Drones Hijacking

multi-dimensional attack vectors and countermeasures

Aaron Luo

Cyber Safety Solution
Agenda

• Introduce the drone architecture
• Vulnerabilities on which component?
• Demo
• Provide the prevention solution
• Release attack/defense tools on GitHub.
Today, our target is...

DJI-Phantom 3 Advanced
DJI Phantom 3A Architecture

**Drone**
- Flight controller
  - 2.4GHz radio module
  - GPS module
- Sensors (compass, Gyroscope, Accelerometer, Barometer...etc.)
- Micro-USB Slug (flight simulating program need this to connect)
- MicroSD Slug (firmware updated usage and photo storage)
- Other Parts (battery, screw propeller, camera, gimbals, pilot lamp)

**Remote Controller**
- 2.4GHz radio module
- USB Slug (I/O function with phone’s App)
- Micro-USB Slug (firmware update usage)
- Other Parts (Joystick, button, lights)

**App/SDK**
- Connect to Remote Control, display drone information (like image of camera, GPS data and Compass)
- Operator Drone (drone takeoff, Automatic return)
DJI Phantom 3A Architecture

- **Drone**
  - Flight controller
    - 2.4GHz radio module
    - GPS module
    - Sensors (compass, Gyroscope, Accelerometer, Barometer...etc.)
    - Micro-USB Slug (flight simulating program need this to connect)
    - MicroSD Slug (firmware updated usage and photo storage)
  - Other Parts (battery, screw propeller, camera, gimbals, pilot lamp)

- **Remote Controller**
  - 2.4GHz radio module
  - USB Slug (I/O function with phone’s App)
  - Micro-USB Slug (firmware update usage)
  - Other Parts (Joystick, button, lights)

- **App/SDK**
  - Connect to Remote Control, display drone information (like image of camera, GPS data and Compass)
  - Operator Drone (drone takeoff, Automatic return)
DJI App/SDK Flow Chart
Crack the SDK Authentication Mechanism

• Download SDK from DJI website

• Find key function with JD-GUI

```java
private void checkPermission()
{
    boolean bool = WifiStateUtil.ping(CF_STAT_TEST_URL, 3000);
    if (!bool)
    {
        if (checkSdkConfigFileExist(mContext))
        {
            dij.internal.a.a.a(sdkManagerCallback, DJISDKError.REGISTERATION_COULDNOT_CONNECT_TO_INTERNET);
            SDK_LEVEL = 0;
            return;
        }
        str1 = getAppID(mContext);
        if (str1 == null) || ("".equalsIgnoreCase(str1))
        {
            dij.internal.a.a.a(sdkManagerCallback, DJISDKError.REGISTERATION_EMPTY_APPKEY);
            SDK_LEVEL = 0;
            return;
        }
        if (str1.length() != 24) || (!AppIdCheck(str1))
        {
            dij.internal.a.a.a(sdkManagerCallback, DJISDKError.REGISTERATION_INVALID_APPKEY);
            SDK_LEVEL = 0;
            return;
        }
        jidi.mong.a.a.a();
        dij.internal.a.a.a(sdkManagerCallback, DJISDKError.REGISTERATION_EMPTY_APPKEY);
        SDK_LEVEL = 0;
        return;
    }
    str2 = getDeviceID(mContext);
    if (str2 == null)
    {
        dij.internal.a.a.a(sdkManagerCallback, DJISDKError.REGISTERATION_INVALID_UUID);
        SDK_LEVEL = 0;
        return;
    }
    checkLocalSdkConfigFile(mContext, str1, str2);
    return;
}
String str1 = getAppID(mContext);
if (str1 == null) || ("".equalsIgnoreCase(str1))
{
    dij.internal.a.a.a(sdkManagerCallback, DJISDKError.REGISTERATION_EMPTY_APPKEY);
    SDK_LEVEL = 0;
    return;
}
if (str1.length() != 24) || (!AppIdCheck(str1))
{
    dij.internal.a.a.a(sdkManagerCallback, DJISDKError.REGISTERATION_INVALID_APPKEY);
    SDK_LEVEL = 0;
    return;
}
if (jidi.mong.a.a.a())
```
Crack the SDK Authentication Mechanism

• Use JBE - Java Bytecode Editor to patch the code
Crack the SDK Authentication Mechanism

• Check the result with JD-GUI

```java
private void checkPermission()
{
    SDK_LEVEL = 2;
}

private static boolean checkSdkConfigFileExist(Context paramContext)
{
    boolean bool = false;
    FileInputStream localFileInputStream = null;
    try {
        localFileInputStream = paramContext.openFileInput("SDK_CONFIG_FILE_NAME");
    } catch (FileNotFoundException localFileNotFoundException) {
        localFileInputStream = null;
    }
    if (localFileInputStream != null) {
        try {
            localFileInputStream.close();
        } catch (IOException localIOException) {
        }
        bool = true;
    }
    return bool;
}
```
Connect to Drone by Cracked SDK

DEMO
Take off/Landing

DEMO
Fly to specified location
How to prevent/improve?

- Protect library file by obfuscator/packer

- Use asymmetric encryption to validate the SDK authentication key between App and Drone and Server, not only just validate from App and Server
Next section:
Firmware Analysis
Firmware Analysis

- Use the “Binwalk” can extract some data, but it is limited.

```
root@ubuntu:/home/hello# binwalk -e P3S_FW_V01.06.0040.bin

<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>HEXADECIMAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>5639310</td>
<td>0x560C8E</td>
<td>LZMA compressed data, properties: 0xC8, dictionary size: 16777216 bytes, uncompressed size: 67108864 bytes</td>
</tr>
<tr>
<td>5639346</td>
<td>0x560CB2</td>
<td>LZMA compressed data, properties: 0x64, dictionary size: 16777216 bytes, uncompressed size: 83886080 bytes</td>
</tr>
<tr>
<td>5639418</td>
<td>0x560CFA</td>
<td>LZMA compressed data, properties: 0xC8, dictionary size: 16777216 bytes, uncompressed size: 134217728 bytes</td>
</tr>
<tr>
<td>5639634</td>
<td>0x560DD2</td>
<td>LZMA compressed data, properties: 0x64, dictionary size: 16777216 bytes, uncompressed size: 134217728 bytes</td>
</tr>
</tbody>
</table>
```
Firmware Analysis

- Use IDA Pro to analyze the incomplete data extracting by the Binwalk
- Use String Reference to find the key function

```c
else if ( sub_E7FE6BC() != 1 )
{
    v8 = *v5;
    sub_E6E1328(
        1,
        1,
        1,
        0,
        0,
        1
    );
    (int)"[DJI_UEG] %s: Invalid FirmwareId! FirmwareId[%d]."
    (unsigned int)"DjiUegPktVersionCheck";
}
```

```c
if ( v39 == 1 && v29 == v28 && !v32 )
{
    ++v29;
    if ( sub_E8774A0(0) == 5 )
        ++v28;
    v1[28] = 3;
    if ( !dword_10B92430 )
        v1[27] = 3;
    v32 = 5;
}
```
Firmware Analysis

- Disassemble and writing the parser

```c
char* check_rom_firmware (char* buffer, int id_major, int id_minor)
{
    unsigned int i;
    for (i = 0; i < 0x21; ++i)
    {
        if (id_major == buffer[188 * i + 132] && id_minor == buffer[188 * i + 136])
            return (char*)buffer[188 * i];
    }
    return 0;
}

int firmware_count = *(unsigned short*)(buffer[0x2C]);
printf("Firmware section count: %d\n", firmware_count);
section_info_header *sh = (section_info_header*)buffer[0x48];
char *rom_offset = &buffer[offset_rom_update_firmware_info];
for (int i = 0; i < firmware_count; i++)
{
    int majorid = sh[i].checksum&0x1F;
    int minorid = sh[i].checksum>>5;
    char *rom_info = check_rom_firmware(rom_offset, majorid, minorid);
    if (rom_info)
    {
        printf("Binary offset: 0x%08x\tMajor: %02d Minor: %02d\tModuleName:
char buf[10];
sprintf(buf, "%d", i);
FILE *fp2 = fopen(rom_info[66], "wb");
fwrite(buffer+sh[i].offset,1,sh[i].size,fp2);
fclose(fp2);
    }
    else
    {
        hexdump(&sh[i].sizeof(section_info_header));
        printf("Binary offset: %x\tMajor: %02d Minor: %02d\tOffset: 0x%08x\n",
    
sub_E6DB488((int)(&v1[10] * *v4 + 29), *(__BYTE *)(sub_E6DB488((int)&v1[10] * *v4 + 30) = (unsigned int)(sub_E6DB488((int)&v1[10] * *v4 + 32) = *(__BYTE *)(sub_E6DB488((int)&v6 = sub_E676FBC(*(v5));
v7 = (unsigned __int8 *)(v6 + 76); (char *)(v3 + 4), 4u);
```
Finally we can extract each firmware module with detailed information.

Firmware Analysis

<table>
<thead>
<tr>
<th>Offset</th>
<th>Major</th>
<th>Minor</th>
<th>Module Name</th>
<th>Binary Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000034e</td>
<td>03</td>
<td>05</td>
<td>MCLDR</td>
<td>PMCLDRFw3.bin</td>
<td>43776</td>
</tr>
<tr>
<td>0x0000ae4e</td>
<td>03</td>
<td>06</td>
<td>MCAPP</td>
<td>PMCAPPFw3.bin</td>
<td>786432</td>
</tr>
<tr>
<td>0x000c4e</td>
<td>04</td>
<td>00</td>
<td>GIMBAL</td>
<td>PGIMBALFw3.bin</td>
<td>93696</td>
</tr>
<tr>
<td>0x000e1c4e</td>
<td>11</td>
<td>00</td>
<td>BATTERY</td>
<td>PBATTERYFw3.bin</td>
<td>19140</td>
</tr>
<tr>
<td>0x000e6712</td>
<td>12</td>
<td>00</td>
<td>ESC0</td>
<td>PESC0Fw3.bin</td>
<td>42496</td>
</tr>
<tr>
<td>0x000f0d12</td>
<td>12</td>
<td>01</td>
<td>ESC1</td>
<td>PESC1Fw3.bin</td>
<td>42496</td>
</tr>
<tr>
<td>0x000fb312</td>
<td>12</td>
<td>02</td>
<td>ESC2</td>
<td>PESC2Fw3.bin</td>
<td>42496</td>
</tr>
<tr>
<td>0x00105912</td>
<td>12</td>
<td>03</td>
<td>ESC3</td>
<td>PESC3Fw3.bin</td>
<td>42496</td>
</tr>
<tr>
<td>0x0010ff12</td>
<td>15</td>
<td>00</td>
<td>68013</td>
<td>P68013Fw3.bin</td>
<td>2680</td>
</tr>
<tr>
<td>0x0011098a</td>
<td>17</td>
<td>00</td>
<td>MUOM4</td>
<td>PMUOM4Fw3.bin</td>
<td>77876</td>
</tr>
<tr>
<td>0x001239be</td>
<td>17</td>
<td>01</td>
<td>MUOM0</td>
<td>PMUOM0Fw3.bin</td>
<td>25908</td>
</tr>
<tr>
<td>0x00129ef2</td>
<td>19</td>
<td>00</td>
<td>FPGA</td>
<td>PPFGAFw3.bin</td>
<td>4194304</td>
</tr>
<tr>
<td>0x0052ef2</td>
<td>01</td>
<td>00</td>
<td>FC300S</td>
<td>PFC300SFw3.bin</td>
<td>56766764</td>
</tr>
<tr>
<td>0x003b4d01e</td>
<td>01</td>
<td>01</td>
<td>CAM0MLDR</td>
<td>PCAMMLDRFw3.bin</td>
<td>412780</td>
</tr>
<tr>
<td>0x003bbc8a</td>
<td>09</td>
<td>00</td>
<td>1765</td>
<td>P1765Fw3.bin</td>
<td>81284</td>
</tr>
</tbody>
</table>

Press any key to continue.
Firmware Analysis

- Extract UBI file system from PFC300SFw3.bin
Firmware Analysis

- extract some interesting things from file system (for example, ssh key data and configuration, /etc/shadow...etc.)

```bash
# IdentityFile ~/.ssh/id_rsa
# IdentityFile ~/.ssh/id_dsa
# Port 22
# Protocol 2,1
# Cipher 3des
# Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc
# MACs hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160
# EscapeChar ~
# Tunnel no
# TunnelDevice any:any
# PermitLocalCommand no
# VisualHostKey no
# ProxyCommand ssh -q -W %h:port gateway.example.com

ssh_config

2048 65537 2484854675734093150147007021859700565427952096031683099734234232144
\n# This is the sshd server system-wide configuration file. See
# sshd_config(5) for more information.
# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin
# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.
#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
# The default requires explicit activation of protocol 1
#Protocol 2
#HostKey for protocol version 1
#HostKey /etc/ssh_host_key

---BEGIN EC PRIVATE KEY-----
MHeCAQEERE+SmCrTkQLjtjYo1qtN8RRhscjItz9V9DOLVFV6UstHJoAoGCCqGSM49
AwEHoUQpDQgAERUf5/zwVCgG6VE2bTs/g9AVUHye+fcgPe9pMW6zhY34Z3yHk86LhGg
103vHcF+4aIckTYect/dY2Koe9uaphbgZQ==
-----END EC PRIVATE KEY-----
```
How to prevent/improve?

- Encrypt the firmware binary, the encryption key storage on hardware, but still need extra careful about storage place must be safety, and the side channel attack.
Next section:
Radio Signal Analysis
Radio Signal Analysis

• How to?

• Buy the SDR (software-defined radio)
Radio Signal Analysis

P3A use two modulation/demodulation
to transfer data with 2.4GHz ISM band
RC to Drone radio spectrum (FHSS)

- Control drone direction (up down left right)
- Frequency 2.400~2.483GHz, each channel about 1MHz
DSSS - Drone to RC radio spectrum

• For drone to remote controller image transmission

• Frequency 2.4015~2.4815 GHz
  • split into 6 channels, each channel is about 10MHz
Finally we found...

- Images have no checksum mechanism, so we can jamming the radio frequency to show wrong image to controller
How to prevent/improve?

• Validate the image checksum

• Transfer the image data by asymmetric encryption (but it need more performance, in this case I think just add the checksum is enough, because reverse the modulation/demodulation are difficult)
Next section:

GPS Modules
GPS Modules

• GPS Modules is general way to hijacking the drone

• The GPS for commercial protocol (C/A code) is open and not encrypt, so attacker can easily to fake this
Which function is associated with GPS?

- No-fly zone
- Return to home
- Follow me
- Waypoint
How to spoof the GPS location?

• There have a good open-source GPS simulator in GitHub, called gps-sdr-sim, but it have some limitation, before you want fake a location, should wait for few minutes to generate the I/Q data

• So we improve the code, let it can in real-time generate GPS signal and can be controlled with the joystick.
Control GPS by Joystick

DEMO
Force landing the drone by no-fly zone
Hijacking Drone by Joystick

DEMO
How to detect the fake GPS signal?

• Validate the GPS sub-frame data
How to detect the fake GPS signal?

- Validate the time between satellite time and real time
How to detect the fake GPS signal?

• Check the motion speed between point to point
  • For example it is impossible to change your location from Taiwan to Las Vegas in one second
Develop the fake GPS detector

- Board: RaspberryPI
- GPS modules: u-blox
Detect Fake GPS Signal

DEMO
Catch The Bad Guys
Conclusion

• Developer should know about the risks of each component.
Acknowledge

• Trend Micro
• All the Cyber Safety Solution partners
• All the support of my friends
Thank You!

If you have any question, please mail to
aaronluo17@gmail.com

GitHub: Aaron-Luo