 로그의 예시는 자료 31 에서 확인하실 수 있습니다. 이 예시에는 4.4 장에서 다룬 대략적인 GPS 위치와 시각 정보가 담긴 로그 또한 포함되어 있습니다.

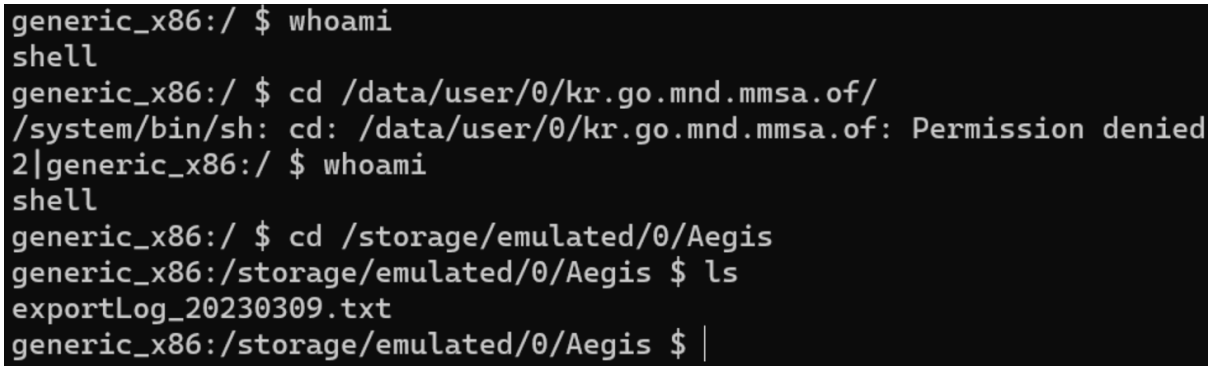
취약점에 관해 좀 더 구체적인 설명을 드리자면, 공격자가 앱에서 직접 로그를 추출하려 했다면 관리자 권한이 없기에 가능하지 않았을 것입니다. 자료 32 를 참고하시면, /data/user/0/kr.go.mnd.mmsa* 경로의 파일에 접근하는 것은 권한이 부족하여 실패하게 됩니다. 그러나 앞서 설명했듯이, 로그 데이터를 저장소 디렉토리에 추출하는 것은 관리자 권한 없이도 가능합니다.



자료 30 로그를 추출할 때의 알림 메시지



자료 31 GPS 위치 등이 포함된 추출된 로그들



자료 32 앱 로그와 추출된 로그의 권한 차이

6_ REFERENCES

[1] Unknown, "Namu," [Online]. Available: https://namu.wiki/w/%EA%B5%AD%EB%B0%A9%EB%AA%A8%EB%B0%94%EC%9D%BC%EB%B3%B4%EC%95%88. [Accessed 3 March 2023].

[2] M. Hyeong-chul, "MetroSeoul," [Online]. Available: https://www.metroseoul.co.kr/article/20220106500181. [Accessed 10 3 2023].

[3] "Apple App Store," Apple, [Online]. Available: https://apps.apple.com/kr/app/%EA%B5%AD%EB%B0%A9%EB%AA%A8%EB%B0%94%EC%9D%BC%EB%B3%B4%EC%95%88/id1485356492?see-all=reviews. [Accessed 10 3 2023].

[4] Mitre. [Online]. Available: https://cwe.mitre.org/data/definitions/532.html. [Accessed 25 4 2023].

[5] law.go.kr. [Online]. Available: https://law.go.kr/LSW/lsInfoP.do?lsiSeq=183644&viewCls=engLsInfoR&urlMode=engLsInfoR#0000. [Accessed 25 4 2023].