# Security Analysis of DJI Phantom 3 Standard

Fernando Trujano, Benjamin Chan, Greg Beams, Reece Rivera

May 11, 2016

**Abstract**

This paper comprises a security analysis of the DJI Phantom 3 Standard [1], a popular consumer drone with an onboard camera and standard set of features. Drones present a novel airborne platform for new commercial and consumer work, and this, in addition to their increasing ubiquity, makes a security analysis especially worthwhile. We present a collection of fatal attacks, minor observations, and associated security vulnerabilities, discovered over the course of a month of research. These vulnerabilities affect both the physical integrity of the drone and the security of user data. Finally, we make a set of recommendations that, if implemented, will dramatically bolster the security of the drone.

## 1 Introduction

In recent years, commercial drones have become increasingly popular among the public, as models such as the DJI Phantom series make the industry more accessible for consumers. These drones are cheaper and easier to fly than ever before. In addition, increased media attention and advances in aerial, battery, and camera technology have made drones especially attractive for endeavors ranging from aerial film projects to research. The growing prevalence of drones in society, however, poses an immediate security concern. When used properly, drones are harmless and are incredibly useful, providing, for instance, beautiful aerial shots and cinematography. However, when misappropriated, drones can directly lead to invasions of privacy, concerns with aircraft safety, and even result in personal injury. It is essential for the owner of the drone to be able to secure it from attack and misuse. Since regulations now require owner registration, and hold owners accountable for illegal misuse, this becomes an immediate consequence for DJI customers, in addition to concerns for public safety. Therefore, the security of commercial drones should be a chief concern for security researchers, the legislative branch, and drone companies.

The following research was performed with permission from DJI through a responsible discolosure process. Please do not publish or discuss findings in this paper until further notice, after DJI has a chance to review and fix any vulnerabilities.

## 2 System Overview

First, we introduce the Phantom 3 Standard system, and its various components.


Figure 1: System overview of the DJI Phantom 3 Standard

### 2.1 Mobile Device

This system supports any iOS or Android mobile device that runs iOS 8+ Android 4.1 and higher. The mobile app serves as the controller for many of the Phantom 3's advanced features. It displays the camera's live feed, displays the drone flight information(altitude, flight path, etc.), and handles many drone flight restrictions. It also, allows the user to manage the WiFi settings of the drone, such as resetting the password.

### 2.2 Controller

The controller serves as the main source of communication to the drone. It creates a mobile WiFi hotspot operating at 2.400GHz-2.483GHz that the mobile device connects to. The mobile device sends instructions and settings to the controller, and the controller relays these instructions to the drone via a 5.725GHz - 5.825GHz radio signal.

### 2.3 Drone

The drone receives radio signals from the controller to dictate its movement. It also uses a combination of on-board sensors such as GPS and barometer readers to stabalize its position in order to combat the wind. Additionally, the Phantom 3 Standard offers several "Smart" features such as "Return to Home" (R2H) that allow the drone to fly autonomously to a given location. The drone communicates back flight data and sensor readings as well as live video feed to be displayed on the mobile device.

## 3 Security

As explained briefly above, security of drones is of extreme importance. First, a customer would want to protect their $500 investment. Most importantly, drones in the US are required by FCC law to be registered to each individual. Therefore a hijacked drone used for malicious purposes could be traced back to the original owner. An attacker, which we define as any person or entitiy with malicious intent within the vicinity of the drone's WiFi network, should not have any access to the system. More specifically, we have determined the following security policy for a camera enabled drone.

1. Any vulnerabilities that can interfere with control of a drone mid flight are critical and should be dealt as so. Such vulnerability would allow an attacker to hijack and steal the drone or interfere with its operation in a way that forces it to crash into the ground.

2. As a secondary goal, we acknowledge the importance of securing data generated by the drone. This includes flight logs and media taken by the camera. Gaining unauthorized access to either of these would constitute a violation of the security policy. Furthermore the ability to modify or insert any additional data would embody an even greater security violation.

3. The user, defined as the owner of the drone or whoever has the remote controller should be able to maintain complete control of the drone. The user should also be able to easily access flight logs and media generated by the camera. A user should not be able to tinker with restrictions set forth by FCC and enforced by DJI via software such as height restrictions and no-fly zones.

# 4 Previous Work

Previous drone hacks generally involve unprotected WiFi, the ability to connect multiple devices, or the ability to gain access to settings during flight. As we will soon cover, many of these weaknesses, reported for the Phantom 2 and Parrot AR 2.0, do not apply specifically to the Phantom 3.

## 4.1 WiFi Insecurities

The Parrot AR 2.0 drones allow multiple connections through WiFi. This allows for someone to connect and tamper with your systems despite not being the owner. By simply searching for these connections, and using appropriate software already developed, it is possible to take full control of unowned Phantom 2 Vision or Parrot AR 2.0 drones. A special mention should be made for early Phantom 2 Vision versions that reportedly allowed the changing of the SSID mid-flight, which kicked all controllers, allowing a new primary controller to connect (most likely the malicious controller), and as a result, fly away with the drone. This is not immediately applicable to the Phantom 3 Standard.

## 4.2 SkyJack

The ultimate goal of our drone hacking is the hijacking of a drone in flight. The SkyJack / Aircrack software is a demonstration of this exact goal, using a raspberry Pi mounted on a Phantom 2 Vision or Parrot AR 2.0 drone, a malicious user is able to fly and search for open Parrot AR 2.0 networks. Once a network is found, the software connects to the connection, changing the SSID and dropping both users. It immediately reconnects and is able to transmit commands through the malicious Phantom 2 Vision to the hijacked drone. Once the hijacked drone is out of range, there is nothing the original drone user can do to recover it. [11] The Phantom 3 Standard is not affected.

### 4.3 Maldrone

Maldrone is a more generalized software acting as a backdoor. Maldrone can be mounted on any program that is able to connect to the WiFi of another drone and insert a file. Maldrone is able to be silently installed and run on a controller's drone from a malicious drone. Once put on the controller's drone, Maldrone forwards communication between the drone and it's controller to fake ports, effectively intercepting all commands. It can remain benign for as long as needed, and once activated, will forward all controller's communication to dummy ports, while forwarding the hijacker's own commands (made using an emulator on a tablet, laptop, etc) to the drone. The hijacker can then force the drone to fly away from the original owner and claim the drone. In addition, all files on the drone can be taken from the drone without forcing Maldrone to take control of the drone itself. [12] Maldrone does not seem to affect the Phantom 3 Standard; however, this is unverified.

### 4.4 Telnet Vulnerability

Previous WiFi hacks include using WiFi repeater to launch a denial of service on Parrot AR 2.0 drones. Multiple people are able to connect to a drone's WiFi, and despite not necessarily having the privileges to operate the drone, just attempting enough communication can interrupt the original controller's commands. By telnetting into the Parrot AR 2.0 from the hijacker's drone, you can kill the original controller's control program. It is possible to kill the fail safe that allows control to be re-established, as well as hijack the drone itself. This is only possible on the Parrot AR 2.0 with telnet, and is made even more manageable because Parrot AR 2.0 drones have open WiFi. [10]. We are not able to telnet nor DDOS (**Section 5.2**) the Phantom 3 Standard.

### 4.5 GPS Attacks

GPS is the last system that has been shown in the past to allow interference on drones of all sorts. This is an ongoing area of study with military application. Given the extensive research done in this sector, in addition to the possible equipment required, we did not attempt to make further progress in this field. [6]. The Phantom 3 Standard is probably vulnerable to this type of attack.

## 5 Attacks

### 5.1 Packet Capture

DJI offers an SDK to build custom iOS and Android applications to interact with their professional drones. This allows an adversary to isolate certain instructions to the drone by implementing a custom application that sends only very specific commands. These commands could include changing the WiFi password or even resetting the WiFi connection. Then, the adversary uses a packet capture application to identify the packets for each specific instruction. This knowledge can be leveraged to send malicious instructions.

## 5.2 Disconnecting Clients

The DJI Phantom 2 Vision, with very little additional equipment, was exposed to basic Denial of Service attacks simply by sending packets according to some users. In our attempts to replicate these attacks on the Phantom 3, we were unsuccessful. Whether these concerns over the Phantom 2's susceptibility to very basic Denial of Service attacks were unfounded or simply ill-explained, we found nothing to show that the Phantom 3 would be susceptible in the normal use case to only a few phones attempting to cause communication issues.

For the DJI Phantom 3, the drone keeps a queue of controllers who connect through WiFi. Based on simple observation with three phones, the queue is actually structured as a queue, and as one controller disconnects, the next in line is able to connect. Because this requires the original client to disconnect, and the Phantom 3 Standard seems to be resilient to basic Denial of Service, we determined that the actual structure of the Phantom 3's queue did not matter for our purposes.

## 5.3 Network Mapping

We analyzed various components of the drone by analyzing the WiFi network exposed by the drone and controller. We took advantage of Android's VPN feature to allow us to capture outgoing packets from any application, including DJI Go. From this we were able to see that the drone was communicating with three subsystems located at 192.168.1/2/3. We then connected our computer to the WiFi network and use network analysis tools to scan for all open ports for each system. From here, we analyzed each subsystem separately.

### 5.3.1 Camera

We noticed that the DJI GO app received large (several Megabytes) UDP packets from this address. As such, we concluded that these packets must compromise the video stream and 192.168.1.3 refers to the camera submodule. An 'nmap -r' of this IP showed that ports 21, 22 and 23 were open for ftp, ssh and telnet respectively. At this point we noticed that while ssh and telnet are password protected, ftp guest is not so we were already able to gain read access to all camera data using another device connected to the network. We attempted several passwords for ssh and telnet, including 19881209 (a previous root password) and 1234124 (the WiFi password) among others.

Going back to FTP on port 21, we were able to login as root with the password 'Big∼9China', which we acquired while decompiling the DJI go android apk. **Section 4.4** describes how we acquired this password in detail. As root, we were able to have read/write access to all media on the device. This vulnerability could allow an attacker to delete all pictures and footage during flight, or even write additional media.

### 5.3.2 Drone

192.168.1.2 most likely corresponds to the drone itself, as it appears to run linux and is only visible when the drone is on. A quick scan of the network revealed that port 21 and 5678 were open for ftp and rrac respectively. Port 21 is running vsFTPd 3.0.2 which as of the time of this writing, only has one minor

known vulnerability. Despite this, we were able to gain root ftp access using the same password as before. Once logged in as root, we were able to see the entire contents of the linux system that runs the drone, including the etc/passwd and etc/shadow files.

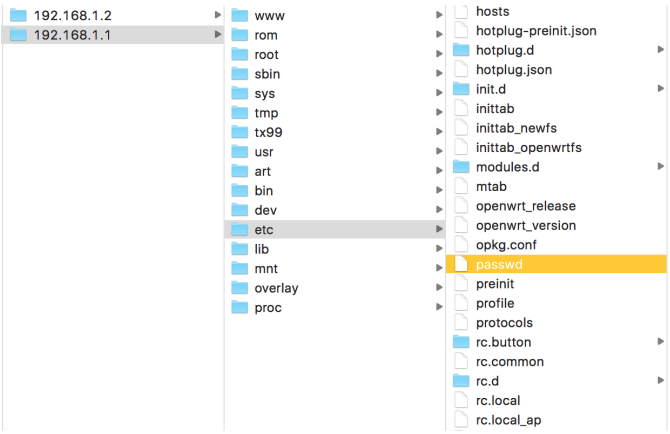


Figure 2: A screenshot of the open root filesystem on the drone and the controller, highlighting the /etc/passwd file

With this information, we discovered that the underlying system is a fork of OpenWRT [9], a Linux operating system for embedded systems. More specifically, system files reveal that OpenWRT 14.07 "Barrier Breaker, r2879, 14.07" built for "ar71xx/generic" is installed. OpenWRT is fairly secure – further work can potentially take advantage of this information.

Surprisingly, we were also able to modify files or write new files to the root. This is a serious vulnerability as an attacker would write a cleverly written init file which would be executed by the drone the next time it is turned on. The system also revealed a config file with the WiFi SSID and password. An attacker could modify this file to change the password, or add nonsense so that WiFi will not work the next time the system boots - rendering the drone unusable. We were able to access and delete files at the root while the drone was mid flight. If an attacker were to delete the entire file system the drone would most likely crashed. We did not test this as the results could be catastrophic and unrecoverable.

### 5.3.3 Controller

The last system lives at 192.168.1.1 and is always exposed regardless of the status of the drone. Therefore we believe this IP corresponds to the WiFi extender on the controller. Similarly, this IP had port 21 open for ftp, which we were able to gain root access to using the same password as before.

This subsystem also runs a fork of OpenWRT [9]. More specifically, it is the exact same version as the subsystem installed on the drone (14.07, Barrier Breaker). Running a diff between the two subsystems (192.168.1.1 and 192.168.1.2) reveals a couple differences, but all are isolated from the main linux system.

```
"groupName": "SkyWiFi",
"ftpPwd": "Big~9China",
"ftpUrl": "192.168.1.2",
"ftpUsername": "root"

"groupName": "GroundWiFi",
"ftpPwd": "Big~9China",
"ftpUrl": "192.168.1.1",
"ftpUsername": "root"
```

Figure 3: raw text taken from a config file in the Android app, containing FTP credentials.

## 5.4 Finding the FTP root password

The DGI Go Android app is easily decompiled with a tool such as Dex2Jar [2] or JADX [3], both of which serve the purpose of converting Java class files to a JAR readable by a tool such as JD-GUI [4], which can then convert it to source. In addition, the APK file bundles a number of assets, including XML files and images.

A grep for 'Pwd' in the app bundle reveals the existence of a config file located at `/dji/res/raw/upgrade_config.json`, which lists the subsystems in the network that can be upgraded over the air. We assume that this is a residue of the software upgrade mechanism for the P3S. In particular, Figure 3 describes the content of this file.

Of key interest are the 'ftpPwd' and 'ftpUsername' entries. Now, we connect to the VSFTP servers located on each subsystem, at 192.168.1.2 and 192.168.1.1, located on port 21 on both systems, using the specified username 'root' and password 'Big~9China' as specified in the config file.

FTPing in this fashion immediately grants the user root access to the entire linux filesystem on both devices, including folders such as /etc and /usr. **Section 5.3** describes the structure of the newly revealed Linux filesystem.

**Section 5.5** describes a related attack that renders it impossible for the Android app to enforce no-fly zones.

## 5.5 Additional Vulnerabilities resulting from the Android App

In the decompiled android app we found a config file, flyforbid.json that contained a list of airports with their corresponding GPS coordinates. We assume that these corresponds to no-fly zones, which DJI enforces via software in order to comply with FCC laws. However, by modifying this file and recompiling the app, one should be able to bypass these restrictions and fly the drone regardless of location.

## 5.6 Password Bruteforcing

### 5.6.1 Telnet, and attempts at finding the Linux root password

Telnet and SSH (on 192.168.1.3) requires a password in order to authenticate. On the other two subsystems, Telnet refuses connections while SSH is not installed.

In order to gain additional access to these systems, we looked into ways of acquiring the root password of each underlying linux filesystem. 192.168.1.3, after all, may share passwords with .1, and .2. In addition, with such a password, we could change the root FTP password (described in **Section 5.4**) whereas we currently cannot with only the FTP password.

We used John the RIPper[7] in an attempt to break the `/etc/passwd` and `/etc/shadow` files acquired from the FTP filedump [5]. We ran the Jumbo version of the tool for 5 days on a 4 core virtual machine in standard MD5 mode. We were unable to obtain the plaintext password.

We also used hydra [8] to attempt to brute force the password. However, given the slow response time to login requests, this method was unfeasible.

In sum, DJI seems to have chosen a strong, secure, hard-to-guess Linux password.

## 5.7 GPS

The drone relies on GPS for many of its primary functions and we suspected that there may be some interesting vulnerabilities with this technology that were worth exploring. We considered the possibility of spoofing the GPS in several ways. First, attempting to change the drone's GPS coordinates for itself. Then, giving the drone false "Home Point" coordinates in order to trick the *Return to Home* feature into returning the drone to another location. The next was to send it false coordinates in a flight path, in order to fly it away from the owner. We realized that we didn't have the resources required to perform these attacks since GPS spoofing has been a known problem and there are many defenses in place to stop these attacks.

## 5.8 Radio

The controller uses radio signals to send flight instructions to the drone. We considered doing a frequency spectral analysis of the signals using a commercial module, HackRF One. This would reveal the specific signals used for each flight instruction, which we could use to send the drone malicious commands. It would also allow us to perform replay attacks on the drone. However, we didn't have the resources to obtain this module. This attack is still promising and is worth future investigation.

# 6 Further Work

The DJI phantom 3 Standard has two USB ports located in the front of the drone and the camera. The USB at the front of the drone is used to provide flight logs when the drone is in flight mode while the USB on the camera provies access to

the content of the Micro-SD card. We did not see any immediate vulnerabilities with these access points but further analysis is needed.

DJI also offers a simulator mode for drones to allow for new pilots to get used to the controls without actually flying the drone. This feature could potentially reveal additional information about how the drone works as all radio information received by the drone is forwarded to the phone to perform the simulation.

We did not look into the security of the phantom software update flow. It would be particularly intersecting to analyze the security of the update server and whether updates are signed by DJI and verified before installation. If the security of the upgrade server were to be compromised, and software was not signed, an attacker could upload malicious software which would be distributed to all drones via a required upgrade.

We considered an attack that relies upon determining the values for the credentials and protocols via packet capture. There are free packet capture methods for android that we utilized, however, the commercial DJI app has many background processes that are continuously running, making it difficult to isolate the necessary packets. Therefore, we decided that a viable attack route would be to use the android SDK to build a custom app in order to isolate the instructions for setting the WiFi password, as well as resetting the wireless connection. The SDK offers methods for both of these operations. An app with these capabilities would also allow an adversary to quickly send the instruction to reset the WiFi, disconnecting the current user, and then immediately connect to the drone before the current user is able to reconnect. However, building an application wasn't in the scope of our project and is left for future implementation.

# 7 Recommendations

Many of the attacks described in here are made possible because of an unsecured WiFi network. While all drones are secured by WPA-PSK2 WiFi, they all share the same default password of 12341234. DJI allows users to change the WiFi password, but this setting is hidden in the app. As such, our main suggestion is to force people to change their WiFi password on first use. This will ensure that drone's network is protected from automatic tools that scan for SSIDs with "PHANTOM" and try default passwords, such as the ones described in section 4.

Additionally, Phantom 3's should not allow root access to their filesystem. This can be done by getting rid of a constant password (Big 9China) and replacing it with a dynamic password that is different for each drone (perhaps related to the serial number). This will ensure that, even if an attacker is able to gain access to the network, they will not be able to enter the file system as root.

Lastly, while we are not completely sure why some ports are open on these systems, we understand that there may be some legitimate reasons to keep them open, such as uploading files before performing a software update. However, these systems should only be exposed when their functions are actually being used. More importantly, there should be no access to FTP in-flight. There is no reason for anyone to need to be able to FTP to an in-flight drone. Refusing communication during this time would be sufficient.

# 8 Conclusion

The Phantom 3 Standard is more secure than its predecessors, yet still fails to meet the unique demands of airborne technology in the context of security. It is essential that drones and related airborne aircraft remain infallible against malicious attempts to control or destroy the drone. In addition, on-board software must mitigate attempts to violate user privacy and limit access to sensitive data.

We believe that if DJI implements our recommendations, as detailed in **Section 7**, the Phantom 3 Standard will be secure enough to withstand the majority of software-centric malicious attacks. Further work is necessary to investigate other attack vectors – which include the update mechanism, GPS, and hardware, among other avenues.

In conclusion, we hope that this work will inspire and inform future security analyses and efforts in the domain of airborne drone technology.

# References


[1] DJI. *Phantom 3 Standard Drone.*
`http://www.dji.com/product/phantom-3-standard`

[2] dex2jar *Tools to work with android .dex and java .class files.*
`https://github.com/pxb1988/dex2jar`.

[3] jadx - Dex to Java decompiler *Command line and GUI tools for produce Java source code from Android Dex and Apk files.*
`https://github.com/skylot/jadx`

[4] JD-GUI *JD-GUI, a standalone graphical utility that displays Java sources from CLASS files.*
`https://github.com/java-decompiler/jd-gui`

[5] Linux Programmer's Manual *crypt, cryptr password and data encryption.*
http://man7.org/linux/man-pages/man3/crypt.3.html

[6] GPS Spoofing *Unmanned vehicle security, GPS spoofing, Counter-UAV*
`https://pdfs.semanticscholar.org/c9d8/5878c56390b614a891d477b90d1b35ceb21b.pdf`

[7] Openwall *John the RIPper password cracker.*
`http://www.openwall.com/john/`

[8] THC *Hydra network logon cracker.*
`https://www.thc.org/thc-hydra/`

[9] OpenWRT. *A Linux distribution for embedded devices*
`https://openwrt.org`

[10] Hak5 *Parrot AR 2.0 Telnet control dropping.*
`https://www.youtube.com/watch?v=Fk1Bpy5ccPU`

[11] Skyjack *Using Skyjack to hijack drones*
`http://samy.pl/skyjack/`

[12] Maldrone *a general backdoor for drones*
http://garage4hackers.com/entry.php?b=3105

[13] Wifi Insecurity *reported wifi SSID changing*
http://forum.dji.com/forum.php?mod=viewthread&tid=6738&page=1#pid45670