Titre: Title:	Using Ontologies to Detect Anomalies in the Sky
Auteur: Author:	Louis-Philippe Morel
Date:	2017
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Morel, L.-P. (2017). Using Ontologies to Detect Anomalies in the Sky [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2818/

UNIVERSITÉ DE MONTRÉAL

USING ONTOLOGIES TO DETECT ANOMALIES IN THE SKY

LOUIS-PHILIPPE MOREL

DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE INFORMATIQUE)

NOVEMBRE 2017

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

USING ONTOLOGIES TO DETECT ANOMALIES IN THE SKY

présenté par : MOREL Louis-Philippe

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. GAGNON Michel, Ph. D., président

M. FERNANDEZ José M., Ph. D., membre et directeur de recherche

M. DEAN Thomas R., Ph. D., membre et codirecteur de recherche

M. LE NY Jérôme, Ph. D., membre

DEDICATION

À mes parents, pour leur soutien et leurs encouragements durant toutes ces années.

À Cleo, pour m'avoir appuyé et encouragé durant cette aventure. Oui oui! J'ai fini!

Aux joueurs de musique traditionnelle, pour toutes les belles soirées de musique.

To my parents, for their support and their encouragements during all these years.

To Cleo, for supporting me and cheering me on. Yes! I'm really done!

To traditional musicians, for all the amazing evenings of music.

ACKNOWLEDGEMENTS

I would like to thank Professor Jose Fernandez and Professor Thomas Dean for the academic and financial support, without whom this research project would not have been possible.

I would like to thank Gergely Csernak for giving us the permission to use his software, EuroScope, and for helping us understand it.

Finally, I would like to thank Simon Malenfant for working on ontologies with me.

RÉSUMÉ

Ce mémoire de maîtrise présente une solution pour améliorer la sécurité des systèmes de contrôle de trafic aérien. Cette solution prend la forme d'un détecteur d'anomalies qui va déceler les manipulations malicieuses de données. Par les mêmes mécanismes, ce détecteur peut aussi détecter les situations d'urgences et les violations des lois du trafic aérien. Les systèmes de contrôle de trafic aérien sont composés de plusieurs capteurs qui envoient des données aux stations de travail des contrôleurs aérien sur un réseau IP en utilisant un protocole de partage de données en temps réel nommé Data Distribution Service. Des données malicieuses comme de fausses positions d'avions peuvent être insérées dans le trafic du réseau en compromettant une machine connectée à celui-ci ou en émettant des signaux contenant les données falsifiées qui seront captées et transmises sur le réseau par les capteurs. Actuellement, une fois que ces données sont sur le réseau, les systèmes ne disposent pas de mécanismes pour différencier les données malicieuses des vraies données et les traiteront de la même façon. La présence de données falsifiées sur le réseau peut causer de la confusion qui peut mener à des situations dangereuses incluant une sécurité aérienne réduite.

Nous avons évalué l'impact des différentes attaques sur les systèmes de contrôle de trafic aérien en construisant un modèle de menaces tout en considérant les procédures d'urgence déjà en place dans le monde de l'aviation. Nous avons conclu qu'il y a plusieurs façons selon lesquelles un adversaire peut injecter des données malicieuses dans les systèmes. Il peut le faire soit en injectant les données directement dans le réseau ou en utilisant une radio logicielle pour émettre des données malicieuses sur les fréquences utilisées par les capteurs qu'ils les transmettent eux-mêmes sur le réseau. Ces données peuvent induire les contrôleurs de trafic aérien en erreur et leur faire prendre une décision dangereuse. Ce modèle de menaces a servi dans l'ébauche des méthodes de détection.

Pour contrer ces menaces, nous avons conçu un système de détection qui utilise les ontologies pour entreposer les données et un moteur de requêtes ontologiques qui s'occupe des requêtes de détection. Utiliser les ontologies nous a permis d'ajouter de la sémantique dans les données, ce qui facilita la création de requêtes de détection dans le langage de requêtes SPARQL. Le système de détection utilise un tableau d'équivalences entre les structures des données circulant sur le réseau et les concepts ontologiques. Le système de détection est installé sur des machines dédiées et envoie des alertes lorsqu'une requête détecte une anomalie. Nous avons conçu le modèle ontologique en nous basant sur les hypothèses à propos des données qui circulent dans les systèmes

de contrôle de trafic aérien. Concevoir un modèle ontologique qui est assez précis pour pouvoir faire une détection appropriée, mais aussi assez générique pour permettre l'ajout de nouvelles capacités de détection sans trop de problèmes s'est avéré être un défi. Nous avons eu des difficultés à ajouter de nouvelles requêtes de détection sans devoir ajouter des concepts au modèle.

Pour tester notre solution, nous avons conçu et construit un simulateur de système de contrôle de trafic aérien qui se veut une imitation de l'architecture et du comportement d'un vrai système de contrôle de trafic aérien. Le simulateur simule le trafic aérien et l'activité réseau qui en découlerait dans un vrai système. Le but est d'avoir une plateforme pour reproduire et démontrer l'impact des attaques identifiées dans notre modèle de menaces et pour éventuellement tester notre système de détection. Nous avons incorporé des logiciels gratuits utilisés dans les communautés d'amateurs de simulation de vol en ligne pour reproduire les composantes physiques d'un vrai système de contrôle de trafic aérien et atteindre les fonctionnalités de base. Il y a plusieurs fonctionnalités avancées qui ajouteraient au réalisme de la simulation comme la météo que nous n'avons pas pu implémenter.

ABSTRACT

This Master's thesis introduces an anomaly detection solution to increase the security of Air Traffic Control Systems against malicious data manipulation threats. At the same time, this detection system can detect emergencies and air traffic rules violations. Air Traffic Control Systems are made of multiple sensors sending data to air traffic controller workstations over an IP network using a publish-subscribe protocol, Data Distribution Service. Malicious data can be inserted into this network by either compromising a machine on the network, or by tricking the sensors into emitting falsified data. Once into the network, the system currently cannot distinguish malicious data from real one and will treat it as such, potentially causing dangerous situations and general confusion that could lead to air traffic safety being compromised.

We quantify the impact different attacks have on the system by building a threat model while considering existing safety procedures already in place in the aviation world. We found that there are multiple ways an attacker can inject malicious data into the system either directly by injecting false data into the network or indirectly by sending spoofed broadcasts that will be picked up by the ground equipment and in turn injected into the network. These data manipulations can induce an air traffic controller into making a wrong decision. This threat model also gives us direction on how to detect potential threats.

To counter these threats, we design a detection solution using ontologies to store data and a query engine to interact with it. By using ontologies, we can add semantics to the data and facilitate the creation of detection queries in the SPARQL query language. It uses a translation table to convert Data Distribution Service data structures into ontological concepts. The detection engine runs on dedicated machines and sends alerts to the concerned computers if a query detects an anomaly. The ontological model was built using the assumptions we made about the data pieces circulating on the Air Traffic Control System's network. Designing an ontology that is specific enough to be useful for detection, but also generic enough to easily add new detection capabilities proved to be a challenge. We found that we often needed to add new concepts to the ontology when we designed new queries.

To test our solution, we designed and built an Air Traffic Control System Simulator to replicate the architecture and behaviour of a real-life Air Traffic Control System. This simulator models air traffic and the resulting network activity that would incur in an actual system. The goal behind this simulator is to have a platform to reproduce and demonstrate the impacts of the attacks

described in the threat model, and to eventually test the detection solution. We used free software from the aviation gaming industry to reproduce the physical components of a real system and achieved basic functionalities. However, there are advanced features that could be added to our simulator to make the air traffic simulation more realistic such as weather.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ADS	Automatic Dependent Surveillance
ADS-B	Automatic Dependent Surveillance – Broadcast
ADS-C	Automatic Dependent Surveillance – Contract
AI	Artificial Intelligence
ANSP	Air Navigation Service Provider
ATC	Air Traffic Control
ATCS	Air Traffic Control System
ATCSS	Air Traffic Control System Simulator
ATM	Air Traffic Management
CAATS	Canadian Automated Air Traffic Control System-Wide
CAS	Collision Avoidance System
CSV	Comma-separated value
DDS	Data Distribution Service
ES	EuroScope
ESCAT	Emergency Security Control of Air Traffic
FAA	Federal Aviation Agency
FBI	Federal Bureau of Investigation
FSD	Flight Simulation Data
FTP	File Transfer Protocol
GPS	Global Positioning System
HTTP	HyperText Transfer Protocol
ICAO	International Civil Aviation Organisation
IFR	Instrument Flight Rules
IP	Internet Protocol

IT	Information Technology
IVAO	International Virtual Aviation Organisation
MEL	Minimum Equipment List
MLAT	Multilateration
NAS	National Airspace System
NextGen	Next Generation Air Transportation System
OSI	Open Systems Interconnection
PCAP	Packet capture
PSR	Primary Surveillance Radar
QoS	Quality of Service
RF	Radio Frequency
RTPS	Real-Time Publish-Subscribe
SDR	Software-Defined Radio
SPARQL	SPARQL Protocol and RDF Query Language
SSR	Secondary Surveillance Radar
SWIM	System-Wide Information Management
TCAS	Traffic alert and Collision Avoidance System
TCP	Transmission Control Protocol
TESLA	Timed Efficient Stream Loss-toleration Authentication
UDP	User Datagram Protocol
USA	United States of America
VATSIM	Virtual Air Traffic Simulation Network
VFR	Visual Flight Rules
VHF	Very High Frequency
VM	Virtual Machine

VoIP	Voice over Internet Protocol
WAMLAT	Wide Area Multilateration

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

1.1 Modern Air Traffic Control Systems (ATCS)

Air travel has become essential to our way of life and increasingly so. The demand, measured by multiplying the number of passengers with the distance they travelled, grew 5.8 % in 2014 compared to 2013, from 5 806 222 million kilometres to 6 144 510 million kilometres, and 57% compared to 2005, from 3 913 613 million, according to the International Civil Aviation Organisation (ICAO) (International Civil Aviation Organisation, 2016). The organization also indicates that aircraft freight and mail traffic is growing; the combined freight and mail volume went up 5% in 2014 compared to 2013 and 26.4% compared to 2005.

To deal with the increase in air transportation and costs of radar equipment maintenance, the Federal Aviation Administration (FAA) created the Next Generation Air Transportation System (NextGen) program in 2004 (Federal Aviation Administration, 2017). NextGen is an air traffic control system (ATCS) that uses a host of new Internet Protocol (IP) based technologies. The transition from the previous ATCS to NextGen has started in 2011 and the FAA aims to equip all American airports and control centers with the new system before 2019. The FAA has already measured the benefits of this new ATCS: helicopter flights in the Gulf of Mexico are on average 14 nautical miles shorter and use 14 gallons of fuel less per flight on average and more low visibility flights can take-off (Federal Aviation Agency, 2014). The FAA estimates that NextGen will save 14.5 billion USD in fuel, 400 million USD in reduction of carbon dioxide emissions, 2 billion USD in FAA efficiency and 37.1 billion in crew reductions and maintenance costs through 2030 (Federal Aviation Agency, n.d.).

Air traffic control systems are essential tools to monitor air traffic. The goal of these systems is to enable air traffic controllers to make the best decision for a given situation by providing the most accurate information about air traffic and weather conditions. They are a made of a continually expanding collection of equipment pieces and technologies that cover various functions. The most basic function is voice communication between the controller and the pilot, which is covered by very high frequency (VHF) radio broadcasts. A very important part of ATCS is air traffic position acquisition. This is done through various sensors that range from the basic echolocation with radio waves of the primary and secondary surveillance radar (PSR and SSR, respectively), to Automatic Dependent Surveillance – Broadcast (ADS-B), a packet-based data

transfer protocol allowing aircraft to self-report their position. These sensors send data to the air traffic controller (ATC) workstation which in turn displays it on a computer screen to the controller. These displays show, among other things, aircraft positions, identification and tracks, i.e. a list of their previous positions. The more advanced ones show additional information like live weather. Flight plans, which contain flight details such as itinerary and departure and takeoff time as declared by the pilot before the actual flight, are stored in a flight plan database. This database can be queried by controllers to get a particular aircraft's flight plan information or to update its flight plan. Air traffic control systems also keep data about current and forecasted weather. In Canada, data concerning current weather and weather forecasts come from Environment and Climate Change Canada, a governmental department that, among other things, measures weather conditions all across Canada. All of these components are typically connected together through an IP network that links the sensors, the flight plan database, and the air traffic controller stations. Due to the real-time nature of this data exchange requirements, ATCS do not use standard TCP or typical application-layer Internet protocols such as HTTP or FTP. Instead, they use specialized higher-level data transfer protocols such as the Data Distribution Service (DDS) (Object Management Group) communication protocol to share data with other members of the network. One of the main features of DDS is the concept of *data topics* which are predetermined data structures that network members use to send or receive information. The process of sending information is called *publishing* and only the members who are *subscribed* to a topic will receive information about it.

1.2 Security and safety issues with ATCS

ATCS are critical to air traffic safety and attacks against them could have serious consequences. Air traffic controllers and pilots rely on the accuracy of the data gathered by the ATCS sensors and data anomalies or errors can cause them to make the wrong decision, which could lead to accidents. If an aircraft appears at a different position on the radar screen than in reality, controllers may direct a nearby aircraft to change course or altitude which can result in a near-miss or a collision with the first aircraft. This could be caused by spoofed or jammed GPS signals, spoofed position reports, or cyber attack on the ATCS. The ATCS displays could be flooded with false aircraft positions or the radar signal could be jammed preventing them from being displayed, effectively shutting down the control center for the duration of the attack. While there are procedures already in place to address similar situations, the increased responsibilities of the backup control centers could lead to potentially dangerous scenarios, flight delays or

cancellations which would cost a lot of money to airlines. Repeated attacks could cause aviation workers to eventually lose trust in the new technologies.

We focus our attention on attacks that do not aim to shut down or otherwise alter the behavior of the computer systems behind the ATCS, but rather make it unreliable and unusable by inserting false data into it. In other words, we concentrate on attacks on the integrity of the data and not the integrity of the systems.

Altering data on the ATCS can be done by compromising a computer on the internal network and publishing false data onto it which will be picked up along with the real data. Many data exchange protocols such as DDS do not have integrated authentication mechanisms, which makes it possible for the attacker to do this from any machine on the IP network. Finally, even if authentication mechanisms are put in place, an attacker could compromise the machine that is a DDS publisher of sensor-related data topics, e.g. radar data.

The introduction of ADS-B presents a particularly significant security challenge. Unlike older surveillance technologies such as radar, it depends on aircraft to create and send position reports. It is also the only one where the aircraft sends data without prior interrogation by ground surveillance equipment. Since ADS-B reports do not include any authentication information either, it is possible for anyone to create and send spoofed ADS-B position reports by simply transmitting an adequately formatted packet on the frequency used by this technology (1090 MHz), for example by using cheap, readily available software-defined radios (SDR). The spoofed broadcast ADS-B signal will be picked up by aircraft and ground stations alike. These spoofed reports would then be injected into the ATCS, potentially leading to wrong and disastrous decision making by air traffic controllers and pilots. What is particularly insidious in this scenario is that it does not require compromising any machines on the ATCS and would be especially hard to detect using traditional computer security techniques. Finally, since the legitimate ADS-B reports are not encrypted and broadcast, anyone with an adequate receiver can have a real-time view of aircraft positions, which represents itself a problem in terms of privacy and physical threats to aircraft.

Due to the unusual nature of these attacks, traditional computer security techniques like firewalls, antiviruses or intrusion detection systems will be less effective. These techniques are designed to enhance the security of the computers themselves, and detect attack vectors related to attacks on generic computer systems, and not specifically computers used in ATCS. Hence, it becomes necessary to design detection solutions specific to ATCS that could detect such

manipulations of data, whether at the source (e.g. fake broadcast ADS-B reports) or on the network (e.g. compromise and manipulation of computers within the ATCS network).

The need for adequate detection technology for ATCS goes beyond the possibility of direct malicious attack against them. There are several situations in aviation, not necessarily involving malicious intent, that require prompt identification and reaction by pilots and controllers alike. Currently, pilots and controllers detect and handle a variety of such scenarios such as fuel emergencies, approaches under weather minima, and overdue aircraft. With increasing levels of traffic within the same airspace, it becomes necessary to have automated systems that can help controllers detect such situations ahead of time or quickly as they develop. While there are some automated systems that detect and raise alarms in more critical situations such as separation conflicts between aircraft, there are many other situations that are only being detected and handled manually by controllers and for which an automated detection system would be useful and could improve safety and efficiency. For example, an alert system could be built to predict fuel emergencies by tracking fuel levels based on initial fuel levels declared in flight plans, nominal aircraft performance and in-flight information such as weather, route changes, etc. which would allow controllers to better predict and handle such situations by making changes to the route or giving adequate priority to certain aircraft. By correlating predicted aircraft trajectories with weather information, congestion at airports with adverse weather conditions could be identified and resolved earlier, in order to help alleviate delays and predict potentially unsafe situations.

As previously stated, traditional computer security approaches are inadequate to detect malicious data manipulation scenarios, in part because some of them would not require manipulation of computer systems, such as ADS-B report spoofing. The same can be said of the detection of non-malicious potentially dangerous situations. On the other hand, human controllers do routinely identify such situations and could potentially, with enough time and information, detect even some of the more insidious data spoofing scenarios. What allows them to do so is their high-level knowledge of air traffic control, usual traffic patterns, and other information such as aircraft performance, weather, etc. It thus becomes clear that an automated detector must be able to represent and reason on the same high-level abstract concepts that an air traffic controller considers when making these decisions.

This is why we propose the use of *ontologies* as the basis for modelling and construction of such detection solutions. Ontologies are knowledge representation technique that allows humans

to represent concepts and the relationship between them in a manner that is understandable by machines. Ontological databases can be used to store and query data about the represented concepts, and allows machines to make deductions about these concepts and their instances (examples of these concepts) that are in the database. Hence, our proposal is to construct an ontological model of relevant concepts in Air Traffic Control, and populate an ontological database with real-time data about air traffic, that a machine-based automated process can leverage to reason and make deductions about malicious data manipulation and potentially dangerous scenarios.

1.3 Research objectives

The goal of this research project is to propose a method for detecting threats to Air Traffic Control, including malicious manipulation of data and potentially dangerous scenarios in Air Traffic Control Systems that are not necessarily malicious in nature.

Much has been said about the threat that ADS-B spoofing and other malicious manipulations of ATCS represent to ATC and aviation safety. On the one hand, it is easy to imagine doomsday scenarios in the aviation domain where minimal failures can lead to catastrophic consequences. On the other, aviation in general and ATC in particular incorporates many fail-safe procedures and practices put in place to improve safety in many adverse situations. Nonetheless, there is a difference between *safety*, where the objective is to protect against accidents and equipment failures vs. *security* where the objective is to protect against scenarios driven by a determined and capable malicious adversary. While aviation has been traditionally focused on the former, the advent of airborne terrorism in the last few decades has increased security awareness in aviation professionals and organizations. However, the threats that have been considered are mostly physical in nature, e.g. hijacking and aircraft bombing. Accordingly, it becomes important to evaluate the actual risk related to the kinds of cyber attacks targeting the integrity of the data in ATCS we have described above, which leads to our first research question:

1. *What are the actual probability and impact of attacks on the integrity of ATCS data through sensor data spoofing or cyber attacks on the ATCS network, taking into consideration existing safety and security procedures and practices in ATC?*

We have hinted in the last section that detection of malicious manipulations and potentially dangerous scenarios requires high-level concepts and reasoning, and further suggested the use of

ontologies for the construction of an automated detection system for ATCS. Unfortunately, ontological databases are not as widespread as SQL-based relational databases that might be present and used in modern ATCS. Furthermore, temporary data items such as aircraft positions reports and radar tracks will not necessarily be stored in a persistent database. This poses several technical questions on how data-in-motion in an ATCS (such as radar data and weather) and data-at-rest (such as flight plans) can be integrated into a single ontological system onto which our proposed detector can get the necessary information to make its deductions and detections. This raises our second research question:

2. *How can an ontological system be viably constructed atop an ATCS, integrating both stored information and real-time data obtained from ATC sensors, while maintaining adequate speed of detection*?

Verifying the functionality, performance and discrimination of such a detection system for ATCS represents in itself a formidable challenge. First of all, the details of the architecture, components and software of ATCS are very expensive and closely guarded by their vendors and the Air Navigation Service Providers (ANSP) that operate ATC and ATCS for civil aviation authorities, partly for commercial reasons, but also for national and aviation security reasons. Thus, we needed to propose an alternate approach to evaluate our ATCS detection solution that would involve testing against an operational ATCS-like system, while retaining a certain level of realism. This requirement has two important parts. First, it would be necessary to reproduce with adequate fidelity the Information Technology (IT) infrastructure behind the ATCS in order to reproduce and evaluate the performance of the solution and its detection capabilities against traditional cyber attacks. Second, we need to reproduce realistic data streams that would be present in a typical ATCS and, furthermore, have the ability to measure and demonstrate the real-world consequences of such data being manipulated. Finally, we need to be able to reproduce in an efficient and accurate manner the various attack scenarios and dangerous situations that we will be evaluating our solution against. How to integrate the cyber and physical world aspects of ATCS in a reproducible evaluation approach represents an important challenge addressed by our third research question:

3. *How to construct an evaluation platform for ATCS detection solutions that has 1) a high level of fidelity of the cyber aspect of the ATCS, 2) reproduces and demonstrates real-world physical impact of attacks and dangerous*

scenarios on ATC, and 3) can easily be used to reproduce results of previous evaluations?

One of the main difficulties and challenges in the use of ontologies is to determine the adequate level of detail vs. abstraction. A good balance will allow the ontological model to be complete enough to implement the required deduction logic for various situations, while at the same time make the construction and maintenance of the ontological model a manageable proposition for the domain experts that will be involved in that process. Indeed, a very wide and all-encompassing ontology will allow, in our case, the domain expert to easily represent all detection rules as interrogations or queries onto the ontological database. However, the process of building such an ontology could be unmanageable in terms of time and effort, and, furthermore, become too specific to a particular scenario, a particular ANSP organization, or a particular ATCS. The difficulties related to this delicate balance between detail and abstraction, specificity and flexibility, are at the core of the following fourth and last research question:

4. *How do we build an ontology that is complete enough to address the identified attack scenarios, but general and flexible enough to cover future, unpredicted scenarios while remaining easily maintainable?*

1.4 Thesis roadmap

This Master's thesis introduces an ontology-based data anomaly detection system to address malicious and non-malicious threats to air traffic control systems. This solution is based on an ontological model of the data generated by an ATCS and ontological queries designed to detect anomalies caused by attacks on the system or potentially dangerous situations. We also provide a threat model to better guide our detection queries. Finally, we present an architecture for a platform that emulates an ATCS in order to test our solution.

We begin by providing in Chapter 2 background material to familiarize the reader with the domain of air traffic control and its related technologies. We also discuss provide a brief description of the threats against them and discuss the relative (in)applicability of existing IT security solutions. Finally, we provide a description of ontologies and how they have been used in security and aviation-like application domains.

In 0, we describe the concept and architecture for our ontology-based ATCS detection solution. We describe in Chapter 4 the air traffic control system simulator (ATCSS) that we designed to

demonstrate and evaluate the detection solution we proposed. In Chapter 5, we describe our threat model in more details, as well as the details of the ontology and the ontological queries that are the core of the detector. Finally, we analyze the design process of the solution and we evaluate the architecture of the simulation environment in the conclusions in Chapter 6.

CHAPTER 2 A REVIEW OF ATC AND RELATED TECHNOLOGIES

2.1 Air traffic controllers

The role of air traffic controllers is to direct air traffic in order to maintain aircraft safety and efficiency. They take charge of aircraft at the departure gate and release them only at the arrival gate. They are constantly in contact with them in between. An air traffic controller in a given area will be assigned a specific frequency will use that frequency to talk to pilots in his area. Over the course of a flight, a pilot will talk to multiple controllers. The first controller, named *apron*, will instruct the pilot on his way to the takeoff runway. Then, a series of controllers will take responsibility for takeoff instructions and for the area surrounding the airport, namely the *tower* and *terminal* controller. Once the aircraft is far enough from the departure airport, the control responsibility falls unto the regional control centers, called *area*. Then, a controller is assigned to the pilot until the aircraft leaves the control area or approaches the destination airport. If the flight crosses an ocean, the pilot will be instructed by a controller called *oceanic* that take care of oceanic routes and time slots. Controllers can, among other things, order an aircraft to change altitude in order to respect minimum separation distances with other aircraft or a route change depending on weather conditions or traffic density in the aircraft's projected trajectory. Pilots can also request those changes to the controllers, but the latter make the final decision.

2.2 Air traffic control systems (ATCS)

Air traffic control systems are at the heart of the civil and commercial aviation infrastructure. They are an aggregation of systems and data acquired by different sources and are a critical and essential tool for air traffic controllers. Each air navigation service provider (ANSP) owns their own; the FAA in the United States has NextGen and NAV CANADA developed and operates the Canadian Automated Air Traffic Control System (CAATS), to name a few. These systems receive data from a slew of sensors including primary and secondary surveillance radars, ADS-B antennae, weather information providers and flight plan databases. In CAATS, data travels through a private IP network using the DDS protocol (Real-Time Innovations, 2013), which we talk about in more details in section 2.7.

2.2.1 Sources of data in ATCS

2.2.1.1 Primary surveillance radar (PSR)

The first technology we cover is Primary Surveillance Radar (PSR). The Allied military developed it during the Second World War and then adapted it for civil use. These radars send an electromagnetic impulse that bounces off aircraft and other physical objects such as birds and mountains. When the radars receive the bounced impulse, they compute the distance between themselves and the perceived aircraft using the travel time of the impulse. The primary surveillance radar can also measure the azimuth of the aircraft by using the angle the radar was facing when it received the impulse. The radar then deducts the longitude and latitude of the aircraft using the distance and azimuth. However, this surveillance device cannot measure the altitude of an aircraft. Primary surveillance radars are autonomous in the sense that they do not need aircraft to carry a special instrument to detect them.

2.2.1.2 Secondary surveillance radar (SSR)

The Secondary Surveillance Radar (SSR) is a technology that completes the primary surveillance radar by providing the missing data needed for an accurate picture of the air traffic. It is a by-product of the technological advancements made during the Second World War. Originally, the Allied military developed this technology to be able to differentiate enemy aircraft from allied ones, something the primary surveillance radar cannot do. Aircraft need to have a transponder to communicate with the secondary surveillance radars using radio frequencies. The pilot enters a code made of four numbers each ranging from 0 to 7 called SQUAWK on the aircraft's transponder. The transponder then broadcasts the code when a secondary surveillance radar interrogates it. This code becomes the aircraft's identifier for the duration of its flight. A transponder can also communicate with other transponders such as in the Traffic Alert and Collision Avoidance System (TCAS). For civil aircraft, transponders can operate using three modes: A, C, or S, chosen by the interrogating radar. If a transponder uses mode A, it transmits only the SQUAWK code given by the air traffic controller to the pilot. If a secondary surveillance radar interrogates a transponder operating under mode C, the transponder transmits the flight level, i.e. the aircraft's altitude by increments of 100 feet, as read on the aircraft's altimeter. Due to those advantages, most transponders use mode C with mode A. Finally, mode S allows the transponder to transmit a unique

24-bit identification number, different from the SQUAWK code given by the air traffic controller, as well as a data field of variable length.

2.2.1.3 Automatic Dependent Surveillance (ADS)

ADS is a more cost-efficient and precise surveillance technology that can provide better coverage in areas where access is difficult like the Hudson Bay and the rest of the Great Canadian North. To be able to transmit ADS data, an aircraft needs a GPS receiver and a transponder. The transponder is the same one that transmits data to secondary surveillance radars and uses the mode S data field to relay the data. The first version of this technology is ADS-C. Under this communication protocol, a contract is established between the ADS ground receiver and the aircraft's ADS system to specify which data will be transmitted and when. When the data transmission conditions meet the conditions specified in the contract, the ADS-C system will broadcast the data specified in the contract to a ground station. ADS-B is the most important ADS implementation for data acquisition in ATCS like NextGen and CAATS. This technology is made of two features: ADS-B In and ADS-B Out. Contrary to ADS-C, aircraft equipped with ADS-B Out systems broadcast their flight data to whoever can receive it without having a data contract with them. Data receivers are ground stations or another aircraft's ADS-B In system. When an aircraft's ADS-B In system receives data, it updates its airspace display with the new information. That way, pilots have an accurate representation of the other aircraft around them. When ground stations receive aircraft data, they relay that information to the ATCS to update it with the most recent information. NAV CANADA plans to use the Iridium satellite constellation as ADS-B receivers and transmitters to achieve a more complete coverage of its surveillance territory (NAV CANADA, n.d.). Since ADS-B data is unencrypted and not authenticated, antennae owned by civilians can pick up broadcasts. This is how flight tracking websites gather their data. There is currently no way to verify the integrity or the authenticity of ADS data.

Table 2.1 depicts the differences between the three major surveillance technologies, as well as the advantages and disadvantages of each of them.

Table 2.1: Comparison of the main surveillance technologies

Sensor	PSR	SSR	ADS-B
Data acquired	Azimuth Distance	Azimuth Distance SQUAWK Altitude Identification number	Identification number GPS position Altitude Airspeed Ground speed Route
Cost per aircraft (USD)	0	3000-4000[1]	4000[1]
Cost per antenna (USD)	6 million[2]	3 million[2]	100 000 to 400 000[3]
Security features	Laws of physics	Multilateration when available	None
Advantages	Hard to deceive Independent	Accurate data	Complete data set Cost
Disadvantages	Cost Needs to be used with SSR for complete data set	Cost Needs to be used with PSR for complete data set Aircraft needs a transponder	No security Aircraft needs a transponder and ADS-B equipment

2.2.1.4 Flight plans

Flight plans are a tool used by pilots to provide various information to controllers about aircraft and the route they want to take. Pilots provide them in advance and generally need approval from the ANSP before the flight can take place. The content varies depending on the ANSP, but they generally include information about the aircraft such as identification number, type, capacity, as well as information about the flight such as departure and arrival airports, cruising speed, altitude, and route, which consists of a series of waypoints through which the aircraft will go. The pilot also needs to specify whether the flight is under Instrument Flight Rules (IFR) or Visual Flight Rules (VFR). The set of rules chosen largely depends on the weather conditions along the flight's route. VFR requires the pilot to be able to see clearly where the aircraft is going. If those conditions are not met, the pilot has to fly using the onboard instruments, hence the name IFR. A controller can modify a flight plan after the aircraft has taken off due to traffic conflicts or at a pilot's request.

[1] (Gulf Coast Avionics)

[2] (International Civil Aviation Organisation - Asia and Pacific Office, 2007)

[3] (Perreault, 2015), (Dunstone, 2012)

They are stored in a database in the ATCS and the system fetches and displays them one of the many screens that are part of a controller's workstation.

2.2.1.5 Aircraft track

A track is the modelling of an aircraft's movement. It contains information about the flight plan, the position of the aircraft as reported by radar or where the aircraft reported its position with ADS-B, and other data points used by the controller over the duration of the flight from the departure gate to the arrival gate. Surveillance radars, ADS-B antennae, and ATC, through directives like route change, update the track. ATCs use this to keep track of where an aircraft is, was, and where it is going in order to give the appropriate directives.

2.2.2 Typical Modern ATCS Architecture

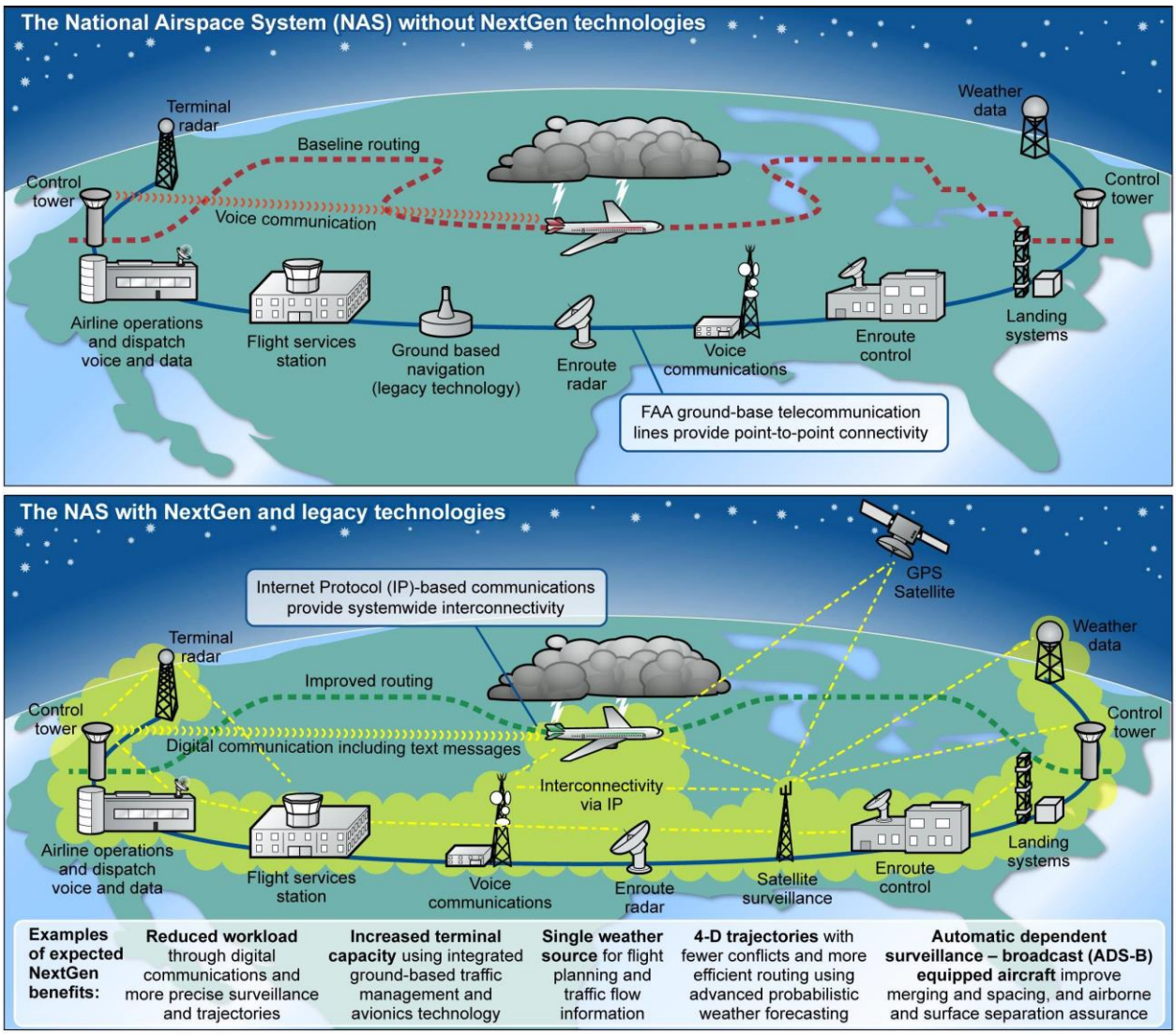

Figure 2.1: Modern ATCS architecture

As illustrated in Figure 2.1, an ATCS is made of multiple machines that have different roles. Some of them are sources of data. Examples of such machines are PSR, SSR and ADS-B. These machines send data over the network. Other machines are both data producers and consumers, like the flight plan database – comprised in the *Flight services station* on Figure 2.1 – and the controller workstation, included in the enroute control and in the control tower in the figure. The controller workstation is the computer terminal where air traffic controllers sit and work. They usually have computer screens to display aircraft position, and a screen to display flight plans. These data-

consuming machines receive data from the producers and interpret it or store it depending on their role in the system. In the controller workstation's case, it sends data whenever there is a flight plan modification. Modern machines are regular computers with generic operating systems that run custom software, as opposed to the custom-made black boxes of the past. All of these machines are connected to each other through an IP network. The individual sites like enroute control and control tower each have their own Local Area Network (LAN). This allows local machines to share data with each other before sending it to the machines at other sites, which reduces the outbound traffic and thus the bandwidth needed for timely delivery of information. The sites are interconnected with a Wide Area Network (WAN) to allow the information to go where it is needed. Data shared across this network includes flight plans, weather, surveillance data, Voice over IP (VoIP), and links between the latter and very high frequency (VHF) voice communication. The system is also connected to other ATCS though another WAN. This is important as it allows easy information sharing between different ATCS to facilitate operations. For example, an ATCS in Canada (CAATS) shares its flight plans with an ATCS in the United States (NextGen). This way, when a flight crosses the border and is handed off to an American controller from a Canadian one, the American controller already has the flight plan and the information needed to guarantee proper handling of the inbound aircraft. The shared information includes the flight plans, weather, and surveillance data. The sharing of information also enables an ATCS site to take over air traffic control duty in case another ATCS becomes inoperable.

2.3 Air traffic rules violations and emergency scenarios

In this section, we explain the different air traffic violations scenarios. While our solution is aimed at detecting malicious air traffic data injections, it can be extended to also detect potential emergencies as well as air traffic rules violations.

The first scenario is that of *loss of separation conflicts*. At all times, aircraft must keep a certain distance between them. This distance is defined by a radius around them, as well as a minimum altitude separation both above and under. The resulting volume is a cylinder centered on each aircraft. A loss of separation conflict happens when one aircraft enters another aircraft's cylinder.

The second scenario is *route deviation*. Normally, an aircraft flying under IFR submits a flight plan that, among other things, defines a route between its departure airport and its arrival

airport. A route is a trajectory made of a series of fixed waypoints with a flying altitude tied to them. Air traffic controllers can make changes to an aircraft's route in case of bad weather along the original one, for example. The ATC updates the flight plan and informs the pilot of the changes. A deviation occurs when an aircraft is not following its determined route. This can make controlling other aircraft in the same area more difficult due to the unpredictable trajectory of the deviating aircraft and can even, in extreme cases, cause loss of separation conflicts.

Another scenario is that related to *noise reduction*. Noise reduction is done in two ways: around airports and along routes. Airports have noise reduction rules in order to avoid disturbing people living close. For example, at John Wayne airport in southern California, aircraft are limited in from landing and takeoff to specific time slots. In addition, they must fly specific routes that take them out over the Pacific Ocean before using full-engine power to limit the amount of noise over the city. This airport also has very strict noise limits that vary depending on the day of the week as well as the hour of the day (John Wayne Airport, Orange County, 2013). Those rules include landing and takeoff time slots during which aircraft are prohibited from landing or taking off. The ANSP will also keep aircraft off low-altitude routes that are too close to heavily populated areas during the night (Transport Canada, 2017).

Another scenario is that involving Minimum Equipment Lists (MEL) that aircraft must have on board and that must be functional for aircraft to operate under certain conditions. For example, to fly over large bodies of water, civil aviation authorities require aircraft to have additional equipment such as life rafts and a pyrotechnic signalling device (Federal Aviation Association, 2017). Doing so without the equipment can be dangerous for the passengers of the aircraft and is a violation of air traffic rules that can lead to a fine.

The last rule violation scenario is that of *prohibited airspace intrusion*. A prohibited airspace is a three-dimensional zone where regular air traffic is not allowed for various reasons including the presence of military bases nearby. Currently, military radars and military ATCs contact any plane that is approaching such zones to instruct them to divert course or be intercepted and even attacked.

Fuel emergencies happen whenever a pilot evaluates in flight that the destination or alternate airports cannot be reached due to, for example, stronger winds than forecasted or route changes. The pilot then needs to request priority handling, which can cause disruption to other aircraft.

Another emergency scenario we address is *interrupted track*. During flight, an aircraft's instruments will periodically send information such as SSR replies or ADS-B messages to ground stations in order to keep ATC up to date on their location and situation. When an ATCS receives information from an aircraft, it updates its track. If no information has been received recently, the track is considered interrupted. This can happen because the aircraft has landed and turned off its instruments, but it can also happen because of instrument malfunction, which the pilot may or may not be aware of. This can be problematic because ATC can now only rely on the low-information PSR returns to gather information about aircraft. This makes ATCs lose some awareness of air traffic and can impact their decision making.

The last emergency we cover is *anomalous descent*. During flight, it is normal for pilots to change altitude to follow routes and ATCs instructions. However, there are normal rates of descent and an aircraft that descends too fast or in unusual circumstances indicates that something could be wrong.

2.4 Attack model on ATCS

For the purpose of making the next sections easier to understand, we summarise the attack model we developed by briefly presenting the techniques that can be used. The model is presented in further detail in Chapter 5. We split the attacks on air traffic control systems into four categories: *identification and localisation*, *jamming*, *GPS spoofing* and *packet injection*. Our criteria to differentiate them was the technique used to perpetrate an attack and its effect on the system.

In the first category, *identification and localisation*, an attacker wants to identify an aircraft currently flying to track it. We consider this an attack because it serves as the basis for more dangerous ones. We have identified two different techniques to execute this attack. The first one consists of listening to nearby aircraft's ADS-B broadcasts with a personal ADS-B antenna and the second one is to go on websites that offer unfiltered ADS-B traffic data. The second category is *radio-frequency jamming*. This category is straightforward; using a radio emitter, an attacker jams the frequencies used by air traffic control communications in order to disrupt voice and instrument communications. The third category, *GPS spoofing*, aims to trick an aircraft's GPS receiver into passing erroneous position data to the ADS-B emitter which will then be sent to the ATCS. This attack is realised by spoofing a GPS satellite signal in order to make the aircraft's GPS receiver calculate a wrong position. The last category is *packet injection* and consists of directly sending

spoofed ADS-B packets. In this attack category, a rogue ADS-B emitting station sends spoofed ADS-B packets in order to eventually input false data into an ATCS or into surrounding aircraft's ADS-B In system.

2.5 Traditional IT Security solutions

2.5.1 Intrusion detection systems (IDS)

As the Internet grew in popularity, network security became an issue that needed to be addressed and user activity needed to be monitored to ensure that no one was tampering with the computer systems. The first systems to do this task, named intrusion detection systems, were built in the 1980s and were very basic in their detection method; they used a list of threat signatures to identify and filter traffic. This method evolved to become rule-based intrusion detection; a technique that uses rules based on the vulnerabilities themselves as opposed to a specific exploit. This technique operates on the principle that the vulnerabilities of the system are known and their detection can be expressed in terms of rules to filter out traffic that could exploit them. The other main method of detection is using artificial intelligence (AI) to classify traffic and detect anomalies in the pattern. This method is called statistical analysis. The principle behind the technique is that there is a set of features –quantifiable traffic data and metadata– that make malicious traffic stand out from normal traffic. The features can be defined by a human being or can be computed by giving a set of traffic data to a machine learning algorithm. IDSs come in two modes of operation: host-based (HIDS) or network-based (NIDS), which is independent of the detection technique they use. As described by the name, HIDSs are installed on a computer and analyse the computer's own in and out traffic. NIDSs, on the other hand, are deployed on dedicated machines connected to a network and analyze all traffic passing through it. They can detect denial of service (DoS) attacks as well as port scanners and other potentially malicious activity.

The traditional intrusion detection methods and systems are not useful for this project because the attacks we investigate are not perpetrated at the network traffic level. They are perpetrated before the packets are created as they target the data sent to the sensors on the radio frequency (RF) level. This means that the packets will seem normal to traditional intrusion detection methods. For this reason, we designed a different approach to intrusion detection by reasoning on the data on a higher level using ontologies. Instead of looking at the networking data

in the packet, we look at the air traffic data the packet is carrying and analyse it to see if it makes physical and logical sense with the data previously received.

2.5.2 Antiviruses

Antiviruses are another important part of computer security. Unlike the IDSs, they are used to enhance the security of the computer on which they are installed. They were created as an answer to computer viruses and worms which started to spread in the late 1980s. As viruses evolved to go undetected by antiviruses, new virus detection techniques were invented. The game of cat and mouse has been going on since then. One of these techniques is signature-based detection. This technique consists of scanning a program's code to try to match a sequence of bytes –the signature– with sequences that the antivirus knows to be contained in viruses - the definitions. If there is a match, then the program is flagged as a virus. This technique relies on having up-to-date definitions of viruses. Another technique used is behavioral analysis. Instead of looking directly at the virus, this detection technique uses the virus' behaviour to categorize it. In the same vein as signature-based detection, they compare the behaviour of the program to a database of known malicious virus behaviours. This technique is effective against polymorphic viruses because different copies of this category of viruses do not have the same signature, but behave in the same way.

While antiviruses have proven their effectiveness in other cases, they do not entirely prevent attacks on ATCS. They would only be useful to defend against an attacker who wants to control a machine inside the network to carry out the attacks. An effective antivirus would detect that the computer is compromised and alert the people in charge. However, antiviruses are ineffective if the attack is perpetrated by emitting false data on the ADS-B frequency since the attacks we study do not require infecting a computer on the ATCS's network with a virus or tampering with them.

2.6 Air traffic control security solutions

Researchers proposed many solutions to increase air traffic control system security and reliability. However, most of them do not take into account the vulnerabilities of the ADS technology summarized in the previous section. In this section, we present the most promising and complete solutions in terms of ATCS security.

Multilateration (MLAT) is a hardware-based security solution. It consists of a cluster of sensors that listen for SSR replies or ADS-B broadcasts from an aircraft. It uses the transmission

times of arrival differences to estimate the location of an aircraft. If an aircraft broadcasts the wrong position on the ADS-B link, this technology will be able to detect that the aircraft is not where it says it is and alert the ATCS about the situation. This solution is very costly because it requires the installation of multiple antennae and is not applicable everywhere since there are places, such as in the Great North or over the ocean, where putting up antennae is impractical.

Research on ATCS network vulnerabilities and ways to mitigate potential attacks has been done in *Aviation communication infrastructure security* (Karmarkar, 2012). The author is concerned with traditional cyber security threats such as denial of service and viruses. The author's proposed security measures reflect this focus: defense-in-depth, rate-limiting, access lists, etc. The threats we want to address with our research are not directly related to traditional cyber security because they do not compromise the computers themselves and the solutions proposed in this article do not concern the attacks we considered.

Sampigethaya and his team (Sampigethaya & Poovendran, Visualization & assessment of ADS-B security for green ATM, 2010) propose a novel solution to guarantee ADS-B position report integrity. The technique the authors developed uses groups of interconnected aircraft and an aircraft-based multilateration technique to validate the position data contained in an ADS-B message. Aircraft would share the data they receive with the group they are part of using a different, secure data link. Then, each aircraft estimates the validity of the position report and votes to reject it or not. While this solution addresses many of the packet injection issues, it only concerns aircraft while we focused our research on ATCS.

In *Securing the skies: in requirements we trust* (Nuseibeh, Haley, & Foster, 2009), the authors share their experience with security requirements in the air traffic control domain. As part of their project, they developed a framework to define and analyze security requirements all while taking into account the functional requirements. The example they used was ADS-B position reports. After defining the functional requirements of the system, the authors defined security goals as well as threats in order to define security goals. While the project brought security requirements analysis to the air traffic control domain, their focus was on the design phase of the software life cycle. Given that the aviation world is very slow to change and adapt to new technologies or threats, new threats could emerge after the system goes live. Modifying the design and its implementation would take a very long time and be very costly, on top of the inherent safety hazard a vulnerability poses.

Sink or SWIM: Information Security in the Sky (Jaatun & Fægri, 2013) introduces the System-Wide Information Management (SWIM): a publish-subscribe network that would allow different ATCS to share aeronautical, flight trajectory, aerodrome operations, weather, air traffic flow, capacity, and demand data. The information sources are pilots, airports, airline operations centres, ANSPs, weather providers, and military operations centres. The goal of this infrastructure is to provide a secure and reliable way for different ATCS to share information in order to improve efficiency and air space utilization. The authors talk about network security, but there is no mention in the paper about the validity or authenticity of the information going through the network. The authors also mention specifically that surveillance information will not travel through SWIM. It is worth noting that ATCS currently use SWIM.

Krozel et al. developed a Kalman filter technique to enable aircraft to verify the integrity of an ADS-B broadcast in *Aircraft ADS-B Data Integrity Check* (Krozel, Andrisani, Ayoubi, Hoshizaki, & Schwalm, 2004). This technique was designed to cover transmission errors, loss of signal, and noise filtering. It does so by using the previous data points to infer the tracked aircraft's trajectory and intent. Since it relies on the cooperation of the broadcaster, it could not be adapted to detect most of the previously discussed attacks. This approach could, however, detect a careless attack where the attacker broadcasts random aircraft position, but a more sophisticated attack with a realistic simulated trajectory would fool this solution.

The papers *Methods to provide system-wide security ADS-B backup, validation and security* (Smith, Cassell, Breen, & Hulstrom, 2006) and *Detecting malicious ADS-B broadcasts using wide area multilateration* (Monteiro, Barreto, & Kacem, 2015) both propose a similar solution to validate ADS-B broadcasts. Both solutions consist of using Wide Area Multilateration (WAMLAT) to verify and validate the position reported in the ADS-B broadcasts using time-of-arrival difference between multiple stations that are in the aircraft's range. Thus, this solution requires that multiple operating ground stations be within range. This might not always be the case in the event that a spoofing attack is committed in tandem with a jamming attack on surrounding ground stations, leaving only one station in operation, or in the case where there is only one receiver in range of the aircraft.

ERA Systems Corporation filed a patent for a *Method and apparatus to improve ADS-B security* (United States of America Patent No. US20060119515, 2006), invented by Alexander Smith. The patent describes a method to verify the identity of an aircraft's identification broadcast

using the information sent by the aircraft as well as information coming from another source like primary or secondary surveillance radars. Contrarily to our solution, this method presumes that the old surveillance technologies will still be in use and available in the affected area, which may not be the case.

Kacem et al. introduce an ontological approach in *Security Requirements Analysis of ADS-B Networks* (Kacem, Wijesekeram, Costa, & Barreto, 2014). Their method focuses on classifying attacks based on misuse cases and properties extracted from surveillance data. While the system described in the paper can detect attacks, the authors do not provide a way to mitigate them. It also does not account for attacks that emerge after the system is put in place.

2.7 Data Distribution Service (DDS) protocol

The Data Distribution Service (DDS) protocol is a OSI layer 7 real-time protocol that uses the publish-subscribe design pattern (Pardo-Castellote, 2005). It is used by financial trading applications as well as in air traffic control systems because it offers dependable, high-performance data exchanges. It is data-centric since the data sent through DDS comes with instructions on how the application should interpret it.

Thales, Real-Time Innovations Inc., and Object Interface Systems, Inc. originally developed DDS in collaboration. The DDS specification is under the responsibility of the Object Management Group (OMG). OMG released the first version of DDS in 2004 (Object Management Group, 2004). Since its inception, many vendors have developed their own implementation of DDS. The version used in this research project, DDS 1.4, was released in April 2015 and the implementation we chose is OpenDDS 3.7 (Object Computing, Inc., 2015), released on September 11, 2015.

A DDS application has three major components: data topics, DDS domains, and participants, which are either publishers or subscribers. Data topics are a collection of classes that specify the different ways the shared data is structured. A DDS domain is a collection of such structures and does not overlap with other domains. Participants can be either subscribers or publishers. Subscribers have to subscribe to a particular topic in order to receive updates. They do so by notifying the router that they want to part of the assigned multicast group for a given topic. When it receives packets for a given multicast group, the router forwards them only to the members of the specified group.

There are two ways for participants to discover new ones in OpenDDS. The first one is with a centralized information repository managed by a separate process. The other one is Real-Time Publish-Subscribe (RTPS) discovery, which is an application layer protocol that automatically discovers new services and subscribes to the relevant multicast groups. We used the latter in our project.

Many ATCS use DDS as their data distribution protocol (Object Management Group). The PSR, SSR and ADS-B sensors gather the relevant information and use a DDS publisher to send it over the network. Other publishers are ATC workstations, for flight plan modifications, the flight plan database, and weather information providers. Machines that use a DDS subscriber to receive data from the network are the controller workstations and the flight plan database.

2.8 Ontologies

Ontologies are collections of concepts and the relationships between them. The concept of ontologies comes from philosophy, but it has since been adapted to computing in the 1980s as part of research on artificial intelligence. Since then, there have been efforts to use this concept in other areas of computer science such as the Semantic Web and biomedical informatics. Our project aims to classify data coming from surveillance equipment into an ontology and do various analysis to it in order to detect anomalies. This paradigm is well suited for our anomaly detector because the data and the relationships between different pieces of information are well-defined in our application domain. The domain concepts are thus easy to model in an ontology and it simplifies the analysis.

2.8.1 Ontological solutions

An Extended Ontology for Security Requirements (Massacci, Mylopoulos, Paci, Tun, & Yu, 2011) brings the concept of ontologies to ATCS security. However, they populated their ontology with actors, threats, goals, actions, etc. related to a specific case study threat, i.e. *GPS spoofing*. This approach is not optimal since the ontology needs a large update whenever a new threat emerges. Furthermore, the ontology does not allow for direct attack detection queries as the concepts needed are not represented in it. The solution we investigate is more general and preventative as it does not rely on ontological representations of attacks.

Automated Reasoning for Maritime Anomaly Detection (Roy) details an ontological system that would use data gathered from sensors to generate facts and use them to infer new situational knowledge with the use of automated reasoning. They use a system composed of reasoners, a database containing facts, and a management service to control the flow of information. The architecture of our solution is inspired by this project.

A theoretical basis for our research comes from the work on trajectory modelling made in *A Conceptual View on Trajectories* (Spaccapietra, et al., 2008). This paper is the first to define trajectories from a semantical point of view. This definition takes into account the temporal aspect of a trajectory by including the path, stops, which can be instant, moves, and the start and end of a trajectory. The author also specifically addresses trajectories defined by points made of a space-time pair, which are directly related to our research's domain since it is how ADS-B works.

Baglioni et al. iterated on the previous article in *An Ontology-Based Approach for the Semantic Modelling and Reasoning on Trajectories* (Baglioni, de Macedo, Renso, & Wachowicz, 2008) and provided a methodology to transform trajectory data into an ontology containing geographic data, domain knowledge, as well as the stops and moves that define the trajectory. The article also gives an example of trajectory reasoning using Web Ontology Language (OWL) to detect malicious behaviour among players of a game.

*An Enhanced Spatial Reasoning Ontology for Maritime Anomaly Detectio*n (Vandecasteele & Napoli, 2012) uses the concepts introduced by the two previous articles to detect and characterize anomalies in ship behaviour. Abnormal behaviour detection is done by matching rules defined by experts against an ontology containing sensor data. The article gives ships navigating in restricted zones and trajectory analysis as detection examples. The latter example is very relevant to our research project as trajectory analysis is one of our anomaly detection method.

2.9 Summary

In this chapter, we reviewed the technologies used by ANSPs to control air traffic. We also exposed the weaknesses these technologies bring and how they can be exploited by malicious people. This overview allowed us to have a clear picture of the air traffic control domain. We then explored the most recent solutions proposed in the scientific literature to see if they addressed the attacks identified in our attack model.

The traditional security solutions like IDSs and antiviruses were not adequate in the context of our attack model since the attacks did not require tampering directly with the ATCS, but rather falsifying the data before it enters it. We also looked at solutions specific to our application domain. A lot of the solutions we found rely on radar coverage, which will not always be available, or on installing extra equipment to enable multilateration. Other solutions were not easily adaptable to new threats.

Academic research in this particular application domain is limited. Most research on the vulnerabilities of aviation technologies and the threats they pose do not take into account real-life consequences and how they would impact the way air traffic is directed. This research project is the first to bridge the gap between cybersecurity and the aviation world by evaluating how air traffic controllers and pilots would react to specific attacks and how they would operate under such circumstances.

CHAPTER 3 ANOMALY DETECTION FOR ATCS

In this chapter, we will describe an approach and an architecture for constructing an anomaly detection system for ATCS that will allow us to both detect malicious manipulation of data (described briefly in Section 2.4 and in more detail in Section 5.1) and also some of the potentially dangerous ATC situations described in Section 2.2.2. The detector gathers data from data sources already on the ATCS and uses it to populate an ontological database. It detects anomalies in the data by running queries against the database. If anomalies are detected, it dispatches alerts and information about the detection to the relevant network actors.

3.1 System architecture

3.1.1 Key assumptions on target ATCS

Before we talk about the design our solution, we need to gather all the information we have about real-life ATCS. Doing so gives us guidance on basic, technical decisions about the solution. The first such decision is to decide where to put the detection engine, i.e. on every computer in the ATCS do to detection for the computer only or on its own separate machine doing detection for the entire ATCS. The second question is how the detection system gets the data needed to function. This question becomes important in the event that the detection engine runs on its own machine since it will need to interact with the network in order to receive the data. This is not a problem if it runs on every machine currently in the ATCS since they already receive the data relevant to them and have it available locally. The third question we need to answer is how the detection information, i.e. the results of the queries, is dispatched to the concerned network actors. Similarly to the second question, this one is also important if the detection system runs on a separate machine because it needs to communicate the results to the other machines. This question becomes a non-issue if the system runs on the existing machines since it will only do the detection relevant to its host and can communicate directly without going through the network. In order to answer these questions, we need to make basic assumptions about what a typical ATCS architecture looks like. We described this hypothetical architecture in section 2.2.2.

In short, we presume that ATCS are made of regular computer hardware and operating systems that run specialized software. The machines communicate through an IP network isolated from the Internet. The key element of this network is the data exchange protocol, DDS, that is used

to transmit data in real time. Typical networking protocols such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Hypertext Transfer Protocol (HTTP), and File Transfer Protocol (FTP) cannot be used alone for two reasons. First, the nature of the data transfers on the network is not point-to-point, which makes most of the previously mentioned protocols inapplicable by themselves. Second, the real-time and Quality of Service (QoS) requirements are very strict since the timely delivery of the information can affect real-life decisions taken by ATC and could compromise air traffic safety. There are many higher OSI layer protocols that satisfy those requirements and can be used on top of TCP or UDP, but we found evidence that CAATS (Real-Time Innovations, 2013) uses DDS as its communication protocol. It is not far-fetched to presume that other ATCS use the same protocol. We chose DDS as an example, but this is without loss of generality due to the design of our solution since it is easily adaptable to different communication protocols.

3.1.2 Detector localization

We know that ATCS currently have mechanisms for detecting potentially dangerous situations, in particular *loss of separation* situations. Whenever two planes are too close to each other, the ATCS notifies the controllers by sending a visual alert on their computer screens. This computation is fairly simple to do since it only requires checking if a point is inside a cylinder, as we described in Section 2.2.2. This example of known-to-exist anomaly detector in ATCS is a good starting point for us to discuss various architecture choices for the more general detector we propose. The two obvious options are the following.

1. **Decentralized detection:** In this case, the detection is done directly where the information is needed, e.g. controller's workstation and local servers at an area control centre. This is akin to how Host-based Intrusion Detection systems (HIDS) work in traditional IT security. The advantages of having a detection system running on a controller's workstation is that it can do computations only on the area that concerns the controller. This design makes sense for the *loss of separation* scenario since the aircraft need to be in the same area to be close to each other. Moreover, the controllers of the other areas do not need to see the alerts that do not concern them. The same result could be achieved by a centralized design, but the time to process the data needed for the *loss of separation* scenario for every aircraft in every area covered by the ATCS could

take too long and the response time would in turn be too long to prevent this scenario. Fortunately, in this specific example, aircraft have their own prevention system. Another advantage is that this design is lighter on the ATCS network's bandwidth. No new machines are added to the network and the existing network components already receive most data needed for detection. Moreover, the alerts stay on the same machine, which guarantees timely delivery and also does not take any network bandwidth.

2. **Centralized detection:** The centralized design operates similarly to a Network-based Intrusion Detection System (NIDS), in the sense that the detection is done on a completely separate machine connected to the network. The alerts are then sent as DDS topics to the concerned network participants. An advantage of this design is that the detection system is that it does not need to run on an already-existing ATCS component. This greatly increases portability and enables system updates without having to update an entire component. Furthermore, since the detection system is encapsulated and separate from existing machines, only one version needs to be created instead of having custom versions for all the different types of controller workstations, servers, etc. Another advantage of this design is that all the data needed for detection is gathered at the same machine. Having one machine also simplifies managing and maintaining the detection system. The disadvantage of a centralized design is performance. Since we are introducing a new machine on the network, it will need to receive all of the relevant data, increasing the amount of bandwidth needed. The detection system also sends alerts to the concerned components, which takes even more bandwidth. Timely delivery of alerts is crucial for air traffic safety and sufficient bandwidth is necessary to achieve this goal.

In the end, we chose the centralized detection system design for our solution. This allows us to design a solution that can be adapted to different environments without much modifications to the solution or to the ATCS itself. It also addresses more network architecture issues, which is important since there are no two ATCS alike. We evaluate that we only need to add a few data topics in the system as well as a slight modification to the subscriber of the controller's workstation to enable it to subscribe to the new topics. It is also more general, which is why we feel it is better

for proof of concept. Performance engineering problems could be addressed by other means such as filtering, compression, networking configurations, etc. and are out of scope for this project.

3.2 Architecture

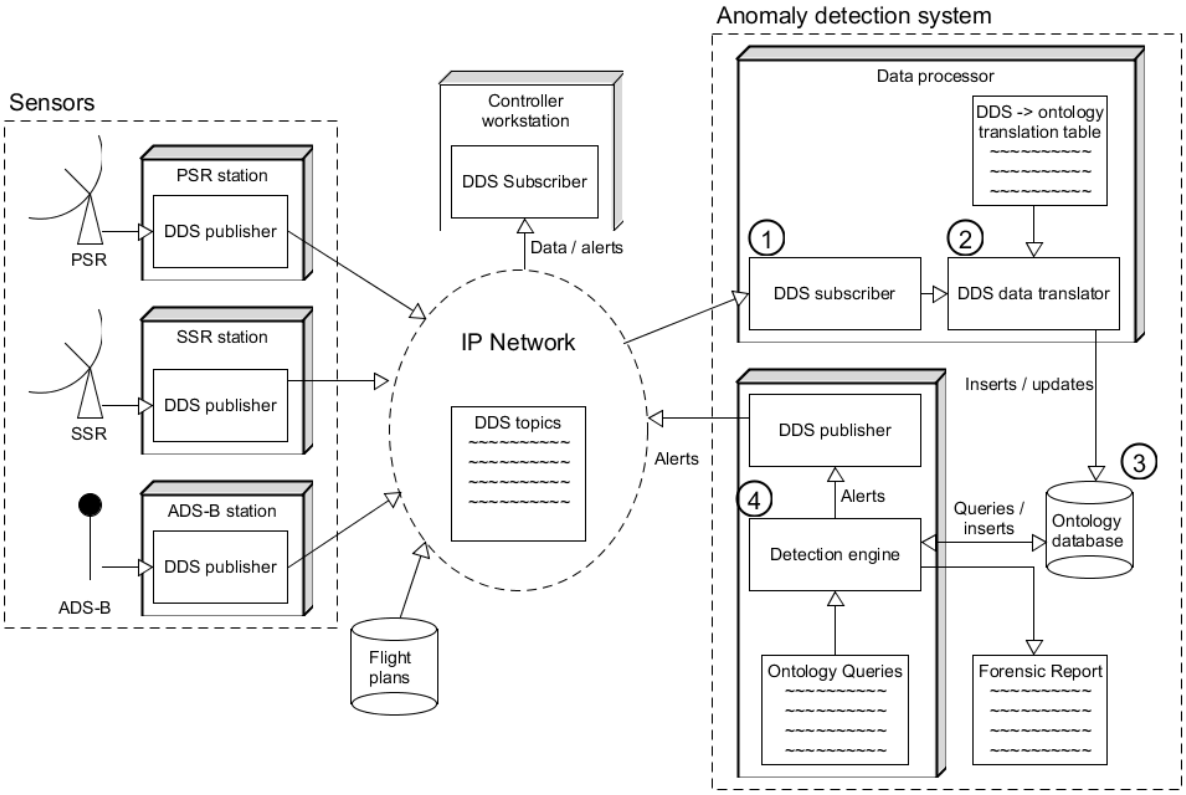

Figure 3.1 : Detection system architecture

The centralized detection architecture we propose is depicted in Figure 3.1. Our solution is entirely encapsulated and separated from the rest of the system. It consists of the following main components:

1. **DDS data collector:** This module subscribes to relevant DDS topics being published by the various data sources on the ATCS. It is the equivalent of an IDS sensor on a traditional IT network. Unlike those, however, no special network configuration adjustments must be made to accommodate them. The machine running the data collector must simply subscribe to the relevant DDS topics using the built-in DDS subscribe requests. It provides

as output a stream of DDS messages of the relevant topics in a format that can be easily parsed and interpreted by downstream modules.

2. **DDS data translator:** This module populates the ontological database based on the DDS topic messages that the data collector has received. This is in principle a simple script that parses the DDS message and uses the API of the ontological database to generate and launch procedures that insert or update instances of concepts in the database. Parsing is done with the help of the *DDS topics to ontology concepts translation table* that defines the equivalences between data points in DDS topics and concept attributes in the ontology, i.e. where to put each data piece. This table is described in more details in section 5.3.

3. **Ontological Database:** The ontological database is used to store the ATCS data as concepts and as relations between concepts, i.e. the semantics of the data. It receives insert and update commands from the DDS data translator. The structure in which the data is stored is called an RDF Triplestore (Ontotext, n.d.) and is made of a *subject*, an *object*, and a *predicate*, linking the two. For example, our ontology database can have an *aircraft* subject, a *track* object, and a *has a* predicate. This semantic structure represents the fact that an aircraft has a track. The process by which this model was created is described in more detail in section 3.3.

4. **Detection engine:** The detection engine runs a SPARQL Protocol and RDF Query Language (SPARQL) engine that runs various detection queries. It polls the database to gather the data necessary for the query that is running. If the query finds an anomaly, the engine creates an alert and sends it to the DDS publisher running on the same machine. The publisher then creates an appropriate instance of the *Alert* DDS topic and publishes it on the network for the concerned subscribers to receive. The DDS topics are described in more details in section 4.7. The detection queries run at different frequencies depending on what kind of anomalies they are designed to find and on how frequent the data they require is updated.

3.3 ATC ontology design process

The model for the ontological database was designed in collaboration with another Master's student, Simon Malenfant, who is specialised in ontologies. We expect this consultation process between domain experts and ontological experts to be representative of what would happen during the implementation of our solution with an actual ANSP.

The first step was to explain the inner workings of ATC to the ontological expert. Since the ontological expert was not familiar with the air traffic control domain, we had to do an overview of ATC to detail the context for the ontology. We started with the background workings of a regular flight, from the departure gate to the arrival gate. We also talked about air traffic regulations to give the ontological expert a better idea of the emergencies and rules violations we want the ontology to cover. The overview is similar in content to what is described in Chapter 2.

Once the expert was familiar with the inner workings and procedures of air traffic control, we talked about the technology behind ATC, namely the ATCS and its components. The goal of this step is to identify the data circulating on the network in order to familiarize the ontology expert with the data that is available for inclusion in the model. We started with an overview of the main sensors that are part of the ATCS, namely PSR, SSR and ADS-B antennae. We also detailed the data present in flight plans as it provides important information for the emergency scenarios detection such as fuel level. Detailing the pieces of data also helped to determine the DDS topics for our ATCS simulator which are detailed in section 4.7. This step is important because the data pieces available to be inserted into the ontological database is the basis for our model.

In the third step, we gave meaning to the data by adding semantical relationships. We started by linking individual data pieces together when it made sense, either because they were circulating together on the ATCS network such as the positioning data coming from radars, or because they were related and part of a bigger data structure like flight plans. These larger data structures are called concepts and are the building blocks of the ontology.

Finally, we added semantical relationships between the concepts to form the ontological model. Adding relationships between them links every piece of information together in one coherent model. Having such a model allows us to store the data in a way that allows us to easily query it to find anomalies. The final model is presented in Figure 5.2.

3.4 Anomaly detection system

The anomaly detection system gathers data by subscribing to the relevant DDS topics identified in section 4.7. Once the *subscriber* receives data updates, it forwards them to the *DDS data translator* to be inserted into the ontological database. The *DDS data translator* uses the *DDS to ontology translation table* to make equivalencies between DDS topics attributes and ontological concepts attributes for easy data conversion. The details of the table are presented in section 5.3. This step also filters out irrelevant data, i.e. data that does not have an equivalent attribute in the ontological model. Once the ontological mapping has been done, the *translator* inserts the converted data into the ontological database.

Using a connection to the ontological database, the *detection engine* queries the database to gather data needed for the detection queries it is currently running. The detection queries are written in SPARQL and run on a regular basis triggered by a separate polling process that keeps track of the frequencies of the queries. The frequencies will vary between queries as some of them run very frequently because they need to act on every new piece of data that comes into the system and others are only run on demand or every few hours for forensic purposes. For example, queries that use radar data to detect loss of separation run very frequently, but queries that are used for the forensic mode run every few hours or even on demand. The duality of query frequencies introduces design challenges to the system. On the one hand, some queries need to run at a very high frequency, i.e. as soon as there is new data available – which is every second or faster. On the other, some need large amounts of data to produce forensic reports. Thus, the system needs to be very efficient for quick responses while also being capable of storing large amounts of data. Since these decisions are technical in nature, we chose to keep them out of the scope of this thesis and we do not address them.

The results of the real-time queries are alerts that need to be dispatched to the concerned machines. To do so, we added an *Alert* DDS topic to the network's existing topics. When the *detection engine* produces an alert, it forwards it to the *DDS subscriber* sitting on the same machine as the engine. The *DDS subscriber* then converts the alert to the correct DDS topic structure and publishes it on the network. To receive alerts, the computers such as the controller's workstation need to subscribe to the new *Alert* topic. Alert processing if left to the concerned machines once they receive an alert through their respective *DDS subscriber*. When the *detection engine* runs in forensic mode, it produces a report that contains the results of the queries.

3.5 Implementation

Due to time constraints, we chose not to build a working prototype of the detection engine. Instead, we concentrated our efforts on building the ATCS simulator and constructing the ontological model.

Having a working ATCS simulator was important because it gives us good data and a working testbed for future research, something much IDS research lack. We tried contacting ANSP and ATCS vendors to form a partnership with them to facilitate our research, but they all declined. ATCS design and implementation are closely guarded secrets for two reasons. First, ATCS are very expensive and there is a lot of competition between vendors and they want to keep their ATCS design as an industrial secret. Second, there are security issues associated with doing research on a real ATCS that can lead to real-life safety issues if technical details about ATCS were to be leaked. Building the simulator was a critical step in our laboratory's research plan. Lots of time and effort was put into its design and prototyping, which left us with no time to focus on the detector.

One of the challenges of designing the ontology is to make it specific enough that it can detect what we need it for, but also generic enough so that it is easily expandable for new detection capabilities. This problem comes up every time an ontology is to be designed and is not exclusive to our research.

Building the above components to have a working detection system would not present a research challenge and would constitute a lot of simple work for a small incremental gain, which is why we left this step out of scope. We decided to focus our efforts on what we evaluated was the hard and interesting parts, the design of the solution, the ontology and its queries, and the ATCS simulator.

CHAPTER 4 AIR TRAFFIC CONTROL SYSTEM SIMULATOR (ATCSS)

In this chapter, we describe the proposed architecture of our air traffic control system simulator (ATCSS). We first explain the goals we set to achieve. Then, we focus on the most basic parts: the radar screen, the simulation engine, the sensors and the communication protocol used by the system. We then explain how we put together those building blocks to emulate an air traffic control system and the DDS traffic passing through it. We finish this chapter by detailing the DDS topics we chose.

4.1 Goals

The main goal behind the design of the air traffic control system simulator is to create an environment that allows us to simulate attacks on an ATCS and to visualise and measure their effects. For our experiments to be valid, the system needs to be as close as possible in functionality and architecture to an actual ATCS and thus needs to include its critical components as well as its communication protocols. The components we identified as critical to the functioning of an ATCS are the sensors, namely PSR, SSR and ADS-B antenna, and the radar screen. We chose to use DDS for our simulator as it is the communication protocol used by ACTSs in Canada and Europe (Object Management Group). The second goal of the design was to be able to create custom air traffic scenarios that suit the needs of our experiments and to be able to repeat those experiments. Finally, we want to be able to visualise the air traffic that is being simulated in order to ensure the adherence of our air traffic scenarios to real traffic patterns and to validate the hypotheses of our attack model about the impacts of the attacks.

4.2 Radar screen

The first step in this project was to check for existing software we could use as part of the simulator. We directed our search towards virtual flight simulation communities like VATSIM and IVAO. After testing different software used the communities' virtual air traffic controllers, we chose EuroScope (Csernak, n.d.), an air traffic controller console application that includes an air traffic simulation engine. It also includes server software to allow multiple instances of EuroScope and multiple flight simulators to take part in the same virtual flight environment. We contacted the developer, Gergely Csernak, to ask permission to use his software as part of our research project,

which he granted. Once we installed the software, we were able to capture packets in order to understand the Flight Simulator Data (FSD) communication protocol used by the software. Mr. Csernak gave us a specification document (ProtoDev Development Group, 2002) that explained the different packet types used by the protocol. It helped understand the packet sequence of a communication between a simulated aircraft, EuroScope, and the server. EuroScope allows the user to visualize the flight data of locally simulated aircraft or of flight simulators connected to the same server, as shown in Figure 4.1. It also displays the flight plans filed by the aircraft's pilot, as well as sector information such as runways, sector boundaries, and waypoints.

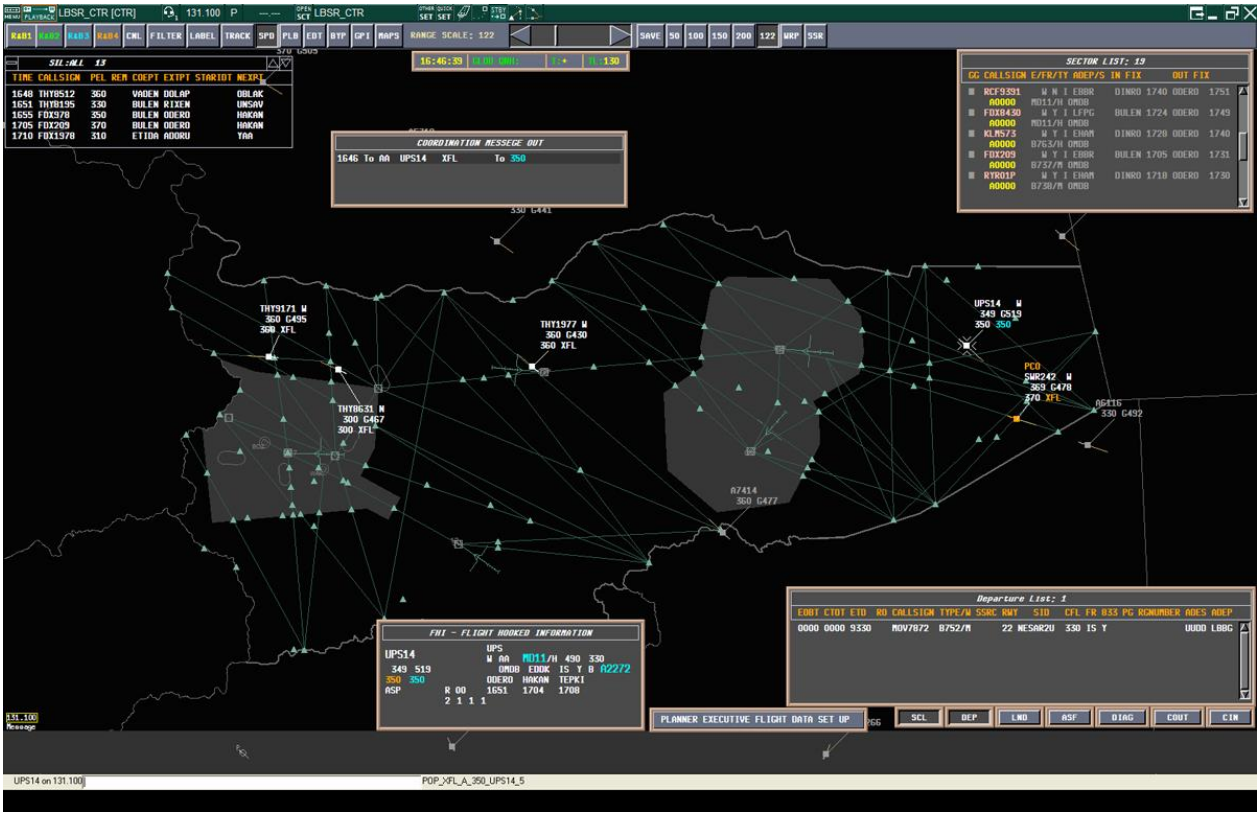

Figure 4.1 : EuroScope

Since this software is designed to use the FSD protocol for data sharing, we had to create a translator that takes the incoming DDS packets and convert them to FSD packets. We chose to use Python (Python Software Foundation, 2017) because it is very flexible and enables quick development of software prototypes. This translator connects to the DDS subscriber on the same machine as the radar screen to receive the DDS packets. Then, using the data contained in the packet, it will create an FSD packet and send it to the radar screen to be displayed.

4.3 Simulation engine

As mentioned in the previous section, EuroScope has its own built-in simulation engine. It moves aircraft around according to air traffic rules and allows an air traffic controller to give instructions to the simulated aircraft such as takeoff, landing, route change, etc. The simulation engine is a separate process that sends the simulation data on the loopback network on a specific port. Then, Euroscope sends the simulation information to other instances of Euroscope that are connected to it. The information about air traffic comes from a scenario file that details existing aircraft and their routes. Those files can be manually created, but there are tools to generate them with proper syntax and coherence (Phillips, 2011).

4.4 Sensors

The next components of the air traffic control system simulator are the sensors. These are abstractions of PSRs, SSRs and ADS-B antennae. Their role is to receive data from the simulation engine and translate it into properly formed DDS packets that are then published on the DDS network. The topic under which they publish the data and the type of data depends on which sensor they are emulating. These are explained in more details in section 4.6. Separating the sensors by category adds realism to the system's simulation and enables us to add different levels of trust to the data depending on its source. For example, data coming from PSRs is very reliable while data coming from ADS-B could have been spoofed. Therefore, we prioritise data coming from a trustworthy source when it is available.

4.5 Flight Simulation Data protocol

The FSD protocol is an OSI model layer 7 protocol that uses TCP connections. The data segment of the packets has three parts: a single character indicating the packet style, a pair of characters indicating the packet command, and the rest is the packet's actual content, which comprises multiple data fields each separated by a colon. Table 4.1 shows the most important commands and packet styles. The FSD protocol is the standard used by multiplayer plugins written for the most popular flight simulators – X-Plane (Laminar Research, 2015), FlightGear (FlightGear Flight Simulator, 2016), Microsoft Flight Simulator (Microsoft, 2006), and Prepar3D (Lockheed Martin, 2015) – to connect to servers belonging to the virtual communities VATSIM and IVAO. Table 4.1 shows the different types of packets part of the FSD specification.

Table 4.1 : FSD packets description

Prefix	Style Description	Command	Command Description
$	Administrative packet	FP	Send a flight plan to the server.
		HO	Request a hand-off to a different air traffic controller.
		HA	Response to the HO request.
		CQ	Request client data (e.g. flight plan, pilot details, current server, etc.).
		CR	Response to the CQ request.
#	Communication packet	AA	Add an air traffic controller to the server.
		DA	Remove an air traffic controller from the server.
		AP	Add a pilot to the server.
		DP	Remove a pilot from the server.
		TM	Send text message.
		SB	Request the aircraft model to display in the flight simulator.
@	Update the aircraft data.		
%	Update the air traffic controller data.		

4.6 System architecture

Figure 4.2 shows the architecture of the system. It features two networks: a control network, in red, which acts to provide the data source – the “real world” - for the simulation network, in blue, which is our equivalent of a real ATCS.

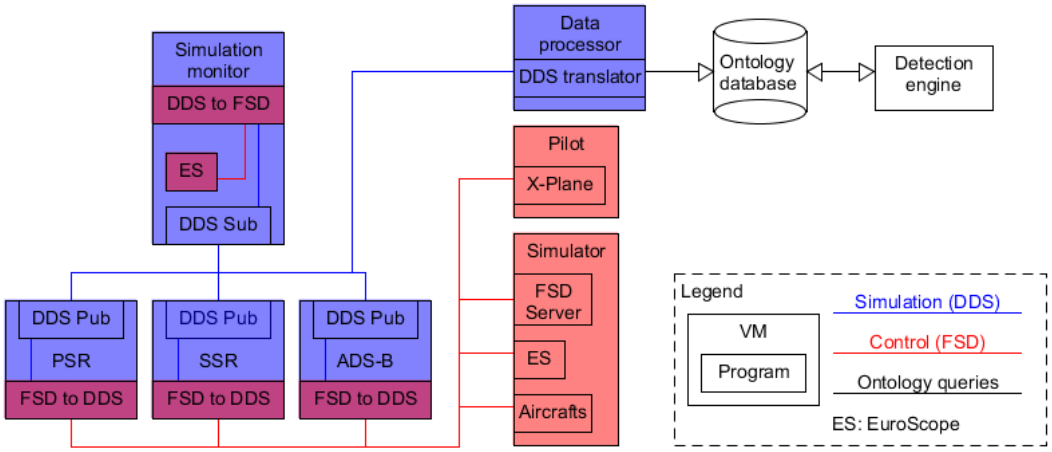

Figure 4.2 : Air traffic management system simulator architecture

4.6.1 Simulation network (DDS)

The components of the blue network include a simulation monitor, the equivalent of an ATC's workstation, as well as multiple surveillance radars and ADS-B antennae.

The Simulation monitor virtual machine (VM) consists of the radar display software, EuroScope, a DDS subscriber and a Python script. EuroScope's purpose is to visualize the aircraft in the simulation and to issue directives. The Python script sends the packets containing the directives and the administrative data to the server. The DDS subscriber subscribes to the relevant topics - aircraft positions and flight plans – and transfers the data to EuroScope.

The PSR, SSR, and ADS-B VMs, are the equivalent of the surveillance equipment. They get the aircraft's positioning data from a Python script that connects directly to the server using the FSD protocol. The script relays the data to the C++ DDS publishers who publishes it under the appropriate topic.

The next component is the DDS capture module. This module captures all DDS traffic during a simulation session and sends it to a translator to extract the relevant data. The translator then sends the extracted data to another script which populates the ontology with the new data. The SPARQL queries are then run against this ontology by the Detection engine to verify that there are no anomalies in the data, i.e. to detect potential attacks, rule violations, and emergencies.

4.6.2 Control network (FSD)

The modules of the control network are the simulation server, a VM dedicated to capturing DDS packets with replay functionality, and a VM with a flight simulator installed on it.

The simulation server is where all the information about the aircraft positioning, the flight plans and the air traffic scenario information. The EuroScope instance runs the simulation and manages the simulated aircraft. It is also a reference since the effects of the various attacks will not reach this instance. The simulated aircraft send their positioning data directly to the FSD server, which will in turn send it to the subscribed Python scripts using the FSD protocol.

The last piece is the Pilot VM. This optional component runs a flight simulator that can connect directly to the FSD server and integrate the player's positioning data in the simulation. This feature allows us to test the interaction of an air traffic controller with a real pilot in one of our attack scenarios.

4.7 DDS topics

DDS topics are data structure definitions used by publishers and subscribers to send and interpret that data circulating on the DDS network. We have identified three DDS topics that are used by the sensors. The first one is the primary surveillance radar data broadcast. It contains the data gathered by a primary surveillance radar, i.e. the azimuth and distance of a radar return, as well as a timestamp, and the radar's identification number in order to translate the data points into latitude and longitude. The second topic is the secondary surveillance radar data broadcast. Similarly to the primary surveillance radar's broadcast, it contains a timestamp and the identification number of the radar, but it also contains the SQUAWK, altitude, and identification number of the communicating aircraft. The third topic in our DDS definitions is the ADS-B report. It is made up of an aircraft's reported coordinates, altitude, callsign, ground speed, heading, airspeed, and a timestamp. The other DDS topics are used for pilot-controller interactions. The first of those topics is the flight plan, which contains the information listed in section 2.2.1.4. Our topics also include an alert topic used by the anomaly detector to send messages to ATCs about anomalies and to highlight the aircraft concerned by the anomaly. The last topics, open flight plan, close flight plan, and route change are sent by the ATC to the FSS to interact with flight plans and perform the function described by their names.

CHAPTER 5 THREAT MODEL AND ONTOLOGICAL SOLUTION

In this chapter, we present our theoretical contributions. The first such contribution is our attack model, which we summarized in section 2.4. It is presented here in much more details. The next contributions are the ontology, which is our method of storing the data to be analysed, and the SPARQL reasoner, which runs the analysis queries.

5.1 Attack model

In this section, we detail the attack model we developed in order to evaluate the best solution. The vulnerabilities at the core of this model are based on multiple research articles (McCallie, Butts, & Mills, 2011) (Purton, Abbas, & Alam, 2010) (Martinovic & Strohmeier, 2013) (Costin & Francillon, 2012) (Haines, 2012) (Wood, 2009) (Sampigethaya, Assessment and mitigation of cyber exploits in future aircraft surveillance, 2010) (Tippenhauer, Pöpper, Rasmussen, & Capkun, 2011) (Strohmeier, Lenders, & Martinovic, On the Security of the Automatic Dependent Surveillance-Broadcast Protocol, 2015) (Sampigethaya, Privacy of future air traffic management broadcasts, 2009) (Magazu III, 2012) (Lim, 2014), as well as various other types of articles (Kelly, 2016) (Marks, 2011) (Zetter, 2012) (Walker, 2012) (Greenberg, 2012) (Henn, 2012) (Cenciotti, 2014) (Le Monde, 2015) (Miller, 2015) (Mark, 2015) (Pasztor, 2015) (Laboda, 2015) (Thurber, 2012) (BBC News, 2012). We organized our attack model in sections that each represents a different attack category. For each category, we explain the different scenarios, consequences, and possible solutions. Schäfer et al. from Oxford University (Schäfer, Lenders, & Martinovic, 2013) developed a similar attack model in parallel and verified it by surveying industry specialists, which adds validity to our findings. We split the attack scenarios into five categories: identification and localization, RF jamming, GPS spoofing, packet injection and replay, and DDS intrusion. Due to technological and time constraints, the solution we developed only addresses the packet injection and replay attack scenarios.

5.1.1 Identification and localization

A core feature of ADS-B is the broadcast of position and flight information by the aircraft. Currently, the information transmitted is not encrypted, which means that anyone can listen to the broadcasts and get the position and flight information about all aircraft surrounding an ADS-B antenna. While flight tracking websites make use of this feature of ADS-B coupled with official

information shared by ANSPs to offer their services to the public, a malicious person can use the data broadcast by an aircraft to enable different attacks.

An example of a malicious activity enabled by this feature is stalking whether done by paparazzi or by someone with worse intent. The identification information of an aircraft, such as the callsign or tail number, is publicly available on public aircraft registration databases on the Internet (Transport Canada, 2014). A malicious person can use this information to track the targeted aircraft by listening to its ADS-B broadcasts with a homemade antenna and receiver, even if the owner of the aircraft has requested aircraft tracking websites to filter out data about the aircraft. If access to the equipment is too costly, there are websites that provide unfiltered aircraft tracking features where ADS-B tracking enthusiasts provide the information (ADS-B Exchange, n.d.). Examples of aircraft that do not appear on commercial websites but appear on amateur communities' websites are FBI surveillance aircraft, shown in Figure 5.1, U-2 reconnaissance planes (Aircraft Spots, 2017), and even Air Force One (RTL-SDR.COM, 2015).

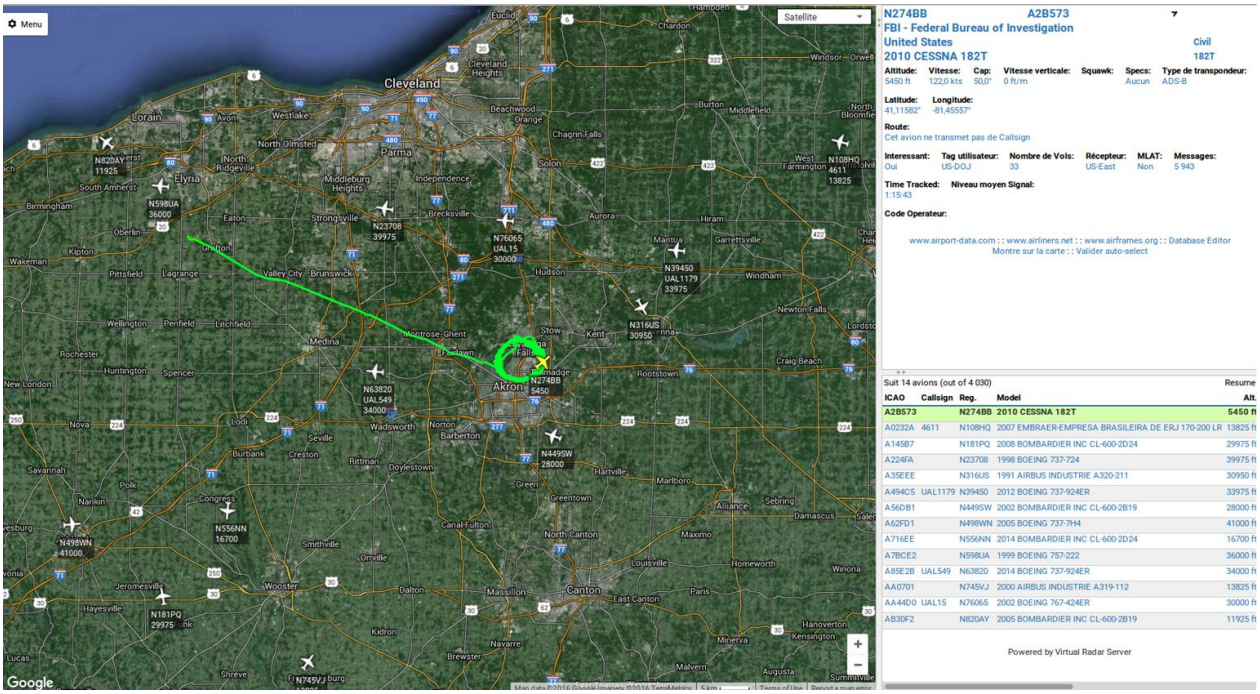

Figure 5.1 : FBI plane spotted

The ability to track aircraft accurately by listening to ADS-B broadcasts facilitates aircraft targeting. For example, a terrorist organization wanting to blow up an aircraft can use this feature to know exactly where the aircraft is and evaluate when it is in a vulnerable position such as when approaching or landing, where it will be nearer to the ground and outside of the boundaries of the

airport, to attack it with an explosive device. The consequences of this attack range from loss of life to loss of money from the reduced confidence of the public in air travel depending on the attack this vulnerability made possible.

The solution to these attacks is to encrypt ADS-B communications. Doing so would not only prevent unwanted listeners from gaining information about aircraft currently flying, but would also authenticate the communications, which would prevent spoofing. However, as we show later in this chapter, encryption does not prevent every attack on ADS-B.

5.1.2 Jamming

ATCS rely on radio frequency communications, such as ADS-B, to operate, which makes them vulnerable to jamming. The impact depends on the target of the jamming attack. We have identified three possible attack scenarios that take advantage of this vulnerability: jamming a piece of ground equipment, jamming an aircraft, and area-wide jamming.

The most basic form of a jamming attack is to emit a powerful signal in the direction of a piece of ground equipment in order to prevent it from receiving ADS-B packets or SSR replies from surrounding aircraft. ATCs will most likely consider the piece of equipment malfunctioning until they investigate. If there are other, unaffected ADS-B receivers or SSRs in the range covered by the targeted equipment, the consequences will be minimal. If there are no unaffected receivers or radars in the target's range, there will be a loss of coverage, which might in turn lower the ATC’s confidence in ADS-B or SSR as a surveillance technology.

A different version of the jamming attack is to target an aircraft. The goal in this scenario is to prevent an aircraft from receiving radio communications from ground equipment or from other aircraft. Similarly to the previous scenario, a malicious person uses a software-defined radio (SDR) to emit a powerful radio signal in the direction of the target, which the attacker acquires by listening to its ADS-B broadcasts. The consequences of this attack vary on the frequency that is being jammed. If the attacker targets the frequency used for voice communications, the pilot and controller will switch to an available one. If the attacker targets the 1090 MHz frequency, the victim's transponder will not be able to hear the interrogations of the SSR, which means it will not send an answer. It also means that the ADS-B In equipment, if present on the aircraft, will not be able to receive ADS-B updates from ground equipment or hear collision avoidance system (CAS)

signals from surrounding aircraft. The pilot of the targeted aircraft will treat the affected equipment as malfunctioning and will apply the directives related to equipment malfunction.

The final scenario of this type of attack is area-wide jamming. Due to the scope of this attack, we presume that it is carried out by an organization and not an individual. A malicious organization wants to cause chaos around a busy air traffic area or around an airport by shutting down all communications. To do so, they choose to jam the 1090 MHz frequency used by ADS and SSR as well as the ones used by VHF over an area. One potential way to do this is with a swarm of SDR-equipped drones. The first step is to scatter them over the targeted area. When everything is in place and the weather conditions are favourable for chaos, e.g. low visibility, the terrorist organization activates the drones. They will then soar out of small arms reach and start broadcasting a powerful radio signal across the targeted radio frequencies. This attack will most likely trigger an emergency security control of air traffic (ESCAT) procedure, as well as the ATC Zero directive. The consequences for this catastrophic scenario include loss of human lives due to possible collisions as well as monetary losses in both loss of material, flight cancellations, and loss of customer's confidence.

Increasing the density of ADS-B receivers can decrease the loss of coverage in case of an attack on a ground antenna, which reduces the impact of such an attack, but does not fully prevent or solve one.

5.1.3 GPS spoofing

Since ADS-B Out systems use a GPS receiver to determine the aircraft's coordinates, the system is vulnerable to GPS spoofing. In this scenario, the attacker uses GPS spoofing equipment to trick the aircraft's GPS receiver into thinking the fake signal is coming from an actual GPS satellite. The ADS-B equipment will then start broadcasting the position the GPS unit calculated using the satellite signals it received, including the spoofed one. Since the signal is spoofed, the calculated position is not valid. Encryption will not fix this vulnerability because the ADS-B data is compromised before it is encrypted. Multilateration would help in the event that there are multiple, operational ADS-B receivers in the targeted aircraft's range but it is costly and not always viable such as in remote places like the Great Canadian North and over the oceans.

5.1.4 Packet injection

The attacks in this category take advantage of ADS-B's lack of authentication and encryption to trick the ATCS as well as aircraft's ADS-B In equipment by sending fake ADS-B broadcasts using an SDR. The scenarios associated with this attack category are ghost aircraft, radar screen clutter, broadcast spoofing, and packet replay.

In the ghost aircraft scenario, a malicious person wants to create confusion by tricking ground equipment or aircraft ADS-B receivers into displaying on the ADS-B feed an aircraft that is not there. The perpetrator of the attack uses an SDR to broadcast ADS-B packets with fake data, which ground equipment as well as nearby aircraft will pick up. Those systems then update their display with the fake data. The consequences will vary depending on the situational awareness of the victims of this attack. A very aware pilot might be able to tell that the aircraft is fake and the only consequence will be a loss in confidence in ADS-B from the pilot and ATC. If the pilot is not well aware, he or she might ask the ATC for a course change in order to avoid the fake aircraft. This might result in higher fuel consumption and loss of confidence in ADS-B if the ATC finds out that the aircraft is fake. If the ghost aircraft appears near an airport, it can result in monetary losses due to delays or flight cancellations. By carefully selecting the ghost aircraft's identification data, the attacker can instead create political tensions. For example, a malicious person in China can broadcast ADS-B packets containing the identification information of an American U-2 reconnaissance plane, which is available on aircraft registration websites, and place it somewhere above one of China's military facilities. The ground equipment picks up the packets and updates the radar screens accordingly, which might cause confusion among Chinese air traffic controllers, who might report the incident. Until they find the cause of the ghost aircraft, if they find it at all, there will most likely be suspicion towards the United States of America. Other consequences include money wasted on the investigation and loss of confidence in ADS-B if they find the cause.

The radar screen clutter attack is similar to the ghost aircraft attack except that the goal is to introduce many fake aircraft in order to clutter the ATC’s radar screens to prevent controllers from distinguishing real aircraft from fake ones. The technique is also similar to the previous scenario, except that the attacker will send ADS-B packets at a higher rate and containing different aircraft information and positioning data. The aircraft's ADS-B In systems, if present, that are affected by the attack display a multitude of aircraft, which means that the pilot will not be able to trust the ADS-B In feed and will most likely ignore it. The consequence will be a loss of confidence

in the ADS-B technology. On the air traffic control's side, the controllers will most likely ignore the ADS-B feed and will instead rely on radar surveillance. The consequence will be a loss of confidence in ADS-B, as well as a loss of situational awareness from the lack of ADS-B data. If the area affected has no radar coverage, such as the north of Canada, ATCs will most likely have to operate under lower air travel volume capacity in that area for the duration of the attack since there are no other means of surveillance. This causes monetary losses associated to delays, cancellations, route changes, and loss of confidence in air travel safety from the public, as well as loss of confidence in ADS-B from ATC.

The broadcast spoofing attack iterates on the ghost aircraft scenario by using the identification data of an aircraft that is currently flying in the same area as the malicious person. The attacker can learn which aircraft are flying nearby by listening to their ADS-B broadcasts, which also provides the aircraft's identification information that the attackers need in order to carry out this attack. The perpetrator will begin by targeting a nearby aircraft and gather its identification information. Then, this information is inserted in the spoofed ADS-B packets in order to trick the ATCS and other aircraft's ADS-B In equipment into thinking those packets are sent by the aircraft. Since the goal is to create confusion, the attacker inserts fake positioning data in the packets before broadcasting them. This causes the ADS-B equipment to update the displays with the spoofed position and possibly alternate between the spoofed position and the real one when it receives a new ADS-B packet from the aircraft or from the rogue emitter. However, as in the previous scenario, consequences could include monetary losses due to route changes, cancellations, delays, and loss of confidence in the event that there is no radar coverage.

The final scenario in the packet injection attack category we consider is packet replay. In this scenario, similarly to the other packet injection attacks, a malicious person wants to trick the ATCS into thinking an aircraft that is currently in the air is not where it really is. The attacker listens for an aircraft's ADS-B broadcasts and records them in order to rebroadcast them later. Since there is currently no timestamp on ADS-B packets, the ATCS's ground stations and nearby aircraft's ADS-B In receivers will not be able to tell that someone is rebroadcasting old packets. As usual, the consequences on the ATC's side are minimal in the event that there is primary or secondary surveillance radar coverage. If not, the consequence is a loss of situational awareness, which could lead to worse outcomes as stated in previous scenarios. In any case, the ADS-B

information on the surrounding aircraft's displays will not be accurate and will cause a loss of situational awareness on the pilot's side.

Encrypting ADS-B communications would solve these issues to an extent, but it would be useless in the event that someone steals the private key associated with the aircraft's registration information. Another scenario to consider is a pilot who purposely or accidentally enters the wrong private key or identification information in the aircraft's ADS-B system. The positioning data will be accurate, which means that multilateration will not filter it out, but the identification data will be wrong and could be a duplicate of an aircraft already flying in the area, which would confuse ATC. Encryption would effectively prevent the packet replay attack because the ADS receiver knows when the aircraft generated the packet and can ignore it if the timestamp is too far in the past. However, all of these solutions would require a change in the ADS-B specifications as the current one does not have room for more data on an ADS-B packet, or an easy way to implement an encryption scheme.

5.1.5 DDS intrusion

This attack category is different in execution from the others. Indeed, in the *DDS intrusion* case, the attacker relies on compromising a machine that is part of the ATCS local network, instead of emitting data over the air. Traditional IT security solutions may help here by making it harder to compromise a machine, but they are ineffective once the machine is compromised, as we discussed in section 2.5. The attack scenarios here are similar to the previous categories in goals and effects, so we will not repeat them here. What we do talk about here is the execution. This attack technique relies on compromising a machine that is running a DDS subscriber. This can be done by directly taking control of that machine and manually modifying its behaviour or by infecting it with a virus that autonomously publishes false data. Obviously, an antivirus is of great help in this scenario as it would prevent the virus infection in the first place. Since this scenario is outside of the realm of ADS-B and other radar technologies, techniques like multilateration and ADS-B encryption would not be effective at preventing this type of attack. This technique is much harder to execute because the machines part of the ATCS are isolated from the outside world and are difficult to physically access. This type of attack presents a different set of challenges to security experts as it concerns both traditional IT security and malicious data manipulations. However, this can also be seen as an advantage since there are complementary solutions in the form of antiviruses to prevent it and our proposed anomaly detection engine to mitigate the effects.

5.2 Ontological model

The ontology is the basis of our solution. It is used to store data coming from the sensors in a way that expresses what each data point represents and establishes relations to add context and meaning. This step is crucial for the reasoner because it translates data into concepts. The ontology was developed jointly with another Master's student, Simon Malenfant, who works in the same research laboratory and who is specialized in ontology design.

The first and most basic concept in our ontology is the *Position*, which is made of three data points that represent latitude, longitude and altitude. The next concepts augment the *Position* in different ways to express more complex data relations. The *Position* can be either a *PSR position*, *SSR position*, or an *ADS-B position*, depending on which instrument sent the data. These subclasses have different level of trust, according to our attack model.

The first complex relation we address is how we represent an aircraft and its trajectory. The *Measured position*, which is made of a *Position* and a timestamp, is used as a point in space and time. A *Track* is a series of *Measured position* and represents the trajectory of an *Aircraft*. The *Aircraft* class also contains a *Zone of separation* and a list of *Equipment.* A *Zone of separation* represents the cylinder centered on an aircraft in flight that represents the minimum distance or altitude aircraft need to keep between them. It is made up of a radius and an altitude. An *Equipment* represents a piece of special equipment on board of the aircraft like life rafts and signaling devices that are mandatory for certain types of flight.

The next complex concept is an aircraft's route, which is needed as a key part of the representation of a *Flight plan* in the ontology. For this concept, we introduce the *Named position*, which includes a *Position*, as well as a name to express the waypoints aircraft use during IFR. A *Route* is a list of those positions and represents the announced waypoints the pilot will fly through. A *Flight plan* also contains data points to express fuel level, cruise speed, departure time, arrival time, and flight plan status which is either *opened* or *closed*. Other key elements of the *Flight plan* are the departure and destination airports, which we conceptualize with the *Airport* class. A *Prohibited zone* represents prohibited airspace that aircraft cannot enter without special authorizations. It is categorized by a *Position* as well as a radius and an altitude. In this case, the altitude component represents the maximum altitude of the prohibited zone.

The *Airport* class contains a *Named position* property and a *Regulation* property used to conceptualize the periods where take-offs and landings are restricted to reduce noise pollution around an airport. The *Regulation* property has a start time and an end time that represent the time slot where landings and take-offs are allowed.

Figure 5.2 depicts how those concepts are related to each other.

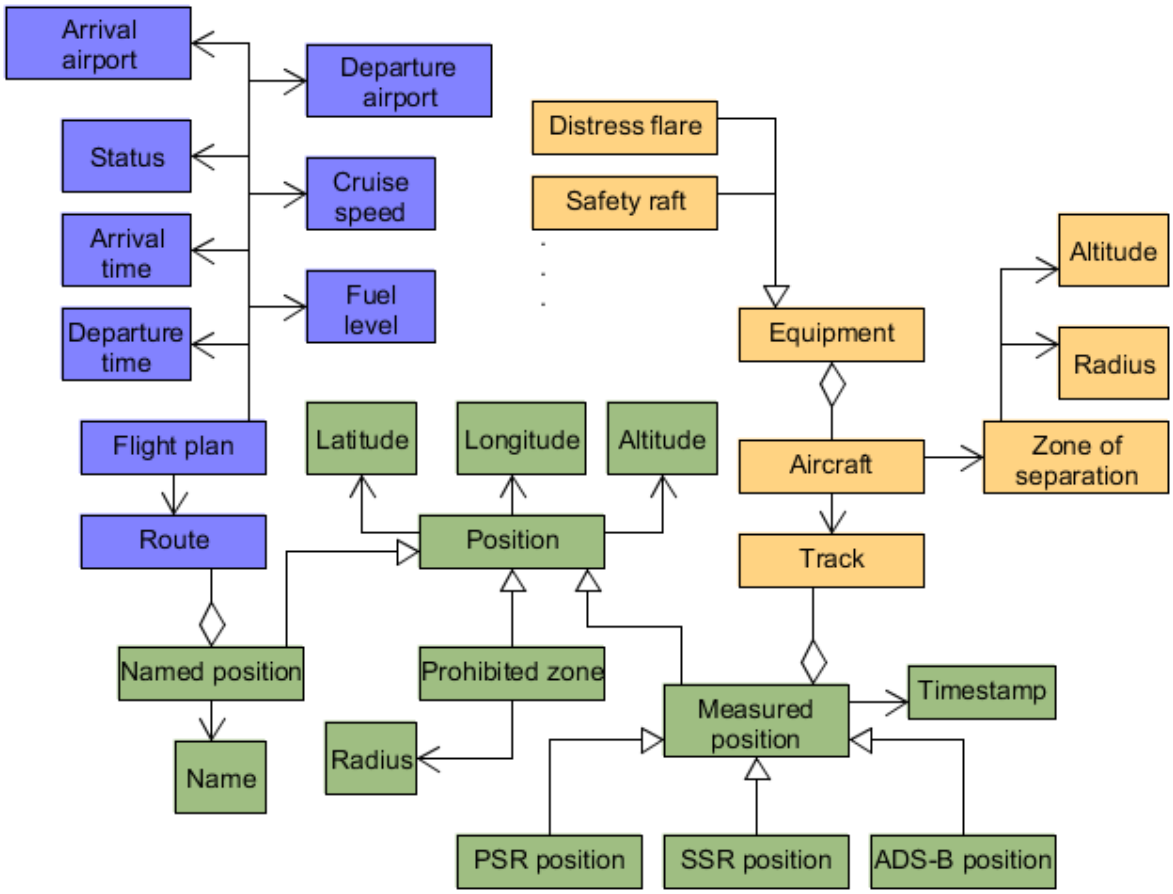

Figure 5.2 : Ontological model

5.3 DDS topics to Ontology subjects data translation table

The equivalence table contains a list of all the data topics used by the ATCS, a list of all ontology subjects and instructions on how to create the ontology subjects with the data from the DDS topics. This is another part of our design that allows for great portability as it allows easy adaptation to any data topic definitions, which can vary from one ATCS to another. The only data

topics that have to be added in the existing system are the topics for the alerts. It tells the DDS adapter where to find each data point needed by the ontology. The relationships between the DDS topics defined in Chapter 4 and the ontological concepts described in section 5.2 are described in the following table. This table becomes the specification for constructing the DDS data translator module described in section 3.2. Table 5.1 shows an example of what a DDS translation table could look like.

Table 5.1: DDS topics to ontological concepts translation table

DDS topics		Ontological concepts
PSR data broadcast	Azimuth	PSR position.Azimuth Track.Measured_position.Azimuth
	Distance	PSR position.Distance Track.Measured_position.Distance
	Timestamp	PSR position.Timestamp Track.Measured_position.Timestamp
	Radar ID	PSR position.Radar ID
SSR data broadcast	SQUAWK	SSR position.SQUAWK Track.Measured_position.SQUAWK
	Altitude	SSR position.Altitude Track.Measured_position.Altitude
	Aircraft ID	SSR position.Aircraft ID
	Timestamp	SSR position.Timestamp
	Radar ID	SSR position.Radar ID
ADS-B report	Latitude	ADS-B position.Latitude Track.Measured_position.Latitude
	Longitude	ADS-B position.Longitude Track.Measured_position.Longitude
	Altitude	ADS-B position.Altitude Track.Measured_position.Altitude
	Callsign	ADS-B position.Callsign Track.Measured_position.Callsign
	Ground speed	ADS-B position.Ground_speed Track.Measured_position.Ground_speed

Table 5.1: DDS topics to ontological concepts translation table

DDS topics		Ontological concepts
	Heading	ADS-B position.Heading Track.Measured_position.Heading
	Airspeed	ADS-B position.Air_speed Track.Measured_position.Air_speed
	Timestamp	ADS-B position.Timestamp Track.Measured_position.Timestamp
	Radar ID	ADS-B position.Radar ID
Flight plan	Equipment	Aircraft.Equipment
	Route	Flight_plan.Route
	Departure airport	Flight_plan.Departure_airport
	Arrival airport	Flight_plan.Arrival_airport
	Cruise speed	Flight_plan.Cruise_speed
	Cruise altitude	Flight_plan.Cruise_altitude
	Flight rules	Flight_plan.Flight_rules
Alert		Alert

5.4 Detection logic

The reasoner is the cornerstone of this solution. This module uses a set of queries to analyze the data stored in the ontology to detect dangerous situations such as loss of separation and anomalies in the data such as impossible aircraft movements. Our reasoner uses the SPARQL Protocol and RDF Query Language (SPARQL) (W3C, n.d.). These rules aim to detect data anomalies that could be symptoms of an attack or emergency situations. Attack symptoms are data anomalies that would indicate that an attacker is trying to compromise data integrity and accuracy. We have included some SPARQL pseudo-code queries as examples throughout this section. The rest of them can be found in the appendices.

5.4.1 Detecting fake aircraft

5.4.1.1 Aircraft materialization

The first abnormal situation we address is *aircraft materialization*. This event occurs when a new aircraft track starts at an unexpected location. Expected locations include the areas near airports and near sector limits since these are the places where an aircraft would normally come into the ATC's radar screen. The query does so by looking at the start location of a new track when it is registered in the system. If the coordinates are not near an expected location, then an alert is shown to the ATC. By marking and segregating suspicious aircraft, ATCs can decide to hide them in the event of a screen clutter attack. We evaluate that this query only needs to run each time a new track is added into the system.

5.4.1.2 Violation of physical laws

This query is used to detect whenever an aircraft appears to have impossible behaviour. Examples of impossible behaviour are aircraft that are moving too fast, turning too tightly or simply appear to teleport on the radar screen. Other examples are sudden altitude changes and aircraft appearing out of nowhere on the ADS-B feed but that were never seen by the primary or secondary surveillance radars. This situation could be caused by track reconstruction errors, but it may also be the result of an attack on the ATCS. Figure 5.3 shows the SPARQL query to detect violation of physical laws.

```
01 TIME_RANGE = maximum time interval between data points to compare them
02 MAX_SPEED = maximum allowed speed
03 SELECT ?aircraftID ?timestamp1 ?sensor
04 { ?ddsSubject1 rdf:type :ADSBReport
05   ?ddsSubject2 rdf:type :ADSBReport }
06 UNION
07 { ?ddsSubject1 rdf:type :SSRPosition
08   ?ddsSubject2 rdf:type :SSRPosition }
09 ?ddsSubject1 :ID ?aircraftID
10 ?ddsSubject2 :ID ?aircraftID
11 FILTER(?ddsSubject1 != ?ddsSubject2)
12 ?ddsSubject1 :timestamp ?timestamp1
13 ?ddsSubject2 :timestamp ?timestamp2
14 FILTER(|?timestamp1 - ?timestamp2| < TIME_RANGE)
15 ?ddsSubject1 :longitude ?longitude1
16 ?ddsSubject1 :latitude ?latitude1
17 ?ddsSubject2 :longitude ?longitude2
18 ?ddsSubject2 :latitude ?latitude2
19 ?distance = getDistance(?longitude1, ?latitude1, ?longitude2,
               ?latitude2)
```

Figure 5.3 : Violation of physical laws

```
20 FILTER(?distance/|?timestamp1 - ?timestamp2| > MAX_SPEED)
21 ?ddsSubject1 :publishedBy ?sensor
```

Figure 5.3 : Violation of physical laws

For every aircraft in the monitored area, the reasoner compares the history of reported positions (lines 4 to 11) up to a certain time in the past (lines 12 to 14) and verifies if the longitude, latitude, airspeed, and ground speed are consistent with each other. It also verifies that the ground speed is lower than the allowed maximum (lines 15 to 20). If any of those conditions are not met, the suspicious aircraft is highlighted on the radar screen to let ATCs know that its movement is not normal. In order to be effective, this query needs to run every time a new position update is added to the ontological database.

5.4.2 Detecting emergencies and air traffic rules violations

5.4.2.1 Loss of separation

While our solution is aimed at detecting malicious air traffic data injections, it can be extended to also detect air traffic rules violations. One of these is loss of separation. During flight, aircraft have to keep a certain distance between them, either in latitude/longitude or in altitude. This zone of separation can be seen as a hockey puck centered on an aircraft. A loss of separation incident happens when an aircraft enters another aircraft's hockey puck.

```
01 TIME_RANGE = Maximum time difference between two Measured positions
02 ALTITUDE_MIN = Minimum altitude difference between aircraft
03 DIST_MIN - Minimum distance between aircraft
04 SELECT ?aircraft1ID ?aircraft2ID ?timestamp1
05 ?ddsSubject1 rdf:type :ADSBReportType1
06 ?ddsSubject2 rdf:type :ADSBReportType1
07 FILTER(?ddsSubject1 != ?ddsSubject2)
08 ?ddsSubject1 :timestamp ?timestamp1
09 ?ddsSubject2 :timestamp ?timestamp2
10 FILTER(|?timestamp1 - ?timestamp2| < TIME_RANGE)
11 ?ddsSubject1 :altitude ?altitude1
12 ?ddsSubject2 :altitude ?altitude2
13 FILTER(|?altitude1 - ?altitude2| < ALTITUDE_MIN)
14 ?ddsSubject1 :longitude ?longitude1
15 ?ddsSubject1 :latitude ?latitude1
16 ?ddsSubject2 :longitude ?longitude2
17 ?ddsSubject2 :latitude ?latitude2
18 ?distance = getDistance(?longitude1, ?latitude1, ?longitude2,
               ?latitude2)
19 FILTER(?distance < DIST_MIN)
20 ?ddsSubject1 :publishedBy ?sensor
21 ?ddsSubject2 :publishedBy ?sensor
```

Figure 5.4 : Loss of separation

As shown in Figure 5.4, the query looks at all pairs of aircraft (lines 4 to 7) within a given time range (lines 8 to 10). The query will then verify if the aircraft are too close vertically (lines 11 to 13), and horizontally and vertically (lines 14 to 18). Aircraft that meet these conditions are highlighted on the ATC's screen to alert the controller that the aircraft are too close together. Since the goal of this query is to be an early warning to avoid two aircraft getting in loss of separation distance, this query does not need to run every time the database receives a new position update. We evaluate that the frequency at which this query runs for each aircraft depends on the density of the air traffic. When there are multiple aircraft in a given area, this query will run more often and vice versa.

5.4.2.2 Route deviation

Another traffic rule violation our solution can detect is *route deviation*. Before taking off, pilots have to file a flight plan to tell ATCs of their intentions. The flight plan contains a route that consists of a series of waypoints that the aircraft plans to go over. Unless instructed otherwise by an air traffic controller, the pilot has to fly along the routes between the waypoints. A route deviation incident happens when an aircraft strays too far from the routes between the waypoints. The query computes the time it should take to reach the aircraft's next waypoint from the previous one and takes into consideration the aircraft's ground speed to compensate for wind speeds. The query raises an alert if the aircraft takes longer than that time to reach the next waypoint, within a certain threshold. Every time an aircraft reaches a waypoint, the system computes how long it should take for the aircraft to reach the next waypoint plus a grace period to allow for slight delays. It then schedules this query to run after the time has expired.

5.4.2.3 Prohibited zones

Our solution also helps prevent *prohibited airspace intrusion*. A prohibited airspace is a well-defined three-dimensional cylinder centered on a given point. By looking at the distance between an aircraft and the center of a prohibited airspace and comparing it to the radius of the zone as well as the difference between the altitude of the aircraft and the ceiling of the zone, the query can detect aircraft that are too close to the prohibited airspace. An alert is then shown to ATCs so they can contact and warn the aircraft to prevent it from going too close to the prohibited zone and have military ATCs contact the pilot of the aircraft. Similarly to the *loss of separation*

query, this query also serves as an early warning. Thus, it only needs to run every so often for each aircraft.

5.4.2.4 Noise reduction

Our ontological solution can help detect *noise reduction rules violations* by looking at the time an aircraft requests a take-off or landing permission and at allowed take-off and landing periods. It can also verify the route that a given aircraft will take and make sure it does not break any noise reduction rules along the route. The query looks at all the take-off and landing requests as well as the routes in the ontology and compares them to the effective noise reduction rules. Since this is an *audit mode* query, it does not need to be scheduled to run regularly and instead runs on demand.

5.4.2.5 Fuel emergency

Our solution has the added benefits of being able to predict and help resolve emergencies like *fuel emergencies*. With the centralized data store our solution offers, the reasoner can keep track of the amount of fuel left in an aircraft's tank by submitting a query to the ontology. The query allows the reasoner to know when the aircraft took off and how much fuel the pilot indicated in the flight plan. It can then alert the ATCs when an aircraft is projected to run out of fuel before reaching its destination airport. Similarly to the *route deviation* query, this query can be scheduled ahead of time. Since we know how much fuel a given aircraft has in flight time from this original flight plan and from ADS-B updates, we can schedule the query to run every so often to check on the fuel level of the aircraft and adjust the frequency of the query accordingly. If the fuel levels of a given aircraft are getting low, the query can run more often for that aircraft in order to warn ATC as quickly as possible if the fuel levels get too low and the aircraft needs special attention.

5.4.2.6 Interrupted track

In this scenario, an aircraft suddenly stopped transmitting data. This could happen for a number of reasons: loss of radar or ADS-B coverage, malfunctioning equipment on the aircraft or a crash. Figure 5.5 shows how the query would detect such an event.

```
01 TIMEOUT = Maximum delay since last update before considering track
             interrupted
02 NOW = The current time
03 SELECT ?aircraftID ?sensor
```

Figure 5.5 : Interrupted track

```
04 {?ddsSubject rdf:type :ADSBReport}
05 UNION
06 {?ddsSubject rdf:type :SSRPosition}
07 ?ddsSubject :ID ?aircraftID
08 ?ddsSubject :timestamp ?timestamp
09 FILTER( [NOW] - ?timestamp > TIMEOUT)
10 ?ddsSubject :publishedBy ?sensor
```

Figure 5.5 : Interrupted track

The query looks at the latest update of every aircraft (lines 3 to 6) and checks if an aircraft has not updated its position for a certain amount of time (lines 7 to 9). The aircraft that fulfill this condition are identified as such and the system raises the appropriate alert. This query also runs on a scheduled basis. Every time a new update comes in, the system calculates a delay in which a new update is expected. If the end of the delay is reached without any new update, this query triggers the alert.

5.4.2.7 Anomalous descent

An *anomalous descent* occurs when an aircraft has a descent rate that is too high to be a normal descent and would indicate a problem with the aircraft. The rate of descent is the difference of altitude between two position updates divided by the timespan between those updates. This query looks at every aircraft's last position update and another one a certain amount of time in the past and computes the rate of descent using the altitudes reported on those two updates. If the rate of descent is too high, the module can alert the ATCs. Since this query calculates differences between updates to detect an emergency situation, it needs to run every time a new update comes in.

5.4.3 Detecting high-level attacks

The alerts triggered by the previous can be analysed on a macro level to detect the nature of the attack. We identified four types of attacks from our attack model that can be deduced using the metadata from the anomalies detected by our ontological reasoner as symptoms of those attacks. These scenarios help decide whether the alerts raised by the previous detection rules are anomalies due to errors in the system or actual attacks. Since the following query are meta-analysis, they run every time a new alert is triggered.

5.4.3.1 Ghost aircraft

The first and most basic high-level attack type we look at is the *ghost aircraft*. In this scenario, an attacker wants to make a fake aircraft appear on the controllers' screen. Depending on the execution and sophistication of the attack, it can be detected in different ways. The first sign a ghost aircraft is being injected into the system is the presence of aircraft materialization alerts, which would be triggered by a careless attacker who makes an aircraft appear in the middle of nowhere instead of near an airport or sector boundaries, for example. The second sign is a route deviation alert. This can occur when an attacker is spoofing the information of an existing aircraft with modified coordinates. The last and surest sign is that the ADS-B broadcasts of an aircraft do not match any aircraft detected by the primary surveillance radar. However, this detection technique only works in area with primary surveillance radar coverage.

5.4.3.2 Screen clutter attack

The goal of a *screen clutter attack* is to fill the ATC radar screens with fake aircraft in order to make real aircraft hard to discern by hiding them under clutter. It is similar to the ghost aircraft attack in technique, but with a different goal. By looking at how many aircraft materialization, laws of physics violation and ghost aircraft alerts, the reasoner can deduce that those anomalies are related and injected into the system with a purpose, namely to clutter the screens of ATCs.

5.4.3.3 Jamming

An attacker may choose to jam the frequencies used by aircraft equipment to disrupt communications. While it cannot be directly solved due to the nature of attack, our reasoner can detect whenever such an attack happens. By routinely compiling average air traffic density at any hour of the day, we can compare those averages to the current air traffic density. A much lower density value than usual would potentially indicate a jamming attack. Another detection technique our reasoner uses is to analyse how many interrupted track events it detected in the recent past. Too many events close to each other geographically and temporally indicate that the area may be victim of a jamming attack.

5.4.3.4 System tampering

The last detection scenario is *system tampering*. It is important to detect such events because this attack technique is at the core of the problems introduced by ADS-B and also serves as a check

on the system's integrity. By looking at the alerts that concern ADS-B data, namely aircraft materialization, violation of laws of physics, loss of separation, route deviation, prohibited zones, interrupted track, anomalous descent, ghost aircraft, screen clutter, and jamming as well as how often they have been raised in the recent past, the query can evaluate if the system is compromised and if the data displayed on radar screens can be trusted.

5.5 Discussion and lessons learned

The application domain was fairly easy to break down into classes for the ontology. We think the biggest factor was that it is a concrete application domain with clearly defined real-world objects and relations as opposed to a more abstract application domain like IP network traffic. However, we could not represent everything we originally wanted to. For example, we are not able to easily represent and implement the notion of a *body of water* in our ontology, which prevented us from creating some rules for the audit mode of the detection system. Prohibited zones were approximated to circles which may not be representative of the actual area. We evaluated that this approximation was good enough as the alert raised by the detection system is an early warning when an aircraft is approaching a prohibited zone.

We also found that adding new detection rules to the detection engine was hard without going back to the ontology to add new classes or properties. However, once the most basic classes like *Position* and *Aircraft* were added, it was easy to expand them to fulfill our needs. For example, we needed to differentiate the positions by the surveillance equipment that report them, so we created the *PSR position*, *SSR position*, and *ADS-B position* subclasses.

CHAPTER 6 CONCLUSION AND RECOMMENDATIONS

This Master's thesis makes three main contributions: 1) an architecture for a rule-based anomaly detection system for Air Traffic Control Systems (ATCS) employing an ontology of high-level concepts in Air Traffic Control (ATC) and aviation that detects data manipulation attacks and potentially dangerous situations, 2) an architecture and implementation of a high-fidelity air traffic control system simulator (ATCSS) that allows us to measure the real-time impact of cyber attacks, and 3) an attack model detailing the goals and techniques that could be used by modern attackers to attack ATCS. In order to construct ontological model of ATCS and attacks on them, we introduced a threat model of realistic attacks against modern ATCS. As far as we know, this is one of the first threat models that encompasses both physical and cyber threats to ATC and ATCS and partially answers our first research question.

After defining the attack model, we designed and built an ATCSS to reproduce the effects of those attacks and test eventual solutions. This simulator leverages free software used in aviation gaming communities to reproduce the physical simulation components of an ATCSS, which allows us to measure and demonstrate the real-life impact of cyber attacks, something that we do not believe has been done before. This achievement was one of most time-consuming and technically challenging tasks of this research and we believe adequately addresses our third research question. Nonetheless, the ATCSS we built could certainly be improved upon. The most important features missing from our air traffic simulator is the integration of weather and flight plan information, and more realistic pilots, whether in the way of "smart pilots" - artificial intelligence that would simulate the behaviour - or real pilots interacting with the system through a flight simulator. Adding weather would improve the realism of our air traffic by adding wind patterns and turbulence that the smart pilots would interact with by requesting flight level changes or route deviations to the ATC. Weather would also include storms, fog, heavy snowfalls and other weather phenomena that cause route deviations and airport delays or closures. Despite the fact that it is able to simulate Automatic Dependent Surveillance-Broadcast (ADS-B) data sources, our simulation engine does not allow the aircraft to broadcast its heading or airspeed, which would have helped make the simulation more realistic, but would also have provided extra data points for the detection engine to analyze. Also, the simulated aircraft does not keep track of the amount of fuel on board and will thus never have unforeseen fuel emergencies due to unexpected winds or route changes. Other limitations in our architecture and data are pilot errors like missed approach or missed landing,

Primary Surveillance Radar (PSR) false-positives such as birds or mountains, aircraft Traffic alert and Collision Avoidance System (TCAS) alerts, areas without or with limited radar or ADS-B coverage, prohibited zones, and bodies of water. The next step would be to address the limitations of the simulator to have a more realistic air traffic simulation. Another direction would be to include ADS-B data taken from flight tracking websites. This option would provide the most realistic test data because it is taken directly from real air traffic. Finally, it would be desirable to use the flight simulator plugins and the radar screens to test different attack scenarios with real pilots and air traffic controllers. This would allow measurement of their reaction to attacks and to evaluate different remediation techniques.

The next step in our research was to establish which Data Distribution Service (DDS) topics would be realistically part of a typical ATCS. Using the available data in those topics, we mapped them to the high-level concepts in an ontology that we designed, such as positions and tracks. Then, we used these concepts to design SPARQL Protocol and RDF Query Language (SPARQL) queries that address the attacks pointed out in the first step and other non-malicious potentially dangerous scenarios. In this process, we were able to address our fourth research question by providing important answers about the difficulty of this process and discover some of the techniques and rules of thumbs to keep this process on track. This part of the research was done in collaboration with another Master's student, Simon Malenfant, and these results on the ontological design process are mainly his contribution.

The anomaly detection solution we proposed in 0 takes advantage of ontologies to find anomalies in the data coming from the different sensors. We opted for a Network-based Intrusion Detection System (NIDS)-like architecture where our detection engine runs on dedicated machines connected to the ATCS network. This decision was somewhat confirmed to be easier to adapt to different ATCS than the Host-based Intrusion Detection System (HIDS) design since modifying our controller workstation (EuroScope) was fairly complicated. The advantages this design has over other design and over other solutions is portability and extensibility. Since the detection system is independent from the air traffic control system, it is easily adaptable to different ATCS. Furthermore, the data sources across ATCS are mostly the same and only the interface between the ATCS network and the ontology would need to be modified. Our design also allows air traffic management organizations to extend the detection to include other attacks and dangerous scenario than the ones presented in our model since detecting a new attack or scenario would be a matter of

writing a new SPARQL query, which does not require a thorough knowledge of the detection system.

While addressing some of the most pressing issues concerning ATCS vulnerabilities to data manipulation attacks, our proposed detection system is not complete. A proof-of-concept prototype of the anomaly detection solution described in 0 was not fully implemented during the course of this work, and this is an important limitation of our work; our second research question remains only answered in theory. We consciously chose to concentrate our research efforts on the construction of an adequate simulator architecture (described in Chapter 4) and the ontological modelling process (Chapter 5). On the one hand, the construction of an adequate simulator was key to allow further research by the research group, but was also a necessary step to gain better domain knowledge on ATC in order to meaningfully participate in the ontological modelling process. On the other hand, the research team knew that the main difficulty and open research questions lay in the process of creating such an ontology for a concrete purpose, in this case anomaly detection. Thus, integrating the ontological model and queries that we constructed into a fully functional prototype to be tested in realistic simulations on the ATCSS that we built would have been the logical next step. Unfortunately, we simply ran out of time.

BIBLIOGRAPHY

ADS-B Exchange. (n.d.). *Global Radar View*. Retrieved July 14, 2016, from ADS-B Exchange: https://www.adsbexchange.com/

Aircraft Spots. (2017, March 8). *Aircraft Spots.* Récupéré sur Twitter: https://twitter.com/aircraftspots/status/839624252810641408

Baglioni, M., de Macedo, J. A., Renso, J., & Wachowicz, M. (2008). An Ontology-Based Approach for the Semantic Modelling and Reasoning on Trajectories. In *Advances in Conceptual Modeling - Challenges and Opportunities* (pp. 344-353). Springer Berlin Heidelberg. doi:10.1007/978-3-540-87991-6_41

BBC News. (2012, June 29). *Researchers use spoofing to 'hack' into a flying drone.* Retrieved July 14 2016, from BBC News: http://www.bbc.com/news/technology-18643134

Cenciotti, D. (2014, August 13). *U.S. airborne communication plane could be tracked on the Web for 9 hours during air strike that killed Taliban leaders in Afghanistan.* Retrieved May 4, 2016, from The Avionist: https://theaviationist.com/2014/08/13/bacn-supports-air-strike-afghanistan/

Costin, A., & Francillon, A. (2012). *Ghost in the Air (Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices.* Retrieved June 15, 2016, from https://media.blackhat.com/bh-us-12/Briefings/Costin/BH_US_12_Costin_Ghosts_In_Air_WP.pdf

Csernak, G. (n.d.). *EuroScope*. Retrieved from EuroScope: http://www.euroscope.hu/main.php

DenQuixote. (2014, October 10). *Wiresharking VATSIM and IVAO*. Retrieved June 29, 2016, from SimFed - The Simulator Federation Project: https://simfed.org/blog/2014/10/10/wiresharking-vatsim-and-ivao/

Dunstone, G. (2012). *ADS-B Introduction.* Retrieved July 5, 2016, from ICAO: http://www.icao.int/APAC/Meetings/2012_SEA_BOB_ADSB_WG8/SP01_AUS%20-%20ADS-B%20Basics.pdf

Federal Aviation Administration. (2016, November 28). *Measuring the Performance of Airports*. Retrieved January 5, 2017, from Federal Aviation Administration: https://www.faa.gov/nextgen/snapshots/airport/

Federal Aviation Administration. (2017, January 4). *A Brief History of the FAA*. Retrieved January 5, 2017, from Federal Aviation Administration: https://www.faa.gov/about/history/brief_history/

Federal Aviation Agency. (2014). *NextGen Works for the Gulf of Mexico.*

Federal Aviation Agency. (n.d.). *Next Generation Air Transportation System (NextGen)*. Retrieved March 22, 2016, from https://www.faa.gov/nextgen/

Federal Aviation Agency. (n.d.). *NextGen - NextGen Programs*. Retrieved from https://www.faa.gov/nextgen/programs/

Federal Aviation Association. (2017, Februaru 28). *Electronic Code of Federal Regulations.* Retrieved March 1, 2017, from U.S. Government Publishing Office: http://www.ecfr.gov/cgi-bin/text-idx?c=ecfr&sid=3efaad1b0a259d4e48f1150a34d1aa77&rgn=div5&view=text&node=14:2.0.1.3.10&idno=14#se14.2.91_1509

FlightGear Flight Simulator. (2016). Retrieved March 27, 2016, from FlightGear: http://www.flightgear.org/

Grappel, R. D., & Wiken, R. T. (2007). *Guidance Material for Mode-S-Specific Protocol Applicaiton Avionics.* Lexington: Massachusetts Institute of Technology.

Greenberg, A. (2012, July 25). *Next-Gen Air Traffic Control Vulnerable To Hackers Spoofing Planes Out Of Thin Air*. Retrieved Mai 14, 2016, from Forbes: http://www.forbes.com/sites/andygreenberg/2012/07/25/next-gen-air-traffic-control-vulnerable-to-hackers-spoofing-planes-out-of-thin-air/#71bcd3ff3cad

Gulf Coast Avionics. (n.d.). *Avionics | Aircraft Transponders | Search Results | Gulf Coast Avionics.* Retrieved March 11, 2016, from https://www.gulfcoastavionics.com/category/68-transponders.aspx

Haines, B. (2012). *Hackers + Airplanes No Good Can Come Of This.* Retrieved May 15, 2016, from YouTube: https://www.youtube.com/watch?v=mY2uiLfXmaI

Henn, S. (2012, August 14). Could The New Air Traffic Control System Be Hacked? *All Things Considered*. NPR. Retrieved July 23, 2016, from

http://www.npr.org/sections/alltechconsidered/2012/08/16/158758161/could-the-new-air-traffic-control-system-be-hacked

Hieb, J., Graham, J., & Guan, J. (2009). An Ontology for Identifying Cyber Intrusion Induced Faults in Process Control Systems. In *Critical Infrastructure Protection III* (pp. 125-138). Springer Berlin Heidelberg.

International Civil Aviation Organisation - Asia and Pacific Office. (2007). *Guidance Material on Comparison of Surveillance Technologies (GMST).* Retrieved June 18, 2016, from http://www.icao.int/APAC/Documents/edocs/cns/gmst_technology.pdf

International Civil Aviation Organisation. (2014). *Annual Report of the ICAO Council.* Montréal: ICAO.

International Civil Aviation Organisation. (2016). *Facts and Figure*. Retrieved from http://www.icao.int/sustainability/Pages/FactsFigures.aspx

International Civil Aviation Organisation. (2016). *Medium-Term Passaenger and Freight Traffic Forecasts.* Retrieved August 10, 2016, from ICAO: http://www.icao.int/sustainability/pages/eap_fp_forecastmed.aspx

Jaatun, M. G., & Fægri, T. E. (2013). Sink or SWIM: Information Security Requirements in the Sky. *Eighth International Conference on Availability, Reliability and Security (ARES)* (pp. 794-801). Regensburg: IEEE. doi:10.1109/ARES.2013.106

Jaiganesh, V., Mangayarkarasi, S., & Sumathi, P. (2013, April). ntrusion Detection Systems: A Survey and Analysis of Classification Techniques. *International Journal of Advanced Research in Computer and Communication Engineering, 2*(4).

John Wayne Airport, Orange County. (2013). *GA Noise Abatement.* Récupéré sur John Wayne Airport, Orange County: http://www.ocair.com/generalaviation/noise

Kacem, T., Wijesekeram, D., Costa, P., & Barreto, A. (2014). Security Requirements Analysis of ADS-B Networks. *Semantic Technologies for Intelligence, Defense, and Security.* Faifax, VA.

Karmarkar, A. R. (2012). Aviation communication infrastructure security. *Integrated Communications, Navigation and Surveillance Conference (ICNS)* (pp. E7-1 - E7-9). Herndon, VA: IEEE. doi:10.1109/ICNSurv.2012.6218392

Kelly, H. (2016, July 26). *Researcher: New air traffic control system is hackable.* Retrieved August 4, 2016, from CNN: http://www.cnn.com/2012/07/26/tech/web/air-traffic-control-security/index.html

Krozel, J., Andrisani, D., Ayoubi, M., Hoshizaki, T., & Schwalm, C. (2004). Aircraft ADS-B Data Integrity Check. *4th Aviation Technology, Integration and Operations (ATIO) Forum.* Chicago: AIAA.

Laboda, A. (2015, November 14). *Unencrypted ADS-B OUT Confounds Aircraft Blocking.* Retrieved June 27, 2016, from AINonline: https://www.ainonline.com/aviation-news/business-aviation/2015-11-14/unencrypted-ads-b-out-confounds-aircraft-blocking-0

Laminar Research. (2015). *X-Plane*. Retrieved January 14, 2016, from http://www.x-plane.com/desktop/home/

Le Monde. (2015, November 11). *Les avions civils pourraient être localisables en temps réel sur toute la Terre en 2017.* Retrieved February 6, 2016, from Le Monde: http://www.lemonde.fr/international/article/2015/11/11/les-avions-civils-devraient-etre-localisables-en-temps-reel-sur-toute-la-terre-en-2017_4807585_3210.html

Liao, H., Lin, C., Lin, Y., & Tung, K. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications, 36*, 16-24.

Lim, B. (2014). Emerging Threats from Cyber Security in Aviation – Challenges and Mitigations. *Journal of Aviation Management*, 81-91.

Lockheed Martin. (2015). *Prepar3D*. Retrieved June 15, 2016, from Prepar3D: http://www.prepar3d.com/

Magazu III, D. (2012). *Exploiting the Automatic Dependent Surveillance-Broadcast System via False Target Injection.* Master's Thesis, Air Force University, Air Force Institue of Technology, Wright-Patterson Air Force Base. Retrieved July 16, 2016, from http://www.jmargolin.com/sense3/ref15_magazu_usaf.pdf

Mark, R. P. (2015, September 1). *TCAS, ADS-B Unreliable on East Coast During September.* Retrieved March 21, 2016, from AINonline: http://www.ainonline.com/aviation-news/aerospace/2015-09-01/tcas-ads-b-unreliable-east-coast-during-september

Marks, P. (2011, September 10). Air traffic system vulnerable to cyber attack. *New Scientist, 211*(2829), 22-23. doi:10.1016/S0262-4079(11)62203-3

Martinovic, I., & Strohmeier, M. (2013). *Security of ADS-B: State of the Art and Beyond.* Retrieved December 17, 2015, from http://www.cs.ox.ac.uk/publications/publication6980-abstract.html

Massacci, F., Mylopoulos, J., Paci, F., Tun, T. T., & Yu, Y. (2011). An Extended Ontology for Security Requirements. In C. Salinesi, & O. Pastor (Eds.), *Advanced Information Systems Engineering Workshops* (Vol. 83, pp. 622-636). London: Springer Berlin Heidelberg. doi:10.1007/978-3-642-22056-2_64

McCallie, D., Butts, J., & Mills, R. (2011). Security Analysis of the ADS-B implementation in the next generation air transportation system. *International Journal of Critical Infrastructure Protection, 4*(2), 78-87. doi:http://doi.org/10.1016/j.ijcip.2011.06.001

Microsoft. (2006). *Microsoft Flight Simulator X*. Retrieved March 27, 2016, from https://www.microsoft.com/france/jeux/flight-simulator-x.aspx

Miller, S. (2015, December 14). *Why Boing 787s are appearing to lose their way.* Retrieved January 26, 2016, from Runway Girl Network: https://www.runwaygirlnetwork.com/2015/12/14/why-boeing-787s-are-appearing-to-lose-their-way/

Mitchell, R., & Chen, I. (2014, March). A Survey of Intrusion Detection Techniques for Cyber-Physical Systems. *ACM Computing Surveys, 46*(4). doi: http://dx.doi.org/10.1145/2542049

Monteiro, M., Barreto, A., & Kacem, T. (2015). Detecting malicious ADS-B broadcasts using wide area multilateration. *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)* (pp. 4A3-1 - 4A3-12). Pargue: IEEE.

NAV CANADA. (2016, July 21). Part 2 - En Route. *Aeronautical Information Publication*, pp. 1-55. Retrieved from https://www.navcanada.ca/EN/products-and-services/Documents/AIP/Next/part_2_enr/2enr_eng.pdf

NAV CANADA. (n.d.). *Surveillance dépendante automatique en mode diffusion*. Retrieved January 5, 2017, from NAV CANADA: http://www.navcanada.ca/fr/products-and-services/pages/on-board-operational-initiatives-ads-b.aspx

Nuseibeh, B., Haley, C. B., & Foster, C. (2009, September 9). Securing the skies: in requirements we trust. *Computer, 42*(9), 64-72. doi:10.1109/MC.2009.299

Object Computing, Inc. (2015, September 11). *OpenDDS 3.7.* Consulté le January 5, 2017, sur GitHub: https://github.com/objectcomputing/OpenDDS/releases/tag/DDS-3.7

Object Management Group. (2004, December). *Data Distribution Service for Real-time Systems Specification.* Récupéré sur Object Management Group: http://www.omg.org/spec/DDS/1.0/PDF/

Object Management Group. (2007). *Data Distribution Service for Real-time Systems Version 1.2.* Retrieved from http://www.omg.org/spec/DDS/1.2/

Object Management Group. (n.d.). *What is DDS?* Retrieved January 4, 2017, from http://portals.omg.org/dds/what-is-dds-3/

Object Management Group. (n.d.). *Who's Using DDS?* Retrieved from http://portals.omg.org/dds/who-is-using-dds-2/

Ontotext. (n.d.). *What is RDF Triplestore?* Retrieved August 25, 2017, from ontotext: https://ontotext.com/knowledgehub/fundamentals/what-is-rdf-triplestore/

Pardo-Castellote, G. (2005, January). *OMG Data Distribution Service: Real-Time Publish/Subscribe Becomes a Standard.* Retrieved January 4, 2017, from Real-Time Innovations: https://info.rti.com/hubfs/docs/reprint_rti.pdf

Pasztor, A. (2015, April 10). *Pilot Union Highlights Cybersecurity Concerns for Air-Traffic Control.* Retrieved January 11, 2016, from The Wall Street Journal: http://www.wsj.com/articles/pilot-union-highlights-cybersecurity-concerns-for-air- traffic-control-1428702355

Perreault, M. (2015). Révolution Dans Le Trafic Aérien. *La Presse+*. Retrieved January 19, 2016, from http://plus.lapresse.ca/screens/7436d8ee-dfd8-4034-bd40-fe5af5341e3d|2nG6ZIn4tZxP.html

Perrig, A., Canetti, R., Tygar, J., & Song, D. (2005). The TESLA Broadcast Authentication Protocol. *RSA CryptoBytes*.

Phillips, C. (2011). Récupéré sur Aircraft Situation Editor: http://www.craig-phillips.co.uk/ase/

ProtoDev Development Group. (2002, March 18). Current Protocol / Specification Document .

Purton, L., Abbas, H., & Alam, S. (2010). Identification of ADS-B System Vulnerabilities and Threats. *Australasian Transport Research Forum.* Sydney.

Python Software Foundation. (2017). *Welcome to Python.* Récupéré sur Python: https://www.python.org/

Real-Time Innovations. (2013, September 5). *NAV CANADA Enhances ATM Technology Platform with RTI Connext*. Retrieved from http://news.rti.com/pr/nav-canada-navcantrac

Roe, C. (2012, June 7). *A Short History of Ontology: It's not just a Matter of Philosophy Anymore*. Retrieved June 15, 2016, from http://www.dataversity.net/a-short-history-of-ontology-its-not-just-a-matter-of-philosophy-anymore/

Roy, J. (n.d.). *Automated Reasoning for Maritime Anomaly Detection.* Québec: DRDC. Retrieved June 13, 2016, from http://cradpdf.drdc-rddc.gc.ca/PDFS/unc92/p532648.pdf

RTL-SDR.COM. (2015, October 31). *AN UNFILTERED ADS-B CO-OP: ADSBEXCHANGE*. Consulté le July 21, 2017, sur RTL-SDR.COM: http://www.rtl-sdr.com/an-unfiltered-ads-b-co-op-adsbexchange/

Sampigethaya, K. (2009). Privacy of future air traffic management broadcasts. *2009 IEEE/AIAA 28th Digital Avionics Systems Conference* (pp. 6.A.1-1 - 6.A.1-11). Orlando: IEEE.

Sampigethaya, K. (2010). Assessment and mitigation of cyber exploits in future aircraft surveillance. *Aerospace Conference* (pp. 1-10). Big Sky: IEEE. doi:10.1109/AERO.2010.5446905

Sampigethaya, K., & Poovendran, R. (2010). Visualization & assessment of ADS-B security for green ATM. *Digital Avionics Systems Conference (DASC)* (pp. 3.A.3-1 - 3.A.3-16). Salt Lake City: IEEE. doi:10.1109/DASC.2010.5655382

Sandhu, U., Haider, S., Naseer, S., & Ateeb, O. (2011). A Survey of Intrusion Detection & Prevention Techniques. *2011 International Conference on Information Communication and Management, IPCSIT, 16.*

Schäfer, M., Lenders, V., & Martinovic, I. (2013). Experimental Analysis of Attacks on Next Generation Air Traffic Communication. In M. Jacobson, M. Locasto, P. Mohassel, & R. Safavi-Naini (Eds.), *Applied Cryptography and Network Security* (Vol. 7954, pp. 253-271). Banff, Alberta: Springer Berlin Heidelberg. doi:10.1007/978-3-642-38980-1_16

Schwab, P. (2015, September 9). *The History of Intrusion Detection Systems (IDS) - Part 1*. Retrieved November 13, 2016, from Threat Stack: http://blog.threatstack.com/the-history-of-intrusion-detection-systems-ids-part-1

SKYbrary. (2015, September 3). *Automatic Dependent Surveillance - Contract (ADS-C)*. Retrieved March 14, 2016, from SKYbrary Aviation Safety: http://www.skybrary.aero/index.php/Automatic_Dependent_Surveillance_-_Contract_(ADS-C)

SKYbrary. (2015, November 3). *Automatic Dependent Surveillance Broadcast (ADS-B)*. Retrieved March 14, 2016, from SKYbrary Aviation Safety: http://www.skybrary.aero/index.php/ADS-B

Smith, A. E. (2006). *United States of America Patent No. US20060119515*.

Smith, A., Cassell, R., Breen, T., & Hulstrom, R. (2006). Methods to Provide System-Wide ADS-B Back-Up, Validation and Security. *ieee/aiaa 25TH Digital Avionics Systems Conference* (pp. 1-7). Portland: IEEE.

Spaccapietra, S., Parent, C., Damiani, M. L., de Macedo, J. A., Porto, F., & Vangenot, C. (2008, April). A conceptual view on trajectories. *Data & Knowledge Engineering, 65*(1), 126-146.

Strohmeier, M., Lenders, V., & Martinovic, I. (2015, May 19). On the Security of the Automatic Dependent Surveillance-Broadcast Protocol. *IEEE Communications Surveys & Tutorials, 17*(2), 1066-1087. doi:10.1109/COMST.2014.2365951

Strohmeier, M., Schäfer, M., Pinheiro, R., Lenders, V., & Martinovic, I. (2016). *On Perception and Reality in Wireless Air Traffic Communications Security*. Retrieved from arXiv: http://arxiv.org/abs/1602.08777

The Apache Software Foundation. (2016). *Apache Jena - Jena Ontology API*. Consulté le June 16, 2016, sur Apache: https://jena.apache.org/documentation/ontology/

Thurber, M. (2012, August 21). *Hackers, FRAA Disagree Over ADS-B Vulnerability*. Retrieved January 26, 2016, from AINonline: http://www.ainonline.com/aviation-news/air-transport/2012-08-21/hackers-faa-disagree-over-ads-b-vulnerability

Tippenhauer, N. O., Pöpper, C., Rasmussen, K. B., & Capkun, S. (2011). On the Requirements for Successful GPS Spoofing Attacks. *Proceedings of the 18th ACM Conference on Computer and Communications Security* (pp. 75-86). New York: ACM.

Transport Canada. (2014, February 19). *Canadian Civil Aircraft Register: Quick Search*. Retrieved from Transport Canada: http://wwwapps.tc.gc.ca/saf-sec-sur/2/ccarcs-riacc/RchSimp.aspx

Transport Canada. (2017, March 01). *Managing noise from aircraft.* Retrieved March 1, 2017, from Transport Canada: https://www.tc.gc.ca/eng/civilaviation/standards/aerodromeairnav-standards-noise-menu-923.htm#_Modifying_procedures_to

U.S. Government Accountability Office. (2015). *Air Traffic Control: FAA Needs a More Comprehensive Approach to Address Cybersecurity As Agency Transitions to NextGen.* Washington. Retrieved from https://www.gao.gov/products/GAO-15-370

U.S. Government Accountability Office. (2015). *FAA Needs a More Comprehensive Approach to Address Cybersecurity As Agency Transitions to NextGen.* Washington.

Vabre, P. (n.d.). *Primary and Secondary Radar*. Retrieved January 25, 2016, from http://www.airwaysmuseum.com/Surveillance.htm

Vandecasteele, A., & Napoli, A. (2012). An enhanced spatial reasoning ontology for maritime anomaly detection. *System of Systems Engineering* (pp. 1-6). Genoa: IEEE.

Vinod, P., Jaipur, R., Laxmi, V., & Gaur, M. (2009). Survey on Malware Detection Methods. *Proceedings of the 3rd Hackers' Workshop on computer and Internet security (IITKHACK'09)*, (pp. 74-79). Kanpur.

W3C. (2016, June 1). *RDF - Semantic Web Standards*. Retrieved from W3C: https://www.w3.org/RDF/

W3C. (n.d.). *SPARQL - Semantic Web Standards*. Retrieved from W3C: https://www.w3.org/2001/sw/wiki/SPARQL

Walker, G. (2012, October 28). *Is air traffic control a soft target for hackers?* Retrieved January 23, 2016, from NATS blog: http://nats.aero/blog/2013/10/is-air-traffic-control-a-soft-target-for-hackers/

Wood, R. G. (2009, May). *Security Risk Analysis of the Data Communications Network Proposed in the Nextgen Air Traffic Control System.* Dissertation, Oklahoma State University, Graduate College, Stillwater. Retrieved May 16, 2016, from https://shareok.org/bitstream/handle/11244/7198/School%20of%20Educational%20Studies_22.pdf

Zetter, K. (2012, July 26). *Air traffic controllers pick the wrong week to quit using radar.* Retrieved February 17, 2016, from Wired: https://www.wired.com/2012/07/adsb-spoofing/

APPENDIX A – SPARQL QUERIES

AIRCRAFT MATERIALIZATION

```
01 SELECT ?aircraftID ?sensor
02 { ?ddsSubject rdf:type :ADSBReport }
03 UNION
04 { ?ddsSubject rdf:type :SSRPosition }
05 UNION
06 { ?ddsSubject rdf:type :PSRPosition }
07 SELECT ?airport
08 ?ddsSubject :latitude ?latitude1
09 ?ddsSubject :longitude ?longitude1
10 ?airport :position :latitude ?latitude2
11 ?airport :position :longitude ?longitude2
12 ?distance = getDistance(?longitude1, ?latitude1, ?longitude2,
   ?latitude2)
12 FILTER(?distance < DIST_MIN)
13 ?ddsSubject :publishedBy ?sensor
```

ROUTE DEVIATION

Part 1: scheduling the verification query

```
01 NOW = current time
02 SELECT ?flightplan
03 ?flightplan :nextwaypoint ?nextwaypoint
04 ?flightplan :lastwaypoint ?lastwaypoint
05 ?flightplan :aircraftPlanID ?aircraftPlanID
06 SELECT ?aircraftID ?airspeed
07 { ?ddsSubject rdf:type :ADSBReport }
08 FILTER(?aircraftID == ?aircraftPlanID)
09 ?traveltime = computeTravelTime(?lastwaypoint, ?nextwaypoint,
   ?airspeed)
10 scheduleQuery(NOW + ?travelTime, ?nextwaypoint, ?aircraftPlanID)
```

Part 2: the verification query

```
01 ID = ID of the aircraft for which the verification query was scheduled
02 WAYPOINT = the waypoint the aircraft was supposed to reach before this
   query runs
03 SELECT ?flightplan
04 ?flightplan :aircraftPlanID ?aircraftPlanID
05 ?flightplan :lastwaypoint ?lastwaypoint
06 FILTER(?aircraftPlanID == ID && ?lastwaypoint != WAYPOINT)
07 ?aircraftPlanID
```

PROHIBITED ZONES

```
01 ZONE_RADIUS = radius of the prohibited zone
02 SELECT ?aircraftID ?sensor
03 { ?ddsSubject rdf:type :ADSBReport }
04 UNION
05 { ?ddsSubject rdf:type :SSRPosition }
06 UNION
07 { ?ddsSubject rdf:type :PSRPosition }
08 SELECT ?zoneID
09 ?zoneID :latitude ?zonelat
10 ?zoneID :longitude ?zonelong
11 ?zoneID :altitude ?zoneceil
12 ?aircraftID :altitude ?aircraftalt
13 FILTER(?zoneceil > ?aircraftalt)
14 ?aircraftID :latitude ?aircraftlat
15 ?aircraftID :longitude ?aircraftlong
16 ?distance = getDistance(?aircraftlat, ?aircraftlong, zonelat,
   zonelong)
17 FILTER(?distance < ZONE_RADIUS)
18 ?aircraftID :publishedBy ?sensor
```

NOISE REDUCTION

```
01 SELECT ?flightplan ?airport
02 ?flightplan :departureAirport ?departureAirport
03 ?flightplan :departureTime ?departureTime
04 ?flightplan :arrivalAirport ?arrivalAirport
05 ?flightplan :arrivalTime ?arrivalTime
06 ?airport :noiseReductionTimeStart ?noiseStart
07 ?airport :noiseReductionTimeEnd ?noiseEnd
08 FILTER(?departureAirport == ?airport || ?arrivalAirport == ?airport)
09 FILTER(
       (?departureTime > ?noiseStart && ?departureTime < ?noiseEnd)
    || (?arrivalTime >?noiseStart && ?arrivalTime < noiseEnd))
10 ?flightplan :aircraftPlanID
```

FUEL EMERGENCY

Part 1: scheduling the verification query

```
01 ID = ID of the aircraft for which we are scheduling the verification
   query
02 NOW = current time
03 SELECT ?flightplan
04 ?flightplan :aircraftID ?aircraftID
05 FILTER(?aircraftID == ID)
06 ?flightplan :fuelLevel ?fuelLevel
07 ?flightplan :route ?route
08 ?flightTime = computeFlightTime(?route, ?fuelLevel)
09 scheduleQuery(?flightTime + NOW)
```

Part 2: the verification query

```
01 ID = ID of the aircraft this query needs to look up
02 LOW_FUEL = amount of fuel that is considered low
03 SELECT ?aircraftID ?status ?fuelLevel
04 FILTER(?aircraftID == ID && ?status == OPEN && ?fuelLevel <= LOW_FUEL)
05 ?aircraftID
```

ANOMALOUS DESCENT

```
01 ALT_DIFFERENCE = maximum reasonable difference between two positions'
   altitudes
02 TIME_RANGE = maximum time difference between two updates to compare
   them
03 SELECT ?aircraftID ?timestamp1 ?sensor
04 { ?ddsSubject1 rdf:type :ADSBReport
05 ?ddsSubject2 rdf:type :ADSBReport }
06 UNION
07 { ?ddsSubject1 rdf:type :SSRPosition
08 ?ddsSubject2 rdf:type :SSRPosition }
09 ?ddsSubject1 :ID ?aircraftID
10 ?ddsSubject2 :ID ?aircraftID
11 FILTER(?ddsSubject1 != ?ddsSubject2)
12 ?ddsSubject1 :timestamp ?timestamp1
13 ?ddsSubject2 :timestamp ?timestamp2
14 FILTER(|?timestamp1 - ?timestamp2| < TIME_RANGE)
15 ?ddsSubject1 :altitude ?altitude1
16 ?ddsSubject2 :altitude ?altitude2
17 FILTER(|altitude1 - altitude2| > ALT_DIFFERENCE)
18 ?ddsSubject1 :publishedBy ?sensor
```

GHOST AIRCRAFT AND SCREEN CLUTTER

```
01 GHOST_FREQUENCY = alert frequency to consider ghost aircraft attack
02 CLUTTER_FREQUENCY = alert frequency to consider screen clutter attack
03 TIMESPAN = timespan to consider alerts
04 NOW = current time
05 SELECT ?timestamp
06 { ?alert rdf:type :materialization }
07 UNION
08 { ?alert rdf:type :violationphysics }
09 FILTER (NOW – TIMESPAN < ?timestamp)
10 ?alertfreq = COUNT(?timestamp) / TIMESPAN
11 IF(?alertfreq > GHOST_FREQUENCY && ?alertfreq < CLUTTER_FREQUENCY)
12 THEN SEND_GHOST_ALERT()
13 ELSE IF(?alertfreq > CLUTTER_FREQUENCY)
14 THEN SEND_CLUTTER_ALERT()
15 END
```

JAMMING AND SCREEN CLUTTER

```
01 NOW = current time
02 TIMESPAN = timespan to compute average traffic
03 AVERAGE_DENSITY = average air traffic density at current time
04 DENSITY_DIFF = maximum allowed difference between current and average
   densities
05 SELECT UNIQUE ?aircraftID ?timestamp
06 { ?ddsSubject rdf:type :ADSBReport }
07 UNION
08 { ?ddsSubject rdf:type :SSRPosition }
09 UNION
10 { ?ddsSubject rdf:type :PSRPosition }
11 FILTER(NOW - TIMESPAN < ?timestamp)
12 ?density = COUNT(?aircraftID) / TIMESPAN
13 IF(AVERAGE_DENSITY - DENSITY_DIFF > ?density)
14 THEN SEND_JAMMING_ALERT()
15 ELSE IF(AVERAGE_DENSITY + DENSITY_DIFF < ?density)
16 THEN SEND_CLUTTER_ALERT()
17 END
```

SYSTEM TAMPERING

```
01 AVERAGE_FREQUENCY = regular alert frequency
02 TIMESPAN = timespan to consider recent alerts
03 NOW = current time
04 FREQUENCY_DIFF = maximum difference in frequencies allowed
05 SELECT ?timestamp
06 { ?alert }
07 FILTER(NOW - TIMESPAN < ?timestamp)
08 ?frequency = COUNT(?timestamp) / TIMESPAN
09 IF(?frequency > AVERAGE_FREQUENCY + FREQUENCY_DIFF)
10 THEN SEND_TAMPERING_ALERT()
11 END
```