Linköping University | Department of Computer and Information Science
Bachelor's thesis, 15 ECTS | Information Technology
2021 | LIU-IDA/LITH-EX-G--2021/072--SE

Detecting ADS-B spoofing attacks

- using collected and simulated data

Insamling och simulering av ADS-B meddelanden för detektion av attacker

Joakim Thorn
Alex Wahlgren

Supervisor : Andrei Gurtov
Examiner : Marcus Bendtsen

External supervisor : Suleman Khan

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Copyright

Abstract

In a time where general technology is progressing at a rapid rate, this thesis aims to present possible advancements to security in regard to air traffic communication. By highlighting how data can be extracted using simple hardware and open-source software the transparency and lack of authentication is showcased. The research is specifically narrowed down to discovering vulnerabilities of the ADS-B protocol in order to apply countermeasures. Through fetching live aircraft data with OpenSky-Network and through fetching simulated ADS-B attack data with OpenScope, this thesis develops a data set with both authentic and malicious ADS-B messages. The data set was cleaned in order to remove outliers and other improper data. A machine learning model was later trained with the data set in order to detect malicious ADS-B messages. With the use of Support Vector Machine (SVM), it was possible to produce a model that can detect four different types of aviation communications attacks as well as allow authentic messages to pass through the IDS. The finished model was able to detect incoming ADS-B attacks with an overall accuracy of 83.10%.
Keywords: ADS-B, ATC, Spoofing, Air Communication, OpenSky, OpenScope, Security, SVM, Machine learning

Acknowledgments

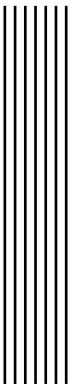
We would like to thank our supervisor, professor Andrei Gurtov, for his continuous guidance and support throughout the semester. We also wish to thank research assistant Suleman Khan for helping us greatly with good inputs, general guidance and helpful insights. Another big thank you to Ola Runeson, contributor to openScope, for great insights in aviation technology. Finally we wish to thank last year's students Anton Blåberg and Gustav Lindahl for letting us use and expand on their openScope-extension to simulate air communication attacks and guiding us through the process.

Contents

List of Figures

List of Tables

List of Listings



Abbreviations

ADS-B – Automatic Dependent Surveillance Broadcast

ATC – Air Traffic Control

ATM – Air Traffic Management

CIA – Confidentiality, Integrity, Availability

CPDLC – Controller Pilot Data Link Communications

DNN – Deep Neural Network

EASA – European Aviation Safety Agency

FAA – Federal Aviation Administration

ICAO – International Civil Aviation Organization

IDS – Intrusion Detection System

RF – Radio Frequency

RTL-SDR – Realtek-Sofware Defined Radio

SDR – Software Defined Radio

SSR – Secondary Surveillance Radar

SVM – Support Vector Machine

VTM – Virtual Trajectory Modification

1 Introduction

This thesis is structured as follows:

- Chapter one explains the problems addressed, including background and research questions as well as related work from which a significant amount of information has been gathered.
- In chapter two we present the necessary background information considered needed to move forward with the thesis.
- Chapter three describes the methodology and aims to walk the reader through the process of what has been done. Several approaches are discussed and briefly evaluated.
- Chapter four presents the results obtained by using the methods from the previous chapter.
- In chapter five we discuss the results obtained, as well as methods used and sources of error.
- The sixth and final chapter presents our conclusions.

1.1 Motivation

As air traffic increases all over the world the need for reliable positioning surveillance is vital [1]. In the context of aviation, there is a tendency of focusing on the safety of commercial flights. A common misconception is that safety and security are equivalent. Safety in regards to aviation systems would more appropriately be defined as minimizing the risk of a potential aircraft failure. Security, however, as this thesis will highlight, can be defined as ensuring that both transmitted and received information is authentic and that the integrity of these messages is maintained. This includes the communication between aircraft and Air Traffic Controllers (ATCs) such as aircraft positioning, velocity and headings. The combination of reliable communication and authentic information can then be translated into what we call security [2].

The current state of communication in civil aviation systems leverages ADS-B, Automatic Dependent Surveillance-Broadcast, which has enhanced air travel safety significantly since

deployed. The fundamental concept of ADS-B is that an aircraft broadcasts unencrypted messages that for example contain position, altitude, velocity, and aircraft identification (ICAO address) to both other aircraft and nearby ATCs. Through this technique, it is possible to accurately acquire an overview of the airspace that could be used as decision-making material for pilots and controllers [3].

1.2 Aim

This paper aims to outline security flaws in air traffic communications and provide a solution to distinguish authentic ADS-B messages from fabricated spoofing attacks. By training a machine learning model on both spoofed and authentic ADS-B messages, the thesis aims to develop an Intrusion Detection System (IDS).

1.3 Research questions

Given the aim presented above, the following research questions could be formulated.

1. Through simple means and limited time, how much flight information can be extracted through air traffic communication, and what difficulties may arise?
2. Since ADS-B is a transparent protocol with very limited security measures, could an IDS be a reasonable solution to this and if so, how can it be implemented?

1.4 Delimitations

Due to the current pandemic with COVID-19, the number of airborne entities is heavily reduced. Hence the possibility to manually collect data was severely limited. As will be further discussed in chapter 5 multiple machine learning algorithms are available, but due to time limitations, only one solution is presented in this thesis. Moreover, given the narrow time frame, only a relatively small data set could be produced.

1.5 Related work

The concern of unencrypted and unauthenticated communication has been addressed in a great number of papers and articles.

Kacem et al. [4] presents similar problems and proposes a solution to detect ADS-B attacks using multiple layers of added security measures. Including predicting an aircraft's flight path, generating secure messages, and adding an extra validation algorithm at the receiver endpoint.

Another proposed solution is given by Braeken [5] who discusses multiple previously proposed solutions. Braeken attempts to show that lack of confidentiality and authentication can be solved by taking the cryptography approach using what she refers to as *Holistic Air Protection* (HAP).

Some of the security problems that come with ADS-B are also shared with the CPDLC protocol. Gurtov et al. [6] lift many of the concerns mentioned previously. They also give a brief overview of multiple types of attacks and propose multiple countermeasures, including cryptography solutions and attempts to solve the identification issues that come within air traffic communication.

Ying et al. [7] do the same thing as we do: producing an ADS-B message data set to use for an intrusion detection system. They then attempt to expand the data set with simulated ADS-B spoofing attacks.

Similar to this thesis, Eskilsson, Gustafsson, Khan, and Gurtov [8] shows the ease of which attacks on ADS-B and CPDLC can be performed. They used the same technique for their ADS-B data transmissions and receival as will be discussed further in this thesis.

As will also be discussed further is the approach that was taken by Lindahl and Blåberg [9]. They extended an open-source ATC simulator in order to implement communication attacks on the simulated aircraft. This proved useful for the work in this thesis as further implementations and additions could be made to their former work.

Related to the topic, Matti, Johns, Khan, Gurtov, and Josefsson [10] explored the possibility of deploying XG (5th and 6th generation) networks within different airspaces. The networks could be rapidly set up using beam-forming and the internet connection would provide features such as enhanced security possibilities as well as higher transmission rates.

2 Theory

In this chapter, the background information needed for this thesis is presented. Most of which is gathered from research papers, trusted online sources, and other relevant sources, all primarily written by trusted professors and other researchers within the field.

2.1 Previous air traffic communication standards

Awareness of the air space has always been crucial for Air Traffic Management (ATM) to work properly. In the earlier days, the backbone of ATM was based on radar technology and is even used today as a complement to other aviation technologies. The decision-making material for coordinating aviation entities, both in air and at e.g. airports, consists of information such as position, direction, and identity. Much of this information was possible to gather using radar technology in combination with other techniques. However, as the airspace gets more crowded, aspects such as real-time data, time-delay, and implementation costs which the mentioned system lacked, became more crucial. Apart from the airspace, the accuracy of aircraft positions at crowded airports is also a key aspect from an ATM point of view [11, 12].

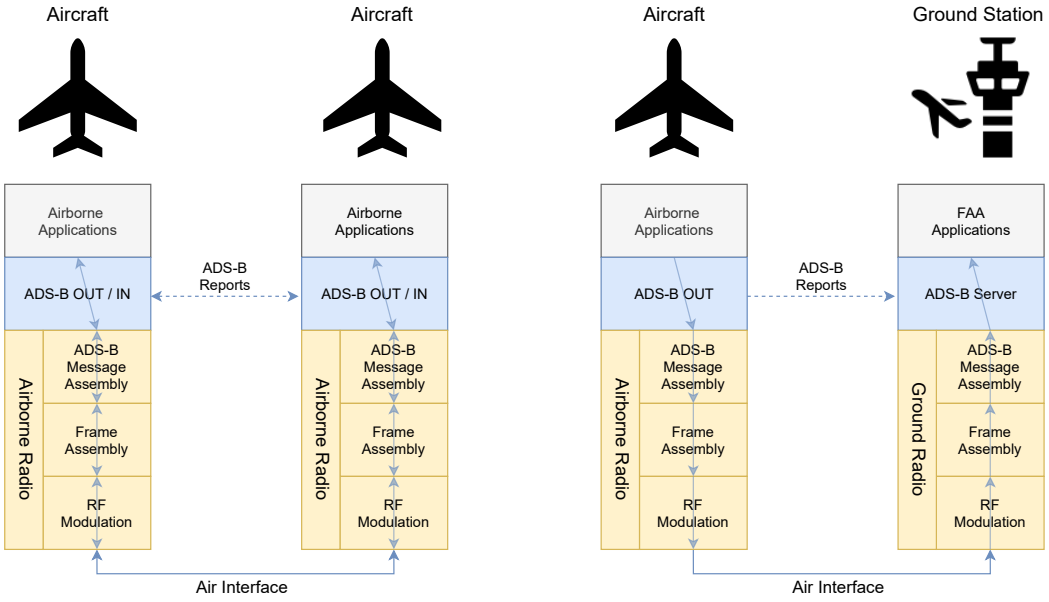
2.2 ADS-B

As the world gets increasingly more globalized, the need for more precise and reliable air traffic surveillance has risen. When ADS-B was developed, it was able to fill this need through the open broadcast of real-time positioning and other useful information to both other aviation entities and ATCs. In contrast to radar technology, ADS-B takes advantage of the Global Positioning System (GPS) transponders that communicate with satellites to retrieve a very accurate estimate of their current position. This makes for better decision-making material for aircraft (equipped with an ADS-B receiver) and the overall ATM. This in combination with the considered low-cost implementation served as a motivation to transition from the radar-based system. As the name suggests, ADS-B is *automatic* in the sense that it transmits *surveillance* data consistently without interference by any third-party actors. The ADS-B system also *depends* on the installed equipment to be handled properly. The transmissions are *broadcast* in a manner so that any entity (e.g. other aircraft or base stations) equipped with a receiver can intercept the messages [12]. The system can be divided into ADS-B OUT and

ADS-B IN. For an entity to be able to broadcast surveillance data through ADS-B, it must be equipped with ADS-B OUT technology. This technology is primarily the broadcast transmitter which is the foundation of the protocol. The reason being that it manages the outbound air traffic communication, and therefore needs to be equipped on the airplane [3]. The receiving part of the ADS-B architecture is referred to as ADS-B IN, which purpose is to gather the broadcast data. This technique is implemented in the ground controllers as a means to improve the ATM. However, as a motivation for aviation entities to use ADS-B, it is also possible for them to be equipped with ADS-B IN, giving pilots an increased air traffic awareness.

The standard frequency on which ADS-B operates is 1090 MHz. On this frequency, it issues both active responses and more general broadcasts. The universal standard of ADS-B is more precisely defined as 1090ES (Extended Squitter). This could be explained through ADS-B transmission being an extension of the Mode-S broadcasting response, which is a selective communication using transponders and unique identifiers called ICAO-addresses. The extended squitter consists of information about the aircraft such as position and velocity, being appended to the Mode-S message [3].

ADS-B can be further dissected into other smaller parts. As presented in Figure 2.1, a simplified version of the ADS-B communication protocol between aircraft can be seen (Figure 2.1a) as well as when ADS-B messages are intercepted by ground stations (Figure 2.1b). The ADS-B protocol stack allows for a simpler and perhaps better understanding of which parts are necessary for a functional communication protocol.



(a) ADS-B Air-to-Air Protocol Stack.

(b) ADS-B Air-to-Ground Protocol Stack.

Figure 2.1: ADS-B Protocol Stack [13].

The ADS-B message

To ensure accuracy between aircraft and ATCs, ADS-B messages containing position and velocity data are broadcast twice every second. Furthermore, messages containing identifiers used to differentiate aircraft are typically broadcast once every 5 seconds. The ADS-B message format uses *Mode-S frames*, which is a data structure primarily used within radar technology (SSR). Mode-S allows ground stations to request a set of data from an aircraft, which in turn replies with the data in combination with its ICAO address. These type of messages are broadcasted using pulse-position-modulation (PPM), giving a data rate of 1 Mbps

(which corresponds to a pulse length of 1 µs) [14]. All this results in a 112-bit message when transmitting an ADS-B message over PPM, which can be seen in Table 2.1.

Bits	Name	Abbreviation
1-5	Downlink Format	DF
6-8	Message Subtype	CA
9-32	ICAO Aircraft Address	ICAO24
33-88	Data Frame	DATA
89-112	Parity Check	PC

Table 2.1: ADS-B message format [15].

The first bits represent the downlink format which describes the format of the rest of the message. The standard value of this field when transmitting ADS-B data is 17, which indicates that the message is an extended squitter (with information about velocity and position etc). The three bits in the capability field represent what capabilities the sending Mode-S transponder has. The ICAO address is appended which consists of 24 bits whereas the actual payload follows with a maximum size of 56 bits. The message is concluded with 24 bits of data that are needed for error detection and to determine whether the message should be discarded due to corruptness or not [14].

Standards and legislations regarding ADS-B

Regarding the regulations of ADS-B OUT technology, there exist certain air traffic authorities that operate within each respective airspace. The Federal Aviation Administration (FAA) operates as such and has the primary purpose of ensuring that flights operating in US airspace are safe [16]. The equivalent of FAA in Europe is the European Aviation Safety Agency (EASA) which has the same purpose of monitoring the airspace and ensuring high safety standards. Related tasks that are handled by these organizations include aircraft inspections, authorization of aircraft design and research targeted at improving overall air traffic safety [17].

The following rules apply in regards to ADS-B:

1. Starting from Jan 1 2020 effectively all aircraft operating over US airspace are required by the FAA to be equipped with at least an ADS-B OUT transmitter [18].

2. Starting from Dec 7 2020 effectively all aircraft operating over EU airspace are required to be equipped with at least an ADS-B OUT transmitter [19].

As of writing this thesis, both of the discussed air spaces have implemented laws that require civil aircraft to be operated with ADS-B technology as can be seen in Figure 2.2 which can be further clarified by Table 2.2.

Figure 2.2: Current requirements for an ADS-B OUT transmitter in US airspace, according to the FAA [18].

Airspace	Altitude
Class A	All
Class B	Generally, from surface to 10,000 feet mean sea level (MSL) including the airspace from portions of Class Bravo that extend beyond the Mode C Veil up to 10,000 feet MSL (e.g. LAX, LAS, PHX)
Class C	Generally, from surface up to 4,000 feet MSL including the airspace above the horizontal boundary up to 10,000 feet MSL
Class E	Above 10,000 feet MSL over the 48 states and DC, excluding airspace at and below 2,500 feet AGL
	Over the Gulf of Mexico at and above 3,000 feet MSL within 12 nautical miles of the coastline of the United States
Mode C Veil	Airspace within a 30 NM radius of any airport listed in Appendix D, Section 1 of Part 91 (e.g. SEA, CLE, PHX) from the surface up to 10,000 feet MSL

Table 2.2: Clarification of the US airspace classes as determined by the FAA [18].

ADS-B vulnerabilities

The idea of ADS-B technology is to have a network with continuous and accessible aircraft data. Therefore, it is in some manner thought to be safe in regards to a large amount of decision-making material. On the other hand, the lack of authentication methods in combination with the transparency could also open up for vulnerabilities. These vulnerabilities are much like the ones found on the internet, and to understand these, one has to take into account the ADS-B protocol stack, as different attacks are executed at different layers [14]. In Table 2.3 an overview of the different attack types is shown with examples [6].

Threat Type	Actor Type	Affected Attributes	Example Attack
Eavesdropping	Passive	Confidentiality	Reading control messages
Jamming	Active	Availability	Channel blocking
Flooding	Active	Availabiliy	Ground station / aircraft flooding
Injection	Active	Availability / Integrity / Confidentiality	Ghost messaging
Modification	Active	Integrity	False message values
Masquerading	Active	Authentication	Ghost aircraft / ground station identity

Table 2.3: Overview and examples of attack types and their respective primary affected CIA attribute [6].

Eavesdropping

The possibility of eavesdropping is a direct consequence of operating with ADS-B as a main feature of the protocol is that it is available. The procedure when eavesdropping on ADS-B communication consists of listening to the broadcast messages from the sending entities. To do this, one has to be equipped with a radio frequency receiver that can intercept transmissions at the most commonly used frequency of 1090 MHz. On its own, this threat type is considered quite harmless, as a core principle of ADS-B is transparency and as it has no impact on the ATM [14]. Flightradar24[1] is a service that for instance applies this technique with a worldwide network of receivers, to get a good appreciation of the airspace in real-time.

Jamming

Jamming, in the context of ADS-B, is a form of denial of service (DoS) attack where the adversary exploits the physical layer of ADS-B. The chain of events in a jamming attack includes interfering with an authentic ADS-B entity by transmitting/requesting messages at a high frequency. This may lead to the transponder becoming overloaded and a resulting downtime prevents the authentic entity to transmit real aircraft data. The most likely approach to this attack is through so-called Ground Station Flood Denial or Aircraft Flood Denial. The attack that targets a ground station is more viable in the sense that the range could be shortened much easier compared to a given aircraft, hence it requires a lesser signal power to perform [14]. Worth noting is that this approach is not specific to the ADS-B protocol as it can be applied to other protocols that use radio frequency-based communication at the physical layer [3].

Message deletion

This attack focuses on removing already present ADS-B messages from the data link channel. There are two approaches to this method and they both take advantage of signals at the same frequency level as the targeted message. The first one attempts to corrupt the message by causing bit errors. Through the parity check field in the message, the receiving ATC will determine if to drop the message due to too many bit errors. The second approach is to destroy the ADS-B message by transmitting a synchronized signal that periodically causes negative interference and thereby cancels out the broadcast message [14].

Message injection

As the ADS-B protocol in itself does not implement any authentication procedures, the sanity checks are primarily focused on the actual ADS-B message payload and structure. This means that if an adversary can control the data link transmissions then it is possible to inject

[1] `https://www.flightradar24.com/`

messages with information about the position, velocity, altitude, etc. that looks legitimate. This could be done through using existing aircraft identification or by using a non-existing one, depending on the purpose [14].

Message modification

Message modification intends to manipulate the data of the messages sent by an authentic aircraft. For a modified message to be broadcast and deemed as authentic by a possible receiver, it requires specific hardware that ensures its legitimacy. As this demands expensive hardware compared to for example a jamming attack, it is less likely. The attack can be executed using a combination of message deletion followed by a message injection, through bit manipulation of the ADS-B message, or by overshadowing the actual message by transmitting the modified one with a higher signal power [14].

2.3 Software

In this section, the main two softwares used will be explained. These will be further discussed throughout the thesis.

OpenSky-Network

OpenSky-Network[2] is an open-source air traffic service that provides live ATC data and aims at "improving security, reliability and efficiency" of the airspace. OpenSky-Network also supplies its open API which can be used to collect live data from all over the planet based on ADS-B receivers, both from organizations and individuals. OpenSky uses so-called state vectors, which are representations of ADS-B data. The difference is that each vector consists of a combination of a sequence of ADS-B messages. For instance, if one ADS-B message contains identifier and speed whereas the other ADS-B message contains position and altitude, the state vector would include the information from both of these messages [20].

OpenScope Simulator

OpenScope[3] is an open-source project that simulates air traffic and allows the user to control it through simplified ATM procedures, such as commands for routing and landing, departing, and alter velocities. When running the software the user is visually presented with an airport surrounded by multiple simulated aircraft which can then be manipulated using commands. If no commands are entered the aircraft will follow the predetermined routes and this can be seen and followed in real-time.

Last year Lindahl and Blåberg [9] presented an extended version of OpenScope[4] that enables the user to perform ADS-B attacks on the simulated aircraft. An expanded GUI was implemented along with four different types of attacks as well as an algorithm that distributes the attacks across the simulated aircraft.

The attacks implemented last year were the following:

- **Jumping aircraft**
  The purpose of this attack is to simulate a message modification attack where the varying data are the positional values. This is done through retrieving authentic aircraft data (in the simulator context), modifying the longitudinal and latitudinal coordinates, and then displaying the false data to the receiving ATC. When the data changes the user can visually see the aircraft "jumping" to an other location within the simulator. With

[2] `https://opensky-network.org`
[3] `https://github.com/openscope/openscope`
[4] `https://gitlab.liu.se/gusli687/openscope-attacks`

the current implementation, data such as velocity and altitude will not be altered and the frequency of these positional modifications are relatively low. After the so-called jump, the aircraft will still have its original destination but the following messages are also fabricated with positional re-calculations until the aircraft reaches it's destination.

In a real-world scenario this could be performed by fetching real ADS-B messages and re-transmitting the message with all the same data, except for the longitudinal and latitudinal fields. However, in order to give such an attack any meaningful impact, the attacker would have to calculate a new route for the aircraft after the jump to be able to keep broadcasting false ADS-B messages.

- **Displaying false data**
  This attack is similar to the one previously mentioned in the sense that it's also a message modification attack. The difference, however, is that this attack impacts other fields (namely velocity and altitude) and it is not continuous. The attack not being continuous means that the ATC receives the false values once in a while and thereby also receives the real values in between.

- **Flooding**
  This attack simulates a large number of aircraft that aren't actually airborne. In a real case scenario, this would mean that a large number of fabricated ADS-B messages are broadcast on the same frequency as other, authentic, ADS-B messages and thereby result in flooding on that data link channel.

- **Non-responding aircraft**
  The attack of non-responding aircraft targets the communication between aircraft and ATCs. The aircraft that are affected by these attacks will not respond to commands issued by the ATC. As this attack is not targeting ADS-B communication, our collection of attack data will not consider this [9].

2.4 Support Vector Machine

Support Vector Machine (SVM) is an algorithm that is good in the sense that it requires low computation power while still providing a very high classification accuracy if implemented correctly. The idea of SVM is to find a so-called hyperplane that separates different classes of an object optimally. This is done by optimizing the margin between the hyperplane and each class to provide room for future classification. Worth noting is that the hyperplane is always one dimension less than the space that defines the data points. For example, if the data points exist in a coordinate system, which is a two-dimensional plane, then the hyperplane will be a one-dimensional line, as can be seen in Figure 2.3 [21].

The marginal distance is the distance between the two parallel margins. The main goal of this method is to optimize the marginal distance v by choosing the best hyperplane.

Support Vectors: The vector points that are incredibly similar to the hyperplane are known as support vector points, and these two data points directly relate to the algorithm's performance, while other data points have no noticeable effects. It is also necessary to note that removing support vectors changes the direction of the hyperplane.

Good Margin: For the majority of entities, the support vectors in the positive and negative classes would be closely but not equally located to the hyperplane, creating a good margin of separation between them.

Bad Margin: The hyperplane that is close to either positive or negative support class vectors is referred to as the bad margin.

Hard Margin: The data point is distinguished from positive and negative classes by keeping the marginal difference between the parallel hyperplanes.

Soft Margin: By drawing a hyperplane, these data points cannot be distinguished from the positive and negative groups. Soft margins are impossible to handle when positive and

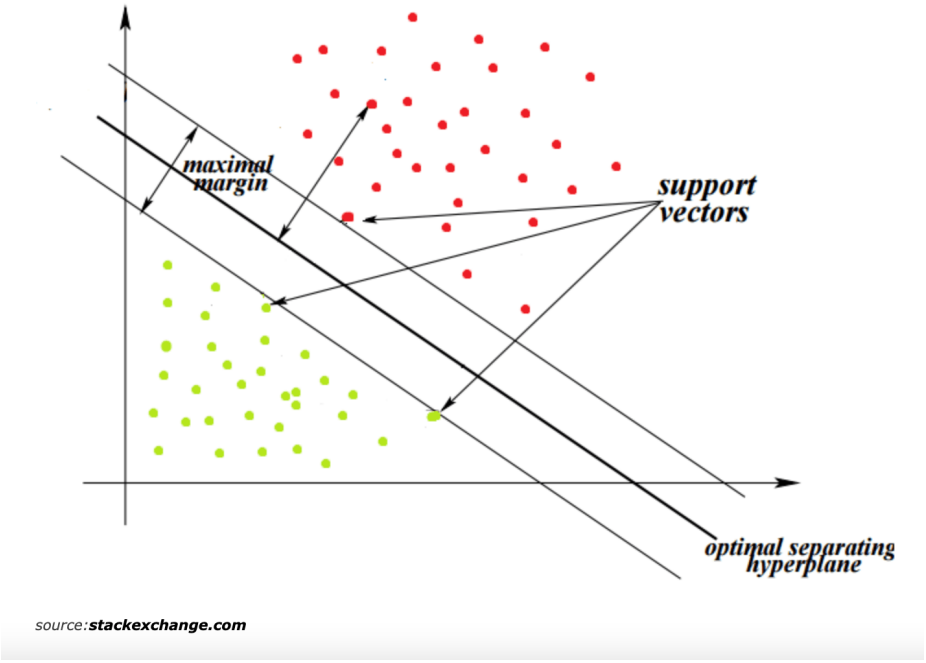


Figure 2.3: Example of an SVM graph [22].

negative data points are combined together, reducing accuracy efficiency. Equation 2.1 gives the line equation.

$$y = xa + b \tag{2.1}$$

The two vectors are S, v.

$$S \begin{pmatrix} -b \\ -a \\ 1 \end{pmatrix} \text{ and } v \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$$

$$\begin{aligned} S^T v &= -b \times (1) + (-a) \times x + 1 \times y \\ S^T v &= y - ax - b \end{aligned} \tag{2.2}$$

The hyperplane equation is given in equation 2.3.

$$S^T v + b = 0 \tag{2.3}$$

Where, b is bias and S, v is vector.

3 Method

This chapter aims to explain the methods used for the data collection, data handling, attack simulation, and lastly the process of producing the IDS. This chapter includes both the thought process as well as the actual methods used. The methods will later be discussed and evaluated in chapter 5.

3.1 Manual collection using hardware

As mentioned previously, ADS-B is split into two different parts – ADS-B IN and ADS-B OUT. To simulate this, two pieces of hardware were used. The first one was the RTL-SDR dongle, which is a USB dongle that connects to an antenna. Combined with the Mode-S decoder *dump1090*[1], it was possible to collect all ADS-B messages within the range of the antenna. With the RTL-SDR dongle connected, the command in listing 3.1 was used to start fetching the ADS-B messages. Using the *–interactive* flag the terminal displays a list of all messages received and using the *–net* flag allows the user to open the web browser and visualize any flights, from which messages have been received, on an actual map. In the example in listing 3.1 and for all tests the frequency was set to 868 MHz, which is a license-free frequency, meaning it can be used by anyone for anything [23]. Because the purpose was to only listen to messages that were sent manually, the 868 MHz frequency was suitable as the intention was to not interfere with the actual air traffic on the 1090 MHz frequency.

```
% ./dump1090 --interactive --net --freq 868000000
```

Listing 3.1: The dump1090 command to receive our own generated ADS-B messages on the 868 MHz frequency using the *interactive* and *net* options.

The second piece of hardware used was the *HackRF One* – a Software Defined Radio. This acts as the ADS-B OUT transmitter and allows the user to transmit messages over any frequency from 1 MHz to 6 GHz [24]. To transmit ADS-B messages, the use of a message encoder was necessary. To this end, an open-source ADS-B OUT encoder[2] was used, which was very suitable as it works very well in combination with *dump1090* and was very simple to get

[1] https://github.com/antirez/dump1090
[2] https://github.com/lyusupov/ADSB-Out

started with. This allowed the user to easily generate ADS-B messages directly in the terminal and set the aircraft's callsign, altitude, position, velocity, and broadcast the message instantly. Both pieces of hardware can be seen in figure 3.1. All software was run on MacOS 11.2.3.

Figure 3.1: Setup used to send ADS-B messages using the HackRF One transmitter, and the RTL-SDR dongle combined with an antenna to receive the same messages.

3.2 Authentic data collection

The original idea was to collect the data using the hardware described above in section 3.1. However, due to the current situation with COVID-19 and the limited amount of flights being in the airspace, this would take too long. Instead, another approach had to be taken. To fetch the needed amount of data, OpenSky-Network was used. By using the REST API mentioned above, it was possible to collect ADS-B data at a reasonable rate. Through setting the area to cover all of central Europe as a request parameter, there were no longer any positional limitations.

To get an understanding of how the ATC data could be received technically, a blog post from Geomatics [25] provided an example of how to use the OpenSky-Network REST API to collect flight data.

While running, the program takes a snapshot of the most recent aircraft data and then saves every ADS-B message from the given area in a .csv file. This in turn allowed for easier handling of the data if further needed, e.g. sorting by a certain attribute or removing unnecessary information to suit the desired data set.

The API collects data with the timestamps in UNIX time. UNIX time is the count of seconds since Jan 1th 1970 which in the year 2021 results in a huge integer. This, however, is not preferred. Therefore a python script had to be written which can be seen in listing 3.2. This simply translates all UNIX time elements into a more intuitive format.

```
import pandas
from datetime import datetime

def changeFormat(unix):
   return str(datetime.fromtimestamp(unix))

df = pandas.read_csv('airdata.csv')
df['time_position'] = df['time_position'].apply(changeFormat)
df['last_contact'] = df['last_contact'].apply(changeFormat)
df.to_csv('../Data/out.csv')
```

Listing 3.2: Python script to change the time elements in the data collection.

By running the API and the script in listing 3.2, quite a lot of data could be retrieved in a short amount of time. With the position settings set over central Europe, the API calls collected between 400-800 messages almost instantly, depending on the number of flights currently in the air.

The final data set was gathered by running the API approximately 30 times during different days and times of the day. By varying the dates and times a varied data set could be collected, which is desirable for this project. The few modifications made were adding a field for labeling, which is necessary for the upcoming machine learning model to determine which aircraft are authentic. Because the API code was running multiple times, each collecting only a part of the final data set, another python script was also written to combine all the minor .csv files into one. As mentioned, for each of these minor files, the time format was altered according to listing 3.2.

3.3 Attack data collection

To complete the final data set falsified data was also required. One possible approach was to send seemingly random ADS-B messages manually and collect them using hardware. This however may not be considered a trusted method as it would be hard to recreate and therefore not considered scientific. Another approach would be to use software to generate false ADS-B messages and append them to the data set. Since it is quite time-inefficient to send the data manually, the hardware approach was illustrated simply to highlight the ease of manipulating aircraft communication data. The final attack data set, however, was produced using software and these steps will be further explained in the following sections.

To produce the attack data the extended version of OpenScope, described in section 2.3, was used. However, as the ADS-B data values are only displayed within the software itself further code implementation had to be added. Primarily a way of extracting mentioned data. This was done by adding a download button into the GUI which triggered an event to continuously record ADS-B data of all affected aircraft into a .csv file. This was done through a loop that, for each instance, ran for 10 minutes and continuously fetched simulated ADS-B messages. The final result was very similar to the .csv file gathered from the OpenSky API, as needed to eventually combine them.

As mentioned, the advantage of using this simulation tool is that it doesn't require hardware as well as not being dependent on the real-time airspace. Due to fewer limitations, this way of producing attack data is therefore considered to represent a bigger part of the final data set.

To further extend the OpenScope project and to get a wider variety of the final data set, more attack types were implemented. Following discussions with supervisors, it was decided to add two new types: *Virtual Trajectory Modification* and *Transponder code alteration*. To make the attacks as confusing and unpredictable as possible, a lot of the implementations are based on randomness. Lindahl and Blåberg [9] implemented an attack distribution method

which, depending on which attacks are activated as well as how many aircraft are simulated, distributes attacks to the aircraft accordingly. Both their attacks and the newly implemented ones include the `Math.Random()` function which allows for different results and attack patterns with every run of the software. The `Math.Random()` function returns a float between 0 and 1 and can therefore be used in multiple different ways.

Virtual Trajectory Modification

The Virtual Trajectory Modification (VTM) attack changes an aircraft's current heading (the direction the aircraft is facing) to seemingly random values. Depending on the settings in the code, the changes to the heading can alter with different magnitude and frequency for an even bigger confusion for the ATC. The code in listing 3.3 is located within a function called `updatePhysics()` which updates the simulated aircraft values multiple times each second. This means that the aircraft's heading, depending on the variable `headingRate` changes quite often. The variable `trueHeading` is a placeholder and keeps track of the aircraft's real heading and is used to calculate the new fake direction of the aircraft. Note that even if `headingDiff` is relatively high, each new heading may receive a small change due to the `Math.Random()` function may return a value close to zero. This is intentional since `headingDiff` is meant to calculate the maximum possible change and not the average.

```
/*
Apply false headings:
headingDiff: How much the fake heading should vary from the true heading
headingRate: Rate of which the heading will change
*/
if(this.attackType == 6 && Math.floor(TimeKeeper.accumulatedDeltaTime) %
    this.headingRate == 0){
   this.heading = this.trueHeading * Math.random() * this.headingDiff;
} else if (this.attackType == 6 && (Math.floor(TimeKeeper.
    accumulatedDeltaTime) % this.headingRate/4) == 0){
   this.heading = this.trueHeading;
}
```

Listing 3.3: Virtual Trajectory Modification attack code.

Transponder Code Alteration

Unlike the VTM attack, the Transponder Code Alteration attack instead changes the transponder code once for each affected aircraft, which never changes again. The transponder code (also known as squawk) is a four-digit number given to each aircraft by the ATC for easier communication and information sharing. It allows the pilot to also send emergency messages by changing the squawk code to any of three predetermined emergency codes, depending on the situation. These emergencies include loss of control of the aircraft, in case of hijacking, as well as loss of communication, and finally a non-specified emergency. The purpose of the squawk attack is to confuse the ATCs by broadcasting alarming values. This attack is split into two parts: invalid values and emergency broadcasts.

Because the transponder code cannot contain the digits 8 or 9, any ADS-B broadcasts using these values should therefore be considered invalid and alarming.

The other part is to change the squawk to one of the three emergency codes. This attack does not change the squawk code back at any time to its original value in its current state. Parts of the implementation for the squawk attack can be seen in listing 3.4.

```
/*
Generates a fake squawk code (with emergency calls or invalid values).
50% chance of emergency call,
50% chance of invalid code including the numbers 8 or 9.
7500 - Aircraft hijacking
7600 - Radio failure
7700 - Emergency
*/
getFakeSquawk(){
   var emergency = ["7500", "7600", "7700"];
   if (Math.random() > 0.5){
      this.hasEmergency = true; // Used for visualization
      return emergency[Math.floor(Math.random()*3)];
   }else{
       var code = ""
       for(var i = 0; i <= 3; i++){
         code += String(Math.floor(Math.random() * 10));
      }
      // Replace a random number in the squawk to an 8 or a 9.
      var index = Math.floor(Math.random()*4);
      code = code.substring(0, index) + Math.round(8+Math.random()) + code.
          substring(index + 1);
      return code;
   }
}
```

Listing 3.4: Transponder attack code.

3.4 Data cleaning

As the purpose of the collected data is to be used as training data fed to a machine learning model, it has to be concise. If the collected data in some manner is misrepresentative, it may lead to flaws in the detection mechanism of the system where it reacts on input where it actually should not, or the other way around. Therefore, the gathered data must be given in a form where it is as representative of the real world as possible.

Because the attack simulation data could not yield misrepresentative values, no data cleaning had to be done for this part. Therefore, the focus of the data cleaning lies with the authentic data. The authentic data originates from OpenSky where the data gathering process is unknown and the probability of getting unwanted, missing, or flawed data is considered higher.

Removing missing fields

When observing the gathered authentic data set, some ADS-B messages contained missing data fields. Many of them were missing transponder codes (squawk) and others were missing their callsign. The appropriate measure was to remove these messages from the data set as an ADS-B message should have these fields to be seen as legitimate, as seen in listing 3.5. The *sensor* and *position_source* fields were not considered to be of importance and therefore those fields were removed completely from the data set as can also be seen in the code below.

```
# load file as pandas dataframe
df = pd.read_csv('dataset.csv')
# remove unnecessary columns
del df['sensors']
del df['position_source']
# remove rows with missing squawk and callsign values
df.drop(df.loc[df['squawk'].isnull()].index, inplace=True)
df.drop(df.loc[df['callsign'].isnull()].index, inplace=True)
# export to new file
df.to_csv('./cleanup.csv')
```

Listing 3.5: Python script for removal of missing values from authentic data set.

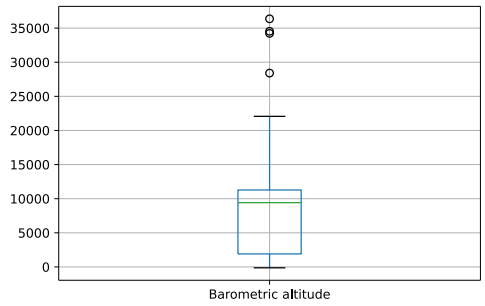
Removing outlying altitudes

Another case to consider when cleaning the data is the possibility of outliers. Outliers are values in a statistical population that are considered to be abnormal in the sense that they are too far away from other values. To put this in the context of the collected ADS-B data, outliers could be messages that have an altitude that is significantly higher than other messages. To keep an even data set, these messages are therefore removed.

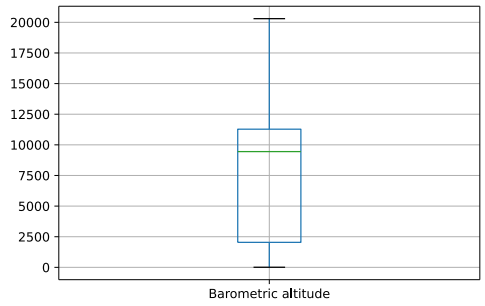
By leveraging statistical functions, the upper and lower bounds were set according to the the following equations, where μ is the mean of the sample, and σ is the standard deviation. Anything outside these bounds were considered to be outliers and therefore removed.

$$\text{Upper bound} = \mu_{baro} + 3 \cdot \sigma_{baro}$$
$$\text{Lower bound} = \mu_{baro} - 3 \cdot \sigma_{baro}$$

Figures 3.2a and 3.2b illustrates the different altitude values before and after the removal of outliers.



(a) A boxplot of the barometric altitudes in the authentic dataset *before* removal of outliers.



(b) A boxplot of the barometric altitudes in the authentic dataset *after* removal of the outliers.

Figure 3.2: Boxplots of the barometric altitudes in the authentic data set. The y-axis displays the measured altitude in meters. The geometric altitudes did not contain any outliers.

As seen in these, outliers only exist in regards to the barometric altitude (viewed as circles in figure 3.2a) and therefore messages with distant values based on this altitude measure are removed from the dataset.

Detecting and removing unreasonable altitude gaps

One problem using the live REST API of OpenSky, documented in their forum[3], is that the altitude may differ by an unreasonable amount within a small period. This means that the authentic data set could contain false data which needs to be removed. Therefore, based on the time and altitude difference it was possible to calculate an average vertical rate between consequent messages of a flight. Using a threshold for the maximum vertical rate allowed, it was also possible to filter out and remove all flights that exceeded this threshold at some point during the flight. The average maximum vertical rate across different aircraft seemed close to 20 m/s and to be sure, the threshold was set to 25 m/s. The python script written for detecting these flights is shown in listing 3.6 below.

```
max_vert_rate = 25
bad_flights = []
for index,row in df.sort_values(by=['icao24', 'time_position']).iterrows():
   if not prev.empty and row['icao24'] == prev['icao24'] and not (row['
       on_ground'] or prev['on_ground']):
      baro_diff = round(row['baro_altitude'] - prev['baro_altitude'])
      geo_diff = round(row['geo_altitude'] - prev['geo_altitude'])
      time_diff = get_time_diff(prev, row)
      baro_vert_rate = baro_diff / time_diff
      geo_vert_rate = geo_diff / time_diff
      if (abs(baro_vert_rate) > max_vert_rate or abs(geo_vert_rate) >
          max_vert_rate):
         if not row['icao24'] in bad_flights:
            bad_flights.append(row['icao24'])
   prev = row
```

Listing 3.6: Python script for detection of flights with unreasonable altitudes between messages.

After removal of outliers from the data set, the non-numeric values were converted to numeric ones. The reason for this is, machine learning algorithms works well on numeric values. Example of this being changing *"True"* and *"False"* in the data set to binary values instead.

[3] https://opensky-network.org/forum

3.5 Applying machine learning

In this thesis, support vector machine (SVM) was chosen for the for intrusion detection system (IDS). SVM is a supervised machine learning algorithm which is mostly used for classification problems. In this thesis, SVM was used to separate authentic messages from anomaly messages. SVC details is given in table 3.1:

SVC	
C	1.0
Break ties	False
Cache size	200
Class weight	None
Coef0	0
Decision function shape	ovr
Degree	8
Gamme	Scale
Kernel	Poly
Max iterations	-1
Probability	False
Random state	None
Shrinking	True
Tol	0.001
Verbose	False

Table 3.1: SVC settings for the final SVM model.

The experiment was carried out using "Google Colab," a Google-provided web Graphical Processing Unit (GPU). The computer used has a Windows 8.1 operating system and an intel I7, 8th generation CPU. Python 3.7 was chosen as the programming language. The authentic and attack data sets was combined into one. 70% of this data set was used for training and validation purpose and 30% of the data set was used for testing and validation purpose. K-Fold validation was used on both training and testing set. In order to measure the performance of the SVM model, per class accuracy score was taken advantage of.

The proposed solution is tested using several performance matrices, including accuracy (A), false alarm rate (FAR), precision (P), recall (R), and F1-Score (F). The matrices described above are based on True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). The confusion matrix is represented in table 3.2.

		Prediction	
		Positive	**Negative**
Actual	**Positive**	TP	FN
	Negative	FP	TN

Table 3.2: Confusion Matrix

4 Results

In this chapter, we present the final results of our work. This includes the results from data hardware testing, data gathering, attack simulations, and the final machine learning results. The chapter is structured similarly to chapter 3.

4.1 Authentic data collection

By using the hardware described in section 3.1 and relatively little software it proved easy to both collect and transmit ADS-B messages manually. Using the open-source project *dump1090* and the commands in listing 3.1 in combination with the ADS-B encoder an aircraft, flying 6666 feet over Campus Valla, Linköping, could be simulated. The commands for the message transmission can be seen in listing 4.1 and the message received can be seen in figure 4.1 along with the *–net* feature in figure 4.2.

Note that all message transmissions were done over the 868 MHz frequency to avoid any risk of confusion for real ATCs or other airborne entities communicating on 1090 MHz.

```
bash-3.2$ python ADSB_Encoder.py 0xABCDEF 58.401362 15.577728 6666
```

Listing 4.1: Command used for *ADSB-OUT* to create our own ADS-B message.

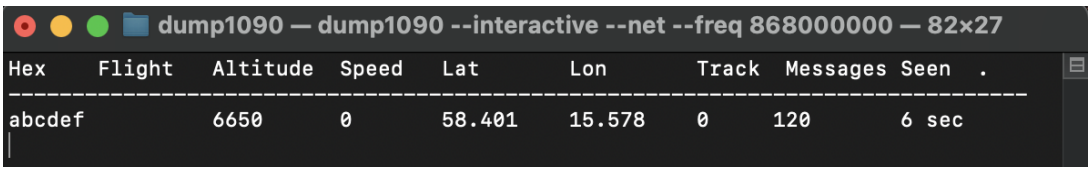


Figure 4.1: Using dump1090 with the *–interactive* displays any received messages directly in the terminal.

Figure 4.2: Using dump1090 with the *–net* flag allows the user visualize the aircraft within the web browser as well.

After fetching the authentic data according to section 3.2 approximately 35000 ADS-B messages were collected and stored into a single .csv file. Since some of the messages, however, were incomplete some data cleaning had to be made. In the case where messages were missing fields due to being on the ground no cleaning was made. Some discrepancies did occur, however, and were solved by removing the message completely from the data set. The most common of these were missing transponder codes (squawks), missing velocities, or geometric altitudes. By applying the data cleaning script described in section 3.4 roughly 9000 messages were removed, meaning that the final authentic data set contained about 26000 messages.

4.2 Attack data collection

When conducting the attack data simulation a majority of the simulated aircraft were affected by an attack. We decided to use the two attack types that we implemented ourselves, as well as two attack types implemented last year to get a wider variety in the data set. The attack settings can be seen in figure 4.3 which is a screenshot taken from the final collection of the attack data. To get a bigger impact of the jumping attack, the probability was set to *high* which results in more frequent jumps. The attack distribution of the data collection can be seen in figure 4.4. Since openScope is originally an ATC simulator, the area is always revolved around airports. The airport used in this thesis and data collection (which can be seen from longitude and latitude values in the attack data) is Seattle-Tacoma International Airport (KSEA).

The attack data collection script was run 6 times which resulted in a total of approximately 25000 messages. Also worth noting is that almost all messages fetched from openScope are affected by an attack and next to no data are gathered from unaffected aircraft.

Also worth mentioning, in regards to the risk analysis in section **??**, is that all of these attacks can be considered message manipulation and consequently fall under the message assembly protocol layer.

Figure 4.3: Settings used for the final attack data collection.

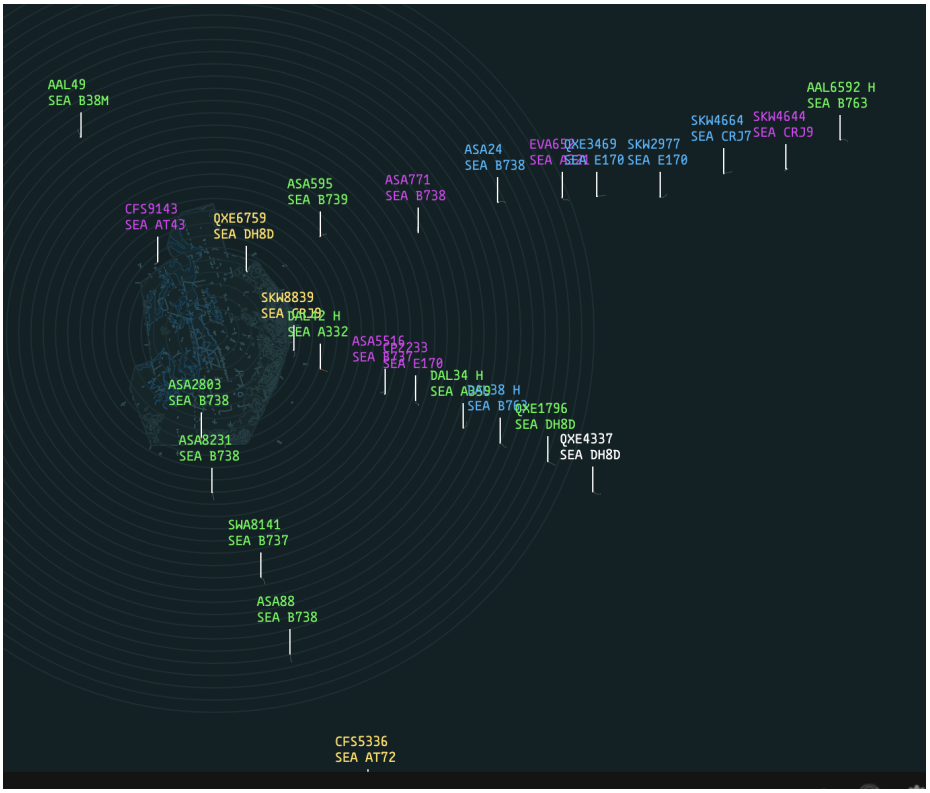


Figure 4.4: Attack type distribution – Blue: False velocity and altitude, Yellow: Jumping aircraft, Purple: VTM, Green: False squawks.

4.3 Machine learning model

Based on the data set produced for the final machine learning model, described in section 3.5, the following results were achieved. The results are also visualized in figure 4.5 below.

For the false heading attack 1433 messages out of 2320 messages were detected correctly, meaning that from all the messages sent using this attack type, 61.80% were detected correctly. The remaining 38.20% of these messages were marked as either other attack types or as authentic messages, scoring the lowest success rate out of the different types. For false information, the success rate landed at 100% as it detected 1752 out of 1752 correctly, while the false squawk attack had an accuracy of 93.90%. The jumping attack landed relatively low at 62.10% and finally, the authentic messages were detected correctly in 7957 out of 7982 cases, landing at an accuracy of 99.70%. Out of all the messages, 14455 out of 15798 were detected correctly giving the model an overall success rate of 91.49%.

What also can be seen in the confusion matrix is that some of the attacks, namely false heading and jumping, have a relatively high percentage of their attacks (8.7% and 8.3%) being predicted as authentic.

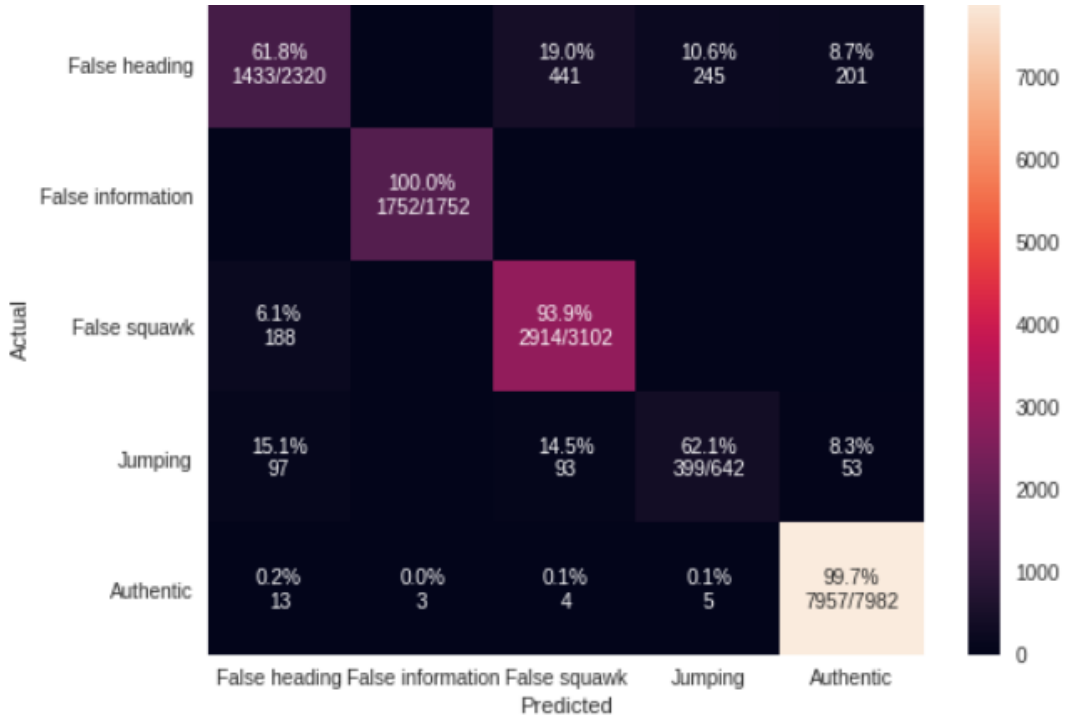


Figure 4.5: Confusion Matrix for SVM. The y-axis represents the actual input to the model whereas the x-axis represents what the model predicts. Thererfore, the diagonal represents the accuracy of correct predictions.

5 Discussion

This chapter aims to discuss and evaluate the used methods and obtained results.

5.1 Method

Even if the desired approach of fetching the messages using hardware was not accomplished, the complete data set still ended up containing 25000 real ADS-B messages. Because of the current situation with Covid-19 and the limited amount of flights around Linköping, the solution of using OpenSky-Network as the data source proved very effective and the final results were satisfactory. Because the project involved using hardware, question one in section 1.3 can still be addressed.

In terms of replicating the process, there should effectively be no problem as the API used is open source and next to no modifications to the code were made. One could argue that more advanced scripts for data cleaning could be used as the ones used for this thesis were quite basic. Another issue could be that the REST API used does not fetch data continuously which limits how far flights can be tracked in the data set. One solution to this would be to implement a loop similar to the one made to openScope to be able to fetch data for a set amount of time.

As mentioned in section 4.2, the attack types used for the attack data set are all some form of message modification. Arguments could be made that a more varied set of attacks would give a better result. However, since the primary focus of this thesis is to detect falsified messages it also seemed reasonable to use attack types that indeed have falsified data and spend less focus on attacks that affect the other protocol layers. These would most likely be more complicated and time-consuming to implement on a realistic level. Regarding this, a judgment call was made on which attacks would yield the best result, both for the thesis and the IDS.

5.2 Results

Overall the results obtained were satisfying in regards to the research questions for this thesis. However, some changes could have further improved the data collection as well as the final intrusion detection system. These parts will be discussed in the following sections.

Data collection

The major downside of using two different tools to collect the data used is the risk of inconsistent data values. For instance, the OpenSky-Network uses the metric system, whereas the openScope simulator uses the imperial system and was then recalculated in the code to suit the other data. Another possible issue is that the data is collected from different places in the world, using central Europe for the authentic data, whereas the attack data is limited to a single airport in the US. One solution to this could be to simulate the authentic data similarly to the attack data using openScope. A choice was however made not to do this since it was desired to have as much real data as possible. In hindsight, however, another area to fetch authentic data from could have been chosen to get an ever wider variety. What was also noticed, after the data collection procedure, was that it is possible to change the airport inside openScope to another one than KSEA. Therefore, another method of reducing the inconsistencies between the authentic and attack data collections would be to fetch attack data from e.g. Frankfurt Main Airport (FRA) which is located in Germany, Europe.

Attack implementation

The aim of the attacks is primarily to cause insecurities within the ATM or to confuse a particular ATC. Therefore the final attacks could be considered less realistic than those performed in real life. This could possibly be solved by changing the variables within each respective attack code without having to completely rewrite the entire attack structure. For this thesis, there was a desire to show that these attacks are possible regardless of what values end up in the final message broadcast. Therefore these values could be further developed to increase how realistically the simulated aircraft are affected. Another argument that could be made is that all the aircraft data are fetched with the same time interval and all "broadcasts" sends the same data with every fetch, which is not the case in real ADS-B as described in section 2.2. In the simulations performed each message contains all data fields of interest and they are all periodically sent simultaneously for all simulated aircraft.

Machine learning and IDS

As mentioned in section 4.3 the success rates of the different attack types vary between 60% to 100%. The reason for this is not necessarily due to one single factor. It is however likely that one major factor is that the *false information attack* is easier detected because it changes more than one field in the message – both altitude and velocity. It also periodically change the values, making it quite obvious for an observer to detect an ongoing intrusion.

Similarly, the *false squawk attack* also has a relatively high detection rate, indicating that the attack significantly differs from the authentic data. One could argue that a more realistic attack would stick to using the legitimate transponder code format, change the squawk to a valid value, and possibly even remove the emergency messages as it makes the attack too obvious for an ATC. This was discussed during the implementation but not used in the final attack. However, as discussed, the main goal of the implemented attacks is to cause confusion, which requires some form of irregularities to be detected.

Aircraft that were affected by the *jumping attack* had less detection rate than expected. This is most likely because the jumps occur quite rarely resulting in the attack not having the expected impact on the data set. This could be improved by either increasing the probability that a jump occurs or changing the impact of the jump itself, by increasing the jump range or also altering the affected aircraft's altitude in the process.

Finally, the *false heading attack* also showed a lower detection rate than expected which may be since no values are invalid or unreasonable. The whole point of this attack is that it shows fast and unexpected heading changes. However, it may be difficult to detect such attacks without a larger data set and an optimal algorithm.

The use of SVM has shown that it is *possible* to produce an IDS using this data collection method. There are however plenty of different algorithms that can be used, and some may work better for detecting certain attacks.

5.3 Future work

Since the presented machine learning model was able to detect some attack types better than others, much work is yet to be done. For example, the classification accuracy of False Heading attacks was only 61.8%, which could have been increased with a larger data set. Focusing on larger data sets would yield the machine learning algorithm more data to practice on which should result in a more accurate model. Other suggestions would be to further experiment and improve the current attack types, as well as implementing new ones.

In regards to machine learning models, one suggestion would also be to take a deep learning approach. Since the data is collected continuously one suggestion is the LTSM deep learning algorithm which could prove better at detecting falsified messages if provided a larger data set. This however could require a more efficient way of collecting data.

Further improvements could also be made to the openScope GUI to better visualize ongoing attacks as well as the data handling for messages.

6 Conclusion

This thesis shows that it is possible to extract information about airborne entities through the usage of simple, easily accessible hardware and open-source software. Although the final data set was not collected through this method, this thesis still highlights the security issues regarding the transparency and authentication of ADS-B. The problems of extracting data with provided equipment were mainly due to the surrounding airspace being vacant of aircraft and the sheer amount of data needed.

After applying the SVM machine learning algorithm, the model was able to identify different ADS-B message types, ranging from 62.1 % to 100 % depending on the message type and overall landed on an average accuracy of 91.49%. The model presented in this paper, therefore, proves to be better than other research implementations, such as the model presented by Ying et al. [7] where the achieved accuracy of the SVM-model was 51.52%. Therefore, using an accurate and well-trained machine learning model to detect ADS-B attacks is a viable approach to strengthen the security of air traffic communication.

Bibliography

[1] *The World of Air Transport in 2019*. URL: https://www.icao.int/annual-report-2019/Pages/the-world-of-air-transport-in-2019.aspx (visited on 03/03/2021).

[2] Martin Strohmeier, Matthias Schäfer, Rui Pinheiro, Vincent Lenders, and Ivan Martinovic. "On Perception and Reality in Wireless Air Traffic Communication Security". In: *IEEE Transactions on Intelligent Transportation Systems* 18.6 (2017), pp. 1338–1357. DOI: 10.1109/TITS.2016.2612584.

[3] Andrei Costin and Aurélien Francillon. "Ghost in the Air(Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices". In: (July 2012).

[4] Thabet Kacem, Duminda Wijesekera, Paulo Costa, and Alexandre Barreto. "An ADS-B Intrusion Detection System". In: *2016 IEEE Trustcom/BigDataSE/ISPA*. 2016, pp. 544–551. DOI: 10.1109/TrustCom.2016.0108.

[5] An Braeken. "Holistic Air Protection Scheme of ADS-B Communication". In: *IEEE Access* 7 (2019), pp. 65251–65262. DOI: 10.1109/ACCESS.2019.2917793.

[6] Andrei Gurtov, Tatiana Polishchuk, and Max Wernberg. "Controller–Pilot Data Link Communication Security". In: *Sensors* 18.5 (2018). ISSN: 1424-8220. DOI: 10.3390/s18051636. URL: https://www.mdpi.com/1424-8220/18/5/1636.

[7] Xuhang Ying, Joanna Mazer, Giuseppe Bernieri, Mauro Conti, Linda Bushnell, and Radha Poovendran. *Detecting ADS-B Spoofing Attacks using Deep Neural Networks*. 2019. arXiv: 1904.09969 [cs.CR].

[8] Sofie Eskilsson, Hanna Gustafsson, Suleman Khan, and Andrei Gurtov. "Demonstrating ADS-B AND CPDLC Attacks with Software-Defined Radio". In: *2020 Integrated Communications Navigation and Surveillance Conference (ICNS)*. 2020, 1B2-1-1B2–9. DOI: 10.1109/ICNS50378.2020.9222945.

[9] Gustav Lindahl and Anton Blåberg. *Simulating ADS-B attacks in air traffic management : By the help of an ATM simulator*. 2020.

[10] Erik Matti, Oliver Johns, Suleman Khan, Andrei Gurtov, and Billy Josefsson. "Aviation Scenarios for 5G and Beyond". In: *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. 2020, pp. 1–10. DOI: 10.1109/DASC50938.2020.9256815.

[11] Donald McCallie, Jonathan Butts, and Robert Mills. "Security analysis of the ADS-B implementation in the next generation air transportation system". In: *International Journal of Critical Infrastructure Protection* 4.2 (2011), pp. 78–87. ISSN: 1874-5482. DOI: https://doi.org/10.1016/j.ijcip.2011.06.001. URL: https://www.sciencedirect.com/science/article/pii/S1874548211000229.

[12] Dean C. Miller William R. Richards Kathleen O'Brienn. "New Air Traffic Surveillance Technology". In: *Boeing Aero Magazine* (2 2010).

[13] Federal Aviation Administration (FAA). *Surveillance and Broadcast Services Description Document, Washington DC., USA, SRT-042, Rev. 01*. 2011.

[14] Mohsen Riahi Manesh and Naima Kaabouch. "Analysis of vulnerabilities, attacks, countermeasures and overall risk of the Automatic Dependent Surveillance-Broadcast (ADS-B) system". In: *International Journal of Critical Infrastructure Protection* 19 (Oct. 2017). DOI: 10.1016/j.ijcip.2017.10.002.

[15] Santiago Toledo, Aurea Anguera de Sojo, j.m Barreiro, Juan Lara, and David Lizcano. "A Reinforcement Learning Model Equipped with Sensors for Generating Perception Patterns: Implementation of a Simulated Air Navigation System Using ADS-B (Automatic Dependent Surveillance-Broadcast) Technology". In: *Sensors* 17 (1) (Jan. 2017). DOI: 10.3390/s17010188.

[16] *Federal Aviation Administration*. URL: https://www.skybrary.aero/index.php/Federal_Aviation_Administration_(FAA) (visited on 04/08/2021).

[17] *About EASA*. URL: https://www.easa.europa.eu/the-agency/faqs/about-easa (visited on 04/08/2021).

[18] *Airspace*. URL: https://www.faa.gov/nextgen/equipadsb/research/airspace/ (visited on 04/08/2021).

[19] *Amendment to the Airspace Requirements on ADS-B and Mode S*. URL: https://www.easa.europa.eu/newsroom-and-events/news/amendment-airspace-requirements-ads-b-and-mode-s (visited on 04/08/2021).

[20] The OpenSky Network. *The OpenSky Network API Documentation*. 2017. URL: https://opensky-network.org/apidoc/ (visited on 03/09/2021).

[21] Ali Kashif Bashir, Suleman Khan, B Prabadevi, N Deepa, Waleed S. Alnumay, Thippa Reddy Gadekallu, and Praveen Kumar Reddy Maddikunta. "Comparative analysis of machine learning algorithms for prediction of smart grid stability". In: *International Transactions on Electrical Energy Systems* (), e12706. DOI: https://doi.org/10.1002/2050-7038.12706. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/2050-7038.12706.

[22] Nitish Kumar. *Introduction to Support Vector Machines (SVMs)*. 2021. URL: https://www.marktechpost.com/2021/03/25/introduction-to-support-vector-machines-svms/ (visited on 05/14/2021).

[23] *Användning av licensfria radiokanaler*. URL: https://pts.se/sv/privat/radio/radiostorningar/anvandning-av-licensfria-radiokanaler/ (visited on 03/18/2021).

[24] *HackRF One*. URL: https://greatscottgadgets.com/hackrf/one/ (visited on 03/24/2021).

[25] Geomatics. *Almost Real Live Data Visualization in QGIs (Air Traffic Use Case)*. URL: https://www.geodose.com/2020/09/realtime%5C%20live%5C%20data%5C%20visualization%5C%20qgis.html (visited on 04/08/2021).