

Rapport de projet long
Attaques sur un AR Drone 2.0 Parrot

Kevyn LEDIEU, Ferréol PENNEL, Alexis PERNOT

11 mars 2019

Table des matières

1	Introduction	2
2	Matériel et Méthode	3
2.1	Matériel : AR DRONE 2.0	3
3	Hypothèse de travail	3
4	Présentation vulgarisée du travail	4
4.1	Premiers pas avec le drone	4
4.2	Prise de contrôle du drone	4
4.3	Injection de commandes	4
4.4	Virus sur le drone	4
4.5	Déni de service	4
4.6	Conclusion	5
5	Présentation scientifique du travail	6
5.1	Introduction	6
5.2	Premier pas avec le drone	6
5.3	Prise de contrôle du drone par désauthentification du client	7
5.4	Injection de commandes	9
5.5	Exploitation d'une connexion Telnet	12
5.6	Déni de service	14
6	Tutoriel	16
6.1	Initialisation de l'application	16
6.2	Menu principal	16
6.3	Prise de contrôle du drone	17
6.4	Injection de commandes	18
6.5	Dépose de virus sur le drone	19
6.6	Déni de service	20
7	Sécurisation du drone	21
7.1	Sécurisation via le Wifi	21
7.2	Renforcement du numéro de séquence	21
7.3	Connexion à distance	22
8	Conclusion	23

1 Introduction

De plus en plus présents autour de nous, les drones de loisirs représentent à la fois des opportunités technologiques et des risques de sécurité. Régulièrement, nous entendons parler d'exemples de drones ayant perturbé le trafic aérien par leur présence aux abords d'un aéroport. Ainsi un premier risque de sécurité qu'ils représentent est leur intégration dans l'espace aérien, espace que ces nouveaux aéronefs partagent avec de nombreux autres de toutes tailles. Aussi des nouvelles règles sont à l'étude afin de réglementer ces activités de loisirs. Toutefois, même une fois réglementée et contrôlée, l'activité des drones de loisir présente un second risque de sécurité lié non plus à la gestion du drone par l'opérateur mais au drone lui-même. En effet, dans la course à l'innovation dans ce domaine porteur qu'est le drone de loisir, les entreprises négligent potentiellement l'aspect sécurité matérielle et logicielle de leurs drones. Aussi, ceux-ci peuvent présenter de nombreuses vulnérabilités permettant à un attaquant extérieur de potentiellement prendre le contrôle du drone. Dans ce contexte, nous avons décidé d'étudier un drone grand public proposé par un des leaders du marché, la société *Parrot*. Nous étudierons les différentes vulnérabilités potentiellement présentes sur ce drone et mettront en oeuvre un scénario d'attaque sous forme de tutoriel.

2 Matériel et Méthode

2.1 Matériel : AR Drone 2.0

Nous allons étudier un AR DRONE 2.0. C'est un hélicoptère quadrirotor pilotable via une liaison WiFi au travers d'une application disponible sous iOS, Android, Linux ou Windows. C'est un drone civil principalement destiné au divertissement. Il est équipé de :

- un *processeur ARM Cortex A8 32 bits cadencé à 1 GHz*
- *1 Go de RAM DDR2 cadencée à 200 MHz*
- un *système d'exploitation Linux 2.6.32*
- un *module WiFi b/g/n*
- un *accéléromètre 3 axes*
- un *gyroscope 3 axes*
- un *capteur de pression*
- un *magnétomètre 3 axes*
- des *capteurs de proximité à ultrasons*
- une *caméra frontale HD (1280x720 @ 30 fps)*
- une *caméra verticale QVGA (320x240 @ 60fps)*
- un *port USB 2.0*

Il emporte une batterie lithium-polymère de 1000 mAh, 11,1 V qui alimente les 4 moteurs brushless. Il dispose d'une coque de protection en polypropylène afin de bénéficier d'une meilleure protection aux chocs. Le drone pèse 420 g avec celle-ci. Son autonomie annoncée est de 12 minutes de vol. La version 2.0 de l'AR Drone a été commercialisée pour la première fois en janvier 2012 et n'est plus disponible à la vente à l'heure actuelle.

3 Hypothèse de travail

Pour ce projet, nous supposons que le drone utilisé est un ARDrone 2.0 qui n'a pas subi de modifications logicielles et qui est donc dans un état identique à sa sortie d'usine. Il dispose ainsi uniquement des protections éventuellement prévues par le constructeur Parrot. Nous supposons dans le cadre de ce projet que nous sommes dans la position de l'attaquant et que l'ARDrone 2.0 est connecté et contrôlé par un client légitime.

4 Présentation vulgarisée du travail

4.1 Premiers pas avec le drone

Ce projet est centré sur la découverte de vulnérabilités sur l'ARDrone 2.0. Une première mise sous tension du drone nous a permis de découvrir que celui-ci propose un réseau Wifi ouvert, donc non protégé, permettant de se connecter à celui-ci et de le contrôler. Cette observation a grandement orienté notre travail. En effet, nous nous attendions à trouver un réseau Wifi sécurisé qu'il aurait été nécessaire de contourner ou de pénétrer afin d'avoir accès au drone. Cependant, le réseau ouvert nous permet en tant qu'attaquant de nous connecter directement au réseau Wifi créé par le drone. L'ARDrone 2.0 supporte la connexion de plusieurs clients simultanés à son réseau Wifi, toutefois seul le premier client à envoyer des paquets UDP de commande est maître du drone et a la possibilité de le contrôler. En supposant qu'un client légitime est connecté au drone, la question est : que peut faire l'attaquant ?

4.2 Prise de contrôle du drone

Une première idée est la prise de contrôle du drone par l'attaquant. Une fois connecté au réseau du drone, l'attaquant n'a qu'à déconnecter l'utilisateur légitime pour devenir le maître du drone. Ainsi en envoyant des messages au drone demandant la déconnexion du client légitime, l'attaquant est capable de prendre le contrôle du drone une fois celui-ci déconnecté.

4.3 Injection de commandes

Dans ce cas, le client légitime reste maître du drone. Toutefois l'attaquant se fait passer pour le client légitime et envoie des messages au drone. Il peut ainsi lui envoyer des instructions que le client légitime n'a jamais envoyées. Par exemple, l'attaquant peut envoyer au drone l'instruction d'atterrir à la place du client légitime.

4.4 Virus sur le drone

Ce troisième point exploite une faille importante du drone. En effet, celui-ci offre à tout utilisateur connecté à son réseau Wifi la possibilité d'accéder directement au système d'exploitation du drone en ayant tous les droits sur celui-ci et sans authentification ni protection. Ainsi, en tant qu'attaquant connecté au réseau Wifi du drone, nous utilisons cet accès au drone pour déposer sur celui-ci un script. Celui-ci est chargé de copier une image sur toute clé USB connectée au drone afin de démontrer le potentiel de l'attaque. Il est en effet aisé de diffuser n'importe quel virus via clé USB grâce au drone et ceci sans difficultés particulières.

4.5 Dénî de service

Cette attaque est une attaque de type "Homme du milieu". Dans cette situation, nous nous plaçons comme une passerelle entre le client et le drone, ce qui nous permet de contrôler tout le flux de communication entre les deux parties. Il est alors possible de bloquer le lien entre le drone et le client, ce qui crée donc une perte de liaison pour le client légitime.

Il est également possible de bloquer uniquement le lien qui transmet les commandes en faisant croire que le client légitime envoie des anciennes commandes.

4.6 Conclusion

Ces quatre attaques différentes permettent de démontrer les vulnérabilités majeures que présente l'ARDrone 2.0. Une des failles principales de celui-ci est son réseau Wifi non protégé. La protection de celui-ci par du WPA2 permettrait de rendre ces quatre différentes attaques beaucoup plus complexes car l'attaquant aurait dans un premier temps besoin de casser le réseau Wifi. De plus, le drone offre des accès non protégés et privilégiés qui sont une faille majeure et permettent à un attaquant une prise de contrôle complète sur le drone.

5 Présentation scientifique du travail

5.1 Introduction

L'ARDrone 2.0 est un drone grand public *Parrot* dont nous allons présenter quatre exploitations de failles de sécurité présentes au sein de celui-ci. Ces attaques seront présentées grâce à une application en **Python** appelée *Krokmou*. La première attaque sera la prise de contrôle du drone par désauthentification du client, la seconde sera l'injection de commandes sur le drone, la troisième sera l'exploitation d'une connexion Telnet sur le drone et la dernière présentera une attaque par déni de service.

5.2 Premier pas avec le drone

Dans un premier temps, après avoir allumé le drone, nous avons étudié celui-ci. La première chose que nous notons est que le drone crée un point d'accès Wifi (Access Point) afin que le client puisse se connecter à celui-ci et le contrôler au travers d'une application dédiée sur des supports Android, iOS ou PC. Toutefois, ce Wifi est un réseau ouvert et non sécurisé. Ainsi, toute personne disposant d'un matériel doté d'une carte Wifi et à portée Wifi du drone peut se connecter à celui-ci sans authentification. Toutes les communications entre le drone et le client sont donc en clair et non sécurisées. Cette faille facilite grandement l'accès au drone pour l'attaquant qui n'a ainsi pas besoin de faire face à un réseau Wifi sécurisé pour accéder au drone. Ayant découvert cet accès facilité au drone, nous nous connectons à celui-ci avec un Linux et lançons un scan avec *nmap* sur l'IP du drone 192.168.1.1 afin de connaître les ports ouverts sur le système d'exploitation du drone et découvrir des moyens d'accéder à celui-ci. Nous obtenons les résultats suivants :

```
Starting Nmap 7.60 ( https://nmap.org ) at 2019-03-05 12:23 CET
Nmap scan report for _gateway (192.168.1.1)
Host is up (0.054s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
23/tcp    open  telnet
5555/tcp  open  freeciv
Nmap done: 1 IP address (1 host up) scanned in 0.73 seconds
```

FIGURE 1 – Résultat du scan *nmap* sur le drone

Nous observons un service **ftp** permettant de partager les vidéos filmées par le drone. Nous notons également un service **telnet** disponible sur le drone. Une rapide connexion à celui-ci, nous permet de voir que nous nous connectons sans identification au drone et, plus important, nous sommes **root** sur le drone ! Nous avons donc déjà potentiellement un contrôle total de celui-ci car nous avons un accès illimité à son système d'exploitation.

Dans le même temps, nous nous documentons sur l'ARDrone 2.0. Grâce au SDK disponible sur le site de **Parrot**, nous sommes en mesure de comprendre comment forger des commandes pour les envoyer au drone ainsi que les règles de contrôle de celui-ci. Ainsi, nous savons que plusieurs clients peuvent être connectés au drone en même temps mais que c'est le premier qui envoie des paquets de commande UDP au drone qui en devient le "maître" et peut le contrôler.

Après cette découverte de l'ARDrone 2.0, nous établissons les trois scénarios d'attaques présentés en introduction que nous allons décrire par la suite.

5.3 Prise de contrôle du drone par désauthentification du client

La première attaque que nous allons présenter est une attaque permettant la prise de contrôle complète de l'ARDrone 2.0. Cette attaque se base sur la vulnérabilité principale du drone : il génère un point d'accès ouvert (sans mot de passe) et autorise 4 connections en simultané. Ainsi, l'attaquant peut se connecter au drone dès qu'il est à portée du signal Wifi et il est alors possible de déconnecter les clients du Wifi.

Détection des clients Une fois connecté au réseau, l'attaquant peut effectuer un rapide scan de celui-ci avec l'outil **Nmap**, qui permet d'obtenir la liste des appareils connectés (adresses IP et MAC).

```
ferreol@spectre: ~  
Fichier Édition Affichage Rechercher Terminal Aide  
ferreol@spectre:~$ nmap -sn 192.168.1.0/24  
  
Starting Nmap 7.60 ( https://nmap.org ) at 2019-03-05 11:01 CET  
Nmap scan report for _gateway (192.168.1.1)  
Host is up (0.0071s latency).  
Nmap scan report for 192.168.1.3  
Host is up (0.024s latency).  
Nmap scan report for spectre (192.168.1.4)  
Host is up (0.00011s latency).  
Nmap done: 256 IP addresses (3 hosts up) scanned in 2.51 seconds  
ferreol@spectre:~$ _
```

FIGURE 2 – Scan Nmap du réseau wifi

Déconnexion du Wifi Une fois la liste établie, il est alors possible de désauthentifier les différents clients du drone. Mais avant cela, regardons de plus près cette attaque :

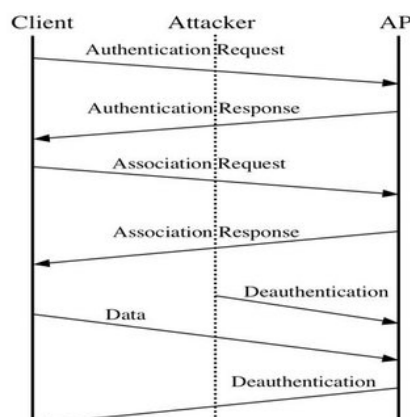


FIGURE 3 – Désauthentification d'un client wifi

Lors d'une connexion Wifi de ce type, les messages liés à la désauthentification ne sont pas chiffrés, il est alors possible pour un attaquant de forger un paquet de désauthentification avec l'adresse de la victime et de l'envoyer au drone. Le drone recevant ce paquet considère donc que la connexion avec le client est close. Pour cela, on utilise la suite d'outils bien connue sur Kali Linux, **Aircrack-ng** et plus particulièrement **Aireplay-ng** dans notre cas. **Aireplay-ng** va nous permettre de forger et d'envoyer ces paquets de désauthentification au drone. La commande est la suivante :


```
aireplay-ng -O 1 -e essid_drone -a @mac_drone -c @mac_client interface
```

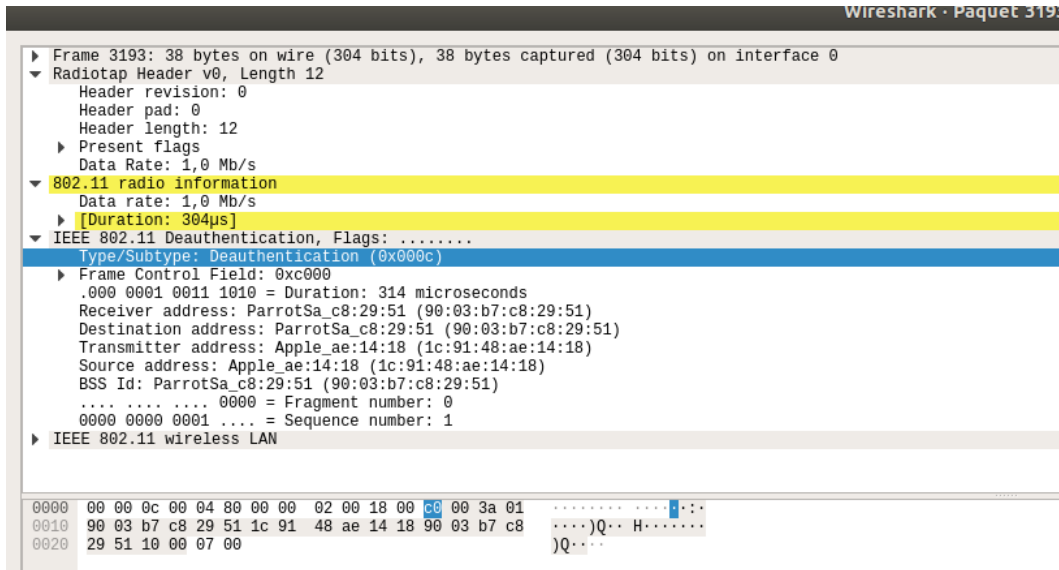
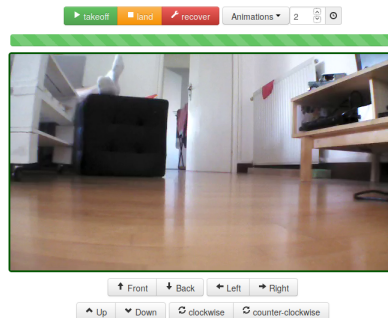


FIGURE 4 – Paquet de désauthentification

Contrôle du drone Une fois les clients désauthentifiés, nous nous reconnectons immédiatement au drone et sommes donc les premiers connectés à celui-ci et à communiquer avec lui donc "maître" du drone. Nous utilisons ensuite une application de contrôle via le navigateur web [6] - disponible à l'adresse suivante : <https://github.com/functino/drone-browser> - afin de piloter le drone avec le PC.



Battery: (Drone not connected)%
 Direction: °
 front/back: °
 left/right: °
 Altitude: m
 Velocity (x/y/z): / /

FIGURE 5 – Server **Nodejs** pour piloter le drone

Cas d'une seule connexion Si on se place dans le cas de figure où une seule connexion Wifi est autorisée par le drone, cette attaque reste faisable, toujours grâce à la suite d'outils

Aircrack-ng. En effet, l'outil **Airodump-ng** permet de détecter les réseaux Wifi aux alentours ainsi que les clients connectés à un réseau sans être nous-même connectés à celui-ci, via la commande suivante :

```
airodump-ng -t -OPN interface
```

```

root@spectre: /home/ferreol/Bureau
Fichier  Édition  Affichage  Rechercher  Terminal  Aide

CH  8 ][ Elapsed: 6 s ][ 2019-03-05 16:04

BSSID                PWR  Beacons    #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
90:03:B7:C8:29:51    -48      16         19   0   6  54e. OPN             ardrone2_022491
68:A3:78:0A:19:AC    -84       6          0   0   5  54e. OPN             FreeWifi
62:95:04:FE:F6:B5    -88       4          0   0  11  54e. OPN             SFR WiFi FON
72:5D:51:CD:46:87    -88       4          0   0   6  54   OPN             SFR WiFi FON
14:0C:76:A4:9D:2B    -90       3          0   0  11  54e. OPN             FreeWifi
C2:1B:29:B3:43:C8    -90       5          0   0  11  54e. OPN             orange

BSSID                STATION            PWR  Rate  Lost  Frames  Probe
(not associated)    68:14:01:7A:8B:F5  -91   0 - 1    0      2
90:03:B7:C8:29:51  8C:F5:A3:1B:54:97  -53  0e-24  162    43

```

FIGURE 6 – Airodump-ng sur le drone

Il suffit ensuite d'envoyer les paquets de désauthentification comme précédemment et nous pouvons alors nous connecter au drone en tant que seul client.

5.4 Injection de commandes

Cette seconde attaque a pour objectif d'injecter des commandes au drone sans en prendre le contrôle complètement. Le but est de faire exécuter certaines commandes tout en laissant le contrôle à l'utilisateur légitime.

Liaison de commande

La liaison de commande se fait entre le client et l'Access Point via le port **5556** en source et destination. Le client ne fait qu'envoyer régulièrement des paquets UDP pour maintenir et/ou commander le drone. Il n'y a pas de retour de la part du drone via cette liaison. Les informations à propos du drone sont envoyées par le drone vers le client via des paquets UDP sur le port **5554** en source et en destination. On va donc uniquement utiliser la liaison de commande pour injecter des commandes.

Format des paquets UDP

Pour créer des paquets de commandes légitimes, il faut étudier la forme de ceux-ci. Les paquets UDP transportent une chaîne de caractère qui sera interprétée par le drone.

Il faut que la chaîne de caractère commence par **AT*** puis vient le type de commandes :

- **REF** pour le décollage, atterrissage et le mode d'urgence
- **PCMD** pour déplacer le drone
- **PCMD_MAG** pour déplacer le drone avec Absolute Control support
- **FTRIM** pour régler la référence au plan horizontal (le drone doit être au sol)
- **CONFIG** pour la configuration de l'AR Drone 2.0
- **CONFIG_IDS** identifiants pour les AT*CONFIG commandes
- **COMWDG** pour réinitialiser la communication
- **CALIB** pour demander au drone de recalibrer le magnétomètre (le drone doit être en vol)

Dans notre cas, nous allons uniquement injecter des commandes **REF** et **PCMD**. Voyons le format de ces deux commandes.

Pour toutes les commandes, il faut que la chaîne de caractère soit suivi d'un = et d'un numéro de séquence. Le client incrémente ce numéro de séquence à chaque envoi de commande. Le drone valide uniquement les commandes donc le numéro de séquence est supérieur au précédent. Il faut donc prendre un numéro de séquence qui est au moins supérieur à celui en cours. Pour cela on prendra **10000000000** en numéro de séquence, ce qui nous assure de façon quasi-sûre d'être supérieur à celui en cours. On incrémentera également celui-ci à chaque envoi de commande injectée.

Pour **REF**, on va chercher à créer des ordres de décollage et d'atterrissage. Pour cela le numéro de séquence doit être suivi d'un argument qui sera égal à **290718208** pour le décollage et **290717696** pour l'atterrissage. Cela correspond à la modification du bit 9 de l'entier codé sur 32 bits.

On a donc **AT*REF=10000000000,290718208<CR>** pour le décollage et **AT*REF=10000000000,290717696<CR>** pour l'atterrissage. On notera que les arguments doivent être séparés par une virgule et que la chaîne de caractère doit se terminer par un retour chariot noté **<CR>**.

Pour **PCMD**, il y a 5 arguments à la suite du numéro de séquence. Le premier informe si on utilise des commandes progressives et/ou le Combined Yaw mode, le second contrôle le déplacement gauche/droite, le troisième le déplacement avant/arrière, le quatrième le déplacement vertical et le cinquième la rotation selon l'axe vertical.

Pour le premier argument, on prend **1** ce qui correspond à l'activation du Combined Yaw mode.

Les autres arguments correspondent à une valeur entre -1 et 1. On demande donc au drone un pourcentage des limites de déplacement et de rotation défini dans la configuration du drone. On choisira une valeur de +/-0.3 pour avoir des commandes non trop sensibles. Cela correspond à passer l'argument +/-1050253722 qui est la représentation de 0.3 en nombre flottant à précision simple sur 32 bits, cette représentation est ensuite transformée en entier représenté sur 32 bits.

On a donc les commandes suivantes :

- Gauche **AT*PCMD=10000000000,1,-1050253722,0,0,0**
- Droite **AT*PCMD=10000000000,1,1050253722,0,0,0**
- Avancer **AT*PCMD=10000000000,1,0,-1050253722,0,0**

- Reculer **AT*PCMD=10000000000,1,0,1050253722,0,0**
- Descendre **AT*PCMD=10000000000,1,0,0,-1050253722,0**
- Monter **AT*PCMD=10000000000,1,0,0,1050253722,0**
- Tourner à gauche **AT*PCMD=10000000000,1,0,0,0,-1050253722**
- Tourner à droite **AT*PCMD=10000000000,1,0,0,0,1050253722**

Se faire passer pour le client légitime

Pour connaître les différents clients connectés au drone, on scanne le réseau WiFi à l'aide de l'outil **nmap**. On obtient alors le couple IP/MAC de chaque client. Pour être sûr de se faire passer pour le client légitime, on va envoyer notre paquet UDP en se faisant passer pour chaque client. Pour cela, on utilise **scapy** qui permet d'envoyer nos paquets UDP avec l'adresse MAC et IP d'un des clients.

Un contrôleur pour injecter des commandes

Pour simplifier l'injection de commande, on a créé un contrôleur simpliste qui permet d'injecter les commandes vues précédemment.

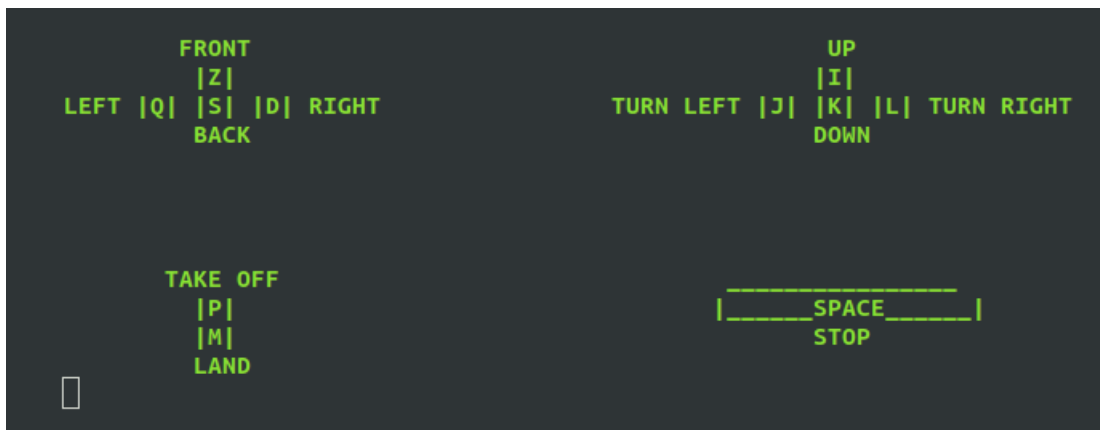


FIGURE 7 – Contrôleur pour l'injection de commandes

Pour chaque commande, on envoie 3 paquets UDP avec un intervalle de 0.3 seconde. Cela est suffisant pour que le drone interprète notre commande.

Conditions du maintien du contrôle par l'utilisateur légitime

L'objectif de l'injection est de faire effectuer des commandes au drone tout en laissant le contrôle au client légitime. Or quand on envoie une commande avec un numéro de séquence 10000000000 ou plus, le numéro de séquence en cours du drone s'actualise à celui-ci. Le client légitime perd donc le contrôle du drone car son numéro de séquence est trop bas. On va donc réinitialiser le numéro de séquence du drone à chaque fin d'injection de commande. Pour cela, il suffit d'envoyer une commande avec le numéro de séquence 1. On utilise alors la commande REF pour réinitialiser le numéro de séquence. On envoie 3 commandes REF à 0.3 seconde d'intervalle, décollage quand le drone est en vol et atterrissage quand le drone est au sol. Cela permet de ne pas avoir d'effet sur le drone autre que celui de réinitialiser le numéro de séquence.

Pour savoir l'état du drone, on considère que celui-ci est en vol au lancement de l'injection de commandes. En effet, les commandes de déplacement sont interprétées uniquement en

vol. La seule commande valable au sol est la demande de décollage. On modifiera cet état en fonction des ordres d'atterrissage et de décollage des injections de commandes. On notera que si le drone est dans un état autre que celui de l'utilisateur légitime alors ce dernier ne peut pas récupérer le contrôle du drone.

5.5 Exploitation d'une connexion Telnet

La troisième attaque exploite une vulnérabilité du drone. Un service **Telnet** est ouvert et permet d'accéder à un shell **root** sur le drone.

```
Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.

BusyBox v1.14.0 ( ) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# id
uid=0(root) gid=0(root)
# █
```

FIGURE 8 – Accès **Telnet root** sur l'ARDrone 2.0

Ainsi une fois connecté au réseau du drone, n'importe qui peut utiliser cette connexion **Telnet** pour être administrateur sur le drone. Une fois le shell **root** obtenu, on se retrouve avec un accès au système d'exploitation du drone, qui est un **Linux**, et il est possible de faire ce que l'on veut. L'attaquant peut alors réaliser une large variété d'attaques directement sur le système d'exploitation. Il peut interagir avec les processus qui tournent sur celui-ci et notamment le processus qui pilote le drone **program.elf**.

```

Krok mou : telnet — Konsole
Fichier  Édition  Affichage  Signets  Configuration  Aide
Mem: 95760K used, 22432K free, 0K shrd, 0K buff, 5484K cached
Load average: 0.00 0.00 0.00 2/90 8603
PID  PPID  USER  STAT  VSZ  %MEM  COMMAND
2535  823  root   S      83912 71%  /bin/program.elf -360p.slices -emergency.restart <===
2643  1  root   S      2832 2%   telnetd -l /bin/sh
822  1  root   S      2824 2%   inetd
8597  8596  root   R      2824 2%   top
786  1  root   S      2744 2%   /bin/sh /bin/memory_check.sh
823  1  root   S      2744 2%   /bin/sh /bin/program.elf.respawner.sh -360p.slices 0 -live.tcp
2646  1  root   S      2740 2%   udhcpd /tmp/udhcpd.conf
8349  2643  root   S      2740 2%   /bin/sh
8596  2643  root   S      2740 2%   /bin/sh
1  0  root   S      2736 2%   init
3269  1  root   S      2736 2%   init
3270  1  root   S      2736 2%   init
3334  1  root   S      2736 2%   /sbin/klogd -n
5616  1  root   S      2736 2%   /sbin/syslogd -n -m 0
8603  786  root   S      2604 2%   sleep 10
586  1  root   S <    1788 2%   udevd --daemon
597  586  root   S <    1788 2%   udevd --daemon
599  586  root   S <    1788 2%   udevd --daemon
792  1  root   S      1676 1%   /bin/bashproxy /tmp/.bashproxyfifo.in /tmp/.bashproxyfifo.out
785  1  root   S      1672 1%   /bin/factory_reset_cb
2661  1  root   S      1540 1%   /bin/parrotauthdaemon
2  0  root   SW     0 0%   [kthreadd]
3  2  root   SW     0 0%   [ksoftirqd/0]
4  2  root   SW     0 0%   [watchdog/0]
5  2  root   SW     0 0%   [events/0]

```

FIGURE 9 – Processus s'exécutant sur l'ARDrone 2.0

Il est ainsi possible de modifier n'importe quel fichier du système d'exploitation et de réaliser par exemple un Déni de Service sur le drone depuis "l'intérieur" de celui-ci. Étant **root** sur le drone sans nécessité d'escalade de privilèges, nous cherchons ce qui serait le plus intéressant de faire avec ce contrôle du drone. Nous décidons de démontrer la possibilité d'implanter un virus sur le drone. Ainsi nous avons développé un script en bash qui sera envoyé sur le drone par l'application *Krok mou* accompagné d'un fichier au choix de l'attaquant. L'envoi se fait via la connexion **FTP** au drone. Une fois les deux fichiers sur le drone, l'application utilise la connexion **Telnet** afin d'exécuter le script. Celui-ci se copie alors au sein du dossier **/bin** du drone et s'ajoute à la liste des processus à lancer au démarrage du drone afin qu'il s'exécute en permanence sur le drone. Le script va alors régulièrement essayer de copier le fichier envoyé par l'attaquant avec le script sur une clé USB qui serait connectée au drone. L'utilisateur légitime peut utiliser une clé USB pour récupérer des vidéos filmées par le drone et enregistrées sur celle-ci. Par défaut et à des fins de démonstration, le script dépose sur les clés USBs connectées une image.

```
> Krokrou : telnet - Konsole
Fichier  Édition  Affichage  Signets  Configuration  Aide
# ls
bin      dev      factory  home     licenses  proc     /sbin     tmp      usr
data     etc      firmware lib       mnt       root      sys      update   var
# cd bin
# ls
US00_check      egrep      mktemp      sed
ash             factory_reset_cb  mount        sh
bashproxy       false      mount_usb.sh sleep
board_check     fgrep      mv           stat
busybox         fsck_msdos netstat      stty
cat             gdbserver  nfs.sh       sync
check_update.sh getopt      pairing_setup.sh tar
checkplf        grep       parallel-stream.sh touch
chgrp           gunzip     parrotauthdaemon true
chmod           gzip       pidof        umount
chown           hostname   ping         umount_usb.sh
cp             init_gpios.sh  program.elf  uname
ctyhack        ip         program.elf.respawner.sh updateEphemeris.sh
date           ipcalc     ps           usleep
dd            kill       pwd          vi
devmem2       kk         random_ip    watch
df            krok_krokrou.png <=== random_mac  wifi_setup.sh
dmesg         ln         repairBoxes  wpa_cli
dragon.sh <=== ls      repairMicronesie.sh wpa_passphrase
dsp           memory_check.sh reset_config.sh wpa_supplicant
dspbridge     mkdir      rm           zcat
echo          mknod     rmdir
#
```

FIGURE 10 – Virus dans le dossier `/bin` du drone

Cette attaque démontre que le drone peut facilement servir de vecteur d'attaque pour diffuser un virus par clé USB. La clé permettant de récupérer des vidéos filmées par le drone, elle sera ensuite très probablement connectée à un ordinateur appartenant au propriétaire du drone permettant ainsi la diffusion du virus. Le drone devient ainsi un vecteur de diffusion facile à exploiter et discret mais qui nécessite tout de même une proximité physique avec le drone afin de pouvoir se connecter au Wifi de celui-ci.

La connexion Telnet peut être exploitée de multiples manières y compris afin de réaliser des attaques plus destructrices sur le drone en supprimant des fichiers majeurs du système d'exploitation.

5.6 Déni de service

La dernière attaque consiste à réaliser un déni de service complet ou uniquement sur la liaison de commande.

Déni de service complet

Pour réaliser un déni de service complet, on va réaliser une attaque du type homme du milieu et bloquer tous les paquets entre le drone et le client.

Pour cela, on va utiliser de nouveau *scapy* pour faire de l'ARP spoofing.

On effectue tout d'abord un scan des différents clients. On réalisera alors l'attaque pour chaque client, on ne cherche pas à connaître le client légitime.

On envoie ensuite régulièrement des paquets ARP *isAt* (toutes les 0.5 seconde) en associant l'adresse IP du ou des clients avec l'adresse MAC de l'attaquant. On fait également de même avec l'IP du drone.

61	1.968038992	HonHaiPr_d8:77:53	Apple_ae:14:18	ARP	42	192.168.1.1	is at	d8:5d:e2:d8:77:53	(duplicate use of 192.168.1.3 detected!)
62	2.389322307	HonHaiPr_d8:77:53	ParrotSa_c8:29:51	ARP	42	192.168.1.3	is at	d8:5d:e2:d8:77:53	
63	2.453648273	HonHaiPr_d8:77:53	Apple_ae:14:18	ARP	42	192.168.1.1	is at	d8:5d:e2:d8:77:53	(duplicate use of 192.168.1.3 detected!)
64	2.582133410	HonHaiPr_d8:77:53	Apple_ae:14:18	ARP	42	192.168.1.1	is at	d8:5d:e2:d8:77:53	(duplicate use of 192.168.1.3 detected!)
65	3.022093828	HonHaiPr_d8:77:53	ParrotSa_c8:29:51	ARP	42	192.168.1.3	is at	d8:5d:e2:d8:77:53	
66	3.072792210	HonHaiPr_d8:77:53	Apple_ae:14:18	ARP	42	192.168.1.1	is at	d8:5d:e2:d8:77:53	(duplicate use of 192.168.1.3 detected!)
67	3.196234275	HonHaiPr_d8:77:53	Apple_ae:14:18	ARP	42	192.168.1.1	is at	d8:5d:e2:d8:77:53	(duplicate use of 192.168.1.3 detected!)

FIGURE 11 – Capture des paquets ARP

Cela a pour conséquence que tous les paquets sont envoyés à l'attaquant. Il suffit alors à l'attaquant de ne pas retransmettre ces paquets.

Quand l'attaquant stoppe son attaque, les caches ARP vont revenir à leurs états normaux après un court délai.

Déni de service sur la liaison de commande

Pour réaliser un déni de service uniquement sur la liaison de commande, on va utiliser le principe de l'injection de commande vue précédemment. On va donc envoyer des paquets UDP de commandes en se faisant passer pour le client légitime.

L'attaquant envoie régulièrement (toutes les 0.1 secondes) la commande de décollage ou d'atterrissage avec un numéro de séquence très élevé, ici **10000000000** ce qui va engendrer le fait que les paquets légitimes ne sont plus interprétés par le drone.

Pour redonner le contrôle du drone, une fois l'attaque arrêtée par l'attaquant. On envoie à plusieurs reprises (20 fois toutes les 0.1 secondes) la commande de décollage ou d'atterrissage, suivant l'état du drone, avec pour numéro de séquence **1**. Cela a pour conséquence de réinitialiser le numéro de séquence comme cela a été vu avec l'injection de commande.

6 Tutoriel

Le tutoriel du projet se présente sous la forme d’une application **Linux** développée en **Python** permettant d’exploiter les quatre types d’attaques présentées précédemment. Disponible à l’adresse suivante : <https://github.com/ferreolpennel/Krokmou>, cette application de démonstration est utilisable par tout utilisateur satisfaisant les pré-requis à son installation. L’application, appelée KROKMOU, est dédiée à l’ARDrone 2.0 et ne permet des attaques que contre ce type de drone et ce à des fins de démonstration uniquement. Elle permet ainsi de prendre le contrôle d’un drone à la place d’un utilisateur légitime déjà connecté au drone, d’envoyer des commandes pirates au drone sans déconnecter l’utilisateur légitime de celui-ci et de déposer un virus de démonstration sur le drone. Elle illustre ainsi les attaques présentées précédemment et met en relief les failles correspondantes sur ce type de drone.

6.1 Initialisation de l’application

L’application permet de sélectionner l’interface Wifi à utiliser pour se connecter au drone. Elle réalise ensuite un scan des réseaux Wifi alentours et affiche ensuite uniquement les réseaux Wifi de drones **Parrot**. Il suffit à l’attaquant de sélectionner le drone auquel il veut se connecter puis l’application configure l’interface Wifi pour se connecter à celui-ci.



```
Krokmou : sudo — Konsole
Fichier  Édition  Affichage  Signets  Configuration  Aide

KROKMOU

Take control of any ARDrone2.0 nearby and make him do funny things !

Choose your Wifi interface:
0 : lo
1 : wlp58s0
2 : vmnet1
3 : vmnet8
Krokmou > 1

Scanning for vulnerable UAVs...

Choose your UAV:
0 : ardrone2_022491
Krokmou > 0
```

FIGURE 12 – Initialisation de l’application

6.2 Menu principal

Le menu principal de l’application permet de sélectionner une des quatre attaques afin de la réaliser sur le drone auquel l’attaquant s’est connecté précédemment.



FIGURE 13 – Menu principal de l'application

6.3 Prise de contrôle du drone

Cette option du menu permet à l'attaquant de prendre le contrôle du drone à la place de l'utilisateur légitime grâce à l'attaque décrite précédemment dans ce rapport. Le contrôle du drone se réalise au travers du navigateur Web et d'un serveur **Node.js** issu du dépôt Github suivant : (<https://github.com/functino/drone-browser>).[6]

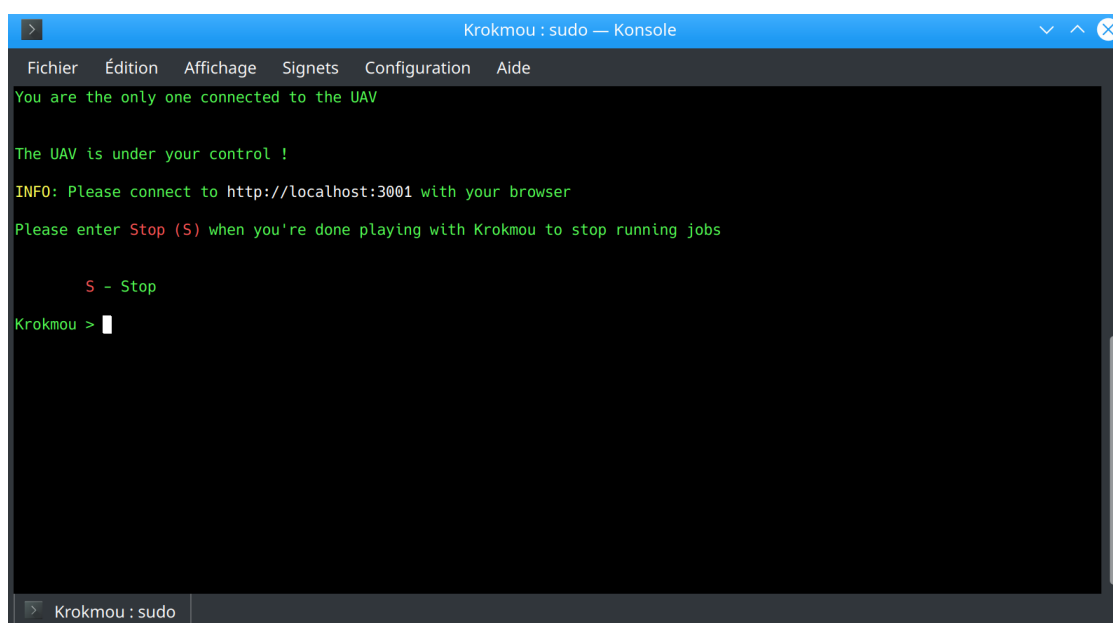


FIGURE 14 – Prise de contrôle du drone par l'application

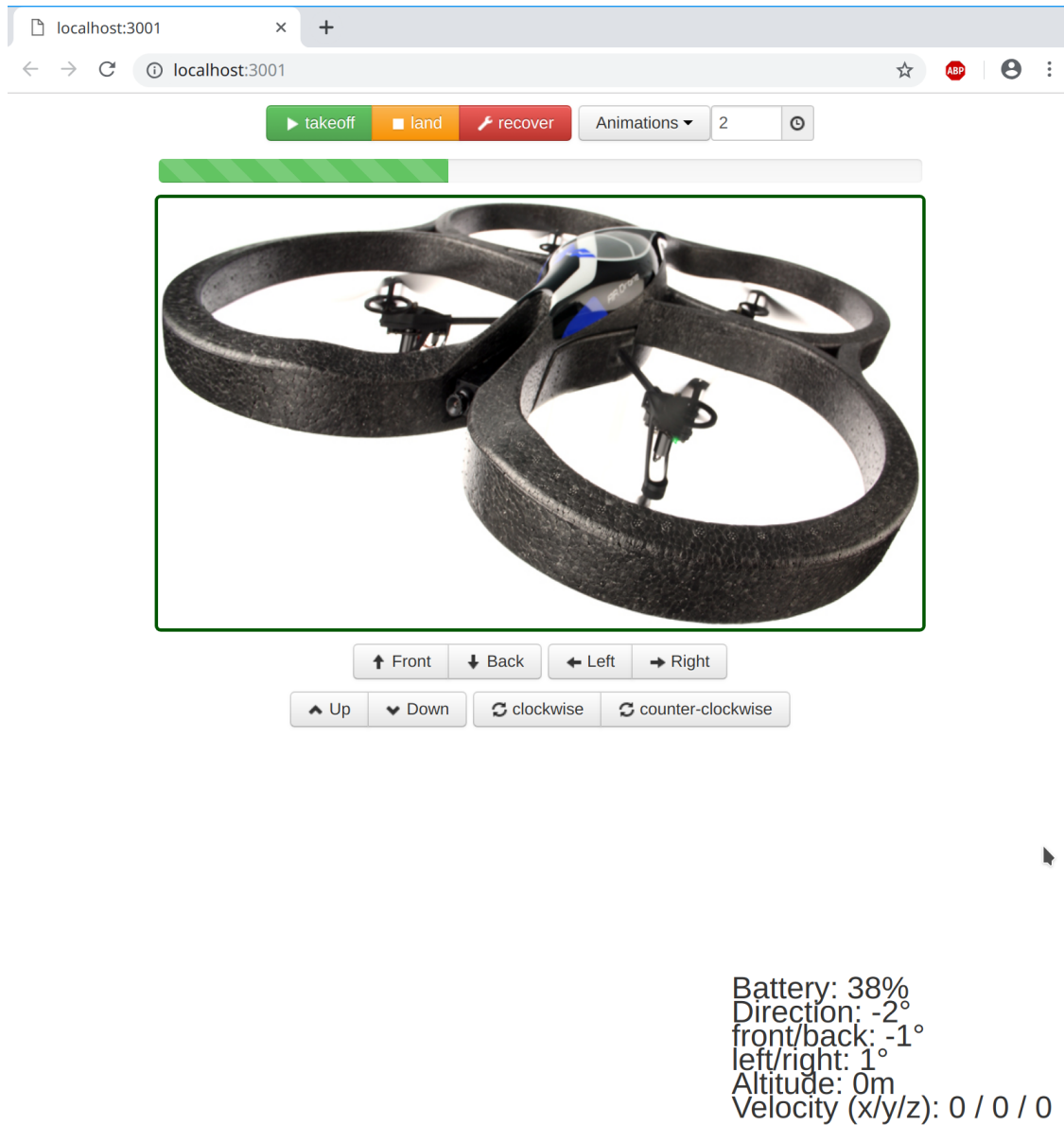


FIGURE 15 – Interface de contrôle web

6.4 Injection de commandes

Ce sous-menu permet à l'attaquant d'envoyer des commandes au drone en laissant le contrôle du drone au client légitime. Les différentes injections de commandes possibles sont présentées dans l'affichage du contrôleur pour l'injection de commandes.



FIGURE 16 – Sous-menu d’injection de commandes

Il est donc possible de faire avancer/reculer, d’aller à gauche/droite, de monter/descendre, de tourner à gauche/droite selon l’axe vertical et de décoller/atterrir. L’utilisation du contrôleur se fait donc à l’aide du clavier.

Le client légitime garde le contrôle du drone à la seule condition que le drone soit dans le même état (en vol/au sol) que celui du client légitime avant l’injection de commandes.

6.5 Dépose de virus sur le drone

Cette option permet d’exploiter une vulnérabilité laissant à l’attaquant un contrôle total du drone. Dans le cadre de la démonstration, l’application dépose un virus et un fichier sélectionné par l’utilisateur sur le drone. Ce fichier sera copié sur toute clé USB qui sera connectée au drone. Celle-ci sera par conséquent considérée comme infectée. Par défaut, l’application dépose le virus et une image sur le drone. C’est cette image qui sera copiée sur toute clé USB connectée au drone toujours à des fins de démonstration. Cependant, l’attaquant peut indiquer le chemin d’un fichier de son choix au moment où l’application lui propose afin de remplacer cette image.

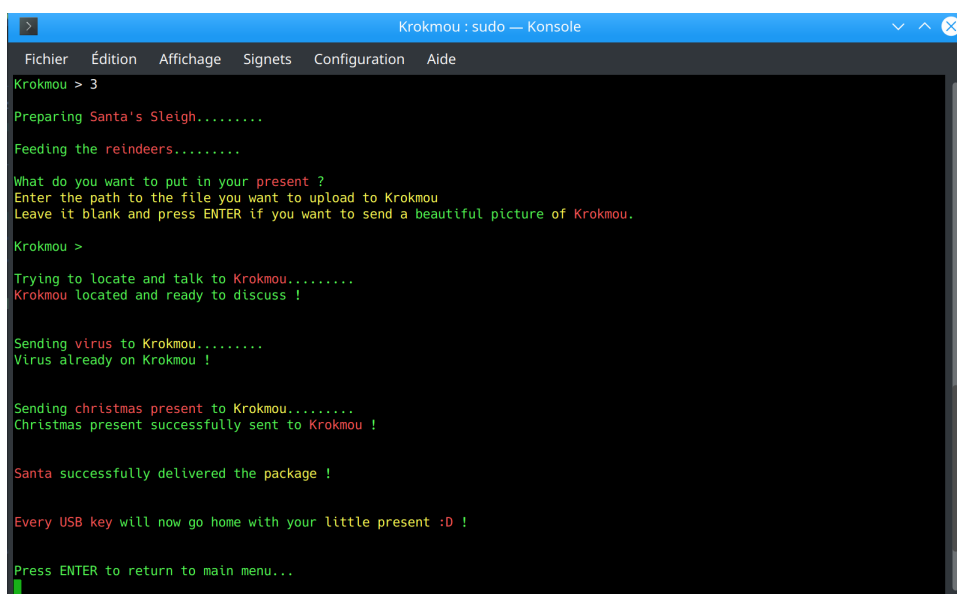


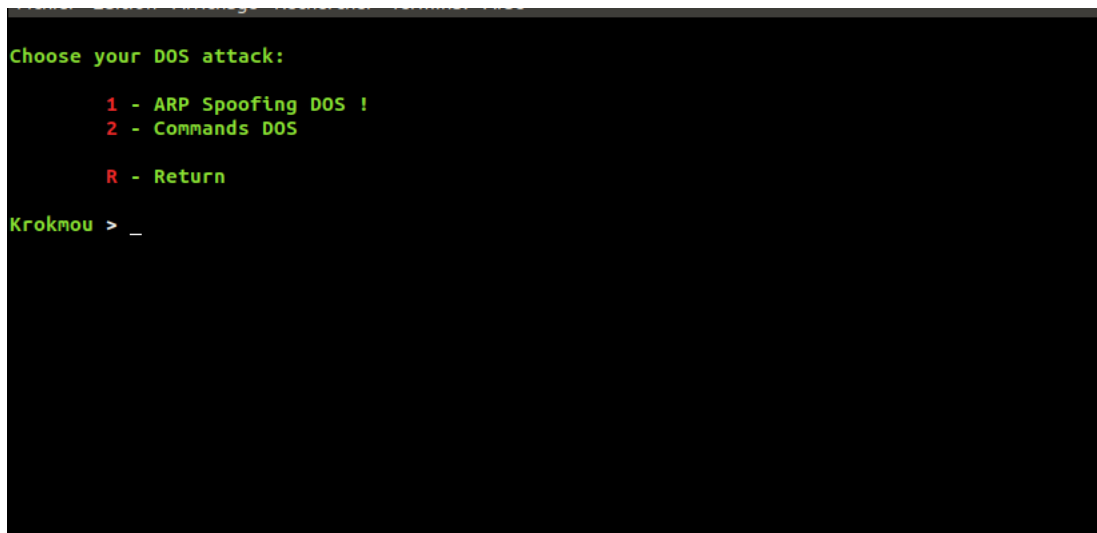
FIGURE 17 – Envoi du virus et du fichier sur le drone

6.6 Déni de service

La dernière option permet de réaliser deux types de dénis de service. La première possibilité, le déni de service complet par ARP spoofing, se base sur l'attaque de "l'Homme du milieu". Une fois sélectionnée, cette option permet de bloquer le lien entre le client et le drone et donc toutes communications.

Pour la seconde possibilité, on bloque la légitimité des paquets de commande du client. Ce qui bloque uniquement la liaison de commande.

Dans les deux cas, le déni de service s'arrête quand l'attaquant le désire.

A screenshot of a terminal window with a black background and green text. The prompt is 'Krokmou > _'. The menu lists two options: '1 - ARP Spoofing DOS !' and '2 - Commands DOS', followed by 'R - Return'.

```
Choose your DOS attack:

 1 - ARP Spoofing DOS !
 2 - Commands DOS

R - Return

Krokmou > _
```

FIGURE 18 – Déni de service sur le drone

7 Sécurisation du drone

Comme nous avons pu le voir, cet ARDrone 2.0 de **Parrot** rencontre de nombreux problèmes de sécurité et reste vulnérable à certaines attaques. L'une des vulnérabilités principales se trouve dans le point d'accès Wifi qui est un réseau ouvert donc accessible à toute personne se trouvant à portée du drone.

7.1 Sécurisation via le Wifi

Modification du point d'accès L'une des premières mesures pour ce drone serait donc une modification de ce point d'accès Wifi en y ajoutant un mot de passe afin de le rendre privé. Afin de minimiser les risques de compromission du réseau, l'utilisation de la norme WPA2 semble optimale. Après un rapide état de l'art sur Internet, il semblerait que personne ne se soit réellement penché sur le problème mais un fichier bash présent sur le drone, *wifi_setup.sh*, laisse présager que cette modification est possible. On peut aussi noter que les nouveaux drone de la société Parrot possèdent une meilleure sécurité au niveau du Wifi. En effet, le Bebop propose un Wifi WPA2 avec mot de passe mais ce n'est pas la configuration par défaut.

Utilisation du drone comme un client Une autre solution consisterait à utiliser le drone non comme un point d'accès mais comme un client du Wifi. Afin de pouvoir connecter le drone à un réseau en WPA2, il est nécessaire d'effectuer de la compilation croisée sur le module *wpa_supplicant*, qui gère les connections au wifi WPA2 sur les environnements Unix. En effet, le drone possède une architecture ARM et non x86. Heureusement, la communauté est riche de talent, et ce travail a déjà été fait dans ce dépôt Github : <https://github.com/daraosn/ardrone-wpa2>.^[4]

Cette manipulation se fait en plusieurs étapes :

- installation du module *wpa_supplicant* sur le drone.
- connecter le drone au point d'accès WPA2 du téléphone.
- faire en sorte que le drone ait l'adresse 192.168.1.1. Changer l'adresse du point d'accès pour le permettre.
- contrôler le drone via le téléphone.

Avec ce procédé, on bénéficie alors d'une connexion sécurisée avec le drone que l'on pilote. Attention toutefois à bien choisir son mot de passe. En effet, si une personne arrive à se connecter au Wifi du téléphone, le drone redevient totalement vulnérable. La contrainte de cette technique est donc l'utilisation d'un point d'accès Wifi annexe avec la bonne adresse IP pour le routeur. Dans un premier temps, nous l'avons essayer rapidement avec un iPhone comme point d'accès mais celui-ci n'offre pas un réseau interne du type *191.168.1.0/24* et il n'est pas possible de modifier cela dans les paramètres.

7.2 Renforcement du numéro de séquence

Il a été mis en évidence que le drone interprète toutes les commandes avec un numéro de séquence supérieur au sien. Même s'il est nettement supérieur à celui en cours, il sera interprété.

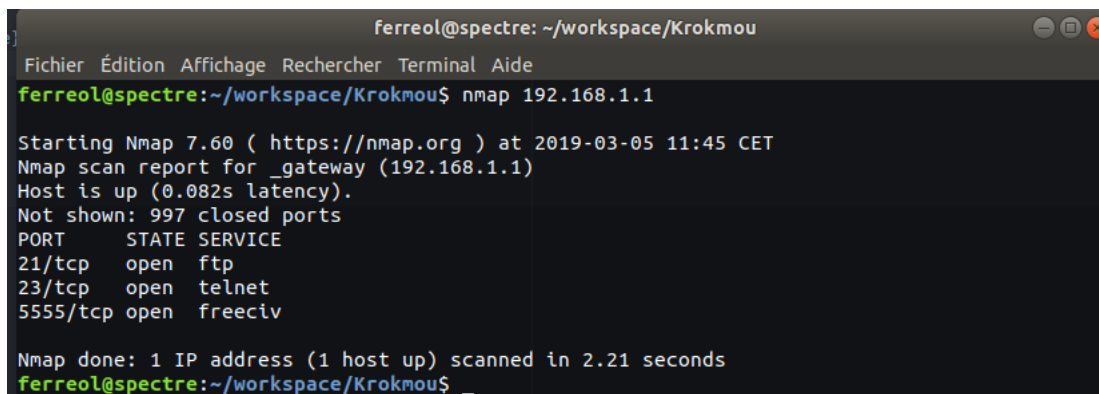
Une idée de renforcement de ce dernier est de prévoir le numéro de séquence probable. On sait que le client envoie des paquets UDP de façon régulière. On peut donc mettre

en place une fourchette pour estimer le numéro de séquence suivant. Ainsi le numéro de séquence doit toujours être supérieur à celui en cours mais aussi inférieur à celui probable en connaissant la fréquence d'envoi du client légitime. On pourra également ajouter une petite marge d'erreur sur la borne supérieur mais il faut que cette dernière soit raisonnable.

Avec ce procédé, on bénéficie alors d'une connexion sécurisée avec le drone que l'on pilote.

7.3 Connexion à distance

Comme vu précédemment, le drone dispose de 3 services **TCP** ouverts dont le service **FTP** et le service **Telnet**. Ces deux services sont très utiles pour la dépose de fichiers ou encore la maintenance du drone à distance. Mais ils représentent surtout deux portes ouvertes sur le drone et sont donc de réelles menaces pour la sécurité du drone.



```
ferreol@spectre: ~/workspace/Krok mou
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
ferreol@spectre:~/workspace/Krok mou$ nmap 192.168.1.1

Starting Nmap 7.60 ( https://nmap.org ) at 2019-03-05 11:45 CET
Nmap scan report for _gateway (192.168.1.1)
Host is up (0.082s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
23/tcp    open  telnet
5555/tcp  open  freeciv

Nmap done: 1 IP address (1 host up) scanned in 2.21 seconds
ferreol@spectre:~/workspace/Krok mou$ _
```

FIGURE 19 – Services disponibles sur le drone

La solution la plus simple serait de supprimer ces deux services et de les remplacer par un seul et même service, le service **SSH**. En effet, celui-ci combine l'aspect transfert de fichier de **FTP** et l'accès à distance de **Telnet** en plus d'offrir des garanties de sécurité avec notamment la présence d'une authentification à la connexion. Attention cependant, cette solution n'est viable que si le mot de passe par défaut est changé à la première utilisation du drone.

8 Conclusion

L'ARDrone 2.0 de **Parrot** est donc un drone grand public de loisirs qui présente des vulnérabilités importantes. La principale est le réseau Wifi ouvert qui autorise toute personne à portée de signal Wifi à se connecter au drone. La sécurisation du réseau Wifi créé par le drone en utilisant un réseau Wifi WPA2 permettrait de combler cette faille majeure. Dans un second temps, les ports ouverts sur le drone donnent accès à des services sur celui-ci sans authentification de la part de l'utilisateur. Ainsi un accès **FTP** permettant un accès aux fichiers du drone est laissé à tout utilisateur connecté au réseau Wifi de celui-ci. De plus, faille plus importante, un service **Telnet** est également accessible fournissant un accès **root** sur le drone à tout utilisateur connecté au Wifi. Cet accès est une faille majeure car elle permet à un attaquant d'avoir tous les droits sur le drone et ainsi de pouvoir faire ce qu'il veut au niveau du système d'exploitation de celui-ci. Dans un troisième temps, le contrôle du drone présente également des failles dans la gestion des commandes envoyées par l'utilisateur. En effet, il est possible pour un attaquant de forger de fausses commandes en se faisant passer pour l'utilisateur légitime et ainsi de contrôler le drone. Le drone n'effectue pas d'authentification des commandes qu'il reçoit et ne vérifie donc pas que celle-ci viennent de l'utilisateur légitime. Pour terminer, nous avons démontré qu'il était possible d'exploiter ces différentes failles afin de prendre le contrôle du drone mais elles peuvent également servir à réaliser des attaques de type Déni de Service sur celui-ci et ainsi bloquer la connexion entre lui et le client.

La place de plus en plus importante que prennent ces drones de loisir dans l'espace aérien va nécessiter dans le futur que ceux-ci ne présentent plus ce type de failles majeures afin d'empêcher que des attaquants puissent en prendre le contrôle ou plus simplement puissent réfuter le contrôle de l'aéronef à son utilisateur légitime, et ceci pour des raisons de sécurité vis à vis des autres aéronefs évoluant dans le même espace.

Bibliographie

- [1] Parrot ar.drone. https://fr.wikipedia.org/wiki/Parrot_AR.Drone.
- [2] Parrot ar.drone 2.0. <https://www.parrot.com/fr/drones/parrot-ardrone-20-elite-edition#parrot-ardrone-20-elite-edition>.
- [3] Thomas TROMPETTE Antoine MARINIER, Mickael RANAIVOARISOA. Étude de la sécurité d'un drone. https://ensiwiki.ensimag.fr/index.php?title=Etude_de_la_s%C3%A9curit%C3%A9_d%27un_drone#Sc.C3.A9narios_d.E2.80.99attaque.
- [4] Diego ARAOS. ardrone-wpa2. <https://github.com/daraosn/ardrone-wpa2>.
- [5] DOYEN-LE BOULAIRE Marine CLAURE CABRERA Oscar Mike. Analyse du système de sécurité du drone :ar.drone 2.0quad-copter. <https://ensiwiki.ensimag.fr/images/a/ab/Drone.pdf>.
- [6] Functino. drone-browser. <https://github.com/functino/drone-browser>.
- [7] Samy KAMKAR. Skyjack. <https://github.com/samyk/skyjack>.