MAVIDS:
An Intelligent Intrusion Detection System for Autonomous Unmanned Aerial Vehicles

by

Jason P. Whelan

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of
Master of Science in Computer Science

University of Ontario Institute of Technology (Ontario Tech University)
Oshawa, Ontario, Canada
July 2021

Thesis Examination Information

Submitted by: **Jason P. Whelan**

Master of Science in **Computer Science**

MAVIDS: An Intelligent Intrusion Detection System for Autonomous Unmanned Aerial Vehicles

An oral defense of this thesis took place on July 16, 2021 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Miguel Vargas Martin
Research Supervisor	Dr. Khalil El-Khatib
Research Co-supervisor	Dr. Abdulaziz Almehmadi
Examining Committee Member	Dr. Shahram Heydari
Examining Committee Member	Dr. Richard Pazzi
Examining Committee Member	Dr. Patrick Hung

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Unmanned Aerial Vehicles (UAVs) face a large threat landscape, being used in numerous industries in hostile environments while relying on wireless communication. As attacks against UAVs increase, an intelligent Intrusion Detection System (IDS) is needed to aid the UAV in identifying attacks. The UAV domain presents unique challenges for intelligent IDS development, primarily the variety of components, communication protocols, and dataset availability. A novelty-based approach to intrusion detection in UAVs is proposed by using one-class classifiers, exploiting the use of flight logs for training. The proposed technique is integrated into a fully developed IDS which operates onboard the UAV, allowing it to detect and mitigate attacks even when communication to the ground control station is lost. The approach shows promising results when faced with a number of common attacks, including macro averaged F1 scores of up to 90.57% and 94.3% for live GPS spoofing and jamming respectively.

Keywords: unmanned aerial vehicle; intrusion detection systems; machine learning; novelty detection; cyber-physical systems

Author's Declaration

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

_____

Jason P. Whelan

Statement of Contributions

The research work presented in this thesis was published in a number of venues.

- The work in Section 2.2 was made possible by the cooperation of SkyX Systems, in allowing us to conduct security research on their operational UAS. The results of this study were published as:

  Jason Whelan, Abdulaziz Almehmadi, Jason Braverman, and Khalil El-Khatib. "Threat Analysis of a Long Range Autonomous Unmanned Aerial System." In 2020 International Conference on Computing and Information Technology (ICCIT-1441), pp. 230-234. IEEE, 2020.

- The intrusion detection method in Chapter 3 was published as:

  Jason Whelan, Thanigajan Sangarapillai, Omar Minawi, Abdulaziz Almehmadi, and Khalil El-Khatib. "Novelty-based Intrusion Detection of Sensor Attacks on Unmanned Aerial Vehicles." In Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, pp. 23-28. 2020.

- A journal paper discussing MAVIDS has been submitted to the Journal of Computers and Electrical Engineering (Elsevier) for the special issue on Intelligent Approaches in Security and Privacy.

Acknowledgments

I would first like to thank my partner, Tasha, for her patience and motivation. I would also like to thank my supervisors, Dr. Khalil El-Khatib and Dr. Abdulaziz Almehmadi, for their continued guidance and support, and Research Assistants Thani Sangarapillai and Omar Minawi for their help in various stages during the development of the proposed system. Finally, I would like to thank SkyX Systems for allowing us to conduct a security assessment of their operational Unmanned Aerial System (UAS), and the Automotive Center of Excellence (ACE) at Ontario Tech University for assistance in carrying out live experiments.

Contents

List of Acronyms

ACE Automotive Centre of Excellence

ADAM Adaptive Moment Estimation

ADS-B Automatic Dependant Surveillance - Broadcast

AI Artificial Intelligence

API Application Programming Interface

C2 Command and Control

CIA Confidentiality Integrity Availability

CSV Comma Separated Value

DoS Denial of Service

DTL Django Template Language

ESC Electronic Speed Controller

FANET Flying Ad-Hoc Network

FGCS Field Ground Control Station

GCS Ground Control Station

GNSS Global Navigation Satellite System

GPS Global Positioning System

HGCS Home Ground Control Station

HIDS Host-based Intrusion Detection System

HITL Hardware-in-the-loop

IDS Intrusion Detection System

IMU Inertial Measurement Unit

INS Inertial Navigation System

IoT Internet of Things

IPS Intrusion Prevention System

KNN K-Nearest Neighbours

LIDAR Light Detection and Ranging

LOF Local Outlier Factor

LOS Line of Sight

LSTM Long Short-term Memory

MAC Media Access Control

MANET Mobile Ad-Hoc Network

MAVIDS MAVLink Intrusion Detection System

MEMS Micro-electro-mechanical Systems

MITM Man-in-the-middle

ML Machine Learning

MSE Mean Squared Error

NASA National Aeronautics and Space Administration

NATO North Atlantic Treaty Organization

OC-SVM One-Class Support Vector Machine

OODA Observe Orient Decide Act

PCA Principal Component Analysis

PID Proportional Integral Derivative

PSO Particle Swarm Optimization

RBF Radial Basis Function

RF Radio Frequency

ROS Robot Operating System

RTK Real Time Kinematic

RTPS Real Time Publish Subscribe

SDF Simulation Description Format

SDR Software-defined Radio

SITL Software-in-the-loop

SNIR Signal-to-noise-plus-interference Ratio

SSID Service Set Identifier

SSL Secure Sockets Layer

STANAG Standardization Agreement

SVM Support Vector Machine

TOA Time of Arrival

UAS Unmanned Aerial System

UAV Unmanned Aerial Vehicle

UAVCAN Uncomplicated Application-level Vehicular Computing and Networking

UGV Unmanned Ground Vehicle

UUV Unmanned Underwater Vehicle

VSM Vehicle Specific Module

VTOL Virtual Take-off and Landing

XML Extensible Markup Language

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The use of UAVs is increasing at a steady pace. The number of commercial UAVs in North America alone is estimated to rise to approximately 1.4 million by the year 2025 with the market reaching $4.2 billion [1]. In addition to commercial usage, there are over 40 countries currently using UAVs for military operations [2]. The increase in the number of UAVs is mirrored by the increase in the number of applications of UAVs. Modern UAVs are being used in numerous industries ranging from agriculture, law enforcement, emergency management, and military. In these industries, UAVs are usually employed to carry out surveillance, payload delivery, and mapping operations [3].

While operating in high risk environments, UAVs are often up against highly motivated and sophisticated threat actors. Additionally, UAVs rely heavily on wireless protocols and input from the environment around them in order to navigate it. This widens the threat landscape as their sensors and communications are available for interference from threat actors within the environment. As a result, attacks against

UAVs are becoming more and more prevalent and simpler to conduct. Common threats such as data link jamming, Global Positioning System (GPS) spoofing, and sensor-based injection attacks are difficult to detect and prevent. Easily accessible wireless interfaces combined with their high profile uses make autonomous UAVs high risk targets [4, 5].

UAV components and protocols vary greatly as requirements for the operation of the UAVs mission changes. Because of this, the threat landscape of the UAV can also vary greatly across different implementations. Previous works have presented a threat analysis of the typical UAV, showing the predominantly wireless threats the UAV faces [6, 7]. To combat threats against the UAV, a lightweight on-board Intrusion Detection System (IDS) is needed. Unlike other proposed solutions in the literature, having an IDS agent on-board allows for the detection and potential mitigation of cyber attacks, even when jamming causes the communication to the Ground Control Station (GCS) to become lost. This is especially important in autonomous UAVs, as it can provide the UAV with the intelligence needed to detect and mitigate threats without human interaction.

1.2 Research Motivation

UAVs are typically used in high profile operations within hostile environments, such as industrial control systems surveillance and military operations. Successful attacks on these types of UAVs can be devastating to public safety, corporate espionage, or even national security, and they are already happening around the world. In 2009, insurgents used $26 software to intercept satellite traffic from a United States Military UAV, allowing them to capture intelligence [8]. More recently in 2015, Mexican drug cartels were able to spoof GPS data sent to a United States Customs and Border Protection UAV, causing it to become inoperable [9]. In 2018, a simple GPS jamming attack to a swarm of entertainment UAVs in Hong Kong caused over $100,000 in damages [10].

Successful attacks against UAVs can have serious consequences including:

- Compromise of on-board data which could threaten national security or ongoing investigations
- Operational/tactical disadvantage resulting from loss of UAV asset
- Financial loss of UAV asset itself
- Increase attack surface with the UAV itself being used as an attack tool

Attacks against UAVs are becoming more prevalent, however, technology to defend these attacks is limited. Research in this area is becoming a trending topic, however, at the time of writing this thesis, there were not any available IDS systems specifically designed for UAVs that we were aware of. As research in this area is in its infancy, creating an open IDS based on machine learning (ML) methods can make better

security accessible to even the novice UAV developer. This IDS can also serve as a starting point to help the UAV industry introduce IDS technologies.

1.3 Research Objective

Attacks on autonomous UAVs can come from a variety of sources and can target wireless protocols, cyber-physical sensors, or flight logic. Given the nature of UAVs, there is a strong reliance on wireless protocols. For example, a typical UAV will use WiFi or RF links to communicate to the GCS for payload and telemetry data, GPS for positioning, and other wireless protocols for collision avoidance. The severity of WiFi attacks is increased as the UAV can be targeted directly [11]. Compromise of the WiFi security key allows an adversary to access the UAV over the familiar TCP/IP communications stack and conduct traditional service-based attacks. If the UAV has unsecured services open, it can lead to complete compromise [12]. Cyber-physical sensors are used to gather environmental data which is then used to make flight decisions [13]. GPS is an elementary example of a cyber-physical sensor often used by an autonomous UAV. Most commercial UAV deployments will use the public GPS system as the primary navigation system, which is not encrypted. Although widely used, GPS is vulnerable to jamming and spoofing [14]. Autonomous UAVs also face unique threats due to their autonomy. Autonomy of the UAV can be lost from a number of attacks causing it to go off-course or become hijacked. Examples of this include Automatic Dependant Surveillance Broadcast (ADS-B) spoofing and insider threats [15].

Given the array of attack vectors autonomous UAVs face, an intrusion detection system is crucial to their safety. Unfortunately, many of the threats autonomous UAVs face are behaviour-based and cannot be easily detected with traditional signature-based methods. This is because signatures must be created for each attack or individual malicious behaviour, making it maintenance-heavy and costly. In this work,

the objective is to explore the use of machine learning IDS approach to detect common attacks, as well as provide the ability to detect unknown attacks that would otherwise go undetected. This IDS will run onboard the UAV, as some of the most prominent threats to the UAV involve jamming or spoofing of communication links. If these links fail and the IDS logic lies on the GCS, for example, the IDS becomes inoperable. Running a machine learning model onboard a UAV presents a number of challenges. First, there are limitations to the components that can be added to the UAV, as weight, size, and power consumption must be minimized as to not increase the overall weight or reduce flight time. Any added component will need to be limited in processing power to meet these restrictions. Therefore, the machine learning model must be as efficient as possible, as inference on the UAV traffic must be done without causing a bottleneck. Flight data is real-time data and delays can cause a malfunction of the UAV (ie. GPS timing).

In addition to the machine learning models used to build IDS systems, the dataset used for the training and testing of the model is very important. For the research community to create meaningful IDS solutions to detect intrusions on UAVs, a relevant dataset needs to be created in order to serve as a benchmark for comparison. Datasets such as CSE-CIC-IDS2018, developed by the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC), is a labelled dataset for use in developing machine learning IDS for traditional communication networks. A similar dataset for UAV IDS development is needed and would be a contribution to the community.

This thesis aims to accomplish the following objectives:

- Creation of a dataset where attacks against the UAV are recorded for current

and future work

- Research and test a highly accurate intrusion detection technique which exploits the use of flight logs and remains effective on devices with low computational resources
- Design and development of a modular IDS which can run onboard a UAV to detect jamming attacks

1.4 Scope and Limitations

Through active research activities and a review of the literature, it was concluded that the most prominent threats to the operational UAS are those targeting the UAV itself [6]. As the UAV operates within a hostile environment, it cannot be protected by physical security as it relies on entirely wireless data links. Therefore, to reduce and define scope, the proposed intrusion detection system is limited to attacks against the UAV, and does not cover attacks on the whole UAS.

Jamming and spoofing attacks are some of the most common against UAVs. To conduct intrusion detection research, these attacks must be executed in order to investigate how to detect them. In Canada, it is illegal to import, posses, and use jamming devices [16]. This also applies to spoofing, as it involves the potential interruption of legitimate RF signals. To continue research in this area without broadcasting live signals, alternative methods can be employed such as:

- Broadcasting on alternate, public frequency bands
- Conducting live attacks within faraday cages
- Simulating the attacks in software-in-the-loop or hardware-in-the-loop environments

Unfortunately, some of these methods also enter legal grey areas. Other attacks such as command injection can be done legally, however, they can introduce safety concerns. Attacks against a UAV can cause it to act unpredictably. For these reasons, attacks conducted throughout this thesis are done both in simulation as well as through live experiments within a research facility following all legalities. For the initial research, simulations allow for rapid testing without the required resources for

live experiments. Once the intrusion detection method is developed, live experiments help to verify performance. While following these limitations in conducting attacks, the dataset may contain bias. As conducting the attacks must be done either within a simulator or an indoor and controlled environment, it will not account for environmental factors and some sensors may not be used. For example, influences on false positive rates may arise from wind, and the logs may become more "busy" with additional sensors in play.

Popular consumer and commercial UAVs such as those made by DJI and Parrot contain proprietary software, making it difficult to develop deep integrations and observe full data logging. Larger UAVs, such as those used by the military, are not accessible for research purposes. For these reasons, this thesis looks at open source UAV systems. In particular, the ecosystem developed by The Dronecode Foundation is used throughout. This ecosystem is used by UAV developers at various levels, and has seen use around the world by companies such as Intel, Yuneec, Qualcomm, Sony, and Freefly. The Dronecode ecosystem includes open source projects for each aspect of an operational UAS, including:

- PX4 - Flight controller firmware
- MAVLink - Communication protocol for UAVs and their components
- MAVSDK - Library with APIs for MAVLink integration within a number of programming languages
- QGroundControl - Ground control station for MAVLink-based UAVs

Utilizing open source software allows for a closer look at all aspects of UAV operation and allows for efficient development of compatible software.

1.5 Contributions

The UAV domain presents a number of challenges for intrusion detection such as the variety of attacks, sensors, communication protocols, UAV platforms, control configurations, and dataset availability.

This thesis aims to solve these problems through the following contributions:

- Proposes an intrusion detection technique using principal component analysis and one-class classifiers, exploiting the use of pre-existing flight logs as training data
- Presents an operational intrusion detection system with a modular design. This system can be effectively applied to a vast number of UAV systems
- Proposes the use of a lightweight IDS agent on-board the UAV, mitigating the problem of communication loss during common denial of service and jamming attacks
- Provides the research community with a UAV attack dataset to allow the community to compare performances of various developed IDS

1.6 Organization of Thesis

This thesis is divided into six chapters. Chapter 1 starts by providing a brief background into the state of UAV security, the motivation behind the thesis, overall objectives, contributions, and scope. In Chapter 2, a deeper background is provided about UAVs and their architectures, threats against UAVs, intrusion detection systems, machine learning, and a literature review on UAV IDS. Chapter 3 discusses the one-class novelty detection approach to UAV IDS as well as the MAVIDS platform. The performance results of the proposed method are discussed in Chapter 4. Finally, a conclusion is drawn and potential future work is outlined in Chapter 5.

Chapter 2

Background and Related Work

2.1 Unmanned Aerial Vehicles

An Unmanned Aerial Vehicle (UAV) is an aircraft without a pilot on board that is controlled remotely or autonomously through the help of onboard systems. The UAV is one component of the Unmanned Aerial System (UAS), which comprises the UAV itself, a ground control station (GCS), and communication links between the two. Both remotely piloted and autonomous UAVs are used across many industries to provide value while keeping humans safe. Consumer, commercial, and government sectors all benefit from UAV use. UAVs are becoming popular within the consumer world as "drones", being used primarily for amateur and professional photography and videography. Drone manufacturers such as DJI and Parrot have become household names in the consumer drone industry by making "flying cameras" widely accessible. In the commercial and government space, UAVs are often used for tasks deemed too dangerous or too difficult for humans. Being remotely piloted mitigates risk to human operators. To further decrease risk and minimize human error, many of these UAVs are deployed in operations completely autonomously. A UAV is autonomous if it

can observe, orient, decide, and act (OODA) without human interaction [17]. Based on OODA, the United States Air Force has defined ten levels of UAV autonomy [18], from Remotely Piloted to Fully Autonomous. Unlike their non-autonomous counterparts, autonomous UAVs do not have a human pilot remotely controlling the vehicle. Instead, communications management software coordinates the mission and flies the UAV in place of a human [18]. Due to this, autonomous UAVs can takeoff, avoid obstacles, complete the mission, return to base and land without any human interaction. As a result, autonomous UAVs are able to execute missions on exact schedules, with longer range, and are able to fly along predetermined routes with no error. These are feats that UAVs operated manually by humans cannot achieve with the same degree of precision.

There are four common variants of UAVs, each designed for specific tasks or to carry specific payloads: fixed-wing (possesses high cruising speeds), rotary-wing (useful for civilian applications due to the ability to vertically takeoff and hover), blimps (for long endurance, low speed flights), and flapping-wing (small sizes, but possess morphing wings to fly like planes or to perform vertical takeoff and landing (VTOL)). These variants can be further reduced into the most common sub-variants: multicopters, planes, and VTOL. Multicopters are common in the consumer and commercial industries as they have more agility but less endurance. This is ideal for tasks such as photography, short range surveillance, and search and rescue. Planes are fixed-wing UAVs with high endurance and the ability to carry larger payloads. Planes are often used in long range surveillance, military operations, and border patrol. VTOL UAVs are a hybrid between multicopters and planes, with the ability to take off vertically and hover, then transition into fixed-wing flight.

2.1.1 Architecture

Behind a UAVs ability to fly and perform tasks is an unmanned aerial system (UAS). A UAS encompasses the components used to remotely control the UAV. Within a non-autonomous UAS, a pilot exists to operate the UAV remotely. Aside from this variable, most standard UAS consist of three main components: UAV, Ground Control Station (GCS), and Command and Control Link (C2). UAVs themselves consist of the frame, flight specific hardware (wings, propellers, etc.), communication subsystem (telemetry, flight controls, GPS), payload and payload controller, and the flight controller. The components onboard the UAV can vary depending on the UAV type. For example, depending on the mission, a UAV may be equipped with receivers for different communication mediums. Smaller UAVs operating close to the GCS may use RF or WiFi for the C2 link, whereas long range UAVs often rely on cellular or satellite connectivity. While operating within civilian airspace, the UAV may be required to be equipped with a collision avoidance system that receives location information from manned aircraft. The onboard sensor components are discussed further in Section 2.1.3.

The GCS takes flight commands from the pilot and transmits them wirelessly to the flight controller onboard the UAV. It is also responsible for the data transmission between the pilot and the UAV such as media feeds and telemetry information. The pilot issues flight controls and views telemetry information through the GCS, which utilizes wireless mediums to communicate with the UAV. Depending on the deployment, the UAS may consist of a single GCS (Home Ground Control Station (HGCS)) or multiple GCS strategically placed in the field (Field Ground Control Station (FGCS)). The FGCS architecture may be used to help offload large payload

data as well as a strategic automatic charging or fueling station. This architecture can help to increase the effective mission range of a UAV without needing to engineer better power or fuel consumption.

The modern UAS is rarely standalone, as processing and additional functionality can be increased by utilizing a back-end cloud infrastructure. Offloading processing such as artificial intelligence tasks helps to reduce computational load and power consumption onboard the UAV. Another common application of the cloud back-end infrastructure is client or command dashboards. Using a web interface, clients or UAV operators can view mission information anywhere over the internet. In commercial applications, clients of the UAV operator could review surveillance video or pipeline defects detected by the UAV AI. In the military and law enforcement space, the cloud infrastructure allows officials to view live footage to assist with tactical decision making.

UAVs communicate with the ground control station through C2 links. Although always wireless, different mediums such as RF, cellular, satellite, and 802.11 WiFi can be used. These mediums are chosen based on the requirements of the communication channel. Requirements are broken down based on the distance between the UAV and the GCS. For UAVs within short operating distances, WiFi or RF communication might be used, however at longer ranges, cellular or satellite communications would suit better. These links are used to communicate with the GCS, satellites, cellular towers, and other UAVs, and carry GPS data, media feeds, flight controls, collision avoidance broadcasts, telemetry information, and more. Figure 2.1 shows the architecture of the typical UAS.

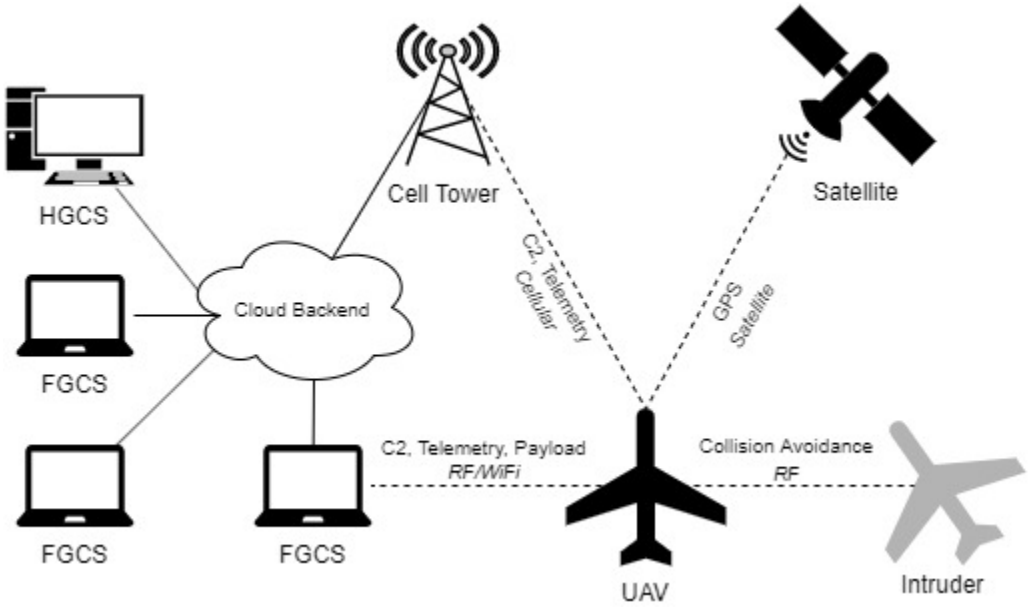

Figure 2.1: Typical Autonomous UAS

2.1.2 Communication Protocols

UAVs deploy a number of communication mediums and protocols depending on the intended use case and mission. For example, a long range plane UAV might use a satellite or cellular connection for command and control communication. Smaller UAVs operating locally will often use RF or WiFi for telemetry communication. Depending on the purpose of the communication and the medium being used, the underlying protocol in place will change. Due to the specialized use case, many communication protocols have been developed specifically for unmanned vehicle communications including MAVLink, ROS, uORB messaging and STANAG 4586.

MAVLink

The Micro Air Vehicle Link protocol, or MAVLink in short, is a popular open source communication protocol for use between an autopilot system and a GCS [19]. It allows

for messages to be sent to and from the autopilot or GCS over most serial connections without regard for the underlying medium. The protocol serializes messages into a stream of bytes before sending the messages across the medium. Therefore, it can be used among a multitude of mediums (WiFi, RF, Cellular), and it also has lower overhead compared to other serialization methods [20]. As such, the nature of the protocol makes it a viable choice for many autopilots (e.g. PX4 and ArduPilot), due to its versatility for use in various mediums and performance. The flight controller will monitor the onboard sensors as well as the state of various components on the UAV. It can then send the data to the GCS with the use of specific messages, designed for various components and sensors. Once a connection between the GCS and autopilot is established, the UAV will begin sending and receiving messages, while also sending a MAVLink *HEARTBEAT* message at a frequency of 1Hz. The *HEARTBEAT* message indicates that the device is currently active on the network. The GCS can then query the state of the device through various messages, for example, the *SYS_STATUS* message. The *SYS_STATUS* message informs the GCS of the general system state. The GCS can also query a specific component as required, such as servos, cameras, GPS, and ADS-B receivers.

As a bidirectional protocol, MAVLink is also used to send commands to the UAV. The *COMMAND_LONG* message is a multi purpose message used for sending commands. It uses a system ID and a component ID to specify the target system, as well as the component that should execute the command [20]. A payload is packaged into the message, which specifies the command to execute as well as other parameters. MAVLink is a binary telemetry protocol which operates in a multicast design, using

one of two modes: topic mode and point-to-point mode. Topic mode is a publish-subscribe multicast mode where messages such as positioning information is published and picked up by all components subscribing to the message. In point-to-point mode, delivery can be guaranteed by issuing a target ID and target component. In a UAV system, communication link bandwidth can vary as it crosses various mediums and resource constraint components, and these components may or may not need to receive the communication. Depending on these variables, the MAVLink message may be sent using topic mode (position data to GCS and flight controller) or point-to-point mode (mission commands to flight controller).

MAVLink can be expanded into different dialects in order to assist with the communication with protocol and vendor specific components. For example, custom messages can be defined to allow MAVLink communication to a specific manufacturers component. Dialects are defined in Extensible Markup Language (XML) files. By default, MAVLink will use the common message set which contains message definitions that are maintained by the open source community. Each definition contains a message ID, name, and description. The actual functionality of the message does not lie within these definitions, but rather it is up to the receiving system to interpret and act accordingly.

MAVLink version 1 has recently been replaced by version 2, which introduces a number of improvements including packet signing, a new 24 bit message ID, compatibility flags, and empty-byte payload truncation. These improvements increase the security, backward compatibility, and efficiency of the protocol. The MAVLink V2 packet structure is shown Figure 2.2.

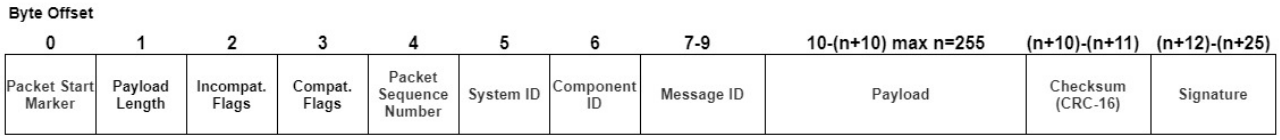

Figure 2.2: MAVLink V2 Packet Structure

ROS

The Robot Operating System (ROS) is an open source middle-ware for robotics [21]. Contrary to the name, ROS is not an operating system itself but provides similar hardware abstraction and communication between robotic systems. ROS is uses a publish-subscribe model where topics can be published and received in multicast fashion by those subscribing to the message. Topics in ROS are named buses that are used by nodes for the communication of messages in the aforementioned publish-subscribe fashion. Messages are published to the topic, which are in turn subscribed to by nodes. The messages are defined in text files which defined the structure and type of the message. The message can contain an optional header including a timestamp and frame ID. Once the message is published to the topic, the message is transported via TCP (TDPROS) or UDP (UDPROS). Nodes have the ability to negotiate which protocol they will use based on both compatibility and which mode the subscriber requests. As topics are unidirectional, there may be a use case for a node to make a remote procedure call. In this case, the node will instead use a ROS service. Services within ROS are a combination of ROS messages that define a request and reply. Lastly, ROS nodes can make use of a parameter server which is a shared dictionary accessible to the nodes over the network. The parameter server contains named variables with their given values. Figure 2.3 shows an example of nodes and topic communication flow.

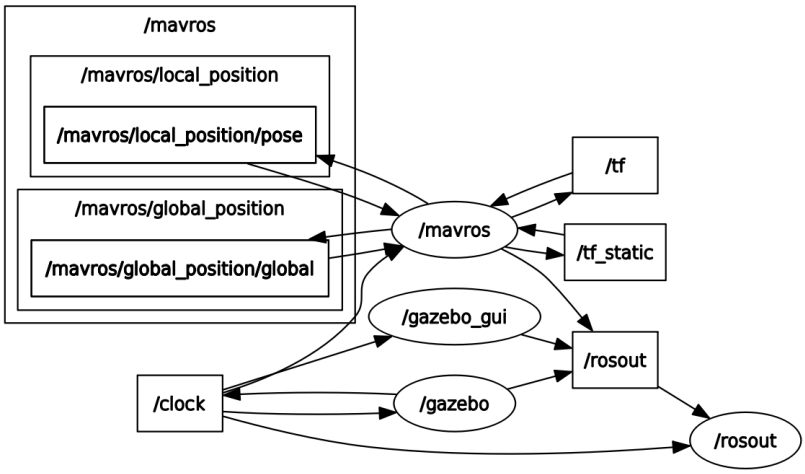

Figure 2.3: ROS Graph of MAVROS Communication Flow

uORB Messaging

uORB, or Object Request Broker, is a publish-subscribe middle-wear used communication between onboard processes and is part of the PX4 open source autopilot system [22]. uORB is similar to ROS, but further simplified to run within the resource constraint flight controller. The hardware abstraction of uORB allows inter-process communication between various components using a common communication protocol. uORB also relies on defined messages and topics, with default messages being defined with the option of user-defined custom messages. uORB is used within the PX4 autopilot as a central message bus between both components such as the GPS, flight control processes such as state estimation, logging, and even the transport of other protocols.

UAVCAN

UAVCAN (Uncomplicated Application-level Vehicular Computing And Networking) is a newer open technology similar to the traditional vehicular CAN protocol that is geared towards unmanned vehicles [23]. UAVCAN focuses on communications between multiple UAVs and components. It performs similarly to the MAVLink protocol by using heartbeats for node status, as well as assisting with communication between on-board system components. System component data is exchanged with the autopilot through the use of defined messages. UAVCAN can be used on both CAN networks using 29-bit identifiers, or via UDP on Ethernet networks. The advantages of UAVCAN versus other publish-subscribe messaging protocols is its ability to operate in CAN FD networks, its simplistic and robust design, and low computational requirements.

STANAG 4586

Standardization Agreement (STANAG) 4586 is a standards document developed by NATO that defines architectures, interfaces, communication protocols, data elements, and message formats, in order to allow for inter-operability amongst military UAV operations [24]. In modern tactical missions, the sharing of UAV assets by allies can improve communication and effective use of the UAV. For example, a UAV operating by one country could stream a tactical video link to the GCS of another country, both of which are different architectures and made by different manufacturers. Unfortunately, UAVs developed by different manufacturers will differ greatly in their onboard components, communication protocols, interfaces, and more. STANAG 4586 aims to standardize various levels of inter-operability between the three main

UAS components: the UAV, surface unit, and data link. In order to gain compatibility, manufacturers can developer vehicle specific modules (VSMs) which translate their own proprietary communication protocols into a STANAG 4586 compliant format. Five levels of inter-operability are defined by the standard which provide various amounts of control over the different components:

- Level 1: Indirect receipt and transmission of UAV data
- Level 2: Direct receipt and transmission of UAV data
- Level 3: Control and monitoring of the UAV payload
- Level 4: Control and monitoring of the UAV without launch and recovery capabilities
- Level 5: Control and monitoring of the UAV including launch and recovery

2.1.3 Onboard Sensors

UAVs utilize a number of onboard sensors in order to understand the environment around them, assist with positioning, and more. Given that the majority of UAV attacks will target the various onboard sensors, it is important to understand their purposes and general operation.

Global Positioning System Module UAV navigation is often reliant on the Global Navigation Satellite System (GNSS). GNSS is a term for a constellations of satellites that provide positioning and that have global coverage. The Global Positioning System (GPS) is one of the many GNSS'. The GPS system uses a network of

satellites equipped with transmitters and synchronized clocks. Each satellite transmits the current system time, as well its orbital information. A receiver on Earth can utilize messages from four of the satellites in order to triangulate its location. Three of the messages are used to figure out its location. Since the speed of light can be assumed as a constant, a GPS receiver can calculate the distance between itself and a satellite by comparing the message's time of arrival (TOA) to the time specified in the message. Three of these messages are enough to obtain the receiver's location, but a fourth message can further increase accuracy by providing height as well as an accurate time. When combined with the Internal Measurement Unit, the UAV is able to navigate in 3D space.

GPS modules can also be used for mapping payloads on the UAV, such as Real Time Kinematics (RTK) GPS receivers. These receivers take additional input from a fixed ground station and can provide accuracy down to the centimetre.

Inertial Measurement Unit The Inertial Measurement Unit (IMU) is an essential component within the UAV. The purposes of the IMU is to determine the attitude, altitude changes, and gravitational forces. This is done by relying on input from other sensors within, including:

- Accelerometer: measures gravitational forces, used to determine proper acceleration
- Magnetometer: measures magnetism, used in a compass to determine 2D direction
- Gyroscope: measures angular velocity, used to determine orientation

The input from the various sensors within the IMU are then filtered in order to reduce noise and error. The output of the IMU is the raw acceleration and angular rates, often noisy and unfiltered. The Kalman Filter is often used for this application in order to estimate orientation and angles. The IMU is typically contained within the flight controller, making the orientation of the controller during installation important as the compass and gyroscope rely on it.

Data Link Receiver UAVs can incorporate a number of different data links for various purposes, such as the communication of telemetry data and flight commands. Depending on the UAV deployment and use case, various communication mediums can be used. For example, closer range consumer or commercial UAVs can rely on 802.11 WiFi or RF radios. One common telemetry radio suite is the SiK Telemetry Radios. These radios are light, inexpensive, and built on open source firmware. They are common amongst the custom UAV and open source communities, offering both 433MHz and 900MHz versions both Europe and North America respectively. With a range of over 300m, simple RF radios are suited for most commercial deployments. In addition, consumer and commercial UAVs can also employ line of sight (LOS) controls by way of RC radios and remote controllers.

In long range deployments, the UAV will often be equipped with satellite or cellular communications. This allows for near global communication of telemetry, command and control, and payload data. Unlike RF radios, satellite and cellular are less prone to interference and have higher bandwidth available.

Optical Flow Optical flow sensors are used to stabilize the UAV in GPS-denied environments. The sensor is comprised of a downward facing camera which can

measure the visual motion of the perceived ground plane. Optical flow determines the motion of objects between visual frames captured by the camera. When a UAV moves, the optical flow algorithm determines the deformation between the frames and reports it as velocity. The measured velocity can then be used to calculate the distance moved by the UAV. In GPS-denied environments, the IMU will cause enough error to make the UAV drift off target. By using optical flow, the UAV can compensate for the calculated calculated drift.

Collision Avoidance Sensor Collision avoidance sensors can be used to both avoid other UAVs or manned aircraft, as well as obstacles such as trees, buildings, etc. Obstacle avoidance uses algorithms along with specialty sensors such as IR, LiDAR, and Sonar, in order to map the surrounding environment or sense an objects existence. Obstacle avoidance helps to ensure safety and protection of the UAV from an accidental crash. After detecting a obstacle, the UAV can implement a detect-and-avoid function to move around the obstacle or stop before colliding with it. Obstacle avoidance is primarily seen in consumer and commercial UAVs, as larger UAVs fly at higher altitudes where they are less likely to encounter these types of objects in their path.

Large UAVs that operate within the same airspace as manned aircraft will often be equipped with a collision avoidance system to detect and avoid aircraft and other UAVs. Interoperability between the UAV collision avoidance system and that used by manned aircraft is important. The collision avoidance system will take input from a surveillance source and act accordingly. In today's airspace, the Automatic Dependant Surveillance Broadcast (ADS-B) is the most common surveillance technology. ADS-B sensors are equipped onboard the aircraft and periodically broadcast

the location of the aircraft as determined from GPS. ADS-B Out will broadcast various details about the aircraft including its coordinates, velocity, call sign, and more. ADS-B In will receive this information and use it to alert the pilot of nearby aircraft, or as input into the automatic collision avoidance system. ADS-B is being incorporated into UAVs as well as it allows for the UAV and manned aircraft to detect and avoid each-other without relying on expensive hardware.

2.1.4 Flight Mechanics and Control

Understanding the basics of UAV flight mechanics and control is necessary to be able to see how and why the UAV reacts to different types of attacks. Flight mechanics refers to the control and orientation of the UAV in the air. This movement, also called attitude, is defined by the three axis in which an aircraft can alter its orientation in the air: roll (longitudinal), pitch (transverse), and yaw (vertical). How the UAV is able to take off, stabilize, and maneuver through the air largely depends on the type of UAV in question. Multicopters, planes, vertical takeoff and landing (VTOL) UAVs each employ different mechanics for flight and control.

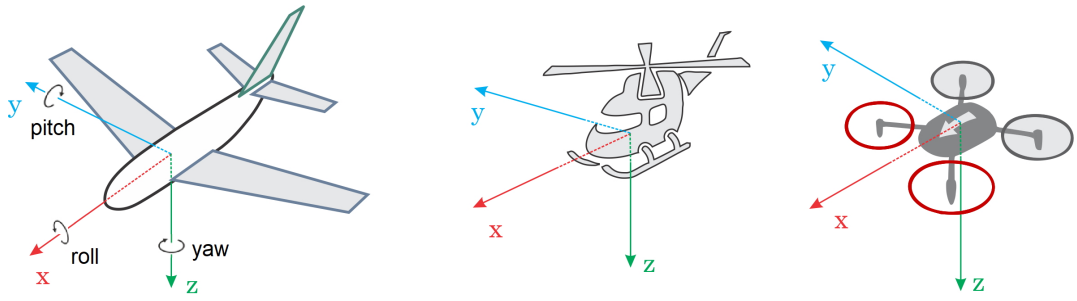

Figure 2.4: UAV Heading and Direction [25]

Multicopters Multicopters are UAVs that use multiple rotors to generate lift. These UAVs are often divided into sub-categories depending on the number of rotors equipped. For example, a quadcopter has four rotors whereas a hexacopter has six. Rotors are controlled by motors, whose speed is controlled by an electronic speed controller (ESC). Rotors diagonal to each other rotate in opposite directions in order to equalize the net reaction torque. When the rotor speed is increased to the point where the lift force exceeds the weight of the UAV, it will rise off the ground. Multicopters can hover when the lift is equal to the weight. Yaw motions are done by introducing different speeds between the pair of diagonally opposite direction rotors. Pitch is controlled similarly, except by altering the speeds between the front rotors and the rear rotors. When the speed of the rear rotors exceeds that of the front, the UAV will pitch forward and vice versa. Applying the same technique to the left and right side pairs will cause the UAV to roll in the direction of lower speed rotors. Figure 2.5 shows a DJI Matrice 600, an example of a commercial-grade hexacopter, equipped with a camera.

Planes UAVs of the plane configuration are similar to their manned counterparts, albeit equipped with sensors and flight stacks that allow for unmanned remote control or autonomous flight. The fixed-wings of the plane use airfoil designs in order to produce lift. The angle of attack is the angle of the plane against the oncoming air. With a greater angle of attack, the airfoil will generate more lift force. Flaps and slats of the wings can be engaged in order to create a higher angle and therefore lift. For a plane to takeoff, the lift force must be increased to a level where the lift is greater than its weight. This is done by increasing thrust from the motors as well as activating flaps and slats of the things for increased lift. Once the speed is high

Figure 2.5: DJI Matrice 600 Hexacopter [26]

enough, the rear elevator flap is activated upwards in order to increase the angle of attack. All of these activities combined cause the plane to take off into the air. Speed will continue to increase if the thrust is greater than the drag. After reaching level flight, the plane can be maneuvered by adjusting the rutter and ailrons on the wings to roll into a turn. Figure 2.6 shows the AeroVironment RQ-20 Puma, a UAV in the plane configuration, being launched by a member of the Royal Canadian Navy.

Vertical Takeoff & Landing Vertical Takeoff and Landing (VTOL) UAVs are a hybrid of both multicopters and fixed-wing aircraft. Rotors are used to lift the UAV vertically during take off and landing, and to provide the initial thrust before transitioning into fixed-wing mode. Being a hybrid, the VTOL UAV uses its multicopter features for initial propulsion and attitude control, then it's fixed wings for forward flight. This approach provides the advantages of multicopter operation, with

Figure 2.6: AeroVironment RQ-20 Puma Plane [27]

vertical takeoff capabilities and stabilization for limited takeoff space. Once transitioned, the VTOL gains the advantages of fixed-wing flight such as increased speed and endurance. Figure 2.7 shows a DeltaQuad Surveillance UAV flying in fixed-wing mode.

Figure 2.7: DeltaQuad VTOL Surveillance UAV [28]

UAV Control As UAV flight dynamics are variable and constantly changing, UAVs must be equipped with the hardware sensors and software needed to maintain a steady attitude. Various control architectures are used to calculate the error between a desired output value and a process value. Corrections are then applied to keep the output at the desired value while receiving changing input. In the case of UAVs, control architectures are used for various purposes including position control, velocity control, and attitude control. The control architecture can be either open loop, closed loop, or a hyrbid of the two. Open loop architectures do not monitor the output or use feedback from the sensors to adjust. Closed loop, however, makes use of sensor feedback to calculate adjustments. An example of a commonly used control loop is the Proportional Integral Derivative (PID) controller. PID a type of closed loop architecture that reads sensor values and calculates the appropriate motor speeds needed to control the UAV.

2.2 UAV Security Threats

Security threats facing UAVs have the potential to compromise one of the three pillars of the CIA (Confidentiality, Integrity, and Availability) triad. Confidentiality refers to the ability to keep sensitive or classified information out of the hands of unauthorized parties. The confidentiality of the UAV would be compromised if a threat actor was able to view a data feed, for example. Integrity refers to the trustworthiness and accuracy of data. If data is altered within transit, its integrity would be lost. An example of an attack against the UAV that compromises integrity is command injection. If the communication between the UAV and the command and control system are compromised, an attacker can potentially modify, replay, or inject false flight commands. When these messages are fabricated or altered, the UAV will continue to trust them even through their integrity is not in tact. Attacks against the availability of the UAV attempt to cause it to stop functioning. A common attack of this type is jamming, where a threat actor will generate enough RF noise to cause the legitimate signals to be disrupted.

To gain a good understanding of the threats UAVs face, a security assessment engagement was initially conducted with a local UAV manufacturer with operations around the world. During this exercise, penetration testing was conducted against each component of the UAS including the GCS, cloud infrastructure, communication links, and the UAV itself. The threats discussed below are a combination of the findings from this exercise as well as research in UAV vulnerability and IDS literature. The results of the vulnerability survey are shown in Table 2.1 [6].

2.2.1 Jamming Attacks

Jamming occurs when an adversary transmits radio signals with the intention to disrupt legitimate communications, compromising the availability of the UAV. This occurs when the signal-to-interference-plus-noise (SNIR) ratio deteriorates to the point where there is more noise than legitimate signal Vadlamani et al. [29]. Jamming attacks are common as they do not require sophisticated knowledge and can often be done using inexpensive software-defined radios (SDR). Apart from signal-based jamming, denial of service (DoS) attacks through other means can cause the same effect. For example, flooding the telemetry link with requests or triggering a memory leak are examples of other means to deteriorate system availability.

Telemetry Link Jamming Data links for UAVs can be used to transmit and receive telemetry or C2 data. A large portion of an UAVs data comes wireless data links such as GPS, WiFi, cellular, radio, ADS-B, etc. By rendering these data links unusable, a malicious party can cut the UAV off from its external environment. Some UAVs do not depend on human interaction, so cutting off the data link between the GCS and the UAV does not have a severe effect. Those that do depend on human interaction will be crippled by the loss of connection. In both scenarios, a malicious party can advance the jamming to conduct further attacks on the UAV and C2 will not be aware.

GPS Jamming The GPS is crucial component to many UAVs. Most of them, if not all, depend on the GPS as their primary source of location data. Preventing an UAV from obtaining its location can have adverse effects, ranging from mission failure to the destruction of the drone. GPS signals come from satellites over 20,000km away.

The typical strength of a GPS signal is only 1.5 dB, making jamming a GPS signal an easy task for an adversary with even a small device [30]. GPS jamming is one of the most common and dangerous threats to UAVs, given the simplicity and availability of specialized devices to carry out the attack. One example of this is an attack against a swarm of UAVs used in a Hong Kong light show, which were taken down by a GPS jamming attack. The attack caused over $100,000 in damages [10].

2.2.2 Spoofing Attacks

Spoofing attacks occur when a threat actor impersonates another device, system, or user when interacting with the UAV. These types of attacks compromise the integrity of received signals and can be devastating to the operation of the UAV. Without proper authentication and signatures, it can be difficult to detect and prevent these attacks.

GPS Spoofing The GPS system is relied upon heavily by UAVs. Spoofing of GPS signals can lead to devastating results. GPS spoofing occurs when a malicious party broadcasts fake GPS signals, either by replaying a modified set of signals or by creating their own. This could potentially cause the target GPS receiver to falsely estimate its position. GPS spoofing usually occurs in two ways: Commonly, this occurs by broadcasting signals synchronized with genuine signals, then gradually increasing the power of the fake signals, causing the receiver to switch over to receiving messages from the fake broadcaster. Once the UAV locks to the spoofing coordinates, it will falsely assume it is off course. In this situation, the UAV will try to adjust its trajectory to get back on target. Successful GPS spoofing can not only cause a UAV to crash, but lead to hijacking. GPS spoofing has been seen in real world attacks

against UAVs, such as in 2017 when a Mexican cartel used GPS spoofing against a U.S. Customs and Border Protection UAV [9].

ADS-B Spoofing Autonomous UAVs often coexist in airspace with manned aircraft. For this reason, it is necessary that the UAV be equipped with a compatible collision avoidance system. Modern airborne collision avoidance systems typically utilize a stack of communication protocols and logic-based algorithms to be able to detect and avoid other aircraft. The most common communication protocol that is often a basis for collision avoidance systems is Automatic Dependant Surveillance - Broadcast (ADS-B). ADS-B is part of an Air Traffic Management and Control (ATM/ATC) Surveillance system, in which an aircraft determines its position via satellite and periodically broadcasts it to enable other aircraft and ground devices to track it. This information is often used in conjunction with on-board Secondary Surveillance Radar (SSR), as ADS-B does not require any significant equipment on the ground. ADS-B is automatic as it is not designed to take any external input, and dependant because it is reliant on the aircrafts navigation system. The information relayed over ADS-B includes call sign, current position, altitude, and velocity, which is broadcast approximately every second. The advantages of ADS-B versus radar include increased visibility and situational awareness, higher efficiency, and significantly cheaper. For these reasons, ADS-B is beginning to be used in some UAVs as an air traffic surveillance source to the collision avoidance system [31].

Although ADS-B has been broadly used around the world for decades, it is plagued with many security vulnerabilities compromising the CIA of the aircraft [15, 32, 33]. UAVs are primarily required to yield to larger aircraft. This makes it common practice for the UAVs onboard ADS-B sensor be used as input to the collision avoidance

system. As ADS-B has no authentication and its messages are fundamentally trusted by collision avoidance systems, spoofing fake ADS-B messages becomes trivial. In this scenario, an attackers spoofed ADS-B messages can result in trajectory modification and hijacking of the UAV.

Outside of collision avoidance in larger UAVs, smaller UAVs can be prone to hijacking through exploitation of the obstacle avoidance system. As sensor data comes from the environment, it is fundamentally not trusted and has the potential to be manipulated by an adversary. As an example, Pierpaoli and Rahmani [34] have shown the exploitation of predictable obstacle avoidance logic in UAVs, where a pursuer is able to steer the evader off course to an arbitrary target. This vulnerability leads to a complete loss of autonomy.

Optical Sensor Spoofing Optical flow sensors are able to determine differences in subsequent images, in order to determine the displacement of an object. In the context of UAVs, optical flow sensors can be used to discover moving objects, or to measure the velocity of the drone itself. A downward facing sensor can compare differences between frames to estimate the rough ground velocity of an UAV. This can also be used to determine whether an UAV is stationary or if it is currently in motion. If an UAV uses optical flow sensors to maintain a stationary position, by measuring displacement and issuing a countering movement, it becomes possible to spoof the sensor in order to manipulate the motion of the UAV. A malicious party can manipulate the flow sensor to believe that there is a change in position, and as a result it causes the UAV to move in order to counter the change.

Optical flow sensors use feature detection algorithms to identify features in frames,

and then determine displacement by comparing the position of features between successive frames. Malicious parties can successfully influence the sensor by altering or influencing the image with feature rich patterns. This can be done from a distance by projecting light patterns onto the surface that the optical flow sensor is facing, using a projector or laser [35].

2.2.3 Injection Attacks

Injection attacks occur when unauthorized or unsanitized malicious input is accepted and processed by the receiving system. Injection attacks are common in traditional computer security, where an adversary can injection malicious code into a process, such as Structured Query Language (SQL) injection. In the UAV domain, injection attacks occur similarly when input from the environment is trusted and processed without sanitation or authentication. This type of attack could be carried out against multiple different components of the UAS depending on their software implementations.

Command Injection Command injection attacks occur when an adversary gains the ability to execute arbitrary commands against a system. The most prominent command injection attack against UAVs are those conducted against the flight controller. Depending on the communication protocol and flight controller firmware, the system may not authenticate nor verify the source of flight commands. One example of this is an attack chain against the MAVLink protocol which leads to command injection. As with most command injection vulnerabilities, the attacker must be able to first communicate with the target system in order for the injection to occur. MAVLink uses a *NetID* parameter to pair telemetry radios with the GCS, in order

to avoid two radio pairs within close proximity from interfering with one another. Lacking any sort of key exchange or authentication, an attacker can simply change their *NetID* to that of the target system in order to be able to communicate with it. Moreover, as the flight controller does not authenticate or sign commands, the attacker can use this new connection to inject arbitrary commands into the flight controller for processing [36, 37]. The more recent version 2 of the MAVLink protocol has introduced packet signing to mitigate this attack, however, it is disabled by default [38].

Acoustic Noise Injection Acoustic noise injection is an attack that targets specific types of gyroscopes within the Inertial Measurement Unit (IMU). The Micro-Electro-Mechanical Systems (MEMS) gyroscopes are common amongst smaller UAVs as they are lightweight and inexpensive. MEMS are vibrating structure gyroscopes which use a vibrating structure in order to determine the rate at which it is rotating. These types of gyroscopes have been shown to be vulnerable to an attack called acoustic noise injection [39, 40, 41]. Out-of-band analog acoustic noise is emitted by the attacker that closely matches that of the natural frequency. This injection causes the sensor to enter resonance. Once this signal is digitized, the output of the manipulated signal can then influence the control of the UAV. A successful acoustic noise attack will cause the UAV to crash to make an unintended maneuver. The threat of this attack is increase by the potential use of long range sonic weapons, capable of projecting targeted acoustic noise up to 2.5km.

2.2.4 Pilot and Ground Control Station Attacks

Ground Control Station Security The GCS is where mission planning, launching, and monitoring occurs. Compromise of the physical security of these stations by unauthorized parties can lead to the compromise of the network or sabotage of the physical assets within. In addition, if a multiple signature approach is not in place to prevent rogue launching or mission commands, insider threats will be possible. Long range UAV deployments may place additional GCSs along the autonomous UAVs route to supply charging and payload upload capabilities. Because of their often remote locations, physical security plays an important role. Without alarms or remote monitoring, their physical security can be compromised fairly easily.

In addition to physical security, the lack of network security of the internal GCS network or corporate networks could cause a compromise of either one of these networks by pivoting from one to the other. This also increases the risk of insider threats if corporate users are able to access the GCS networks. As well, if the network is not properly segmented it increases the threat to outside local actors if wireless networking is used. Cracking strong WiFi such as WPA2 has become trivial in recent years with the introduction of new attacks such as PMKID [42]. In addition to cracking the WPA key, other WiFi attacks are possible such as denial of service, de-authentication and man-in-the-middle attacks [43]. Hiding the network SSID does not provide security, but rather obscurity in an attempt to mitigate such attacks. Any hardwired external network endpoints located outside of the station should also be treated with a high threat level due to their increased exposure. In the case of a physical security perimeter breach, an adversary could connect to endpoints such as weather stations or external power-over-ethernet cameras to gain network access without needing to

crack a WiFi key.

Eavesdropping attacks aim to intercept communications with the goal of gaining sensitive information. These attacks can be used to listen in on UAV commands, telemetry data, and even payloads. Eavesdropping is common amongst traditional TCP/IP networks in the form of man-in-the-middle (MITM) attacks. In TCP/IP, most traffic is unicast, and therefore an additional step is required in order to capture this traffic. An example of this is ARP spoofing, where an attacker replies to ARP requests with spoofed MAC addresses matching their own. This causes future unicast traffic to the legitimate destination to be forwarded to the attackers machine. Although trivial in most networks, the ability to eavesdrop on traditional TCP/IP networks requires slightly more sophistication than simply listening to broadcast messages. This is further complicated when encryption schemes, such as TLS/SSL are implemented. Unfortunately, many UAV communication protocols are based on entirely broadcast communications, and often do not have any encryption schemes implemented.

A backend cloud infrastructure is typically used in the UAS for payload storage and processing and the display of mission data. The cloud infrastructure is used to store sensitive information about the customer, their missions, and the payload results. For this reason, the security and privacy of the cloud is an important aspect of the UAS threat analysis. Fortunately, the security of the cloud is not unique to a UAS and its best practices have been well defined. Singh and Chatterjee [44] have presented a survey of the general security challenges faced with cloud infrastructure. An in-depth threat analysis of general cloud infrastructure is out of the scope of this thesis, however, there are some threats that are prominent in UAS. One such example

is the security of the applications hosted in the cloud and the connection to the UAV as they carry the highest risk due to their exposure.

Insider Threats Insider threats are those that come from within the organization. In the UAV domain, the vehicle operator is the primary user with the potential to become an insider threat. Although some UAVs are autonomous, they still have remote operators. These operators use a control station to issue C2 commands, monitor telemetry information, execute payloads, and download payload data. Insider threats are similar in some ways to other attacks, such as command injection or eavesdropping, however, they come from the operator themselves with malicious intent [45]. Examples of malicious insider actions include:

- Downloading of classified data onboard the UAV
- Payload execution outside of the pre-determined mission parameters
- Rogue launching of missions, termination of missions, etc.

Table 2.1: Survey of UAV Vulnerabilities

Vulnerability	**Targeted Systems and Sensors**	**Impact**	**References**
Telemetry/C2 Link Jamming, Denial of Service	Telemetry communication links, RF, WiFi, Cellular radios	Loss of availability, lack of telemetry, lack of command and control	Pärlin, Alam, and Le Moullec [46], Vuong et al. [47]
GPS Jamming	GPS sensor, navigation system	Loss of availability, loss of navigation	Hu and Wei [48]
GPS Spoofing	GPS sensor, navigation system	Loss of integrity, loss of availability, hijacking/trajectory modification, loss of navigation	Kerns et al. [49], Faughnan et al. [50], Tippenhauer et al. [51]
ADS-B Spoofing	ADS-B sensor, collision avoidance system	Loss of integrity, hijacking/trajectory modification	Costin and Francillon [15], Faughnan et al. [50]
Optical Sensor Spoofing	Optical flow sensor, navigation system	Loss of integrity, hijacking/trajectory modification	Davidson et al. [35]
Acoustic Noise Injection	Gyroscope, inertial measurement unit	Loss of availability, loss of integrity	Khazaaleh et al. [41], Tu et al. [40], Son et al. [39], Khazaaleh et al. [41]
Command Injection	Control/autopilot system	Loss of availability, loss of integrity, hijacking/trajectory modification, unauthorized payload delivery	Butcher, Stewart, and Biaz [52], Marty [53], Vuong et al. [47]
Eavesdropping	Telemetry communication links, command and control links	Loss of confidentiality	Breiling, Dieber, and Schartner [54], Butcher, Stewart, and Biaz [52], Marty [53]
Insider Threats	Command and control system	Loss of availability, loss of integrity, loss of confidentiality, hijacking/trajectory modification, unauthorized payload delivery	Ahmad Yousef et al. [55]

2.3 Intrusion Detection Systems

An intrusion detection system (IDS) monitors the activities of a device or network with the intent of identifying suspicious activity or potential threats. When a threat is detected, an alert is triggered and sent to the security staff or system administrator. The information contained in the alert will assist the user in identifying the type of attack, the source of the attack, and the targeted system. Further, an intrusion prevention system (IPS) has the ability to further *prevent* the identified intrusion. This is done by automatically blocking the identified attacking host or adding a firewall rule to prevent further attacks.

An IDS can be deployed within a network to pick up on traffic anomalies or directly on a host as a host-based IDS (HIDS). In either case, a "sensor" is deployed and tasked with monitoring logs, traffic, or user activity. In a HIDS, the sensor is deployed on the host, inspecting logs and activity to detect intrusions. A host can be an endpoint computer, server, specific system component, etc. HIDS have a number of advantages to network-based IDS. A HIDS can monitor applications as well as malicious activity that do not communicate externally outside of the host. One disadvantage to HIDS, is that in the event of system compromise, the HIDS could be disabled or the alert removed. The HIDS can run standalone or be part of a larger IDS incorporating both host and network-based sensors.

Network sensors are deployed with the intention of inspecting network packets in transmission. This type of IDS allow for the detection of attacks before they reach a host, and are simpler to deploy and maintain in larger environments compared to their host-based counterparts. As the sensor must keep state information about each network connection, they can suffer from significant overhead.

The deployment of an IDS requires consideration and knowledge of the environment it will be operating in. The IDS placement is important in order to detect threats against identified assets. In the case of UAVs, threats can target the ground control station, the UAV sensors, and the communication links between the two. If the IDS is only placed on the ground control station, it will miss attacks toward the UAV itself. In addition, the maintenance requirements of the underlying detection method should be considered. The placement and maintenance needs of the IDS should be revisited regularly [56].

The IDS can use a number of different methods for the detection of attacks. The most common methods are classified as signature-based, specification-based, and anomaly-based.

Signature-Based Signature-based methods of detecting attacks were originally deployed by anti-virus applications. Signatures make up known patterns or hashes and are used to find a match with an occurring threat. For example, the patterns could be a domain name, IP address, or series of packets or bytes that make the threat indicators unique. Signature-based methods to intrusion detection often have very high true positive rates with low false positive rates. This is due to the fact that if a known attack is occurring and a signature exists, the IDS simply needs to match the known features. The limitation to this approach is that if a signature is not created, the attack will not be detected. As new attacks and current attacks are modified, it becomes difficult to the IDS creator to keep up with the creation and implementation of new signatures. In addition, network traffic may be encrypted, making it difficult for the IDS to identify indicators of compromise.

Specification-Based In specification-based intrusion detection, a set of boundaries or "behavioural specifications" are defined, and the IDS triggers when actions of the user or an attacker deviates from these specifications. The defined specifications can be attributed from a security policy, or in the case of a UAV, normal behaviours such as known flight paths. This method of intrusion detection can be effective in specific use cases, but it has many drawbacks. First, it is difficult to scale as specifications need to be defined for each behaviour we want to detect. In addition, it can be difficult to verify the created specifications are correct, and therefore requires costly testing to be confident the definition triggers correctly [57].

Anomaly-Based Anomaly-based techniques have been used more recently to detect emerging threats. A baseline of traffic patterns or user activities is created, and any observation outside of the baselines is classified as a anomaly. The advantage of anomaly-based techniques is their inherent ability to detect new and unknown attacks with minimal or no ongoing maintenance. Creating a baseline of normal activity requires a number of samples to generate. If an attack occurs in which very little traffic or actions are required, the IDS may be unable to detect it. On systems where the IDS is attempting to detect behaviour-based anomalies, it is creating a baseline of the users activity. Although technical systems may remain relatively constant in their operation, human operators are more likely to carry out actions outside of the norm. In general, anomaly-based techniques can be effective with the advantage of flexibility, however, they often produce higher false positives in return.

2.4 Challenges in UAV IDS

The design and development of an intrusion detection system for UAVs comes with its own challenges. Although there are some similarities between the UAV domain and other domains, such as robotics and IoT, it offers a unique combination of differences that require consideration. First, as UAVs operate in the air, they are faced with size and weight constraints. On-board components must be physically small in order to fit within the UAV frame. In addition, the weight of the components is extremely important as it can have a negative effect on the flight dynamics and lift of the UAV. As the UAV is essentially a flying computer, the on-board components also face computational power limitations. As the electronic components are limited in size and weight, they in turn are faced with the resulting computational power limitations. As machine learning-based intrusion detection is often computationally heavy, it is especially challenging to design and develop one for UAVs. Regardless of the type of IDS, it must be able to process data coming from the UAV fast enough to not impede its operations. If the IDS cannot process incoming communications fast enough, it will result in a bottleneck. In larger UAVs, the size and weight of on-board IDS components may not be as much of a concern, however, the components can still draw significant battery power from the UAV. This can lead to other components starving for power or a reduce flight time.

Depending on the IDS design, the aforementioned challenges may not apply. For example, an IDS that forwards all requests to a ground control station for processing does not suffer from the limitations of on-board components. Unfortunately,

approaches based on GCS processing can introduce other problems. UAVs rely completely on wireless protocols such as RF, cellular, and WiFi. Unlike wired communications, these protocols can suffer from latency over the link. The IDS communications can introduce more traffic on the telemetry or command and control links, making them less responsive or not responsive at all. In addition, pre-existing latency may reduce the effectiveness of the IDS, as messages may be dropped or corrupt. Jamming and other denial of service-based attacks are the most simplistic and therefore the most common attacks against a UAV [58, 46, 47]. IDS solutions that rely on a GCS for processing simply become inoperable during a jamming or denial of service-based attack, as the communication between the UAV and GCS will become lost. To avoid this issue, the IDS must reside onboard the UAV, introducing the aforementioned size and weight constraints.

One of the main challenges in developing an IDS for UAVs is the variety of platforms and onboard equipment. UAVs can have significant differences depending on the mission objectives. For example, a surveillance mission will often deploy a plane or VTOL UAV, where a quadcopter is more fitted for short range missions. Different UAV platforms employ different control configurations and flight dynamics, which changes how an intrusion detection system must interpret values from the onboard components. The mission of the UAV can also dictate which sensors and communication mediums are needed. Communication mediums such as cellular or satellite are more fitted towards long range command and control or telemetry communications, whereas WiFi and RF may be more suited for short range flights.

Machine learning-based approaches to intrusion detection can address the above noted challenges, as they can be used to learn from the data for accurate detection.

Machine learning approaches typically require labelled datasets for high accuracy, however, there are currently no such public datasets available to the community. This is likely because of the aforementioned variances in UAVs themselves, making it difficult to build a "one size fits all" dataset for UAV IDS. It is also important to note that as UAVs are cyber-physical systems, environmental influences such as high winds could possibly trigger false positives with machine learning approaches.

2.5 Machine Learning

Machine learning is a subset of artificial intelligence that uses techniques that allow computers to learn based on experience, without explicit programming. Machine learning techniques aim to iteratively teach algorithms how to improve their predictions on given data. Machine learning algorithms are being used more and more in across industries to solve complex problems such as image recognition, financial predictions, medical diagnosis, and fraud detection.

When approaching a machine learning problem, we first take a dataset in which to learn and derive future predictions from. This dataset is often split into two parts: a training set and a test set. The training set is used by the chosen algorithm in an optimization task of finding the minima or maxima of a cost function, therefore generating an accurate predictive model. The test set is then used to test and calculate the accuracy of the generated model. The aforementioned process is typical in regression problems where predictions need to be made, such as stock prices, but machine learning can be useful in other problems like clustering. In clustering problems, we look to learn something we do not already know about a dataset. Clustering can help by learning underlying patterns, or features, and grouping related data together.

There are three main types of machine learning techniques that are used to tackle different problems, each with their own prerequisite requirements and performance benefits. These algorithms are chosen based on the task at hand, the dataset that is available, and the resulting prediction or type of clustering we want to achieve as a result. The three types of techniques are supervised learning, unsupervised learning, and reinforcement learning.

Supervised Learning Supervised learning is the most common type of machine learning, being the most straight forward to understand and implement. This type of learning requires a labelled training dataset, where the data and its associated true prediction, or label, are paired together. The chosen algorithm goes through an iterative learning process where it makes a prediction and is given feedback as to whether the predicted label is correct or not. As the training process progresses, the predictions made by the algorithm will become more and more accurate. Once an acceptable level of accuracy is obtained without overfitting the data, the model can be exported and used to predict labels on previously unseen data. Common supervised algorithms include linear regression, naive bayes, decision trees, and neural networks.

Supervised learning algorithms can be improved by providing more training samples for the algorithm to learn from. This can become an issue in certain scenarios as a lack of data can hinder the feasibility of a machine learning approach. Depending on the required dataset size, labelling a large dataset can also be costly and tedious. With these limitations aside, supervised learning is arguably the most used machine learning technique, and is used in many mainstream artificial intelligence applications such as facial recognition, object detection, and spam filtering.

Unsupervised Learning In contrast to supervised learning, unsupervised techniques do not have labels available with the training dataset. It is left up to the chosen algorithm to identify features and cluster the data appropriately. There is no prediction output with unsupervised learning, but rather the algorithm aims to detect patterns and group the data based on the learned features of the underlying data. Examples of unsupervised learning algorithms include k-means clustering, principal component analysis, and independent component analysis. Unsupervised learning is

popular as most data that is acquired is unlabelled and is not feasible to label. One example of unsupervised learning is clustering, where data is categorized into learned groups.

Reinforcement Learning Reinforcement learning is a machine learning technique where the algorithm takes actions based on observations from the environment. The actions are based on calculated risk, where the action that maximizes reward is selected. Just like supervised learning, reinforcement algorithms take iterative steps, but learn from its experience within the environment. This is continued until the algorithm has investigated all possible states. As the name suggests, if the algorithm selects the best action, it will receive reinforcement in the form of a reward. Reinforcement learning algorithms are used for a number of decision-based applications such as computer played board games, self-driving cars, and robotics. Examples of reinforcement learning algorithms are temporal difference and deep adversarial networks.

2.6 Review of IDS in the Literature

The design and development of an effective intrusion detection system (IDS) for unmanned aerial vehicles (UAVs) has become a trending topic for researchers. With human safety and confidential data on the line, a successful cyber attack against a UAV can be devastating. Success in this area of research is critical to the safe and secure use of UAV technology for years to come. Intrusion detection in the UAV domain comes with its own unique challenges. Unlike traditional networked computer systems, UAVs are flying computers that must rely on wireless technologies, must have low latency, are limited to low computational power, and have size and weight constraints to onboard components. In addition, depending on the intended use case, UAVs are created with different communication mediums and protocols, various sensors (GPS, Lidar, RF radio, etc), and have different frames and control configurations. For example, a UAV designed for surveillance purposes requires longer endurance with a faster flight time, longer distance communications, and various cameras. In this example a plane or VTOL UAV with cellular or satellite communications is likely the best option. On the other hand, a WiFi-based quadcopter might be more suitable for a mission requiring a short range UAV with more agility. This wide array of technologies involved in UAVs makes IDS development much more difficult than in traditional systems.

Although the UAV domain has unique challenges for IDS development, UAVs do have some overlap with traditional systems. UAVs are considered low-powered devices, as they are constrained by size and weight limitations. Intrusion detection in traditional low-powered devices, such as the Internet of Things (IoT), is well researched. Lessons learned from IoT intrusion detection techniques could be applied

to the UAV problem. IoT devices are internet-connected physical "things", hardware with embedded sensors and software. Examples of IoT devices include connected appliances, wearable health monitors, and smart locks. These devices have constrained resources and are typically based on low-power lossy networks (LLNs) [59]. LLNs are networks in which both the end device and the network itself are resource constrained, which is typically the case with UAVs. IoT devices also share some similar threats to UAVs, mostly due to their reliance on wireless protocols. A simple example of a shared threat between both the IoT and UAV domains is the use of Zigbee and IEEE 802.15.4. As popular low-data rate wireless protocols, they have seen use in the UAV domain for various communications purposes such as telemetry data transmission [60, 61]. Vulnerabilities against Zigbee are well documented and include denial of service, replay attacks, and eavesdropping [62, 63]. Additional shared threats to low-powered IoT devices include jamming and denial of service, insecure communications, spoofing, and software and firmware vulnerabilities [59].

One of the most prominent intrusion detection solutions for low-powered and IoT devices is game-theoretic approaches. Game theory comprises mathematical models of strategic decision making amongst adversarial contestants. Similar to a traditional game such as chess, the "game" consists of rules outcomes. Sedjelmaci, Senouci, and Taleb [64] proposed the use of game theory where the results of a security game cause the additional signature and anomaly-based detection techniques to be initiated. The proposed system aims to produce high results while minimizing energy consumption, even when scaled to a high number of IoT devices and attacks. This system uses both signatures as well as different anomaly detection techniques including support vector machine and neural networks. With the help of Nash equilibrium, an IDS

agent can determine the equilibrium state, and accordingly activate the anomaly detection technique in order to detect a new attack pattern. A set of rules is created to model the attack, which can then be used by the signature detection technique to capture any similar attacks in the future. The main contribution of this paper, however, is the introduction of the security game. A set of players is defined where each represents either the IDS agent running on the IoT device or the attacker themselves. This is a complete information game where both sides know the payoff. The key premise is that in an IoT network, we can use game theory to help determine the reputation of a node, given normal, suspect, or malicious. Depending on the outcome of the security games, a signature or anomaly-based intrusion detection technique is deployed. When an attacker is identified, it is ejected from the IoT network. The use of game theory helps reduce false positives and ensures the costly signature computation or anomaly detection tasks are only carried out when an attack is probable. A similar game-theoretic approach is proposed by Tang, Ren, and Han [65] for the detection of jamming attacks in low-powered IoT devices with the goal of reducing the computational power required to effectively mitigate the attack. Game-theoretic approaches can satisfy the requirements of low-powered networks, similar to those in UAV networks.

Early approaches to intrusion detection within the UAV domain also make use of game-theoretic and Bayesian approaches. This is because UAV networks are similar to IoT, given the low computational power limitations of on-board components. Wang and Feng [66] discussed the use of game theory in for UAV intrusion detection. Their work used game theory as a basis for anomaly detection using a complete information game. Utility function matrices are created by looking at the cost and

benefit of attacking and defending each sensor. The Nash equilibrium is then found by either opponent during an attack. The primary goal of this method is to reduce computational load and power consumption on the UAV. To do this, it only "protects" the components most likely to be attacked by using game theory. A theoretical experiment was done using Matlab, and the results are mixed. A win can be done with low overhead. This approach does reduce computational cost, however, there are a number of drawbacks. When a loss occurs, it will be a catastrophic because the defender will be "protecting" the wrong sensor, leaving it wide open for attack. In addition, this approach aims to reduce computational load and power, but compromises too much in the detection rate to get this result.

Similar approaches use Bayesian games in which the players do not have complete information about the other players. Sun et al. [67] proposed a Bayesian game theory approach. The architecture used consists of multiple UAVs arranged in clusters similar to an ad-hoc network with cluster heads and a sink node (base station). An IDS instance runs on each of these elements. When an alert is triggered, the nodes route the messages through the network to the sink node for further actions. In this type of architecture, a high detection rate would create a lot of overhead on the network as the messages traverse to the sink. The IDS and the attacker are the players in the game. By finding the Nash equilibrium, the best balance between a high detection rate and low overhead can be found. The results are compared with two other similar approaches in the same simulated environment. It is shown that the Bayesian game theory approach does help to produce a high detection rate while maintaining a low false positive rate. A similar approach is proposed by Sedjelmaci, Senouci, and Ansari [68]. Their work focused on intrusion detection within mobile ad-hoc networks, where

UAVs fill the network disconnect. The authors introduced a method of activating the IDS only when a neighbouring UAV is expected to conduct a malicious activity on the network. It is suggested that an malicious UAV should not be ejected from the network immediately as it might be a false positive or be the cause of noise or interference. To combat this, a method of ejecting malicious UAVs from the network only when their actions continue is discussed. The issue with this framework is that it is designed for MANETs where there is a network of UAVs. Without this, the IDS would never get activated at all (ie. attack from the ground control station or adversarial UAV not part of the MANET). UAVs operating in MANETs are only a subset of the uses of modern UAVs. Those operating in surveillance, for example, operate alone without neighbouring nodes and therefore the proposed technique is not applicable.

Specification-based intrusion detection techniques are also common in the literature. These approaches identify behavioral specifications as the basis for detecting attacks. When the specification is altered, it can be assumed an attack is occurring. An example of this is specifying white-listed locations, and triggering an insider threat alert when the UAV travels outside of these. Mitchell and Chen [69] introduce the use of a specification-based IDS for UAVs with close to 100% accuracy for detection. Mitchell and Chen [70] also use a specification-based system, but to detect other UAVs that are operating maliciously within the airspace. A "behaviour rule-based" approach is taken where normal behaviour is defined with given values, and anything outside of these ranges is considered abnormal (ie. landing gear down while outside of an airport). These values are then put into a state machine where variances outside the norm can be detected. This IDS can only detect compromised

UAVs, not those currently under attack. In addition, it is only able to detect attacks in which rules have been created for. For example, a GPS spoofing attack, one of the most common against UAVs, would go undetected. A drawback to the design of this system is that it expects trusted neighbouring UAVs to monitor each other, making this system difficult in operations where a UAV fleet is not necessary or available.

Fotohi [71] proposes the use of a human immune system (HIS) applied to the unmanned aerial system of networked UAVs. The authors proposed to develop an immunity system against threats, similar to how the immune system works within the human body. This is done by creating an HIS algorithm, which requires a number of manual steps to generate. Four attacks are used as examples: wormhole attack, black hole attack, grey hole attack, and fake information dissemination. With the exception of fake information dissemination (eg. GPS spoofing), these attacks require specific circumstances and have not been seen as major threats amongst the literature. The method has a heavy reliance on packet routing, but discusses its uses theoretically. It is also unclear how this method would work with real communication protocols and UAVs.

Machine learning-based approaches are also common in the literature. These approaches aim to make use of machine learning techniques to identify anomalies or malicious heuristics within various contexts of UAVs operation. Intrusion detection using support vector machines has been proposed by Panice et al. [72] to detect GPS spoofing. A three phase approach is proposed starting with segmentation. In this phase, the euclidian distance between the GPS and INS data is calculated. Next, each set of points is used to create a support vector data description (SVDD) or

v-support vector machine (v-SVM) models. Finally, the number of outliers is multiplied by a user-defined constant in order to control the resistance to false positives. This approach showed promising results in a simulated environment, however, its effectiveness against other attacks is unknown. In addition, it requires a labelled dataset.

Arthur [73] proposed an intrusion detection method utilizing multi-class support vector machines. GPS spoofing and jamming are used as the attacks of concentration. It is acknowledged that labelled datasets are unavailable and unlikely for the UAV domain. The approach first uses self-taught learning in order to gain a representation of features from the unlabelled dataset. These features come from onboard component data, such as actuators, sensors, and navigational systems. A supervised learning method can be used to further classify new data using the output of the self-taught learning algorithm. Hierarchical-based multi-class support vector machines are used in the proposed IDS which provides the ability to differentiate between attacks. This approach also includes mitigation techniques, reverting the UAV to "self-routing" when a GPS jamming or spoofing attack is detected. The results show a minimum accuracy of 72% with the highest being 94%. The performance impacts on the UAV from this approach are unknown, as is its potential for detecting attacks outside of GPS spoofing and jamming.

Tan et al. [74] proposed an IDS method using a deep belief network (DBN), optimized by particle swarm optimization (PSO). A DBN is a class of deep neural network which can graphically represent all possible values. The DBN is used to generate a classification model from the dataset, then PSO is applied to optimize the number of hidden layer nodes. This method claims to have higher accuracy compared

to other machine learning approaches such as those using support vector machines and artificial neural networks. The dataset used for experiments is the KDD Cup 99 dataset, which is a well known dataset within intrusion detection research [75]. This dataset was created for intrusion detection on traditional TCP/IP networks. The attacks and communication protocol used in this dataset do not necessarily translate to those within UAV networks. This makes the results of this paper difficult to understand within the context of UAV attacks and further verification is needed.

Khan et al. [76] introduced the use of unsupervised machine learning algorithms to detect anomalies in the UAV. Although the system detects anomalies, it is looking over overall system health, not security incidents. As a useful reference, this paper provides a brief survey of other approaches to anomaly detection in the UAV space. Analyzed methods include isolation forest, One-class SVM, Gaussian mixture model and Mahalanobis distance. To conduct the machine learning experiments, a dataset is required. At the time of writing there wasn't any publicly available IDS related datasets in the UAV space, so the authors created one. Sensor data for a normal flight was taken, however, it was only a 6 minute flight which may not provide the best results in a machine learning algorithm. Synthetic anomalies were created within the dataset and the results showed the success of the anomaly detection. A drawback to this system is that once an alert is triggered, there is no way of knowing why the anomaly occurred. This limits the solution for intrusion detection, as we cannot determine what attack is occurring in order to mitigate it. The performance of the proposed algorithms is also fairly weak, with a maximum detection accuracy of 89.3% with isolation forest.

Hybrid approaches combine multiple types of intrusion detection, such as signature-based, anomaly-based or specification-based, to create an effective ensemble. Muniraj and Farhood [13] discussed a framework for detection sensor attacks on small UAVs. Their framework makes use of anomaly detection using attack signatures generated from "safe sensor" measurements, and anomaly detection based on residuals. The residuals are computed by a residual generator from outputs of a state estimator. Three anomaly detection methods are considered: sequential probability ratio test (SPRT), cumulative sum (CUSUM) detector, and the binary hypothesis testing (BHT) detector. The proposed method shows varying results against GPS spoofing, although its effectiveness against other attacks is unknown. Condomines, Zhang, and Larrieu [77] proposes a hybrid approach for flying ad-hoc networks (FANETs). The method first creates statistical signatures of the traffic within the network by using wavelet leader multi-fractal analysis (WLM), then uses a robust state observer. The results of this approach look promising, however, it is only effective against network-based attacks against the ground control station and UAV links. Sedjelmaci, Senouci, and Ansari [78] also proposes a hybrid approach to IDS in UAV networks. This method operates on both the UAV and the ground control station, in an attempt to detect four types of attacks. Rules are developed for each attack, then determines if the attack is an anomaly using support vector machines on the ground station. The proposed method works quickly as it has pre-defined rules in place, and can minimize false positives due to the secondary anomaly detector. The downside to this is that it requires background research and knowledge on how each attack is detected in order to develop rules. Although the aforementioned methods may be effective in UAV networks, their usability within sole UAV environments has not been proven.

Recently, novelty-based intrusion detection in UAVs using autoencoder neural networks has been introduced for use in UAV IDS. Autoencoders are neural networks that learn a compressed representation of the input in an attempt to reconstruct it. When reconstruction error is high, the input can be seen as a novelty. The difference between novelty detection and anomaly detection is that the outliers do not need to pre-exist in the training dataset. This is advantageous within the UAV IDS domain, as it allows for the use of standard flight logs for training (although this idea is not yet explicitly proposed in the literature). Bae and Joe [79] (2019) briefly proposed the use of a vanilla autoencoder as well long short-term memory (LSTM) autoencoders. The proposed method utilizes a distributed artificial intelligence model which deploys both LSTM and regular autoencoders. Unfortunately, their work lacks significant detail such as flight data acquisition method, feature selection, pre-processing, and performance evaluation against attacks. Park, Park, and Kim [80] (2020) also proposed the use of an autoencoder for UAV IDS. This paper was based on the dataset created for this thesis and a related paper, which was made publicly available to assist fellow researchers [81, 82]. This work manually extracts features from the logs, then uses feature scaling and timestamp pooling to normalize the data. An autoencoder is then trained on the normal data and reconstruction loss is recorded as the indication of an attack above a chosen threshold.

Novelty detection for UAV IDS is a promising approach which requires minimal maintenance and has the potential to accurately detect both known and unknown attacks. This is a new area of research within the UAV IDS domain, and as of writing only the use of autoencoder neural networks have been explored. Further research in this area investigating other one-class classifiers could result in a strong

and versatile solution that is easy to deploy and maintain. One-class classifiers can help solve the unavailability of labelled datasets, introduce the possibility of detecting zero-day attacks, and can trained to understand an underlying distribution of log data for the specific UAV they are operating on. This last step is especially important as there is a wide number of UAV platforms and control configurations which must be accounted for in many of the aforementioned methods. In addition, this approach can be agnostic to the communication protocol being used.

The literature is advancing quickly with contributions being made regularly. There have been a number of novel contributions to the area of UAV IDS, all with the goal of making UAV deployment safer and more secure. Although each piece of literature demonstrates successful detection of specific attacks, they still face a number of common challenges. Many solutions require specific environments in order for the IDS to operate, such as within a FANET. Additional challenges include the lack of testing across UAV platforms and attacks, and the unavailability of appropriate datasets. A novelty detection approach has the potential to address each of these challenges, and deserves further research. As recent literature has introduced the use of autoencoders in UAV IDS, there still remains a number of gaps in research such as the performance of other classifiers, UAV platforms, attacks, and more. Other one-class classifiers that can be evaluated such as local outlier factor and one-class support vector machines. Table 2.2 provides a summary of UAV IDS techniques in literature with key contributions and areas in which they can be improved upon.

Table 2.2: UAV Intrusion Detection Techniques in Literature

Reference	Technique	Key Contributions	Challenges
Sedjelmaci, Senouci, and Ansari [68]	Game theoretic	IDS activated only upon result of Bayesian game to minimize resource consumption	Designed to operate only in MANETs
Sun et al. [67]	Game theoretic	Bayesian game is used to reduce false positives and reduce computational overhead	Designed for UAV networks, not applicable to single UAV deployments
Wang and Feng [66]	Game theoretic	Result of complete information game triggers anomaly detection, reducing false positives and overhead	Game loss results in very low detection rate
Mitchell and Chen [69]	Specification-based	Behaviour rules lead to very high detection rate and low false positives	Can only detect UAVs that have already been compromised, high maintenance
Mitchell and Chen [70]	Specification-based	Behaviour rules fed into state machine for high detection rate and low false positives	Can only detect UAVs that have already been compromised, high maintenance
Fotohi [71]	Human immune system	HIS applied to networked UAVs	Attacks used require specific circumstances to be applicable, high maintenance to generate HIS algorithm
Panice et al. [72]	Anomaly detection	Uses SVDD or v-SVM support vector machines	Only tested in simulation, requires labelled dataset
Arthur [73]	Anomaly detection	Uses multi-class support vector machines	Unknown performance impacts and potential outside of discussed attacks
Tan et al. [74]	Anomaly detection	Deep belief network with particle swarm optimization	Only validated against IDS dataset of traditional attacks, not those facing UAVs

Khan et al. [76]	Anomaly detection	Shows effectiveness of multiple unsupervised algorithms for anomaly detection	Geared more towards overall system health anomaly detection, not security
Bae and Joe [79]	Novelty detection	Autoencoder and LTSM autoencoder	Lacks significant detail in all aspects
Park, Park, and Kim [80]	Novelty detection	Autoencoder with manual feature selection	Requires knowledge of underlying system to select features, not versatile
Muniraj and Farhood [13]	Hybrid	Uses anomaly detection (SPRT, CUSUM, BHT) and signatures	Effectiveness only shown against one attack
Sedjelmaci, Senouci, and Ansari [78]	Hybrid	Defined rules, then anomaly detection with SVM	Maintenance in rule development, processing on GCS
Condomines, Zhang, and Larrieu [77]	Hybrid	Statistical signatures of traffic using WLM	Based on FANETs, detects attacks against data link and GCS only

2.7 Summary

The increase in the usage of UAVs is mirrored by the attacks against them. Unfortunately, many of the vulnerabilities UAVs are plagued with come from security flaws in the underlying technologies they use such as GPS and ADS-B. Given the increased usage of UAVs across industries, including in high risk military and law enforcement operations, the detection and mitigation of attacks is paramount. An IDS for UAVs can increase security detection and response rapidly without the need for the re-engineering of underlying technologies.

With the apparent need of an effective IDS solution, this area has become a hot topic for academic research. Section 2.6 provides a background into the state of the art in UAV IDS. Overall, the proposed solutions fall into one of six categories: game theoretic, specification-based, human immune system, anomaly-based, and novelty-based. Table 2.2 shows the a summary of the literature, including key contributions of the papers and their associated drawbacks and challenges. The results of the literature review survey show a number of key themes that require improvement. First, many solutions are developed on the basis that they will be deployed within networks of UAVs or MANETS. Because of their reliance on a multi-UAV architecture, they fail to protect singular UAVs conducting loan missions. This is often the case in surveillance missions, one of the main military and law enforcement uses of UAVs. Other solutions require similar specific circumstances to be effective or are only tested against a small subset of attacks with no ability to detect novel attacks. Similarly, many approaches require a significant amount of maintenance, such as developing signatures, in order to remain current. Anomaly-based techniques can mitigate the aforementioned challenges, however, they require labelled training datasets which are

hard to come by for a number of reasons. Given that an attack must be recorded for the creation of a labelled dataset, the IDS developer must have the equipment and knowledge to safely and legally carry out the attack in order to obtain the necessary data. There are also barriers in making the IDS ubiquitous as the wide variety of UAV configurations leads to a requirement to create a labelled dataset for each type the IDS is intended to be used on. Recently, novelty-based approaches have been introduced which use semi-supervised machine learning techniques to learn an underlying normality and to detect attacks which do not conform to this norm. Hybrid approaches can also been introduced, which tend to succeed in one area but lack in another.

A novelty-based approach can use one-class classifiers to exploit the use of pre-existing flight logs, which are created by UAVs by default for troubleshooting purposes. This method of intrusion detection would also allow for the potential detection of future attacks, and requires near zero maintenance. These benefits make it a suitable candidate for an IDS which can be applied to a wide variety of UAVs regardless of frame type, control configuration, onboard sensors, communication mediums, or the need for a UAV network. Although novelty-based approaches appear promising, the literature is in its infancy. Currently there are only two works which use this technique, one which lacks significant detail and another which is based upon some of the work in this thesis. The latter technique is also shown to require manual feature selection. The next chapter will introduce a novelty-based approach using one-class classifiers to address the aforementioned challenges, including the lack of available datasets.

Chapter 3

One-Class IDS Approach

Given the takeaways from the current research in the area of UAV IDS development, a novelty detection using one-class classifiers stands out as a promising approach. A detection technique using one-class classifiers has a number of advantages to previous works. First, being a machine learning-based approach, it provides the ability to detect attacks from the underlying data with minimal maintenance and maximum agility. In addition, unlike other machine learning techniques, one-class classifiers do not require a labelled dataset nor one where attacks are contained within the data. This allows for the use of pre-existing flight logs which are readily available in UAVs.

3.1 One-Class Classification

3.1.1 Novelty Detection

An anomaly, or outlier, is defined as a rare observation that is outside of the norm. Using various machine learning and statistical techniques, anomalies can be detected in large datasets as observations that differ from the majority of the data. Anomaly-based machine learning techniques are applied to a number of problems including

fraud detection, system health monitoring, and intrusion detection. Anomalies often represent events of importance, such as abnormal transactions in banking, high CPU usage on a critical system, or an imminent mechanical failure such as higher than normal vibration. Anomaly detection is especially useful in intrusion detection solutions, due to the high dimentionality of network data, user heuristics, etc. Depending on the available data, supervised, unsupervised, and semi-supervised techniques can be applied to the anomaly detection problem. Typically in anomaly detection problems, it is the goal to discover the anomalies within the data. For this reason, supervised techniques are rare in anomaly detection because a labelled dataset is required. If a labelled dataset is available, however, algorithms such as logistic regression can perform the anomaly detection task. More commonly, however, a labelled dataset is unavailable. By using a sample dataset consisting of both normal and abnormal observations, many algorithms such as k-nearest neighbours (KNN) can be used to attempt to successfully classify normal from abnormal observations. In problems where labelled datasets or even those containing anomalies are hard to come by, algorithms such as KNN cannot be applied. This is where semi-supervised techniques can prevail One-class classification is a semi-supervised technique where all data within the training set is assumed to belong to the same class. In one-class classification, we can use a training set containing only normal observations, and assume these observations are part of the "normal" class. One-class classifiers such as local outlier factor or one-class support vector machine can be trained on the normal observation dataset and use various techniques to classify anomalies from this learned "normal". This is different from other anomaly detection techniques as the algorithms are learning a

model of normality, rather than aiming to separate normal observations from coexisting abnormalities. Anomalies detected by one-class classification are often referred to as *novelties* to differentiate them as outliers detected by means of only observing normal data throughout the training process.

Anomaly detection machine learning techniques are attractive to the UAV IDS problem for a number of reasons. First, machine learning approaches have the ability to detect both known and unknown attacks. As UAVs are an emerging technology, their threats and vulnerabilities are ongoing and not fully understood. Machine learning can utilize underlying data to detect intrusions, which if implemented properly, can mitigate the maintenance required with other solutions such as a signature or behaviour-based IDS.

Previous works have been successful in implementing machine learning approaches to UAV IDS, however, they are only effective on specific platforms or in specific environments and scenarios. However, there is a wide variety of sensors, UAV platforms, communication protocols, and control configurations that may be implemented within a specific UAV. This makes the intrusion detection problem even more difficult and has lead to a lack of universal training data for machine learning-based approaches. Maintenance-heavy techniques such as signatures would be unrealistic and expensive to develop due to this massive mix of component and technology usage. This also makes the creation of a common IDS dataset, such as the KDD99 dataset used for traditional systems, unrealistic within the UAV domain [75]. By applying a novelty-based approach to UAV IDS, we can exploit the use of flight logs for training data which are created by default during flight for troubleshooting purposes. This approach removes the barrier of requiring a labelled dataset, and even the need for a

dataset containing the anomalies within it. Using flight logs for training data means the only requirement to acquire this data is a flight in which no attacks occur. One-class classifiers can then be trained on this data to create a model of normality, and new observations outside of this can be classified as novelties and therefore potentially malicious.

One-class classifiers are typically categorized into three types of approaches: density-based, boundary-based, and reconstruction-based. This thesis examines the use of each one of these types of approaches in order to gauge and compare their effectiveness. The following sections will provide some background into how each of these approaches work using a simple example dataset inspired by the *scikit-learn* documentation [83]. The training dataset was created using 200 points of two dimensional "normal" observations. The normal observations are random floating point values which follow a standard normal distribution. Next, a new array is constructed from these values, centering the clusters around -2 and 2 in the X column. This results in the two clusters shown in white in Figure 3.1. The same method is used to create another array consisting of 40 more "normal" observations that fall within the same standard normal distribution. These new normal observations are shown in Figure 3.1 in green. Finally, 20 new "abnormal" observations are created as a uniform distribution over an interval of -4 to 4 as shown in red in Figure 3.1.

3.1.2 Density-based

Density-based approaches to one-class classification look to identify the density of an observation and compares that to its neighbours. With a low density, it is likely to

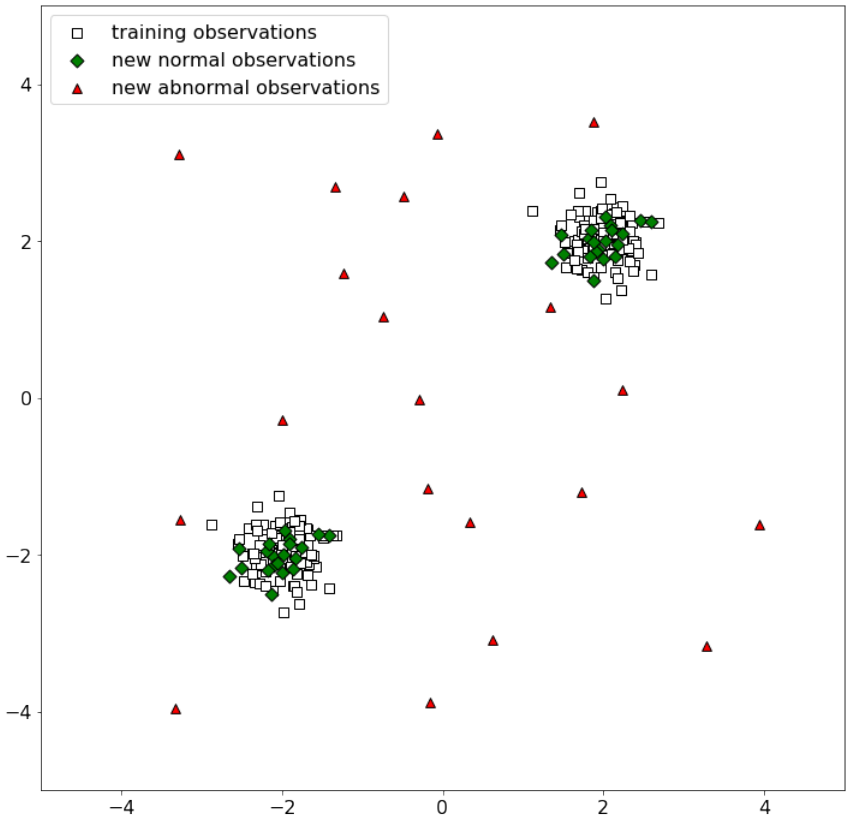

Figure 3.1: Example Novelty Detection Dataset

be an outlier as it is further away from other observations. One common density-based algorithm that can be used for one-class classification is Local Outlier Factor (LOF) [84]. LOF first computes the local density of each observation, where the locality is defined by a hyperparameter representing k nearest neighbours. k is the number of neighbours to consider when determining the local density. This value should be greater than the cluster size, but small enough to allow for encompassing outliers. The *k-distance* is then determined, which provides the distance of a given observation to the kth closest observation. Using the *k-distance* of each observation, we can calculate the reachability distance, which determines the larger of the distance between the two observations and the *k-distance* of the second observation. This is

done to stabilize the results by smoothing fluctuations. Finally, the local reachability density is calculated which determines the distance from the current observation to the next. When this density is lower, the point is closer and vise versa. The local reachability density can be used to compare the current observations density to that of the k neighbours. The resulting local outlier factor is a ratio of how dense a point is compared to its neighbours, and when greater than 1, it is considered less dense. A point that is less dense than those around it suggests it is a novelty. LOF becomes less effective as the data increases in dimensionality. Figure 3.2 shows the same example data points and their relative outlier scores.

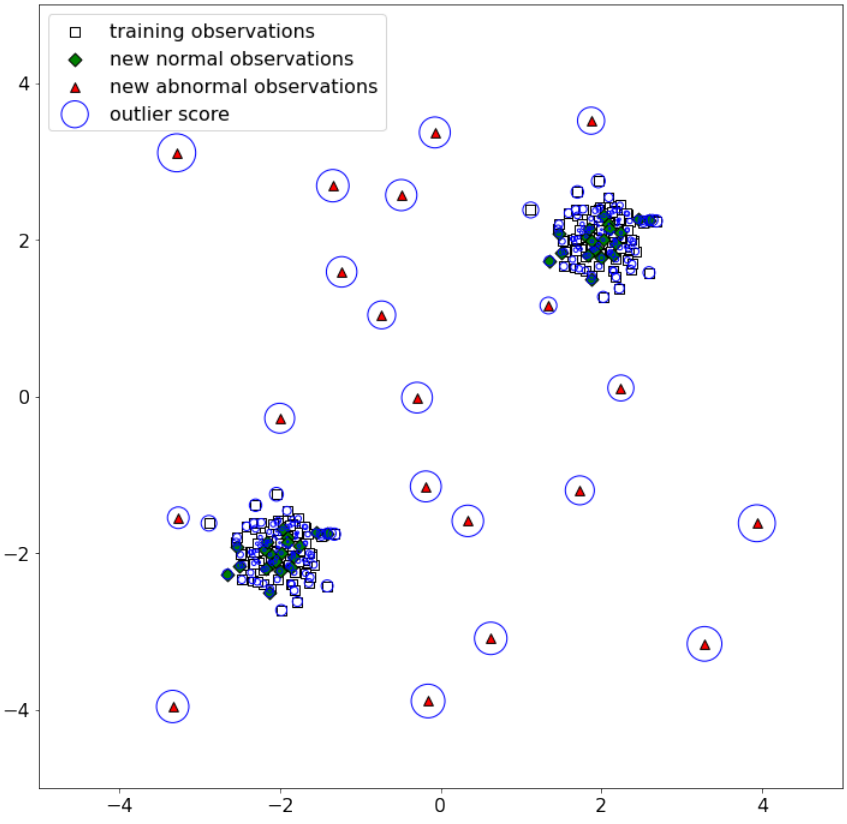

Figure 3.2: LOF Outlier Scores of Example Dataset

3.1.3 Boundary-based

In boundary-based approaches, the algorithm creates a decision boundary which separates the classes. A common example of boundary-based classification algorithms are those that use support vectors. Support Vector Machines (SVMs) are supervised algorithms that can be used for classification and regression. They are effective when used in higher dimensional data, however, overfitting can occur when the number of dimensions is greater than the number of training observations. SVM uses different kernel functions for pattern analysis in order to transform original non-linear data into a new space. For example, the polynomial kernel can be used allow learning of non-linear models by representing the observations within a polynomial feature space. An SVM algorithm is given two sets of data and tries to create a model to separate categories with the maximum margin. The SVM then makes decisions based on which side of the boundary a certain point is placed.

Although SVM algorithms are typically used for multi-class classification or supervised problems, One-Class SVM (OC-SVM) can be used for novelty detection [85]. OC-SVM is an unsupervised algorithm which is trained only on the normal data, learning the density, or support, of the observations in order to separate between normal and abnormal classes. For most one-class classification problems, the radial basis kernel (RBF) is used. OC-SVM using RBF makes use of two hyperparameters, *nu* (ν) and *gamma* (γ). *nu* controls the sensitivity of the novelties and as such should be tuned to be equal to the number of novelties expected to see. With a *nu* of 0.02 the model will only be able to classify up to 2% of the observations and at least 2% of them will become support vectors. *gamma* dictates how far the influence of each

training observation will reach. A smaller *gamma* means more variance so the influence of will reach further, and vice versa. Figure 3.3 shows the example data with the support vector observations shown in blue.

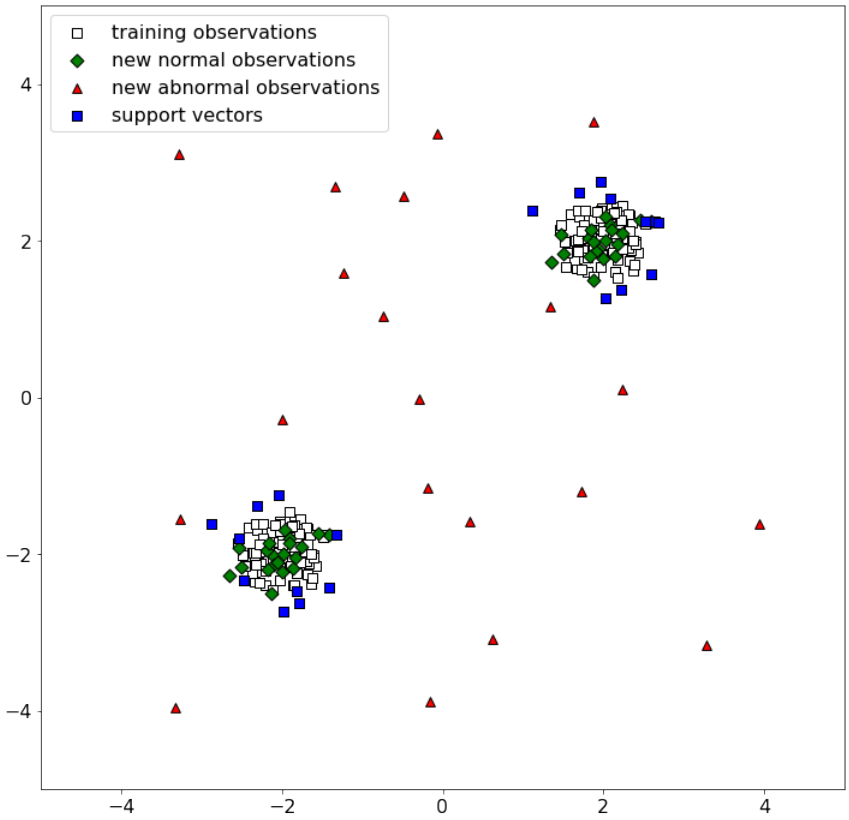

Figure 3.3: OC-SVM Support Vectors of Example Dataset

3.1.4 Reconstruction-based

Reconstruction-based approaches to one-class classification learn from the input then attempt to reconstruct it. One strong technique is the use of an autoencoder. An autoencoder is a type of artificial neural network with an architecture that imposes an informational bottleneck to force a compression of the input. The neural network architecture is usually designed to have the same number of input and output nodes,

with a smaller number of nodes as part of a "hidden" bottleneck layer. Figure 3.4 shows an autoencoder with 5 input nodes, 3 hidden nodes, and 5 output nodes.

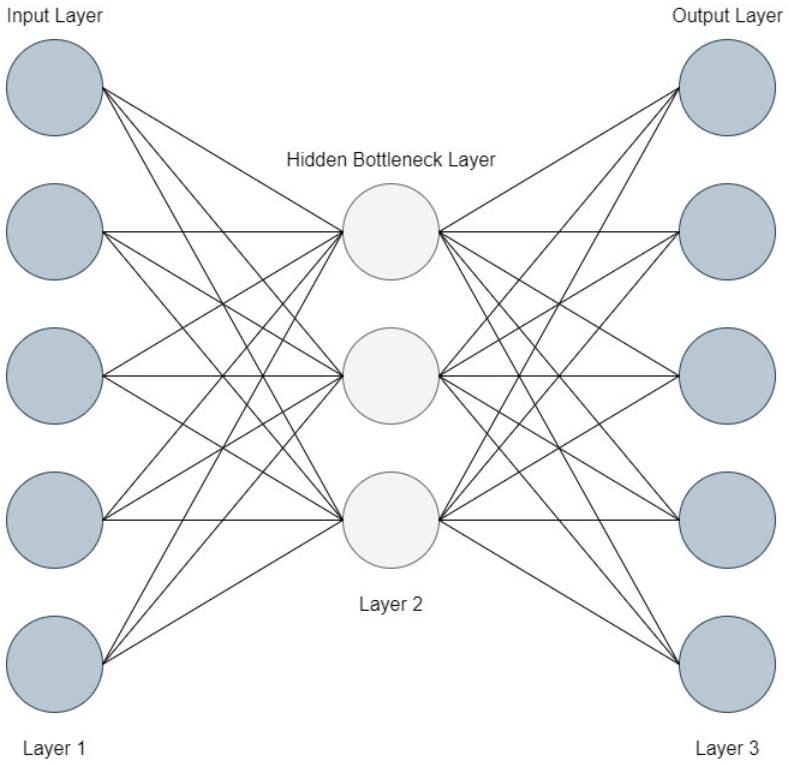

Figure 3.4: Visualization of an Autoencoder Neural Network

Autoencoders function by taking the input, compressing it and trying to reproduce the same input as its output. The autoencoder is comprised of the following four parts: input data, encoding function, decoding function and loss function. The input data is the data that is getting encoded and decoded. The encoding function takes the input data, and encodes it. The decoding function takes the encoded input and decodes it. The loss function is responsible for evaluating how optimal the autoencoder is performing. The autoencoder functions at a high level of success when data is encoded then decoded and the result is very close to the original data. When the reconstruction error is high, the observation can be classified as an anomaly. Classification is usually

done through the use of of a threshold, T, that helps to define a novelty and tune performance. When the reconstruction error is above the given threshold it can be classified as a novelty. Figure 3.5 shows the reconstruction error as mean squared error (MSE) of each observation. A threshold on this example was specified as 70%.

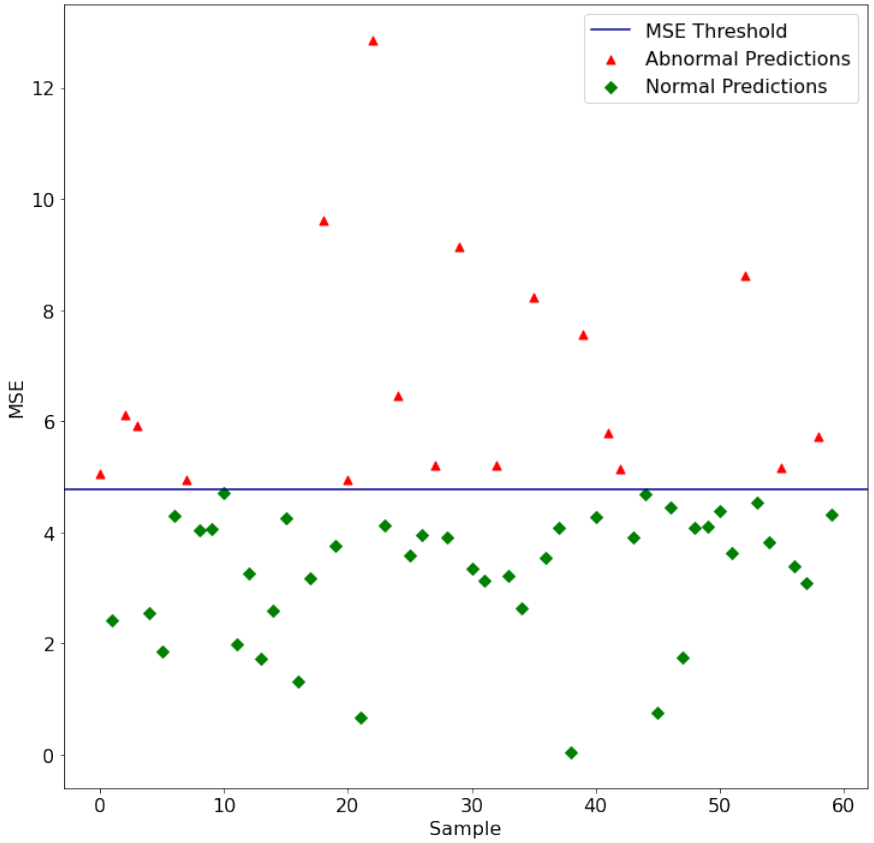

Figure 3.5: MSE of Example Dataset

These classifiers were selected as they each represent a particular classification technique. When applying each classifier to the UAV IDS problem, they can be compared in terms of both detection and computational performance. By using multiple classifiers, the best fit for the problem at hand can be determined.

3.2 Dataset Creation

One of the major challenges in UAV IDS research and development is the lack of an available UAV attack dataset. Creating a dataset provides a number of advantages for research. First, it allows for acceleration of work as research can begin without the need to first create a dataset. Second, it allows for an even comparison of performance results from different research outcomes. As no suitable dataset currently exists, one must be created.

Flight logs are required from missions under different conditions. Logs where no attacks are required for training the proposed system and are simple to obtain. Most UAVs will create these logs by default during each flight for troubleshooting purposes. The more difficult task is obtaining flight logs from the same UAV where an attack is experienced. Due to legal reasons, many of the attacks facing UAVs are not easy to conduct. For example, it is illegal to use or possess an RF jamming device in Canada. This makes real spoofing and jamming difficult. In addition, conducting attacks against a live flying UAV can introduce a number of safety hazards as the attacks can cause unpredictable behaviour.

The literature and previous research show the largest attacks surface towards the UAS are the external sensors on the UAV itself [6, 35, 40, 41, 86, 46]. For this reason, attacks against external sensors are chosen as a focus for the IDS development: GPS spoofing, GPS jamming, and a ping Denial of Service (DoS) attack. GPS spoofing is one of the most common attacks facing UAVs today due to its simplicity and effectiveness using inexpensive hardware [9, 14, 87, 88]. Other prominent attacks are those that cause denial of service on the UAVs sensors, such as GPS jamming. Attacks such as RF jamming and command injection can both lead to denial of

service as well. A hybrid attack causing denial of service is done using a flood of *PING* MAVLink commands. When a flood of ping commands are sent to the UAV, the flight controller must process them causing a bottleneck and resource exhaustion. This large increase in link throughput can also cause the telemetry link to drop packets or timeout completely. This hybrid attack was chosen as it represents jamming, denial of service, and command injection.

Simulated attacks are conducted first as simulations allow for rapid testing during the development phase with minimal financial cost and time investment. Once the IDS detection method has been created, it is then verified through live experiments.

3.2.1 Simulated Experiments

To conduct realistic simulations, both hardware-in-the-loop (HITL) and software-in-the-loop (SITL) architectures are used. Using HITL and SITL simulation provides a number of advantages. First, different UAV types can be tested without needing to purchase each type. Simulated models of different types of VTOL, multicopters, planes, and fixed-wing UAVs can be used without needing the physical hardware of each one. Simulated flights can take place at any location and at altitudes that would otherwise be unsafe. As attacks against the UAV can cause unexpected behaviour, simulation can mitigate this risk. Conducting GPS spoofing and denial of service HITL and SITL simulation are used as the basis for developing and testing a detection method.

To conduct HITL and live experiments, a physical UAV is needed. This UAV must allow for various sensors to be outfitted and must be based on an open platform to streamline IDS testing and development. The chosen technologies used for the

research and development throughout the thesis are part of the Dronecode ecosystem [89]. The Dronecode Foundation is a collaborative project with the Linux Foundation, and encompasses open source projects for each component of a UAS. This includes the flight control firmware, ground control station software, and communication protocol. Dronecode is sponsored and used by a number of prominent companies in the industry including Microsoft, NXP, DroneDeploy, Yuneec, and Intel.

The Droncode flight stack consists of the PX4 flight control firmware, various communication protocols including MAVLink, UAVCAN, and RTPS, as well as the QGroundControl ground control station software. Dronecode and PX4 were chosen over Ardupilot due to their perceived dominance in the commercial UAV development space. Although Ardupilot is another open source flight stack, the Dronecode ecosystem includes the MAVLink protocol and also offers more options for SITL and HITL simulation environments and UAV types. PX4 supports autonomous flight across a number of different UAV platforms including multi-copters, planes, and VTOLs. PX4 was created using a modular and extensible design, supporting various hardware flight controllers and onboard components. The firmware has been tested by the community and commercial systems through thousands of hours of flight time. In addition, the firmware contains interoperability to other standardized UAV technologies as well as a number of safety features required for commercial flight.

MAVLink is an open source lightweight communication protocol and part of the Dronecode ecosystem. MAVLink uses a publish-subscribe model to allow for communication of various topics between the UAV and the ground control station. These topics can include data such as various autopilot configuration parameters, autonomous mission plans, telemetry information, and more. MAVLink is a popular

choice amongst both hobbyists and commercial UAV developers as the protocol is open, salable, and reliable. The protocol allows for communication with up to 255 systems, making it salable with multi-UAV deployments. For resiliency, the protocol has the ability to detect packet loss and message corruption. For our testing, it is important to use a protocol that supports a wide variety of onboard components. MAVLink can be used for communications between both offboard and onboard systems, including the configuration and communication the ADS-B sensor, GPS, and various telemetry radios. Being extensible, MAVLink provides a software development kit called MAVSDK. MAVSDK contains libraries for multiple programming languages making interfacing with MAVLink enabled systems easier. MAVSDK can then be used to pull and set parameters within the IDS GCS client, and can also allow for the communication between the onboard agent and the autopilot.

QGroundControl is ground control station for MAVLink enabled UAVs. It is open source as part of the Dronecode ecosystem, and is available for mobile and desktop environments across multiple operating systems. QGroundControl can facilitate the setup of various UAV types, including calibrating of the compass, ESCs, and various sensors. In addition, it can be used for planning autonomous missions, as well as communicating telemetry data with the UAV during flight.

A UAV is needed for HITL and live experiments. Multi-copter UAVs are simpler to operate in a small controlled environment and thus this type of UAV was chosen. A quadcopter using the latest Pixhawk 4 flight controller was constructed using the Holybro S500 frame. To provide a diverse platform for vulnerability testing, the UAV is also equipped with a number of components, including:

- Pixhawk 4 Flight controller running PX4 1.11.3

- Pixhawk 4 GPS receiver
- Holybro SiK 915MHz telemetry radio
- FS-IA10B 2.4GHz RC receiver
- WiFi MAVLink bridge
- uAvionix Ping ADS-B receiver
- PXFlow optical flow smart camera
- Raspberry Pi Zero companion computer

The UAV and its components are shown in Figure 3.6.

The design and development of a UAV IDS requires the ability to rapidly conduct flights and test results. Many of the flights need to include attacks against the UAV as well, which can be illegal and dangerous. Both of these requirements make it difficult to use live flights as a basis for testing and development. Both software-in-the-loop (SITL) and hardware-in-the-loop (HITL) simulations can be used to create an accurate test bench for UAV IDS development. With SITL simulation, the PX4 firmware runs on the simulation computer and interfaces with a simulator for environmental feedback. In HITL, a similar setup is used, however, the PX4 firmware is running on a real hardware flight controller. The Gazebo simulator is used for environment simulation for both SITL and HITL experiments [90]. Gazebo is a common open source robotics simulator used by researchers and industry for the development and testing of robotic vehicles. PX4 offers official support for the Gazebo simulator for both SITL and HITL. Although other simulators are also supported, Gazebo provides the most versatile number of vehicles including planes, multi-copters, tailsitters, and

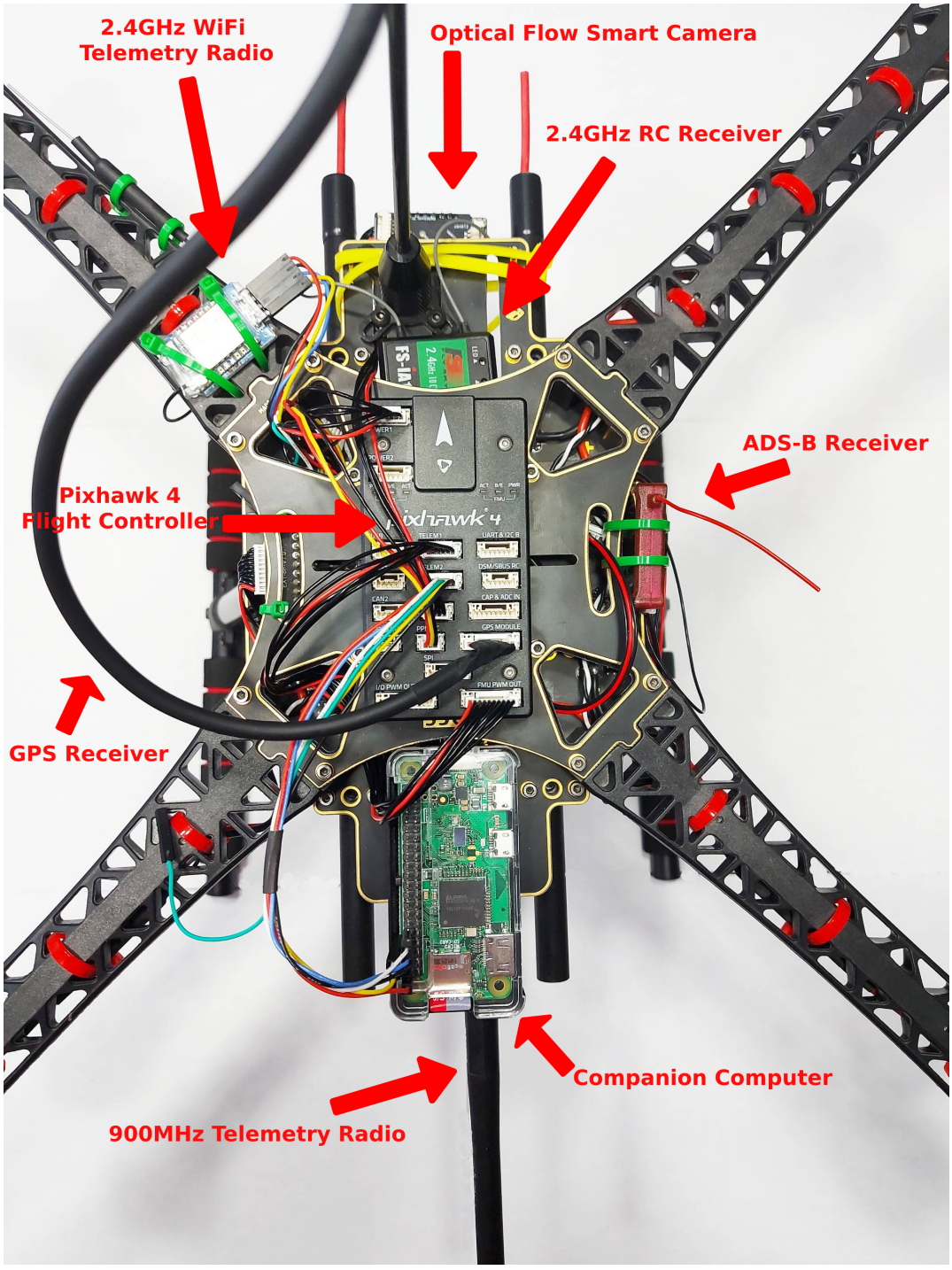

Figure 3.6: S500 Custom UAV Used for HITL and Live Experiments

VTOL UAVs. Gazebo also supports the development of plugins where additional functionality can be built and interfaced with the simulator. This is useful for the UAV IDS testing and development, as it can be helpful in simulating attacks from the environment. Gazebo uses *world files* to define the environment including which model to deploy, various lighting sources, sensors and objects such as trees or people. Models are objects with various properties and are defined by *model files*. The models can be anything dynamically loaded into the environment such as the UAV or other objects within the environment. Both the *world files* and *model files* are formatted using the Simulation Description Format (SDF). Gazebo uses a client-server model, where the server simulates the specified world and models and the client provides a graphical user interface and a virtualization of the simulated environment. Gazebo can also make use of plugins which are a simple way to interface with the server. Plugins are also defined using the SDF format.

The standard Gazebo-PX4 simulation environment consists of the build of the PX4 firmware compiled for use with SITL or HITL. This build will include code to interpret messages to and from the simulator through the MAVLink API. Control signals and telemetry information are communicated between the GCS and the PX4 autopilot. Sensor inputs are communicated from the simulated environment within Gazebo to the PX4 autopilot and the actuator control outputs are returned. Figure 3.7 shows a high level overview of the simulation environment.

The simulation environment follows the standard setup recommended by the PX4 documentation. The simulation host computer is running Ubuntu 18.04 and the requirements of the simulation environment are installed using the provided *ubuntu.sh*

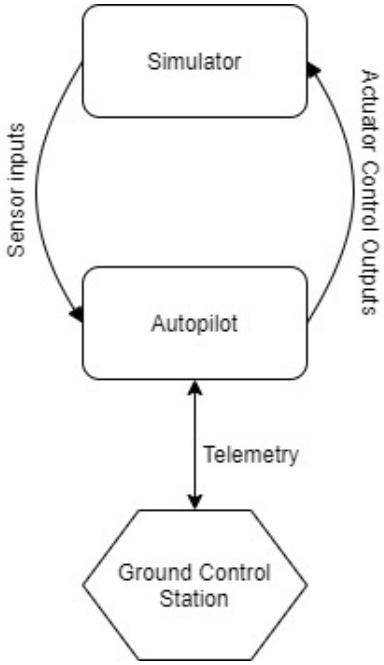

Figure 3.7: Simulation Environment

development script. This script will install Gazebo 9 as well as the required dependencies and tools. The PX4 firmware can then be cloned from Github recursively in order to also download the PX4 Gazebo tools and worlds:

```
1 git clone https://github.com/PX4/PX4-Autopilot.git --branch v1.10.1 --recursive
```

The PX4 firmware is then compiled specifying PX4_SITL and the target UAV model. For example, compiling the PX4 firmware to simulate the Yuneec Typhoon H480 the following *make* command would be used:

```
1 make px4_sitl_default gazebo_typhoon_h480 no_sim=1
```

Once the PX4 firmware is compiled and run, the Gazebo environment must be initialized. This is done with the help of the PX4 *setup_gazebo.bash* script and the pre-configured *world file* for each model. The setup script is used to export the

environment variables which specify the paths to the plugin, model, and library directories. A starting latitude, longitude and altitude can be defined as environment variables. For each simulated flight, the center of the open commons at Ontario Tech University is specified with an initial altitude of 50 metres. Finally, Gazebo can be run specifying the world to use:

```
export PX4_HOME_LAT=43.945155
export PX4_HOME_LON=-78.896851
export PX4_HOME_ALT=50
source Tools/setup_gazebo.bash $(pwd) $(pwd)/build/px4_sitl_default
gazebo Tools/sitl_gazebo/worlds/typhoon_h480.world --verbose
```

In SITL, the PX4 firmware is run after it is compiled along with the Gazebo simulator. The PX4 flight stack will then communicate with the simulator sending control outputs, while Gazebo will respond with sensor inputs. The telemetry information from the autopilot is communicated to the GCS, and the GCS will send command and control data to the simulated UAV. A HITL is similar, but requires additional setup of the physical UAV. PX4 supports a lesser number of UAV frames for HITL, and the appropriate frame type must be selected during UAV setup. Sensors such as the compass and accelerometer must be calibrated as if the UAV were being setup for a real flight. The physical UAV hardware is connected to the simulation computer, then the Gazebo simulator is launched. Figure 3.8 shows the Iris Quadcopter flying within Gazebo.

An autonomous survey mission of the University is conducted for each simulated flight. The survey is approximately 20 minutes of flight time depending on the UAV type. For example, a quadcopter during the survey travels at 5 metres per second, whereas a fixed wing UAV such as a plane will travel faster. With a higher velocity, the flight time will be reduced. The survey mission was created to provide enough

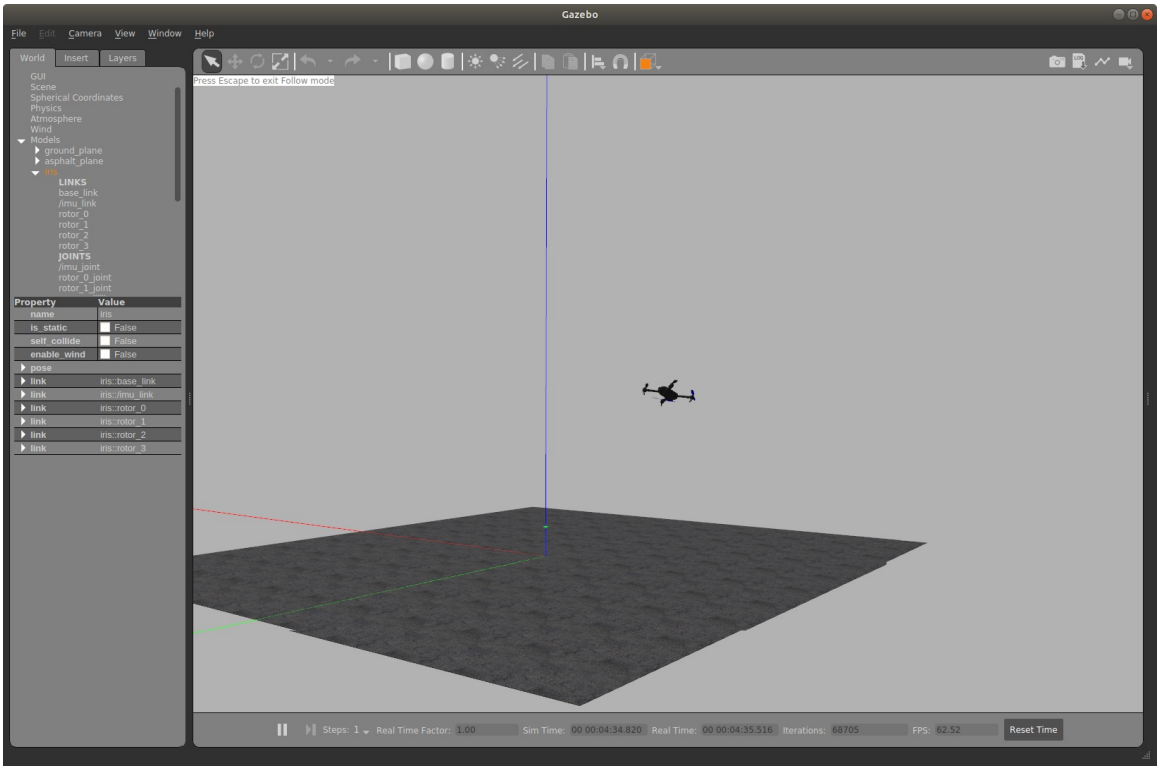

Figure 3.8: Gazebo Simulation of Iris Quadcopter

time to collect sufficient data within the flight logs for machine learning training. Using the same survey mission for each flight ensures a consistent comparison of results between UAV types. Fixed wing UAVs, however, require a takeoff and land flight path to be added to the mission. This specifies the takeoff and approach the UAV will take during takeoff and landing. Figure 3.9 shows the autonomous survey mission for multicopter UAVs, whereas Figure 3.10 shows the same with the takeoff and landing flight path added.

A common limitation to previous works in the area of UAV IDS is the scope of the UAV types used. Due to differences in flight dynamics, control configurations, and motors onboard the UAV, it becomes important to verify the IDS effectiveness across different UAV types. For this reason, experiments are conducted using a variety of

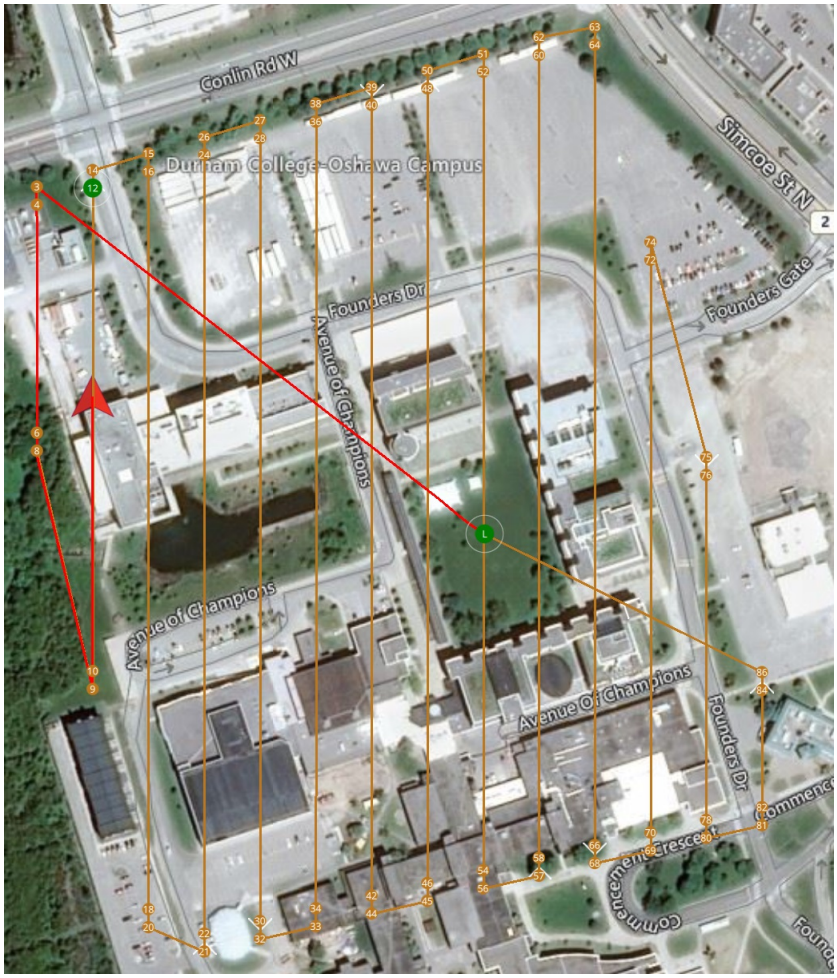

Figure 3.9: OTU Survey Mission - Multicopter

UAV models and simulation types. Both SITL and HITL are used where supported by PX4 as well as live fights. The list of platforms, UAV models, and experiment types conducted are shown in Figure 3.1.

Attack Simulation

It is known from previous research that the attacks against the UAVs sensors are the most vulnerable to threats [6, 35, 40, 41, 86, 46]. This is because they are the

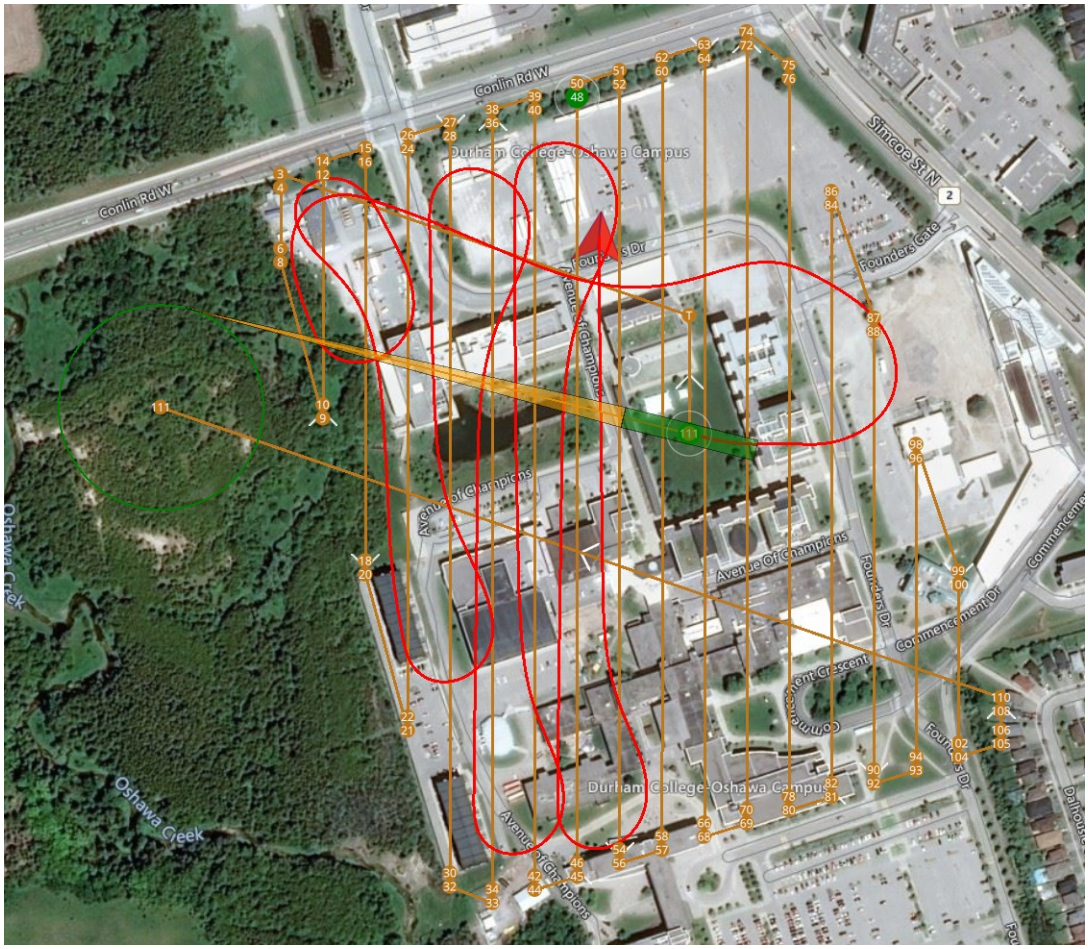

Figure 3.10: OTU Survey Mission - Fixed Wing

Table 3.1: UAVs Used for Experiments

Platform	Model	Experiment Type
Quadcopter	Holybro S500	Live Flight
Quadcopter	Holybro S500	Hardware-in-the-loop
Quadcopter	3DR IRIS+	Software-in-the-loop
Hexacopter	Yuneec H480	Software-in-the-loop
VTOL	DeltaQuad VTOL	Software-in-the-loop
Tailsitter	Standard Tailsitter	Software-in-the-loop
Plane	Standard Plane	Software-in-the-loop

most accessible component within the UAS and generally accept and trust input from the environment. Examples of these sensors include the GPS receiver, optical flow sensor, IMU, telemetry radios, etc. Attacks against the onboard sensors also have the potential to cause the most damage during a mission. Attacks such as GPS spoofing can send the UAV off course, and denial of service or jamming attacks can overwhelm the data link causing it lose communication with the GCS. In non-autonomous UAVs, losing communication with the GCS often means a complete loss of control as it relies upon this communication for flight commands. Autonomous UAVs can also suffer from these attacks as they can lead to resource exhaustion within the flight controller causing the delay or denial of command processing. For this reason, the developed IDS method aims to detect and potentially mitigate attacks that target wireless sensors onboard the UAV.

Two of the most common attacks against UAVs are spoofing and jamming attacks [7]. To limit the scope of this thesis, these three attacks were chosen as examples for the performance evaluation and benchmarking of UAV intrusion detection techniques: GPS spoofing GPS jamming, and data link ping DoS. For the initial simulated testing, these attacks can be realistically simulated within both SITL and HITL environments. Given the simulation environment from Gazebo-PX4 simulation environment, a new subsystem can be created from which attacks are generated. Depending on the attack, the malicious data will sent to the target sensor. The data link ping command denial of service attack must be sent to data link between the GCS and the autopilot, causing an overload of system resources and a decrease in link bandwidth. Attacks targeting the GPS sensor will target the Gazebo simulated environment where those signals will then be used as input by the autopilot. The attack simulation environment is

shown in Figure 3.11. Once initial testing is complete, the results can be verified by conducting live GPS spoofing and comparing those results to the simulated GPS spoofing. In addition, a new attack, live GPS jamming is also conducted.

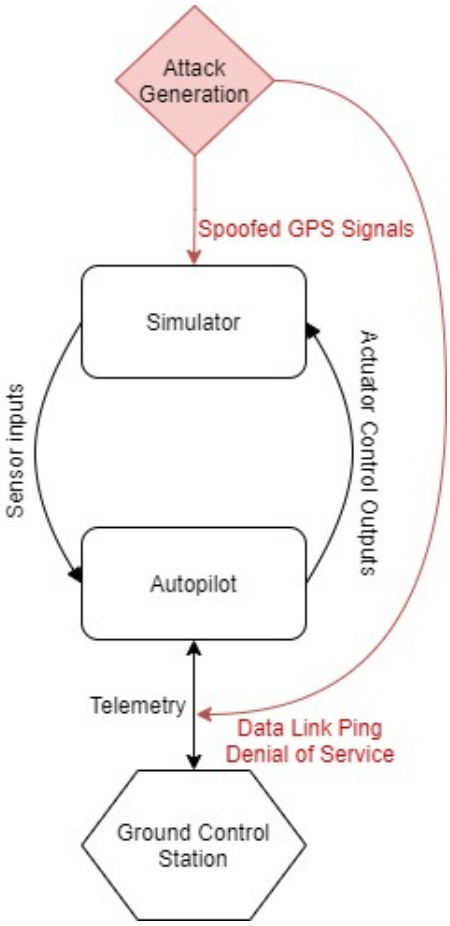

Figure 3.11: Attack Simulation Environment

Simulated GPS Spoofing GPS spoofing is a very common threat to UAVs, as most operate within the unencrypted public Global Navigation Satellite System (GNSS). Adversaries can use inexpensive hardware to broadcast spoofed GPS messages. If the spoofed broadcast can overpower the legitimate signals, the UAV may

estimate incorrect positioning and attempt to correct itself. When done with precision, the attacker can effectively high-jack the UAV. A simple and unsophisticated attack can cause the UAV to crash.

PX4 includes a Gazebo plugin that is used to communicate GPS data from the simulation environment to the autopilot. This plugin is part of the *PX4-SITL_gazebo project* [91]. This project includes the Gazebo plugins needed for PX4-Gazebo communication of various components, such as the IMU, GPS, camera, and optical flow. These plugins interface with the PX4 firmware, exchanging data between the simulated environment and the autopilot. In the case the GPS, the autopilot will pick up GPS readings coming from the Gazebo GPS plugin. This creates an opportunity to simulate the GPS spoofing by modifying this plugin and injecting spoofed coordinates during the mission. The PX4 GPS plugin source file, *gazebo_gps_plugin.cpp*, and the header, *gazebo_gps_plugin.h*, are modified. Additions are made to the code to subscribe to a specific topic, and upon receipt, to "broadcast" modified GPS coordinates continually until the topic is no longer received. After the code is modified, the PX4 firmware is recompiled. A client application is then created which will publish the new topic to the plugin. Successful GPS spoofing requires the attacker to stop the victim from receiving legitimate GPS signals [51]. To satisfy this requirement, the plugin will check if the spoofing parameter is set before publishing spoofed coordinates. This causes the UAV to lock to the new coordinates, and a GPS spoofing situation occurs. When the UAV is conducting a mission, it will change its trajectory to get back to its target, resulting in trajectory modification/hijacking. For the experiments in this thesis, the spoofed coordinates are the original coordinates plus a pre-defined offset. The attack is conducted for 30 seconds to keep the experiments consistent, as this is

enough time to cause the UAV to change course drastically.

Telemetry Denial of Service RF jamming is a common threat to the availability of the UAV. In a jamming attack, the adversary broadcasts a stronger signal which will overpower any legitimate signals the UAV is trying to receive. This will disrupt any communication over the data link causing it to become unusable. If the UAV relies on this data link for command and control or telemetry information, it may lead to a loss of autonomy or a crash. In Canada, however, it is illegal to conduct RF jamming in open space. An alternative attack is one conducted through command injection. A command injection attack can cause a similar threat to the availability of the UAV by flooding the UAV with MAVLink *PING* messages, causing data link packet loss and flight controller resource exhaustion. This hybrid attack of using command injection to cause denial of service through the telemetry data link allows for simulating an attack on availability without the need to conduct live RF jamming.

MAVLink *PING* messages are used to calculate the latency of a connection within the UAS. The system initiating the ping will send the *PING* message which contains a timestamp, sequence number, target system, and target component. The timestamp is the current system time when the message is sent. The sequence number is incremented for each message sent and is used along with the timestamp to calculate the round-trip time of the message. In addition to these two fields, the *PING* message also specifies the target system and target components to be pinged. The systems that are pinged will send ping responses with the targets set to their specific component or system ID.

To simulate a denial of service attack, a command injection and flood of *PING* messages is done. A flood of *PING* messages specifying zero for the system and

component IDs will request all systems and components to reply. This will inevitably lead to resource exhaustion within the flight controller as the system tries to keep up with the flood of messages in addition to benign MAVLink messages needed for normal operation of the UAV. With the flood of messages, the telemetry link will also become congested leading to dropped packets. Conducting the attack is done by creating a Python script which sends 200,000 MAVLink *PING* messages to the autopilot. This number of messages causes the UAV to struggle to keep up with benign flight commands and will lose control or crash. The following Python script sends the commands to the autopilot targeting all systems and components:

```
1  import time
2  from pymavlink import mavutil
3  mavutil.set_dialect("standard")
4  autopilot = mavutil.mavlink_connection('udpout:127.0.0.1:14570')
5  msg = None
6  i=1
7  while(i<=200000):
8       autopilot.mav.ping_send(int(time.time() * 1000), 0, 0, 0)
9       print("Sent ping %d" % i)
10      i+=1
```

3.2.2 Live Experiments

Disclaimer: *Live experiments were conducted at a University research facility within a Faraday cage. Check local laws before attempting to replicate the experiments described in this section.*

Although simulated attacks can provide an accurate representation of how well the IDS performs, live experiments are important to verify these results. GPS spoofing and jamming were chosen as they are common attacks which can be conducted using a fairly inexpensive software-defined radio (SDR). Broadcasting live GPS signals as

well as jamming them is illegal in many countries. To conduct the experiments safely and legally, they are carried out within the Automotive Centre of Excelleint (ACE) research facility at Ontario Tech University. The UAV is tethered to the floor at all times and flown within a Faraday cage. Figure 3.12 shows the UAV flying within a chamber at ACE.

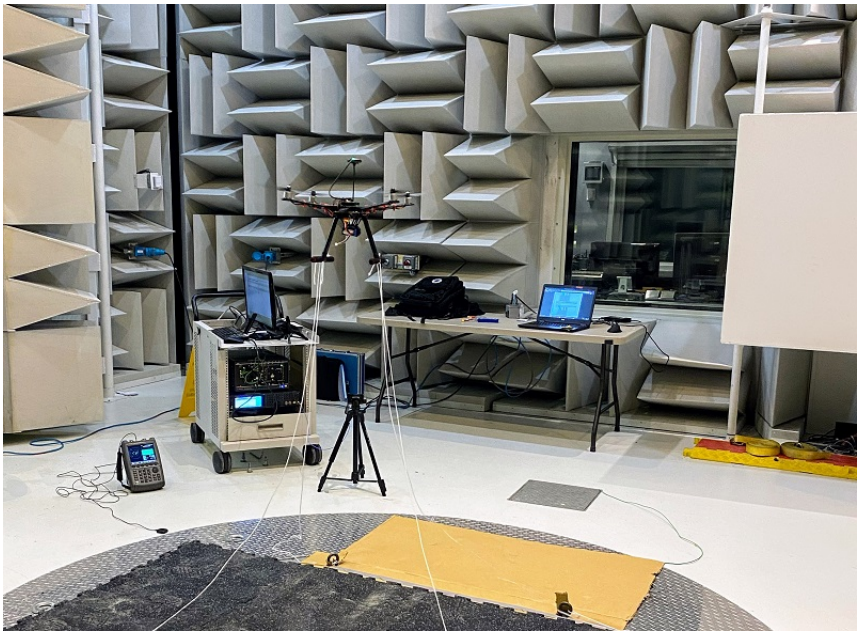

Figure 3.12: Live Experiment Within ACE

The Great Scott Gadgets HackRF SDR is used with an ANT500 antenna to conduct the attacks as it can broadcast within the GPS bands. The HackRF is shown in Figure 3.13. As the experiments are conducted within a Faraday cage, the UAV is unable to receive regular GPS signals. To provide the ground truth for the experiments, the Keysight EXG N5172B signal generator is used. This device broadcasts GPS signals to a location in Shanghai, China, a location pre-loaded in the Keysight software. Once running, the UAV is able to view up to thirteen "satellites" and obtains a lock. The signal generator is shown in Figure 3.14. All experimental

flights are conducted in position mode, where the UAV relies on a stable GPS signal. Fail-safes related to GPS are disabled so the UAV does not divert into a manual mode. Before any attacks are conducted, the UAV does a benign flight which is used later for machine learning training.

Figure 3.13: HackRF SDR with ANT500 Antenna

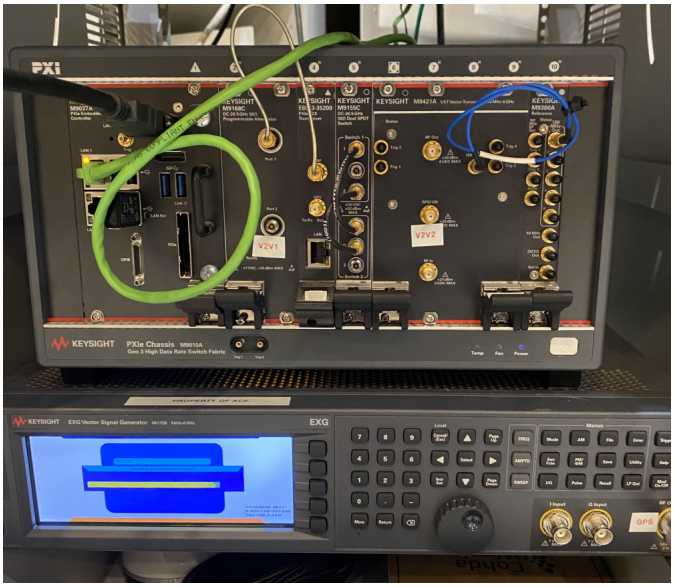

Figure 3.14: Keysight EXG N5172B Signal Generator

GPS Spoofing Once the UAV has completed the training flight, the attack experiments can begin. The GPS-SDR-SIM tool is used to generate GPS baseband signal data streams [92]. A daily GPS broadcast ephemeris file is downloaded from NASA and provided to the tool for generation [93]. The standard static coordinates are used as the spoofed location:

```
./gps-sdr-sim -e brdc3540.14n -l 30.286502,120.032669,100
```

Once the binary data stream is generated, it can be transmitted by the HackRF:

```
hackrf_transfer -t gpssim.bin -f 1575420000 -s 2600000 -a 1 -x 0
```

The attack is started after the UAV has flown for a few minutes. During each experiment, the UAV drifted around its initial hovering position, until the drifting became more drastic and the UAV crashed.

GPS Jamming GPS jamming occurs when RF noise is introduced which prohibits the target from receiving legitimate signals. Using the GNU Radio Companion, a flowgraph can be created to emulate a jamming signal. Previous work on effective GPS jamming have shown success using white Gaussian noise with an amplitude of 0.3 and a gain of -48dB [88]. To broadcast the signal, an osmocom sink is added and configured for the HackRF. A GUI time sink is also added to help visualize the output signal. The created flowgraph is shown in Figure 3.15 with the GUI output shown in Figure 3.16. Similar to the GPS spoofing attack, jamming causes the UAV to become unstable and crash.

3.2.3 Data Capture and Flight Log Extraction

Two types of logs are created during each flight: telemetry logs and system logs. The telemetry log is stored in TLog format and contains telemetry data such as mission

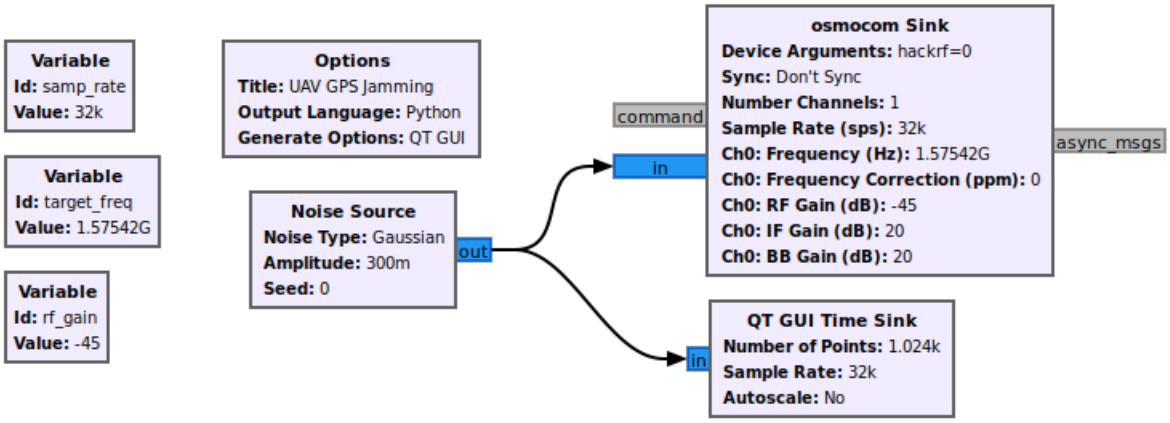

Figure 3.15: GRC GPS Jamming Flowgraph

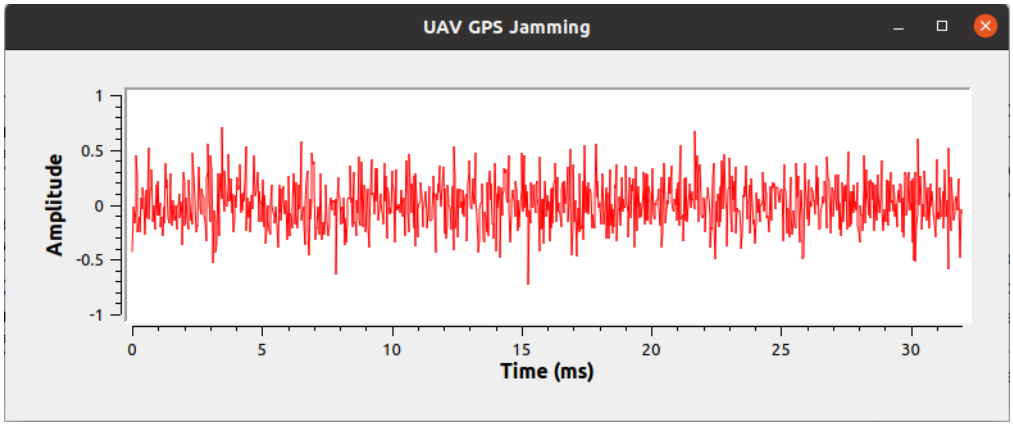

Figure 3.16: GRC Gaussian Noise Output

waypoints, vehicle speed, to list a few. The system logs are stored in ULog format and contains system data such as sensor readings. The ULog system log is written to the SD card onboard the flight controller. This is important as it can record system data when the UAV is disconnected from a jamming or denial of service attack. For the proposed system, the ULog file is of interest as it contains sensor readings and is the standard logging type for PX4. After each flight, the logs are downloaded from the flight controller and are extracted into CSV files. The CSV files are generated using

the *ulog2csv.py* script from the *pyulog* library [94]. This script is used to extract the parameters from the definition section of the log by type resulting in multiple CSVs.

The logs are downloaded from the UAV after each flight, and timestamps are removed from all logs before training. Many, but not all log entries contain a time feature and they may reference different times. For example, one log may use up-time as a timestamp, whereas another may reference the systems local time. Due to this discrepancy, the time is removed to insure those values do not introduce a variance bias during pre-processing, as they have no effect as individual data points on the UAVs operation. In addition, the chosen one-class classifiers do not address time-series data.

Attack start and end times are recorded in order to assist in the labelling process. As we are using a one-class approach, labelling is only necessary for performance analysis. A deeper description of the pre-processing steps is outlined in Section 3.3.1. Tables 3.2, 3.4, and 3.3 show the resulting number of benign and malicious data points for the simulated GPS spoofing, ping telemetry denial of service, live GPS spoofing, and live GPS jamming attacks respectively.

Table 3.2: Dataset Description - Simulated GPS Spoofing

UAV Model	**Benign**	**Malicious**	**Size (MB)**
3DR IRIS+	305140	6596	295.5
Holybro S500	349722	7164	338.6
Yuneec H480	54377	1123	46.8
DeltaQuad VTOL	18308	1111	16.2
Standard Tailsitter	17921	1113	16.3
Standard Plane	23198	1055	20.1

Table 3.3: Dataset Description - Ping Telemetry Denial of Service

UAV Model	Benign	Malicious	Size (MB)
3DR IRIS+	4466	18	1.1
Holybro S500	4444	19	1.1
Yuneec H480	5561	18	1.3
DeltaQuad VTOL	3060	49	0.491
Standard Tailsitter	2971	52	0.471
Standard Plane	1923	17	0.472

Table 3.4: Dataset Description - Live GPS Spoofing

UAV Model	Benign	Malicious	Size (MB)
Holybro S500	6078	498	2.4

Table 3.5: Dataset Description - Live GPS Jamming

UAV Model	Benign	Malicious	Size (MB)
Holybro S500	6078	1460	4.4

Flight Log Format

UAVs write to log files during flight to assist with troubleshooting and investigation when an incident occurs or a component malfunctions. These logs can also be utilized as training data for a one-class classification approach to intrusion detection. The autopilot firmware used in the experiments, PX4, supports two types of logs: telemetry and system data.

TLog Telemetry logs within the MAVLink ecosystem are typically created in TLog format. These logs are created by the GCS, such as QGroundControl, by recording MAVLink messages between the GCS and the autopilot. Telemtry logs are focused on recording the telemetry information during flight in the form of MAVLink communications. Examples of this type of data is vehicle velocity, GPS coordinates, mission state, etc. Although these logs are useful for troubleshooting, they do not contain

as much information about the data flow within the autopilot itself. For this reason, these logs are not used in the training of the IDS. The TLog telemetry log files are saved after each flight for future reference if telemetry data is needed for other research using the dataset.

ULog Apart from telemetry logs, UAVs will also typically record a log of system-related data. The PX4 autopilot uses uORB messaging for communication between onboard processes and components. More information on uORB is discussed in Section 2.1.2. The autopilot writes uORB topics to a ULog formatted file to the SD card within the flight controller. This is log of choice for machine learning training within the IDS because it contains deeper data about the operation of the sensors and flight controller, and also can continue to log even when communication to the GCS is lost due to an attack. The ULog file is in a binary format consisting of header, definition, and data sections. The header contains the file magic number, log version, and timestamp. The definition sections contains the logged attributes and values themselves. The data section contains informational, debug, warning, and emergency information sent from the autopilot to the GCS. ULog data logging begins once the UAV is armed and stops logging when it is disarmed. Typically the UAV disarms once it lands, however, a disarm can also be triggered as an emergency stop if the UAV is out of control. The ULog file is downloaded from the flight controller after each flight and extracted into CSV files using the *ulog2csv.py* script. This script extracts the parameters from the definition section of the log by type resulting in multiple CSVs containing logged messages. These CSV files are then pre-processed and used for machine learning training.

3.3 Training & Tuning

Due to the vast number of potential sensors onboard a UAV and other variations, a data extraction and pre-processing method must be deployed which can effectively gather features and reduce their dimentionality while keeping those with the most potential influence. One-class classifiers are used to exploit the use of the already existing flight logs for training data. Three one-class classifiers are evaluated: Local Outlier Factor, One-Class Support Vector Machine, and an Autoencoder neural network. These three classifiers respectively represent each of the one-class classification techniques: density-based, boundary-based, and reconstruction-based. Other classifiers such as isolation forest were also evaluated, however, due to poor performance on the high dimensional data they were excluded.

3.3.1 Pre-processing

Once flight logs are downloaded from the UAV after a flight where no attack occurs, the pre-processing can begin. As with any machine learning approach, relevant features must be selected. Due to the vast number of sensors that could be onboard the UAV as well as different flight controller firmware, the features within the flight log can change. Additionally, manual feature selection requires knowledge of how an attack will affect the features in question. This makes manual feature selection difficult and requires costly maintenance. For this reason, all available features surrounding a sensor are used and their dimentionality is reduced before training begins.

Both the previously discussed TLog and ULog formats are available from the UAVs used for simulation. The TLog file contains telemetry information such as velocity, altitude, current coordinates, etc. This file is primarily used to log where

the UAV travelled throughout its flight. ULog format logs, however, keep more detailed data about the onboard components such as sensor values, various vehicle states, and environmental data. This makes the TLog files the better choice between the two types of logs. After the training flight where no attack occurs is complete, the logs can be downloaded from the flight controller. The logs are then extracted into comma-separated value (CSV) files based on uORB topics. A Python script provided by PX4 is used for the extraction of the flight logs into multiple CSV files. This script creates one CSV file per uORB message with the associated fields within. Pseudo code of the ULog to CSV extraction script is show in Algorithm 1.

Algorithm 1: ULog to CSV Extraction

```
Result: CSV per uORB message
Initialization;
for Data entry in flight log do
    Make timestamp the first field;
    Insert header row of fields;
    for Each entry do
        for Each message field do
            Insert comma separated data into row;
        end
    end
    Write CSV of message;
end
```

The result of the flight log extraction is multiple CSV files, each related to a specific sensors or element of the UAV. As the attacks will target a specific sensors such as the GPS receiver or telemetry system, any CSV related to the sensor of interest is used for future processing. All features within the sensor-specific CSV are kept in order to avoid manual feature selection, although they will go through dimentionality reduction later on in the process. Table 3.6 show the GPS-specific CSV files and

Table 3.6: GPS System CSVs and Features

CSV File	Contained Features
vehicle_local_position_0.csv	timestamp, ref_timestamp, ref_lat, ref_lon, x, y, z, delta_xy[0], delta_xy[1] ,delta_z ,vx, vy, vz ,z_deriv ,delta_vxy[0] ,delta_vxy[1] ,delta_vz ,ax ,ay ,az, yaw, ref_alt, dist_bottom, dist_bottom_rate, eph, epv, evh, evv, vxy_max, vz_max, hagl_min, hagl_max, xy_valid, z_valid, v_xy_valid, v_z_valid, xy_reset_counter, z_reset_counter, vxy_reset_counter, vz_reset_counter, xy_global, z_global, dist_bottom_valid
vehicle_global_position_0.csv	timestamp, lat, lon, alt, alt_ellipsoid, delta_alt, vel_n, vel_e, vel_d, yaw, eph, epv, terrain_alt, lat_lon_reset_counter, alt_reset_counter, terrain_alt_valid, dead_reckoning
vehicle_gps_position_0.csv	timestamp, time_utc_usec, lat, lon, alt, alt_ellipsoid, s_variance_m_s, c_variance_rad, eph, epv, hdop, vdop, noise_per_ms, jamming_indicator, vel_m_s, vel_n_m_s, vel_e_m_s, vel_d_m_s, cog_rad, timestamp_time_relative, heading, heading_offset, fix_type, vel_ned_valid, satellites_used
vehicle_attitude_0.csv	timestamp, q[0], q[1], q[2], q[3], delta_q_reset[0], delta_q_reset[1], delta_q_reset[2], delta_q_reset[3], quat_reset_counter

their associated features (fields), while Figure 3.7 shows the same for the telemetry system. Describing each feature is out of the scope of this thesis, however, they can be understood by reading their associated *.msg* files on the PX4 Github repository [95].

Often these CSV files will contain all of the data around a specific sensor, but not always. If more than one sensor log is required they are clustered together into one condensed file. This helps to keep all features sorted with the same key. The clustering process also helps to clean the data before training. The timestamp is used as the primary key for sorting features, however, some messages will be polled from the autopilot at different rates. When merging the clusters of related CSVs together,

Table 3.7: Telemetry System CSVs and Features

CSV File	Contained Features
vehicle_status_0.csv	timestamp, onboard_control_sensors_present, onboard_control_sensors_enabled, onboard_control_sensors_health, arspd_check_level, load_factor_ratio, nav_state, arming_state, hil_state, failsafe, system_type, system_id, component_id, vehicle_type, is_vtol, vtol_fw_permanent_stab, in_transition_mode, in_transition_to_fw, rc_signal_lost, rc_input_mode, data_link_lost, data_link_lost_counter, high_latency_data_link_lost, engine_failure, mission_failure, failure_detector_status, aspd_check_failing, aspd_fault_declared, aspd_use_inhibit, aspd_fail_rtl
telemetry_status_0.csv	timestamp, heartbeat_time, data_rate, rate_multiplier, rate_rx, rate_tx, rate_txerr, remote_system_id, remote_component_id, remote_type, remote_system_status, type, mode, flow_control, forwarding, mavlink_v2, ftp, streams

the timestamp key will be used to sort the messages after the merge is complete. Some fields may also contain Not a Number (NaN) values or values that are infinity. Linear extrapolation in both directions is used to attempt to smooth out these values. If this is not possible, the column will be dropped. Algorithm 2 shows the pseudo code for the CSV clustering and merging process. Before any training, the timestamp column is dropped.

One-class classification by design does not require a labelled dataset for training. When testing the classifiers, however, labels are required in order to examine their performance. Rows are labelled "benign" or "malicious" based on the timing of the attack and other known factors. Labels are dropped during training, and only used to calculate the resulting confusion matrices and F1 scores.

Algorithm 2: Feature Extraction and Clustering

```
Result: CSV dataset per cluster
Initialization;
for CSV from flight log do
    if CSV is related to sensor then
        Merge CSV into sensor's dataframe with key=timestamp;
    end
end
Set index to timestamp and sort;
Apply linear interpolation of NaN values;
Drop any remaining columns with infinity or NaN values;
```

Principal Component Analysis

Following the clustering and feature extraction phase, the features are standardized into Z-scores by using the *scikit-learn* StandardScalar function. This will standardize the features with a mean and standard deviation of the training set. Standardization is necessary as the sensor log features can have a broad range of values with different scales, potentially falsely influencing the variance of the feature extraction process. Figure 3.17 shows 3 selected features with highly variable ranges from the GPS Jamming dataset before standardization, with Figure 3.18 showing the same data after standardization. The Z-score calculation is shown below, where x is the particular sample, μ is the mean of the samples, and σ is the standard deviation of the samples.

$$Z = \frac{(x - \mu)}{\sigma}$$

Once features are standardized, there still remains a high number of features. To reduce the dimensionality of the dataset and to improve the performance and results of the machine learning algorithms, Principal Component Analysis (PCA) is applied. Principal component analysis is a technique used for feature extraction. PCA is able

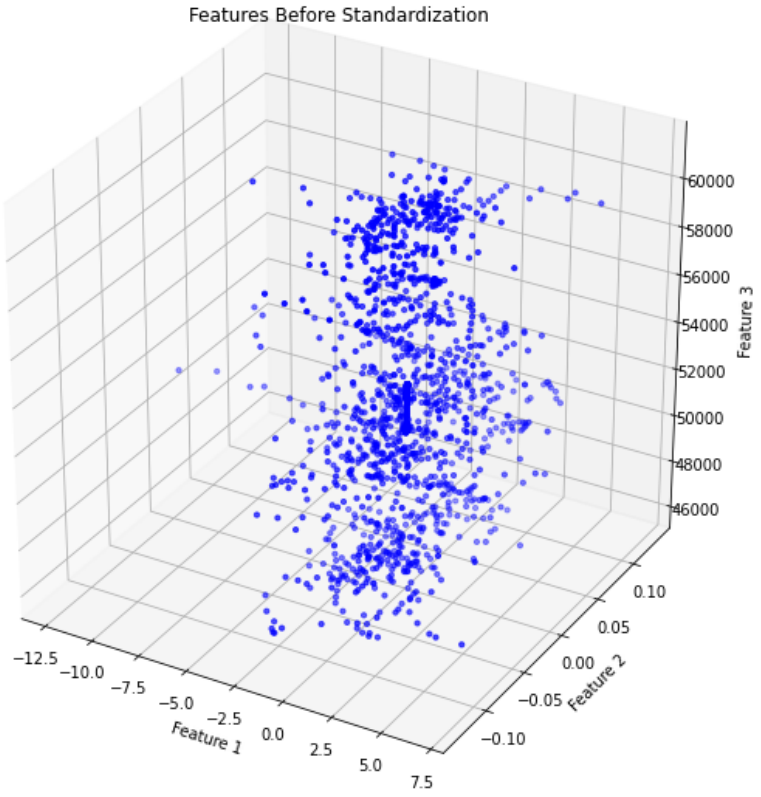

Figure 3.17: 3 Features Before Standardization

to transform a set of features into principal components to better explain the variance in the original set. These principal components are created from linear combinations of the original features in order to capture a certain percentage of variance from the original set while reducing dimensionality. Useful data contains variance, and the more variance the data has, the higher it's importance. As it is able to reduce dimensionality to a few principal components, it can also be used to visualize data. For demonstration purposes, Figure 3.19 shows a 3D scatter plot created by reducing the live GPS spoofing set of 85 features to only 3 principal components, while Figure 3.20 shows 3 principal components of the live GPS jamming attack. The number of

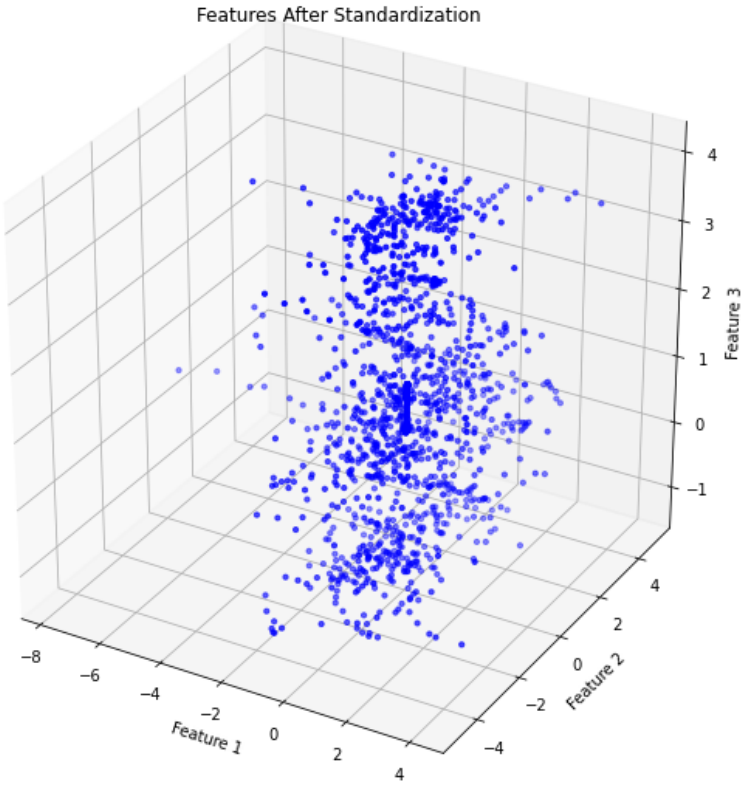

Figure 3.18: 3 Features After Standardization

principal components can be specified by explicitly declaring a number of components, like above, or by specifying how much variance to retain. In practice, the aim is to keep as much variance as possible while minimizing the amount of features. By specifying the target amount of explained variance, we can gain an accurate representation of the original data without needing to use trial and error to find the best number of components. Different percentages of explained variance are specified in which the variance remains high while minimizing the number of features. For the GPS-related attacks, the PCA for SVM and LOF are configured to capture 85% of variance from the dataset, while the autoencoder performed better with 95% variance. These values

were selected as they provided the highest performance while minimizing the number of resulting features. For the Ping DoS attack, the variance is set at 85% for all of the tested classifiers. The training set is used to fit PCA, then it is used to transform both the testing and training set. As an example, PCA applied to the live GPS spoofing data reduced the number of features from 83 to 13.

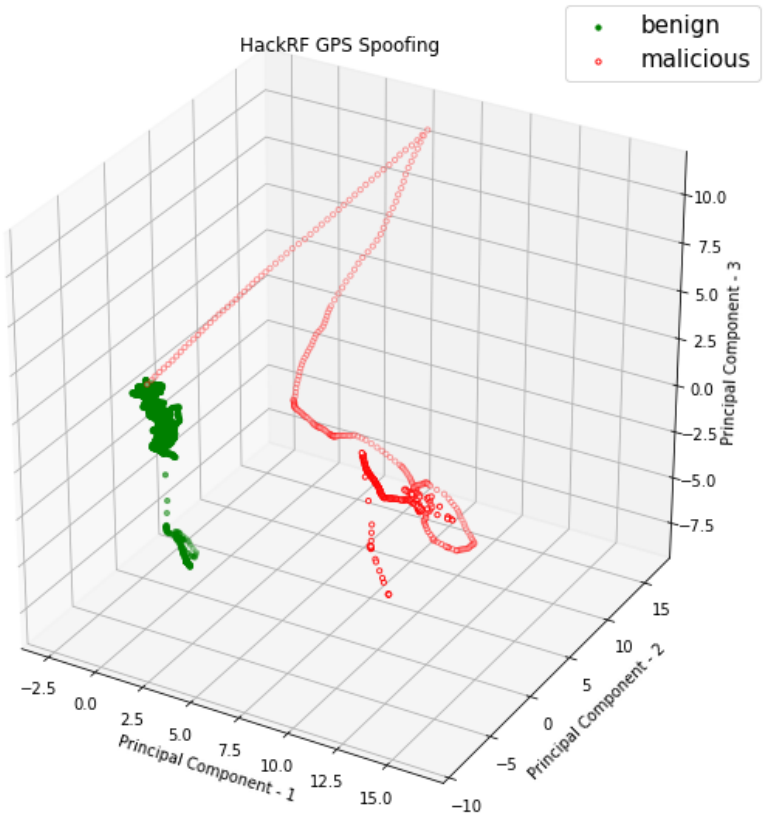

Figure 3.19: 3 PC of Live GPS Spoofing

The result of the pre-processing and feature selection processes is a condensed CSV file per sensor and a reduced set of features with high variance. The advantage of this method of pre-processing is that it is able to take an arbitrary number of sensor-related log files and fields and reduce them to a small set of standardized

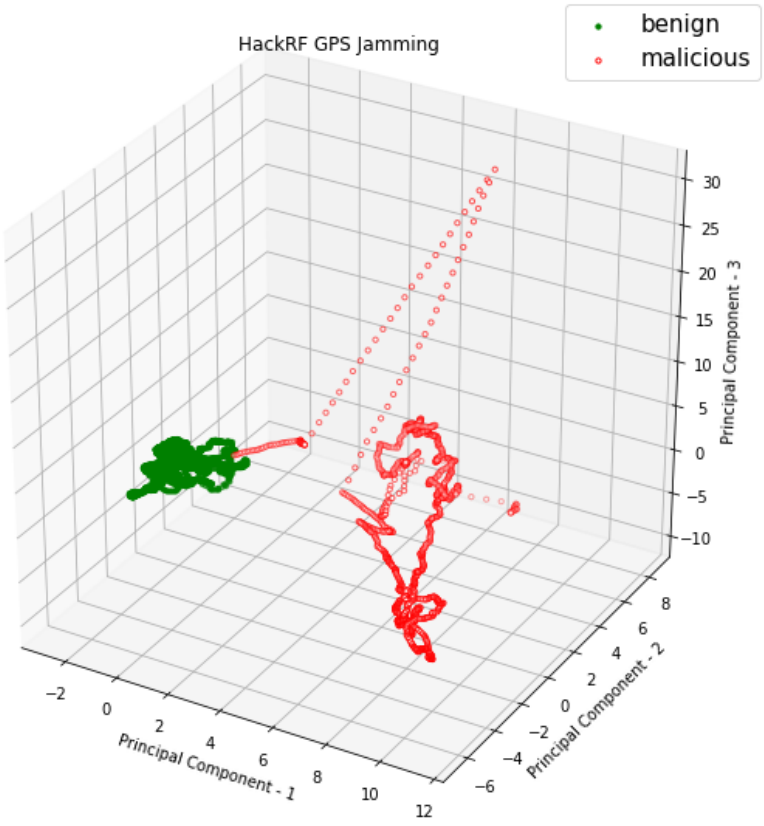

Figure 3.20: 3 PC of Live GPS Jamming

features with high variance. As the goal of the IDS is to operate directly onboard the UAV, the least number of features is desirable to reduce computational cost.

3.3.2 Local Outlier Factor

The *scikit-learn* implementation of the LOF algorithm is used. As discussed in Section 3.1.2, LOF computes the local density of an observation with respect to its neighbours and then classifies it as a novelty if its density is significantly lower. The algorithm is trained on the dataset obtained from a normal flight where no attack occurs and therefore no anomalies are present. Predictions are then made using the

model against the logs obtained from flights where an attacks occurs. Manual hyperparameter tuning is done for this algorithm. In general, the number of neighbours was significantly high in order to obtained decent results. A static contamination parameter is given relative to the number of true novelties within the test set in order to get an understanding of the best possible performance. In addition, the *novelty* parameter is set to *True* which makes the implementation perform in a semi-supervised manor requiring a training set. The *n_jobs* parameter is set to -1 which tells the algorithm to use all available processing power. Full performance of each dataset with the associated hyperparameters are shown in Section 4.1.

3.3.3 One-Class Support Vector Machine

The *scikit-learn* implementation of OC-SVM is used with the Gaussian Radial Basis Function (RBF) kernel. This kernel was chosen due to its performance in one-class classification problems and those where there is no prior knowledge of the data [96]. This is the case here, as the algorithm is trained only on the flight data where no attacks occur. The *gamma* and *nu* hyperparameters are automatically tuned for the best precision and recall by using a grid search fit and score method (GridSearchCV). Occasionally, the resulting hyperparameters can be manually tuned further for optimal performance. After the optimized parameters are known, final training is done on the benign dataset and predictions are done against the dataset where attacks were experienced with labels removed.

3.3.4 Autoencoder Neural Network

The Keras neural network library is used to create the autoencoder [97]. As described in Section 3.1.4, the autoencoder must be created with the same number of input and output nodes and a smaller hidden layer to impose an informational bottleneck. The neural network is created with three layers. The first layer is the input layer which is constructed with twenty-five nodes. The middle layer is constructed with only three nodes as to cause a bottleneck. The output layer has the same number of nodes as the input layer. The rectified linear activation function is used as it can output true zero values. This is important for the autoencoder as it allows a sparsity when learning the compressed representation [98]. The Adaptive Moment Estimation (ADAM) optimizer is used as it is computationally efficient and requires little memory [99]. Mean Squared Error (MSE) is used as the loss function, and is kept for each observation. Training is run for a maximum of 100 epochs with early stopping based on the loss. During training, the MSE should remain low as the autoencoder is able to closely reconstruct the input. During a regular flight, new benign observations will also be reconstructed with low error. When an attack occurs, however, the MSE will be significantly higher than the benign observations. To accurately detect attacks, a threshold parameter can be used to separate benign and malicious observations and to better tune for false positives. Given a percentile threshold, T, a new threshold can be calculated as the quantile with respect to the MSE. This new value is the true threshold used to classify observations as benign or malicious if they are below or above the threshold respectively. Full performance of the autoencoder on each dataset are shown in Section 4.1.

3.4 MAVIDS

Once an effective intrusion detection method has been designed and tested, it can be deployed into a developed intrusion detection system. MAVIDS (Mirco Air Vehicle Intrusion Detection System) is the IDS proposed which includes a GCS client application and an onboard UAV agent. The IDS allows for UAV operators to easily create machine learning models for their specific UAV by simply carrying out a flight in where no attack is expected to occur, and uploading the resulting flight log to MAVIDS for training. This model is then uploaded to the onboard agent for inference during flight, where the agent receives a mirror of the flight data the flight controller is processing. This IDS model for provides a number of benefits:

- Ease of use for UAV operator: no technical knowledge needed to train a model
- Trained models are not generic; they are trained specifically for the UAV they are to operate on, providing increase performance
- Onboard agent approach allows for the detection of attacks even when communication to the GCS is lost (ie. jamming attacks)
- If communication to the GCS is lost, the hook into the flight controller allows the agent to trigger mitigating actions autonomously without the need for operator input

MAVIDS is developed using open source frameworks including Django, Bootstrap, Tensorflow, and Scikit-learn. These frameworks allow for rapid development of the front and back-end components as well as the one-class classifiers.

3.4.1 Architecture

The MAVIDS architecture consists of multiple modules and stages from log extraction to detection of attacks onboard the UAV. Flight logs are required for training, where no attacks or anomalies are experienced. Most UAVs will create flight logs by default for troubleshooting purposes, as is the case with the PX4 flight control firmware. The UAV operator is first required to conduct a flight with their UAV where no attacks are experienced. This can be done on a non-hostile environment such as controlled environment or military base. Once this flight is complete, the logs are downloaded from the UAV and the first stage in the machine learning model development begins. The MAVIDS system uses a GCS web application (portal) for the management of the IDS, which includes interfaces for the changing of settings, training of the classifiers, interaction with the UAV agent, etc. This is the interface the end user will see and interact with, in order to create a layer of abstraction and therefore simplicity to the user. With the non-anomalous flight log downloaded from the UAV, it can then be uploaded to the GCS portal for training. Behind the scenes, the portal will extract the binary log format into CSV files per uORB message, then cluster them into a single log per sensor. This method follows the clustering technique described in Section 3.3.1. From the single sensor-based logs, PCA is applied to reduce dimentionality. The output of this process is the training set used to train the various one-class classifiers. Training and tuning of the algorithms is performed on this set, which then produces a packaged model. This model is uploaded to the UAV agent for onboard inference. During flight, the UAV agent receives a mirrored copy of any uORB messages processed by the flight controller. These messages are inspected by the UAV agent and predicted to be benign or malicious based on the outcome of

the selected classifier algorithm. If an attack is detected, the GCS portal serves as a dashboard for incoming alerts and provides the UAV operator with a number of mitigation options. Some attacks, such as jamming, will cause the communication between the UAV agent and the GCS portal to become unusable. In this scenario, the UAV agent will wait a specified number of seconds until executed a pre-defined "default" mitigation action. This allows for the potential autonomous mitigation of denial of service and jamming attacks in which most other IDS approaches would fail. Figure 3.21 shows a high level overview of the MAVIDS architecture.

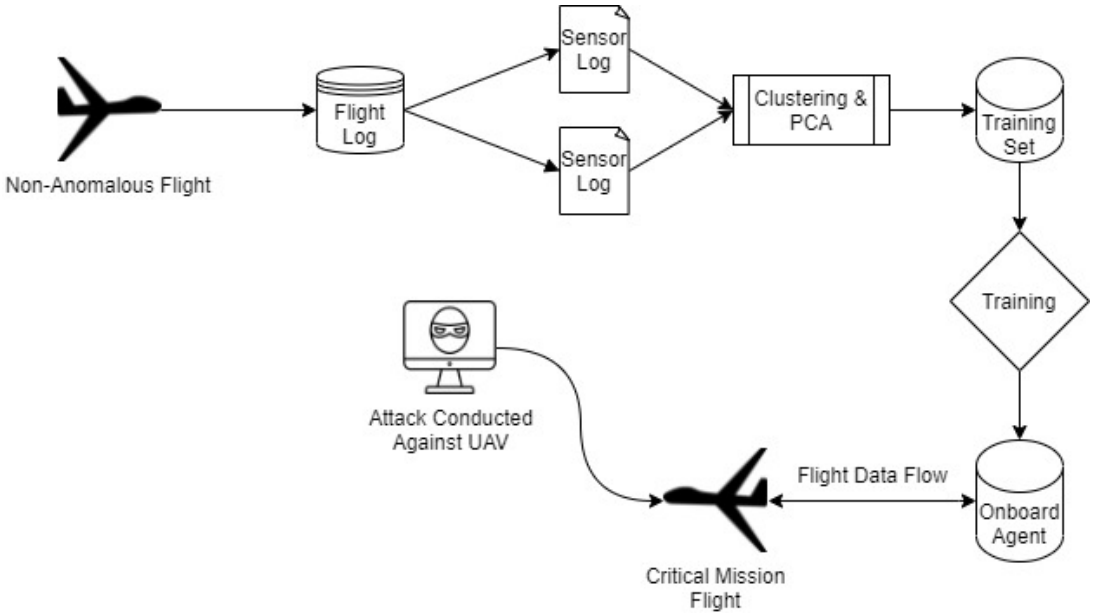

Figure 3.21: MAVIDS High Level Architecture

3.4.2 GCS Portal

The MAVIDS GCS portal is the interface used by the UAV operator to interact with the IDS. The portal is a web application built using Django and Bootstrap. Django is an open source Python framework for the rapid development of web applications

[100]. This framework provides a number of advantages as the back-end of the GCS portal. As Django is based on Python, native machine learning libraries for Python such as Tensorflow and Scikit-learn can be used without needing to call non-native code as they can be imported directly into the portal code directly. This also helps with consistency, as the machine learning code used for testing can also be used on the live system without the need for any modifications which may impact performance. Django allows for rapid development as it is a template-based framework with many of the heavy development tasks built-in. For example, user management and working with databases are built into the framework and can be used quickly without needing to reinvent the wheel. As Django is open source, the security of the built-in functionality has been reviewed and can provide protection against various web-based attacks out of the box, including SQL injection, cross-site request forgery, cross-site scripting, and more. This allows for rapid development without leaving the security of the application behind.

The Django application serves both the front end back-end of the GCS portal. The front-end design elements and layout utilize the popular Bootstrap framework [101]. Bootstrap allows for rapid front-end development that is both responsive and visually appealing. Page templates are created using HTML, CSS and Javascript, for each view. Django template language (DTL) code is then inserted to create dynamic content.

There are many ways for a GCS to communicate with the UAV including dedicated communication links. Although this may increase bandwidth, it introduces a new attack surface and requires additional hardware. To mitigate this issue, the communication between the GCS portal application and the UAV agent utilize the

Table 3.8: MAVIDS MAVLink Message Format

Message Field	Description
time_usec	Timestamp from system boot or epoch time
target_system	Target system: If set to 0, the message was sent by the GCS portal. If 1, it was sent by the UAV agent
message_mode	Bitmap of the information contained within the message. Settings, attack, responses, etc
ignore_alert	Flag that indicates whether to ignore the alert or not, message_mode[2] must be set to 1
alert_id	Alert ID number, used as the primary key
attack_name	Name of the attack, represented by 3 characters (GPS, DOS, etc)
attack_score	Float representing the machine learning prediction score
default_action	Default mitigation action to carry out when an attack is detected
default_initiate_time	Time until the default mitigation action is initiated
default_return_time	Time until the UAV will return to the previous flight mode, ending any mitigation action
modules_enabled	Bitmap of the enabled machine learning models (GPS spoofing, DoS, etc)

existing telemetry link. This is done by defining a custom MAVLink message called *MAVIDS*. This message includes all of the information needed by the GCS portal and the UAV agent as shown in Table 3.8.

The *MAVIDS* message is defined within a custom MAVLink dialect based on the PX4 "common" definitions. When both the GCS portal application and the UAV agent use this dialect, they will be able to parse the messages content to send and receive alert details, various settings, and mitigation actions. As PX4 automatically forwards MAVLink messages from the onboard computer to other components, inter-system communication can be achieved without the need for additional hardware or

dedicated communication links.

The GCS Django application manages this communication as well as other tasks such as training through background sub-processes. The primary sub-process manages the connection with the onboard companion computer which runs the IDS agent. An initial connection is established using the user-defined settings (or the default), including the type of connection, port, and IP address. Typically this is the local GCS connection, such as QGroundControl. Using this connection, the MAVLink commands are broadcast throughout to other systems and components including the flight controller and companion computer. Once the connection is established, the primary sub-process will check for *HEARTBEAT* MAVLink messages from the system to ensure the connection is still active. This sub-process is used to communicate the *MAVIDS* message to the IDS agent. Another sub-process is initalized for machine learning training purposes. After a ULog file is uploaded, this sub-process will extract the log and commence pre-processing and training for the selected algorithms.

Upon logging into the MAVIDS GCS portal, the user is prompted with the dashboard. This is the main view used during the flight of the UAV, showing any activate alerts and prompting the operator for mitigation actions. The left pain shows any active alerts as they are detected by the UAV agent. By clicking on one of these alerts, the operator will see the time remaining until the default mitigation action occurs, and is provided with the option to change this action or abort and suppress the alert as a false positive. The MAVIDS GCS dashboard is shown Figure 3.22. Future work could utilize the abort and suppress feature to further train the machine learning algorithms.

The settings view of the portal allows the user to change settings related to the

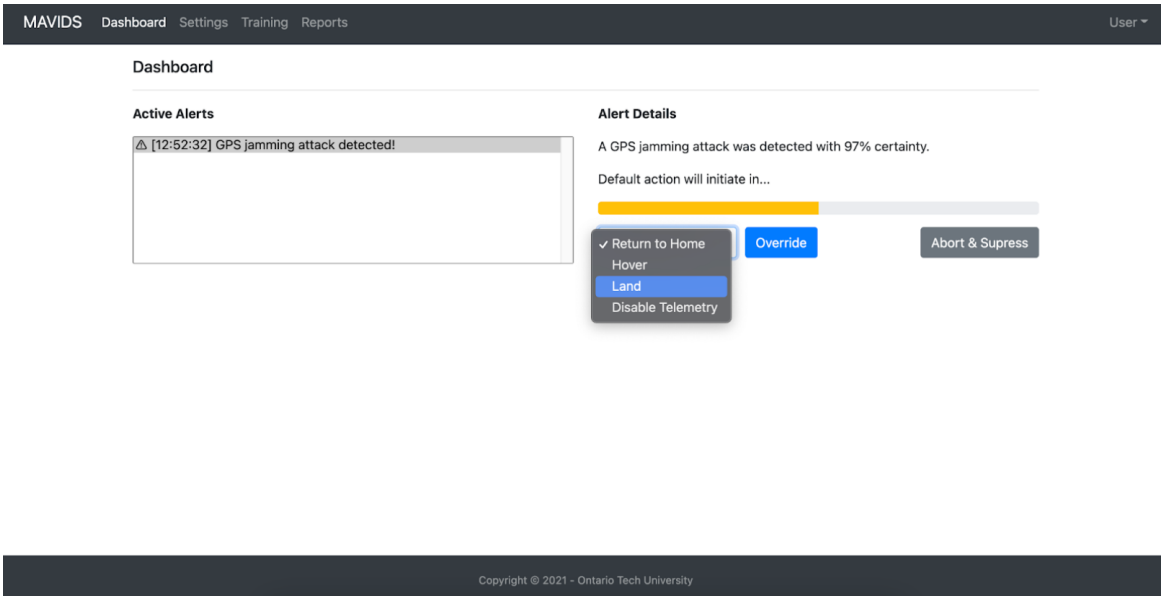

Figure 3.22: MAVIDS GCS Dashboard

connection, detection, and alerting. The Django database integration is used to store the persistent settings into an SQLite database. Settings that can be configured by the user include:

- Enabled detection modules: specifies which attacks to detect and train for
- Default mitigation action: specifies the default mitigation action that the agent should initiate when an attack is detected, such as hovering, landing, disabling GPS, etc.
- Time until default mitigation action is initiated: time delay until the default action is initiated, used to allow the operator to intervene and override if needed
- Time until returning to previous flight mode: depending on the mitigation action, the UAV may change flight mode or disable functionality. This timer

defines whether the UAV should return to its previous state and if so, how long to wait until doing so

- UAV connection method: allows the user to specify how to connect to the UAV and the associated connection settings

The settings view is shown in Figure 3.23.

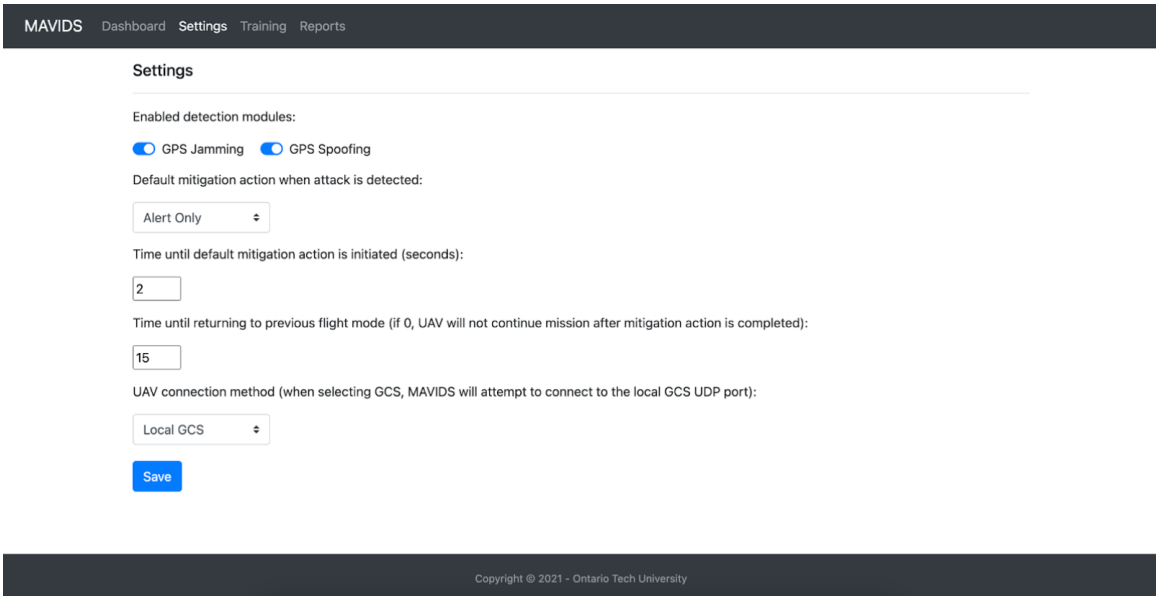

Figure 3.23: MAVIDS GCS Settings Tab

The training view allows the operator to select the machine learning algorithms they would like to train and specify any manual hyperparameters. The operator can then upload the ULog flight log from their training flight and begin the training process. An output window shows training process from the underlying Python scripts. The training view is shown in Figure 3.24.

Reporting is a way of logging any detected attacks during flight, as well as the actions the operator took. This view shows any available reporting logs and allows

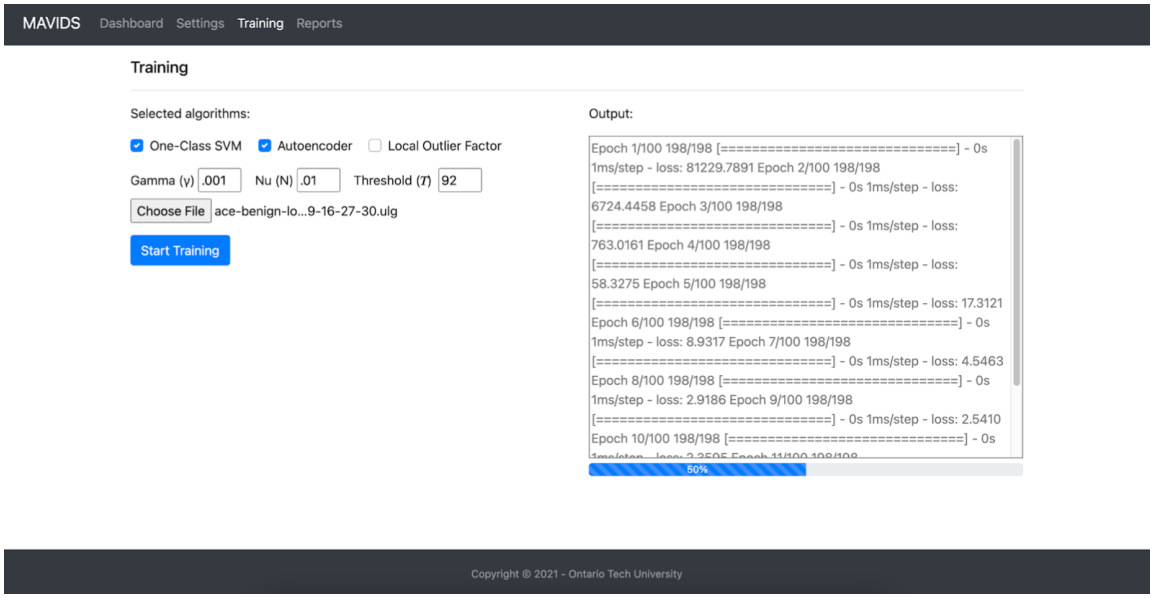

Figure 3.24: MAVIDS GCS Training Tab

the operator to download them for review. The reporting view is shown in Figure 3.25.

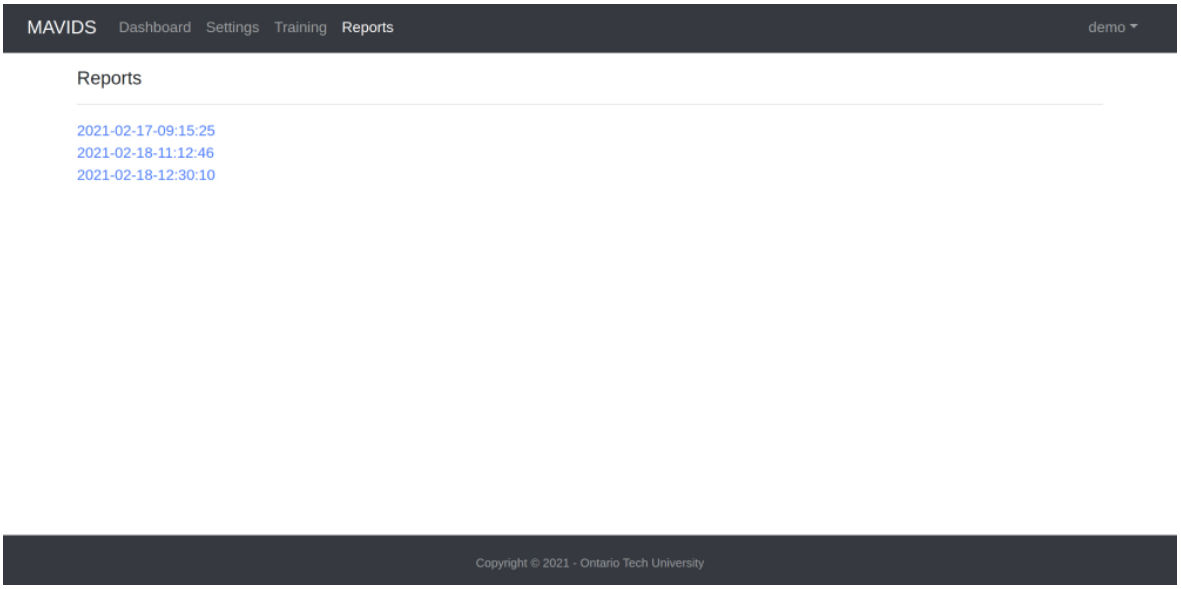

Figure 3.25: MAVIDS GCS Reports Tab

3.4.3 UAV Agent

The MAVIDS UAV Agent runs on a companion computer onboard the UAV to detect and attempt to mitigate attacks. Examples of an onboard computer include the Raspberry Pi Zero, Raspberry Pi 4, and NVIDIA Jetson Nano. The companion computer is connected to the spare TELEM port on the flight controller in order to communicate with the flight controller and GCS via *pymavlink*. Once training of the specified detection modules is complete, the trained models are uploaded to the UAV from the GCS portal. During flight, the UAV Agent will pull the data required for inference from the flight controller using one of several available methods:

- Requesting the specific MAVLink messages: Limited
- FastRTPS (Fast Real Time Publish Subscribe): Allows for more data to be pulled per request
- MAVLink log streaming: Allows for the companion computer to receive a stream of ULog data, which is then stored in a time series database such as Influx DB

Once the live data is received and clustered, PCA is applied to keep the incoming data consistent with training. During training, however, a specified percentage of variance given. When conducting inference, the number of principal components is given which matches those generated during training. This keeps the number of features consistent across training and inference, as with minimal incoming data the variance percentage may produce a variable number of features. The processed incoming stream of data from the flight controller is then fed into the machine learning models for inference. If any data is classified as malicious, the MAVIDS message is sent to the GCS to notify the operator. If default mitigation actions are specified

within the MAVIDS settings, those actions will commence upon detection unless the operator overrides them.

During flight, the onboard computer maintains a heartbeat with the flight controller and GCS portal. The GCS portal will show a message if the heartbeats are not received and the connection drops. MAVIDS messages when sent from the GCS portal will first be received by the flight controller, and will then be forwarded out of the spare TELEM port to the UAV Agent. These messages can include settings changes, alerts, and alert responses. Figure 3.26 shows a Raspberry Pi Zero running the UAV Agent onboard the S500 UAV.

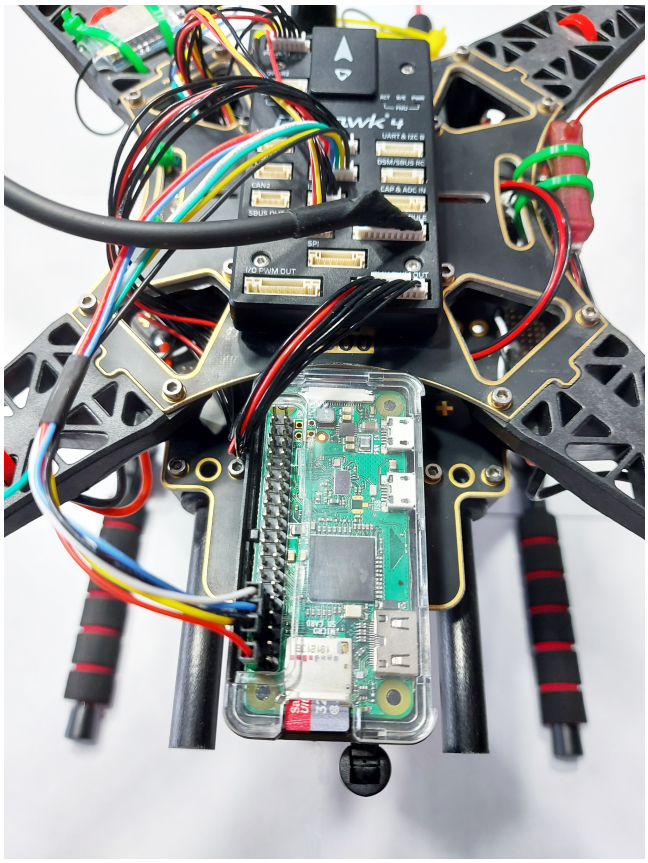

Figure 3.26: MAVIDS Agent Running Onboard UAV

Chapter 4

Performance Evaluation

The effectiveness of an IDS depends on its detection performance. By using standard metrics, the detection performance can be analyzed and compared with other solutions. Common metrics include accuracy, precision, recall, and F1 scores. Although the detection method used by MAVIDS does not require a labelled dataset, the data is labelled solely for the purpose of measuring detection performance. These labels are not used during the training process, as described in Section 3.3.

Intrusion detection within the UAV domain offers a number of unique challenges. Not only is the detection performance important, but the onboard device has a number of constraints. First is the prediction latency. As the UAV processes flight commands, the IDS must be able to keep up as to not cause a bottleneck when conducting inference on those commands. Additionally, the onboard companion computer must be light enough and have minimal power draw to keep the impact on flight performance low.

4.1 Detection Performance

Detection performance measures the success of the IDS in detecting various attacks. Accuracy is a common metric used to evaluate an IDS, however, as this is an imbalanced classification problem it can be a deceiving metric. With the low ratio of malicious to benign observations, a high accuracy percentage can be achieved with poor true performance. For example, using the S500 Ping DoS dataset with 19 malicious observations and 4444 benign observations, a high accuracy of 99.57% is achieved even when the IDS fails to identify any malicious samples. For this reason, the evaluation of class imbalanced machine learning classification problems are often based on precision, recall, and F1 score [102].

Classification problems in intrusion detection are made up of positive and negative classes for malicious and benign classifications respectively. Each of these calculations are made up of the following metrics, which are the possible prediction outcomes:

- True positive (TP): Malicious prediction of truly malicious observation
- False positive (FP): Malicious prediction of truly benign observation
- True negative (TN): Benign prediction of truly benign observation
- False negative (FN): Benign prediction of truly malicious observation

Precision measures the proportion of correctly classified observations. For the intrusion detection problem, precision represents the measure of observations that are correctly identified as malicious out of all truly malicious observations. It is calculated as:

$$Precision = \frac{TP}{TP + FP}$$

Recall measures the proportion of truly malicious observations. For all observations, recall will measure how many malicious observations were correctly detected. Recall is calculated as:

$$Recall = \frac{TP}{TP + FN}$$

Both precision and recall are important metrics for intrusion detection. In practical terms, precision helps to determine how well the IDS was able to detect the malicious data observations whereas recall determines how well the IDS was able to identify the attack from all incoming data. An F1 score is a scaled value from 0 to 1 which considers both precision and recall. This helps to score the overall performance of the classifier with 0 being the lowest and 1 being the highest. The F1 score is calculated by finding the harmonic mean of the precision and recall:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The hyperparameters of each classifier are tuned for optimal performance as discussed in Section 3.3. The resulting precision, recall, and F1 scores as well as the hyperparameters used to gain these scores are shown below for each attack. Table 4.3 shows the results for simulated GPS spoofing whereas Table 4.2 shows the results from live GPS spoofing. Table 4.1 shows the performance of the classifiers against a GPS jamming attack. Finally, Table 4.4 shows the results from the Ping DoS attack.

Table 4.1: Detection Performance - Live GPS Jamming

Model	Classifier	Label	Precision	Recall	F1 Score	Hyperparameters
Holybro S500	OC-SVM	Malicious	0.98182	0.07397	0.13758	$\nu = 0.0005357$
		Benign	0.99927	0.99138	0.99531	$\gamma = 0.000106$
	LOF	Malicious	0.98559	0.46849	0.63510	$k\_neighbors = 2910$
		Benign	0.86507	0.99799	0.92679	
	Autoencoder	Malicious	0.84606	0.99384	0.91402	$T = 73.4\%$
		Benign	0.99810	0.94704	0.97190	

Table 4.2: Detection Performance - Live GPS Spoofing

Model	Classifier	Label	Precision	Recall	F1 Score	Hyperparameters
Holybro S500	OC-SVM	Malicious	1.00000	0.66867	0.80144	$\nu = 0.01$
		Benign	0.94983	1.00000	0.97427	$\gamma = 0.001$
	LOF	Malicious	0.92067	0.76908	0.83807	$k\_neighbors = 2910$
		Benign	0.96413	0.98944	0.97662	
	Autoencoder	Malicious	0.74727	0.96185	0.84109	$T = 82.3\%$
		Benign	0.99363	0.94814	0.97035	

4.2 Onboard Computational Performance

The MAVIDS agent runs onboard the UAV to allow for the detection and potentially the mitigation of attacks even under duress from jamming or denial of service attacks. The use of an onboard companion computer introduces a number of constrains, such as the device size, weight, power draw, and computational power limitations. As the companion computer is mounted onboard the UAV, the size and weight as it may be too large or heavy for the UAV. In addition, added weight and power draw can both lead to reduced flight times. Size, weight, and power draw are obvious attributes of a companion computer that are available as part of the specifications. Computational performance, however, is largely subjective depending on the use case. In the case of the IDS agent, it is running on the companion computer to process the internal messages from the flight controller. This means it must be able to keep up with the rate at which these messages are transmitted. Failure to do so would lead to a

Table 4.3: Detection Performance - Simulated GPS Spoofing

Model	Classifier	Label	Precision	Recall	F1 Score	Hyperparameters
3DR IRIS+	OC-SVM	Malicious	0.62074	1.00000	0.76600	$\nu = 0.011$
		Benign	1.00000	0.98707	0.99335	$\gamma = 0.000211$
	LOF	Malicious	0.04801	1.00000	0.09162	$k\_neighbors = 1000$
		Benign	1.00000	0.57136	0.72722	
	Autoencoder	Malicious	0.75495	0.99909	0.86003	$T = 97.2\%$
		Benign	0.99998	0.99299	0.99647	
Holybro S500	OC-SVM	Malicious	0.69628	0.96482	0.80885	$\nu = 0.011$
		Benign	0.99927	0.99138	0.99531	$\gamma = 0.000211$
	LOF	Malicious	0.04571	1.00000	0.08742	$k\_neighbors = 3500$
		Benign	1.00000	0.57234	0.72801	
	Autoencoder	Malicious	0.64561	0.99707	0.78374	$T = 96.89\%$
		Benign	0.99994	0.98879	0.99433	
Yuneec H480	OC-SVM	Malicious	0.51196	0.99110	0.67516	$\nu = 0.0211$
		Benign	0.99981	0.98049	0.99006	$\gamma = 0.0003$
	LOF	Malicious	0.09658	1.00000	0.17614	$k\_neighbors = 3100$
		Benign	1.00000	0.80681	0.89308	
	Autoencoder	Malicious	0.83483	0.99020	0.90591	$T = 97.6\%$
		Benign	0.99980	0.99595	0.99787	
DeltaQuad VTOL	OC-SVM	Malicious	0.41419	0.99280	0.58453	$\nu = 0.1$
		Benign	0.99952	0.91479	0.95528	$\gamma = 0.000211$
	LOF	Malicious	0.38128	0.99730	0.55166	$k\_neighbors = 3100$
		Benign	0.99982	0.90179	0.94280	
	Autoencoder	Malicious	0.87728	0.99730	0.99730	$T = 93.5\%$
		Benign	0.99983	0.99153	0.99567	
Standard Tailsitter	OC-SVM	Malicious	0.56739	0.99102	0.72162	$\nu = 0.19$
		Benign	0.99941	0.95307	0.97569	$\gamma = 0.000211$
	LOF	Malicious	0.48570	0.99191	0.65210	$k\_neighbors = 3100$
		Benign	0.99946	0.93477	0.96603	
	Autoencoder	Malicious	0.87728	0.99730	0.93345	$T = 93.5\%$
		Benign	0.99983	0.99153	0.99567	
Standard Plane	OC-SVM	Malicious	0.25993	0.99242	0.41196	$\nu = 0.1811$
		Benign	0.99960	0.87150	0.93117	$\gamma = 0.000254$
	LOF	Malicious	0.21127	0.99147	0.34832	$k\_neighbors = 3500$
		Benign	0.99953	0.83167	0.90791	
	Autoencoder	Malicious	0.86068	0.98957	0.92063	$T = 95\%$
		Benign	0.99952	0.99271	0.99611	

processing bottleneck.

In order to benchmark computational performance, common metrics are needed which demonstrate the companion computers ability to make single predictions as well as the prediction throughput. Prediction latency represents the time it takes the

Table 4.4: Detection Performance - Ping Telemetry Denial of Service

Model	Classifier	Label	Precision	Recall	F1 Score	Hyperparameters
3DR IRIS+	OC-SVM	Malicious	0.00000	0.00000	0.00000	$\nu = 0.0211$
		Benign	0.99598	0.99955	0.99776	$\gamma = 0.01$
	LOF	Malicious	0.00278	0.38889	0.00552	$k\_neighbors = 1000$
		Benign	0.99440	0.43775	0.60790	
	Autoencoder	Malicious	0.01336	0.33333	0.02570	$T = 90\%$
		Benign	0.99703	0.90081	0.94648	
Holybro S500	OC-SVM	Malicious	1.00000	0.78947	0.88235	$\nu = 0.0211$
		Benign	0.99910	1.00000	0.99955	$\gamma = 0.01$
	LOF	Malicious	1.00000	0.78947	0.88235	$k\_neighbors = 2500$
		Benign	0.99910	1.00000	0.999955	
	Autoencoder	Malicious	0.02013	0.47368	0.03863	$T = 90\%$
		Benign	0.99751	0.90144	0.94704	
Yuneec H480	OC-SVM	Malicious	0.02083	0.66667	0.04040	$\nu = 0.0211$
		Benign	0.99880	0.89858	0.94604	$\gamma = 0.01$
	LOF	Malicious	0.02764	0.94444	0.053371	$k\_neighbors = 5000$
		Benign	0.99980	0.89247	0.94309	
	Autoencoder	Malicious	0.52941	1.00000	0.69231	$T = 99.4\%$
		Benign	1.00000	0.99712	0.99856	
DeltaQuad VTOL	OC-SVM	Malicious	0.06260	0.77551	0.11585	$\nu = 0.15$
		Benign	0.99560	0.81405	0.89572	$\gamma = 0.000106$
	LOF	Malicious	0.38824	0.67347	0.49254	$k\_neighbors = 2500$
		Benign	0.99471	0.98301	0.98882	
	Autoencoder	Malicious	0.69841	0.89796	0.78571	$T = 98\%$
		Benign	0.99836	0.99379	0.99607	
Standard Tailsitter	OC-SVM	Malicious	0.09651	0.90385	0.17440	$\nu = 0.0211$
		Benign	0.99803	0.85190	0.91919	$\gamma = 0.08$
	LOF	Malicious	0.21053	0.69231	0.32287	$k\_neighbors = 2500$
		Benign	0.99439	0.95456	0.97407	
	Autoencoder	Malicious	0.21698	0.88462	0.34848	$T = 93\%$
		Benign	0.99787	0.94413	0.97025	
Standard Plane	OC-SVM	Malicious	0.80952	1.00000	0.89474	$\nu = 0.0211$
		Benign	1.00000	0.99792	0.99896	$\gamma = 0.01$
	LOF	Malicious	0.80952	1.00000	0.89474	$k\_neighbors = 2000$
		Benign	1.00000	0.99792	0.99896	
	Autoencoder	Malicious	0.77273	1.00000	0.87179	$T = 98.9\%$
		Benign	1.00000	0.99740	0.99870	

classifier to make a classification prediction. In this case, atomic prediction latency is measured in which the classifier makes predictions on single observations at a time, as this is how the data is streamed to the UAV agent. Prediction latency can be affected by the number of features, the representation of the input data, and the complexity

of the model.

Prediction throughput determines how many predictions can be made during a specific time period. This is an important metric to know for the UAV agent as a low prediction throughput can cause a bottleneck. It was determined through sampling that the average number of uORB messages transmitted through the experimental flights was on average 350 messages per second. This means that any of the classifiers must have a prediction throughput of 350 or higher to avoid causing a bottleneck on normal operations.

Three companion computers were chosen to demonstrate a variation in size, weight, power draw, and computational performance. All three are popular hardware devices for companion computers onboard smaller UAVs. The three computers are the Raspberry Pi Zero, Raspberry Pi 4 Model B, and the NVIDIA Jetson Nano. The companion computer specifications are shown in Table 4.5.

Table 4.5: Companion Computer Specifications

Companion	**CPU**	**GPU**	**RAM**	**Weight**	**Dimensions**	**Power Draw**
Raspberry Pi Zero	1GHz ARM1176JZF-S	250MHz Video-Core 4	512MB	11.5g	65mm x 30mm	0.4W to 0.7W
Raspberry Pi 4 Model B	1.5GHz Cortex-A72	500MHz Video-Core 4	4GB	46g	85.60mm x 56.5mm	3W to 6.5W
NVIDIA Jetson Nano	1.42GHz Cortex-A57	128-core NVIDIA Tegra X1	4GB	250g	100mm x 80mm	5W to 10W

The Raspberry Pi Zero is an ARM-based computer equipped with 512MB of RAM and a 1GHz CPU. It is extremely small and light with the smallest power

draw, however, suffers from the worst computational performance. This companion computer was chosen to test the absolute minimum in requirements for the MAVIDS UAV agent. The Pi Zero connected via headers to the Pixhawk 4 flight controller is shown in Figure 4.1. Further optimization of the Pi Zero could be done by using AI hats such as XaLogic XAPIZ3500.

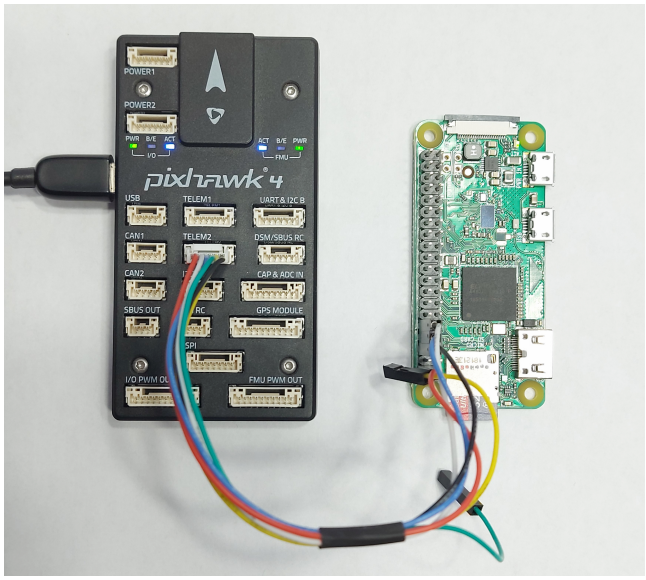

Figure 4.1: Raspberry Pi Zero Companion Computer

The Raspberry Pi 4 Model B is the standard Raspberry Pi with more memory and GPU power than the Zero, albeit with a larger size, weight, and power draw. The model used is the 4GB RAM version. Both Raspberry Pi computer are running Raspbian Buster Lite with no desktop environment to maximize performance. The Pi 4 Model B is shown in Figure 4.2.

The NVIDIA Jetson Nano is a small AI-accelerated computer designed for embedded and Internet of Things (IoT) use cases. The Jetson line of products are common companion computers for UAVs for various other purposes such as AI-based image

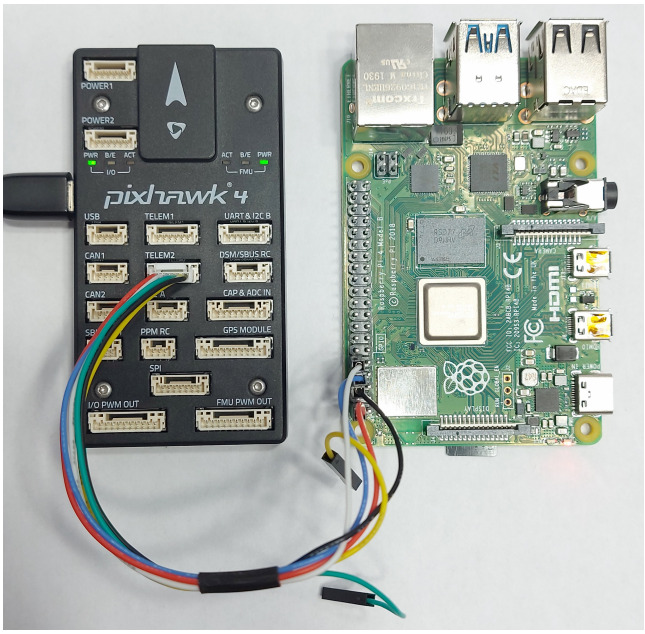

Figure 4.2: Raspberry Pi 4 Model B Companion Computer

classification. The Nano is the largest and heaviest of the three companion computers tested, although it utilizes NVIDIA CUDA cores for accelerated inference. This advantage, in theory, will allow for better performance where the CUDA cores an be utilized. Although it has a GPU advantage, the CPU on the Jetson Nano is not as powerful as the one in the Pi 4 Model B.

Size constraints will be introduced depending on the UAV frame used. Weight and power draw, however, are more difficult to address. Both of these attributes can affect flight endurance, so they are important factors to consider when choosing a companion computer. Generally speaking, these are common companion computers that are used throughout the UAV industry for other onboard processing purposes, and should not significantly impact the flight endurance. This impact could be calculated similarly to any other component using the weight and power draw listed in Figure 4.5 and other values specific to the vehicle such as the battery in use and the power draw of

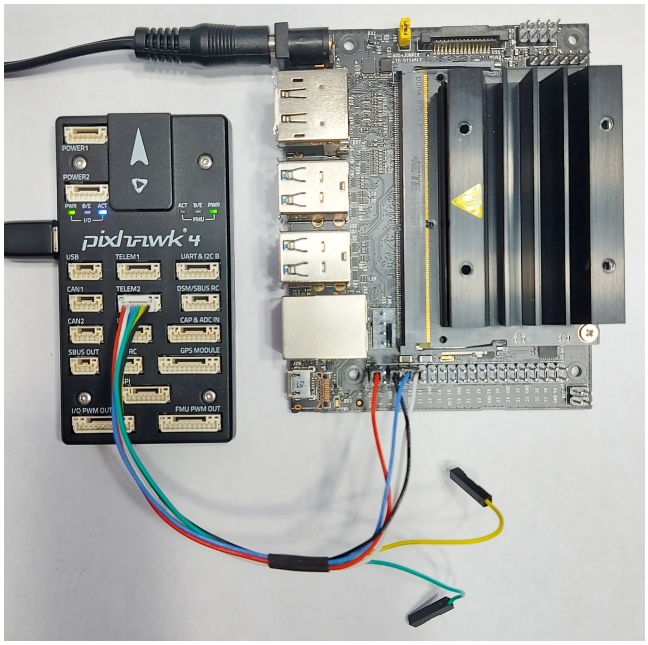

Figure 4.3: NVIDIA Jetson Nano Companion Computer

other components.

To keep the results consistent, the Holybro S500 HITL simulated GPS spoofing models are used for benchmarking across each companion computer. The same input is used across each computer as well for consistency and all code is run with the same versions of libraries within a Jupyter notebook. The standard scikit-learn models are used for the LOF and OC-SVM classifiers, however, the autoencoder models are converted to other formats depending on the companion computer they will run on.

For the autoencoder, the exported Keras models are converted to Tensorflow Lite (.tflite) format and inference is done using Tensorflow Lite on both Raspberry Pi computers. Tensorflow Lite is designed for inference on systems with computational constraints and therefore produces higher performance on the selected companion computers. As the Jetson Nano has CUDA cores, its performance is better realized when using TensorRT [103]. In order to utilize TensorRT, the saved Keras model

must be converted to Onnx format. The below code is an example of benchmarking prediction latency:

Scikit-learn OC-SVM Prediction Latency on Pi Zero

```
converter = tf.lite.TFLiteConverter.from_saved_model('gps-svm')
tflite_model = converter.convert()

times = []
for x in range (0,50):
  start = time.perf_counter()
  predictions = model.predict(df_malicious_flight_pred[:1])
  end = time.perf_counter()
  elapsed = end - start
  times.append(elapsed)

print ("Quickest Time: ", min(times))
print ("Slowest Time: ", max(times))
print ("Average Time: ", sum(times) / 50)
```

An example of the throughput benchmarking code for the autoencoder running on the Pi Zero is shown below:

Autoencoder Throughput Using Tensorflow Lite on Pi Zero

```
interpreter = Interpreter(model_path='gps-ae.tflite')
interpreter.resize_tensor_input(input_details[0]['index'], (1,9))
interpreter.resize_tensor_input(output_details[0]['index'], (1,9))
interpreter.allocate_tensors()

duration_secs = 0.1
start_time = time.time()
n_predictions = 0

while(time.time() - start_time) < duration_secs:
    pred_numpy = np.array(df_malicious_flight_pred[:1], dtype=np.float32)
    interpreter.set_tensor(input_details[0]['index'], pred_numpy)
    interpreter.invoke()
    n_predictions += 1
```

```
15
16 throughput = n_predictions / duration_secs
17 print(throughput)
```

The results from the performance benchmarking are shown in Table 4.6.

Table 4.6: Onboard Agent Performance

Companion	**Method**	**Atomic Prediction Latency (ms)**	**Prediction Throughput (p/s)**
Raspberry Pi Zero	OC-SVM	19.8882	40
	LOF	37.1986	20
	Autoencoder	2.3740	410
Raspberry Pi 4 Model B	OC-SVM	2.8677	270
	LOF	109.0102	10
	Autoencoder	0.4174	2340
NVIDIA Jetson Nano	OC-SVM	2.4258	400
	LOF	123.5816	10
	Autoencoder	0.4520	2260

4.3 Performance Conclusion

The average F1 score of all benign and malicious classifications were computed to determine the overall highest performer. In terms of detection performance, the reconstruction-based classifier, an autoencoder neural network, performed best in for most UAV models and attacks. The autoencoder also had the lowest prediction latency and highest prediction throughput on each companion computer.

One-class support vector machine, a boundary-based classifier, offered mediocre performance both in terms of detection performance as well as onboard computational

performance.

Local outlier factor is not known to perform well on highly dimensional data and required a high number of neighbours to achieve decent detection results. This classifier also created large model exports which need to be loaded into memory for inference. On low resource devices such as the Pi Zero, this requirement significantly hampered the amount of available RAM. As longer training flights would inevitably create larger models, this could lead to the removal of this classifier as an option.

Overall, the classifiers' detection performance had a correlation to its computational performance for each respective companion computer. The autoencoder had the highest F1 scores on average and also boasted the best prediction latency and throughput. Similarly, the performance of the OC-SVM landed in the middle for both detection and computational performance with LOF falling behind. The autoencoder proved to be the most promising for the UAV intrusion detection problem, given both its computational and detection performance.

Chapter 5

Summary, Conclusions & Future Work

5.1 Summary

Unmanned aerial vehicles (UAVs) are aircraft without a pilot on-board that are controlled remotely or autonomously. UAVs are used across industries for tasks that are too costly, dangerous, or difficult for human operators. Examples of UAV usage include industrial control surveillance, military and law enforcement operations, aerial inspections, and medical response. Depending on the intended mission of the UAV, different frame types will be used such as fixed wing or multi-copters. In addition, various unmanned aerial system (UAS) system architectures will be deployed which utilize communication protocols that best fit the task and environment at hand.

Modern UAVs face a large threat landscape due to the tasks they face and their hostile operating environments. Common attacks against UAVs include command injection, denial of service, spoofing, and jamming. Many of these attacks are difficult to mitigate as many of the vulnerabilities come from underlying technologies such as GPS. As the realization of these attacks increase, an intelligent intrusion detection system is needed to help identify and potentially mitigate them. Traditional manned

vehicles such as cars have static features including a single motor, four wheels, and a common communication protocol such as the CAN bus. In the UAV domain, all of these features become variables, making intrusion detection a difficult task.

The MAVIDS solution proposed in this thesis provides an off-the-shelf UAV intrusion detection system that is both simple to initiate and implement. Existing approaches to UAV IDS operate only within specific use cases such as MANETS, have high maintenance requirements, or are effective against a small subset of attacks. With the large number of attacks UAVs face as well as the introduction of future novel attacks, a machine learning approach is the most promising solution to solve these challenges. The major hurdle in machine learning-based approaches to UAV intrusion detection is the lack of available, consistent and labelled datasets for training. By using a one-class classification approach to intrusion detection, the IDS can be trained to be effective on the specific UAV it will be operating on while exploiting the use of existing flight logs.

In the case of autonomous UAVs, MAVIDS can utilize its integration with the flight controller to deploy mitigation techniques such as temporarily disabling sensors. Using existing communication infrastructure, MAVIDS can communicate alert notification as well as various settings between the UAV agent and the GCS portal. This allows for the UAV operator to define default attack mitigation actions and override those actions if necessary.

In summary, this thesis has made the following contributions to the UAV IDS problem:

- Demonstrates a technique using principal component analysis and one-class classifiers for intrusion detection while exploiting the use of pre-existing flight

logs for training data, which performed with macro averaged F1 scores of up to 90.57% and 94.3% for live GPS spoofing and jamming respectively

- Presents an operational intrusion detection system with a modular design which can be applied to a vast number of UAV systems
- Proposes the use of a lightweight IDS agent on-board the UAV, mitigating the problem of communication loss during common denial of service and jamming attacks.
- Provides the research community with a UAV attack dataset to allow the community to compare performances of various developed IDS.

5.2 Conclusion

UAVs perform critical tasks in high risk environments. Given the large threat landscape they face, an intelligent IDS is necessary to help detect cyber attacks. Unfortunately, due to the wide range of components and technologies used in UAV deployments, it becomes very difficult to create an accurate IDS that can be used with many different implementations. As UAVs often operate in hostile environments far away from their operator, the highest threats are those facing the UAV itself through its various onboard sensors. This thesis has proposed the use of one-class classifiers to train a novelty-based IDS, which learns normal sensor values from previous flight logs. The experiments conducted show promising results, particularly with an autoencoder neural network, to detect sensor-based attacks against UAVs. The autoencoder performed with macro averaged F1 scores of up to 90.57% and 94.3% for live GPS spoofing and jamming respectively. The proposed approach shows to be effective across a variety of UAV platforms and control configurations.

Placing IDS agent directly on-board the UAV has the potential to mitigate denial of service and jamming attacks even when the connection to the GCS is lost. This opens the door to not only the detection of attacks under duress, but also the potential for autonomous mitigation.

As a result of this work, a dataset of both benign and malicious flight logs from various UAVs undergoing DoS and GPS spoofing attacks was created and published on IEEE DataPort [81]. This dataset allows other researchers to begin working on UAV IDS solutions without needing to start from scratch in developing a dataset. The dataset has already been used as a base for peer reviewed work by researchers at the School of Cybersecurity at Korea University [80].

5.3 Future Work

MAVIDS provides a simple yet effective solution to UAV intrusion detection. By utilizing this solution, UAV manufacturers and operators can protect their assets out-of-the-box or improve upon it to suit their needs. As attacks against UAVs evolve, future work on the IDS may involve creating new detection modules for newly identified attacks. The detection method behind MAVIDS has the ability to identify novel attacks on a per-sensor basis, however, future work in this area would need to be done to help differentiate between the occurrence of a novel attack versus a non-malicious component anomaly.

To this end, the detection method could be modified and used for other purposes within the UAV domain. Commercial, military and law enforcement UAVs are required to be resilient systems due to their use cases and environment. A UAV is resilient if it can continue to carry out its mission in the face of duress, whether that be from a security threat or otherwise. For example, a component failure such as a motor may cause the UAV to crash. With system resilience the UAV may be able to predict the motor failure and land before a crash, or posses the ability to continue flight while down a motor. In future work, the MAVIDS detection method could be re-purposed to predict system anomalies and component failures. An example of this would be rising vibration within a motor that would ultimately lead to failure.

Outside of the UAV domain, MAVIDS could be applied to other similar cyber-physical and autonomous systems such as unmanned underwater vehicles (UUV) or unmanned ground vehicles (UGV). As the system has been proven to perform well even on small devices such as the Raspberry Pi Zero, it can most certainly be adapted to these systems which do not have such severe constraints.

Bibliography

[1] "Commercial and Military Drone Market Assessment and Forecasts 2016 - 2025". In: *Research and Markets* (2016).

[2] New America. *Who Has What: Countries with Armed Drones*. 2020. URL: `https://www.newamerica.org/international-security/reports/world-drones/who-has-what-countries-with-armed-drones/` (visited on 02/13/2020).

[3] Yasmina Bestaoui Sebbane. *Intelligent Autonomy of UAVs: Advanced Missions and Future Use*. Chapman and Hall/CRC, 2018.

[4] Ahmad Y Javaid, Weiqing Sun, Vijay K Devabhaktuni, and Mansoor Alam. "Cyber security threat analysis and modeling of an unmanned aerial vehicle system". In: *2012 IEEE Conference on Technologies for Homeland Security (HST)*. IEEE. 2012, pp. 585–590.

[5] Katrina Mansfield, Timothy Eveleigh, Thomas H Holzer, and Shahryar Sarkani. "Unmanned aerial vehicle smart device ground control station cyber security threat model". In: *2013 IEEE International Conference on Technologies for Homeland Security (HST)*. IEEE. 2013, pp. 722–728.

[6] Jason Whelan, Abdulaziz Almehmadi, Jason Braverman, and Khalil El-Khatib. "Threat Analysis of a Long Range Autonomous Unmanned Aerial System". In: *2020 International Conference on Computing and Information Technology (ICCIT-1441)*. IEEE. 2020, pp. 230–234.

[7] Vinay Chamola, Pavan Kotesh, Aayush Agarwal, Navneet Gupta, Mohsen Guizani, et al. "A Comprehensive Review of Unmanned Aerial Vehicle Attacks and Neutralization Techniques". In: *Ad Hoc Networks* (2020), p. 102324.

[8] Ewen MacAskill. *US drones hacked by Iraqi insurgents*. 2009. URL: `https://www.theguardian.com/world/2009/dec/17/skygrabber-american-drones-hacked` (visited on 10/30/2019).

[9] D Goward. *GPS spoofing incident points to fragility of navigation satellites*. 2017.

[10] South China Morning Post. *HK$1 million in damage caused by GPS jamming that caused 46 drones to plummet during Hong Kong show*. 2018. URL: `https://www.scmp.com/news/hong-kong/law-and-crime/article/2170669/hk13-million-damage-caused-gps-jamming-caused-46-drones` (visited on 02/13/2020).

[11] Michael Hooper, Yifan Tian, Runxuan Zhou, Bin Cao, Adrian P Lauf, Lanier Watkins, William H Robinson, and Wlajimir Alexis. "Securing commercial wifi-based uavs from common security attacks". In: *MILCOM 2016-2016 IEEE Military Communications Conference*. IEEE. 2016, pp. 1213–1218.

[12] Johann-Sebastian Pleban, Ricardo Band, and Reiner Creutzburg. "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*. Vol. 9030. International Society for Optics and Photonics. 2014, p. 90300L.

[13] Devaprakash Muniraj and Mazen Farhood. "A framework for detection of sensor attacks on small unmanned aircraft systems". In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2017, pp. 1189–1198.

[14] Manuel Cuntz, Andriy Konovaltsev, Achim Dreher, and Michael Meurer. "Jamming and Spoofing in GPS/GNSS Based Applications and Services–Threats and Countermeasures". In: *Future Security Research Conference*. Springer. 2012, pp. 196–199.

[15] Andrei Costin and Aurélien Francillon. "Ghost in the Air (Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices". In: *Black Hat USA* (2012), pp. 1–12.

[16] Industry Canada. *Jamming Devices are Prohibited in Canada: That's The Law*. 2011. URL: `https://www.ic.gc.ca/eic/site/smt-gst.nsf/eng/sf10048.html` (visited on 02/13/2020).

[17] Yasmina Bestaoui Sebbane. *Smart autonomous aircraft: flight control and planning for UAV*. Crc Press, 2015.

[18] Hui-Min Huang, Kerry Pavek, Brian Novak, James Albus, and E Messin. "A framework for autonomy levels for unmanned systems (ALFUS)". In: *Proceedings of the AUVSI's Unmanned Systems North America* (2005), pp. 849–863.

[19] The Dronecode Foundation. *MAVLink - Marshalling / communication library for drones*. 2021. URL: https://github.com/mavlink/mavlink (visited on 05/27/2022).

[20] Anis Koubaa, Azza Allouch, Maram Alajlan, Yasir Javed, Abdelfettah Belghith, and Mohamed Khalgui. "Micro Air Vehicle Link (MAVLink) in a Nutshell: A Survey". In: *arXiv preprint arXiv:1906.10641* (2019).

[21] Open Robotics. *ROS - Robot Operating System*. 2021. URL: http://wiki.ros.org/ (visited on 05/27/2022).

[22] The Dronecode Foundation. *PX4 - uORB Messaging*. 2021. URL: https://docs.px4.io/master/en/middleware/uorb.html (visited on 05/27/2022).

[23] UAVCAN Consortium. *UAVCAN - Uncomplicated Application-layer Vehicular Computing And Networking*. 2021. URL: https://uavcan.org/ (visited on 05/27/2022).

[24] Terry Bandzul. "STANAG 4586–Enabling Interoperability". In: *CDL Systems Ltd* (2010).

[25] PX4. *Basic Concepts - PX4 User Guide*. 2020. URL: https://docs.px4.io/master/en/getting_started/px4_basic_concepts.html (visited on 02/13/2020).

[26] WPPilot. *DJI Matrice 600 Pro UAS unit by D Ramey Logan*. 2017. URL: https://commons.wikimedia.org/wiki/File:DJI_Matrice_600_Pro_UAS_unit_by_D_Ramey_Logan.jpg (visited on 02/13/2020).

[27] Canadian Forces Combat Camera. *Operation CARIBBE*. 2020. URL: https://www.flickr.com/photos/cfcombatcamera/50583388926/ (visited on 02/13/2020).

[28] Sanderux. *DeltaQuad VTOL surveillance UAV.jpg*. 2018. URL: https://commons.wikimedia.org/w/index.php?curid=77480813 (visited on 02/13/2020).

[29] Satish Vadlamani, Burak Eksioglu, Hugh Medal, and Apurba Nandi. "Jamming attacks on wireless networks: A taxonomic survey". In: *International Journal of Production Economics* 172 (2016), pp. 76–94.

[30] Jeff Coffed. "The threat of gps jamming: The risk to an information utility". In: *Report of EXELIS* (2014), pp. 6–10.

[31] DJI. *DJI Adds Airplane And Helicopter Detectors To New Consumer Drones*. 2019. URL: https://www.dji.com/ca/newsroom/news/dji-adds-airplane-and-helicopter-detectors-to-new-consumer-drones (visited on 10/30/2019).

[32] Mohsen Riahi Manesh, Michael Mullins, Kyle Foerster, and Naima Kaabouch. "A preliminary effort toward investigating the impacts of ADS-B message injection attack". In: *2018 IEEE Aerospace Conference*. IEEE. 2018, pp. 1–6.

[33] Donald L McCallie. *Exploring Potential ADS-B Vulnerabilites in the FAA's Nextgen Air Transportation System*. Tech. rep. AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH DEPT OF ELECTRICAL AND ..., 2011.

[34] Pietro Pierpaoli and Amir Rahmani. "UAV collision avoidance exploitation for noncooperative trajectory modification". In: *Aerospace Science and Technology* 73 (2018), pp. 173–183.

[35] Drew Davidson, Hao Wu, Rob Jellinek, Vikas Singh, and Thomas Ristenpart. "Controlling UAVs with sensor input spoofing attacks". In: *10th {USENIX} Workshop on Offensive Technologies ({WOOT} 16)*. 2016.

[36] Azza Allouch, Omar Cheikhrouhou, Anis Koubâa, Mohamed Khalgui, and Tarek Abbes. "MAVSec: Securing the MAVLink protocol for ardupilot/PX4 unmanned aerial systems". In: *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE. 2019, pp. 621–628.

[37] shellIntel. *Drone Code Execution Part 1*. 2015. URL: `https://www.shellntel.com/blog/2015/9/25/drone-code-execution` (visited on 02/15/2020).

[38] Dronecode. *MAVLink packet signing proposal*. 2015. URL: `https://docs.google.com/document/d/1ETle6qQRcaNWAmpG2wz0oOpFKSF_bcTmYMQvtTGI8ns/edit` (visited on 10/30/2019).

[39] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. "Rocking drones with intentional sound noise on gyroscopic sensors". In: *24th {USENIX} Security Symposium ({USENIX} Security 15)*. 2015, pp. 881–896.

[40] Yazhou Tu, Zhiqiang Lin, Insup Lee, and Xiali Hei. "Injected and delivered: Fabricating implicit control over actuation systems by spoofing inertial sensors". In: *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 2018, pp. 1545–1562.

[41] Shadi Khazaaleh, Georgios Korres, Mohammed Eid, Mahmoud Rasras, and Mohammed F Daqaq. "Vulnerability of MEMS gyroscopes to targeted acoustic attacks". In: *IEEE Access* 7 (2019), pp. 89534–89543.

[42] Christopher Kohlios and Thaier Hayajneh. "A comprehensive attack flow model and security analysis for wi-fi and WPA3". In: *Electronics* 7.11 (2018), p. 284.

[43] Ottilia Westerlund and Rameez Asif. "Drone Hacking with Raspberry-Pi 3 and WiFi Pineapple: Security and Privacy Threats for the Internet-of-Things". In: *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*. IEEE. 2019, pp. 1–10.

[44] Ashish Singh and Kakali Chatterjee. "Cloud security issues and challenges: A survey". In: *Journal of Network and Computer Applications* 79 (2017), pp. 88–115.

[45] Abdulaziz Almehmadi and Khalil El-Khatib. "On the possibility of insider threat prevention using intent-based access control (IBAC)". In: *IEEE Systems Journal* 11.2 (2015), pp. 373–384.

[46] Karel Pärlin, Muhammad Mahtab Alam, and Yannick Le Moullec. "Jamming of UAV remote control systems using software defined radio". In: *2018 International Conference on Military Communications and Information Systems (ICMCIS)*. IEEE. 2018, pp. 1–6.

[47] Tuan Phan Vuong, George Loukas, Diane Gan, and Anatolij Bezemskij. "Decision tree-based detection of denial of service and command injection attacks on robotic vehicles". In: *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE. 2015, pp. 1–6.

[48] Hui Hu and Na Wei. "A study of GPS jamming and anti-jamming". In: *2009 2nd international conference on power electronics and intelligent transportation system (PEITS)*. Vol. 1. IEEE. 2009, pp. 388–391.

[49] Andrew J Kerns, Daniel P Shepard, Jahshan A Bhatti, and Todd E Humphreys. "Unmanned aircraft capture and control via GPS spoofing". In: *Journal of Field Robotics* 31.4 (2014), pp. 617–636.

[50] Michelle S Faughnan, Brian J Hourican, G Collins MacDonald, Megha Srivastava, John-Patrick A Wright, Yacov Y Haimes, Eva Andrijcic, Zhenyu Guo, and James C White. "Risk analysis of unmanned aerial vehicle hijacking and methods of its detection". In: *2013 IEEE Systems and Information Engineering Design Symposium*. IEEE. 2013, pp. 145–150.

[51] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Capkun. "On the requirements for successful GPS spoofing attacks". In: *Proceedings of the 18th ACM conference on Computer and communications security*. 2011, pp. 75–86.

[52] Neil Butcher, Angela Stewart, and Saad Biaz. "Securing the mavlink communication protocol for unmanned aircraft systems". In: *Appalachian State University, Auburn University, USA* (2013).

[53] Joseph A Marty. *Vulnerability analysis of the mavlink protocol for command and control of unmanned aircraft*. Tech. rep. AIR FORCE INSTITUTE OF TECHNOLOGY WRIGHT-PATTERSON AFB OH GRADUATE SCHOOL OF ..., 2013.

[54] Benjamin Breiling, Bernhard Dieber, and Peter Schartner. "Secure communication for the robot operating system". In: *2017 annual IEEE international systems conference (SysCon)*. IEEE. 2017, pp. 1–6.

[55] Khalil M Ahmad Yousef, Anas AlMajali, Salah Abu Ghalyon, Waleed Dweik, and Bassam J Mohd. "Analyzing cyber-physical threats on robotic platforms". In: *Sensors* 18.5 (2018), p. 1643.

[56] John McHugh, Alan Christie, and Julia Allen. "Defending yourself: The role of intrusion detection systems". In: *IEEE software* 17.5 (2000), pp. 42–51.

[57] Robin Berthier and William H Sanders. "Specification-based intrusion detection for advanced metering infrastructures". In: *2011 IEEE 17th Pacific Rim International Symposium on Dependable Computing*. IEEE. 2011, pp. 184–193.

[58] NBC News. *Russia has figured out how to jam U.S. drones in Syria, officials say*. 2020. URL: `https://www.nbcnews.com/news/military/russia-has-figured-out-how-jam-u-s-drones-syria-n863931`.

[59] Minhaj Ahmad Khan and Khaled Salah. "IoT security: Review, blockchain solutions, and open challenges". In: *Future Generation Computer Systems* 82 (2018), pp. 395–411.

[60] TH Nasution, I Siregar, and M Yasir. "UAV telemetry communications using ZigBee protocol". In: *J. Phys* 941 (2017), p. 012001.

[61] Diego S Pereira, Mateus Rodrigues De Morais, Luís BP Nascimento, Pablo J Alsina, Vitor G Santos, Daniel HS Fernandes, and Maurício R Silva. "Zigbee Protocol-Based Communication Network for Multi-Unmanned Aerial Vehicle Networks". In: *IEEE Access* 8 (2020), pp. 57762–57771.

[62] Olayemi Olawumi, Keijo Haataja, Mikko Asikainen, Niko Vidgren, and Pekka Toivanen. "Three practical attacks against ZigBee security: Attack scenario definitions, practical experiments, countermeasures, and lessons learned". In:

2014 14th International Conference on Hybrid Intelligent Systems. IEEE. 2014, pp. 199–206.

[63] J. Wright. *KillerBee: Practical ZigBee Exploitation Framework*. 2015. URL: `http://www.willhackforsushi.com/presentations/toorcon11-wright.pdf` (visited on 10/30/2019).

[64] Hichem Sedjelmaci, Sidi Mohamed Senouci, and Tarik Taleb. "An accurate security game for low-resource IoT devices". In: *IEEE Transactions on Vehicular Technology* 66.10 (2017), pp. 9381–9393.

[65] Xiao Tang, Pinyi Ren, and Zhu Han. "Jamming mitigation via hierarchical security game for IoT communications". In: *IEEE Access* 6 (2018), pp. 5766–5779.

[66] Dinghua Wang and Dongqin Feng. "Research on the Strategy of Drones Intrusion Detection Based on Game Theory". In: *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE. 2018, pp. 613–619.

[67] Jianguo Sun, Wenshan Wang, Qingan Da, Liang Kou, Guodong Zhao, Liguo Zhang, and Qilong Han. "An intrusion detection based on bayesian game theory for UAV network". In: *11th EAI International Conference on Mobile Multimedia Communications*. European Alliance for Innovation (EAI). 2018, p. 56.

[68] Hichem Sedjelmaci, Sidi Mohammed Senouci, and Nirwan Ansari. "Intrusion detection and ejection framework against lethal attacks in UAV-aided networks: A Bayesian game-theoretic methodology". In: *IEEE Transactions on Intelligent Transportation Systems* 18.5 (2016), pp. 1143–1153.

[69] Robert Mitchell and Ing-Ray Chen. "Specification based intrusion detection for unmanned aircraft systems". In: *Proceedings of the first ACM MobiHoc workshop on Airborne Networks and Communications*. 2012, pp. 31–36.

[70] Robert Mitchell and Ray Chen. "Adaptive intrusion detection of malicious unmanned air vehicles using behavior rule specifications". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44.5 (2013), pp. 593–604.

[71] Reza Fotohi. "Securing of Unmanned Aerial Systems (UAS) against security threats using human immune system". In: *Reliability Engineering & System Safety* 193 (2020), p. 106675.

[72] G Panice, Salvatore Luongo, Gabriella Gigante, Domenico Pascarella, Carlo Di Benedetto, Angela Vozella, and Antonio Pescapè. "A SVM-based detection approach for GPS spoofing attacks to UAV". In: *2017 23rd International Conference on Automation and Computing (ICAC)*. IEEE. 2017, pp. 1–11.

[73] Menaka Pushpa Arthur. "Detecting Signal Spoofing and Jamming Attacks in UAV Networks using a Lightweight IDS". In: *2019 International Conference on Computer, Information and Telecommunication Systems (CITS)*. IEEE. 2019, pp. 1–5.

[74] Xiaopeng Tan, Shaojing Su, Zhen Zuo, Xiaojun Guo, and Xiaoyong Sun. "Intrusion detection of UAVs based on the deep belief network optimized by PSO". In: *Sensors* 19.24 (2019), p. 5529.

[75] Irvine University of California. *KDD Cup 1999 Data*. 1999. URL: `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html` (visited on 01/14/2021).

[76] Samir Khan, Chun Fui Liew, Takehisa Yairi, and Richard McWilliam. "Unsupervised anomaly detection in unmanned aerial vehicles". In: *Applied Soft Computing* 83 (2019), p. 105650.

[77] Jean-Philippe Condomines, Ruohao Zhang, and Nicolas Larrieu. "Network intrusion detection system for UAV ad-hoc communication: From methodology design to real test validation". In: *Ad Hoc Networks* 90 (2019), p. 101759.

[78] Hichem Sedjelmaci, Sidi Mohammed Senouci, and Nirwan Ansari. "A hierarchical detection and response system to enhance security against lethal cyber-attacks in UAV networks". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48.9 (2017), pp. 1594–1606.

[79] Gimin Bae and Inwhee Joe. "UAV Anomaly Detection with Distributed Artificial Intelligence Based on LSTM-AE and AE". In: *Advanced Multimedia and Ubiquitous Engineering*. Springer, 2019, pp. 305–310.

[80] Kyung Ho Park, Eunji Park, and Huy Kang Kim. "Unsupervised Intrusion Detection System for Unmanned Aerial Vehicle with Less Labeling Effort". In: *International Conference on Information Security Applications*. Springer. 2020, pp. 45–58.

[81] Jason Whelan; Thanigajan Sangarapillai; Omar Minawi; Abdulaziz Almehmadi; Khalil El-Khatib. *UAV Attack Dataset*. 2020. DOI: 10.21227/00dg-0d12. URL: https://dx.doi.org/10.21227/00dg-0d12.

[82] Jason Whelan, Thanigajan Sangarapillai, Omar Minawi, Abdulaziz Almehmadi, and Khalil El-Khatib. "Novelty-based Intrusion Detection of Sensor Attacks

on Unmanned Aerial Vehicles". In: *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*. 2020, pp. 23–28.

[83] scikit-learn. *Examples - scikit-learn 0.24.1 documentation*. 2020. URL: `https://scikit-learn.org/stable/auto_examples/index.html` (visited on 02/17/2020).

[84] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. "LOF: identifying density-based local outliers". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 93–104.

[85] Bernhard Schölkopf, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, John C Platt, et al. "Support vector method for novelty detection." In: *NIPS*. Vol. 12. Citeseer. 1999, pp. 582–588.

[86] Jie Su, Jianping He, Peng Cheng, and Jiming Chen. "A stealthy gps spoofing strategy for manipulating the trajectory of an unmanned aerial vehicle". In: *IFAC-PapersOnLine* 49.22 (2016), pp. 291–296.

[87] Sandra Pérez Arteaga, Luis Alberto Martínez Hernández, Gabriel Sánchez Pérez, Ana Lucila Sandoval Orozco, and Luis Javier García Villalba. "Analysis of the GPS Spoofing Vulnerability in the Drone 3DR Solo". In: *IEEE Access* 7 (2019), pp. 51782–51789.

[88] Diogo Alexandre Martins da Silva. "GPS Jamming and Spoofing using Software Defined Radio". University Institute of Lisbon, 2017.

[89] Dronecode Foundation. *We are setting the standards in the drone industry with open-source*. 2020. URL: https://www.dronecode.org/ (visited on 02/17/2020).

[90] Open Source Robotics Foundation. *Gazebo*. 2021. URL: http://gazebosim.org/ (visited on 04/09/2021).

[91] PX4. *GitHub - PX4/PX4-SITL_gazebo: Set of plugins, models and worlds to use with OSRF Gazebo Simulator in SITL and HITL*. 2021. URL: https://github.com/PX4/PX4-SITL_gazebo (visited on 04/09/2021).

[92] osqzss. *osqzss/gps-sdr-sim: Software-Defined GPS Signal Simulator*. 2021. URL: https://github.com/osqzss/gps-sdr-sim (visited on 05/29/2021).

[93] NASA. *CDDIS —— Data and Derived Products — DNSS — broadcast ephemeris data*. 2021. URL: https://cddis.nasa.gov/Data_and_Derived_Products/GNSS/broadcast_ephemeris_data.html (visited on 05/01/2021).

[94] PX4. *GitHub - PX4/pyulog: Python module & scripts for ULog files*. 2021. URL: https://github.com/PX4/pyulog (visited on 04/09/2021).

[95] PX4 Github. *uORB Messages - PX4-Autopilot/msg at master*. 2020. URL: https://github.com/PX4/PX4-Autopilot/tree/master/msg (visited on 02/17/2020).

[96] A. Bounsiar and M. G. Madden. "Kernels for One-Class Support Vector Machines". In: *2014 International Conference on Information Science Applications (ICISA)*. 2014, pp. 1–4.

[97] Keras. *Keras: the Python deep learning API*. 2020. URL: https://keras.io/ (visited on 02/17/2020).

[98] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016, p. 507. ISBN: 0262035618.

[99] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[100] Django. *The Web framework for perfectionists with deadlines — Django*. 2021. URL: `https://www.djangoproject.com/` (visited on 05/07/2021).

[101] Bootstrap. *Bootstrap - The most popular HTML, CSS, and JS library in the world.* 2021. URL: `https://getbootstrap.com/` (visited on 05/07/2021).

[102] Gulshan Kumar. "Evaluation metrics for intrusion detection systems-a study". In: *Evaluation* 2.11 (2014), pp. 11–7.

[103] NVIDIA. *TensorRT is a C++ library for high performance inference on NVIDIA GPUs and deep learning accelerators.* 2021. URL: `https://github.com/nvidia/TensorRT` (visited on 05/25/2021).