Rapport de projet long
Attaques sur un AR Drone 2.0 Parrot

Kevyn Ledieu, Ferréol Pennel, Alexis Pernot

5 mars 2019

Table des matières

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 pouvons observer des exemples de drones ayant perturber 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 les 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 hardware 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 *un magnétomètre 3 axes*
- des *capteurs de proximité à ultrasons*
- une *caméra vericale QVGA*
- un *port USB 2.0*

2.2 Méthode

3 Hypothèse de travail

Pour ce projet, nous supposerons 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 éventuelles 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 a 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 reprendre 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 envoyé. 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 et 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. Ce script est chargé de copier une image sur toute clé USB connectée au drone ceci afin de démontrer le potentiel de l'attaque. Il est en effet aisé de diffuser n'importe quel virus qui se diffuse par clé USB grâce au drone et ceci sans difficultés particulières.

4.5 Conclusion

Ces trois 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 trois 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 trois 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 et la dernière sera l'exploitation d'une connexion Telnet sur le drone.

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 à 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 dur drone

```
192.168.1.1
```

afin de connaître les ports ouverts sur le drone et découvrir des moyens d'accéder à celui-ci. Nous obtenons les resultats suivants :

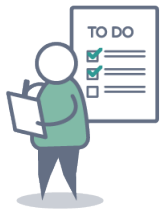

FIGURE 1 – Résultat du scan *nmap* sur le drone

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 du drone car nous avons un accès illimité au système d'exploitation du drone.
Dans le même temps, nous nous documentons sur l'ARDrone 2.0. Grâce au SDK disponible sur le site **Parrot**, nous sommes en mesure de comprendre comment forger des commandes pour les envoyer au drone ainsi que les règles de contrôle du drone. 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 du drone, 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 en déconnectant le client légitime. Une fois connecté au réseau Wifi du drone, nous effectuons une désauthentification des clients connectés au drone grâce à **Aireplay** (cf Figure X.X).

```
def eject_client(client, mac_drone, drone_essid, iface_mon):
    channel = find_channel(mac_drone)
    os.system("sudo iwconfig {0} channel {1}".format(iface_mon, channel))
    cmd = "sudo aireplay-ng -0 1 -e {3} -a {0} -c {1} {2} >>/dev/null 2>>/dev/null".format(mac_drone, client.mac, iface_mon, drone_essid)
    for i in range(10):
        os.system(cmd)
```

FIGURE 2 – Utilisation d'**Aireplay** pour désauthentifier les clients

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

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** régler la référence au plan horizontal (doit être au sol)
- **CONFIG** configuration de l'AR Drone 2.0
- **CONFIG_IDS** identifiants pour les AT*CONFIG commandes
- **COMWDG** réinitialise la communication
- **CALIB** demande au drone de recalibrer le magnétomètre (doit être en vol)

Dans notre cas, nous allons uniquement injecter des commandes **REF** et **PCMD**. Voyons le format des 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 envoie 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-sûr d'être supérieur à celui en cours. On incrémentera également celui-ci à chaque envoie 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 le décollage. 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 puis en l'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 fessant 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 3 – Contrôleur pour l'injection de commande

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

FIGURE 4 – 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 **AJOUTER NOM PROCESSUS**.

FIGURE 5 – Processus s'exécutant sur l'ARDrone 2.0

Il 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. Etant **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 *Krokmou* 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.

FIGURE 6 – Dépose du virus et du fichier par l'application

Cette attaque démontre que le drone peut facilement servir de vecteur d'attaque pour diffuser un virus par clé USB.

6 Tutoriel

Le tutoriel du projet se présente sous la forme d'une application Linux développée en Python permettant d'exploiter les trois 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 drone 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 au drone.

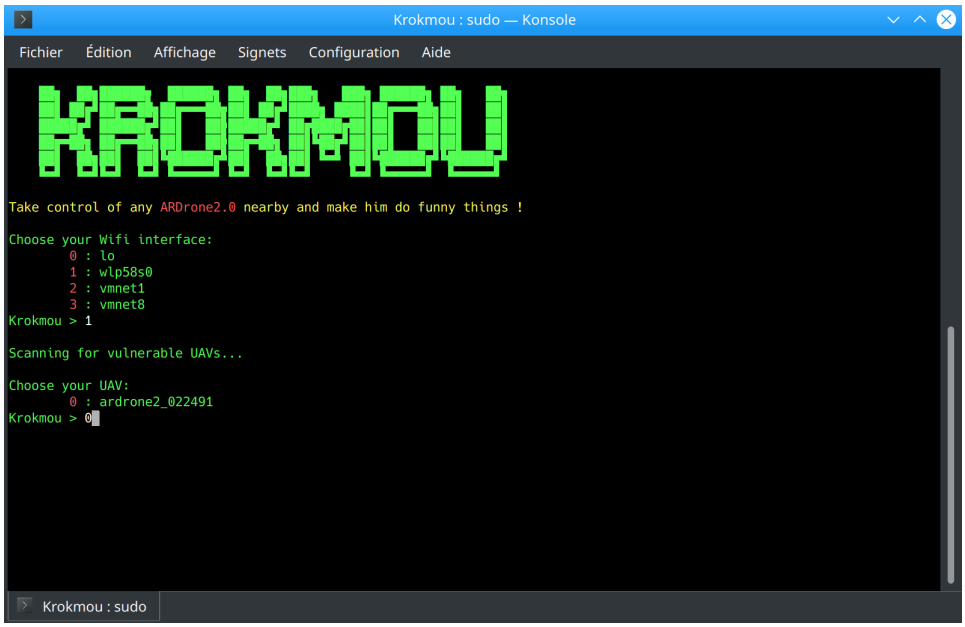

FIGURE 7 – Initialisation de l'application

6.2 Menu principal

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

FIGURE 8 – 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 d'un dépôt Github (`https://github.com/functino/drone-browser`).

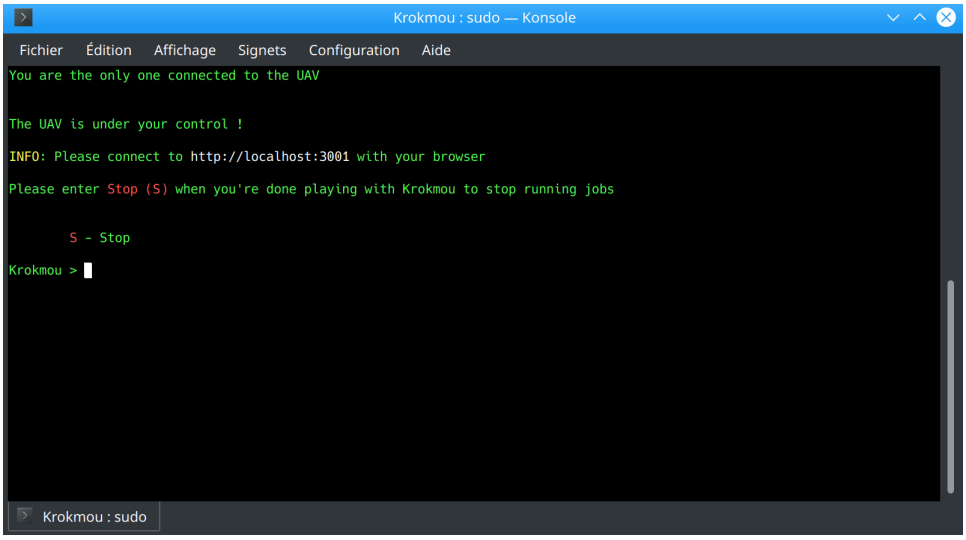

FIGURE 9 – Prise de contrôle du drone par l'application

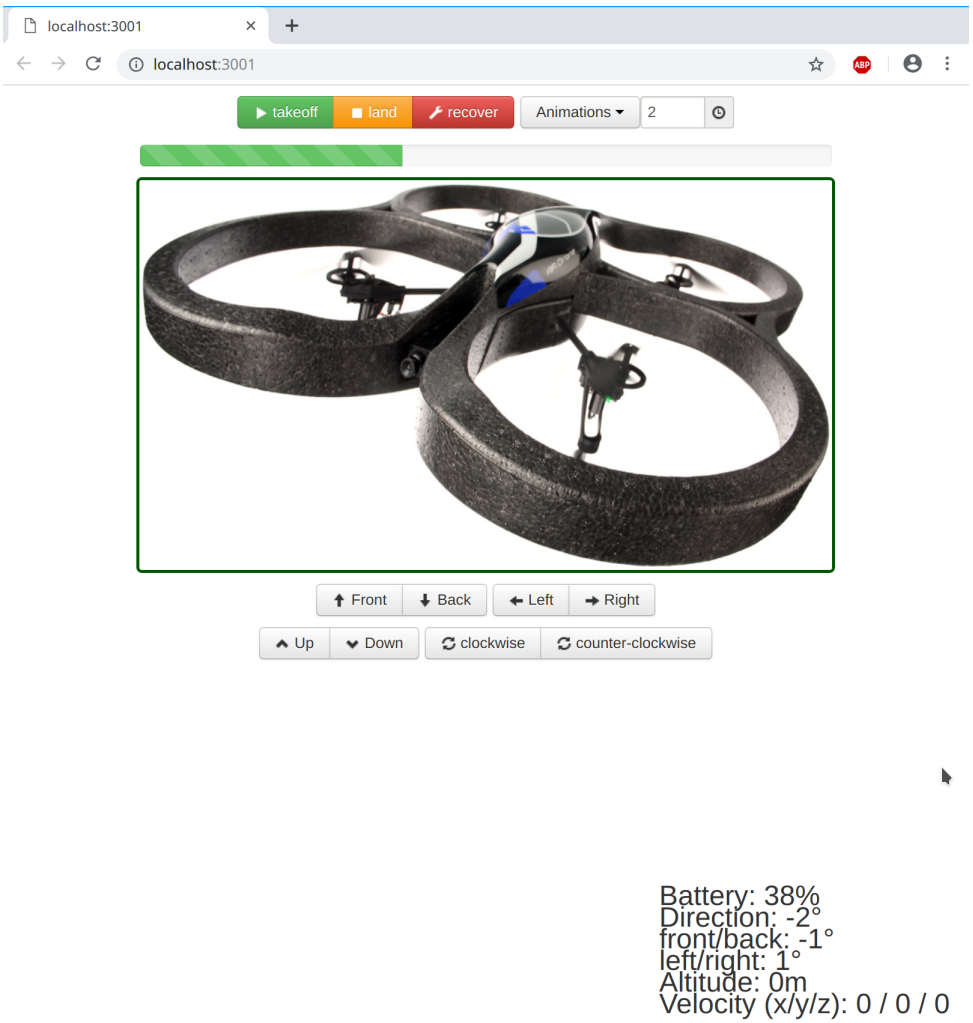

FIGURE 10 – Interface de contrôle web

6.4 Injection de commandes

Ce sous-menu permet à l'attaquant d'envoyer des instructions au drone sans déconnecter le client légitime de celui-ci à travers une injection de paquets.

FIGURE 11 – Sous-menu d'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 qui 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.

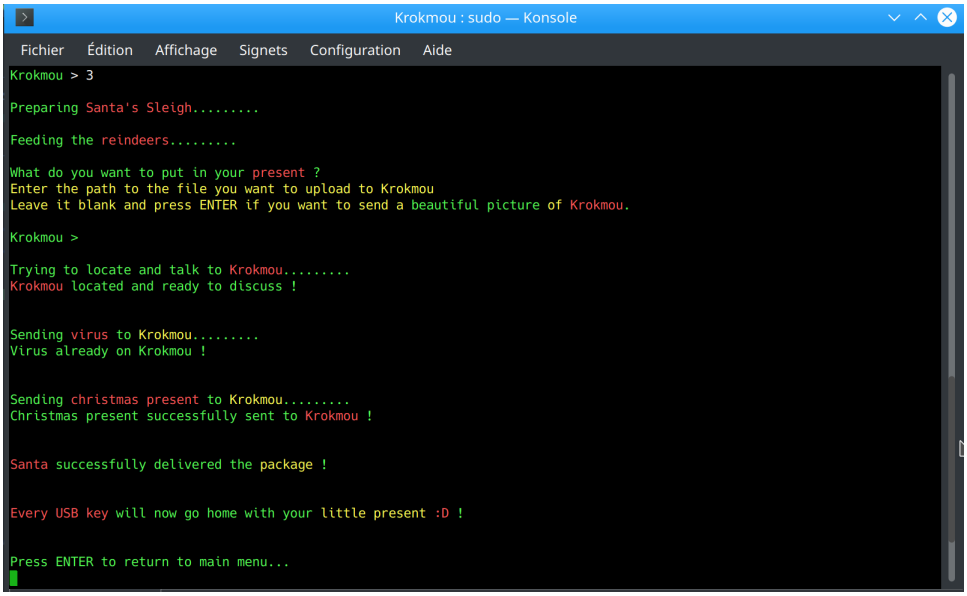

FIGURE 12 – Envoi du virus et du fichier sur le drone

7 Sécurisation du drone

Comme nous avons pu le voir, cet AR Drone 2.0 de Parrot rencontre de nombreux problèmes de sécurité et reste vulnérable a certaines attaques. L'une des vulnérabilité principale 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 Modification du point d'accès Wifi

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 privée. Afin de minimiser les risques de compromission du réseau, l'utilisation de la norme WPA2 semble optimale. Cependant, il ne semble pas possible de changer la configuration Wifi de l'AR Drone. Il est donc nécessaire d'inverser la connection et d'utiliser le drone comme un client Wifi et non comme un Access-Point. 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 environement Unix. En effet, le drone possède un architecture ARM et non x86. Heureusement, la communauté est riche de talent, et ce travail à déjà était fait dans ce dépot Github : `https://github.com/daraosn/ardrone-wpa2`.

Cette manipulation se fait en plusieurs étapes :

— installation du module *wpa_supplicant* sur le drone.
— conecter le drone au point d'accès WPA2 du téléphone.
— faire en sorte que le drone est 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.

8 Conclusion