White Paper:

Anatomy of DJI's Drone Identification Implementation

November, 2017

department 13

Table of Contents

Table of Figures

Introduction

In 2016, U. S. lawmakers who were concerned about the increasing number of incidents that involved small unmanned aircraft created a law to monitor and track unmanned aircraft. Congress passed Public Law 114-190 on July 15, 2017. The new law created the groundwork to "facilitate the development of consensus standards for remotely identifying operators and owners of unmanned aircraft systems (UAS) and associated unmanned aircraft." [1] Known as the FAA Extension, Safety, and Security Act of 2016, P.L. 114-190 "included language requiring FAA to develop standards for remote identification of unmanned aircraft." [2] The law provides a potentially good solution to an important problem. However, creating a system that conforms to the law, without creating new and worse problems, will be difficult. This white paper will examine several nuances in DJI Technology's implementation of Remote Drone Identification, commonly known as 'DroneID' and 'UAS-ID'.

Figure 1. DJI ID AeroScope ground unit (photo by Gareth Corfield)

[1] https://www.congress.gov/114/plaws/publ190/PLAW-114publ190.pdf
[2] https://fas.org/sgp/crs/misc/R44791.pdf

Background

To understand the origin of DroneID it is important to understand the FAA's reasoning for implementing P.L. 114-190 and how the FAA worked with industry and stakeholders to define requirements. The FAA created a committee to define the requirements and to obtain feedback from relevant parties. The committee relevant to DroneID is the UAS-ID ARC.

"Remote Identifier Would Provide Accountability While Protecting Operator Privacy" [3] Brendan Schulman

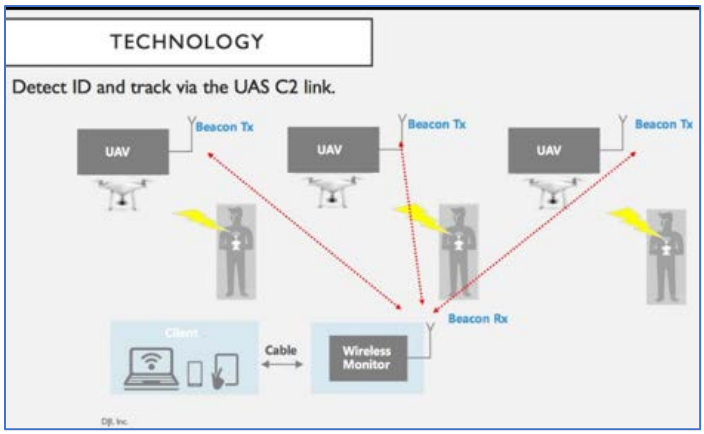

Figure 2. High-Level Overview of Drone ID Sensing [4]

Advisory and Rule-making Committees (ARC) are standard mechanisms that the FAA, and other federal agencies, use to obtain advice and recommendations. UAS-ID ARC was created for Drone ID with a charter to: "provide a forum to discuss and provide recommendations to the FAA regarding technologies available for the remote identification and tracking of UAS" [5]. Specific UAS-ID ARC goals include:

- "Identify, categorize and recommend available and emerging technology"
- "Identify the requirements for meeting the security and public safety needs"
- "Evaluate the feasibility and affordability of possible Drone ID solutions" [6]

[3] https://www.dji.com/newsroom/news/dji-proposes-electronic-identification-framework-for-small-drones
[4] https://www.icao.int/Meetings/UAS2017/Documents/Walter%20Stockwell_Stream%20A.pdf
[5] https://www.faa.gov/news/updates/media/UAS_ID_and_Tracking_ARC_Charter.pdf
[6] https://www.faa.gov/news/updates/media/UAS_ID_and_Tracking_ARC_Charter.pdf

The UAS-ID ARC membership "represents a diverse collection of stakeholders, including the unmanned aircraft industry, the aviation community and industry member organizations, manufacturers, researchers, and standards groups." [7] For the purpose of this white paper, it should be noted that Dà-Jiāng Innovations Science and Technology Co. ("DJI Technology") is a member of UAS-ID ARC. More importantly, DJI Technology was among the first companies to publicly describe and implement a functional remote identification platform for unmanned aerial vehicles (UAV) that operates within the constraints of P.L. 114-190.

"The best solution is usually the simplest"

- DJI Whitepaper

Figure 3. Example of Operational DJI Drone ID Deployment [8]

This white paper focuses on DJI's implementation of a remote drone identification system based on the influences of the drone community. In the recent technology white paper "What's in a Name? A Call for a Balanced Remote Identification Approach", DJI specifically mentions "The Privacy Interests of the Operator" [9] as a potential hurdle to Drone ID. Potential privacy issues were highlighted with a set of "examples of companies and individuals who have a legitimate reason not to have their operations of UAS tracked and recorded, or otherwise made available to faraway observers who may include competitors." [10]

Many other concerns are downplayed and likened to public perception on requirements for license plate security for motor vehicles. Since license plates are readily visible to the public, it is implied that security concerns should be minimal. Similarly, the drone community, and DJI

[7] https://www.faa.gov/news/updates/?newsId=88289&omniRss=news_updatesAoc&cid=101_N_U
[8] https://www.icao.int/Meetings/UAS2017/Documents/Walter%20Stockwell_Stream%20A.pdf
[9] https://www.dropbox.com/s/v4lkyr2kdp8ukvx/DJI%20Remote%20Identification%20Whitepaper%203-22-17.pdf?dl=0
[10] https://www.dropbox.com/s/v4lkyr2kdp8ukvx/DJI%20Remote%20Identification%20Whitepaper%203-22-17.pdf?dl=0

specifically, imply that distinct drone identification technologies should have minimal security and privacy concerns.

According to DJI's drone ID proposal, DJI suggests that using a non-networked, localized ID [11] that is associated with a distinct drone provides a means of balancing the constraints between public safety, security, and drone operator accountability with drone operator privacy and safety." [12] According to DJI's proposal, this same type of Non-Networked ID can allegedly be used to "create an identification mechanism that provides localized identification without permanent recording or logging". The DJI proposal states further that non-networked IDs associated with distinct drones will result in a localized and transient identification and logging mechanism. [13] "An identifier, such as a registration number, together with position information about the drone, and perhaps voluntary information if the operator wishes, is transmitted from the drone." [14] This information is subsequently "available to all receivers that are within range." [15] While these comments and ideas seem reasonable, there are some clear security concerns that DJI, in part, raises in its own proposal.

"It must have been enabled across the fleet they picked up someone else's drone at an AeroScope demo…"

- Anonymous Participant

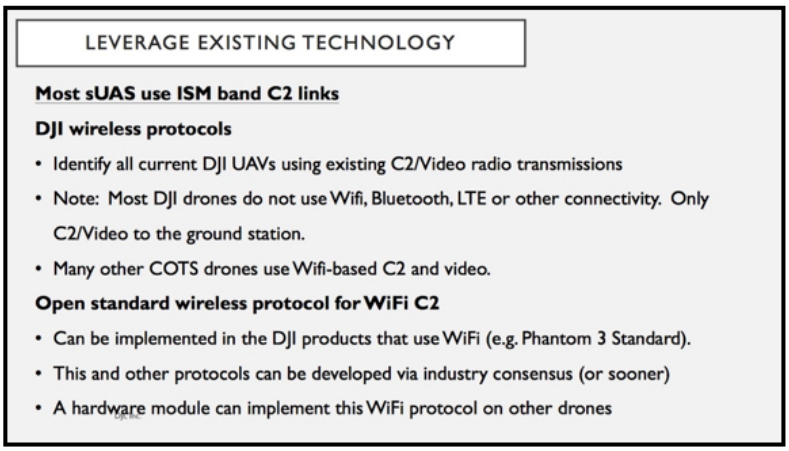

Figure 4. Open call to use existing hardware for Drone ID [16]

[11] https://www.dropbox.com/s/v4lkyr2kdp8ukvx/DJI%20Remote%20Identification%20Whitepaper%203-22-17.pdf?dl=0
[12] https://www.dropbox.com/s/v4lkyr2kdp8ukvx/DJI%20Remote%20Identification%20Whitepaper%203-22-17.pdf?dl=0
[13] https://www.dropbox.com/s/v4lkyr2kdp8ukvx/DJI%20Remote%20Identification%20Whitepaper%203-22-17.pdf?dl=0
[14] https://www.dropbox.com/s/v4lkyr2kdp8ukvx/DJI%20Remote%20Identification%20Whitepaper%203-22-17.pdf?dl=0
[15] https://www.dropbox.com/s/v4lkyr2kdp8ukvx/DJI%20Remote%20Identification%20Whitepaper%203-22-17.pdf?dl=0
[16] https://www.icao.int/Meetings/UAS2017/Documents/Walter%20Stockwell_Stream%20A.pdf

The community raised two main concerns regarding DJI's proposal. First, tracking drone ID data and metadata opens the door for future exploitation. For example, networked solutions increase "the possibility that all UAS operations will be tracked and recorded for future unknown exploitation, including enforcement quotas or business espionage." [17] Second, there is a potential for drone ID system hacking. For example, a networked system may be "susceptible to system-wide hacking, or the creation by detractors of false entries of drone operations that do not exist." [18] It should be noted that some of the risks that were presented are also inherent in "localized" [19] implementations. In some respects, it would be wise for DJI to put the Drone ID technology implementation up for examination via a public review Request for Comments (RFC). The fact that DJI has not put out an RFC, a common practice, may draw criticism from the security community as DJI pushes forward to have their work become the Remote Drone ID standard.

DJI's first generation Drone ID is live and is already generating privacy and security concerns among a subset of researchers and community members regarding the tracking abilities. As an example a recent DJI forum post one user is quoted as saying: "it sounds good in one way but it is more like BIG BROTHER is watching you…" [20] DJI's recent PR event demonstrated to international reporters that their Drone ID technology is live. It was previously stated in March, 2017 that the Technology Readiness Level of the DJI platform was such that "There could be some level of readiness for initial operations this summer 2017" [21]. In a recent ICAO (International Civil Aviation Organization) presentation on "UTM – Registration, Identification and Tracking", [22] it was revealed that DJI's system was "Already deployed at two international airports." [23]

During its ICAO presentation, DJI demonstrated a functioning system, which bolstered its readiness statements in the "What's in a Name" white paper. The system was subsequently dubbed AeroScope" [24] and was touted in various media reports. Walter Stockwell, Director of Technical Standards at DJI, stated:

---

[17] https://www.dropbox.com/s/v4lkyr2kdp8ukvx/DJI%20Remote%20Identification%20Whitepaper%203-22-17.pdf?dl=0
[18] https://www.dropbox.com/s/v4lkyr2kdp8ukvx/DJI%20Remote%20Identification%20Whitepaper%203-22-17.pdf?dl=0
[19] https://www.dropbox.com/s/v4lkyr2kdp8ukvx/DJI%20Remote%20Identification%20Whitepaper%203-22-17.pdf?dl=0
[20] https://forum.dji.com/forum.php?mod=redirect&goto=findpost&ptid=116517&pid=998496
[21] https://www.dropbox.com/s/v4lkyr2kdp8ukvx/DJI%20Remote%20Identification%20Whitepaper%203-22-17.pdf?dl=0
[22] https://www.youtube.com/watch?v=uP6KIVuJsjU
[23] https://www.youtube.com/watch?v=uP6KIVuJsjU
[24] https://www.dji.com/newsroom/news/dji-unveils-technology-to-identify-and-track-airborne-drones

"We have this built into our c2 links for our wireless protocols. Any DJI drone now on the market can be detected by this method... This is really ready, now. We have this packet implemented in our devices... We have receivers developed, built and deployed... they are collecting data as we speak on drones within broadcast range." [25]

Again, it's important to note that DJI's actions are occurring in isolation, without DJI working with the community to address security concerns, or providing information about issues such as how the system works and how data is handled. Some of DJI's approaches have clear security issues with no apparent remediation. Essentially, DJI is playing God with the community's data, and disregarding the outcomes on the community. The community needs to be warned and should assemble a watchdog group to push back and assert transparency.

Issues in DJI's Drone ID Variant

Outlined below is a case study from the DJI reverse engineering group that demonstrates the ability to hack DJI's drone ID system. This is significant because it showcases how the system can be compromised and opens the possibility for malicious use.

One of DJI's potential security issues is the ability for malicious actors to spoof Drone ID. As noted in the previous section, DJI is pitching an Open Source protocol that "Anyone can implement through a software update by adding packets to the Wi-Fi." [26] At the same time, DJI stated that all of their currently marketed drone products have implemented the required packets, and a reference version of the Drone ID specifications. This is important because, as will be discussed later in this paper, a savvy engineer could reverse DJI's implementation for malicious purposes to spoof Drone ID beacons.

Currently, two DJI products include Wi-Fi: the Mavic and the Spark. A cursory glance through archived firmware provided by the DJI Slack Reverse engineering group indicates that DJI implemented the Drone ID features on Mavic in mid-July, 2017.

[25] https://www.youtube.com/watch?v=uP6KIVuJsjU&feature=youtu.be&t=1220
[26] https://youtu.be/uP6KIVuJsjU?t=1248

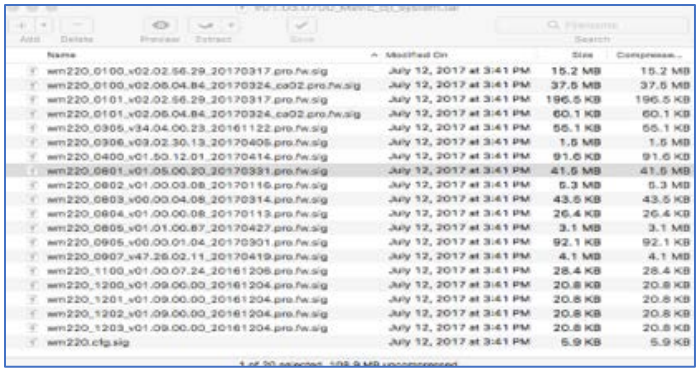

Figure 5. Extracted firmware showing module dates

As an exercise to detect the presence of the Drone ID features, all DJI firmware images can be extracted and every module can be subsequently checked for changes that could identify how the feature was implemented on DJI drones. Once all images are iterated, they can be extracted using a tool known as image.py by Freek Van Tienen. Figure 5 displays how the module creation dates stick out when firmware versions known to have Drone ID are compared with versions that do not have Drone ID. The task of comparing dates can be accomplished by extracting the Android Recovery ROM's from DJI firmware files on any drone in question, using any 'tar' compliant archiving program.

```
$ tar tvf V01.03.0600_Mavic_dji_system.bin | grep 801
-rw-rw-rw-  0              41515328 Jul 19 04:59 wm220_0801_v01.05.00.20_20170331.pro.fw.sig
$ tar tvf V01.03.0900_Mavic_dji_system.bin | grep 801
-rwxrwxrwx  0              41527680 Jul 12 18:31 wm220_0801_v01.05.02.08_20170619.pro.fw.sig
```

Figure 6. 801 Android ROM modules inside DJI firmware

The prefix "0801" is used as an identifier specifically for the Android file system module in DJI firmware files. "0801" firmware modules are similar to regular firmware binaries. They are both regular jar files that can be extracted after being unwrapped, using image.py. Taking MD5 sums of the extracted modules is an easy means of comparing firmware versions.

```
$ ~/Desktop/dji_research/tools/image.py wm220_0801_v01.04.17.03_20170120.pro.fw.sig
{   'auth_key': b'PRAK',
    'blocks_cnt': 1,
    'enc_key': b'IAEK',
    'header_size': 224,
    'image_name': b'0801',
    'magic': b'IM*H',
    'payload_size': 55785632,
    'rsa_sig_size': 256,
    'scramble_key': <__main__.c_ubyte_Array_16 object at 0x10b704400>,
    'sha256_payload': <__main__.c_ubyte_Array_32 object at 0x10b704488>,
    'total_size': 55786112,
    'version': 1}
Unpacking block {   'attrib': 1,
    'name': b'0801',
    'output_size': 55785631,
    'start_offset': 0}
```

Figure 7. Extraction of an Android 801 module using image.py

Many DJI firmware versions use the same kernel module, which is useful when trying to determine whether Drone ID is enabled on a specific drone. In the image below, two of the kernel modules were renamed to help show the presence of Drone ID functions. This makes it easy to highlight the MD5 sums in previous versions when compared to more recent versions.

```
$ md5 `find *801 -name "ath6kl_*.ko"`
MD5 (wm220_0801_v01.04.17.03_20170120.pro.fw_0801/lib/modules/ath6k/ath6kl_usb.ko) = b4808de30d31ec3e0f49ad2aa679e8e9
MD5 (wm220_0801_v01.05.00.08_20170227.pro.fw_0801/lib/modules/ath6k/ath6kl_usb.ko) = b4808de30d31ec3e0f49ad2aa679e8e9
MD5 (wm220_0801_v01.05.00.20_20170331.pro.fw_0801/lib/modules/ath6k/ath6kl_usb_nodroneid.ko) = b4808de30d31ec3e0f49ad2aa679e8e9
MD5 (wm220_0801_v01.05.02.08_20170619.pro.fw_0801/lib/modules/ath6k/ath6kl_usb_droneid.ko) = 3d07a732ac90667c618144f46eb3cee4
```

Figure 8. MD5 sums shown across various firmware versions

Extracting strings from the kernel module ath6kl_usb.ko, which is used for the wireless network card inside DJI Mavic and Spark, identifies the presence of Drone ID functionality.

```
--- nodrone.txt 2017-10-23 14:25:42.411620460 +0200
+++ drone.txt   2017-10-23 14:25:55.154699461 +0200
@@ -705,6 +705,8 @@
 disconnect reason is %d
 disconnect_timeout
  dist_flags      : 0x%X
+dji_ie
+dji_key
 do_send_completion
 dot3_2_dix error
 down_interruptible failed
@@ -724,6 +726,7 @@
 EAPOL %s_handshake_protect reject scan
 !\Ecfg80211_connect_result
 ecfg80211_testmode_event
+ECFm
 eeth_type_trans
 eid %d is not mapped!
 eid %d is not mapped!
@@ -872,6 +875,7 @@
      flag: %d
 flags 0x%x snr_comp %d last_update %lld now %lld
 flags: %d, mode: %d, automatic: %d, dataThresh: %d, rssiThresh: %d
+flight_info
 flowctrl_info: num_of_conn=%d ac_map[0]=0x%x ac_map[1]=0x%x ac_map[2]=0x%x
   Flushing RX packet:0x%lX, length:%d, ep:%d
 Flush pending connect work first.
@@ -1702,7 +1706,7 @@
 spurious upper snr threshold event: %d
 %s: (queue:0x%lX depth:%d)
 %s: queue: 0x%lX, pkts %d
-srcversion=45545E76072F37A8B749181
+srcversion=921082C57284991C3D0A9E2
 %s: reason=%u
 %s: reason=%u, proto_reason %u
 %s: rec power
```

Figure 9. Extracted strings from ath6kl kernel model

Disassembly will, of course, provide a better picture. In this case, the DJI code makes use of ath6kl_wmi_set_appie_cmd(), from wmi.c in the Android Ath6kl USB driver, to send 'flight_info' data. Figure 10 displays a sample flight_info entry in hex and ASCII format.

```
ath7kl — hexedit flight_info — 86×9
00000000   11 30 38 52  58 58 58 58  30 30 58 58  58 4C 38 00  .08RXXXX00XXXL8.
00000010   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000020   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000030   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000040   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000050   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000060   00 00 00 00  00 00 00 00  40 00 00 00  03 00 00 00  ........@.......
00000070   14 58 58 58  58 58 05 B7  58 58 05 B7  00 00 00 00  .XXXXX..XX.....
-**  flight_info       --0x7F/0x81-----------------------------------------------
```

Figure 10. Flight_info file shown in hex

```
11  if ( evt[3].msg_id & 8 )                        // OSD Message -> Motors on bit
12  {
13    duss_osal_mutex_lock(v5, -1);
14    *((_BYTE *)v3 + 848) = 1;
15    duss_osal_mutex_unlock(v3[211]);
16    if ( !last_motor_9924 )                       // Check if broadcasting wasn't started
17    {
18      net_start_regulation_broadcast(v3);         // Start broadcast Flight Info and Flight purpose
19      if ( v3[193] == 1 )
20      {
21        v6 = "echo 8 > /sys/kernel/debug/ieee80211/phy0/ath6kl/fixed_rates";
22LABEL_8:
23        system(v6);
24        goto LABEL_9;
25      }
26    }
27  }
28  else
29  {
30    duss_osal_mutex_lock(v5, -1);
31    *((_BYTE *)v3 + 848) = 0;
32    duss_osal_mutex_unlock(v3[211]);
33    if ( last_motor_9924 == 1 )                   // Check if broadcasting was started
34    {
35      net_stop_regulation_broadcast();            // Stop broadcast Flight Info and Flight purpose
36      if ( v3[193] == 1 )
37      {
38        v6 = "echo 0xFF > /sys/kernel/debug/ieee80211/phy0/ath6kl/fixed_rates";
39        goto LABEL_8;
40      }
41    }
42  }
43LABEL_9:
44  last_motor_9924 = (LOBYTE(v4[3].msg_id) >> 3) & 1;// OSD Message -> motors on
```

Figure 11. Drone ID checking for motors being started before broadcast

Raw text strings alone provide enough information to start deconstructing the DJI Drone ID implementation. The "flight_info" string jumped out immediately as something important. Combing the drone's file system for "flight_info" led us to disassemble some of the functions inside the target dji_network. Disassembling dji_network, in turn, yielded more detail about how DJI drones enable and disable Drone ID functions via an interface in /sys for the ath6kl kernel module.

```
    {
      v16 = flight_info_len;
      if ( flight_info_len )
      {
        beabuf[0] = 0xDDu;
        *(_WORD *)&beabuf[2] = 0x3726;
        *(_WORD *)&beabuf[4] = 0x5812;
        *(_WORD *)&beabuf[6] = 0x1362;
        memcpy(&beabuf[8], flight_info, flight_info_len);
        beabuf[1] = v16 + 6;
      }
    }
    ath6kl_wmi_set_appie_cmd(v4[17], v12->fw_vif_idx, 0, beabuf, beabuf[1] + 2);
    dji_ie_on = 1;
    return v5;
```

Figure 12. ath6kl_wmi_set_appie_cmd used for Drone ID

As can be seen in Figure 13, disassembled dji_network code flight_info is read in as text, and is then passed on as a beacon to the Wi-Fi subsystem. Normally, the flight_info is null, and would not be populated until the drone's motors have started. Intercepting reads and/or writes to this file would directly allow the Drone ID file's content to be manipulated.

```
void __fastcall reg_broadcast(void *p)
{
  uint64_t v1; // r4
  int v2; // r0
  int v3; // r4
  int v4; // r0
  int v5; // r4
  Flight_Reg_Info flight_reg_info; // [sp+0h] [bp-98h]

  v1 = (seq_9138++ + 1) & 1;
  if ( v1 )
  {
    if ( !flight_regulation_lock )
      return;
    duss_osal_mutex_lock(flight_regulation_lock, -1);
    memcpy(&flight_reg_info, &g_flight_purpose, 76u);
    g_fp_updated = 0;
    duss_osal_mutex_unlock(flight_regulation_lock);
    v2 = open("/sys/kernel/debug/ieee80211/phy0/ath6kl/flight_info", 1);
    v3 = v2;
    if ( v2 > 0 )
      write(v2, &flight_reg_info, 129u);
    close(v3);
  }
  else
  {
    if ( !flight_regulation_lock )
      return;
    duss_osal_mutex_lock(flight_regulation_lock, -1);
    memcpy(&flight_reg_info, &g_flight_reg_info, 76u);
    g_fr_updated = 0;
    duss_osal_mutex_unlock(flight_regulation_lock);
    v4 = open("/sys/kernel/debug/ieee80211/phy0/ath6kl/flight_info", 1);
    v5 = v4;
    if ( v4 > 0 )
      write(v4, &flight_reg_info, 76u);
    close(v5);
  }
  system("echo 1 > /sys/kernel/debug/ieee80211/phy0/ath6kl/dji_ie");
}
```

Figure 13. dji_ie interface to enable / disable Drone ID

Patching either the kernel module, or the dji_network, would also directly allow a workaround to the claim of a "unique ID built into the hardware that the user can't change or spoof. [27] "It is actually quite trivial to spoof Wi-Fi-based Drone ID data at this point. Due to a memcpy() error found by Freek van Tienen, the Drone ID packets are truncated to 76 bytes in length, although they may actually be much longer. Regardless of semantics, once "dji_ie" is set to "1" the Drone ID beacon packets are immediately emitted. It should be noted that with root access, end users can actually change the Drone ID packet information, despite DJI claims to the contrary. Jan Dumon has provided code to specifically change the Flight Purpose & DroneID values over a USB connection to any DJI aircraft. [28]

[27] https://www.youtube.com/watch?v=uP6KIVuJsjU

[28] https://github.com/MAVProxyUser/DUMLrub/commit/dd4913d944646e4ae4e6fe9498f969f6e7c0de4e

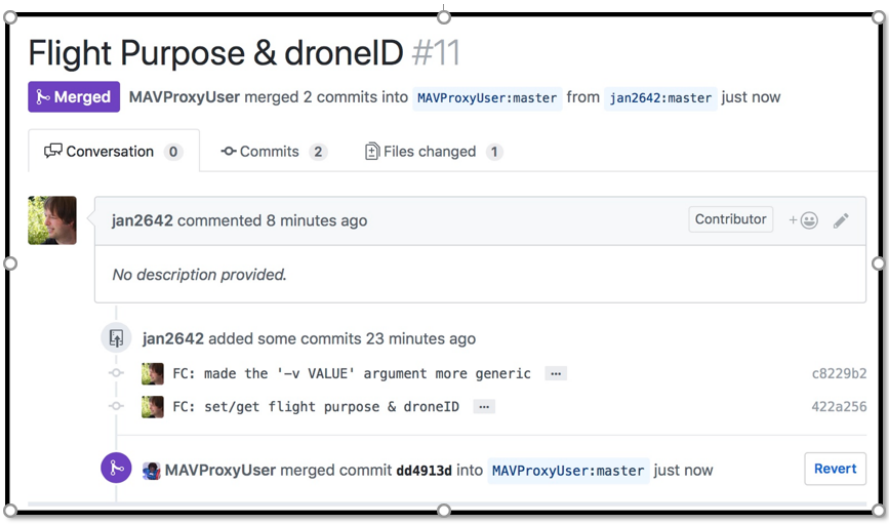

Figure 14. GitHub Repo featuring code to edit Drone ID parameters

A theme in the Drone ID marketing pitch is "Use the Radio that you have". [29] The pitch states that "Anyone with a capable receiver can get tracking and telemetry if they are in range." [30] In the case of Wi-Fi, that is pretty much everyone in the general population. Wi-Fi hardware is a commodity and is ubiquitous. As such, the purveyors of this specific research worked with the author of Kismet Wireless to bring a public Drone ID implementation ahead of DJI's reported open source efforts. This underscores the importance of the community taking action against DJI for operating in isolation. The risk is based on the openness and pervasiveness of radios. DJI is a target to get hacked and the outcome will be privacy concerns for the community.

With the help of the DJI Slack reverse engineering group, Department 13's Mike Kershaw, the author of Kismet, has added Drone ID support to the Kismet Wireless project. Kismet makes use of KaiTai, a declarative binary format parsing language, to extract Drone ID specific fields and values from 802.11 vendor tags contained in DJI products Wi-Fi beacon packets. Kismet integrates the Drone ID packet data into a device object known as 'uav.device'. Kismet tracks the Wi-Fi mac address, the UAV serial number, and the UAV's telemetry history.

[29] https://www.icao.int/Meetings/UAS2017/Documents/Walter%20Stockwell_Stream%20A.pdf
[30] https://www.icao.int/Meetings/UAS2017/Documents/Walter%20Stockwell_Stream%20A.pdf

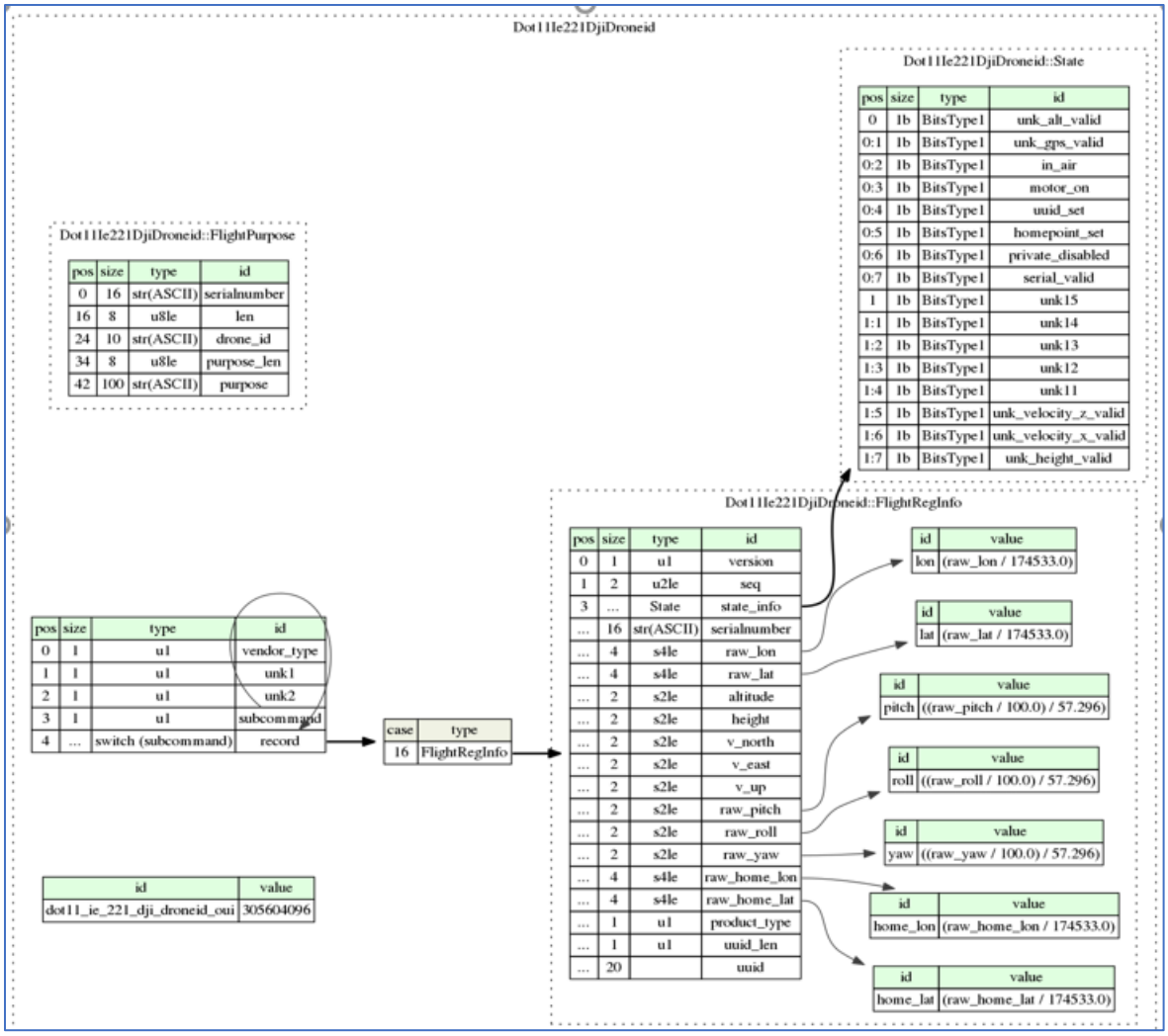

Figure 15. KaiTai packet description of DJI Drone ID packets

Two packet types were documented by the DJI Reverse engineering Slack group's Jan Dumon and Freek Van Tienen. Details on the structure of Flight_Purpose, and Flight_Reg_Info packets inside the dji_network binary are shown in Figure 16. These packets will be sent down the Wi-Fi link, every 200ms in an alternating fashion.

```
(gdb) ptype Flight_Reg_Info
type = struct Flight_Reg_Info_t {
    uint8_t sub_cmd;
    uint8_t ver;
    uint16_t seq;
    uint16_t state_info;
    uint8_t sn[16];
    int32_t longitude;
    int32_t latitude;
    int16_t altitude;
    int16_t height;
    int16_t v_north;
    int16_t v_east;
    int16_t v_up;
    int16_t pitch;
    int16_t roll;
    int16_t yaw;
    int32_t longitude_home;
    int32_t latitude_home;
    uint8_t product_type;
    uint8_t uuid_len;
    uint8_t uuid[20];
}
```

Figure 16. Flight Reg Info Packet

```
(gdb) ptype Flight_Purpose
type = struct Flight_Purpose_t {
    uint8_t sub_cmd;
    char sn[16];
    uint8_t drone_id_len;
    char drone_id[10];
    uint8_t purpose_len;
    char purpose[100];
}
```

Figure 17. Flight Purpose Packet

In Walter Stockwell's talk "UAS Remote ID Use The Radio You Have" [31], he downplayed the potential risk of 3rd parties capturing Drone ID packets. Stockwell stated "By having this ID and tracking limited to the broadcast range, and limited to the time that the person is flying, so someone has to be there with an RX to get it, that is sort of limiting the impact of the data collection." [32]

[31] https://www.youtube.com/watch?v=uP6KIVuJsjU
[32] https://www.youtube.com/watch?v=uP6KIVuJsjU

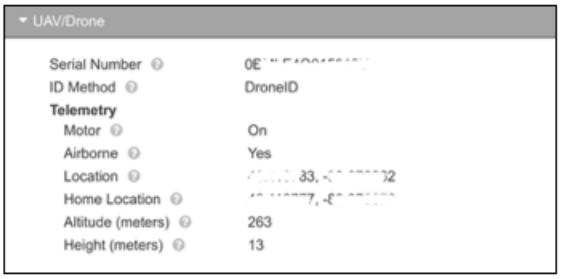

Figure 18. Kismet displaying Drone ID Packet Info

As mentioned previously, several other security attestations have been made about DJI's Drone ID implementation. These attestations include: "not something the user has a choice about turning on, or off" [33], "no issues with person perhaps complaining 'now they are being tracked'" [34], "Tamper proof solution, because it is part of the C2 link" [35], "We can report… Unique ID built into the hardware to the user can't change or spoof that." [36] Figure 19 shows the full extent of data that DJI can report in the current implementation.

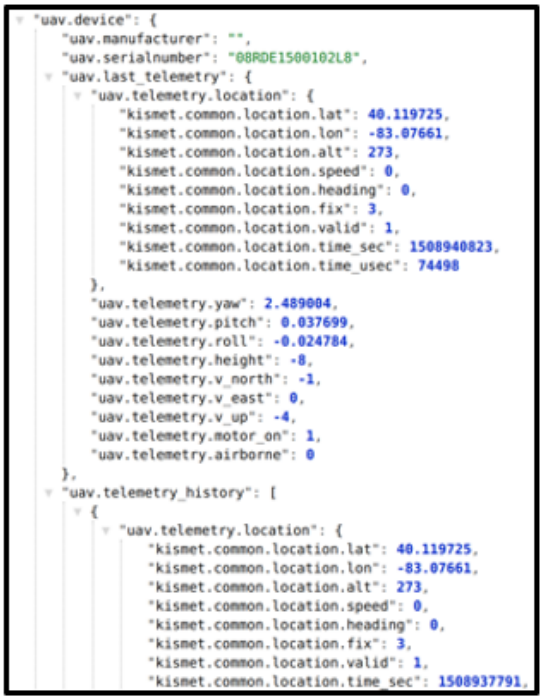

Figure 19. Fully Decoded Drone ID Packet

[33] https://www.youtube.com/watch?v=uP6KIVuJsjU
[34] https://www.youtube.com/watch?v=uP6KIVuJsjU
[35] https://www.youtube.com/watch?v=uP6KIVuJsjU
[36] https://www.youtube.com/watch?v=uP6KIVuJsjU

In the face of the DJI Reverse engineering, jailbreaking, and modification scene, some of these statements are questionable at best. When an end user jailbreaks a DJI drone, a number of these statements become immediately false. In other cases, the statements made are blatantly incorrect, and show a lack of comprehension of various internals of DJI's drone platforms, as well as of general digital privacy concerns.

The statements about whether the system is "tamper proof" [37] are also questionable in the presence of built-in backdoors in DJI's Drone ID implementation. The backdoors allow various aspects of the system to be circumvented, possibly for law enforcement purposes, or other situations in which Drone ID data broadcasts are unwanted. Based on information obtained via disassembly done in private by Freek van Tienen, there appears to be a "private mode" in the DJI flight controller settings. This mode allows for one of the following four options:

- Option 1: Disable the sending of state information in the `flight_info`
- Option 2: Disable the sending of a home location
- Option 3: Hide drone ID in the flight purpose packet
- Option 4: Send "fake" instead of a real serial number

In his speech to the ICAO, Walter Stockwell stated that he believed in "encouraging people that the ID's tracking benefits outweigh the negative prospect that sensitive personal or commercial information may be publicly available." [38] It may be worth having a larger public debate on the security ramifications in question. Many of the dumbed down examples seem to blatantly miss larger privacy issues. For example, an aspect that needs deeper analysis is the constant comparison to license plate technology as a means of downplaying data privacy risks. Stockwell, for example, mentioned "In theory it is sort of like if you are speeding in a car, if someone is there to see you speed, they can catch you and pull you over, if you are speeding in the middle of the country, and no one catches you, I guess that is not a bad thing, you haven't hurt anyone, why do you need to be tracked for that?" [39]

When laying out potential scenarios of focus, and points of concern for future implementations, DJI likened the concept of Drone ID to how automobile license plates are used. It is not difficult to draw simple privacy and security parallels between Drone ID, and

[37] https://www.wired.com/2015/05/virginia-man-sues-police-license-plate-database/
[38] https://arstechnica.com/tech-policy/2013/12/boston-police-indefinitely-suspends-license-plate-reader-program/
[39] https://www.bostonglobe.com/metro/2013/12/14/boston-police-suspend-use-high-tech-licence-plate-readers-amid-privacy-concerns/B2hy9UIzC7KzebnGyQ0JNM/story.html

conventional license plate technology. The ready-made commentary comparing Drone ID technology to automobile license plates is alarming however, given the existing controversy surrounding Automated License Plate Reader (ALPR) technology. There are fundamental differences in an RF-based solution that make the comparison slightly flawed. For example, RF works through walls, whereas license plates cannot currently be detected on a car inside a garage. Drone ID would, however, have such a detection capability. Reports of what appear to be non-law enforcement groups using ALPR technology to make a profit have become more prevalent as hardware becomes more available. This concept alone specifically draws added concern to the Drone ID comparison to license plates.

Figure 20. Non-Law Enforcement use of ALPR

There has been no shortage of ALPR-related privacy concerns as the technology has become increasingly widespread.

When considering an automobile license plate as a proxy for a drone ID:

- The privacy issues that are related to license plates also relate to drone ID.
- Drone ID privacy issues are exacerbated by the greater potential for system hacking.
- The analysis outcomes of using automated license plate readers and their associated bottlenecks also apply to drone ID.
- There is already a pushback on license plate tracking, so the same may hold true for drone ID.

Recently in Virginia, a complaint asserted that the database in which ALPR data was stored violates a Virginia statute: the Government Data Collection and Dissemination Practices Act. The Act prohibits government agencies from unnecessarily collecting, storing, or disseminating the personal information of individuals. [40] In another example, "The Boston

[40] https://www.eff.org/deeplinks/2015/10/license-plate-readers-exposed-how-public-safety-agencies-responded-massive

Police Department has suspended their use of license plate scanners for now. It seems the optical character recognition technology was working just fine, but the department wasn't following up on all of the hot crime fighting leads the technology was generating." At one point Boston also had to discontinue using its ALPR because of a leak to the media that was not redacted "revealing full plate numbers and GPS location data for more than 40,000 different vehicles, most of which belonged to private citizens." [41] The struggle with license plate privacy should not be taken lightly, especially when used as a comparison for Drone ID.

The Electronic Freedom Foundation has gone so far as to say "As longtime critics of mass surveillance systems, EFF would like nothing more than to see a law enforcement agency take its ALPR networks offline. [42] The specific report in which this statement was made discussed the vulnerability of the data that is hosted online for the ALPR systems. The parallels in license plate security and Drone ID security should certainly be taken into account. It would be foolish not to heed existing warnings. "Eight Days in the Life of Oakland's Automatic License Plate Readers" [43] is an excellent documentary that warns of license plate reading technology and its potential ramifications. Is there a reason that Remote Drone Identification technology should be treated any differently? This is a topic that is still up for debate.

As a means of showcasing the potential security and privacy vulnerabilities of a system such as Drone ID, publicly available systems for tracking flights should be considered. Traditional aircraft beacons, such as ADS-B, make use of the same principle and concepts as those used with Drone ID. Specifically, ADS-B can only be received when it is in range. Similar to Drone ID anyone with a capable receiver can view the traffic in question. Companies such as Flightradar24 have capitalized on this limited range broadcast data by supplying hardware that can receive ADS-B for people to place in their homes all around the world. Flightradar24 has provided this hardware for free, in effect, to extend their network reception capabilities in turn facilitating an online database with world wide access.

One unintended side effect of the "Use the Radio that you have" [44] concept is that Drone ID packets can not only be received by anyone, they can also be transmitted by anyone. Unlike ADS-B, it is legal for anyone to transmit Wi-Fi packets that contain Drone ID information. Security researcher RenderMan has already openly demonstrated the concepts that apply to

[41] https://www.youtube.com/watch?v=FqMkIsUmPcI
[42] https://www.icao.int/Meetings/UAS2017/Documents/Walter%20Stockwell_Stream%20A.pdf
[43] https://www.icao.int/Meetings/UAS2017/Documents/Walter%20Stockwell_Stream%20A.pdf
[44] https://www.icao.int/Meetings/UAS2017/Documents/Walter%20Stockwell_Stream%20A.pdf

spoofed aircraft transponder data. Drone ID spoofing is no different, short of the fact that in this sample use case, both the risk and difficulty levels are very low due to Wi-Fi-based technology being used as the foundation. Spoofing ADS-B can be both expensive and risky, although costs have come down in recent years.

By using tools such as Scapy, spoofing Wi-Fi based Drone ID packets is a fairly simple task for persons with basic Python programming skills, and a Wi-Fi card that is capable of raw packet injection.

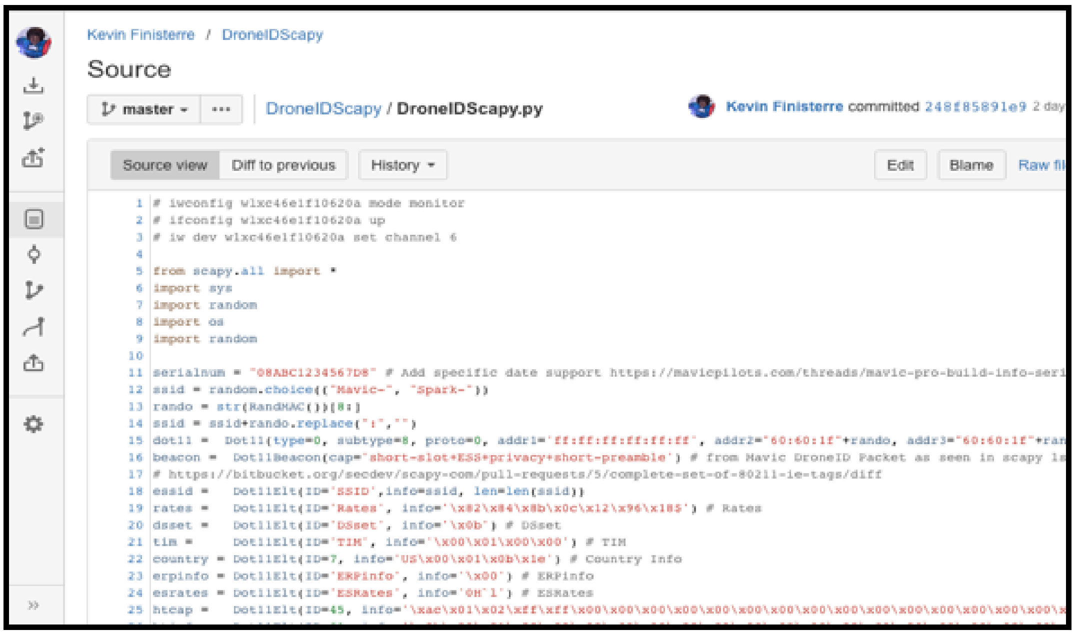

Figure 21. Scapy code that demonstrates Wi-Fi Drone ID Spoofing

For persons who are less adept at programming, an ESP8266 will suffice as a platform to enable low cost, low skill, casual spoofing of Drone ID packets in Wi-Fi bands. Thanks to Wi-Fi_send_pkt_freedom()-based packet injection via MicroPython, small portable ESP8266-based Drone ID "throwies" are something that anyone can create, given enough time and a small financial investment.

Other DJI C2 links are also susceptible to being abused. However, the technology required to spoof C2 links, such as LightBridge and Occusync, are tightly held by hackers and CUAS companies alike. It is not currently feasible to distribute a low-cost OccuSync, or Lightbridge throwie. However, low cost options may become available in the near future.

In conclusion, it is clear that marketing for various future Drone ID-based solutions will mandate a close inspection by qualified subject matter experts before the technology is adopted. In this case, it has been proven that various security attestations may be viewed in a different light when examined by security experts. It is simply not enough to take the word of a company spokesperson, without also examining the practices that have been implemented by the company's engineering team. Both security and privacy hinge on the minute details of each implementation.