DJI ANDROID GO 4 APPLICATION SECURITY ANALYSIS

Written by The Team - 23/07/2020 - in Systems , Reverse-engineering

Drones are currently one of the most dynamic products, with multiple use cases across sectors such as personal and commercial videography, farming and land surveying, law enforcement and national security, and more. One of the market leaders, China-based Daijiang Innovations (DJI), is often in the news for suspected cybersecurity and data privacy issues.

While there are technical reports sponsored by DJI stating that their associated mobile application, DJI GO 4, is harmless and does not send any personal information back to the Chinese manufacturer, we wanted to assess the technical capabilities of the application ourselves.

Drones are currently one of the most dynamic products, with multiple use cases across sectors such as personal and commercial videography, farming and land surveying, law enforcement and national security, and more. One of the market leaders, China-based Daijiang Innovations (DJI), is often in the news for suspected cybersecurity and data privacy issues [1][2][3].

While there are technical reports [4] sponsored by DJI stating that their associated mobile application, DJI GO 4, is harmless and does not send any personal information back to the Chinese manufacturer, we wanted to assess the technical capabilities of the application ourselves.

We found that :

- Despite being under scrutiny, DJI did not improve the transparency surrounding the potential abuse of its Android mobile application: DJI GO 4 application makes use of the similar anti-analysis techniques as malware, such as anti-debug, obfuscation, packing and dynamic encryption.
- After de-obfuscation, our research located two features of the software that call home and wait for a file that orders the user's phone to install a forced update or install a new software. This mechanism is very similar to command and control servers encountered with malwares. Given the wide permissions required by DJI GO 4 [5] (access contacts, microphone, camera, location, storage, change network connectivity, etc.), the DJI or Weibo Chinese servers have almost full control over the user's phone. This way of updating an Android App or pushing a new app completely circumvents Google feature module delivery [6] or in-app updates [7]. Google is not able then to do any verification on update and modifications pushed by DJI. According to Google Play, the application has been installed on more than a million personal devices, suggesting any security risks are widespread.
- The MobTech component embedded in recent versions of DJI Android GO 4 application collects personal data such as IMSI, IMEI, the serial number of the SIM card, etc. This data is not relevant or necessary for drone flights and go beyond DJI privacy policy [8]. For example, IMSI is used by cellular network operators. These sensitive, unique, persistent data identifiers can be used by intelligence agencies or malicious people to later track individuals or eavesdrop communications.
- The DJI GO 4 application on the Android platform does not close when the user closes the app with a swipe right. The app continues to run in the background and makes network requests.
- Whereas our findings affect the Android version of DJI GO 4, the iOS version of the application is not obfuscated and doesn't have the hidden update mechanisms.

Thus, users of the DJI drone are advised to use caution, due to the risks of leakage or misuse of sensitive data elements, and hidden command and control features, seemingly not needed for safe or secure use of the product.

THE DJI GO 4 APPLICATION FOR ANDROID

The recommended way to pilot DJI drones is to connect a phone to the remote controller and use the *DJI GO 4* application. It is not possible to pilot these aircrafts by default without the application running on a mobile device. We did our testing with a Mavic 2 but DJI GO 4 is used for most DJI drones.

We analyzed several samples of the application on the Android plateform, namely:

- version *v4.3.36_200426* updated May 12, 2020 (SHA256 `3887b663709c5f4b586289d25449a740f268a3b2c66f78a53ab33a124c9e9208`)
- version *V4.3.25-2036168* released October 9, 2019 (SHA256 `1e6f1a0151c2655069e45c1d858b5541a017b474c520c73b615cba6fa57a394f`)
- version *V4.1.22-3028592-nosecneo* released in December 2017 (SHA256 `8e29e190e9b0879960dac27d73a63100959ef52aada1baa51cf45eccfc5beeb2`).

UNPACKING

While it is possible to retrieve unencrypted older versions of this application, we had to unpack the last version before proceeding with the analysis.

A variation of the `SecNeo/BangCle` protection scheme [9] is used, which decrypts several `Dex` files and loads them in memory.

The unpacking routine is implemented in a native library, named `libDexHelper.so`. This library is obfuscated using code flattening and almost all the strings are decrypted on the stack before use.

For example, in the following snippet, the library is looking for a specific Huawei phone model.

```
decrypt_string(lpszRoProductModel, 16, 0xD4, 0xE6);// ro.product.model
((void (__fastcall *)(char *, char *))system_property_get)(lpszRoProductModel, v71);
memset(v69, 0, 0xFu);
v69[3] = 0xF5;
v69[4] = 0xE1;
v69[5] = 0xF7;
v69[8] = 0x80;
v69[9] = 0xC5;
v69[6] = 0xE5;
v69[1] = 0x74;
v69[7] = 0xE9;
v69[2] = 0xE8;
v69[10] = 0xE8;
v69[11] = 0x98;
v69[12] = 0x98;
v69[13] = 0x90;                       // HUAWEI eH880
decrypt_string(v69, 12, 0xD4, 0x90);
if ( !strcmp(v71, v69) )
goto LABEL_59;
```

While it would be fun to completely statically analyze this library, we want to have access to the rest of the application quickly.

Breaking the SecNeo/BangCle packer has been described publicly [10], however the provided tools did not work on our setup. In order to dump the decrypted dex files, we used Frida [11] to hook into the application and to dump the memory mapped files to disk. The usual SecNeo/BangCle decrypts one dex file, but we noticed that this app decrypts no less than seven dex files.

STATIC ANALYSIS

With the application fully decrypted, we decompiled it with `jadx` [12] in order to analyze it.

However, the developers left us some more challenges to slow down the security analysis.

DECRYPTING THE STRINGS

Almost all the strings in the application are ciphered using a `XOR` based scheme. The key used for the decryption is the same in both samples and a decryption script was reimplemented. This script was then executed on each decompiled java file, which resulted in almost all the strings being decrypted.

```
import base64
def decrypt(buf):
    """
        Decryption of the strings inside the DJI GO 4 Android application
    """
    key = b"Y*IBg^Yd"
    out = bytearray()
    for i, x in enumerate(base64.b64decode(buf)):
        out.append(x^key[i%len(key)])
    return out
```

SIDE QUEST: BREAKING THE AES WHITEBOX

The application uses an AES Whitebox in order to decrypt what it considers as sensitive elements:

- The hashes of the trusted certificates in the certificate pinning utility
- The RSA public key which is used to verify the signature of the No-Fly-Zone file
- The hashes of several FlySafe related databases

The whitebox functions are based on `JNI` wrapped linking to the `libwaes.so` library.

At first, we ripped the AES whitebox in order to statically decrypt the data, but we also succeeded in breaking it using *Differential Fault Analysis* with the *Deadpool* tools [13].

Breaking this whitebox can be done in a few steps:

- Ripping off the whitebox data from the binary (at address `0x5004`)
- Reimplementing the block encoding and encryption function (`0xFF0` and `0x12EC`) in a python script
- Giving the python script and the whitebox data as arguments to *DeadPool*

```
First round key #0 found:
0123456789ABCDEF0123456789ABCDEF
python3 deadpool_dji.py  5.17s user 1.08s system 101% cpu 6.181 total
```

The reimplementation is left as an exercise to the reader. As seen in the example, the key used to hide some of the features is weak. However, the suspicious features we found were not protected with the AES whitebox.

ENCRYPTION OF LOG FILES

DJI application implements its custom log Utility, named DJILogUtils, located in `dji.log`. Logs are encrypted with AES 256 CBC with a PBKDF2 derived password. This could appears as a strong mechanism, but password is defined in the `dji.log.impl.SimpleEncryption`:

```
   public SimpleEncryption() {
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append("DJILog@");
        stringBuilder.append(getClass().getSimpleName());
        this.KEY_STR = stringBuilder.toString();
    }
```

The key is then derived through PBKDF2. The AES key is `e9e856d55943731ac585dcda656f95c5`. The IV is hardcoded: `9d6c5cab5b0281255a222d1c861ddfdf`. All logs are splitted and stored in `/storage/emulated/0/DJI/dji.go.v4/LOG/CACHE/` such as `log-2020-02-18.log` or `BatteryEmbed/log-2020-02-02.log`.

SHADY FEATURES

AUTO UPDATE MECHANISM

While Google Store is usually used to update an application, DJI also implements its own update mechanism in the `DJISelfUpgradeManager` class.

The application checks the URL `hxxps://service-adhoc.dji.com/app/upgrade/public/check` for a configuration file and can even force the update of the application if the flag `forceUpdate` is set in the JSON answer.

The application will then download the provided APK application on the arbitrary provided URL, and prompt the user to install it.

We managed, using a *Frida* [11] script [14] to bypass the SSL pinning and intercept the request. We subsequently modified the request using *Burp* [15] to trigger the update mechanism.

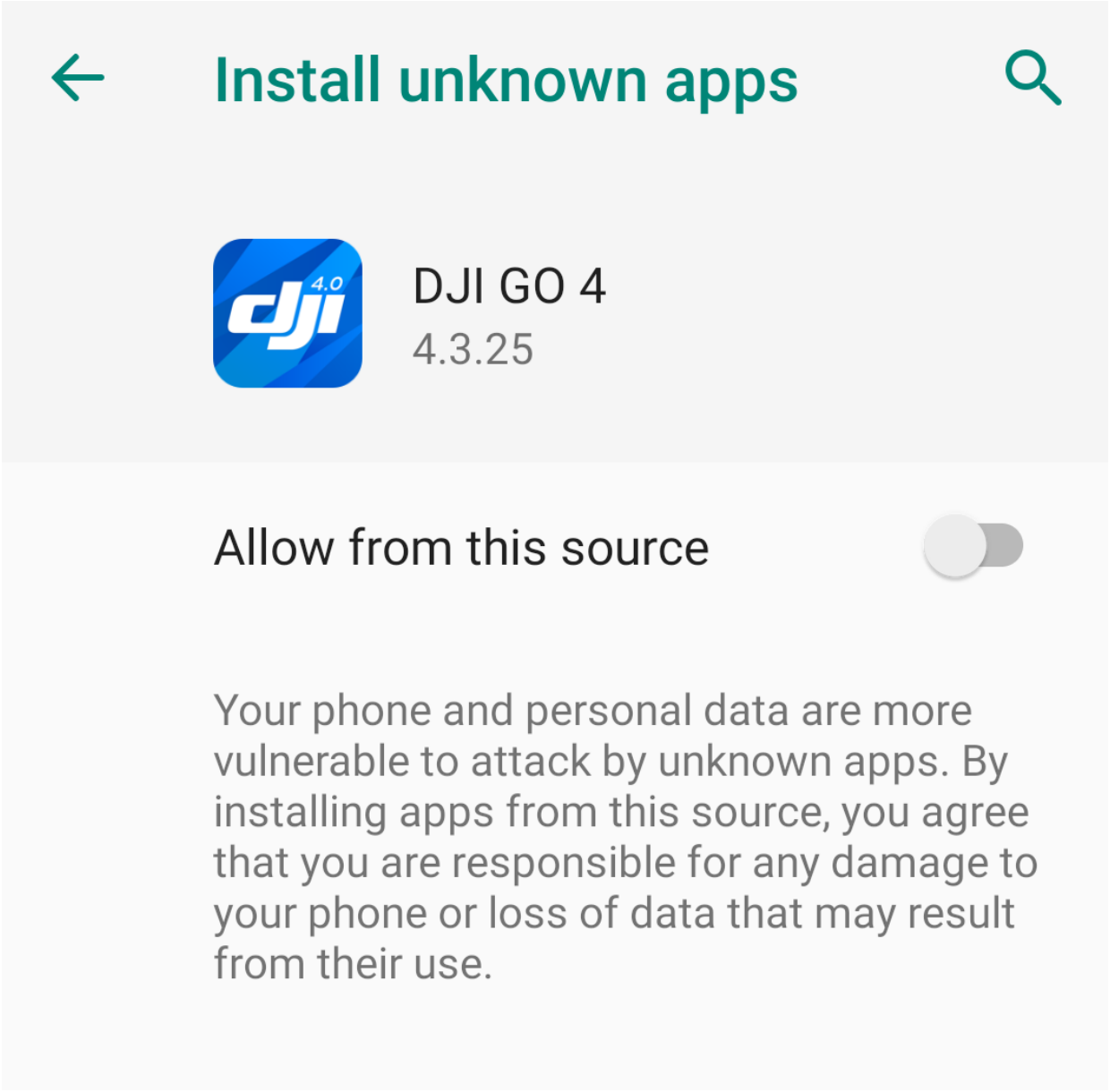
As shown below, the request to the service does not include personal identifiers, however the IP address of the client can be used to correlate to a specific user using the other telemetry services of the application.

```
POST /app/upgrade/public/check HTTP/1.1
Content-Type: application/json
adhoc-sign: REDACTED
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; REDACTED
Host: service-adhoc.dji.com
Connection: close
Accept-Encoding: gzip, deflate
Content-Length: 149

{
  "appKey": "djigo4",
  "appStatus": "release",
  "language": "en",
  "packageName": "dji.go.v4",
  "platform": "android",
  "versionCode": "36168",
  "versionName": "4.3.25"
}
```

An example of answer we got from DJI's server is shown below:

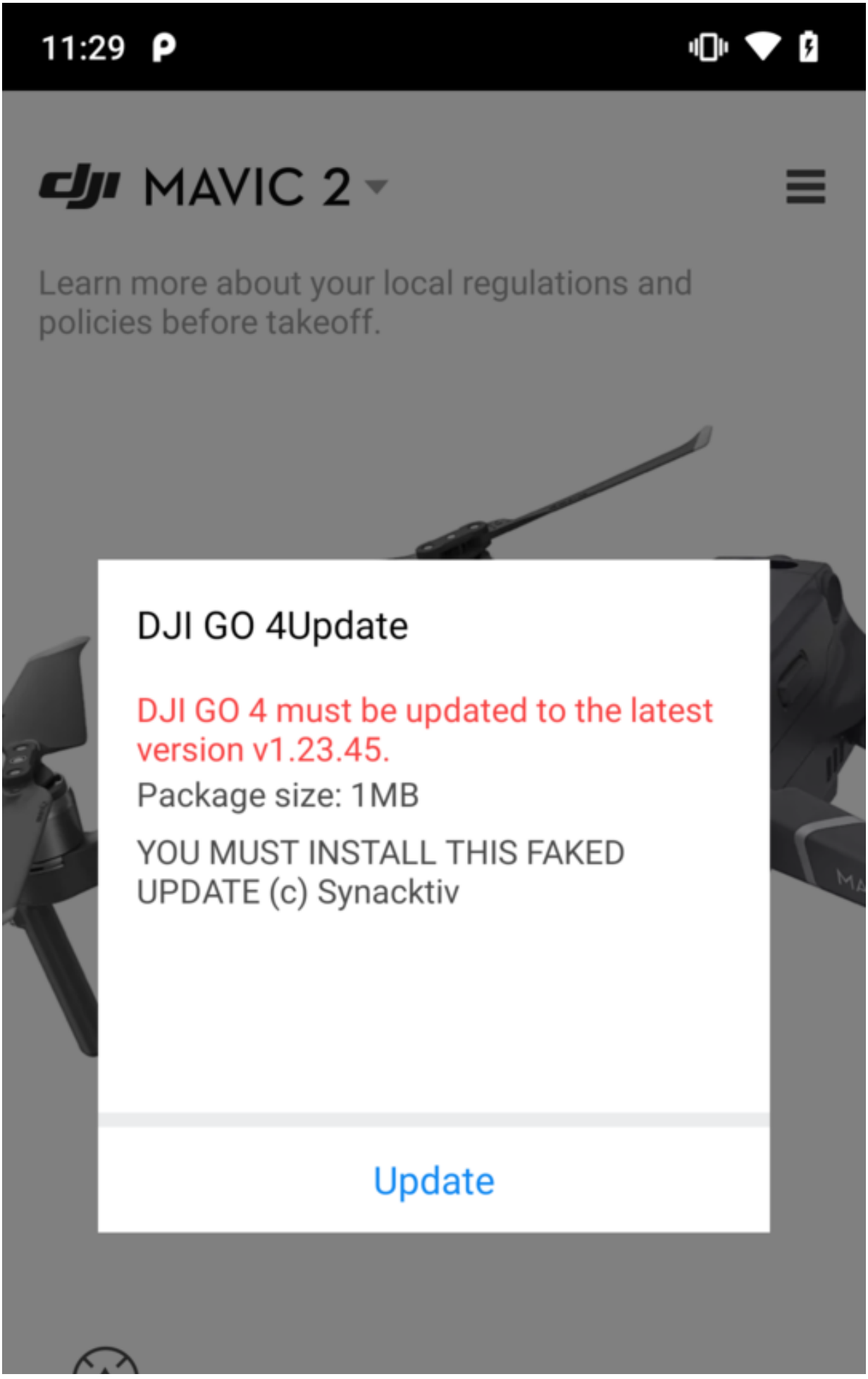
```
HTTP/1.1 200
[...]

{
  "code": 0,
  "message": "成功",
  "data": {
    "update": "true",
    "downloadUrl": "https://terra-2-g.djicdn.com/a81c919a2cba4e93a2147801711c04d1/158
8314879661-DJI-v4.3.36_200426-967-36438_official_sec.apk?auth_key=1595494081-1595407
681750-0-9f4c82d4b211c9395dfbd58116be4660",
    "isForce": 0,
    "newVersionName": "4.3.36",
    "newVersionCode": "3036438",
    "log": "What's New\n- Adds support for French",
    "md5": "3e861b560f9f53d3e5b136fb315a5400",
    "fileSize": 262077083,
    "releaseTime": "2020-01-08T07:00:45"
  }
}
```

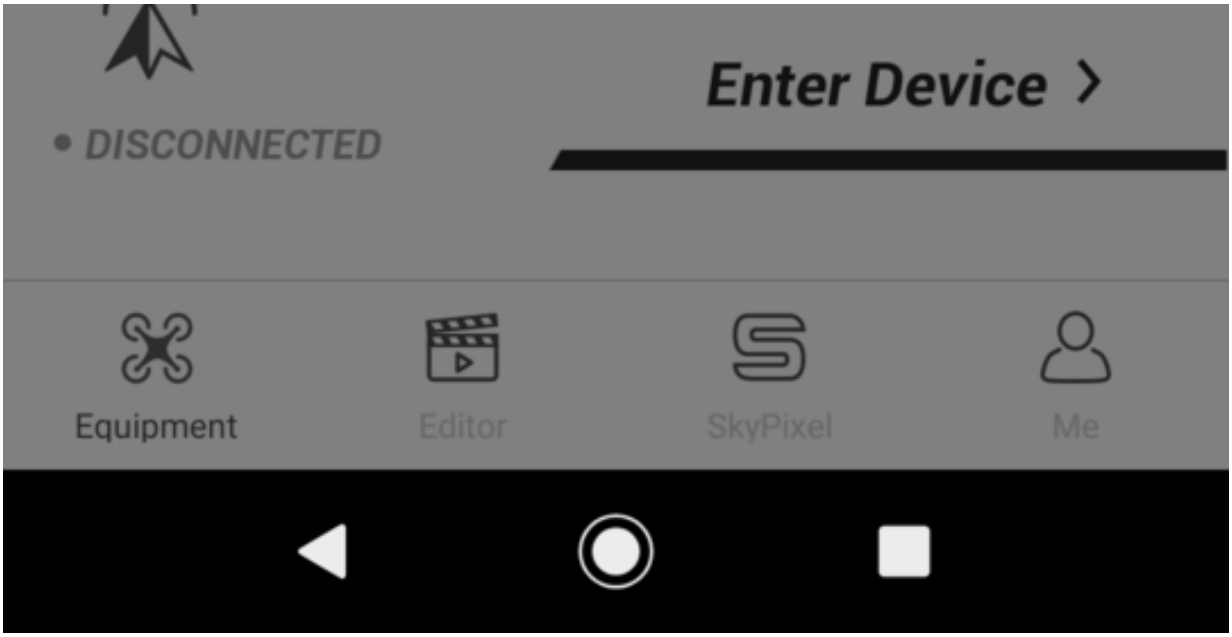
Using *Burp*'s Match and Replace feature, we modified this request to trigger a forced update to an arbitrary application, which prompted the user first for allowing the installation of untrusted applications, then blocking him from using the application until the update was installed.



Screenshot of the prompt asking the user to install untrusted applications

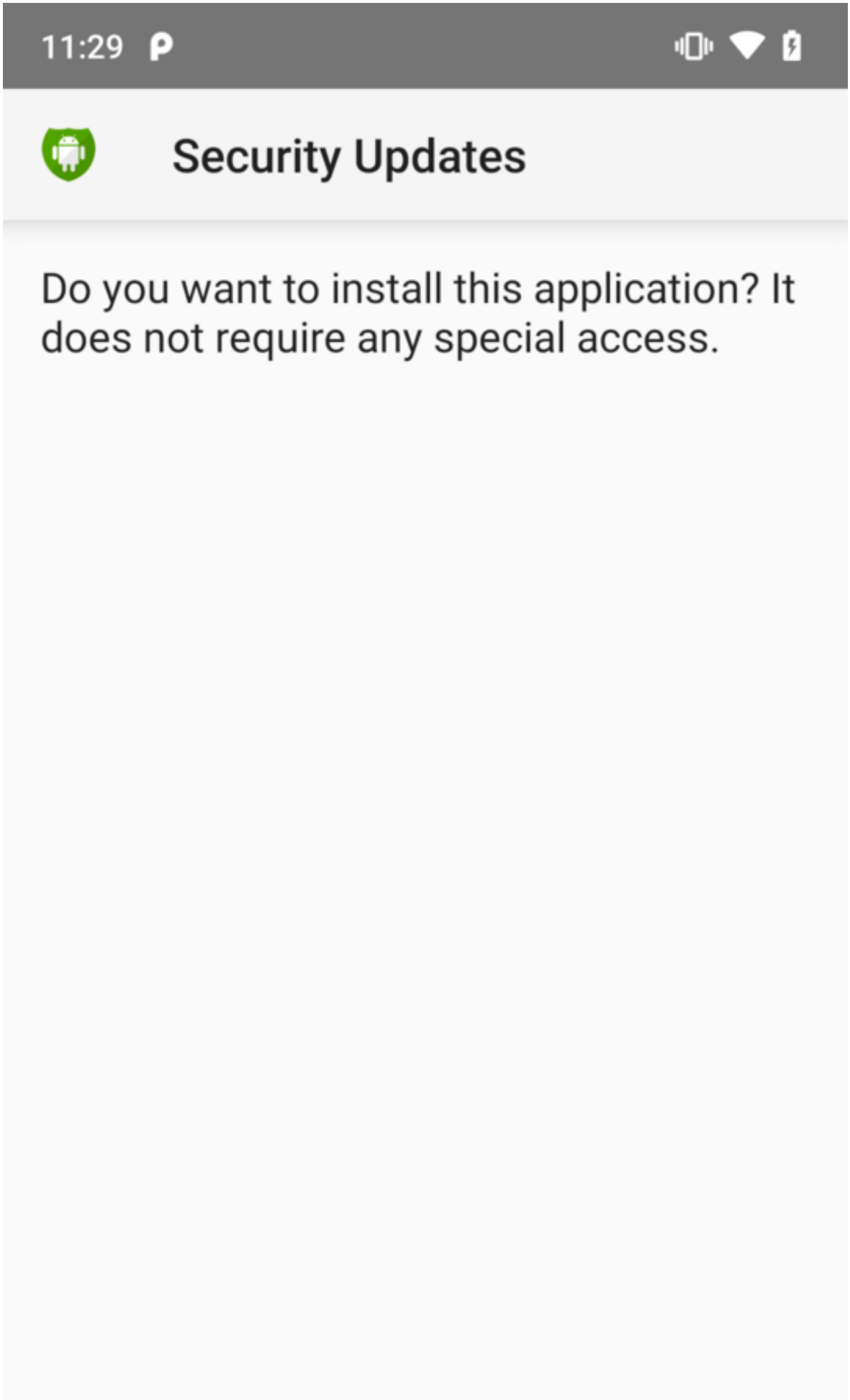
As seen in the screenshot below, DJI has a full control over the message displayed to the user during the second prompt, thus this message can convince the user to install the application.

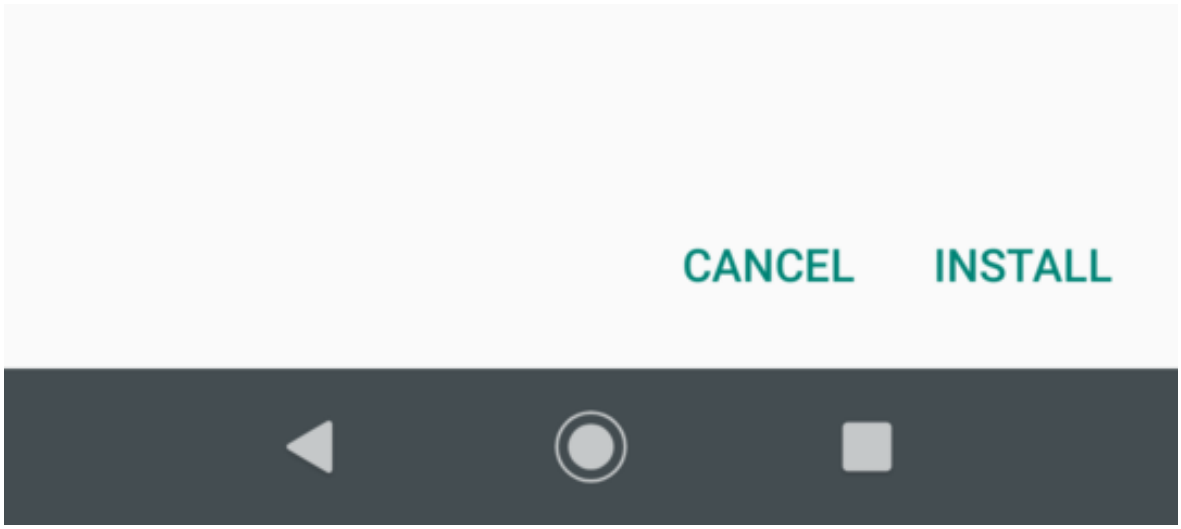

11:29
MAVIC 2
Learn more about your local regulations and policies before takeoff.
DJI GO 4Update
DJI GO 4 must be updated to the latest version v1.23.45.
Package size: 1MB
YOU MUST INSTALL THIS FAKED UPDATE (c) Synacktiv
Update



Screenshot of a message issued from the request and displayed to the user

The last prompt is asking for a confirmation before installation, displaying the application name (here *Security Updates*) and a default message.


11:29
Security Updates
Do you want to install this application? It does not require any special access.



Screenshot of the last prompt before installation

Given the wide permissions required by DJI GO [5] (access contacts, microphone, camera, location, storage, change network connectivity, etc.), the DJI or affiliated third-party servers have almost full control over the user's phone. This way of updating an Android App or pushing a new app completely circumvents Google feature module delivery [6] or in-app updates [7]. Google is not able then to do any verification on update and modifications pushed by DJI.

Therefore, any security assessment made on this application, such as *Kivu* [4], is strongly limited because potential malicious code can be pushed by DJI afterwards through this auto-update mechanism.

The user cannot have control on the version and features of the application he runs.

WEIBO SDK APK DOWNLOADER

The sharing functionality of the application uses a SDK developed by the Chinese company *Weibo*.

This SDK is named `com.sina.weibo.sdk`. When initiated, this SDK starts by checking an application specific token by using the `WbAppActivator` class, then two message handlers are registered by this class: `AppInvokeCmdExecutor` and `AppInstallCmdExecutor`.

The `WbAppActivator` then creates a new thread, querying each hour (by default) new commands on the URL `hxxp://api.weibo.cn/2/client/common_config`.

The result of the request is then decrypted using the AES algorithm in ECB Mode, and a key derived from a ~~strong~~ Game of Thrones secret (eg the MD5 of `Stark`).

The decrypted message is now handled by one of the two command handlers, the most interesting one being `AppInstallCmdExecutor`.

This handler download the URL passed in the command text and then prompts the user to install the arbitrary APK downloaded, acting as a dropper.

This functionality is activated only when the user attempts to stream using *Weibo*.

DJI INTERNAL REQUESTS

The application communicates through a lot of *.dji.com* websites.

At each startup, the app registers itself through the URL `hxxps://mydjiflight.dji.com/api/v2/register_device` with information related to the app.

It then polls this website to get access to geocode information related to localization.

The DJI account used to access the application is used to autolog into the `hxxps://www.skypixel.com` social network.

By hooking some classes with Frida, we can see that the access are not made anonymously. For example, flight logs use an id and token to identify the owner of the drone:

- `hxxps://mydjiflight.dji.com/api/v2/flight_log/profile?user_id=<numerical id>`
- `hxxps://mydjiflight.dji.com/flight/overview?token=<token>`

Other calls are made to other DJI related websites, namely:

- `flysafe-api.dji.com`
- `terra-2-g.djicdn.com`
- `account-api.dji.com`
- `djigo-hk.djiservice.org`
- `djigoapi.djiservice.org`
- `developer.dji.com`
- `store.dji.com`
- `statistical-report.djiservice.org`

TELEMETRY

BUGLY TELEMETRY

`bugly` is a crash reporting module provided by the Chinese company `Tencent`.

It was used in previous versions of the application (including the `v4.1.22` we analyzed).

This SDK registers a crash monitor for the application and store them in a SQLite database before having a network connection and sending them to a configurable URL (for example: `hxxp://android.bugly.qq.com/rqd/async`).

The crash reports include, amongst others, the following elements:

- The IMSI and IMEI serial numbers of the phone
- The MAC address of the Wi-Fi interface
- The `android_id` of the phone
- The serial number of the SIM card
- The status of the mounted filesystems

The information collected at the time by this feature seems to go beyond DJI privacy policy [8] especially concerning identification numbers used only by cellular network operators (IMSI, IMEI, serial number of the SIM card).

This framework was removed from the application between the versions 4.1.22 and 4.3.25. Several concerns about this telemetry were published by security researchers and journalists [16].

MAPBOX TELEMETRY

DJI is using a map service published by the provider *MapBox*. This service is sending telemetry pingbacks to its publisher, as publicly documented [17].

However, we found that this telemetry is quite verbose, as an event is sent each time the map is clicked or dragged.

GAODE/AMAP TELEMETRY

AMAP, also named *Gaode*, is a Chinese cartography service provider and a subsidiary of Alibaba. Its cartography framework is used in the application and contains several interesting items:

- It decrypts its server names using simple xor based algorithms

- If the collection is enabled in the SDK configuration, it sends both the mobile phone IMSI identifier and its current location, which enables the provider to track a specific phone given its IMSI.
- Most of the communications between the application and the servers are performed over HTTP.

However, it seems that this SDK only is only activated when the user is geolocated in China.

MOB SDK FRAMEWORK.

The application embeds a SDK framework developed by mob.com [18] designed to get almost any metadata available from the smartphone.

MobTech is advertised as a data intelligence plateform designed to help developers to gather data from applications.

At least, this framework works as intended: it collects data. A lot of data. As anyone can see in the `com/mob/tools/utils/DeviceHelper.java` classes, almost any data which can be used to track a user is queried. It goes from screen size and brightness to WLAN address and MAC, BSSIDs, Bluetooth addresses, Mac addresses from neighbors, IMEI and IMSI, carrier name, SIM serial Number, SD card information, OS language and kernel version, Location and language and so on.

The full list of data collected cannot fit in this blogpost but the amount and depth of data collected is significant and can be used to track a single person or device. Just like the previous Bugly feature, the information collected by mob.com and DJI seem to go beyond DJI privacy policy [8] especially concerning identification numbers used only by cellular network operators (IMSI, IMEI, serial number of the SIM card). As a side note, MobTech privacy policy [19]mentions way more data being collected compared to DJI privacy policy.

An example of requests sent to mob.com website containing a lot of private information is contained in the snippet below:

```
request:
  {"androidid":"ff912a4d6f25e18",
   "networkType":"wifi",
   "sysverint":21,
   "appkey":"25552aad4fa54",
   "carrier":"-1",
   "mac":"AA:BB:CC:DD:EE:FF",   //redacted
   "front":false,
   "serialNo":"12345678",       //redacted
   "factory":"samsung",
   "lock":0,
   "plat":1,
   "errors":
     {"list":[
       {"ct":1582404426823,
        "mg":"[2020-02-22 21:47:06.823 CET][8][1] ld vr 1",
        "po":1,"et":8}],
      "sd":"12345678-1234-1234-1234-123456789012"},   //redacted
   "home":0,
   "duid":"1234567890123456789012345678901234567890", //redacted
   "imei":"1234567890abcde",                          //redacted
   "brand":"samsung",
   "uiver":"LRX21T.G900FXXU1POK5",
   "appver":3036168,
   "apppkg":"dji.go.v4",
   "clientTime":1582404437003,
   "sdkver":20190826,
```

```
   "sysver":"5.0",
   "appmd5":"1fe72ecb36d7d44aa2b53de11881bf38",
   "model":"SM-G900F"}
url = hxxp://dfe.mic.mob.com/drl
```

A third party company [20] disclosed how much data DJI Mimo application and Mob.com SDK sends abroad. After this publication, the Mob SDK was also removed from the latest version available of DJI GO 4.

UNSTOPPABLE APPLICATION

If you close the application in Android by swipe right, it doesn't close. The app continues to run in the background and makes network requests. A service, called Telemetry provided by *MapBox* will restart the application in the background.

If you want to effectively close the application, you must terminate the service and close the application in the Android Settings. The user might have a false sense of security when he or she thinks the app is closed, whereas the app is still able to collect data or modify its features.

NETWORK CONNECTIVITY

Some users of DJI GO 4 may argue that they switch their phone to flight mode when they flight drones for sensitive missions. It's important to consider that among permissions required by DJI GO 4 [5], the application has permission to change network connectivity.

FURTHER LEADS

Several parts of the applications were not covered by this research, such as many of the native libraries which have not been analysed. It therefore remains untested if DJI Go 4 contains even more shady features.

CONCLUSION

Security concerns on the DJI GO 4 application are likely founded, especially given the lack of transparency around the application capabilities.

The analysis of DJI GO 4 shows similar result to other Chinese applications such as `Study the Great Nation` [21]: obfuscation for hiding functionalities, information gathering including information on the phone, cellular network ID and GPS location of the user or the drone and execution of code without the control of the user (forced updates). Thus, the application should not be used for sensitive purpose.

IOCS

- `hxxp://api.weibo.cn/2/client/common_config`
- `hxxps://service-adhoc.dji.com/app/upgrade/public/check`
- `hxxp://android.bugly.qq.com/rqd/async`
- `hxxp://wb.testing.amap.com`
- `hxxp://group.myamap.com`
- `hxxp://m.map.so.com`
- `hxxp://114.247.50.32`
- `180.96.64.225/mo`

1. DJI likely providing U.S. Critical Infrastructure and Law Enforcement Data to Chinese Government
2. US warns of potential data leaks from Chinese-made drones
3. La controverse autour des drones chinois en cinq questions
4. a. b. Kivu Consulting “technical” report
5. a. b. c. DJI GO 4 permissions
6. a. b. Android Dynamic feature module delivery
7. a. b. Android in-app-updates
8. a. b. c. DJI Privacy policy
9. Bangcle The second generation Android Hardening Protection
10. android-unpacker
11. a. b. FRIDA: Dynamic instrumentation toolkit
12. JADX Dex to Java decompiler
13. Deadpool
14. https://blog.jamie.holdings/2019/01/19/advanced-certificate-bypassing-in-android-with-frida
15. Burp Suite - Application Security Testing Software
16. Yes, drone biz DJI’s Go 4 app does phone home to China – sort of
17. Mapbox Telemetry
18. Mob website
19. MobTech privacy policy
20. ANALYZING DATA USE BY THE DJI MIMO APP
21. Study the Great Nation