Drones Hijacking

multi-dimensional attack vectors and countermeasures

Aaron Luo

Cyber Safety Solution

Agenda

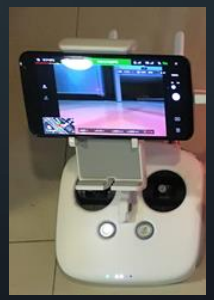
- Introduce the drone architecture
- Vulnerabilities on which component?
- Demo
- Provide the prevention solution
- Release attack/defense tools on GitHub.

Today, our target is…

DJI-Phantom 3 Advanced

DJI Phantom 3A Architecture

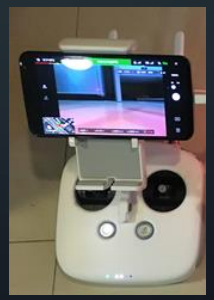
- Drone
  - Flight controller
    - 2.4GHz radio module
    - GPS module
    - Sensors (compass, Gyroscope, Accelerometer, Barometer…etc.)
    - Micro-USB Slug (flight simulating program need this to connect)
    - MicroSD Slug (firmware updated usage and photo storage)
  - Other Parts(battery, screw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

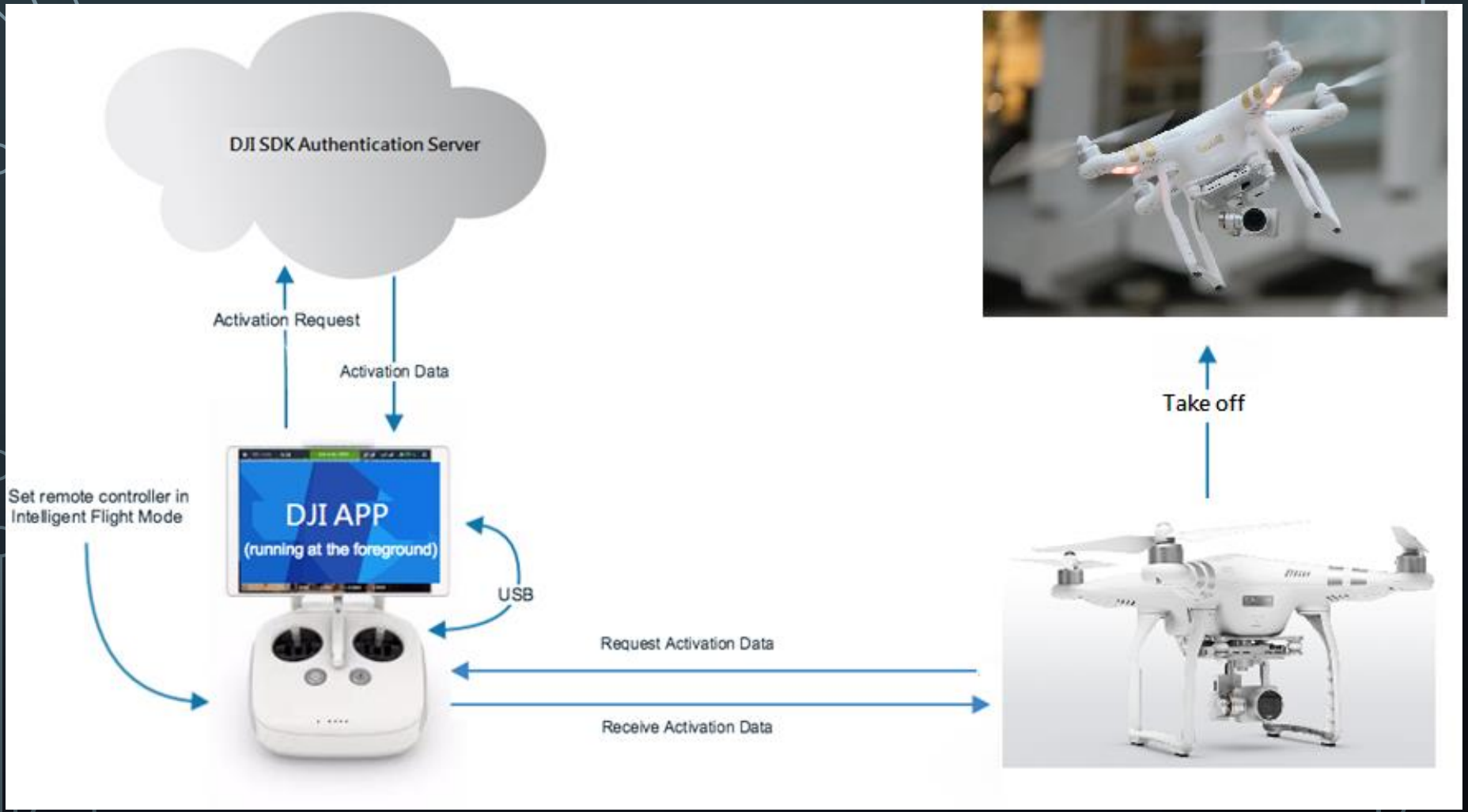
DJI Phantom 3A Architecture

- 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)



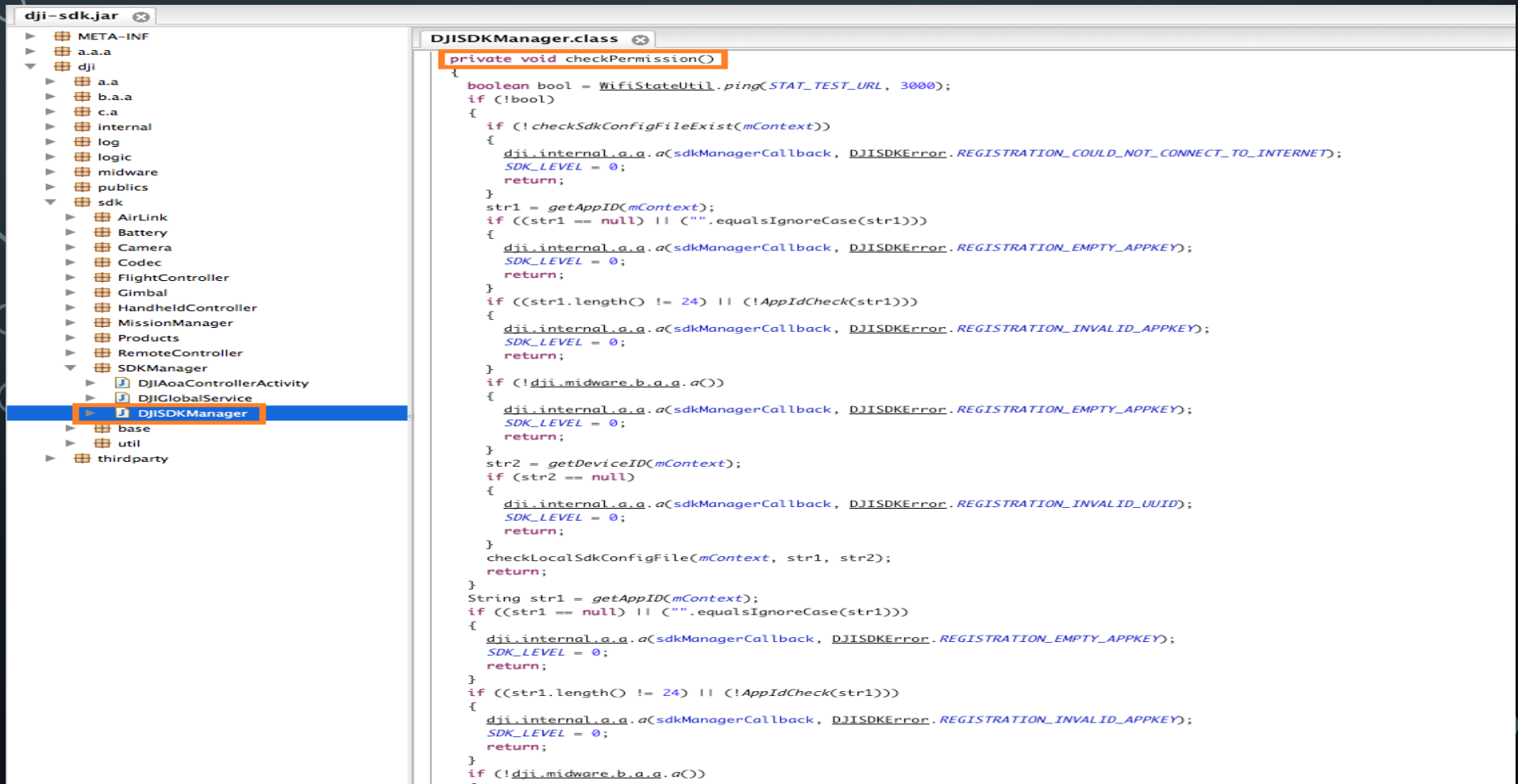
- App/SDK
  - Connect to Remote Control, display drone information (like image of camera, GPS data and Compass)
  - Operator Drone (drone takeoff, Automatic return)

DJI App/SDK Flow Chart



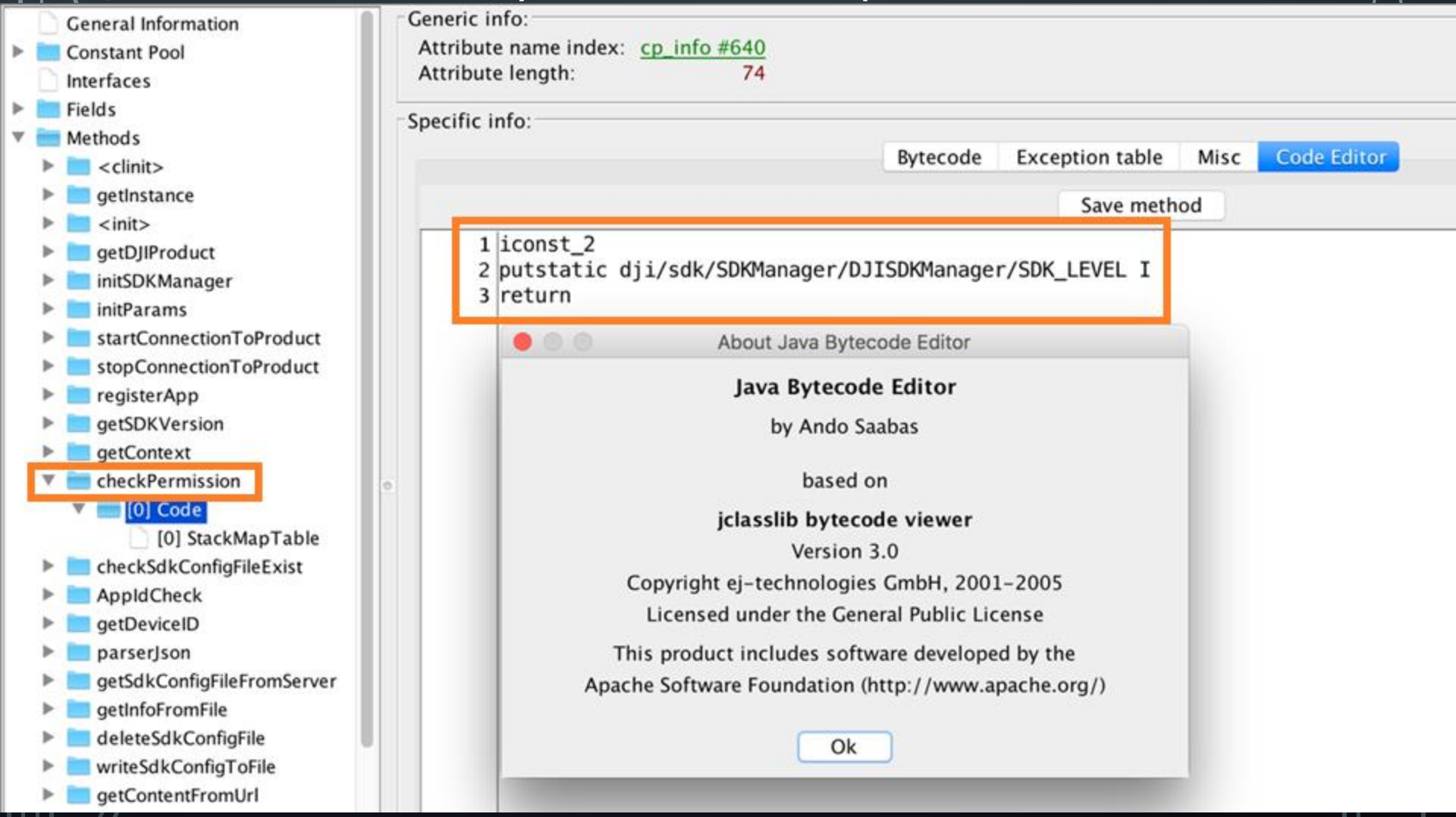
Crack the SDK Authentication Mechanism

- Download SDK from DJI website
- Find key function with JD-GUI



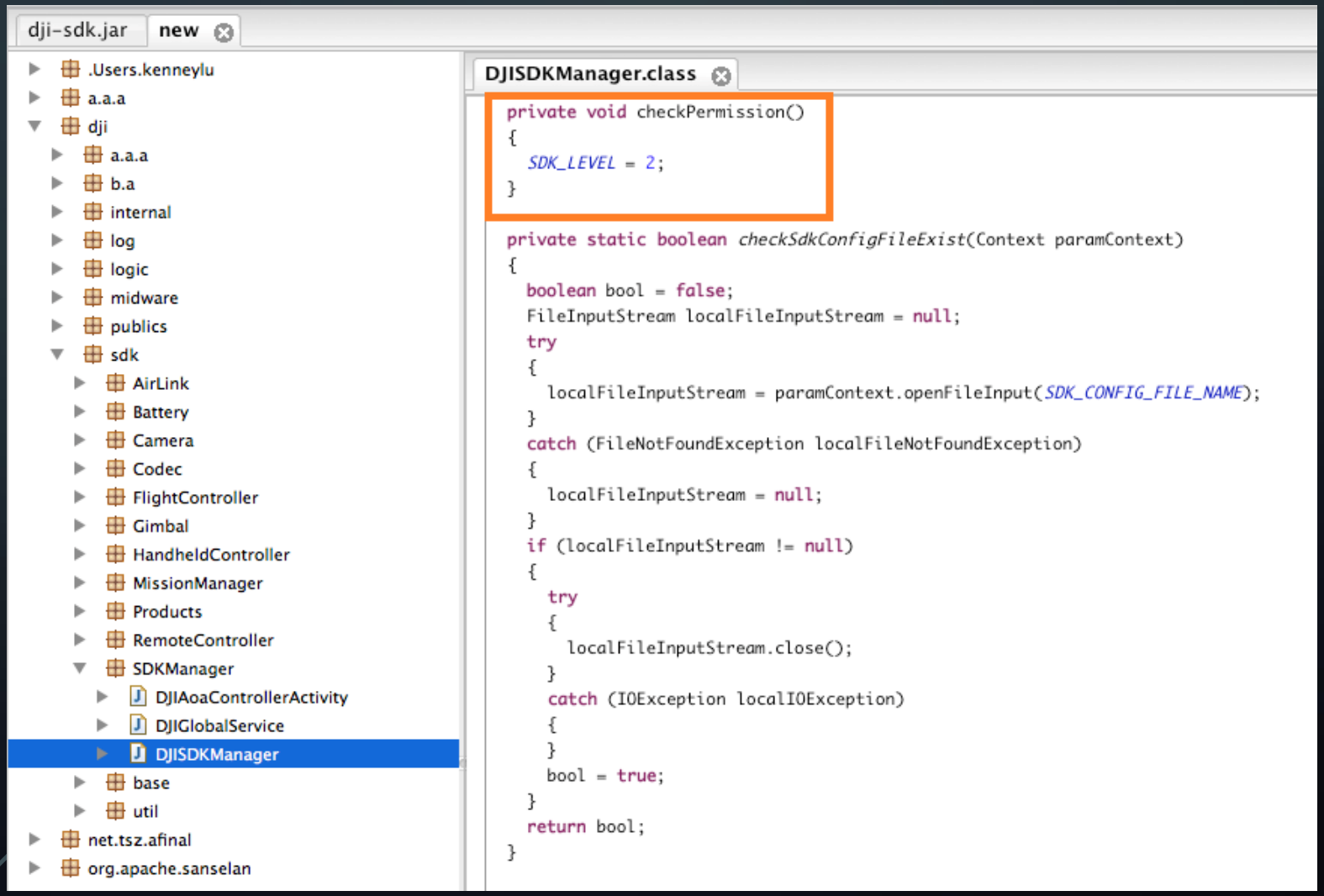
Crack the SDK Authentication Mechanism

- Use JBE - Java Bytecode Editor to patch the code



Crack the SDK Authentication Mechanism

- Check the result with JD-GUI



Connect to Drone by Cracked SDK

DEMO

Take off/Landing

DEMO

Fly to specified location

DEMO

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:

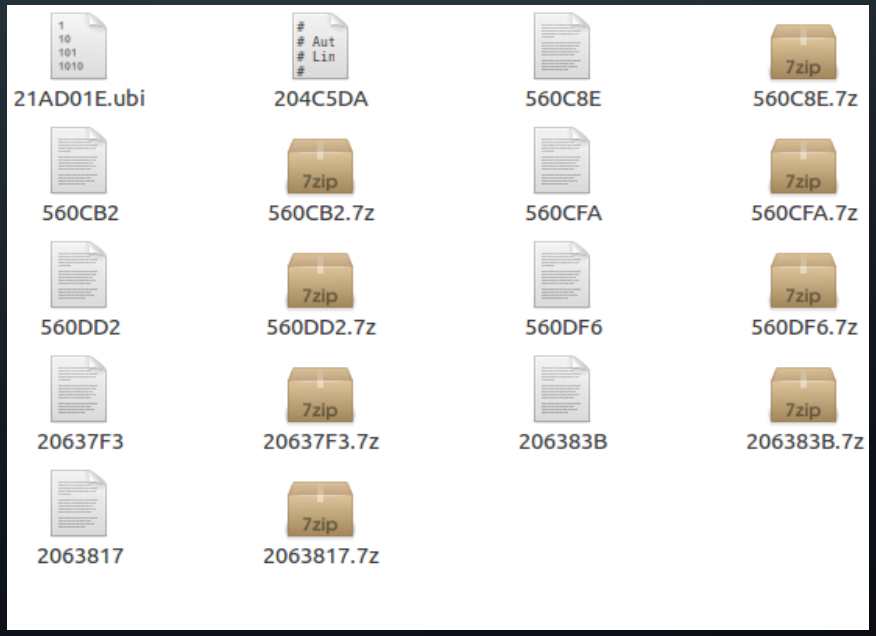
Firmware Analysis

Firmware Analysis

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

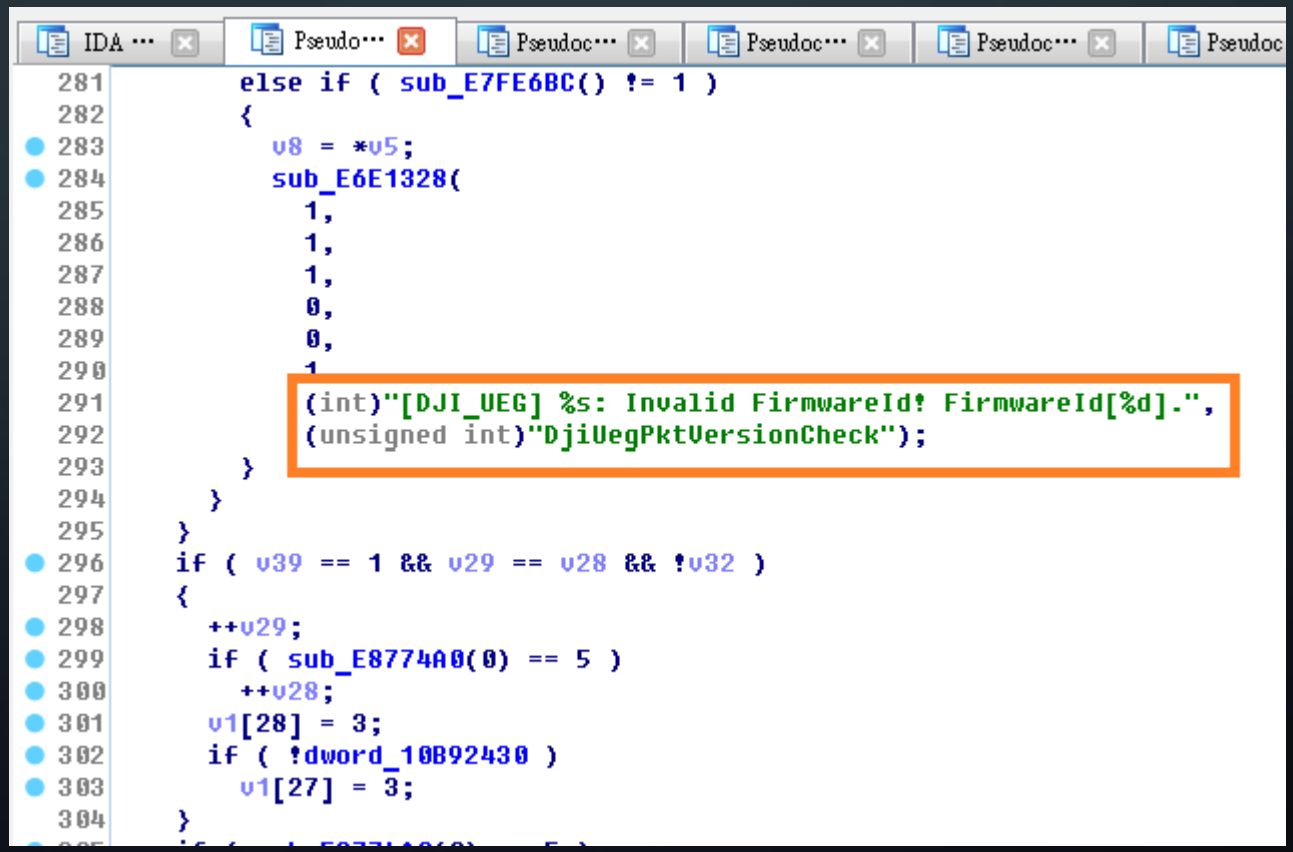
```
root@ubuntu:/home/hello# binwalk -e P3S_FW_V01.06.0040.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
5639310       0x560C8E        LZMA compressed data, properties: 0xC8, dictionary
 size: 16777216 bytes, uncompressed size: 67108864 bytes
5639346       0x560CB2        LZMA compressed data, properties: 0x64, dictionary
 size: 16777216 bytes, uncompressed size: 83886080 bytes
5639418       0x560CFA        LZMA compressed data, properties: 0xC8, dictionary
 size: 16777216 bytes, uncompressed size: 134217728 bytes
5639634       0x560DD2        LZMA compressed data, properties: 0x64, dictionary
```



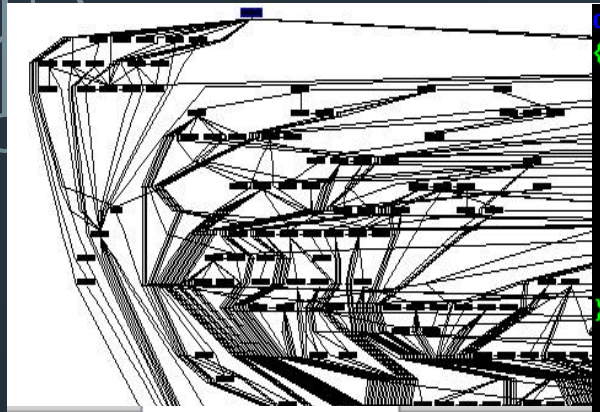
Firmware Analysis

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



Firmware Analysis

- Disassemble and writing the parser



```
if ( v1 )
{
  for ( i = 0; i < *(unsigned
  {
    v3 = v40 + 0x34 * i + 0x40
    v4 = sub_E876BD0(*(_BYTE *
    v5 = v4;
    if ( v4 )
    {
      v1[10 * *v4 + 29] = *(_B
      v1[10 * *v4 + 30] = (uns
      v1[10 * *v4 + 32] = *(_D
      sub_E6DB4B8((int)&v1[10
      sub_E6DB4B8((int)&v1[10
      v6 = sub_E876FBC(*v5);
      v7 = (unsigned __int8 *)
      if ( v6 )
      {
        sub_E6DB4B8((int)(v6 + 76), (char *)(v3 + 4), 4u);
```

```
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[0x40];
    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\
        }
    }
```

Firmware Analysis

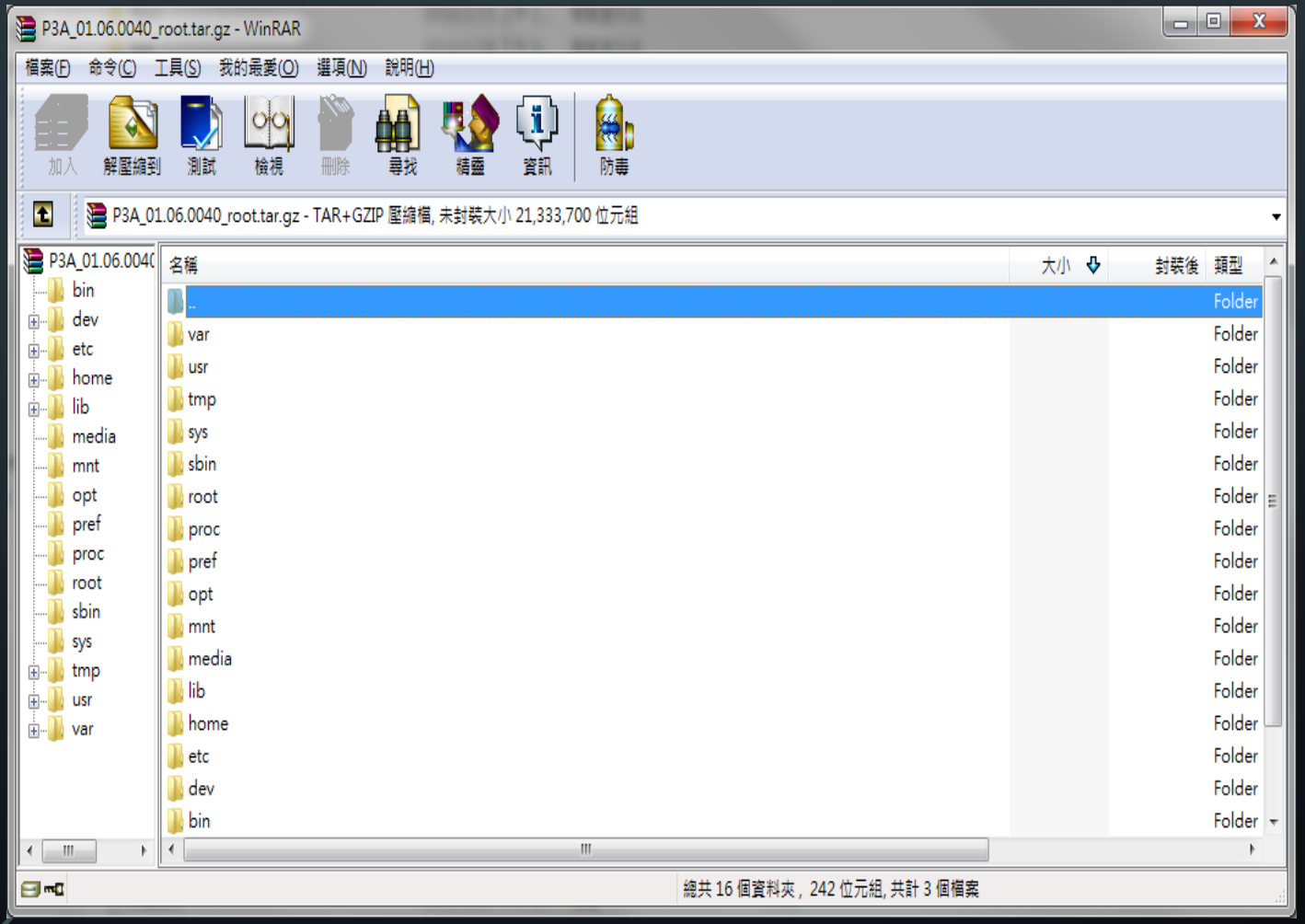
- Finally we can extract each firmware module with detailed information

```
Firmware section count: 15
Offset: 0x0000034e      Major: 03 Minor: 05     ModuleName: MCLDR       BinaryName:   PMCLDRFw3.bin Size: 43776
Offset: 0x0000ae4e      Major: 03 Minor: 06     ModuleName: MCAPP       BinaryName:   PMCAPPFw3.bin Size: 786432
Offset: 0x000cae4e      Major: 04 Minor: 00     ModuleName: GIMBAL      BinaryName:  PGIMBALFw3.bin Size: 93696
Offset: 0x000e1c4e      Major: 11 Minor: 00     ModuleName: BATTERY     BinaryName: PBATTERYFw3.bin Size: 19140
Offset: 0x000e6712      Major: 12 Minor: 00     ModuleName: ESC0        BinaryName:    PESC0Fw3.bin Size: 42496
Offset: 0x000f0d12      Major: 12 Minor: 01     ModuleName: ESC1        BinaryName:    PESC1Fw3.bin Size: 42496
Offset: 0x000fb312      Major: 12 Minor: 02     ModuleName: ESC2        BinaryName:    PESC2Fw3.bin Size: 42496
Offset: 0x00105912      Major: 12 Minor: 03     ModuleName: ESC3        BinaryName:    PESC3Fw3.bin Size: 42496
Offset: 0x0010ff12      Major: 15 Minor: 00     ModuleName: 68013       BinaryName:   P68013Fw3.bin Size: 2680
Offset: 0x0011098a      Major: 17 Minor: 00     ModuleName: MVOM4       BinaryName:   PMVOM4Fw3.bin Size: 77876
Offset: 0x001239be      Major: 17 Minor: 01     ModuleName: MVOM0       BinaryName:   PMVOM0Fw3.bin Size: 25908
Offset: 0x00129ef2      Major: 19 Minor: 00     ModuleName: FPGA        BinaryName:    PFPGAFw3.bin Size: 4194304
Offset: 0x00529ef2      Major: 01 Minor: 00     ModuleName: FC300S      BinaryName:  PFC300SFw3.bin Size: 56766764
Offset: 0x03b4d01e      Major: 01 Minor: 01     ModuleName: CAMLDR      BinaryName:  PCAMLDRFw3.bin Size: 412780
Offset: 0x03bb1c8a      Major: 09 Minor: 00     ModuleName: 1765        BinaryName:    P1765Fw3.bin Size: 81284
Press any key to continue
```

名稱	修改日期	類型	大小
P1765Fw3.bin	2016/2/1 下午 07...	BIN 檔案	80 KB
P68013Fw3.bin	2016/2/1 下午 07...	BIN 檔案	3 KB
PBATTERYFw3.bin	2016/2/1 下午 07...	BIN 檔案	19 KB
PCAMLDRFw3.bin	2016/2/1 下午 07...	BIN 檔案	404 KB
PESC0Fw3.bin	2016/2/1 下午 07...	BIN 檔案	42 KB
PESC1Fw3.bin	2016/2/1 下午 07...	BIN 檔案	42 KB
PESC2Fw3.bin	2016/2/1 下午 07...	BIN 檔案	42 KB
PESC3Fw3.bin	2016/2/1 下午 07...	BIN 檔案	42 KB
PFC300SFw3.bin	2016/2/1 下午 07...	BIN 檔案	55,437 KB
PFPGAFw3.bin	2016/2/1 下午 07...	BIN 檔案	4,096 KB
PGIMBALFw3.bin	2016/2/1 下午 07...	BIN 檔案	92 KB
PMCAPPFw3.bin	2016/2/1 下午 07...	BIN 檔案	768 KB
PMCLDRFw3.bin	2016/2/1 下午 07...	BIN 檔案	43 KB
PMVOM0Fw3.bin	2016/2/1 下午 07...	BIN 檔案	26 KB
PMVOM4Fw3.bin	2016/2/1 下午 07...	BIN 檔案	77 KB

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.)

```
#    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:%p gateway.example.com
ssh_config
2048 65537 24848546757340931501470070218597005654279520960316830997342432314
/5yV
ssh_host_key.pub
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIBmCrTkQLjtjhYolqtN8RRhscjItr9V9DOMLFV6UstHJoAoGCCqGSM49
AwEHoUQDQgAEKUf5/eVCgG6Vf2bTs/g9AVUHyefcgPe9pMW6zhY34ZSyHk86LhGg
1O3vHcF+4aIcKTYect/dY2Kdc9uaphbgZQ==
-----END EC PRIVATE KEY-----
ssh_host_ecdsa_key
# $OpenBSD: sshd_config,v 1.89 2013/02/06 00:20:42 dtucker Exp $
# 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
```

How to prevent/improve ?

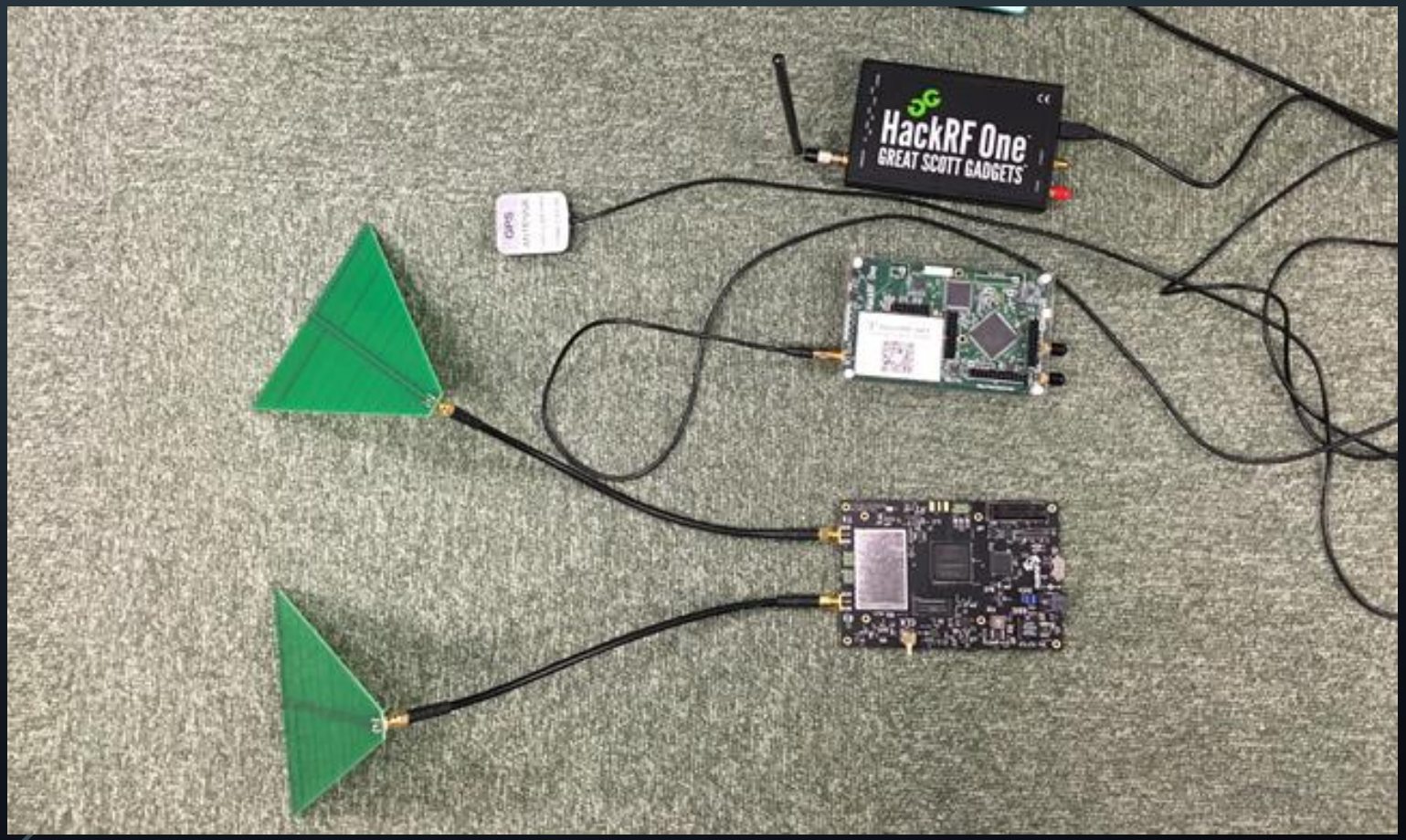
- Encrypt the firmware binary, the encryption key storage on hardware, but still need extra careful about storage place must be safety, and the side channel attack.

Next section:

Radio Signal Analysis

Radio Signal Analysis

- How to?
- Buy the SDR (software-defined radio)



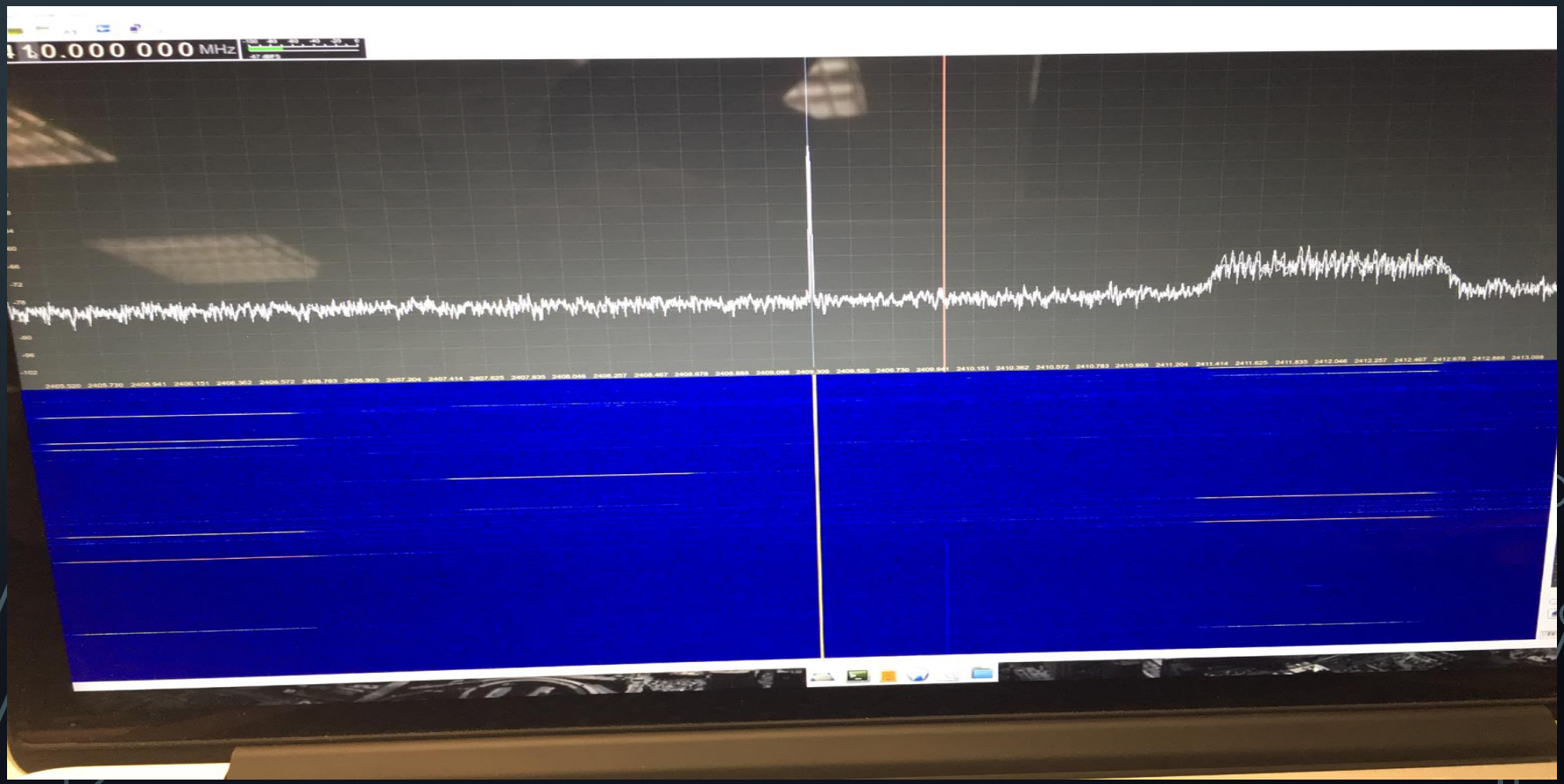
Radio Signal Analysis

P3A use two modulation/demodulation

to transfer data with 2.4GHz ISM band

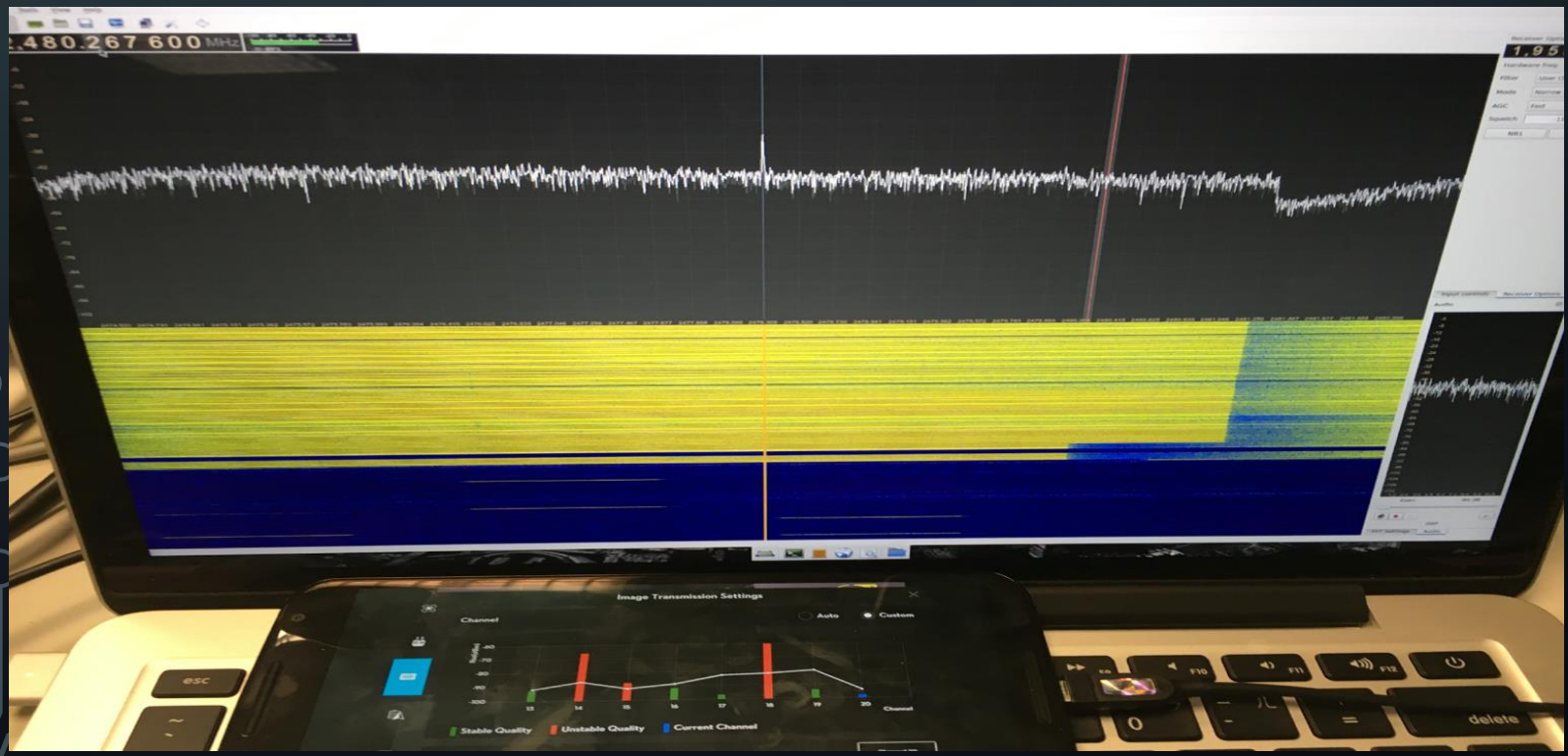
RC to Drone radio spectrum (FHSS)

- Control drone direction (up down left right)
- Frequency 2.400~2.483GHz, each channel about 1MHz



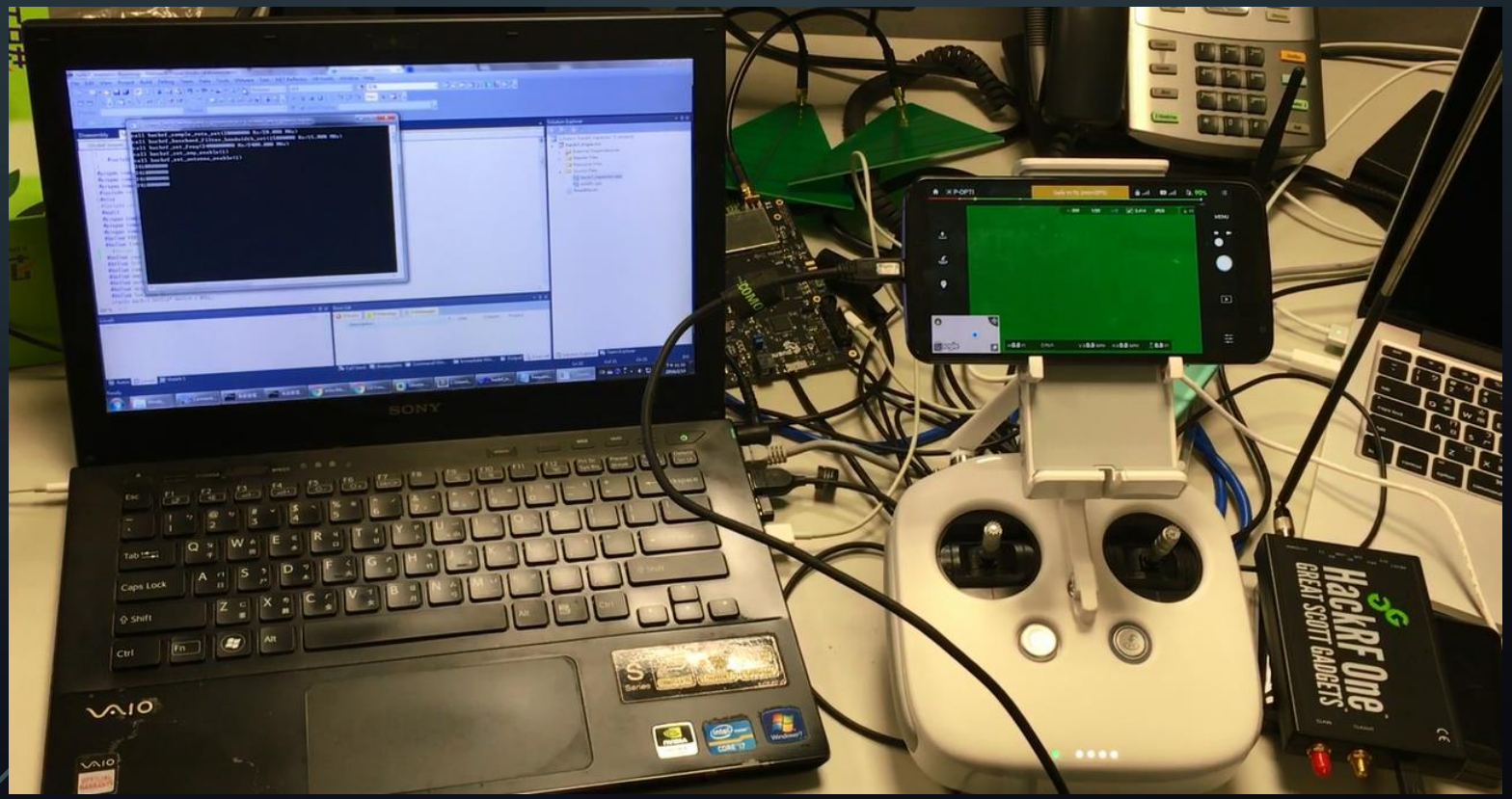
DSSS - Drone to RC radio spectrum

- 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



DEMO

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 associate 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.


P-GPS
Safe to Fly(GPS)
Using static location mode.
Start Time = 1907:468000.0
Satellite:01 Azimuth: 222.8 Elevation: 58.8
Satellite:03 Azimuth: 262.8 Elevation: 49.7
Satellite:04 Azimuth: 54.5 Elevation: 14.2
Satellite:09 Azimuth: 307.9 Elevation: 22.3
Satellite:14 Azimuth: 106.1 Elevation: 63.0
Satellite:16 Azimuth: 164.7 Elevation: 54.3
Satellite:22 Azimuth: 217.5 Elevation: 38.6
Satellite:23 Azimuth: 315.8 Elevation: 67.2
Satellite:26 Azimuth: 79.2 Elevation: 7.3
Satellite:27 Azimuth: 153.2 Elevation: 8.4
Satellite:29 Azimuth: 48.8 Elevation: 31.4
Satellite:31 Azimuth: 47.9 Elevation: 5.8
Satellite:32 Azimuth: 116.1 Elevation:
Number of channels = 12
Generating GPS signals...
Time:538.500000 Latitude:32.865293
Longitude:-96.940132

Control GPS by Joystick

DEMO

Force landing the drone by no-fly zone

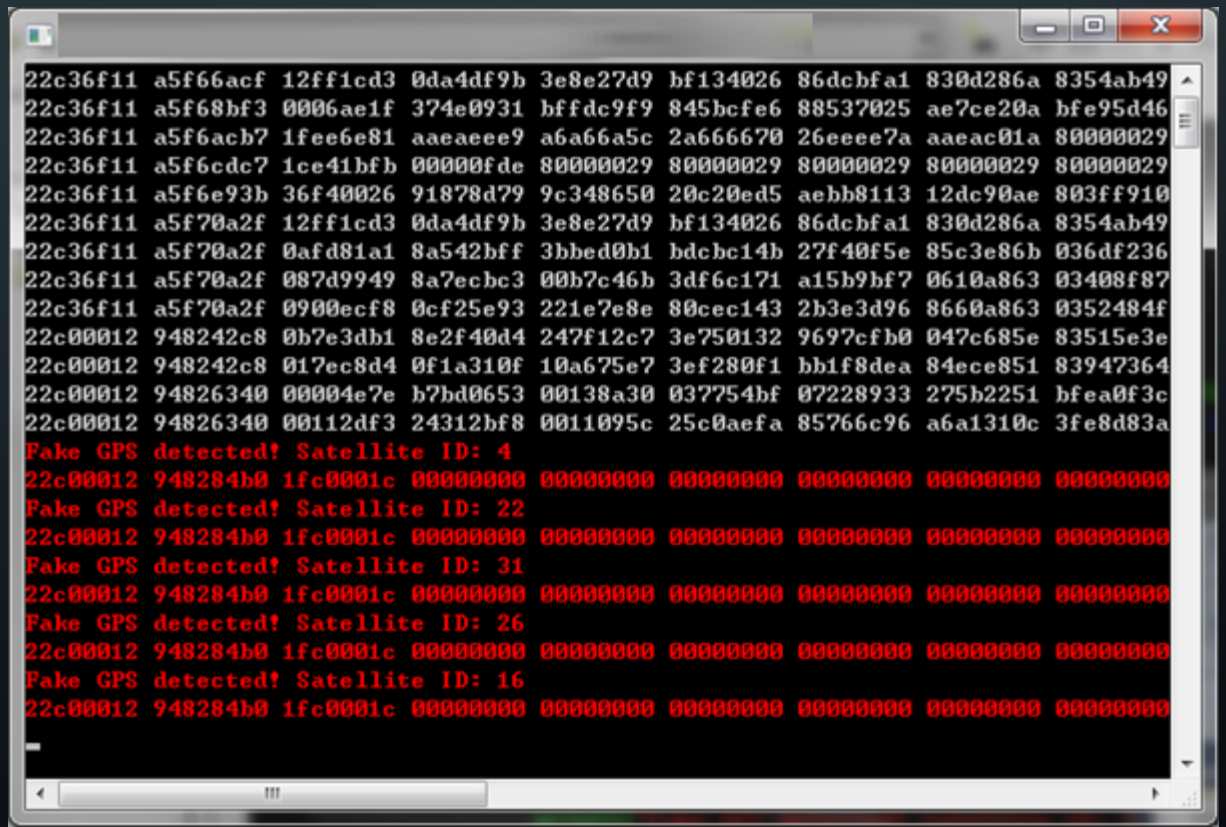
DEMO

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?

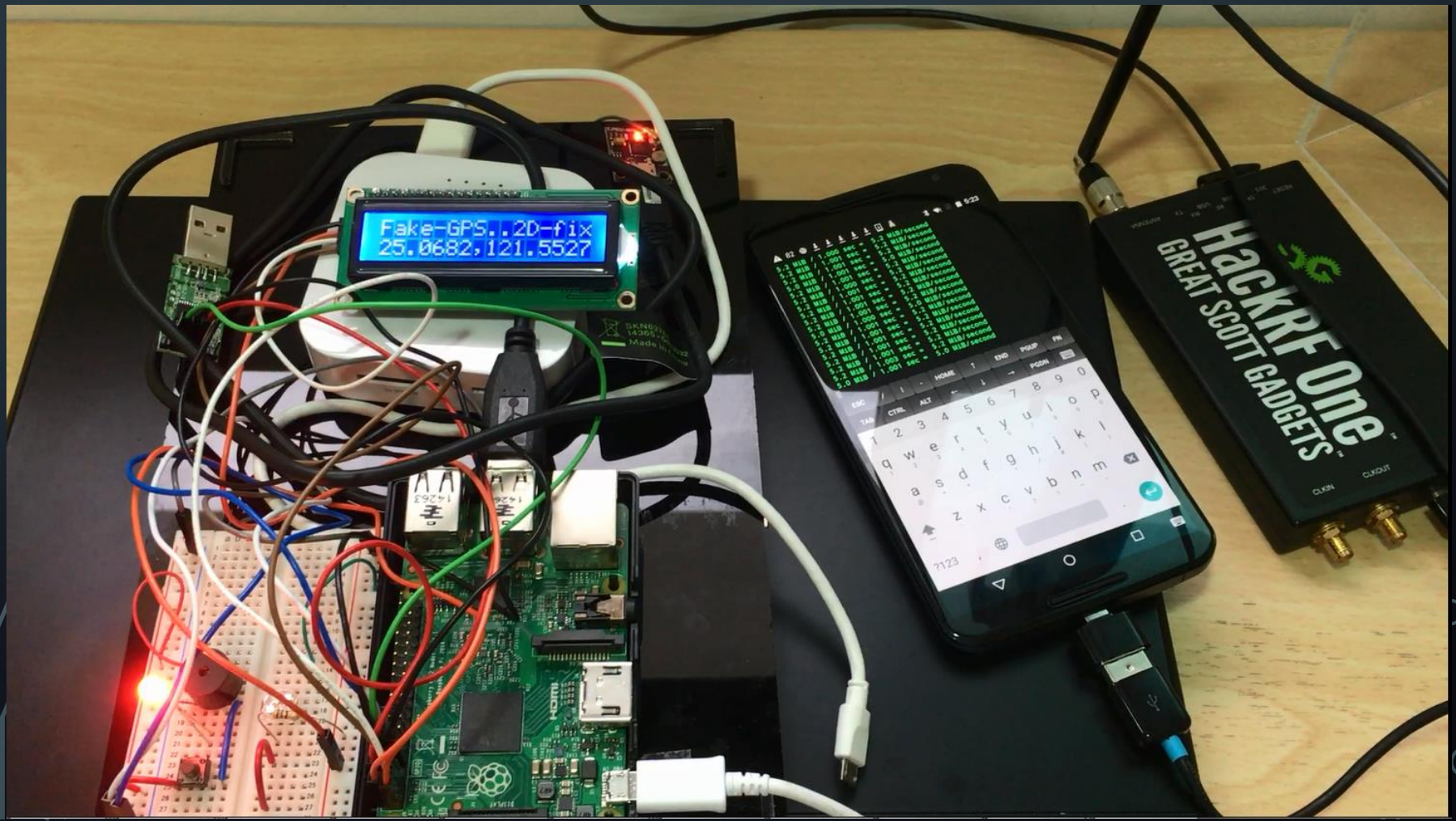
- Validate the time between satellite time and real time

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

DEMO

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