Computers and Electrical Engineering 99 (2022) 107784

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Artificial intelligence for intrusion detection systems in Unmanned Aerial Vehicles[☆]

Jason Whelan [a,*], Abdulaziz Almehmadi [b], Khalil El-Khatib [a]

[a] Faculty of Business and Information Technology, Ontario Tech University, Oshawa, Ontario, Canada
[b] Faculty of Computing and Information Technology, SNCS Research Center, University of Tabuk, Tabuk, Saudi Arabia

ARTICLE INFO

Keywords:
Unmanned aerial vehicles
Intrusion detection
Machine learning
Novelty detection
Cyber–physical systems

ABSTRACT

Unmanned Aerial Vehicles (UAVs) are seeing increased use in critical operations for law enforcement, military, industrial control surveillance and more. These hostile operating environments combined with the UAVs reliance on wireless protocols produces an increased threat level. Many attacks against the UAV are becoming commonplace as they are simple to conduct with inexpensive hardware, such as spoofing and jamming. Unfortunately, as many of these vulnerabilities exist within underlying technologies, securing the UAV becomes a difficult task. A promising approach to identifying and mitigating these attacks is the development of an intelligent intrusion detection system (IDS). The proposed approach uses principal component analysis (PCA) and one-class classifiers to detect attacks. This allows for the use of flight logs for training data, providing a versatile and ubiquitous approach. The proposed detection method is integrated into a fully developed IDS called MAVIDS. This IDS operates onboard the UAV within a resource-constrained agent device, allowing it to detect and potentially mitigate attacks even when communication to the ground control station is lost from jamming. The approach shows to be effective against GPS spoofing and jamming with macro averaged F1 scores of 90.57% and 94.3% respectively.

1. Introduction

The utilization of Unmanned Aerial Vehicles (UAVs) across industries is rapidly increasing. In North America alone, the number of commercial UAVs is estimated to rise to approximately 1.4 million by 2025 with the market reaching $4.2 billion [1]. Commercial UAVs are deployed across industries ranging from agriculture to emergency management and are being used for tasks such as mapping and delivery services. In higher risk domains such as law enforcement and the military, UAVs are being used to conduct missions including payload delivery and surveillance operations. There are currently over 40 countries utilizing UAVs for military operations [2].

With high profile operations in hostile environments, successful attacks against UAVs can have devastating effects to public safety and national security. As UAVs become used more, they become larger targets and the attacks against them become more known. Spoofing and jamming attacks are two of the most known and simplest to conduct, needing only an inexpensive software-defined radio [3–5]. As early as 2015, Global Positioning System (GPS) spoofing attacks have been seen in the wild against UAVs including those operated by the U.S. Customs and Border Protection [6]. Similarly, a GPS jamming attack caused over $100,000 in damages when a swarm of entertainment UAVs in Hong Kong were targeted [7].

☆ This paper is for special section VSI-spc. Reviews were processed by Guest Editor Dr. Atilla Elci and recommended for publication.
* Corresponding author.
E-mail address: jason.whelan@ontariotechu.net (J. Whelan).

https://doi.org/10.1016/j.compeleceng.2022.107784
Received 2 June 2021; Received in revised form 26 January 2022; Accepted 1 February 2022
Available online 15 February 2022

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 research works have presented a threat analysis of the typical UAV, showing the predominantly wireless threats these systems face [3,8]. To combat threats against the UAV, a lightweight on-board intrusion detection system (IDS) is needed. Unlike other proposed solutions, 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 [9]. This is especially important in autonomous UAVs, as it can provide the UAV with the intelligence needed to detect and mitigate attacks without human interaction.

The rest of the paper is organized as follows: Section 2 provides a background on the research problem and related work. Section 3 discusses the one-class novelty detection approach in depth. Section 4 describes the proposed intrusion detection system design. The performance evaluation of the selected methods, including the experiment design, simulation of attacks, and collection of associated flight logs are discussed in Section 5. Finally, we draw our conclusion and discuss future work in Section 6.

2. Background and related works

A 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, a ground control station (GCS), and communication links between the two. With the increase in UAV usage within high risk and hostile environments, the threats against them have also increased. Modern UAVs have been shown to be susceptible to a number of attacks including jamming, denial of service, command injection, spoofing, and more.

Jamming and other denial of service-based attacks are the most simplistic and therefore the most common attacks against a UAV [10,11]. 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 size, weight, computational performance, and power consumption constraints to an on-board agent device.

GPS spoofing is common because most UAVs 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.

As most UAVs 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 20,000 km 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 [12]. 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.

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. Due to vulnerabilities in underlying technologies, securing UAVs can be difficult [3,8]. For this reason, implementing an intrusion detection system (IDS) is a logical approach to improving their security. An 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.

Existing IDS solutions for UAVs typically fall into one of six categories: game theoretic, specification-based, human immune system, anomaly-based, and novelty-based. Many solutions are developed on the basis that they will be deployed within networks of UAVs or MANETS [13,14]. 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 [15,16]. 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 [17,18]. 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 [19–21]. Hybrid approaches can also been introduced, which tend to succeed in one area but lack in another [22].

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. Finally, integrating this method of intrusion detection into an operational IDS can serve to rapidly increase the security of UAV operations with minimal effort.

In addition to the detection method, the placement of the IDS agent should not be overlooked. Without the agent being capable of detecting attacks under duress will inevitably cause the UAV to fall victim. Placing the IDS agent onboard the UAV allows for detection and the potential mitigation of attacks even during duress from jamming and denial of service attacks.

As research in this area is in its infancy, creating an open IDS based on machine learning methods can make better security accessible to even the novice UAV developer. This IDS can also serve as a starting point to help industry introduce IDS technologies.

For the purposes of this research, a one UAV ecosystem was studied and the IDS created for it. The chosen technologies used throughout the paper are part of the Dronecode ecosystem. 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 PX4 flight control firmware, QGroundControl GCS, and MAVLink communication protocol. By using an open source UAV ecosystem, the research and development are streamlined with access to documentation and source code. Of course, the method described in this paper could be replicated to other platforms given its intentional versatility.

3. Novelty-based intrusion detection

Previous research 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 [9]. 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 [23]. 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. By using real world "seed data", accurate training and testing can be completed [24].

3.1. Pre-processing

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. 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.

The particular UAV firmware used for the experiments, PX4, writes flight logs by default in *ULog* format. The PX4 autopilot uses uORB messaging, an inter-process communication protocol, for communication between onboard processes and components. The autopilot writes uORB topics in ULog format to the SD card within the flight controller. The ULog file format is binary and consists of header, definition, and data sections. The header contains the file 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 comma-separated value (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. One CSV file is created per uORB message with the associated fields within. These CSV files are then pre-processed and used for machine learning training.

After the training flight where no attack occurs is complete, the logs can be downloaded from the flight controller. These logs are then extracted into CSV files based on uORB topics using the aforementioned Python script. 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, 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 1

Algorithm 1: Feature Extraction and Clustering

```
Result: CSV dataset per sensor cluster
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
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;
```

shows the pseudo code for the ULog extraction, CSV clustering and merging process. Before any training, the timestamp column is dropped.

Following the clustering and feature extraction phase, the features are standardized into Z-scores. This will standardize the features with a mean of 0 and a standard deviation of 1. This standardization is necessary as the sensor log values can have a broad range of values with different scales. Once standardized, there remains a high number of features. To avoid manual feature selection but still keep the classifier effective, Principal Component Analysis (PCA) is applied. PCA is able 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 its importance [19]. Principal component analysis is a technique which can be utilized for feature extraction. 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. The training set is used to fit PCA, then it is used to transform both the testing and training set.

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

3.2. One-class classifiers

One-class classifiers 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. One-class classifiers are typically categorized into three types of approaches: density-based, and boundary-based reconstruction-based. This paper examines the use of each one of these types of approaches in order to gauge and compare their effectiveness.

3.2.1. Density-based one-class classifiers

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 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) [25]. 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 dimentionality.

3.2.2. Boundary-based one-class classifiers

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 [26]. 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.

3.2.3. Reconstruction-based one-class classifiers

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.

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 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 [19].

4. IDS architecture

Once an effective intrusion detection method has been designed and tested, it can be deployed and including as part of an IDS. MAVIDS (Micro Air Vehicle Intrusion Detection System) is the proposed IDS 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 architecture 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
- On-board 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

The MAVIDS architecture consists of multiple modules and stages from log extraction to detection of attacks onboard the UAV. The UAV operator is first required to conduct a flight with their UAV where no attacks are experienced. Most UAVs will create flight logs by default for troubleshooting purposes, as is the case with the PX4 flight control firmware. The benign flight can be done within a non-hostile area 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 and interaction with the UAV agent. This is the interface the end user will see and

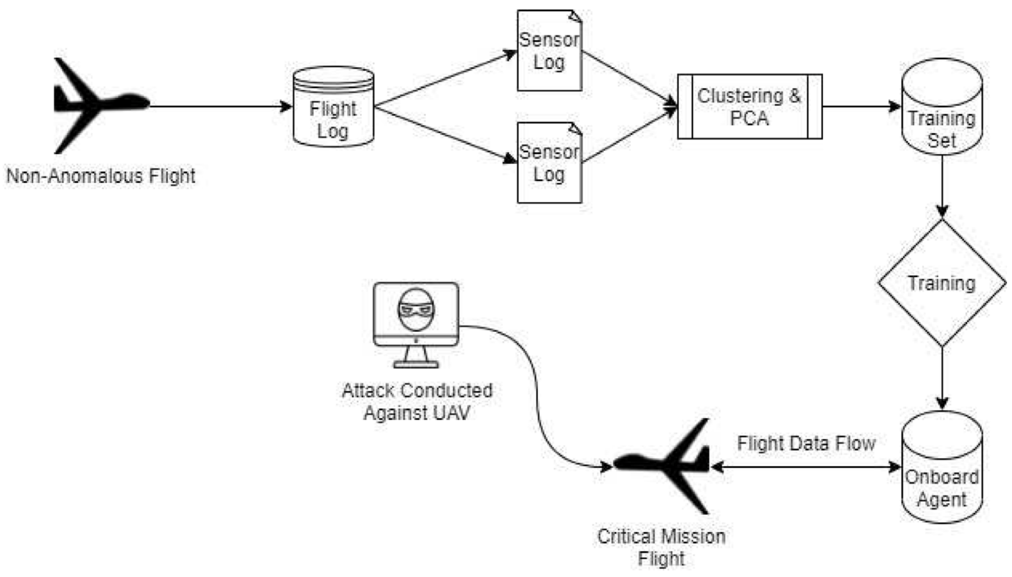

Fig. 1. MAVIDS High Level Architecture.

interact with in order to create a layer of abstraction and therefore simplicity to the user. With the non-anomalous flight logs downloaded from the UAV, they 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. 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 export. This exported 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 executing 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. Fig. 1 shows a high level overview of the MAVIDS architecture.

4.1. 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. 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. 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.

The MAVIDS GCS portal offers four user-accessible views: Dashboard (index), Settings, Training, and Reports. 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 active 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. 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 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 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.

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 the operator to download them for review.

4.2. GCS-UAV communication

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 existing telemetry link. This is done by defining a custom MAVLink message called *MAVIDS* [27]. This message includes all of the information needed by the GCS portal and the UAV agent.

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 also used to communicate the *MAVIDS* message to the IDS agent.

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.

Table 1
Dataset description.

Attack	Benign	Malicious
GPS Spoofing	6078	498
GPS Jamming	6078	1460

Table 2
Features by importance [29].

PC	PX4 Feature name	Description
1	evh	Horizontal velocity error
2	time_utc_usec	UTC time in microseconds
3	lat_y	Latitude 2 (from pre-processing)
4	lat_x	Latitude 1 (from pre-processing)
5	heading	Vehicle Heading (from pre-processing)
6	z_deriv	Down position time derivative (m/s)
7	vz	Z axis velocity
8	ax	North velocity derivative
9	hdop	Horizontal dilution of precision
10	vel_m_s	GPS ground speed (m/s)
11	q [2]	Quaternion rotation from the FRD body frame to the NED earth frame
12	jamming_indicator	PX4 built-in jamming detection
13	vel_e_m_s	GPS east velocity
14	noise_per_ms	GPS RF noise per millisecond

5. Evaluation and results

Given the popularity of GPS spoofing and jamming attacks, these were chosen for evaluating the detection performance and onboard computational performance. A Holybro S500 UAV was assembled and equipped with a Pixhawk 4 flight controller for the following experiments.

5.1. Dataset description

One of the major challenges in UAV IDS research and development is the lack of an available dataset. For the purposes of this paper, live experiments were conducted within the Automotive Centre of Excellence (ACE) research facility at Ontario Tech University. GPS spoofing and jamming are the attacks of choice as they are common and can be conducted using a fairly inexpensive software-defined radio (SDR). A HackRF SDR is used to conduct the attacks as it can broadcast within the GPS bands. 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. Once running, the UAV is able to view up to thirteen "satellites" and obtains a lock. 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.

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 [28]. A daily GPS broadcast ephemeris file is downloaded and provided to the tool for generation. Once the binary data stream is generated, it is transmitted by the HackRF for broadcast. The attack is started after the UAV has flown for a few minutes. After the attack begins, the UAV will become unstable and inevitably crash.

Similarly to GPS spoofing, jamming can be done using the HackRF SDR. 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 −48 dB [4]. To broadcast the signal, an osmocom sink is added and configured for the HackRF. Similar to the GPS spoofing attack, jamming causes the UAV to become unstable and crash. Table 1 shows the resulting observations from the benign flight and the identified malicious observations from the flights where attacks occur.

5.2. Detection performance

The effectiveness of an IDS depends on its detection performance. By using standard metrics, the detection performance can be analysed and compared with other solutions. Common metrics include 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. After PCA is applied, the features which had the most influence are listed in Table 2. More information on each feature can be found within comments in the PX4 firmware message definitions [29]. Table 3 shows the performance results of each classifier and its associated hyperparameters.

Other approaches with comparable attacks tested using the same performance metrics show the success of our detection method [17,21]. The autoencoder specifically either came very close or to, or outperformed, the results of supervised approaches

Table 3
Detection performance.

Attack	Classifier	Label	Precision	Recall	F1 Score	Hyperparameters
GPS Spoofing	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	$k\_neighbours = 2910$
		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	
GPS Jamming	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	$k\_neighbours = 2910$
		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
On-board 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 Pi4 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

which require more maintenance and knowledge from the operator. With the autoencoder having an aggregate GPS spoofing F1 score of .9057, we were able to outperform approaches such as Panice et al. [30] and Wang et al. [21] with F1 scores of .90 and .86 respectively.

5.3. Onboard computational performance

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 such as size, weight, computational performance, and power draw. Three common companion computers are chosen for benchmarking: Raspberry Pi Zero, Raspberry Pi 4 Model B, and the NVIDIA Jetson Nano. The standard Scikit-learn framework is used for OC-SVM and LOF testing. On the Raspberry Pi's, Tensorflow Lite is used for inference whereas the Jetson can make use of CUDA cores and therefore the model is converted and used with TensorRT.

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 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. The results from benchmarking three common companion computers is shown in Table 4. As placing the agent on-board the UAV is a novel approach, there is insufficient data to compare our results to. The performance results do show, however, to have a high enough prediction throughput that will not cause a bottleneck on normal operations.

6. Conclusion

Modern UAVs face many security threats due to their ubiquity and hostile operating environments. Common attacks against UAVs include command injection, denial of service, spoofing, and jamming, to list a few. 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 threats increase, intelligent intrusion detection systems become an essential security tool to help identify attacks and potentially mitigate them. While 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, the UAV domain has large number of variables, making designing intrusion detection systems for UAVs a difficult task.

The MAVIDS solution presented in this paper provides an innovative, machine learning-based UAV intrusion detection system that is effective against a large set of attacks. To overcome the major hurdle in machine learning-based approaches to UAV intrusion detection, mainly the lack of consistent and labelled datasets for training, we used a one-class classification approach to intrusion detection. MAVIDS can be trained to be effective on the specific UAV it will be operating on, while exploiting the use of existing flight logs. Additionally, 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. The simulation results show our approach to be effective against GPS spoofing and jamming with macro averaged F1 scores of 90.57% and 94.3% respectively. In addition, 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 other systems which do not have such severe constraints. Despite these promising results, it will still be interesting to experiment with other machine learning and training algorithms, as well as different types of attacks. Future work could also focus on differentiating between the occurrence of a novel attack versus a non-malicious component anomaly.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Commercial and military drone market assessment and forecasts 2016 - 2025. 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/.
[3] Chamola Vinay, Kotesh Pavan, Agarwal Aayush, Gupta Navneet, Guizani Mohsen, et al. A comprehensive review of unmanned aerial vehicle attacks and neutralization techniques. Ad Hoc Netw 2020;102324.
[4] da Silva Diogo Alexandre Martins. GPS jamming and spoofing using software defined radio. University Institute of Lisbon; 2017.
[5] Cuntz Manuel, Konovaltsev Andriy, Dreher Achim, Meurer Michael. Jamming and spoofing in GPS/GNSS based applications and services–threats and countermeasures. In: Future security research conference. Springer; 2012, p. 196–9.
[6] Goward D. GPS spoofing incident points to fragility of navigation satellites. 2017.
[7] 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.
[8] Whelan Jason, Almehmadi Abdulaziz, Braverman Jason, El-Khatib Khalil. Threat analysis of a long range autonomous unmanned aerial system. In: 2020 international conference on computing and information technology (ICCIT-1441). IEEE; 2020, p. 230–4.
[9] Choudhary Gaurav, Sharma Vishal, You Ilsun, Yim Kangbin, Chen Ray, Cho Jin-Hee. Intrusion detection systems for networked unmanned aerial vehicles: a survey. In: 2018 14th international wireless communications & mobile computing conference (IWCMC). IEEE; 2018, p. 560–5.
[10] Pärlin Karel, Alam Muhammad Mahtab, Le Moullec Yannick. Jamming of UAV remote control systems using software defined radio. In: 2018 international conference on military communications and information systems (ICMCIS). IEEE; 2018, p. 1–6.
[11] Vuong Tuan Phan, Loukas George, Gan Diane, Bezemskij Anatolij. 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, p. 1–6.
[12] Coffed Jeff. The threat of gps jamming: The risk to an information utility. In: Report of EXELIS. 2014, p. 6–10.
[13] Fotohi Reza. Securing of Unmanned Aerial Systems (UAS) against security threats using human immune system. Reliab Eng Syst Saf 2020;193:106675.
[14] Sun Jianguo, Wang Wenshan, Da Qingan, Kou Liang, Zhao Guodong, Zhang Liguo, Han Qilong. 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.
[15] Mitchell Robert, Chen Ray. Adaptive intrusion detection of malicious unmanned air vehicles using behavior rule specifications. IEEE Trans Syst Man Cybern: Syst 2013;44(5):593–604.
[16] Tan Xiaopeng, Su Shaojing, Zuo Zhen, Guo Xiaojun, Sun Xiaoyong. Intrusion detection of UAVs based on the deep belief network optimized by PSO. Sensors 2019;19(24):5529.
[17] Arthur Menaka Pushpa. 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, p. 1–5.
[18] Shafique Arslan, Mehmood Abid, Elhadef Mourad. Detecting signal spoofing attack in UAVs using machine learning models. IEEE Access 2021;9:93803–15.
[19] Whelan Jason, Sangarapillai Thanigajan, Minawi Omar, Almehmadi Abdulaziz, El-Khatib Khalil. 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, p. 23–8.
[20] Bae Gimin, Joe Inwhee. UAV anomaly detection with distributed artificial intelligence based on LSTM-AE and AE. In: Advanced multimedia and ubiquitous engineering. Springer; 2019, p. 305–10.
[21] Wang Shenqing, Wang Jiang, Su Chunhua, Ma Xinshu. Intelligent detection algorithm against UAVs' GPS spoofing attack. In: 2020 IEEE 26th international conference on parallel and distributed systems (ICPADS). IEEE; 2020, p. 382–9.
[22] Condomines Jean-Philippe, Zhang Ruohao, Larrieu Nicolas. Network intrusion detection system for UAV ad-hoc communication: From methodology design to real test validation. Ad Hoc Netw 2019;90:101759.
[23] Irvine University of California. KDD cup 1999 data. 1999, URL http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[24] Straub Jeremy. Development and testing of an intrusion detection system for unmanned aerial systems. In: 2017 IEEE/AIAA 36th digital avionics systems conference (DASC). IEEE; 2017, p. 1–9.
[25] Breunig Markus M, Kriegel Hans-Peter, Ng Raymond T, Sander Jörg. LOF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data. 2000, p. 93–104.
[26] Schölkopf Bernhard, Williamson Robert C, Smola Alexander J, Shawe-Taylor John, Platt John C, et al. Support vector method for novelty detection. In: NIPS, Vol. 12. Citeseer; 1999, p. 582–8.
[27] The Dronecode Foundation. MAVLink - Marshalling / communication library for drones. 2021, URL https://github.com/mavlink/mavlink.
[28] osqzss. Osqzss/gps-sdr-sim: Software-defined GPS signal simulator. 2021, URL https://github.com/osqzss/gps-sdr-sim.
[29] The Dronecode Foundation. PX4-autopilot/msg. 2021, URL https://github.com/PX4/PX4-Autopilot/tree/master/msg.
[30] Panice G, Luongo Salvatore, Gigante Gabriella, Pascarella Domenico, Di Benedetto Carlo, Vozella Angela, Pescapè Antonio. A SVM-based detection approach for GPS spoofing attacks to UAV. In: 2017 23rd international conference on automation and computing (ICAC). IEEE; 2017, p. 1–11.

Jason Whelan received a Bachelor of Information Technology in Networking & IT Security in 2016 and is currently working towards a Master of Science in Computer Science from Ontario Tech University in Ontario, Canada. He has worked as a research assistant on various projects related to penetration testing, vulnerability assessment, and digital forensics. His current research focus is on UAV security and intrusion detection systems.

Abdulaziz Almehmadi, Ph.D., received a bachelor's degree in computer science and a master's degree in information technology security, with a specialty in biometrics, from King Abdulaziz University, Jeddah, Saudi Arabia, and the University of Ontario Institute of Technology (UOIT), Oshawa, ON, Canada, in 2007 and 2010, respectively. Dr. Almehmadi received his Ph.D. in computer science from UOIT in 2015 with a specialty in biometrics and access control. He is an associate professor in the information technology department in the Faculty of Computing and Information Technology (FCIT) at the University of Tabuk, Saudi Arabia. Dr. Almehmadi is an inventor with multiple USPTO patents, including 9,703,952 and 10,559,145 for the design of security-related systems.

Dr. Khalil El-Khatib is a full-time professor in the Networking and IT Security program and the director of Institute for Cyber Security and Resilient Systems at Ontario Tech University. Before joining Ontario Tech's faculty, he was an assistant professor at the University of Western Ontario. He has worked for the National Research Council of Canada as a Research Officer in the Network Computing Group (now Information Security Group) and is a co-director of the Advanced Networking Technologies and Security research lab. His current research interests include: Security and privacy issues in wireless sensor network, mobile wireless ad hoc networks, and vehicular ad hoc networks; Big data and security analytics; Smart grid security; Biometrics; and Smart Communities for Smart Cities.