

# Scarcraft Bolsters Arsenal for targeting individual Android devices

By S2W

Published: 2023-03-27 · Archived: 2026-04-05 19:58:37 UTC



**Author:** BLKSMTH | S2W TALON

*Last Modified: Mar 23, 2023*

Press enter or click to view image in full size

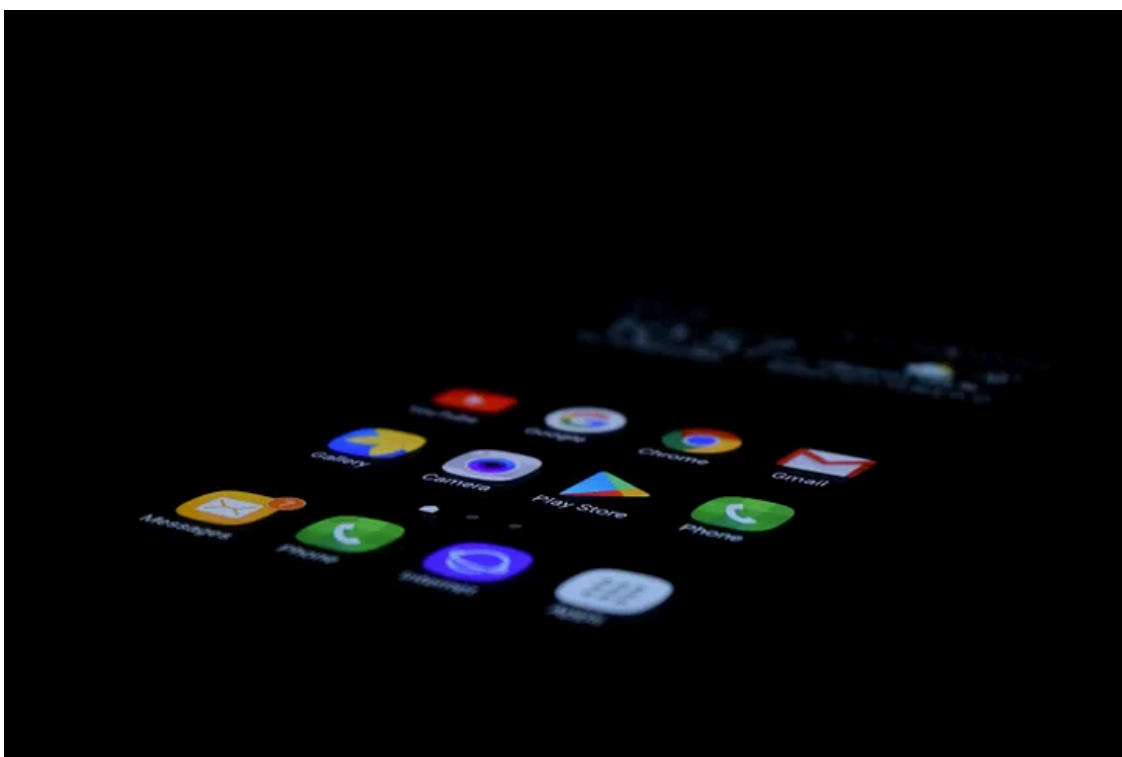


Photo by [Pathum Danthanarayana](#) on [Unsplash](#)

## Executive Summary

- According to an analysis report published by InterLab in December 2022, a South Korean journalist received a message requesting a conversation via the Wechat messenger, and the requestor instructed the journalist to install a malicious APK file disguised as a messenger called “Fizzle.apk” — InterLab named the malicious APK “RambleOn”
- We found similar features and codes to the mobile version of the ROKRAT malware that the [Scarcraft](#) group has been using since 2017.

- In tracking the Scarcraft group, researchers within S2W's Talon have identified additional samples that perform similar functions to those disclosed in this release, with significant upgrades from previous releases.
- Scarcraft is strongly believed to conduct initial penetration by contacting individuals directly via messengers, such as in this case, to trick them into installing a malicious APK disguised as legitimate.
- S2W Talon named "**Cumulus**" in reference to past samples similar to the type of malware disclosed by InterLab, and named the plugin used by Cumulus as "**Clugin**".
- There are three types of Cumulus, depending on whether or not the Clugin is downloaded and the type of messaging service used.
- We observed that the Scarcraft group updated the malware's functionality or installed China-specific applications on test devices to target users with Chinese language and Chinese-manufactured mobile devices.

## Introduction

The Scarcraft Group (aka APT37), a North Korean APT group, is believed to have been active since 2016 and continues to carry out attacks against institutions and political organizations around the world until 2023. In April 2017, the Cisco Talos team disclosed the Scarcraft group's proprietary tool, ROKRAT, a malware that has been continuously modified and used by the group to this day. Initially, only the Windows version of ROKRAT was used, but the Android version of the malware was later identified.

According to a report published by the Financial Security Institute, the Scarcraft group conducted an attack in mid-2017 that distributed malicious APKs to specific devices through a watering hole attack. At the end of 2017, the group also carried out an attack campaign targeting North Korean human rights organization officials and journalists from North Korean media outlets to induce the installation of malicious APKs through KakaoTalk, the most popular messenger in South Korea. In addition, malicious APKs were also distributed by contacting targets through Facebook and uploading APKs to the Google PlayStore. The malicious apps were all identified as mobile versions of ROKRAT.

- Reference: [\[FSI\] Threat Intelligence Report - Campaign DOKKAEBI](#)

According to an [analysis report published by InterLab in December 2022](#), during a conversation with a South Korean journalist via Wechat messenger, the Scarcraft group convinced him to install a malicious APK file disguised as a messenger called "Fizzle.apk", saying that he could not send sensitive files via Wechat messenger. InterLab named the malicious APK "RambleOn", but analysis of the malicious APK revealed similarities to the Scarcraft group's ROKRAT mobile version. Unlike in the past, the APK has the ability to receive data from the attacker via a messaging service called Pushy.

In following the Scarcraft group's trail, Talon, S2W's threat research and intelligence center, identified additional samples that perform similar functions to the published samples. They have similar functionality to the malicious APKs released in 2017, but unlike in the past, the ability to use messaging services has been added. We also found that these APKs have been continuously updated to date. S2W Talon named the malicious APKs "**Cumulus**" and the plugin modules used by Cumulus "**Clugin**".

In this report, we further categorize the identified Cumulus by type and describe the attacker's TTPs and strategy based on our detailed analysis.

## Overview of Cumulus Types

Cumulus (aka. RambleOn) has been used by the Scarcraft APT group since at least 2019 to target Android devices. The group has been using a mobile version of the ROKRAT malware since at least 2017, and S2W Talon separately classifies Cumulus as a type of existing ROKRAT mobile malware with messaging capabilities such as FCM or Pushy added. Cumulus is usually distributed disguised as a legitimate mobile application, such as a CoinMiner, image viewer, or messenger. Although we could not secure more samples, we have also seen them distributed under the package names "com.personal.info", "com.sec.mishat", and "com.data.person". Based on the types of applications Cumulus disguises, we suspect that it is distributed directly to individuals via messengers, such as this RambleOn type.

Press enter or click to view image in full size

App Name	축하통보문	Threema Work	PhotoSecViewer ThreemWork	FreeCoinMiner	Fizzle
App Icon					
Distribution Period	at least end of 2019	at least early 2020	at least early 2022	September, 2022 (for test)	at least end of 2022
Package Name	com.greet.mess agefree	com.threema.wor kfree	com.data.wecoin	com.app.freecoin miner	ch.seme
Type	Type A	Type A	Type B	TEST	Type C
Messaging	FCM (No use)	FCM	FCM	FCM	Puhsy
Device Token	Cloud	Cloud	Firebase Database	Cloud	Cloud
Cloud	Yandex	Yandex	-	pCloud	Yandex pCloud

Table 1. Types of Cumulus

After obtaining additional Cumulus disguised as legitimate applications and analyzing them, we were able to categorize them into three types, as shown below. Types B and C download a separate plugin and perform their main actions in the plugin, which is why we named the plugin downloaded by Cumulus as **Clugin**.

Press enter or click to view image in full size

	Type A	Type B	Type C
Download Clugin	X	O	O
Download Command	O (Cumulus)	O (Clugin)	O (Clugin)
Download CallRecorder	O (Cumulus)	O (Clugin)	O (Clugin)
Messaging	FCM	FCM	Pushy

Table 2. Type Classification

- **Clugin:** Plugin that Cumulus downloads from the cloud and is responsible for information leakage.
- **Command:** Configuration file that a Clugin or Cumulus downloads from the cloud to execute commands.
- **CallRecorder:** An additional Dex file that the Clugin or Cumulus downloads from the cloud to perform call recording functions.

**Type A** downloads and loads the **Command file**, which contains the configuration information necessary to perform the malicious behavior, and **CallRecorder** from the cloud. It then uploads the infected device information and internal files to the cloud. It receives a separate message from the attacker via FCM.

**Type B** downloads the **Clugin** from the cloud. Clugin takes over the functions of Cumulus, downloads **Command file** and **CallRecorder**, steals and uploads information to the cloud. Compared to Type A, by introducing Clugin, Type B organizes modularization by function and secured stable persistence and malware update function. Same as Type A, Type B receives a separate message through FCM.

**TEST** seems to be used by the attacker for testing before the attack and uploads the infected device information and internal files to the cloud without downloading any additional files. Cumulus, which is used in real-world attacks, uses abbreviations to upload each exfiltration data to the cloud, but in the case of TEST, the full word is used for ease of identification during the test.

**Type C** has most of the same features as Type B, but uploads the Device Token to the cloud instead of the Firebase Database and receives messages from the attacker via **Pushy**.

FCM is a service that specializes in message delivery within [Firebase and was also used in the mobile malware used by the Kimsuky group that we disclosed last year](#). The difference is that the Device Token is sent to the cloud or a legitimate Firebase database, rather than to an attacker's C&C server. The most recent version of the Pushy service is a separate third-party service that provides similar functionality to FCM.

## TimeLine

Press enter or click to view image in full size

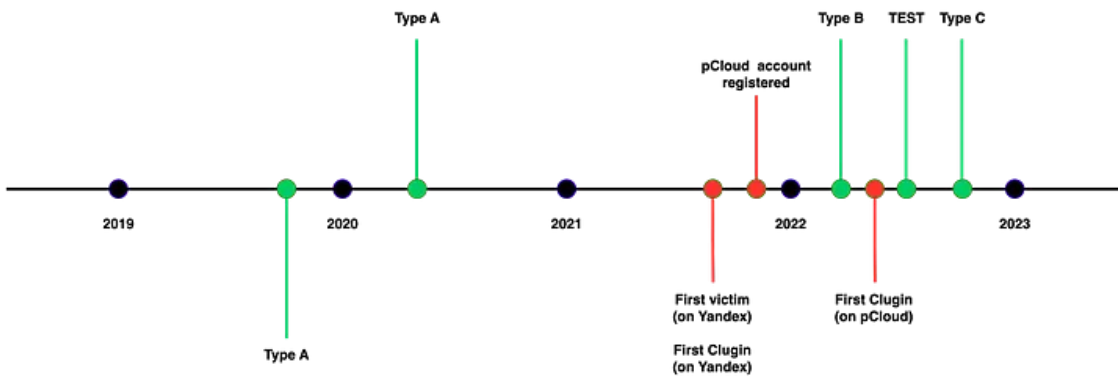


Figure 1. Full Timeline for Cumulus

Clugin appears to have been uploaded to Yandex Cloud and distributed since at least September 2021. Although we do not have an exact date for the creation of Yandex Cloud, we believe that Clugin distribution began around that time. The pCloud account was subsequently created in October 2021, but the data exfiltration we identified was from March 2022. Given that Type B was distributed in March 2022, we believe that the attacker began distributing Clugin via pCloud in a similar way. The attacker appears to have initially distributed Clugin through Yandex Cloud, and then, starting in March, configured it to communicate with pCloud on initial infection and only communicate with Yandex Cloud when passing a separate command. TEST is believed to be a test version to introduce this. The Scarcraft group appears to have set the OAuth key for pCloud communication differently for each distributed APK but kept the OAuth key for the Yandex cloud relatively unchanged.

### 1. Behavior flow for Type A

After infection, **Type A** registers a method to JobScheduler to periodically execute the main malicious behavior. It then downloads a Command file from the Yandex cloud and steals information as specified in the Command file. It additionally downloads and loads a CallRecorder, which performs call recording and saves it to a file.

The collected infected device information and internal files are uploaded to Yandex, which also transmits the device token for FCM communication, allowing the attacker to obtain the Device Token of the infected device from the Yandex cloud. The attacker can use the obtained Device Token to send a message to the infected device via FCM, and Cumulus, which receives the message, checks whether the method that performs the malicious behavior is registered in the JobScheduler and registers it if it is not.

Press enter or click to view image in full size

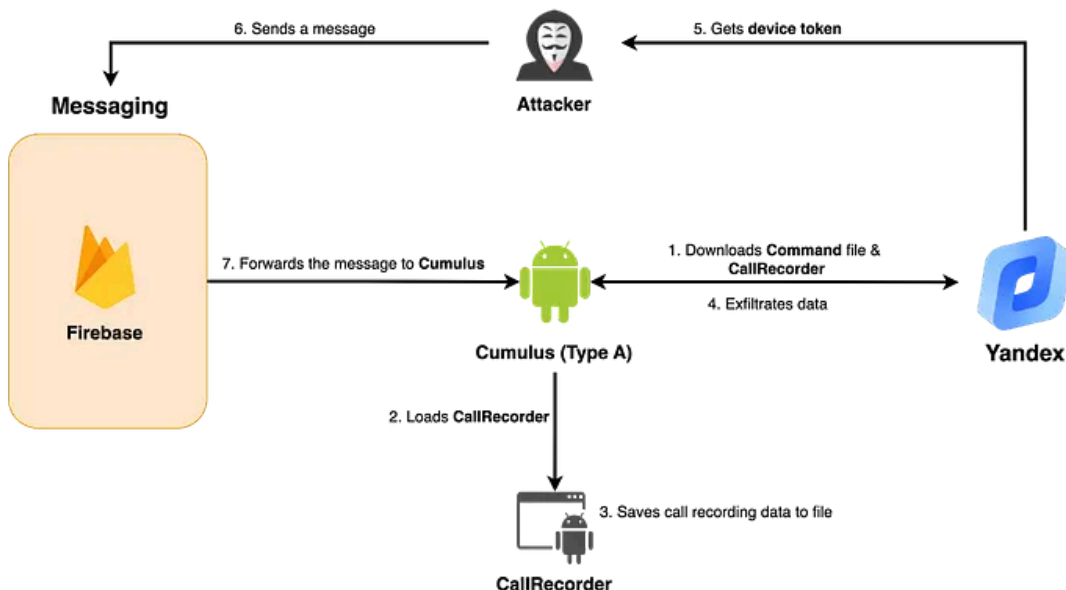


Figure 2. Execution flow of Type A

## 2. Behavior flow for Type B

Cumulus in **Type B** downloads the Clugin from the cloud, then the Clugin downloads CallRecorder, and steals the infected device information and internal files. In addition, Type B receives messages via FCM, adding update functions such as changing cloud storage and changing OAuth Token.

When executed, Type B first sends the Device Token to the Firebase Database. With the sent token, the attacker passes the OAuth Token and the cloud REST API through FCM, which is presumably used to download the Clugin from the cloud. At the time of analysis, we were unable to obtain actual data from FCM, but based on the internal code of Type B, we believe that it is downloaded from Yandex Cloud.

Press enter or click to view image in full size

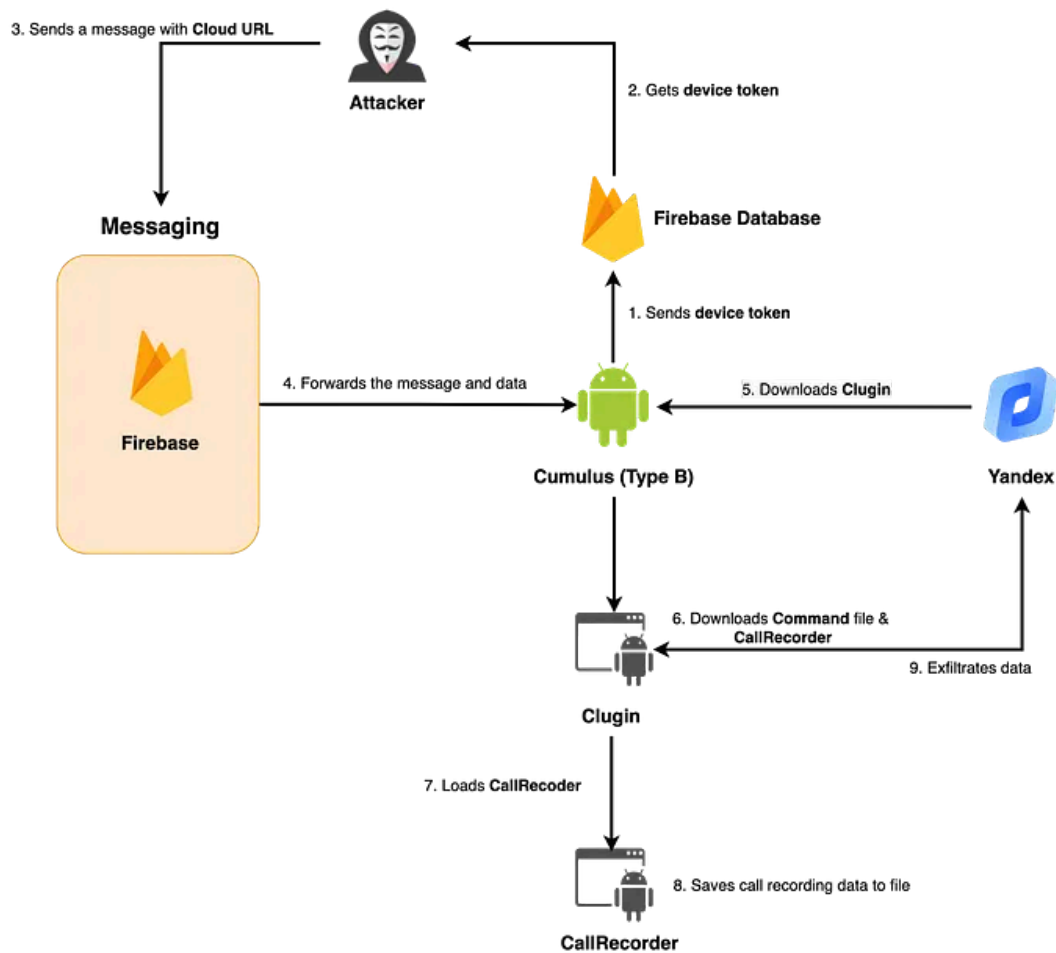


Figure 3. Execution flow of Type B

### 3. Behavior flow for TEST

In the case of **TEST**, when the APK is executed, it steals information such as infected device information, SMS, contacts, internal files, and recordings and uploads them to the pCloud. Although **TEST** includes Yandex Cloud's OAuth Token, it actually uses only pCloud's OAuth Token initialized within the pCloud SDK class and does not use the Yandex Token. In addition, there is no function to send the device's Device Token separately, so we assume that Type A is for testing purposes only. The string "test-pi-d9b7e" is used in the code to initialize Firebase functionality, and the functionality is incomplete compared to other types, suggesting that the attacker used this type for testing.

Press enter or click to view image in full size

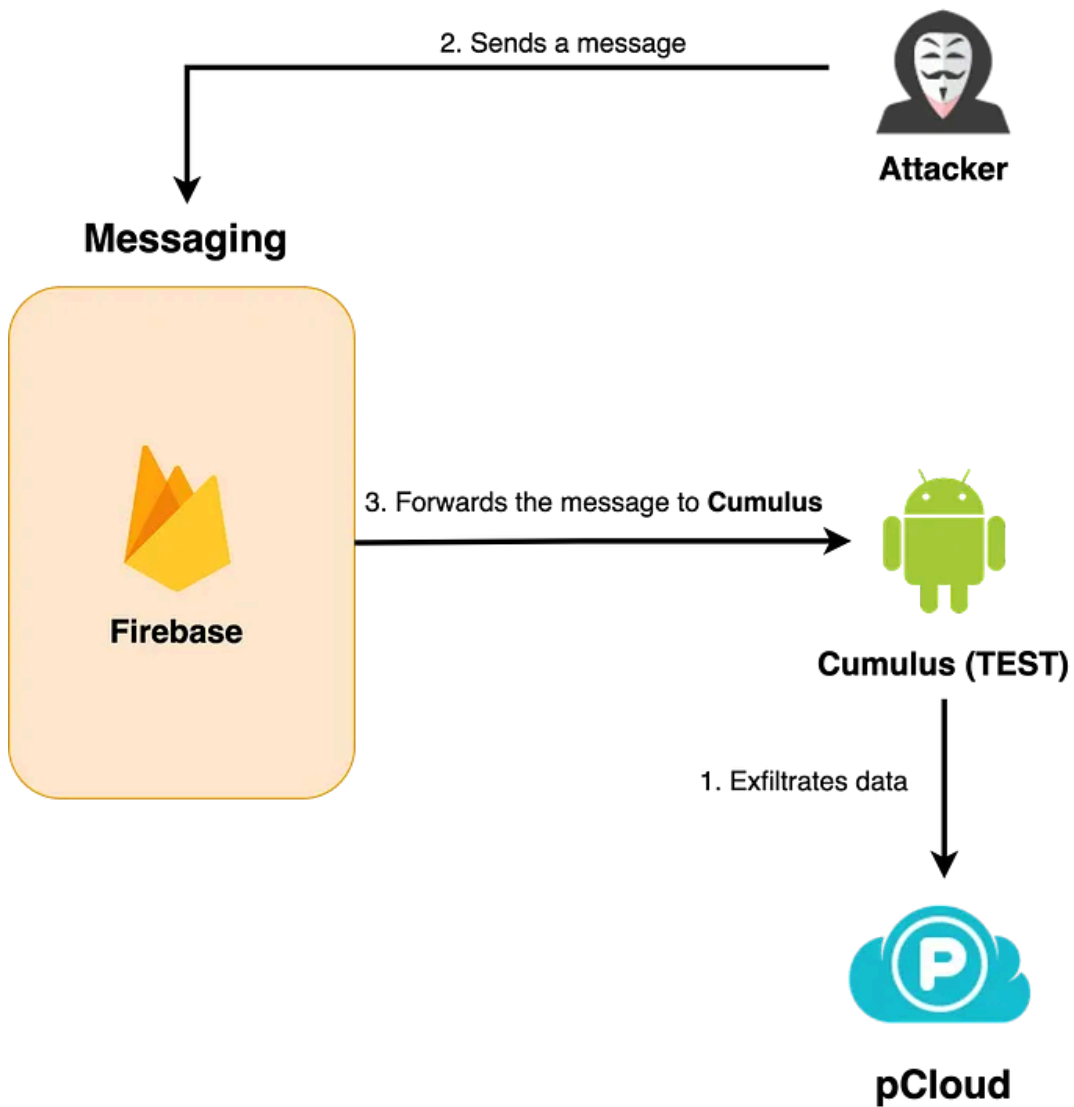


Figure 4. Execution flow of TEST

#### 4. Behavior flow for Type C

**Type C** sends messages to Cumulus via a third-party messaging service called Pushy rather than FCM. Type C has both hardcoded pCloud and Yandex's OAuth Token values, and an attacker can update the type of cloud service and OAuth Token via Pushy.

Press enter or click to view image in full size

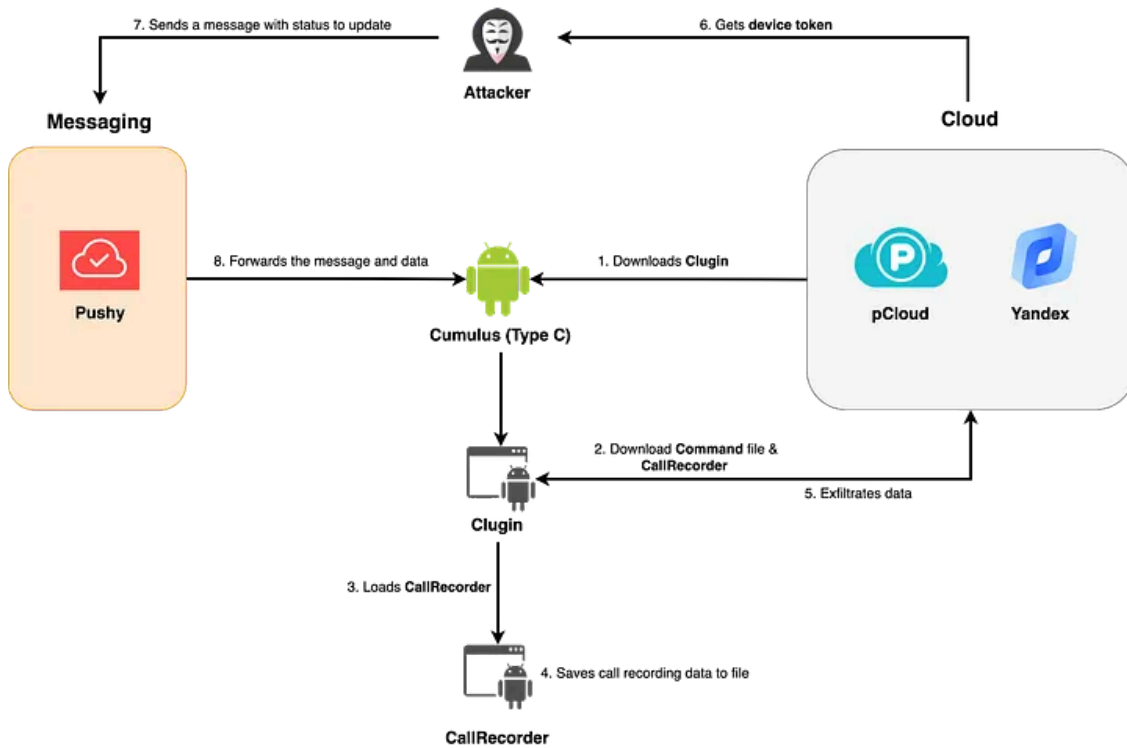


Figure 5. Execution flow of Type C

## Detailed Analysis

We conducted a detailed analysis of a messenger impersonation APK called "Fizzler" (named RambleOn by Interlab) and Clugin version 6.0, which is classified as **Type C of Cumulus types**. Below is the entire execution process of a Type C Cumulus.

Press enter or click to view image in full size

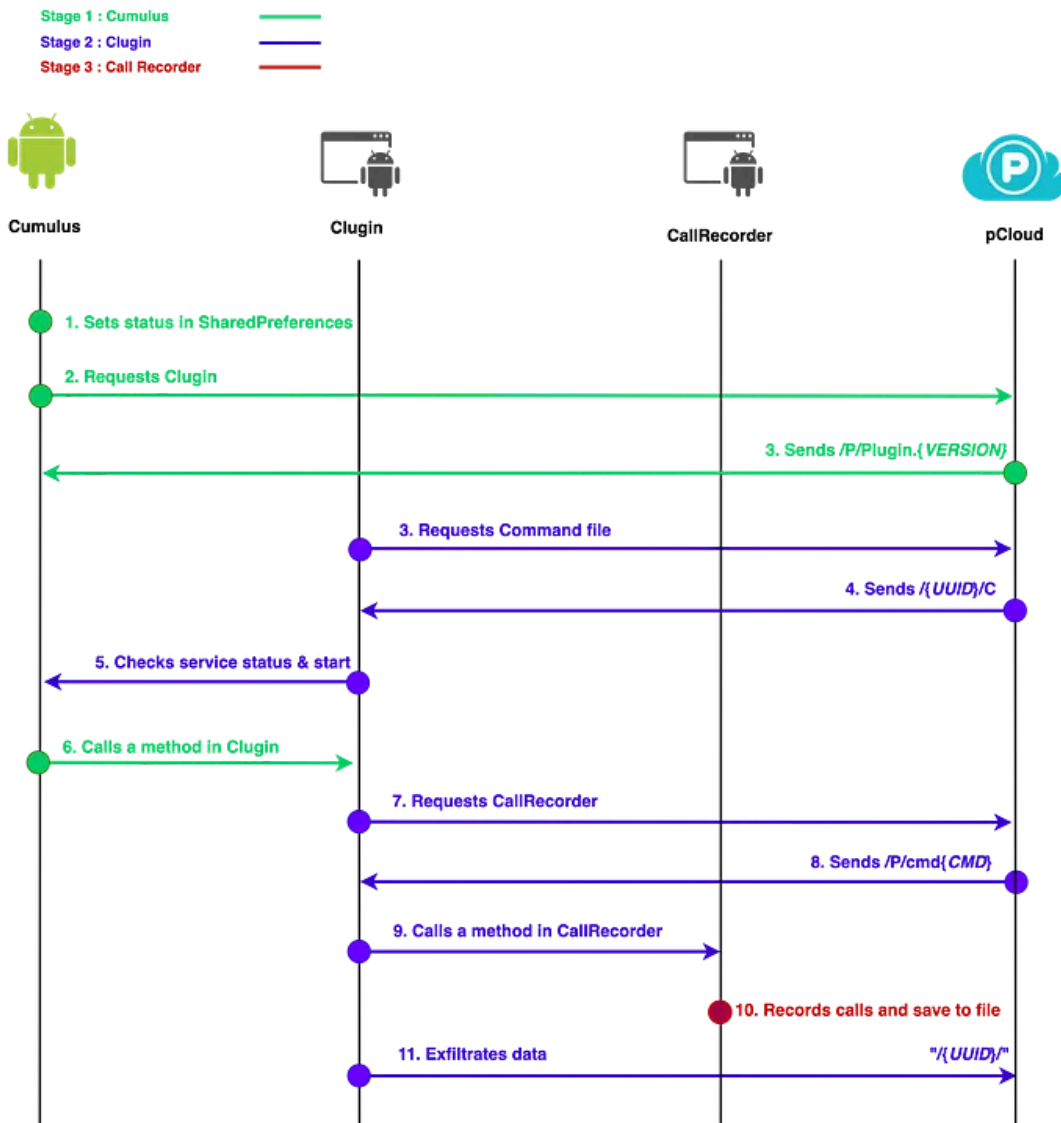


Figure 6. The communication scenario of Cumulus

## Stage 1: Cumulus (Fizzle.apk)

### 1. Status in SharedPreferences

Cumulus manages the status with SharedPreferences and references it to perform its malicious behavior. The *UUID* or *TID* in the status is used as an ID to identify the infected device. Initially, it uses the *UUID*, but if it subsequently receives a message from the attacker via Pushy, it changes the ID to the *TID* contained in the message instead of the *UUID*. Then, store the Device Token for receiving messages from Pushy in *PUSHYT* and set *CLOUD* to P to communicate with pCloud. The OAuth Token required for cloud communication is specified in *PRIMARY\_ACCESSTOKEN*. Also, set the Clugin version to *VERSION* to request the *Plugin{VERSION}* file to the cloud, and set the download success to 1 or 0 in *PLUGININDEXDOWN{VERSION}*. *CLOUD* is only supported for P (pCloud) and Y (Yandex), and is set to P on the first run.

Press enter or click to view image in full size

Name	Description	Value
UUID	Unique ID	Random value
TID	Unique ID	Initialized later by Pushy
PUSHYT	Pushy device token	Device Token
CLOUD	Type of cloud	"P" (Initialized to "Y" by pushy)
PRIMARY_ACCESSTOKEN	Cloud OAuth token	OAuth token for pCloud (Initialized to Yandex's by pushy)
VERSION	Plugin version	4.0
PLUGININDEXDOWN{VERSION}	Flag for successful download (1: Success / 0: Fail)	1 (After downloading Clugin)

Table 3. Values in status

## 2. Download Clugin from Cloud

Cumulus references the status to download the Clugin in Dex form from the cloud service. Since the cloud identifies infected devices by their UUID or TID values, it is possible to install a different Clugin for each device. After downloading, it calls the LogState method of the com.personal.info.plugin class.

- Clugin path on first run (on Cloud): /P/plugin{VERSION}
- Clugin storage path (on infected device): ch.seme/Files/.temp/plugin{VERSION}.dex

Press enter or click to view image in full size

```
try {
    File file1 = logUserService1.getDir("pluginindex", 0);
    DexClassLoader dexClassLoader1 = new DexClassLoader(LogUserService.this.pluginDexPath, file1.getAbsolutePath(), null, LogUserService.this.getClassLoader());
    Constants.classloader = dexClassLoader1;
    Class class1 = dexClassLoader1.loadClass("com.personal.info.plugin");
    Constants.pluginCls = class1;
    Constructor constructor1 = class1.getConstructor(Context.class);
    Constants.pluginConstructor = constructor1;
    Constants.pluginObj = constructor1.newInstance(LogUserService.this.getApplicationContext());
    Method method2 = Constants.pluginCls.getMethod("LogState");
    while(true) {
        label_229:
        method2.invoke(Constants.pluginObj);
        break;
    }
}
```

Figure 7. Downloads Clugin and invoke

## Stage 2: Clugin (DEX)

Cumulus downloads and executes Clugin in the form of plugin from the cloud. In this process, we were able to collect samples of different versions of Clugin, between 1.0 and 6.0. After analyzing each version of the Clugin, the table below summarizes the versions and features that we found to have noticeable changes.

Press enter or click to view image in full size

Plugin Version	2.1	2.2	3.0	6.0
<b>MD5</b>	1F2C23C7C9ECB28B FDC6627A3AD23783	97A9AB76AF215241A D2A07856B40242E	759B26631A660D82F 6A93621991C4292	72182F83E771FCAA A1E86C7C932014CB
<b>Cumulus package name</b>	com.data.person	com.data.wecoin	com.sec.mishat	com.sec.mishat
<b>Class</b>	Build Config R plugin	Build Config R plugin <b>DocumentFileMeta</b> <b>SAFTools</b> <b>Storage</b> <b>Storage11</b>	Build Config R plugin DocumentFileMeta SAFTools Storage Storage11 <b>pCloud</b>	Build Config R plugin DocumentFileMeta SAFTools Storage Storage11 pCloud
<b>Cloud</b>	Yandex	Yandex	Yandex, <b>pCloud</b>	Yandex, pCloud
<b>Library</b>	dagger net	dagger net	<b>okhttp3</b> <b>okio</b> net	okhttp3 okio net
<b>Function</b>	Send SMS Send Contacts Send File Structure Send RealTimeInfo Send DeviceInfo Send AppState Send ServiceLog Audio Recording Send External File Data	Send SMS Send Contacts <b>Send Call Logs</b> Send File Structure Send RealTimeInfo Send DeviceInfo Send AppState Send ServiceLog Audio Recording Send External File Data	Send SMS <b>Send MMS</b> Send Contacts Send Call Logs Send File Structure Send RealTimeInfo Send DeviceInfo Send AppState Send ServiceLog Audio Recording Send External File Data	Send SMS Send MMS <b>Send GPS</b> Send Call Logs Send Contacts Send RealTimeInfo Send DeviceInfo Send AppState Send ServiceLog Audio Recording Send External File Data Send File Structure

Table 4. Feature comparison table of Clugin by version

### 1. Download Command file from Cloud

The Clugin reads the Command file from the cloud and performs information theft according to the values set in each field. For each field, the data is specified in the format {Type} : {Key} : {Value}, and the Key and Value are parsed and registered in SharedPreferences. The C(Command) file can be deleted from the cloud after downloading.

- Command file download path: `/{UUID}/C`

Press enter or click to view image in full size



Figure 8. Command file parsing process

Press enter or click to view image in full size

Key	Description
CMD	Command Flag
CTREQNUM	Contacts Request Number
SMSIT	SMS Interval
CLIT	Call Log Interval
GPSIT	GPS Interval
SMSREALTIME	Recent SMS Flag
DEBUGLOG	DebugLog Flag
LOGREQNUM	Log Request Number
AR	Audio Record Flag
DEVICEINFOREQNUM	Device Info Request Number
APPSTATEREQNUM	AppState Request Number
FILEREQNUM	File Request Number
ARST	Audio Record Start Date
ARET	Audio Record End Date
SDPATH	Storage Path
SDKIND	File extension to steal

Table 5. Keys in Command file

The CMD in the Command file determines whether malicious behavior is performed.

- CMD == 0: Do not perform malicious behavior
- CMD > 0: Run the service and send the information after stealing it
- CMD > 10: Download and load CallRecorder

## 2. Interact with Cumulus to execute malicious services

The Clugin checks whether the AR and SDPATH values are set in the Command file and executes the malicious behavior by interacting with Cumulus. If CMD is greater than 0 in the Command file, the Clugin checks to see if a specific service in Cumulus is currently running, and if not, executes the service through an Intent. The service in Cumulus directly calls specific methods in the Clugin to perform the actual audio recording or file collection behavior. In the figure below, Clugin checks whether a service named "com.sec.mishat.{ServiceName}" is running, which is the package name of Cumulus. The reason for this implementation is that the commands are modularized using Clugin, so the version of Clugin can be updated at any time, taking advantage of the fact that Clugin does not depend on Cumulus. The malicious behavior executed in this way is as follows.

- Update Clugin from the cloud
- Audio Record using CallRecorder
- Collect files from external storage

Press enter or click to view image in full size

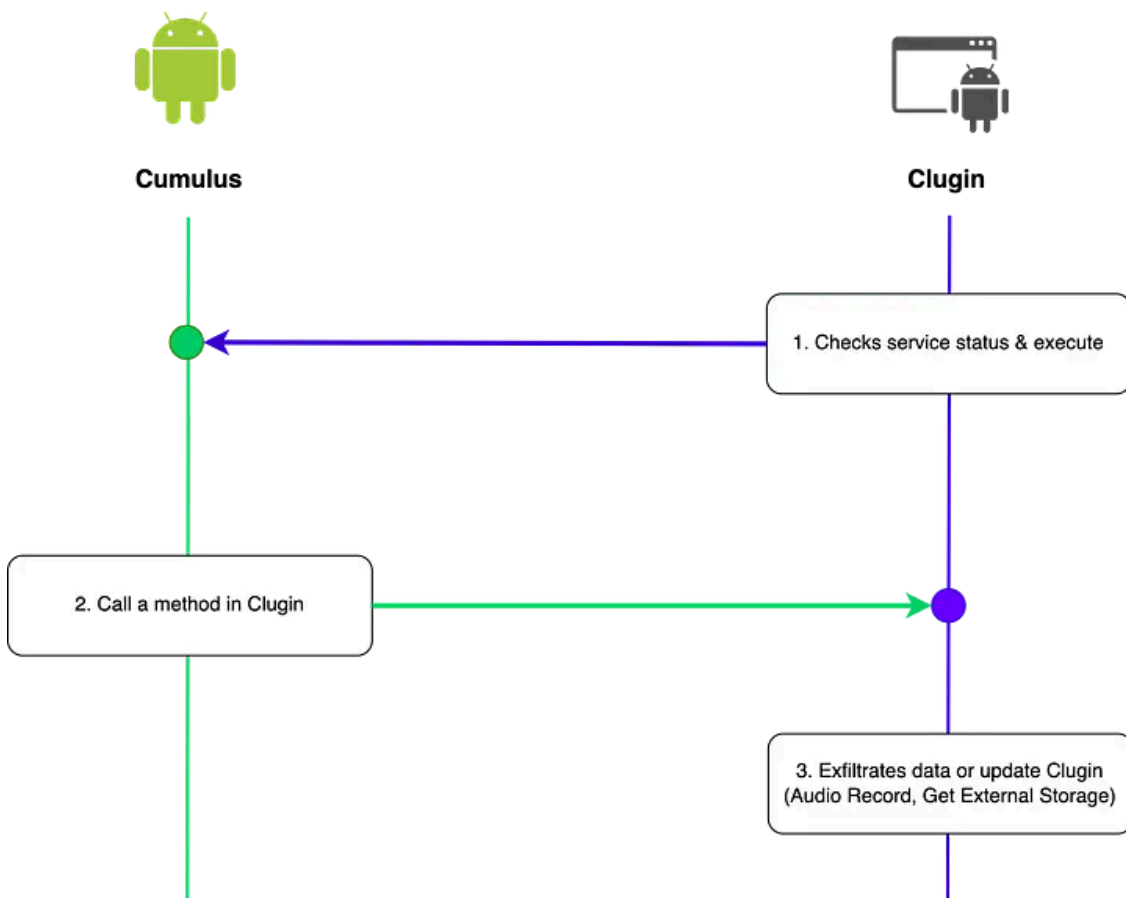


Figure 9. The process of how methods are executed in Clugin

```
if(v > 0 && (!s2.equals("0") && !plugin0.isMyServiceRunning("com.sec.mishat.SyncService"))) {  
    Intent intent1 = new Intent();  
    intent1.setComponent(new ComponentName(plugin0.myContext, "com.sec.mishat.SyncService"));  
    if(Build.VERSION.SDK_INT >= 26) {  
        plugin0.myContext.startForegroundService(intent1);  
    }  
    else {  
        plugin0.myContext.startService(intent1);  
    }  
    plugin.appendLog("SyncService start");  
}
```

Figure 10. Check if a specific service is running in Cumulus

### 3. Download CallRecorder

Cumulus reads the CMD from SharedPreferences and if the value is greater than 10, it downloads an additional CallRecorder from the cloud that performs the call recording function and calls the CallRecorder's "execute" method.

Press enter or click to view image in full size

```
File file1 = new File(s8);
if(sharedPreferences0.getInt(s1 + v2, 0) == 1 && ((file1.exists()) && sharedPreferences0.getInt("CMDEXECUTE" + v2, 0) != 1)) {
    String s9 = plugin0.myContextf.getDir("outdex", 0).getAbsolutePath();
    ClassLoader classLoader0 = plugin0.myContext.getClassLoader();
    Class class0 = new DexClassLoader(plugin0.workDir + s7 + v2 + ".dex", s9, null, classLoader0).loadClass("com.sec.android.acservice.Command" + v2);
    plugin.cmdObj = class0.getConstructor(Context.class).newInstance(plugin0.myContext);
    plugin.execute = class0.getMethod("execute");
    plugin.execute.invoke(plugin.cmdObj);
    plugin.appendLog("dex of command-" + v2 + " executed");
}
```

Figure 11. Downloads CallRecorder and invoke

### 4. Collect & Exfiltration

#### Get S2W's stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

Finally, Clugin collects information from the infected device and sends it to the cloud. Here's what the data is encrypted and how stored in the cloud.

Press enter or click to view image in full size

Data Type	Encrypt	Cloud Path
SMS	O	/{{UUID}}/D/{Timestamp}
MMS	O	/{{UUID}}/D/{Timestamp}
Call Log	O	/{{UUID}}/D/{Timestamp}
Contacts	O	/{{UUID}}/D/{Timestamp}
GPS	O	/{{UUID}}/D/{Timestamp}
Call Record	O	/{{UUID}}/D/{Timestamp}
Audio Record	O	/{{UUID}}/D/{Timestamp}
File Structure		/{{UUID}}/FS/internal.json
Client Info		/{{UUID}}/CI
Phone Info		/{{UUID}}/PI/PI_{Number of requests}
APP Status		/{{UUID}}/AS/AS_{Number of requests}
Job Log		/{{UUID}}/JL/JL_{Number of requests}
External File Data		/{{UUID}}/ED/

Table 6. List of collected data and upload path

For the stolen items stored in the D path on the cloud, encryption is performed before exfiltration, which involves downloading an epk file containing the encryption key from the cloud. The file data is then AES decrypted and Base64 decoded with hardcoded values in Clugin to extract the RSA public key. Each collected file is then encrypted by randomly generating an AES secret key, and the secret key is encrypted with the extracted RSA public key. Finally, the encrypted file data is stored along with the encrypted AES secret key, length of encrypted AES secret key, Custom Path, and length of Custom Path. If the RSA public key does not exist, the generated AES secret key is stored in plain.

- Secret Key: 1qaz2wsx3edc4rfv5tgb6yhn7ujm8ik,
- IV: qwertyuiop456789

Press enter or click to view image in full size

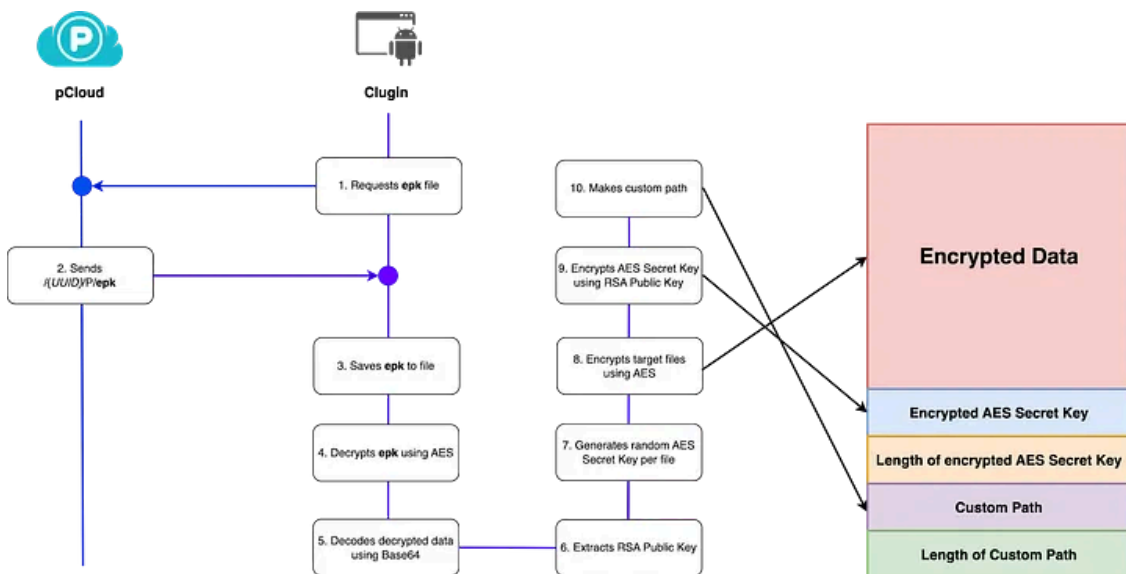


Figure 12. Structure of the files to be stolen

The AES secret key and hardcoded IV value used to encrypt files are shown below.

- Encryption: AES-256-CBC
- Secret Key: Random 32byte
- IV: qwertyuiop456789

```
String s3 = this.getKey();
try {
    this.aesEncrypt(file0, file1, s3, "qwertyuiop456789");
    if(this.PUK == null) {
        arr_b = s3.getBytes();
    }
    else {
        arr_b = this.encryptText(s3, this.PUK);
        if(arr_b == null) {
            arr_b = s3.getBytes();
        }
    }
}
```

Figure 13. Encryption flow

Files containing encrypted file data and additional information are named according to the type of each file. Only the top two formats in the table below are actually used, and a combination of UUID, cell phone number, and data type is used as the Custom Path.

Press enter or click to view image in full size

Date Type	Custom path
SMS, MMS, GPS, Contacts, Call Log	<i>/{{UUID}}/{{Phone Number}}/text/{{Data Type}}_{{Timestamp}}</i>
Audio, Call Record	<i>/{{UUID}}/{{Phone Number}}/audio/{{Original filename}}</i>
-	<i>/{{UUID}}/{{Phone Number}}/extData/{{Original filename}}</i>
-	<i>/{{UUID}}/{{Phone Number}}/chat/{{Original filename}}</i>
-	<i>/{{UUID}}/{{Phone Number}}/other/{{Original filename}}</i>

Table 7. Format of Custom path

### Stage 3: CallRecorder

After analyzing the CallRecorder that is additionally downloaded by the Clugin, we found that it is a DEX file that has a call recording function. CallRecorder records incoming and outgoing calls and saves them as separate files. The saved recording files are sent to the cloud via the Clugin.

- Package Name: com.sec.android.acservice

```
public void start() {
    CR.outputDir = this.ctx.getFilesDir().getAbsolutePath() + "/.temp/.data";
    File file0 = new File(CR.outputDir);
    if(!file0.exists()) {
        file0.mkdirs();
    }

    if(CR.outgoingReceiver == null) {
        CR.outgoingReceiver = new OutgoingReceiver(this);
        IntentFilter intentFilter0 = new IntentFilter("android.intent.action.NEW_OUTGOING_CALL");
        this.ctx.registerReceiver(CR.outgoingReceiver, intentFilter0);
    }

    if(CR.callStateListener == null) {
        CR.callStateListener = new CallStateListener(this);
        this.tm = (TelephonyManager)this.ctx.getSystemService("phone");
        this.tm.listen(CR.callStateListener, 0x20);
    }
}
```

Figure 14. Key features within CallRecorder

### Actions when additional messages are received from pushy

An attacker can send messages to Cumulus using Pushy, a messaging service, to update the status of the malware. This allows the attacker to continuously update the status of the infected device. The following information can be updated via messages

- *TID*: Change the upload path for stolen information on the cloud
- *ACCESSTOKEN*: Change OAuth Token
- *CLOUD*: Change cloud service from pCloud to Yandex

- *VERSION*: Update the Clugin version
- *AUTOSTART*: Set app auto launch

Press enter or click to view image in full size

```
public class PushReceiver extends BroadcastReceiver {
    @Override // android.content.BroadcastReceiver
    public void onReceive(Context context0, Intent intent0) {
        String s = intent0.getStringExtra("CLOUD");
        String s1 = intent0.getStringExtra("ACCESSTOKEN");
        String s2 = intent0.getStringExtra("TID");
        String s3 = intent0.getStringExtra("VERSION");
        String s4 = intent0.getStringExtra("AUTOSTART");
        SharedPreferences.Editor sharedPreferences$Editor0 = PreferenceManager.getDefaultSharedPreferences(context0).edit();
        sharedPreferences$Editor0.putString("CLOUD", s);
        sharedPreferences$Editor0.putString("ACCESSTOKEN", s1);
        sharedPreferences$Editor0.putString("TID", s2);
        sharedPreferences$Editor0.putString("VERSION", s3);
        sharedPreferences$Editor0.putString("AUTOSTART", s4);
        sharedPreferences$Editor0.commit();
        Intent intent1 = new Intent(context0, LogUService.class);
    }
}
```

Figure 15. Status update

The flow of malicious behavior executed by Cumulus via the Pushy message service is shown below. In the first execution, the infected device is identified by its UUID value, but after that, it is identified by its TID value.

Press enter or click to view image in full size

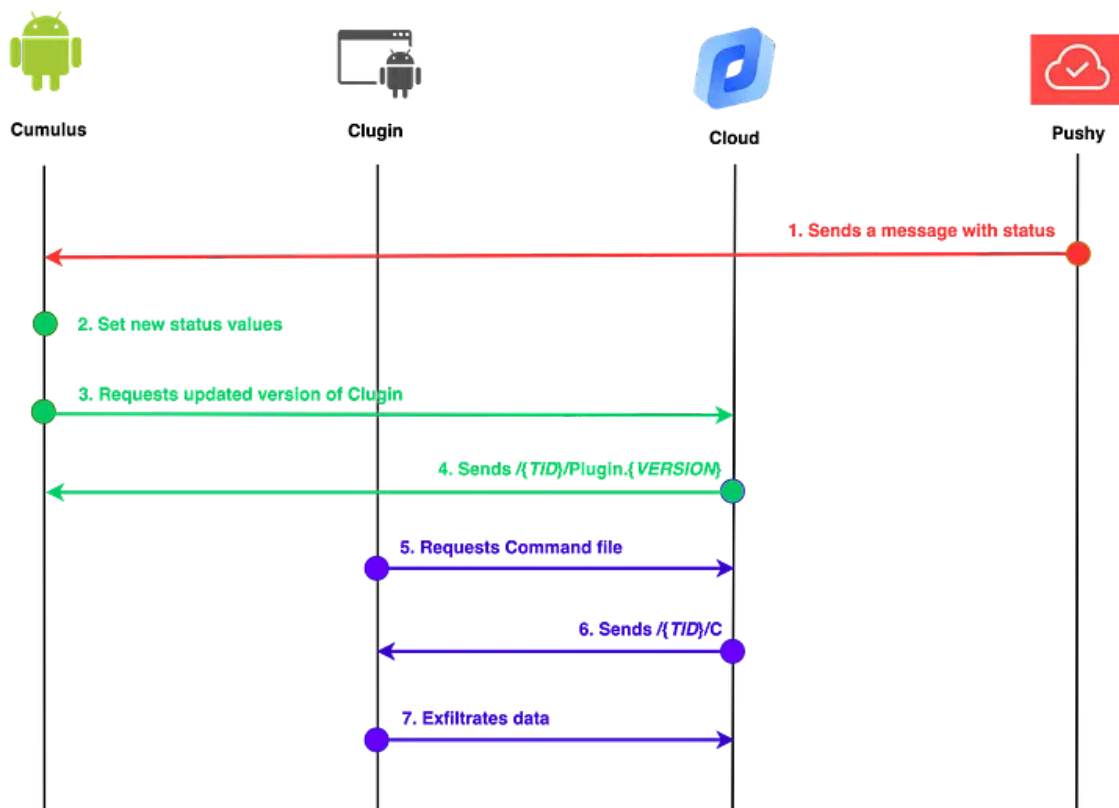


Figure 16. Execution flow when receiving a message from Pushy

## Interesting discoveries

We have been monitoring the group’s attack campaign for a few months and have been able to obtain data from victims compromised by Cumulus and Clugin, as well as test data from attackers leaked by OPsec failures. We were able to see malicious app deployment tests and the context of malicious app distribution via messengers.

## 1. Targeting Chinese Phones

The Scarcraft group has traditionally implemented its messaging capabilities through Firebase, but in the latest version, it uses a third-party service called Pushy. This is believed to be in case the targets use mobile devices made in China, such as Huawei. In fact, Pushy reviews indicate that many people have switched from Firebase to Pushy to ensure a stable implementation in China.

## 2. Installed Packages in Test Environment

We found that the attacker was testing the malicious APK. From the test logs, we could see the information of the attacker's test device, and from the installed application information, we could see that VPN and translation applications were installed.

Astrill VPN is used as a VPN application to bypass internet blocking in China, and SpeedCN is an application that increases the speed of Internet access in China. The presence of a translation application that can translate Chinese among the installed applications suggests that the attacker is preparing to target Chinese-speaking users.

Press enter or click to view image in full size

Installed Package			
Astrill VPN (com.astrill.astrillvpn)	현대중국어1.1 (com.chinese.Changgong)	SpeedCN (cloud.speedcn.speedcnx)	Papago (com.naver.labs.translator)

Table 8. Installed packages on the test device

## Attribution

Our analysis of the Cumulus and Clugin samples reveals a strong similarity in code and functionality to malicious APKs distributed by the Scarcraft group in the past through watering hole attacks. The malware used was a mobile version of ROKRAT, which suggests that the Scarcraft group continued to update it and use it to this day.

- Reference: [\[FSI\] Threat Intelligence Report - Campaign DOKKAEBI](#)

The malicious APK used by Scarcraft in 2017 drops an additional malicious APK with the package name "com.android.systemservice", and code similarities between the APK and Clugin 6.0 were found. The same routine for downloading Command files from the cloud and registering settings via SharedPreferences is present in both malware. We also found the same values for the keys registered by the 2017 sample and those in the Command downloaded by the 2023 sample.

Press enter or click to view image in full size



```
com.sec.android.acservice (2018)

try {
    sharedPreferences$Editor0.putInt("CMD", Integer.parseInt(s2));
    sharedPreferences$Editor0.putLong("ST", Long.parseLong(s3));
    sharedPreferences$Editor0.putLong("ET", Long.parseLong(s4));
    sharedPreferences$Editor0.putLong("GPSIT", Long.parseLong(s5));
    sharedPreferences$Editor0.putLong("CLIT", Long.parseLong(s6));
    sharedPreferences$Editor0.putLong("CTIT", Long.parseLong(s7));
    sharedPreferences$Editor0.putString("KEY", s8);
    sharedPreferences$Editor0.putLong("SDIT", Long.parseLong(s9));
    sharedPreferences$Editor0.putString("YANDEXTOKEN", s10);
    sharedPreferences$Editor0.putString("SDPATH", s11);
    sharedPreferences$Editor0.putString("SDKIND", s12);
    sharedPreferences$Editor0.commit();
    goto label_385;
}
catch(Exception e) {
}
```

## Conclusion

- We found that the Scarcraft group has continued to improve the mobile version of the ROKRAT malware they have been utilizing since 2017 and is still actively using it today.
- The mobile version of the ROKRAT malware can be classified as Cumulus, which receives messages from attackers via messaging services such as FCM or Pushy, and exfiltrates data to cloud services such as pCloud and Yandex.
- As disclosed by Interlab, the group is conducting attack campaigns targeting individuals and using conversations to convince them to install malicious apps disguised as legitimate apps, such as image viewers, messenger programs, etc.
- A multi-channel strategy that utilizes cloud services such as Yandex and pCloud, as well as legitimate services such as Firebase and Pushy for command and control.

## IoCs

Full IoC list can be found our [github](#)

- 5dde5f5fcc1ebfd932e1ef0bfcc7b272
- 957ebfbd0b23a164529d7510ca89ddae
- 3ae92bc233dd6a4412aa77da4dc44a19
- ae767e4658a5d235ec614eaa8655da0d
- be6f13d6e7ae5039aed46d1f8844f3ee
- 0711102cbfcf18a3672a892c4ea31ad1
- e4f781e00bc48f88a717095deb78be6f
- ce3104fe4184558fkea707368846c226
- 97856a842ff8161576fee5ad3fd0ec67
- 580f22dde975ac5e3544f3a74f4a91b9
- 97a750f33812195cc2add4ebd120b468
- 1f2c23c7c9ecb28bfdc6627a3ad23783
- 97a9ab76af215241ad2a07856b40242e
- fe11b08764fba51236325be852ca1406

- a90e3bd0e2de1b6a6bec269dc0f09369
- 15470bafbabf3841bac1813881e6524fa
- 97ecdb46b8325a845e998cfe3bd2262e
- 214ead5c75899b8d1382e558e542574a
- 464df52f091f95a561474d4de62a821b
- 759b26631a660d82f6a93621991c4292
- 445922b01b3f8f463cb9f48d74efd9a8
- 89c669739066ac655a1e2b772bb020f3
- a97e22b8ca16452a4ddcb32284d7c7a7
- 8092bb293352ef572464c682e81f329f
- 1d4683844c8429ad141f9f66bcf29728
- 27e0dcceb68c03b246874c9fcc9b744e
- 72182f83e771fcaaa1e86c7c932014cb
- f58fed1e492f40d28e0bc38dc0f76b35
- d7723de89903a04b93c7a9a92d8309c2

## **ATT&CK Matrix**

### **Credential Access**

- Steal Application Access Token (T1635)

### **Persistence**

- Event Triggered Execution (T1624)

### **Discovery**

- File and Directory Discovery (T1420)
- Location Tracking (T1430)
- Software Discovery (T1418)
- System Information Discovery (T1426)

### **Collection**

- Archive Collected Data (T1532)
- Audio Capture (T1429)

### **Command and Control**

- Web Service (T1481)

### **Exfiltration**

- Exfiltration Over Alternative Protocol (T1639)

Source: <https://medium.com/s2wblog/scarcraft-bolsters-arsenal-for-targeting-individual-android-devices-97d2bcef4ab>