Anomaly Detection in Drone Activities: Data Collection and Unsupervised Machine Learning Modeling

by

Zhuo Chen

B.Eng, University of Regina, Regina, 2023

A Report Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

University of Victoria

Supervisory Committee

Comprehensive Study on Anomaly Detection in Drone Data: Data Collection, Feature Extraction, Sets Selection, and Model Performance Evaluation

by

Zhuo Chen

B.Eng, University of Regina, Regina, 2023

Supervisory Committee

Dr. Issa Traoré, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Mohammad. Mamun, Co-Supervisor
(Department of Electrical and Computer Engineering)

Abstract

As Internet of Things (IoT) devices, drones are among the most popular unmanned aerial vehicles (UAVs), equipped with multiple sensors, cameras, and communication systems. These features expose them to potential vulnerabilities exploitable by hackers. making it crucial to explore these vulnerabilities and implement effective anomaly detection while operating UAVs. This study investigates a DJI Edu Tello drone to comprehensively assess its vulnerabilities and develop anomaly detection mechanisms using different unsupervised machine learning techniques. Two types of data were collected: benign data from legitimate actions and attack data comprising nine types of attacks. Feature extraction and engineering were performed based on scripts from the Canadian Institute for Cybersecurity (CIC), which were modified to suit the specific needs of this project. The modifications aimed to improve the robustness of the detector by removing and modifying existing features and introducing new measurements to represent the captured packets. The anomaly detector was formulated after comparing three unsupervised machine learning algorithms: Isolation Forest, Local Outlier Factor (LOF), and Elliptic Envelope, through extensive performance evaluations and analyses. The study demonstrated the effectiveness of these algorithms in detecting anomalies and enhancing the security of drones. The findings also highlight the critical role of robust feature engineering and careful algorithm selection in developing a reliable anomaly detection system for UAVs.

Table of Contents

List of Tables

List of Figures

Glossary

IoT: Internet of Things

UAV: Unmanned Aerial Vehicle

CIC: Canadian Institute for Cybersecurity

LOF: Local Outlier Factor

IF: Isolation Forest

EE: Elliptic Envelope

ML: Machine Learning

DNS: Domain Name System

DoS: Denial of Service

DDoS: Distributed Denial-of-Service

AI: Artificial Intelligence

IDS: Intrusion Detection System

SVM: Support Vector Machines

FNN: Feedforward Neural Networks

LSTM-RNN: Long Short-Term Memory Recurrent Neural Networks

1D-CNN: 1D Convolutional Neural Networks

UAVCAN: Uncomplicated Application-level Vehicular Computing And Network

CAN: Controller Area Network

RF: Random Forest

SDL: Shallow Deep Learning

VCDL: Vector Convolutional Deep Learning

SNDA: Stacked Nonsymmetric Deep Autoencoders

SCA: Stacked Contractive Autoencoder

LSTM: Long Short-Term Memory

CGAN: Convolutional Generative Adversarial Network

CNN3D: 3D Convolutional Neural Networks

CSV: Comma Separated Value

AP: Access Point\

IP: Internet Protocol

MAC: Media Access Control

OUI: Organizationally Unique Identifier

SSID: Service Set Identification

BSSID: Basic Service Set Identifier

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

ARP: Address Resolution Protocol

MITM: Man-in-the-middle

PCAP: Packet Capture

WPA: Wi-Fi Protected Access

PCA: Principal Component Analysis

ROC Curve: Receiver Operating Characteristic Curve

AUC: Area under the ROC Curve

TP: Ture Positive

FN: False Negative

FP: False Positive

TN: True Negative

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Issa Traoré, for his efficient assistance and invaluable guidance throughout my master's program. He was always responsive whenever I needed direction, and his support made the pursuit of my degree more manageable and gave me confidence. Dr. Issa Traoré encouraged me to develop my skills independently and consistently demonstrated his professionalism, responsiveness, and enthusiasm in providing support. I am truly grateful to have him as my supervisor.

I also want to extend my heartfelt thanks to my parents for their unwavering support in pursuing my master's degree, and to my friend, Chengyu Lou, for his encouragement.

Dedication

I would like to respectfully dedicate my Master of Engineering project report to the memory of my grandfather.

Chapter 1: Introduction

1.1 Context

Civilian drones, also known as Unmanned Aerial Vehicles (UAVs), can be considered part of the Internet of Things (IoTs). They are becoming increasingly popular across many fields such as public safety and security, photography and filmmaking, agriculture, geographic surveying and mapping, and more. This popularity is due to their ability to connect to the Internet directly or indirectly, allowing for data synchronization, remote control, and software updates.

According to Statista Market Insights data from March 2024, the number of drones worldwide has been increasing since 2020 and is expected to reach 9.3 million units by 2028 [1]. With such growth prospects, protecting drones from cyberattacks is becoming increasingly necessary and expected by industries and users, as these devices play a crucial role in enhancing productivity and maintaining their privacy [2].

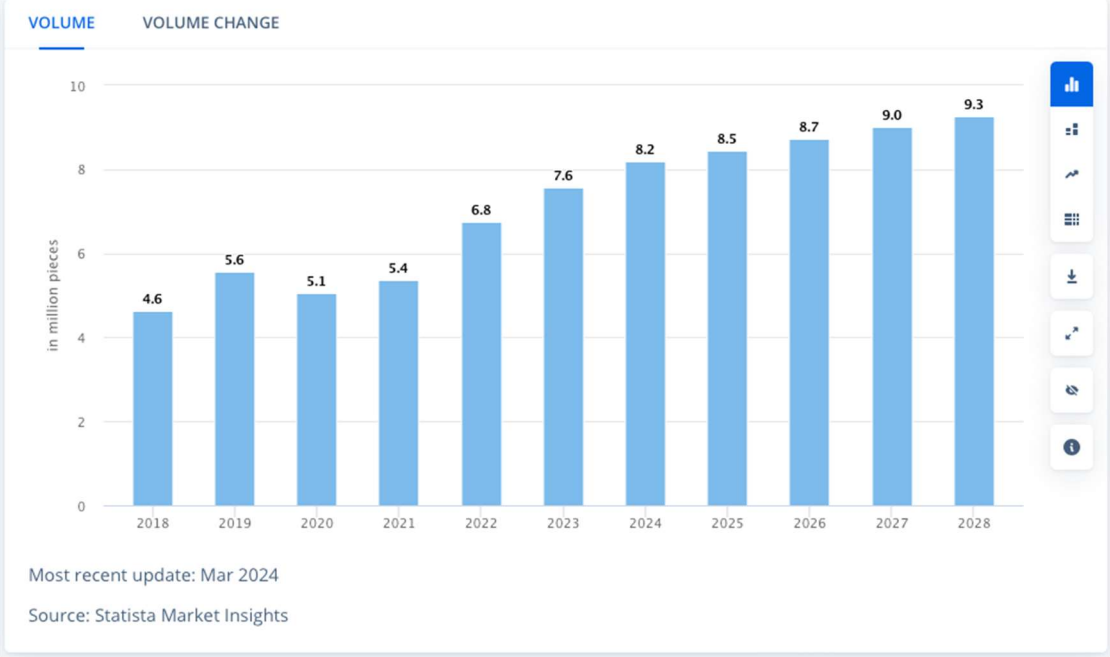

Figure 1.1: Drone Volume Growth Worldwide in 2024 [1].

Cybersecurity for drones has become a fundamental concern to prevent these devices from being exploited for illegal and unauthorized activities. There are two main types of drones categorized based on their communication capabilities: short-range and long-range. Short-range civilian drones commonly use Wi-Fi to establish communication between the devices and the controller (smartphone or tablet) [3]. In contrast, long-range drones are equipped with cellular connectivity (such as LTE, 4G, 5G) or radio frequencies due to their reliability and extended range capabilities [4]. The Wi-Fi-based communication drone is being investigated in this study with Artificial Intelligence (AI) technology.

Machine Learning (ML) has a fundamental connection with cybersecurity because of its properties of scalability and effectiveness in detecting ongoing malicious or unauthorized activities, making the process more practical and efficient than traditional methods that require human involvement [5].

1.2 Related Work

A study by G. Rubbestad and W. Söderqvist (2021) investigated the vulnerabilities and penetration threats of the Tello drone [6], providing information on the feasibility of penetration testing. They executed five attack types: password cracking, denial of service (DoS) attack, Address Resolution Protocol (ARP) spoofing, instruction injection, and video interception. All the attacks were successful. This research evaluated the baseline vulnerabilities of the Tello drone.

Neto et al. (2023) proposed an extensive IoT attack dataset featuring 33 types of attacks across a topology of 105 IoT devices [7]. The attacks are categorized into six main types: Distributed Denial-of-Service (DDoS), Brute Force, Spoofing, DoS, Recon, Web-based, and Mirai botnet. Over 60 features were engineered and extracted from the captured traffic. These features were compiled into a comma-separated values (CSV) format dataset using Python scripts.

Another study by Bertoli, Pereira, & Saotome (2021) focused on DoS attacks on a WiFi-based drone (Parrot AR.Drone) [8]. Various types of DoS attacks were conducted on the

Wi-Fi 802.11 wireless protocol, including de-authentication, User Datagram Protocol (UDP) flood, and Transmission Control Protocol (TCP) flood. The researchers then trained a classifier on the dataset containing both regular and attack data to detect these disruptions.

Hassler et al. (2023) conducted an experiment to develop an Intrusion Detection System (IDS) for the DJI Tello EDU drone, including the creation of datasets and the formulation of the IDS [9]. The datasets consist of three types: cyber-only, physical-only and cyber-physical. For the cyber dataset collection, they performed four types of cyber-attacks: de-authentication DoS attacks, replay attacks, false data injection attacks, and evil twin attacks. The physical dataset contains properties such as height, x, y, z speeds, and the pitch of the drone. They extracted 32978 samples with 37 cyber features and 6236 samples with 16 physical features. When combined, the cyber-physical dataset resulted in 12741 benign samples and 16843 anomalous samples.

Once the dataset was processed, Hassler et al. (2023) trained the IDS with four supervised methods: Support Vector Machines (SVM), Feedforward Neural Networks (FNN), Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN), and 1D Convolutional Neural Networks (1D-CNN) [10]. The study evaluated the performance of the IDS using the proposed fusion dataset, comparing it with the cyber-only and physical-only datasets.

Kim et al. (2022) also contributed to another UAVCAN dataset, [11]. The research team used 10 distinct scenarios to conduct attacks and then collected the resulting data. They focused on three main attack types: Flooding, Fuzzy, and Replay. The attack data was gathered by varying the combination of attacks and intervals across the 10 scenarios. This dataset includes two classes: benign and attack. Each sample contains six specific fields: label, timestamp, interface, Controller Area Network (CAN) ID, data length, and data.

Wu et al. (2024) introduced a new supervised machine learning model for intrusion detection in UAVs: a Tiny Machine Learning Model [12]. This model aims to detect anomalous drone activities with limited computing resources by utilizing an efficient feature selection method. It enhances performance by combining Random Forest (RF) with

Shallow Deep Learning (SDL). The study employed two datasets: CSE-CIC-IDS2018 and CIC-DDOS2019, both of which primarily focus on DoS and DDoS attacks. In the experiment, the proposed model was compared with eight other methods: SVM, vector convolutional deep learning (VCDL), stacked nonsymmetric deep autoencoders (SNDA), stacked contractive autoencoder and SVM (SCA+SVM), long short-term memory (LSTM), convolutional generative adversarial network (CGAN), and 3D Convolutional Neural Networks (CNN3D).

1.3 Objective

This project intends to comprehensively study the Wi-Fi-based DJI Edu Tello drone by collecting a dataset, extracting features from the captured packets, and developing an anomaly detector using unsupervised ML models. In the first phase, we investigated the drone's vulnerabilities to establish baseline for underlying attack vectors. Then, nine types of attacks were conducted to collect the anomalous samples, while regular data samples were also collected during legitimate operations. Feature engineering and extraction were carried out to define robust features representing the captured communication packets. Since this project refers to the feature extraction script from the CIC research team [7] as the baseline, modifications are expected to improve the suitability of the features specifically for this project. Consequently, both the original and proposed versions of the feature set are included in the evaluation process to determine their performance based on specific metrics. While selecting the optimal performance feature set, two of the three ML-based algorithms are concurrently chosen for the final phase of anomaly detection formulation. The detector formulation process involves evaluation and analysis to select the best-performing algorithm. In this way, the project achieves its objectives of contributing to Wi-Fi-based drone datasets, extracting features specifically for the drone, and formulating an anomaly detector to enhance the cybersecurity of Wi-Fi-based drones.

Having outlined the main objectives of this project, it is essential to highlight the distinctions between this work and those previously mentioned in Section 1.2. This project employs unsupervised learning to detect a broad range of anomalous actions, in contrast to

the supervised methods used in related works, which are limited to detecting the types of anomalies seen during training. Additionally, it introduces a comprehensive dataset covering nine distinct attack mechanisms, while the scope of existing datasets focuses on fewer attacks. Furthermore, this work enhances a feature extraction strategy to improve both performance and robustness, beyond the typical focus on performance alone.

1.4 Report Outline

The report structure is as follows:

Chapter 1 introduces the context of the project, the related work, the project objective, and an outline of the report.

Chapter 2 provides background information on the drone device system, including categorization of packets, characterization of its communication, essential knowledge of its vulnerabilities, potential risks associated with drone security breaches, and the anomaly detection mechanisms.

Chapter 3 details the data collection process in the project, including a defined attack set and normal operations, and covers the implementation from low-level techniques to high-level observations.

Chapter 4 explores the feature extraction methods detailing both the original feature engineering approaches and the newly proposed feature scripts. The feature set performance evaluation is also conducted to select a robust feature set for use in the anomaly detector training process.

Chapter 5 outlines the experimental procedure for evaluating the performance of ML-based algorithms for finalizing the anomaly detector. It details the comprehensive performance evaluation and analysis process for the anomaly detection system.

Chapter 6 makes concluding remarks and suggestions for future work.

Chapter 2: Background and Model Architecture

In this chapter, the communication mechanism of the DJI Edu Tello drone will be introduced to provide essential knowledge on its technology and vulnerabilities. This will include background information on classification, characteristics of communications; vulnerability analysis; and the risks of cybersecurity breaches. Additionally, the chapter will give an overview of the anomaly detection system design workflow.

2.1 Basics

The DJI Edu Tello drone is a Wi-Fi-based UAV device that supports remote control through both the Tello mobile application and the Tello SDK. The drone uses the UDP protocol for all communications, including sending and receiving packets. As shown in Figure 2.1, the drone primarily uses port 8889 to communicate with the controller for both the mobile app and the SDK connections. However, it uses different ports for sending video stream packets. When establishing a connection with the drone, it serves as a Wi-Fi access point, creating its own Wi-Fi network (SSID: TELLO-xxxxxx with IP: 192.168.10.1), which enables controllers to connect to its network. The controller must connect to the same network as the drone, regardless of whether the mobile app or the SDK is being used. By default, the drone does not have a Wi-Fi password for authentication. However, both the mobile app and SDK offer the functionality to set a Wi-Fi password manually.

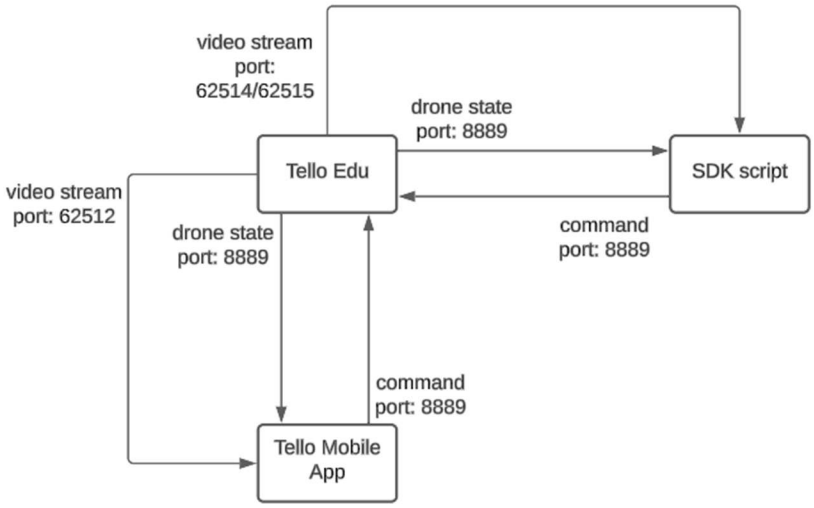

Figure 2.1: Tello Edu Drone Communication Structure.

The mobile app enables users to intuitively control the drone by simply touching and guiding it with their fingers on the screen. The app's interface provides a user-friendly experience, allowing users to manipulate the drone effortlessly through touch gestures. The action space within the mobile app includes:

- Regular operations: takeoff, land, up, down, rotate clockwise, rotate counterclockwise, forward, back, left, right, take photo, take video, flight speed adjustment, Wi-Fi setup.

- Flight modes: Bounce mode, 8D flips, Throw & Go, Up & Away, 360, Circle.

- Basic information display: battery life, height, speed, firmware version, etc.

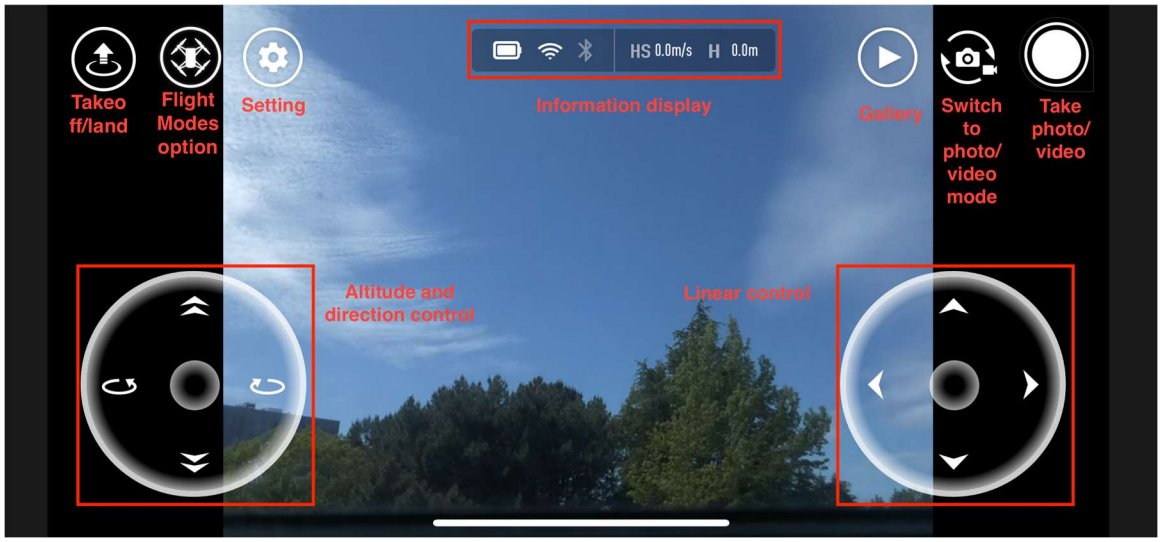

Figure 2.2: Tello Mobile App User Interface.

Figure 2.2 gives an outline of the mobile App interface.

Conversely, the Tello SDK allows users to control the drone through text-based commands via scripts. The SDK offers extensive capabilities like those of the mobile app but includes unique commands such as streamon, streamoff, emergency, curve, and ap ssid pass. A notable feature of the SDK command ‘ap ssid pass’ is its ability to set the drone to station mode, allowing it to connect to an existing network instead of creating its own [13]. This

functionality is particularly useful for coordinated formation flights. After the command is received by the drone, there are two possible responses: either ‘ok’ or ‘error’ as shown in Figure 2.3.

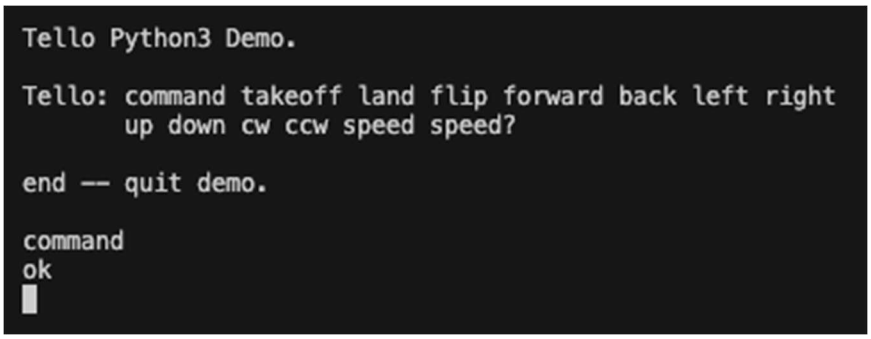

Figure 2.3: Tello SDK Sample Command.

2.2 Classification of Communications

The communications can be categorized into three main types based on the content of the packets: video stream, drone state, and user command. Every type of packet travels in a unidirectional manner from one end to the other.

Video stream communication: This communication is established to transmit video content only from the drone to the controller. The video stream is continuous, and the packet payload size is standard at 1460 bytes. It uses different ports to send the packets; for example, the mobile app uses 62512, while the SDK uses either 62514 or 62515 on the drone.

Drone state communication: It is utilized to update the drone’s state to the controller, including speed, battery life, altitude, Wi-Fi signal strength, firmware version, and more. This type of connection is also one-way from the drone to the controller. Unlike the video stream, which requires a large and intensive packet flow, the state packet size is relatively small, varying from 30 to 500 bytes. It uses port 8889 of the drone consistently for both the mobile app and the SDK.

User command communication: This type of communication is designed to send commands from the controller to the drone. These packets contain any legitimate command

within the action space during regular operation. They are standard packets, typically 22 bytes or less in size. Like the state packets, the drone uses port 8889 to receive these commands for both connection methods.

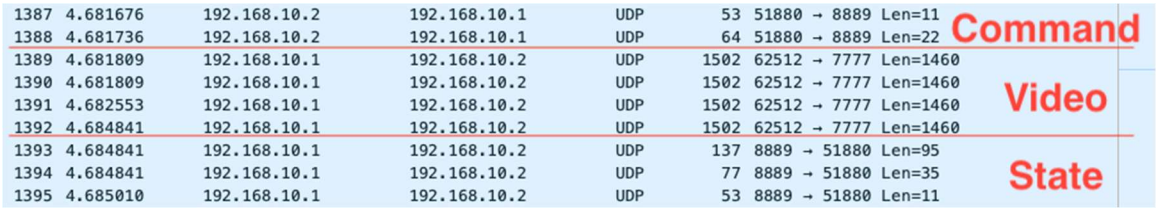

Figure 2.4: Sample benign packets.

It's worth noting that each type of communication is independent. This indicates that as long as the connection is established between the controller and the drone, these three types of packets are not dependent on each other. Figure 2.4 shows sample packets in the different categories.

2.3 Characteristics of Communications

These three types of packets possess certain characteristics that contribute to effective communication between the two devices. These characteristics include the following:

1. Standardized Payload Size: The different types of packets have a standard or small range of payload sizes.
2. Common OUI: The packets share the same Organizationally Unique Identifier (OUI) of 60:60:1F.
3. UDP Protocol and Independence: The packets use the UDP protocol and are transmitted independently, regardless of the state of another device or any previous packets sent and received.
4. Heartbeat Packets: Each type of packet is sent periodically as a heartbeat packet with a specific transmission frequency.
5. IP Layer: Since they use the UDP protocol, all packets include an IP layer.
6. Fixed Ports: The ports used by the drone are fixed for specific purposes, including sending video stream packets, updating state information, and receiving command packets.

Identifying these distinct characteristics can potentially improve the feature extraction process in Chapter 4 by enhancing the robustness of the features in representing the communication packets.

2.4 Vulnerability Analysis of the Drone

The DJI Tello Edu drone, like many IoT devices, possesses several potential vulnerabilities due to the properties of the technologies used and the design of the communication mechanisms, which could be exploited by cyber attackers. These vulnerabilities include:

1. Wi-Fi security: As the drone relies on the IEEE 802.11 wireless network standards for communication, there is a potential risk of exposing its network to hackers for de-authentication attacks, even if the Wi-Fi password is manually set. Following that, subsequent malicious actions could be also performed, such as man-in-the-middle (MITM) attacks, and more.
2. Fixed ports for communication: Since the drone utilize fixed ports for communication, it is vulnerable to exploitation by DoS attacks, if the port numbers are known.
3. Multiple connection: This drone supports multiple simultaneous connections without sufficient security measures, including the SDK and mobile app, which also present potential vulnerabilities. Attackers could connect to the device and perform malicious actions such as command injection and information theft, including intercepting video feeds.
4. Unencrypted Communications: The drone communicates using UDP packets, which do not provide encryption for the content. This exposes the vulnerability of sending unauthorized packets and allows attackers to manipulate the payload of the packets.

Based on these vulnerabilities, specific attack vectors will be detailed and demonstrated in Chapter 3.

2.5 Potential Risks of Cybersecurity Vulnerabilities

The negative impacts of cybersecurity vulnerabilities are discussed as follows:

Privacy data leakage: As the drone is equipped with a camera and multiple sensors [15], sensitive and private data could potentially be intercepted and manipulated by attackers through its vulnerabilities.

Personal safety threat: Unauthorized actions could be performed by hackers exploiting these vulnerabilities, potentially resulting in posing a threat to the personal safety of anyone nearby.

Personal property damage: Some vulnerabilities can cause the drone to lose control from the user, potentially resulting in physical damage to the drone, such as its external structure.

These consequences of the drone's vulnerabilities underscore the crucial role of anomaly detection in preventing unauthorized actions. By detecting anomalous activities in packet communication effectively, the system can mitigate the damage from cyberattacks and enhance the drone's security.

2.6 Anomaly Detection System Overview

Anomaly detection system plays a critical role in identifying unusual patterns of communication, consequently improving the security of the drone. Figure 2.5 describes the design workflow of the proposed anomaly detection system for the Tello drone, which consists of three main components: data collection, feature extraction and selection, and algorithms selection and anomaly detector formulation.

Figure 2.5: Anomaly Detection System design flow chart.

Data collection: To address the absence of a suitable dataset for the Tello drone, attack vectors based on the vulnerability analysis in Section 2.4 are executed to create a comprehensive dataset containing both anomalous and normal data samples. Details are provided in Chapter 3.

Feature extraction and selection: Following data collection, which captures raw data from communications, various feature extraction scripts are utilized to accurately represent this data. The most robust version is selected for detailed analysis. Information on the technology and procedures used is outlined in Chapter 4.

Algorithms selection and anomaly detector formulation: Three ML-based anomaly detection algorithms are evaluated to determine the optimal one for this study. The selection process and detailed performance evaluations are discussed in Chapter 5.

Once the optimal anomaly detection algorithm and the most suitable feature set are determined, the anomaly detection system for the Tello drone will be complete. Its performance analysis will then demonstrate the system's effectiveness in improving the drone's cybersecurity.

Chapter 3: Data Collection

To model the anomaly detector, a dataset serves as the foundation for training, validating, and evaluating the detectors. In this project, it is crucial to have a comprehensive and high-quality dataset because the detector will be trained with normal data, and then validated and tested using a mix of anomalous and normal data to assess its effectiveness in enhancing the drone's cybersecurity. Due to the lack of an existing dataset suitable for the Tello drone, this chapter demonstrates nine types of attack vectors s based on the vulnerabilities analysis in Section 2.4, along with their results. This chapter details the implementation of each penetration test and regular operation, as well as the data capturing process.

3.1 Attack Vectors

Nine types of attacks are considered to enrich the diversity of the anomalous dataset for the detection system's evaluation process. The tools and devices used during the tests are stated below:

- DJI Edu Tello drone
- Tello mobile app
- Tello SDK
- Python programming language
- D-Link AC1200 MU-MIMO Wi-Fi USB Adapter
- iPhone 12 Pro
- Windows 10 PC with following tools:
    - Angry IP Scanner
    - MuMu Player Android Emulator
- Kali Linux with following tools:
    - Nmap
    - Hping3
    - Arpspoof
    - FFmpeg

- Wireshark
- Hulk DoS scripts [16]
- Slowhttptest
- Slowloris
- Aireplay-ng
- Netcat
- Djitellopy
- Scapy

3.1.1 Password Cracking

This attack involves a series of anomalous actions to crack the drone's Wi-Fi password set by the users. A deauthentication frame attack is initially performed to force any devices connected to the drone's network to lose their connections. Subsequently, the user/controller will try to re-authenticate to the drone by sending packets containing the hashed Wi-Fi Protected Access (WPA) key [6]. By capturing these communication packets through adapter monitor mode, hackers can crack the password offline. In this test, the deauthentication attack and re-connection packets are collected as the anomalous data.

3.1.1.1 Method

1. **Enable Monitor Mode**: **airmon-ng start wlan0** enables the monitor mode of the wireless adapter interface wlan0.
2. **Scan Network Information**: Use **airodump-ng wlan0,** to display information about all the wireless networks within range, as shown in Figure 3.1.

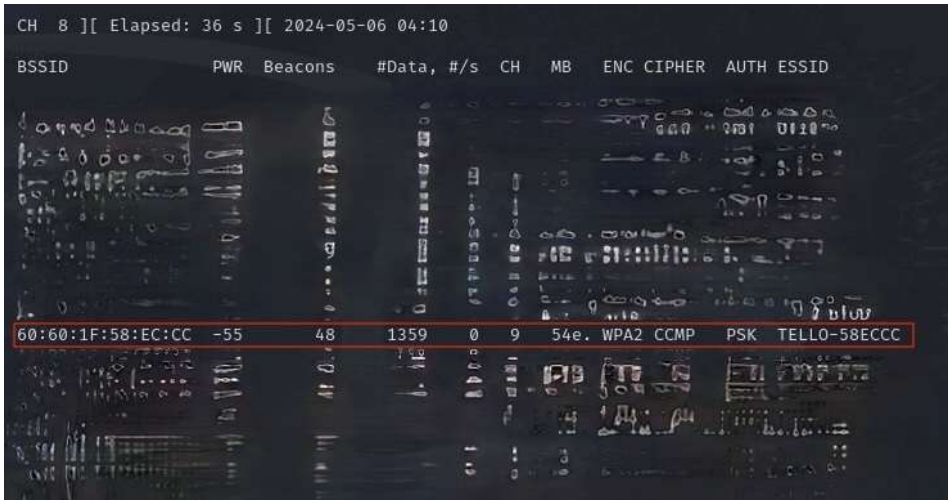

Figure 3.1: Network Information.

3. **Set the Channel**: Use **sudo iwconfig wlan0 channel [channel_number]** to set the specific channel obtained from step 2 for the interface wlan0.
4. **Wireshark Capture**: Launch the Wireshark application, and select the monitor mode interface of the adapter, wlan0, start capturing.
5. **Perform the Attack**: Use **sudo aireplay-ng -0 3 -a 60:60:1F:58:EC:CC wlan0.** The -0 option specifies a deauthentication attack, 3 indicates the number of frames to be sent, followed by the target BSSID obtained in step 2, using the network interface wlan0.

```
┌──(kali㉿kali)-[~]
└─$ sudo aireplay-ng -0 3 -a 60:60:1F:58:EC:CC wlan0
21:11:15  Waiting for beacon frame (BSSID: 60:60:1F:58:EC:CC) on channel 9
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
21:11:16  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:58:EC:CC]
21:11:16  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:58:EC:CC]
21:11:17  Sending DeAuth (code 7) to broadcast -- BSSID: [60:60:1F:58:EC:CC]
```

Figure 3.2: Deauthentication Attack.

6. **Re-authentication**: If the controller does not automatically reconnect to the drone's network, manually join its Wi-Fi network.
7. **Verification:** In Wireshark app, check if there are deauth frames and authentication packets captured, as seen in Figure 3.3.

```
107… 2121.492626   SzDjiTechnol_58:ec…  Broadcast           802.11     38 Deauthentication, SN=299, FN=0, Flags
107… 2121.493163   SzDjiTechnol_58:ec…  5e:9d:58:26:67:48   EAPOL     175 Key (Message 1 of 4)
107… 2121.493810                                           802.11     87 Unknown protocol version: 2[Malformed
107… 2121.495047   SzDjiTechnol_58:ec…  Broadcast           802.11     38 Deauthentication, SN=300, FN=0, Flags
107… 2121.495103   SzDjiTechnol_58:ec…  5e:9d:58:26:67:48   EAPOL     175 Key (Message 1 of 4)
```

Figure 3.3: Password Crack Samples.

8. **Repetition**: Continue repeating steps 4 and 5 for one hour to collect password crack data.

3.1.1.2 Results

This attack instance was successful as every step worked as expected. The deauthentication and re-authentication packets were collected. While performing step 4, the connection between the drone and controller was disconnected. Sometimes, the connection between these two devices automatically reconnected, but other times it required manual reconnection.

3.1.2 Denial of Service Attack

With the fixed ports used in the drone, DoS attacks become feasible for overwhelming the device and cutting off the connection between the drone and the mobile controller. In this attack instance, five different types of DoS attacks were conducted: TCP flood, Slowloris, SlowHTTPTest, and Hulk targeting TCP port 9999, and e UDP flood targeting port 8889.

3.1.2.1 Method

1. **Identify the Target Ports**: From Section 2.2, we know that port 8889 is used for UDP packet communication. The command **sudo nmap -O 192.168.10.1** was used to scan for any open service ports on the drone. As shown in Figure 3.4, there was a service, Abyss, running on port 9999 using TCP.

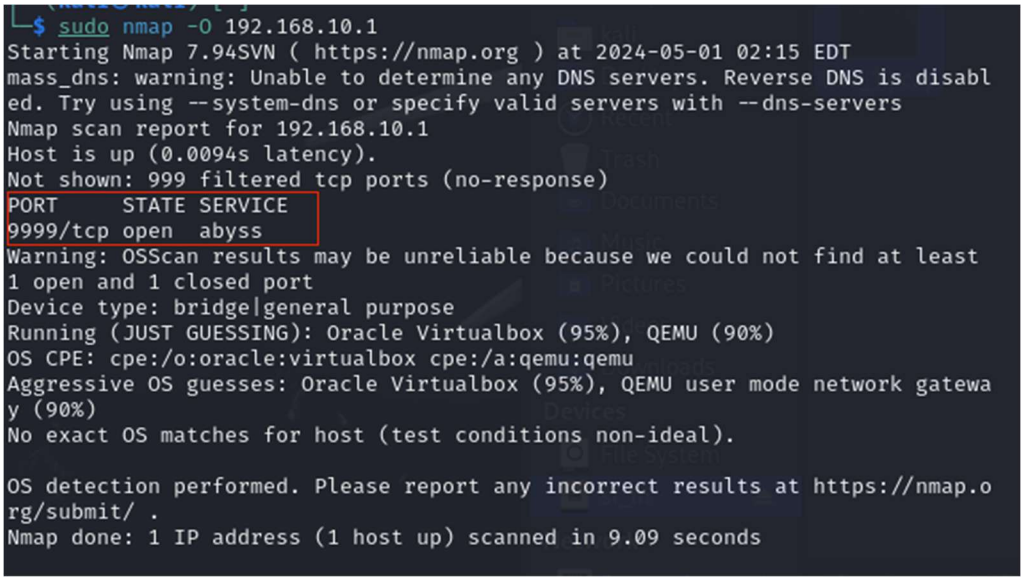

Figure 3.4: Nmap scan.

2. **Wireshark Capture**: Launch the Wireshark application, select the interface of the adapter wlan0, and start capturing.
3. **Perform TCP Flood**: The command **sudo hping3 -S -c 10000 -d 120 -p 9999 --flood 192.168.10.1** initiates a TCP flood to the target port 9999 at the drone's IP address. It sends 10,000 packets, each containing 120 bytes.

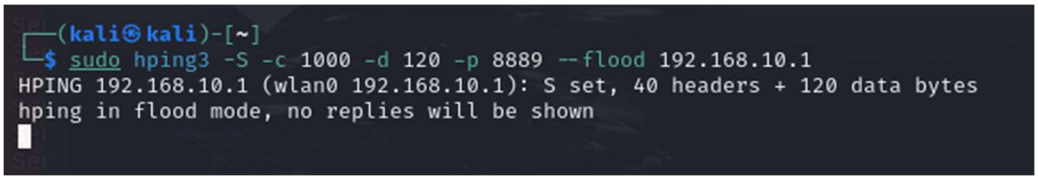

Figure 3.5: TCP Flood Sample.

4. **Perform Slowloris Attack**: **sudo slowloris -s 500 -p 9999 192.168.10.1 -v** to launch a Slowloris attack on the drone's IP and port 9999 to send 500 sockets each time to tie up the server resources.
5. **Perform SlowHTTPTest Attack**: **sudo slowhttptest -c 1000 -H -i 10 -r 5 -t GET -u http://192.168.10.1:9999 -x 24** is for a SlowHTTPTest attack, which establishes 1000 connections and sends HTTP GET requests to exhaust server resources.
6. **Perform Hulk Attack**: Download the Hulk attack source code by using the command **git clone** **https://github.com/grafov/hulk.git** [16], change to the 'hulk' directory, and launch the attack with the command **sudo python2 hulk.py HTTP://192.168.10.1:9999/**.

7. **Perform UDP Flood**: Use the command **sudo hping3 --flood --udp -p 8889 --data <packet size> 192.168.10.1** to conduct a UDP flood on the target port 8889 with varying packet sizes.
8. **Repetition**: Once the drone fails to respond to the controller, proceed to reboot it for reconnection, and continue repeating steps 3 to 7 for each attack continuously for one hour.

3.1.2.2 Results

All five DoS attacks performed as expected each with varying impacts on the drone. The TCP flood and Hulk attacks quickly disrupted the connection between the drone and the controller, while SlowHTTPTest and Slowloris were relatively slower but still managed to disconnect the connection. Once disconnected, the controller could not reconnect to the drone unless the drone was rebooted. Additionally, upon losing connection, the drone did not initiate emergency landing but continued to fly in the air. The UDP flood attack, although less aggressive than the TCP DoS attacks, caused the drone to be less responsive, with delayed responses to controller commands. Furthermore, the video feed was disabled for most of the operating time. However, users could still control the drone.

Since the DoS attacks caused the connection to be terminated, I had to reboot the device, reconnect, and then perform the attack again, which was considered as one round. This process was repeated to collect data for each round, with each DoS attack type lasting for one hour.

3.1.3 Man-In-The-Middle Attack

As a Wi-Fi security vulnerability, ARP spoofing is a potential threat for drones. ARP spoofing is a popular MITM (Man-in-the-Middle) attack which is used to redirect communications between communicating parties without their knowledge. In this project, two types of MITM attacks were performed. The drone, served as an access point (AP), has a client isolation mechanism preventing devices on its network from reaching each other, which limited the impact of the MITM attack. However, as noted in Section 2.1, the Tello SDK supports station mode, allowing the drone to join an existing network and

enabling the full performance of an MITM attack. Therefore, this section contains two MITM attacks: performing the attack under the drone's network with limited impact and using the SDK with station mode to perform a full MITM attack.

3.1.3.1 Method

1. **Enable IP Forwarding**: Use the command **echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward** to allow the system to forward packets from one network interface to another.
2. **Enable Station Mode (full MITM only)**: Use SDK script [14] and entered **ap SSID_name password** to configure the Wi-Fi network settings for the drone.
3. **Determine the Drone's IP**: By default, the drone's IP is 192.168.10.1 when using its own network. In station mode, the drone's IP is determined using the Windows application Angry IP Scanner. In this case, the drone's IP is shown as 192.168.0.104 in Figure 3.6.

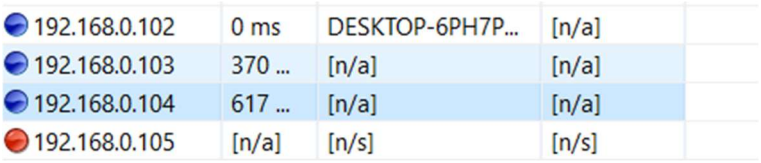

Figure 3.6: Angry IP Scanner for Drone IP.

4. **Determine the controller's IP**: By default, the mobile controller's IP was 192.168.10.2. For a full MITM attack using the SDK, the controller was a Windows PC with the IP 192.168.0.102.
5. **Wireshark Capture**: Launch the Wireshark application, select the interface of the adapter wlan0, and start capturing.
6. **Perform MITM Attack**: Open two terminals and run the commands on the attack machine, for example:
    - **sudo arpspoof -i wlan0 -t 192.168.0.102 192.168.0.104**
    - **sudo arpspoof -i wlan0 -t 192.168.0.104 192.168.0.102**

    Use the controller IP and drone IP from steps 2 and 3 accordingly.

3.1.3.2 Results

Both types of MITM attacks were successful and were able to receive and capture packets from the devices. When using the drone's network, only one-directional packets from the drone to the controller were captured. However, when using the existing network, the attack machine could capture bidirectional packets. Examples of these packets are shown in Figures 3.7 and 3.8. For data collection, each type of attack was conducted for one hour to collect anomalous data.

8	0.003841	192.168.10.1	192.168.10.3	UDP	1502 62513 → 7797 Len=1460
9	0.004007	192.168.10.1	192.168.10.3	UDP	77 8889 → 7777 Len=35
10	0.121018	192.168.10.1	192.168.10.3	UDP	1502 62513 → 7797 Len=1460
11	0.121103	192.168.10.1	192.168.10.3	UDP	1502 62513 → 7797 Len=1460
12	0.121659	192.168.10.1	192.168.10.3	UDP	1502 62513 → 7797 Len=1460
13	0.122230	192.168.10.1	192.168.10.3	UDP	1502 62513 → 7797 Len=1460

Figure 3.7: Sample packets for MITM attack using drone's network.

76.468359	192.168.0.102	192.168.0.104	UDP	50	9000 → 8889 Len=8
76.468380	192.168.0.102	192.168.0.104	UDP	50	9000 → 8889 Len=8
76.468408	192.168.0.104	192.168.0.102	UDP	201	8889 → 8890 Len=159
76.468419	192.168.0.104	192.168.0.102	UDP	201	8889 → 8890 Len=159

Figure 3.8: Sample packets for MITM attack using existing network.

During the MITM attack, the controller could still control the drone, but there was a notable delay for the attack using drone's network and the camera was disabled. When using SDK station mode, the **streamon** command, which is designed to enable video stream mode, was also invalid.

3.1.4 Inject Instructions

This attack vector involved examining multiple connection vulnerabilities and utilizing the SDK Python script with predefined anomalous actions to manipulate the drone while it was connected to the mobile controller by the user.

3.1.4.1 Method

1. **Script Implementation**: Use djitellopy library to implement a python script including any actions. The following actions were used in the collected dataset:

take off, move forward, move backward, rotate clockwise, rotate counterclockwise, and request battery information.

2. **Wireshark Capture**: Launch the Wireshark application, select the interface of the adapter wlan0, and start capturing.
3. **Script Execution**: while the attack machine was connected to the drone's network, run the script from step 1(djitello_inject.py).

3.1.4.2 Results

During this test, anomalous data from one hour of injected instruction attacks were collected. These attacks enabled a connection between the attack machine and the drone, while the drone maintained its connection with the mobile controller but lost the video feed. In this scenario, both the attack machine and the mobile controller were able to control the drone.

3.1.5 Video Interception

This attack vector is based on the inject instruction attack, which exploits the vulnerability of multiple connections. Instead of controlling the drone, this test aims to obtain the private data of the video image by intercepting the video feed packets of the drone while it is controlled by the user.

3.1.5.1 Method

1. **Script Implementation**: Use djitellopy library to implement a python script to send **streamon** command to the drone.
2. **Wireshark Capture**: Launch the Wireshark application, select the interface of the adapter wlan0, and start capturing.
3. **Script Execution**: while the attack machine was connected to the drone's network, run the script from step 1(djitello_video.py).
4. **Video Stream Interception**: During the data capture process, run the command **ffplay -probesize 32 -i udp://@:11111 -framerate 30** to play the intercepted video stream of the drone.

3.1.5.2 Results

As shown in Figure 3.9, the attack machine successfully intercepted the video packets. By executing the command in step 4, the video stream was played on the attack machine. Consequently, the test conclusively demonstrated that this method could lead to privacy leakage. While the video was intercepted, the mobile controller lost the video image. In this test, one hour of video interception attacks were conducted, capturing all video feed packets.

Figure 3.9: Intercepted video image.

3.1.6 Replay Attack

This attack vector exploits the vulnerability of unencrypted communications by capturing command packets sent from the mobile controller to the drone using adapter monitor mode. Then, it uses Python and the Scapy library to send the recorded commands and attempts to replay the user's actions.

3.1.6.1 Method

1. **Enable Monitor Mode**: **airmon-ng start wlan0** enables the monitor mode of the wireless adapter interface wlan0.

2. **Scan the Channel**: **sudo airodump-ng wlan0** is the command to display information about all the wireless networks within range, including the channel, as shown in Figure 3.1.
3. **Set the Channel**: Use **sudo iwconfig wlan0 channel [channel_number]** to set the specific channel obtained from step 2 for the interface wlan0.
4. **Wireshark Capture**: Launch the Wireshark application, select 'wlan0' as the interface for the adapter, and start capturing the user's actions. During the capture process, use the filter condition ip.src==192.168.10.2 to isolate user commands.
5. **Save Captured pcap File**: Export the captured command packets using **Export Specified Packets** option in Wireshark
6. **Enable Managed Mode**: Execute the following command:
    - **sudo ifconfig wlan0 down**
    - **sudo iwconfig wlan0 mode managed**
    - **sudo ifconfig wlan0 up**
7. **Wireshark Capture**: Begin capturing the replay attack packets on the attack machine.
8. **Script Execution**: Use the Scapy library to implement a Python script to send command packets with captured payloads (replay.py).

3.1.6.2 Results

This attack instance also successfully produced the replay attack data, which was collected for one hour. It can duplicate the actions operated by the user. During the replay attack, the victim's mobile controller was connected to the drone. Without any further actions from the users, the drone followed the commands from the attack machine. All functionalities, including the video stream, altitude, speed, and operation commands, were just as normal. As discussed in section 2.2, each type of packet operates independently. This was demonstrated in the test, where the attack machine successfully sent anomaly replay action packets to control the drone. Simultaneously, the drone independently transmitted video streams and state update packets to the victim controller, operating under the assumption that these commands were originating from the mobile controller.

3.1.7 Payload Manipulation

This attack vector builds on the replay attack discussed in Section 3.1.6, but it delves deeper into the packet structure. Instead of replaying captured actions, which are limited by user operations, this attack aims to modify the payload of the normally captured packets to send anomalous commands that are not initiated by the user.

3.1.7.1 Method

1. **Obtain the Normal Packets Structure**: Follow the steps 1 to 5 in the Section 3.1.6.1, regardless of the action commands in the packets.
2. **Obtain the Anomalous Packets Payload**: Follow the steps 4 to 5 in the Section 3.1.6.1 to record any anomalous actions.
3. **Wireshark Capture**: Begin capturing the payload manipulation attack packets on the attack machine.
4. **Script Execution**: Use the Scapy library to implement a Python script to send normal command packets with modified payload (payload_mani.py).

3.1.7.2 Results

Even though the payload was hashed, this attack successfully modified the hashed payload and sent anomalous command actions using a standard packet structure, including proper port numbers, and correct source and destination IPs, pretending to be the victim mobile controller. This attack demonstrates the significant impact of vulnerabilities stemming from unencrypted communications. Such impact ranges from enabling replaying user actions to potentially giving hackers advanced abilities to control the drone. One hour of payload manipulation attack data was collected as anomalous data.

3.1.8 Unauthorized UDP Packets

In the previous sections, all the anomalous packets were sent to the drone via either a predefined library or an SDK. Consequently, the performance of the attacks could be influenced by the security robustness of these tools. This section aims to examine the vulnerabilities associated with unencrypted UDP packets, independent of any libraries or

SDKs. We utilize **echo** and **netcat** to directly send anomalous packets to the drone, bypassing any complex software solutions.

3.1.8.1 Method

1. **Wireshark Capture**: Begin capturing the unauthorized UDP packets on the attack machine.
2. **Send UDP packet**: Use the command **echo -n "command" | nc -u 192.168.10.1 8889** to send any text-based commands directly to the drone port 8889 via terminal.

3.1.8.2 Results

This attack enables straightforward, immediate communication without intermediaries. By using just the **echo** and **netcat** tools, the drone can be controlled as expected. In this scenario, the drone was configured to send state update packets to the attack machines because this type of attack utilized the IP address of the attack machine instead of the victim controller's IP. Consequently, one hour of unauthorized packet data was collected.

3.1.9 IP Spoofing

IP spoofing capitalizes on the absence of authentication in network packets. By using a fake IP address not recognized by the drone's network tables, attackers can send anomalous packets to gain unauthorized control over the device.

3.1.9.1 Method

1. **Obtain the Anomalous Packets Payload**: Refer to Section 3.1.7.1, Step 2, to capture any actions that are to be performed as unauthorized control.
2. **Wireshark Capture**: Begin capturing the IP Spoofing attack packets on the attack machine.
3. **Script Execution**: Use the Scapy library to implement a Python script to send unauthorized packets with a spoofed IP address, such as 192.168.10.18 (IP_spoofing_attack.py).

3.1.9.2 Results

While using a fake IP address with a text-based command as the payload of the packets, the IP spoofing successfully controlled the drone without any authentication process. Consequently, the drone and mobile controller were disconnected, and the victim controller could not reconnect to the drone until the device was rebooted. In the captured one-hour data, only command packets from the attack machine to the drone were visible. This occurred because the drone considered the fake IP as the legitimate controller and sent all informative packets to that fake IP address.

3.2 Legitimate Operation

The benign data serves as the primary source for training the anomaly detector in subsequent stages. In this section, legitimate operations take place to produce normal data. Since the iPhone prevents application packets from being captured by Wireshark, and although the adapter in monitor mode was able to capture the communication packets, it was challenging to export any reassembled video stream packets for further analysis, as shown in Figure 3.10. Therefore, the solution in this section is to use an Android emulator on a Windows PC to simulate the eligible controller environment. Referring to the action space from Section 2.1, the collected normal data includes a comprehensive set of command packets.

10	0.028735	192.168.10.1	192.168.10.2	UDP	524	62512 → 7797 Len=1460
11	0.029407	SzDjiTechnol_58:ec…	5e:9d:58:26:67:48	802.11	588	Fragmented IEEE 802.11 frame
12	0.029413	SzDjiTechnol_58:ec…	5e:9d:58:26:67:48	802.11	588	Fragmented IEEE 802.11 frame
13	0.029419	192.168.10.1	192.168.10.2	UDP	524	62512 → 7797 Len=1460
14	0.029606	SzDjiTechnol_58:ec…	5e:9d:58:26:67:48	802.11	588	Fragmented IEEE 802.11 frame
15	0.029776	SzDjiTechnol_58:ec…	5e:9d:58:26:67:48	802.11	588	Fragmented IEEE 802.11 frame

Figure 3.10: Reassembled Video Packets Captured in Monitor Mode.

To validate the data collection strategy with the Android emulator, the packets captured from monitor mode serve as the baseline. The examination focuses on the video packets, as they are the only ones fragmented due to their large payloads. When looking into the packet structure shown in Figures 3.11 and 3.12, the packets from monitor mode and the Android emulator have the common sections: Frame, IP, UDP, and Data. However,

monitor mode provides additional sections: Radiotap Header, 802.11 radio information, IEEE 802.11 QoS Data, and Logical-Link Control related to Wi-Fi communication, while the Android emulator contains one distinct section: Ethernet.

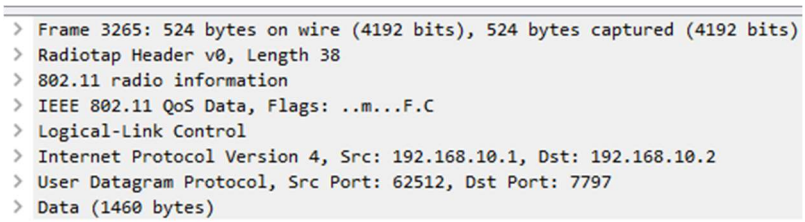

Figure 3.11: Packet View from iPhone in Monitor Mode.

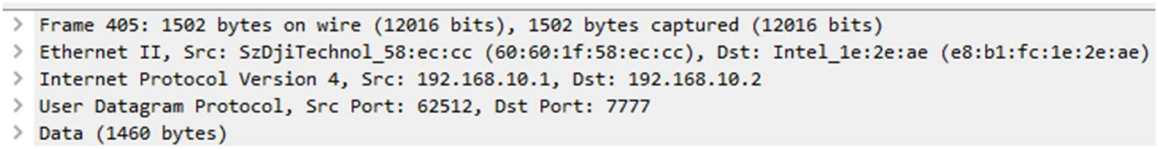

Figure 3.12: Packet View from Android Emulator.

In this project, the IEEE 802.11 and Ethernet sections provide identical MAC address information, which is the only information extracted from the additional sections, as shown in Figure 3.13.

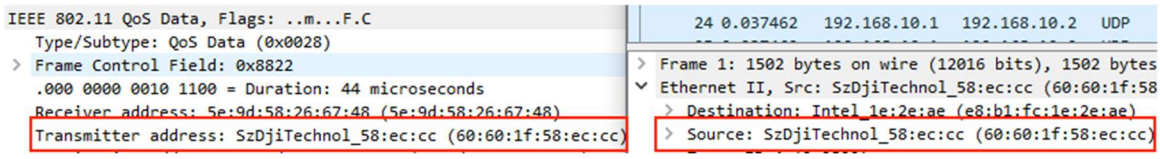

Figure 3.13: MAC Address from Both Packet Views.

When delving into the common sections, Figure 3.14 shows that the detailed information of the IP sections is identical for both packet views. Figure 3.15 demonstrates that the UDP sections are also similar between the two types of packets, except for the destination ports. However, in this project, the destination port is disregarded to prevent overfitting to one specific controller, as mobile controllers with different models and operating systems could use different ports to communicate with

the drone. The payload size for both types of packets is shown in Figure 3.11, which is 1460 bytes.

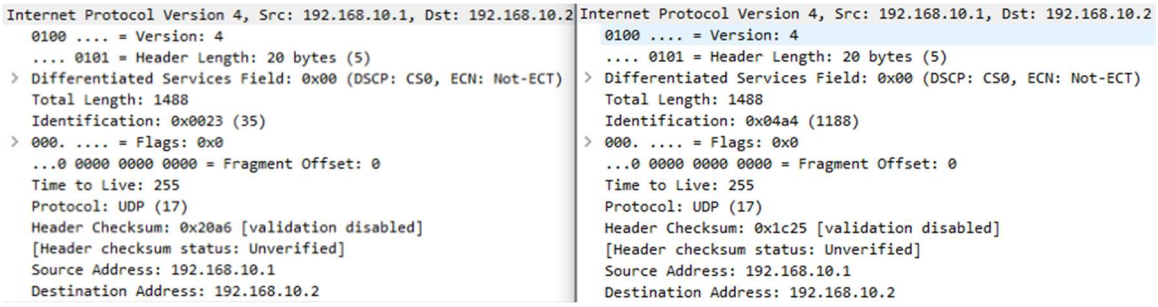

Figure 3.14: IP Sections from Both Packet Views.

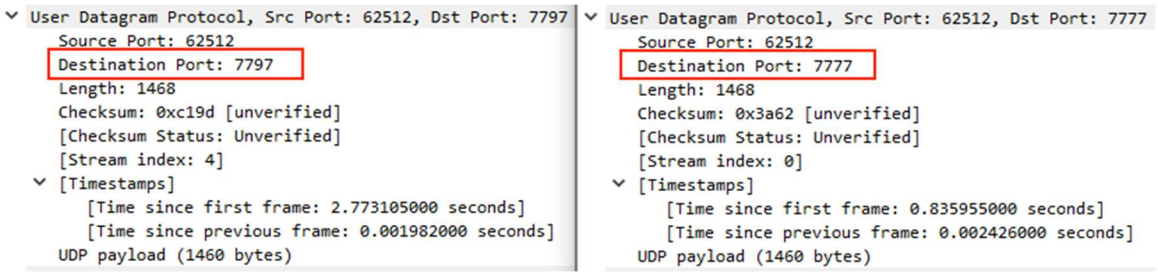

Figure 3.15: UDP Sections from Both Packet Views.

Therefore, the Android emulator is validated as the solution for collecting the regular data.

3.2.1 Method

1. **Emulator Installation**: Install the MuMu Player Android emulator from the website.
2. **Tello Mobile App Installation**: In the emulator, download and install the Tello App.
3. **Configuration**: Connect the PC to the drone.
4. **Wireshark Capture**: Begin capturing the regular packets on the PC machine.
5. **Actions**: Perform any proper actions as displayed on the user interface: takeoff, land, up, down, rotate clockwise, rotate counterclockwise, forward, back, left,

right, take photo, flight speed adjustment and modes: Bounce mode, 8D flips, Throw & Go, Up & Away, 360, Circle.

3.2.2 Results

The benign data collection was straightforward. This process continued for 10 hours to produce sufficient data for the anomaly detection formulation process.

As a result, we have accumulated 14 hours of anomalous data and 10 hours of normal data. All the captured data is considered raw data in the PCAP format, which is ready to be extracted into features representing each packet for analysis. This comprehensive data serves as the foundation for the next phase of our project, where we apply machine learning techniques to develop our anomaly detection system.

Chapter 4: Feature Extraction and Selection

This chapter will demonstrate the feature extraction and selection mechanisms.

4.1 Feature Extraction

In this section, we primarily use feature extraction and feature engineering techniques to extract and derive features from the raw data obtained in Chapter 3, ensuring the original packet information is preserved. Two feature sets have been generated in this Chapter: the original feature set extracted using scripts from CIC IoT research for large-scale anomalous data [7], and a variant specifically proposed to suit this project.

4.1.1 CIC IoT Feature Set

The feature set for the CIC IoT dataset 2023 project, which includes 33 attacks across 105 IoT devices, was developed using Python scripts as outlined in Section 1.2, Related Work. This set comprises a total of 62 features, which can be grouped into five distinct categories:

1. Basic Flow Metrics
2. Protocol Usage
3. Flag Counts
4. Packet Counts
5. Statistical Features

This comprehensive coverage provides an exhaustive representation of the raw packets, and this set is defined as the baseline feature set for comparison.

4.1.2 Proposed Feature Set

The proposed feature set is generated based on a variant of the original scripts to improve the robustness of the feature set and suit this project. Specifically, a total of 13 features have been either introduced, removed, or modified compared to the original ones, designed to enhance the performance of the original feature set:

New features:

1. **Payload Length**: Serves as the primary feature for each packet, focusing only on the data payload. This provides a clear measure of the actual data transmitted in each packet.
2. **Drone_port**: Fixed communication ports on the drone are crucial features, making them reliable indicators of communication patterns.
3. **Entropy**: This feature is used to quantify the randomness of the payload in packets. Each byte of the payload is counted, and the frequency of each byte is used to compute the probability distribution. The entropy is then calculated based on this distribution.
4. **Variance of Payload**: The measure is calculated by using the payload length to assess the variability in a batch of 10 packets. This feature helps detect anomalies by analyzing deviations from the expected distribution of payload sizes.

Removal:

1. **Flow_duration**: This is subject to the size of the raw data file, which varies with uncertainty.
2. **Header_length**: This feature contains limited information and can be implicitly represented by combining the payload and total length of the packets.
3. **MAC**: It is a fixed identifier that remains static and offers limited information into communication behavior for anomaly detection.
4. **Protocol Version**: The protocol version feature was removed from the original dataset due to its lack of variance, as it contained only IPv4 or IPv6 values.

Modification:

1. **DS Status**: Complete the functions to extract this feature, as the original script incorrectly uses BSSID as the value.

2. **Drate and Srate**: Fix the logic error in the functions to properly extract these two features, as a logic error causes Drate to always be 0, and directs all the information to Srate, making them incorrect.
3. **Rate**: This feature represents the payload data rate of each packet, rather than the flow rate in original set, which is subject to uncertainty due to the file size.
4. **Inter-Arrival Time**: The IAT has been adjusted to reflect the difference between two consecutive packets.

Both versions of the scripts were run in the Kali Linux environment. The datasets include packet-level and flow-level features. Packet-level features are extracted directly from each packet, while flow-level features are updated with each new packet that arrives. Each time a new packet is received, the flow features are recalculated and then combined with the packet to create a new row of features.

Once all the packets are processed, summarization techniques are applied to condense information from a batch of ten rows into a single sample. This process reduces data volume, improves computational efficiency, and enhances robustness by providing an aggregated view. To achieve this, statistical measures including the mode, mean, and variance of each batch are utilized.

As a result, all the raw data from the PCAP files were converted into CSV files for both datasets, each containing 1.2 million normal samples and 1.6 million anomalous samples.

4.2 Feature Sets Selection

After the datasets are generated, the two main subsections in this stage are algorithm selection and performance evaluation. They aim to assess the robustness and performance of the two feature sets and choose the optimal one for the anomaly detection system. The performance evaluation results from this section are crucial and will also be referenced for model selection process in Chapter 5.

The dataset is split into three parts: 60% of the normal data is used for training each model; 20% of the normal data combined with 20% of the anomalous data is used for the validation set; and the remaining 20% of the normal data along with 80% of the anomalous data is used for the test set. In this section, only the training and validation sets were used to formulate models and validate their performances. Normal data are labeled as negative class (-1), while anomalous data are labeled as positive class (1).

4.2.1 Algorithm Selection

The performance of the feature sets is assessed using trained anomaly detection models, making algorithm selection a priority for this process. Three algorithms: Isolation Forest (IF), Elliptic Envelope (EE), and Local Outlier Factor (LOF), are used for anomaly detection model formulation based on distinct mechanisms. All these algorithms are suitable for medium-scale datasets because of their time complexity.

- **Isolation Forest**: The Isolation Forest algorithm exhibits a time complexity of O($t*\psi*log\ \psi$) for training and O($n*t*log\ \psi$) for evaluation, where t is the number of trees, ψ is the sub-sampling size, and n is the dataset size. It detects anomalies by recursively isolating data points using random splits [17].
- **Elliptic Envelope**: During the initial experiment, this method's training process completed in about 6 minutes, and the testing process finished within 2 seconds in the project development environment, which is considered efficient. This method detects outliers by fitting an ellipse to normally distributed data [18].
- **Local Outlier Factor**: The time complexity of the LOF algorithm is O(n^2) in the worst case, which is still acceptable for this project. LOF identifies anomalies based on density deviations within the dataset [19].

4.2.2 Performance Evaluation

The performance evaluation consists of two main components: examining the performance of the feature sets through horizontal comparison and evaluating their

robustness through vertical comparison. This section will detail these two types of comparison along with the performance evaluation results and analysis.

4.2.2.1 Horizontal Comparison

In this comparison strategy, all three algorithms were used to train anomaly models using both feature sets separately. In this way, six models were generated, and their performance was evaluated horizontally, with the same algorithms and hyperparameters but different feature sets, based on the following performance metrics:

- **Precision**: $\frac{TP}{TP+FP}$
- **Recall**: $\frac{TP}{TP+FN}$
- **F1-Score**: $2 * \frac{Precision*Recall}{Precision+Rec}$
- **Accuracy**: $\frac{TP+T}{TP+TN+FP+F}$
- **Cross-Validation Accuracy Score**: Use **k** training subsets to evaluate the accuracy of the model.

where True Positive (TP) is the number of positive samples correctly classified as positive; True Negative (TN) is the number of negative samples correctly classified as negative; False Positive (FP) is the number of negative samples incorrectly classified as positive; False Negative (FN) is the number of positive samples incorrectly classified as negative.

4.2.2.2 Vertical Comparison

Vertical comparison aims to evaluate the robustness of each feature set. To achieve this, each model has two variants of its performance measures compared to their original performance:

- **Robustness to Noise**: It introduces noise to the validation set and evaluates the model performance [20].

- **Drifting Test**: It introduces a fixed level of drift to the data distribution and evaluates the model's performance to assess its adaptability to changing data features [21].

4.2.2.3 Performance Evaluation Results and Analysis

Tables 4.1 and 4.2 show the performance results for a total of six models in terms of anomalous data. Table 4.1 displays the performance of the three models with baseline feature sets, while Table 4.2 shows the performance using proposed feature sets. As an anomaly detector, the most important task is to mitigate false positives (FP) and false negatives (FN), which indicate failures in detecting normal and anomalous actions. Section 4.2.2.1 explains that precision is influenced by FP, while recall is influenced by FN.

For horizontal comparison, the IF with baseline feature set has slightly higher performance, around 1% to 3% better, in terms of recall, F1 score, and accuracy compared to the model with the proposed feature set. However, its precision is 1% lower than that of the model with the modified feature set. For EE, both models have very similar performance across all metrics. For LOF, the proposed model outperforms the baseline, especially in Recall, F1-Score, and Accuracy.

In terms of vertical comparison for baseline models, the IF shows a 5% reduction in precision in the robustness to noise test, and a 9% reduction in the drifting test. Its F1-score also decreased by 2% in the noise test and 5% in the drifting test, while recall remained similar. The EE experienced a significant decrease in precision from 92% to 56% in the noise test and drifting test, but recall increased by 10%. Its F1-score and accuracy also decreased greatly, as shown in Table 4.1. The LOF showed a similar trend with a significant decrease in precision, F1-score, and accuracy, while recall values remained similar for both tests. It is notable that its cross-validation accuracy score was 98%, but its accuracy for the validation set is 71%, which implies that overfitting occurred during the training process.

For the proposed models, the IF is relatively robust, with less than 1% variation across all metrics. However, for the EE, there is a similar decrease as seen in the baseline model.

From Table 4.2, there is an observable decrease for the LOF in precision, F1-score, and accuracy, but it is still less significant than that of the baseline model.

As a result of the comparison, the models with baseline feature sets shows less performance and robustness; overfitting occurred for the LOF models. In contrast, the proposed feature set provided more robust and optimal performance across most tests and performance metrics. Therefore, for this project of formulating an anomaly detection system, the proposed feature set will be used in Chapter 5.

Algorithm	Performance Metric	Precision	Recall	F1-Score	Accuracy	Cross-Validation Score (Accuracy on training dataset)
Isolation Forest	Original	0.9525	0.9999	0.9756	0.9719	0.9383
	Robustness to Noise	0.9070	0.9997	0.9511	0.9423	
	Drifting Test	0.8631	0.9999	0.9264	0.9108	
Elliptic Envelope	Original	0.9203	0.8999	0.9100	0.9000	0.8999
	Robustness to Noise	0.5619	0.9999	0.7195	0.5619	
	Drifting Test	0.5619	1.0000	0.7195	0.5619	
Local Outlier Factor	Original	0.9742	0.5103	0.6698	0.7173	0.9828
	Robustness to Noise	0.6827	0.5104	0.5841	0.5916	
	Drifting Test	0.5382	0.5103	0.5239	0.4788	

Table 4.1: Algorithm Performance Comparison – Baseline Feature Set.

Algorithm	Performance Metric	Precision	Recall	F1-Score	Accuracy	Cross-Validation Score (Accuracy on training dataset)
Isolation Forest	Original	0.9643	0.9688	0.9665	0.9621	0.9531
	Robustness to Noise	0.9579	0.9736	0.9657	0.9609	
	Drifting Test	0.9524	0.9762	0.9642	0.9590	
	Original	0.9226	0.9190	0.9208	0.9106	0.9002

Elliptic Envelope	Robustness to Noise	0.5661	0.9997	0.7228	0.5667	
	Drifting Test	0.5653	0.9999	0.7223	0.5654	
Local Outlier Factor	Original	0.9848	0.9954	0.9901	0.9887	0.9796
	Robustness to Noise	0.8628	0.9953	0.9243	0.9079	
	Drifting Test	0.7212	0.9955	0.8364	0.7799	

Table 4.2: Algorithm Performance Comparison – Proposed Feature Set.

Chapter 5: Algorithms Selection and Anomaly Detector Formulation

Building on the contributions of data collection and the proposed feature extraction technique, this chapter will demonstrate the process of algorithm selection, finalize the anomaly detector, and analyze its performance.

5.1 Algorithm Selection

This section details the procedures in terms of data preprocessing, performance evaluation, and selection. Drawing on information from Tables 4.2, it provides insights into the performance of each algorithm when used with the proposed dataset. Specifically, due to their superior performance, the Isolation Forest and Local Outlier Factor methods have been selected for more detailed performance analysis in this section.

5.1.1 Data Preprocessing

After the proposed dataset was loaded, the normal and anomalous data were shuffled at the beginning to prevent bias and order effects. Subsequently, the dataset was split into training, validation, and test sets of appropriate sizes, as outlined in Section 4.2. The next step involved separating the features and labels. Then, a pipeline was utilized to normalize the features using a standard scaler, impute any missing data with a median strategy. This pipeline was applied across all datasets.

5.1.2 Performance Evaluation

To assess the best performance of the IF and LOF algorithms, tuning processes using the training and validation datasets were conducted to determine the optimal hyperparameters of the models and the optimal number of components for Principal Component Analysis (PCA). During these processes, the F1-score was selected as the key performance indicator for the models, reflecting the balance between Precision and Recall, which are critical metrics for anomaly detection, as outlined in Section 4.2.2.3.

While formulating the optimal models, performance evaluation metrics were considered, extending beyond those outlined in Section 4.2.2.1, which includes Cross-Validation, Accuracy, Precision, Recall, and F1-score. To avoid data leakage, the PCA was re-fitted to the data during the cross-validation process. Additionally, two criteria were introduced: a Learning Curves plot, which illustrates the variation in the F1-score as the model is trained with increasingly larger subsets of the training data, and the ROC AUC, which indicates the model's ability to distinguish between positive and negative classes.

5.1.3 Result and Selection

Table 5.1 displays the performance metrics of the tuned models for IF and LOF, both considered optimal in terms of F1 Score. It is observable that the metric values for IF are higher than those for LOF across all fields, especially noteworthy is the recall of 1, indicating that the IF model does not misclassify any anomalous data as normal, which is ideal for the anomaly detector.

From the cross-validation score and F1 score in Table 5.1, it is shown that there is a small variance in LOF of 0.67%, while the values for IF remain almost the same, 0.13% difference. Figures 5.1 and 5.2 also show that there was no overfitting occurring in the IF training process. However, there is a consistent small gap in the LOF training process, indicating slight overfitting.

Since both methods utilized the same training and validation datasets, it is more intuitive to display the combination of confusion matrices as Table 5.2 for comparison. The IF model did not misclassify any anomalous samples as normal and misclassified only 2490 normal samples as anomalies out of a total of 256163 negative samples. In contrast, the LOF model misclassified 2608 normal samples as anomalous and 5193 anomalous samples as normal, out of 333034 positive samples.

From the analysis, it is evident that Isolation Forest has outperformed Local Outlier Factor in all the performance evaluation metrics. It was also more robust, as no overfitting

occurred during the project. Therefore, the tuned Isolation Forest model was determined to be the final anomaly detection model for this project.

Metrics	Isolation Forest	Local Outlier Factor (LOF)
Accuracy	0.9958	0.9868
Precision	0.9926	0.9921
Recall	1.0000	0.9844
F1 Score	0.9963	0.9882
Cross_validation (F1)	0.9950	0.9949
ROC AUC	0.9951	0.9871

Table 5.1: Performance Metrics Comparison.

	Prediction (+1)	Prediction (-1)
Ground Truth (+1)	$333034_{IF}\|327841_{LOF}$	$0_{IF}\|5193_{LOF}$
Ground Truth (-1)	$2490_{IF}\|2608_{LOF}$	$253673_{IF}\|253555_{LOF}$

Table 5.2: Comparative Confusion Matrix for IF and LOF.

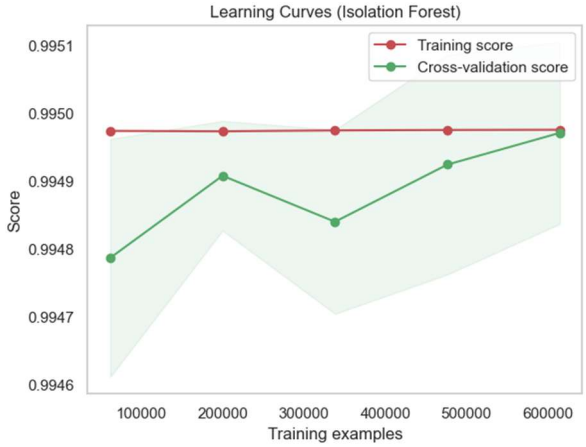

Figure 5.1: Isolation Forest Learning Curves.

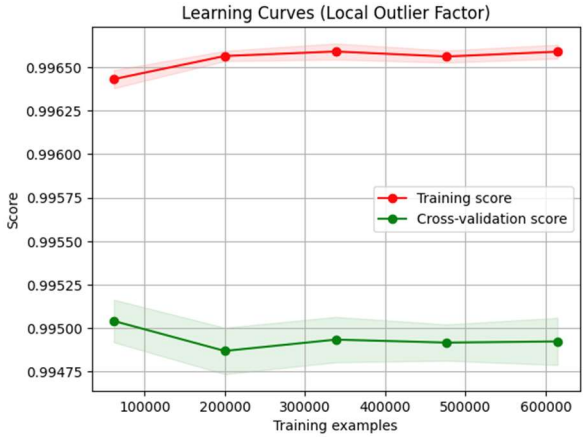

Figure 5.2: Local Outlier Factor Learning Curves.

5.2 Finalization of Anomaly Detection Model

After the tuning process was completed, both optimal models were exported. Consequently, the optimally tuned Isolation Forest model could be directly imported and used in this section. The model is named 'best_isolation_forest_50_0.75_0.01_0.5.pkl', with the optimal hyperparameters indicated in the filename. Up to this point, the anomaly detection system is complete, having achieved the objective of enhancing the cybersecurity of the Tello drone by detecting any anomalous actions. The evaluation and analysis of the system will be presented in the next section.

5.3 System Performance Result and Analysis

In this section, both the performance and robustness of the model are analyzed with the test dataset. Various performance metrics: Accuracy, Precision, Recall, F1-Score, and ROC AUC were used to evaluate the model's performance. Additionally, noise and drifting tests were conducted to assess its robustness.

Table 5.3 demonstrates that the performance of the anomaly detector on the test dataset is very close to the performance shown in Table 5.1. This similarity indicates that the model delivers optimal performance across all validation and test datasets, confirming that no overfitting has occurred. The metric values from the noise and drifting tests also show a very small influence, with a maximum variation of 0.12%, which highlights the robustness of the model. Tables 5.4 to 5.6 further demonstrate the performance for each scenario in detail. The Recall across all scenarios is 1, indicating that the model is exceptionally accurate at detecting anomalous data. This accuracy is attributed to the structure of the heartbeat packets, the high quality of the raw data, the effective features representing the raw data, and a suitable ML-based anomaly detection algorithm.

Overall, the system's performance meets the project's criteria for effectiveness and acceptability. It consistently demonstrates high efficiency and robustness in detecting anomalous actions.

	Original Performance	Robustness Noise Test	Drifting Test
Accuracy	0.9984	0.9982	0.9981
Precision	0.9982	0.9980	0.9977
Recall	1.0000	1.0000	1.0000
F1 Score	0.9991	0.9990	0.9989
ROC AUC	0.9952	0.9946	0.9940

Table 5.3: System Performance Metrics.

	Prediction (+1)	Prediction (-1)
Ground Truth (+1)	1332138	0
Ground Truth (-1)	2463	253700

Table 5.4: Confusion Matrix for IF.

	Prediction (+1)	Prediction (-1)
Ground Truth (+1)	1332138	0
Ground Truth (-1)	2784	253379

Table 5.5: Confusion Matrix for Noise Test.

	Prediction (+1)	Prediction (-1)
Ground Truth (+1)	1332138	0
Ground Truth (-1)	3056	253107

Table 5.6: Confusion Matrix for Drifting Test.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

With the increasing popularity of drones in daily life, it is crucial to be aware of their security vulnerabilities. This project successfully demonstrated the workflow for building an anomaly detection system for the Tello drone, aimed at enhancing its cybersecurity and protecting privacy.

As an anomaly detection system, its primary aim is to identify any anomalous actions apart from legitimate user activities. Thus, vulnerability analysis becomes a crucial initial step to investigate potential threats to the device, which also informs the data collection process. In this project, a diverse dataset of anomalous data was collected, including 9 distinct types of attacks, with an additional 5 sub-categories, totaling 14 types of anomalous data and approximately 1.6 million samples. Additionally, 1.2 million samples of normal data were collected to provide sufficient training, validation, and testing samples.

Once the raw data was captured, feature extraction became the next stage to properly represent the data packets. Baseline scripts from the CIC research team were utilized, and this project proposed a variant of these scripts to better suit its specific needs. Initially, three ML-based anomaly detection algorithms: Isolation Forest, Elliptic Envelope, and Local Outlier Factor, were used to train and evaluate the performance of two feature sets and each model. After a systematic horizontal and vertical comparison, the proposed feature set was proven to have better performance and robustness. Thus, it was selected for this project. Concurrently, the Isolation Forest and Local Outlier Factor methods were chosen for further evaluation due to their better performance with the training and validation datasets. The tuning process was then applied to find the optimal hyperparameters for the best performance of each model, using the training and validation sets. After comparing the optimal performance and robustness of both models, the Isolation Forest was determined to be the final model for this project. The evaluation process showed that using the proposed feature set with the optimal Isolation Forest model formulates a

highly effective anomaly detector with all metrics exceeding 99% and minimal influence from noise and drift.

Overall, this project demonstrates a comprehensive study aimed at formulating an anomaly detection system to enhance the security of a Wi-Fi-based drone. It contributes not only the Tello drone's anomalous and normal datasets but also delivers more robust feature extraction scripts and a systematic algorithm performance comparison strategy. These contributions ensure that the anomaly detection system can detect anomalous actions beyond normal activities with enhanced performance.

6.2 Future Work

Although this project has proposed a high-performance anomaly detection system capable of detecting anomalies, there is still room for improvement and opportunities to enhance its functionalities:

1. **Real-Time Detection**: Although the proposed anomaly detector performs well, it is still worthwhile to explore its deployment in a real-time environment to assess its operational effectiveness.
2. **Algorithms Investigation**: In this project, three algorithms with distinct detection mechanisms were employed. However, given that Isolation Forest demonstrated superior performance, it suggests a direction to further explore its related algorithms for potential performance improvements. Such investigations could include Extended Isolation Forest and Random Cut Forest.
3. **Dataset diversity**: While this project included a wide range of attack vectors, it is always beneficial to consider additional attack methods to further examine the detector's performance.
4. **Cross-platform applications**: From the data collection phase, it is evident that iPhones and Android devices capture packets differently. Therefore, developing a strategy for the detection system to operate across platforms could greatly benefit users.

5. **User interface development**: Now that the system's mechanisms have been developed, the next step would be to create a user interface to visualize the system's results.

These further efforts aim to integrate and enhance every component of the anomaly detection application, from initial design to final deployment. This comprehensive approach will transform the system from a basic detector into a robust commercial application. Additionally, it contributes to enriching the diversity of datasets for future research and potentially improves performance by investigating more related algorithms.

Bibliography

[1] “Drones - Worldwide | Statista Market Forecast,” Statista. https://www.statista.com/outlook/cmo/consumer-electronics/drones/worldwide

[2] C. Kumar and S. Mohanty, "Current Trends in Cyber Security for Drones," 2021 International Carnahan Conference on Security Technology (ICCST), Hatfield, United Kingdom, 2021, pp. 1-5, doi: 10.1109/ICCST49569.2021.9717376.

[3] “Wifi Drones: Everything You Need To Know,” Autelpilot, Sep. 02, 2022. https://www.autelpilot.com/blogs/news/wifi-drones-everything-you-need-to-know

[4] “Drones: The Wireless Technologies That Enable Operation and Control,” www.data-alliance.net. https://www.data-alliance.net/blog/drones-the-wireless-technologies-that-enable-operation-and-control/

[5] M. Ahsan, K. E. Nygard, R. Gomes, M. M. Chowdhury, N. Rifat, and J. F. Connolly, "Cybersecurity Threats and Their Mitigation Approaches Using Machine Learning—A Review," J. Cybersecur. Priv., vol. 2, no. 3, pp. 527-555, 2022, doi: 10.3390/jcp2030027.

[6] G. Rubbestad, “Hacking a Wi-Fi based drone,” KTH Royal Institute of Technology. https://www.diva-portal.org/smash/get/diva2:1586253/FULLTEXT01.pdf

[7] “IoT Dataset 2023 | Datasets | Research | Canadian Institute for Cybersecurity | UNB,” University of New Brunswick. https://www.unb.ca/cic/datasets/iotdataset-2023.html

[8] G. de Carvalho Bertoli, L. A. Pereira and O. Saotome, "Classification of Denial of Service Attacks on Wi-Fi-based Unmanned Aerial Vehicle," 2021 10th Latin-

American Symposium on Dependable Computing (LADC), Florianópolis, Brazil, 2021, pp. 1-6, doi: 10.1109/LADC53747.2021.9672561.

[9] Samuel Hassler, Umair Mughal, Muhammad Ismail, May 25, 2023, "Cyber-Physical Dataset for UAVs Under Normal Operations and Cyber-Attacks", IEEE Dataport, doi: https://dx.doi.org/10.21227/6f22-py65.

[10] S. C. Hassler, U. A. Mughal and M. Ismail, "Cyber-Physical Intrusion Detection System for Unmanned Aerial Vehicles," in IEEE Transactions on Intelligent Transportation Systems, vol. 25, no. 6, pp. 6106-6117, June 2024, doi: 10.1109/TITS.2023.3339728.

[11] D. Kim, Y. Song, S. Kwon, H. Kim, J. D. Yoo, and H. K. Kim, “UAVCAN Dataset Description,” arXiv (Cornell University), Jan. 2022, doi: https://doi.org/10.48550/arxiv.2212.09268.

[12] Y. Wu, L. Yang, L. Zhang, L. Nie and L. Zheng, "Intrusion Detection for Unmanned Aerial Vehicles Security: A Tiny Machine Learning Model," in IEEE Internet of Things Journal, vol. 11, no. 12, pp. 20970-20982, 15 June15, 2024, doi: 10.1109/JIOT.2024.3360231.

[13] “SDK 2.0 User Guide 2,” Ryze. 2018. https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%202.0%20User%20Guide.pdf

[14] Ryzerobotics.com, 2018. https://dl-cdn.ryzerobotics.com/downloads/tello/20180222/Tello3.py

[15] S. Pikalov, E. Azaria, S. Sonnenberg, B. Ben-Moshe, and A. Azaria, "Vision-Less Sensing for Autonomous Micro-Drones," Sensors, vol. 21, no. 16, pp. 5293, 2021, doi: 10.3390/s21165293.

[16] A. Grafov, "Hulk: HTTP Load Testing Kit," GitHub, 2019. https://github.com/grafov/hulk.git

[17] F. T. Liu, K. M. Ting and Z. -H. Zhou, "Isolation Forest," 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 2008, pp. 413-422, doi: 10.1109/ICDM.2008.17.

[18] scikit-learn, "EllipticEnvelope," https://scikit-learn.org/stable/modules/generated/sklearn.covariance.EllipticEnvelope.html

[19] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: identifying density-based local outliers,” ACM SIGMOD Record, vol. 29, no. 2, pp. 93–104, Jun. 2000, doi: https://doi.org/10.1145/335191.335388.

[20] I. Buzhinsky, A. Nerinovsky, and S. Tripakis, “Metrics and methods for robustness evaluation of neural networks with generative models,” Machine Learning, Jul. 2021, doi: https://doi.org/10.1007/s10994-021-05994-9.

[21] S. Ackerman, Eitan Farchi, O. Raz, M. Zalmanovici, and P. Dube, “Detection of data drift and outliers affecting machine learning model performance over time,” Dec. 2020, doi: https://doi.org/10.48550/arxiv.2012.09258.