**OMAP: Military Standard 1553B
(MIL-STD-1553B) device classification framework.
A comparative approach study**

by

Olivier Courchesne

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Applied Science in Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada

May 2021

Copyright ©

2021 - Olivier Courchesne

Abstract

In 2017, Canada presented to its citizen its intent to buy C\$90 million worth of fighter jets from Australia. While these fighter jets need regular modifications to be kept up to date with the airworthiness standard of Canada, they also rely on older aircraft architecture and standards, mainly the Military Standard 1553B (MIL-STD-1553B). This standard dictates how the network and communications within an aircraft must be made. Dating back to 1973, MIL-STD-1553B does not consider cybersecurity within the needs of an aircraft. However, with the drastic increase of cyberattacks, it is imperative to introduce an identification mechanism which can identify the components that are present within a network as well as their version in order to verify that an aircraft is flying with an approved and safe configuration. Such identification tools, like Nmap, exist for Internet of Things (IoT) devices that uses standard communication protocols such as TCP and UDP however they are inapplicable when dealing with atypical communication protocols like the MIL-STD-1553B. In this thesis, we investigate the MIL-STD-1553B technology used in current aircraft systems and explore how aircraft components that rely on this standard can be automatically classified. We propose a two step classification process based on the commonly used IoT cybersecurity software Nmap. The process actively probes the MIL-STD-1553B devices onboard of each aircraft to map their memory content before measuring the response time of the devices on a set of standard operations. Following linux's "Less

is More”, this process, named OMAP, classifies the devices by types and then identifies their version. We propose a novel active scanning approach to establish message timing, built-in test responses and memory anomalies in order to classify the devices plugged in the aircraft’s data bus. The proposed OMAP is able to classify device types and differentiate between two versions of the same device type by mapping the memory contents of each device and measuring response time. To do so, we compared the accuracy of multiple Machine Learning (ML) classification algorithms when exposed to different test case scenarios to obtain the best overall accuracy. The comparative points we studied were the following: One step vs Two step classification, Joined vs Separate spoofed class, Timing features granularity effects. Finally, using Artificial Neural Networks (ANN) and Support Vector Classifier (SVC) it was possible to obtain an accuracy rating over 95% for device type and 88% for version classification. As such, OMAP could become the next tool to monitor proper system configuration and maintain device integrity on military aircraft.

Acknowledgments

First and foremost, with my most sincere gratitude, I would like to acknowledge my supervisor Dr. Wei Shi for her dedication, patience, encouragement, and support. Without her guidance, knowledge and expertise, this thesis would have not been possible. Secondly, I would like to thank the personnel of the Royal Military College for providing me with the instrumentation that allowed me to test the presented framework. Special thanks to Dr. Ronald Smith and Major Guillaume Vigeant for their insight and helpful discussions. I would also like to thank Captain Dakotah Soucy and Major Jeffrey Campbell for providing me with help in operating the tools used in this thesis.

Lastly, I would like to thank my family and friends for their encouragement and support for my research interests and educational pursuits. A special thank goes to my girlfriend Catherine for helping me to stay motivated throughout this thesis.

“The school was not the end; it was rather the instrumental means for improving the way of life.”

Don Bosco

Table of Contents

Abstract	ii
Acknowledgments	iv
Table of Contents	v
List of Tables	viii
List of Figures	ix
Nomenclature	xi
1 Introduction	1
1.1 Motivation	2
1.1.1 Cybersecurity Concerns	3
1.1.2 Cyber attacks	5
1.1.3 Common Cyber tools concepts	6
1.1.4 MIL-STD-1553 aircraft communications	8
1.2 Problem statement	9
1.3 Thesis Contribution	11
1.4 Organization	13
2 Related Work	14

2.1	Device integrity	14
2.2	ML classification based on communication	15
2.3	Fingerprinting	17
2.4	Intrusion detection Systems	18
2.5	MIL-STD-1553B Problem Specific litterature	20
3	Background and Methodology	21
3.1	MIL-STD-1553B	21
3.2	Cyber Attack Techniques	26
3.2.1	Attack Vectors	27
3.2.2	Threats	27
3.3	Classification algorithms	28
3.4	Methodology	31
4	OMAP	32
4.1	Feature Selection	32
4.2	OMAP classification process	37
4.3	Device and Version Identification	42
5	Experimental Evaluation	46
5.1	Dataset	46
5.2	Experiment setup	48
5.3	Experimental result	49
5.3.1	One step classification vs Two step Classification	49
5.3.2	Comparative Evaluation on RT Type Classification	56
5.3.3	Comparative Evaluation on RT Version Classification	59
5.3.4	Comparative execution time analysis	61

6 Conclusion and Future Work	63
6.1 Conclusions	63
6.2 Future Work	66
Appendix A Accuracy and Confusion matrix for device classification joined spoof and legitimate	68
Appendix B Accuracy and Confusion matrix for device classification separated spoof and legitimate	73
Appendix C Accuracy and Confusion matrix for version classification	78
Appendix D Confusion matrix for device Type classification separated spoof and legitimate comparison	83
Bibliography	90

List of Tables

4.1	1553 Message Types examples	33
4.2	1553 exploitable Timings	34
5.1	Accuracy comparison RT Type (4 classes)	57
5.2	Accuracy comparison RT Type (5 classes)	59
5.3	Accuracy comparison RT version	61

List of Figures

3.1	1553 Sample MIL-STD-1553B Multiplex Data Bus Architecture [1]	22
3.2	1553 MIL-STD-1553B Command Word bit usage [1]	23
3.3	1553 MIL-STD-1553B Status Word bit usage [1]	23
3.4	1553 MIL-STD-1553B Data Word bit usage [1]	23
3.5	1553 Information Transfer Formats [2]	24
3.6	1553 MIL-STD-1553B Mode Codes [1]	25
4.1	OMAP flow chart	37
4.2	Example of RT memory contents	45
5.1	Experimental setup of 1553 databus network	48
5.2	Confusion Matrix for the OMAP classification	50
5.3	Confusion Matrix comparison for the OMAP version classification: D0R0 vs D2R2	51
5.4	Confusion Matrix for the OMAP classification no spoofing detection	53
5.5	Confusion Matrix for the OMAP classification with spoofing detection	53
5.6	Accuracy Matrix for the OMAP device classification	54
5.7	Accuracy Matrix for the OMAP version classification	55
5.8	Studied models loss during training period	60
5.9	Studied models execution time	62
A.1	Accuracy Matrix for the OMAP device classification	68
B.1	Accuracy Matrix for the OMAP device classification separate	73

C.1	Accuracy Matrix for the OMAP version classification	78
D.1	Accuracy Matrix for the ANN 5 Types classification	83
D.2	Accuracy Matrix for the SVC 5 Types classification	84
D.3	Accuracy Matrix for the CNN 5 Types classification - 100 iterations .	84
D.4	Accuracy Matrix for the CNN 5 Types classification - 300 iterations .	85
D.5	Accuracy Matrix for the MLP 5 Types classification	85

Nomenclature

Abbreviation	Explanation
ANN	Artificial Neural Networks
API	Application Programming Interface
BC	Bus Controller
BIT	Built-in Tests
BM	Bus Monitor
CIA	Confidentiality, Integrity, Availability
CNN	Convolutional Neural Network
DoS	Denial of Service
IAT	Inter Arrival Time
IDS	Intrusion Detection System
IoT	Internet of Things
KNN	K-Nearest Neighbor
ML	Machine Learning

MLP	Multi-layer Perceptron
NN	Neural Network
OS	Operating System
RNN	Recurrent Neural Network
RT	Remote Terminal
SVC	Support Vector Classifier
SVM	Support Vector Machine
SA	Sub-Address
1553	MIL-STD-1553B

Chapter 1

Introduction

As of 2019, the United States has taken a cyber mission defense team approach to handle the growing risk of cyber attacks on their military systems including weapon systems, aircraft such as the E-3 Sentry Airborne Warning and Control System (AWACS) as well as their network and infrastructure [3]. This includes the use of specific teams focused on a single weapon system, cyber enabled officers and continuous training to upgrade its cyber defense programs. Vulnerability assessments, reverse engineering as well as pattern and anomaly detection are commonly used cybersecurity techniques that rely on specific tools and resources to achieve defensive cyber operations. Nmap is one of these highly specialized tools which allows for the identification of devices and their version on a classic computer network through the use of scanning ports, protocol headers and more. While it is a powerful tool for device discovery and ultimately their classification, it is also bound to the usual web protocols, such as TCP/UDP; rendering it ineffective for uncommon message exchange protocols. This thesis focuses on the use of the military standard 1553B(MIL-STD-1553B), a standard that defines the functional characteristics of a serial data bus used in the Canadian Armed Forces avionics systems. While the standard describes a 1 megabit, time division, half duplex encoded using the Manchester II Bi-Phase (Non-Return to

Zero, NRZ) with data words comprised of 20 bits, its simplicity and lack of informative header can pose a problem in standard classification approaches such as Internet of things (IoT) device classification. Furthermore, since various avionic devices used by the military are outsourced, it is not uncommon for manufacturers to take liberties with the behavior of their devices outside of the MIL-STD-1553B specified protocol. According to [4], very little research has been conducted on the security of the MIL-STD-1553 (1553 from now on) bus and most of the previous work focused on its integration with other systems as well as improving its efficiency. This research thesis aims to develop a 1553 enabled discovery tool, named OMAP, in order to address one of the main cybersecurity triad concept: integrity. This proposed automatic 1553 device identification process can identify faulty data injection in 1553 devices as well as those that are corrupted. It is also possible for OMAP to detect spoofed devices through timing analysis. Finally, through the classification process, it is possible to authenticate the devices that are present on the 1553 aircraft network and correlate them with the appropriate configuration of their residing aircraft. As such, through the use of OMAP it is possible to determine through an automatic data inspection, the type and version of a device as well as whether a hardware component on the 1553 network's or its software integrity has been put at risk.

1.1 Motivation

In the cyber enabled world we live in, cyber threats come in multiple shapes and forms. Gone are the days of single hackers taking down multiple infrastructures on their own. We rather have to face state sponsored cyber attacks and multi-level cyber attacks. This is especially true since the advent of remote work with the Microsoft Teams and ZOOM applications widely adopted through the COVID-19 pandemic as presented in [5]. Hence, more and more attacks occur and their attack vectors and

goals also diversify. Below, in section 1.1.2, we list typical types of attack that are commonly used.

Beyond the fact that more people are using applications with security vulnerabilities to pursue their everyday work, cyber trained individuals now have access to a wide variety of free and readily available tools. By design these powerful cyber tools can be used for beneficial or harmful goals. Tools such as Nmap are made widely available and could be used not only to find weaknesses in a network, but also detect rogue or misconfigured devices on a network. Furthermore, more and more devices, with small form factors like Flipper [6] allow hacking on the go of ready to access devices like Sub-1 GHz transceivers, infrared transceivers, RFID tags, NFC and Bluetooth enabled devices, etc. The ease of access to such hacking tools puts many networks and individuals at risk. Combining the ease of access of these hacking tools with emulators like the MIL1553NE [7], which allows to emulate the basic function of a 1553 BC and Remote Terminal (RT), paints a grim picture for the cybersecurity of 1553 aircraft. As presented in [8], cybersecurity was never meant to be a core consideration of the 1553 standard. Hence, there is a need to acknowledge the existence of spoof capable devices and to counter them through the implementation of defensive tools to assure proper functions of 1553 aircraft as well as its cybersecurity.

1.1.1 Cybersecurity Concerns

On the other hand, institutions are increasingly aware of cyber threats and put in place an increasing amount of tools and guidelines to minimize the risks. Such guidelines include proper training on basic cybersecurity concepts, for instance password management and the cyber triad. Regarded as the stepping stone to cybersecurity the confidentiality, integrity, availability (CIA) cyber triad are usually supplemented by accountability. It is also possible to reinforce these concepts with the use of authentication to bolster the security of a network. The cyber triad as well as the

authenticity and accountability concept are detailed below.

- Confidentiality: Is defined by the NIST special publication (SP) #7316 as the "Assurance that information is not disclosed to unauthorized persons, processes, or devices" [9]. Confidentiality covers data in storage, during processing, and in transit". It relates to the act of separating what information is given to whom. Through the use of security clearance, encryption, access control, file permission, etc., it is possible for an institution to protect their sensitive information from unauthorized eyes.
- Integrity: Is defined by the NIST special publication (SP) #800-56B Rev. 1 as "A property whereby data has not been altered in an unauthorized manner since it was created, transmitted or stored. In this Recommendation, the statement that a cryptographic algorithm "provides data integrity" means that the algorithm is used to detect unauthorized alterations" [10]. This in turn usually brings authenticity and accountability along, in an effort to validate acceptable data modification.
- Availability: As stated by [11], "The state that exists when data can be accessed or a requested service provided within an acceptable period of time". Availability relates to the data being accessible to the user either when needed or at all times, as required.
- Authenticity: Authenticity is the act of proving one is truly who they claim to be. According to computer security resource center of the NIST, authenticity is the "property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator" [12]. Whether by the use of tokens, IP or signatures many authentication processes are available.

- **Accountability:** This last concept, which is less implemented per its technological requirements, relates to keeping track of the changes that are done in a network. According to NIST it is defined in [13] as "the security objective that generates the requirement for actions of an entity to be traced uniquely to that entity". This allows an administrator to know who did all the changes and then determine if they were legitimate.

1.1.2 Cyber attacks

Many types of cyber attacks occur every day. In 2020, Cisco, one of the world's biggest telecommunication company, defined them as "a malicious and deliberate attempt by an individual or organization to breach the information system of another individual or organization" [14]. While some attacks are protocol and application specific, the following categories are not and can pose a threat to all types of systems, independently of the network and devices used in it. As such they prove to be a universal threat and should be considered even in obscure protocols such as 1553.

- **Malware:** Malicious software introduced into a component to impede its proper functioning. Malware usually refers to spyware, ransomware, viruses, worms or software that will eventually lead to their insertion through a vulnerability like a malicious link.
- **Phishing, social engineering and supply chain attacks:** Phishing is one the most prominent cyber attack that is universally used as a malicious entry point. It usually starts as an entity masquerading as a trusted source to gain information. This fraudulent communication can utilize prior insiders knowledge acquired through social engineering to achieve phishing attacks that can lead to infiltration of the supply chain, manufacturers or technicians. Once the access is granted to the attacker, they can modify the software contents and data,

putting at risk the integrity of the targeted device.

- Denial-of-service attack: DoS is an attack that consists of flooding a system with requests to exhaust the resources. By blocking the message transfer and stopping the availability of central message transfer, legitimate traffic and requests are not able to reach their destination, therefore impeding the proper functioning of the system and the network.
- Man-in-the-middle attack: Also known as the eavesdropping attack; this type of attack consist of intercepting traffic to obtain data then forwarding the message back to its supposed destination. This allows the attacker to filter, steal and change data in a two-party communication, putting at risk the integrity of the data as well as its possible confidentiality.

1.1.3 Common Cyber tools concepts

Many tools are at the disposition of engineers who wish to secure their networks. While they all have a specific purpose, the following three common cybersecurity tools represent three broad types of tools used in customary security audits: Intrusion Detection Systems (IDS), Packet Analyzers and Network Scanners. These three types of tools are presented below by presenting one respective application.

- Snort [15] : Created in 1998 by Martin Roesch, Snort is an open source network intrusion detection system and intrusion prevention system. Snort allows its user to create custom rules stopping or rerouting packets. Sitting behind the firewall, Snort adds another layer of security to standard computer networks. With its custom rules it is possible to create a powerful intrusion detection system. Hence, Snort is in need of an experienced administrator to be effective.

- WireShark [16] : Originally named Ethereal, this open source packet analyzer created in 1998 by Gerald Combs is a powerful way, for its users, to examine packets in a human readable format. Applying filters, it is possible to scrutinize specific traffic for network troubleshooting, analysis, software , communications protocol development, and educational purposes.
- Nmap [17] : Being free and open source the network scanner Nmap created by Gordon Lyon in 1997 is widely used by cybersecurity experts working with IOT devices. Mainly used for network discovery and security audits, Nmap allows its user to know what device is present on their network as well as their operating system and version. With features such as feature analysis, host discovery, port scanning, Operating System (OS) detection, services and services version detection, Nmap is regarded as a great tool but also a threat for many network administrators. Nevertheless, as presented in [18] Nmap should be a fundamental part of any security assessment for organizations.

The above mentioned tools are commonly used in typical computer networks, relying on usual computer messages exchange protocols and their specificity. These tools leverage message headers, open ports, protocol types, clear text payloads, responses to messages and more to achieve their goal. Hence, they allow their user to attain a fountain of knowledge from exploitable underlying information. While powerful, these tools are also limited by the technology they use. As such, when dealing with technology that is atypical, such as military devices that use custom messaging exchange protocols, they are rendered useless and will leave their users in need for an alternative. The following section presents specifications of the 1553 protocol that directly impact the design of the proposed 1553 network device discovery, type and version classification tool.

1.1.4 MIL-STD-1553 aircraft communications

In this thesis we used publicly available communications protocol to emulate MIL-STD-1553 network based aircraft communications. The overuse of an archaic cybersecurity process: security through obscurity [19] has plagued the military aircraft community and left them in much need for additional security measures. Applying the CIA cyber triad to an aircraft comes with its lot of consideration from which stems the Military airworthiness and its Technical Airworthiness Authority. Within these, the CIA model is adopted alongside a criticality level based on the risk implied when failing. Hence, on 1553 the integrity of some RT is critical since improper data can potentially lead to a capture and control via GPS spoofing as seen in [20] or even a crash. Furthermore, the threat of social engineering and supply chain infiltration is serious as can be seen in [21]. With the recent Solarwinds supply chain attack incident [22] which introduced malware through a legitimate software update, the need for RT data and version integrity is reinforced. In turn, the need for integrity pushed for accountability features. Through the implementation of a Bus Monitor (BM), 1553 tracks all the messages that propagate through the databus. Hence, the information that is propagated can then be linked back to its source through the messaging black box that is the BM. There is however no authentication process present on a 1553 aircraft leaving the door open for spoofing attacks or malicious changes made through legitimate means as was the case with [22]. Furthermore, since the BC and Remote Terminals (RTs) have no identification mechanism, a 1553 network will benefit of the use of intrusion detection systems to monitor its communications for anomalies. Implementing such mechanism does not however solve the known unknown that is the issue of identification of RT versions. One of the reasons no identification process has been created for 1553 is due to its communication exchange protocol. In 1553, the BC is the center point for all communications. The exchanged packets follow

a protocol that is presented in chapter 3 and exemplary messages are presented in chapter 4 Table 4.1. Nevertheless, the lack of informational headers for these packets renders the use of traditional probes, classification and tools like the ones presented in section 1.1.3 mostly unusable. Their concept, data acquisition process and method of classification are, nevertheless, considered for the development of the OMAP tool designed for this thesis.

1.2 Problem statement

Through the motivation section of this thesis it is now apparent that the cybersecurity of a 1553 aircraft is less than ideal. While many core cybersecurity concepts, such as CIA, apply to 1553, the tools that are usually used to enforce these cybersecurity concepts are not applicable due to the oddity of the 1553 message exchange protocol. Furthermore, while the BC is in charge of all communications that happen on the databus, its knowledge about what is present on the databus is limited by what the technicians implemented into it. Hence, it is in no way capable of distinguishing between a legitimately added component, modified component and their adversarial counterparts. With supply chain attacks like Solarwinds [22] being more common, it is imperative to provide a way to classify the devices present on a 1553 databus as well as their version. Doing so would allow to limit the introduction of malicious components, whether hardware or software, into a 1553 aircraft. Such a tool could allow a technician to quickly assess the integrity of an RT by making sure its behaviour corresponds to what is expected, as well as making sure its data contents integrity is also within regulations. To obtain such a tool, training data must first be acquired. As such, it is necessary to create a custom BC and BM which will allow to probe and record the RTs. Furthermore, since the required data are contained in 1553 messages, it must first be parsed and pre-processed before being utilizable for machine learning

algorithm studies. The following presents these core components of the proposed OMAP tools:

- 1553 network probe: In a effort to obtain features from the RTs, the present research puts in place a 1553 network probe. In a similar fashion to the probe of Nmap that scans for active port, this probe reaches out to every component of the 1553 network by acting like a BC and then requests the information stored within all sub-addresses. Doing so allows us to map the memory content of all RTs, ensuring their content is within regulation and as such that their integrity is confirmed. The probe also collects information regarding specific tasks allowing us to differentiate between versions later on.
- 1553 communication parser: When a single RT has been probed it answers to the OMAP-BC through a series of messages. These messages are not ready to be used for machine learning and artificial intelligence. They must first be parsed to extract the data and then format it adequately before pre-processing the information for the chosen algorithm.
- Emulation of 1553 devices: For testing purposes multiple RTs needed to be created. Based on signatures contained in the GE/BUSTOOLS software and data contained from a radio RT, a series of RT emulation were created. They comprise a series of dataset that are stored as RT requests response messages to probing sequence before being sent over the network and then parsed. Due to the high cost of 1553 devices, it was not possible to obtain additional real devices. With the current experimental setup that is presented in figure 5.1 the cost was already well over 20000\$. While a real device was available at the Royal Military College, the limitation surrounding the 2019-2021 COVID19 pandemic made the access to the device impossible.

1.3 Thesis Contribution

The goal of this thesis is to address several aspects of the cybersecurity issues that exists in aircraft systems using the MIL-STD-1553. We aim to augment the security through the development of a set of cybersecurity tools that are specifically designed for MIL-STD-1553 communications. The following presents the contribution that pertain to the MIL-STD-1553 cyber security field.

- Data acquisition: To create a dataset to be used for classification, data had to first be obtained from a 1553 network. Through this thesis we created a message exchange BC which probed a 1553 network to obtain responses from 6 custom RTs some of them with multiple versions. As stated above, we then parsed these messages to extract the data and timing features from within them prior to structuring them for classification by creating image-like matrices. These images were then preprocessed to scale them to the needs of the used classification algorithm. Through this data acquisition we created datasets comprised of 100 readings for each RT type and version. We then adjusted the timing features of the custom RTs to create datasets that could be used to study the impact of timing granularity on classification. This is the first time an active probing dataset was obtained and cleaned in an effort to map the memory contents of 1553 devices.
- Comparison of best models for 1553 RT classification: By using the same data for multiple models we tested the accuracy, training time and loss for a multitude of algorithms when applied to 1553 data. In this thesis we studied Artificial Neural Networks (ANN), Convolution Neural Networks (CNN), Multi-layer Perceptron (MLP), Support Vector Classifier (SVC) and K-Nearest neighbours (KNN). Our results show that MLP, ANN or CNN provide the best classification results for RT types and SVC for RT version classification.

- Comparison of approach:

One Step vs Two step classification: Since this thesis aims to classify RTs for their version and their type, there surfaces a question as to how to classify both. In an effort to obtain the best overall accuracy, we compared the efficiency and accuracy ratings of classifying in a "One Step" and "Two Step" process. This refers to either classifying RT types and versions all at once or classifying RT types first and then classifying their versions. Through this thesis, it was shown that by adopting a two step classification process and classifying version independently from types, it was possible to achieve better accuracy, 95%, than when using a one step classification where it reached 90% at best.

1553 Device classification application using various Machine Learning (ML) techniques: Regardless of the approach taken, we tested the accuracy of ANN, CNN, MLP and SVC for type classification and SVC and KNN for version classification. We then subjected each of these algorithms to a test case scenario "Joined vs Separate spoofed class" to examine the effect of similar classes on the accuracy. Furthermore, different features were used for both layers of classification which permitted to attain over 95% accuracy for device type classification and between 80% and 95% accuracy for version classification depending on the applied delta value for Built-in Test (BIT) time and reset time.

- Confirmation of data and RT integrity: Through the classification process, it was also possible to establish data integrity and authenticity of the RTs by a second hand effect. The method used shows that it is possible to identify spoofed RTs, wrongfully configured RTs and corrupted data based on standard messages in MIL-STD-1553. By using the classification process that maps the memory contents of each RT, it was shown that it is possible to detect integrity faults within in RT through the presented approach with an accuracy between

80% and 99% depending on the the timing feature scenario. This contribution is however only a second hand effect of obtaining a classification tool that detects faulty RTs as their own class.

1.4 Organization

The following chapters of this research thesis are as follows. Chapter 2 reviews present and past work related to this thesis as well as establish the key takeaways for the present research. Chapter 3 describes in more depth the standard and guidelines that make up the 1553 standard. The main cyber security concerns that apply to a 1553 are also discussed in chapter 3 as well as how they impacted the present research. The methodology used for the OMAP tool is presented at the end of Chapter 3. Chapter 4 further describes the OMAP process and a functional explication of each of the three components that allow the proper classification of the devices (Probe, Parser, Classifier). Chapter 5 presents the experimental evaluation of OMAP as well as the obtained results. Finally, chapter 6 concludes this research thesis with an assessment of OMAP as well as the future work to be made to OMAP in an effort to obtain a multipurpose cyber tool for 1553 aircraft.

Chapter 2

Related Work

2.1 Device integrity

The MIL-STD-1553B comes with its load of Designer's Notes and manual as can be seen in [1]. Consulting these it is easy to see that the security of the network was not its center point. As presented in chapter 3, many threats can follow the loss of data integrity for devices. Hence, their cyber protection is a real concern and integrity should be considered an important element of the security of 1553. However, detecting memory corruption should also be considered as part of a classical incident response plan recovery process. In [23], *Changalvala et al.* introduce the notion of trustworthiness of data in autonomous vehicles. Mentionning the five essential functions of autonomous cars (perception, localization, planning, control, and system management), they argue that if the sensors are corrupted or if their integrity is questionable then the car and its passengers are put at risk. This notion can then be transposed to aircraft using the 1553 protocol fairly easily, since their attack vectors and threats are similar. As such, they introduce a framework using a non-linear data hiding technique called QIM to assess the integrity of a LiDAR sensor by hiding a water marking the data. Using this approach they were able to detect and localize when the raw data of a sensor had been tampered. The method they

proposed in [23] achieved 100% detection and localization rate when the noise was less than the quantization step-size. They then furthered their research in [24] testing the attack vectors and introducing a spread dither 3D QIM which allowed them to test the aforementioned approach and examine how this new water marking affected the bit Error Rate. While this research thesis does not aim in integrating any sort of watermarking to the data, the notion of data integrity is approached in a similar fashion, where a memory mapping process fingerprints the usage of memory space.

2.2 ML classification based on communication

Classification of IoT devices is usually separated in two types based on the fingerprinting mechanism: Active or Passive. In [25], *Noguchi et al.* proposed a device identifier which can be applied to a variety of IOT devices based on the similarity of features. The authors used passive techniques to obtain packets from communication and exchanges to build their dataset. As it is common with standard world wide web (WWW) protocols such as TCP, the packets are wrapped by a header which can be exploited to extract device specific information. Other passive communication features such as periodicity, response time and used port number are leverage by [25] to classify the devices. The authors also used computed features such as maximum value, minimum value, average value to supplement their features. While it introduces an automatic feature selection method from the header information, [25] argues that hand picking features that are device specific such as frequency of transmission/reception would help in differentiating a device from another similar one and can increase accuracy. Similarly, literature [26] presents a technique to improve the identification accuracy of Support Vector Machine (SVM) algorithms using a cosine similarity factor. Employing a two-stage multi-classification algorithm based on One vs All SVM, [26] leverages the use of a series of statistical features, mainly the

sum, minimum, maximum, mean, variance and standard deviation of headers and payloads. Doing so the authors manage to obtain an average identification accuracy of 93.2%. It is to note that they also studied the relationship between identification accuracy and the number of packets which showed 15 packets as the bare minimum vector length to obtain an accuracy over 80%. In turn, this entices the use of multiple features for the present research. This is reinforced by [27], which used a multitude of features such as average inter-arrival times, average packet length, the minimum packet length, the maximum length, IP source address, IP destination address, source port number and destination port. They then applied a binary bag of words to other textual features extracted from the header before classification. From these it can be said that the use of minimum, maximum, sum and average are widely used features and are to be integrated in the present research.

In accordance with the above, in [28] *Sivanathan et al.* used average packet size and average rate of transfer during an 8 packet exchange to attempt classification. Using the KNN method they created a clustering algorithm that achieved high accuracy using only these two key attributes which were normalized and scaled using the Z-score method. This Z-score method is best suited for normalized data that are normally distributed, hence it might not be applicable for skewed, non-normal or non-symmetric data as may be present in 1553 devices. Regardless of the normalization mechanism [28] shows that KNN can achieve good accuracy when using a small amount of features that are correctly hand picked.

Alternatively, in 2019, *Aksoy et al.* presented in [29] an automated device classifier based on their network traffic. The outcome they called SysID fingerprints IoT devices using only a single packet and then classifies them using machine learning. Their feature selection process uses genetic algorithms to identify unique features which were collected in a passive manner. In this way, they collected packets from 23 IoT devices with 20 features each directly taken from the common protocol header

fields. The study they presented in [29] showed it was possible to obtain high accuracy even with as little as 50 samples per class.

While many researchers tend to take the passive approach the same cannot be said for the active approach. It is easy to see that if put to miss-use, an active fingerprinting can be perceived as a threat to the networks IDS, firewall, etc., much like the network scanner of Nmap. However, this is not an indication that active fingerprinting is ineffective. Rather, it shows that it must be use in a more coordinated manner with the network administrator, when dealing with standard network. When dealing with isolated embedded networks like 1553, it is safe to assume that active scanning can be used to obtain valid data with the proper authorization.

2.3 Fingerprinting

In [30] the work done by *Aneja et al.* presents the use of inter message arrival time (IAT) as a metric used in signature generation. Using the first 100 packets transmitted by the devices they then transposed the IAT in a graph which served as signature for the CNN they then used for classification. This CNN then classified devices based on the visual graph traced. Using only two devices they obtained an 86.7% accuracy rating. However, the only two devices used were from the same manufacturer and the IAT plotted graphs were also used to generate fake devices by artificially transforming the graphs. Hence the same device signatures were warped, stretched, inverted and zoomed in linear ways in order to supplement their data which might cause problems with different signatures that aren't as distinct as others. This in turn leaves a lot of latitude for experimentation and allows one to investigate how distinct the signature need to be to obtain adequate accuracy, which is precisely what this research thesis focuses on.

While dating back to 2013, the paper by *Beyah et al.* in [31] presents a signature

creation mechanism to be used within a passive device and device type classification using ANN. First, they extract a signature based upon IATs of the device and compare it to an existing list of known devices. Using a similarity rating the authors are then able to detect if this represents a new device and/or device type. If it is deemed a new signature, it is then added to the database which trains the feed-forward ANN using scaled conjugate gradient back-propagation. Using this technique they achieved an accuracy rating between 83% and 99% depending on the traffic type and whether the device was known or unknown.

Alternatively, the work presented by *Yang et al.* in [32] presents an automatic IoT device fingerprinting technique using active scanning. By sending query packets to a remote host, they were able to obtain responses containing enough information to be classified by a neural network. As presented in their work, they argue that the use of such application can help inventorize and manage the security of the devices in network. [32] argues that this method allows to find leaked information due to misconfiguration as well as identify if some devices with known vulnerabilities, associated with firmware version, are present in the network. Such techniques are also presented in the review article presented by [33] where they argue that passive fingerprinting might be concerning if security policies are to be enforced at run-time. They do, however, reinforce the fact that device specific timing analysis has yielded good results in device fingerprinting and ultimately classification.

2.4 Intrusion detection Systems

IDS are a common tool put in place to detect and respond to intrusions in a network. They come in many forms, whether automatic, manual, rule based, traffic based, etc., research in the IDS field has seen a boom with the soar of cybersecurity.

In 2019, *Mishra et al.* used machine learning techniques to evaluate their application to IDS. They reviewed the effectiveness of many algorithm including ANN, KNN, SVM, etc., when detecting common cyber attacks like DoS, probing attacks and user to root attacks. As presented in [34], they also evaluated the effectiveness of the choice of features and combination of algorithms to achieve better accuracy. Hence, using the Scikit-Learn, TensorFlow, Weka and other environment to extract their features, they affirm that combining multiple task specific algorithms with less individual features will yield better result than a single algorithm that is given more features. Assuming the results presented in [34], its possible to theorize that classification of devices whether IoT or 1553 should be done per device type independently from version. Furthermore, *Mishra et al.* showed that, when used in combination with other algorithms, even simple ANN can outperform other algorithms.

Alternatively, *Chockwnich et al.* in [35] used deep learning, the TensorFlow application and the Keras Application Programming Interface (API) to study if recurrent neural network (RNN), stacked RNN and CNN could be used to classify attack types without information from the packet payload. When comparing their method, which reached accuracy nearing 100% regardless of the algorithm, with other IDS such as snort, they noted a drastic improvement both in precision and computing resources utilization; Snort obtaining 47% accuracy and using 2.7% of the CPU compared to 0.1% while testing the ML algorithms [35]. This paper's approach gives guidance to the present research about what platform to use but also that deep neural networks yields better accuracy when dealing with a dataset with many features.

In 2019, the department of National Defence and the Royal Military College of Canada released a thesis called "MAIDENS: MIL-STD-1553 Anomaly-Based Intrusion Detection System Using Time-Based Histogram Comparison". While [21], is not a classification process, it utilizes key metrics of MIL-STD-1553 to achieve intrusion detection. Furthermore, it showed that time-based intrusion detection is well suited

for technologies such as the MIL-STD-1553 attaining 100% detection of intrusions. Hence, MAIDENS points to a high probability for timings to be exploited as a classification feature for 1553 devices.

2.5 MIL-STD-1553B Problem Specific literature

A challenge that present's itself when dealing with atypical protocols and devices such as 1553 is the lack of known classes. Hence, in the application of the present research, its likely that the classification algorithm will face unknown devices during development. Consequently, there is a need to deal with unknown classes. [36] presents an approach using Class Discovery Tree, a data structure algorithm closely related to decision tree. Using this technique they were able to estimate queue time with an accuracy between 95.43% and 72.98% for # of ranges from 3.6 to 12 compared to a Naive Bayes technique which achieved accuracy between 89.76% and 64.28% for known and unknown device classes.

As for dealing with high numbers of classes, literature [37] found a useful property of decision tree learners which allowed them to obtain a higher accuracy and smaller tree size the more classes the learner was given. Using a grouping technique they observed a reduction of the error rate between 2-5 % by having more specific classes, leading to more total classes, rather than larger classes. Alternatively, [38] proposes a simple, scalable method to achieve a reliable confidence score using either distance based or adversarial training. Using these techniques, combined with a confidence score the authors were able to obtain an accuracy over 90%.

While these topics will have an effect on the direction on the present research they are topics that should be integrated in future work as the present research aims to create the stepping stone of a 1553 alternative for Nmap.

Chapter 3

Background and Methodology

3.1 MIL-STD-1553B

Since its installment in 1973, the MIL-STD-1553 has become a cornerstone of aircraft and aerospace engineering. With over 30,000 aircraft (e.g, F-16, Aurora, F-35) and implementing almost 1 million 1553 terminals [39], MIL-STD-1553 is a standard used in aircraft in Canada and other countries that rely on legacy aircraft technology. Regarded as reliable and easy to attain its specifications, 1553 is a 1 megabit-per-second serial physical layer and messaging protocol that uses half duplex communication encoded with Manchester II Bi-Phase (Non-Return to Zero, NRZ) in a time division multiplexed manner. Furthermore, MIL-STD-1553 is schedule based on a clock bound manner, meaning it executes a list of pre-determined tasks in a cyclic manner. Using the communications requirements presented in the designers guide [2] will allow an aircraft to have a bit error rate as low as 10^{-12} , hence minimising the bandwidth used for re-transmission. A typical 1553 aircraft network is comprised of 3 different component type: BC, BM, RT. As illustrated in figure 3.1, a standard 1553 infrastructure contains at least one of each of these device type, but a typical aircraft will comprise multiple RTs, 1 main BC broadcasting on either of the two data bus, 1 fail-safe replacement BC and 1 monitoring BM.

Figure 3.1: 1553 Sample MIL-STD-1553B Multiplex Data Bus Architecture [1]

BC: The Bus Controller serves as the main controller for all communications, it tasks every other system and is the sole component that can initiate data transfers in the network. Upon configuration, every critical RT is given a frequency at which it will be probed by the BC from the BC schedule. As such, there is very little time for asynchronous messaging to occur outside of this schedule. While, many components can have some BC capabilities, only 1 BC can exist at a time on a 1553 network. In turn, as mentioned in chapter 2, this can lead to DoS where a rogue device acts as the BC by sending command words in place of the real BC.

BM: The Bus Monitor is a passive device that serves as a black box for the aircraft, it records all transmitted information through the data bus. These exchanges and recordings can then be leveraged to create a dataset as well as observe the state of the system.

RT: The Remote Terminals and subsystems are computers or components that require information from other RTs, aircraft systems, sensors or are autonomous and need to provide information back to other RTs. Every subsystem of a 1553 aircraft is connected to the databus through an RT. These RTs include but are

not limited to the controls, radios, radar, global positioning system, graphical user interface, weapon systems, interface between different protocol devices, etc.

In standard 1553 communications there is only 3 types of messages that can be transmitted: Command, Status and Data words. Figure 3.2 presents the bit usage for command words, figure 3.3 status words and figure 3.4 data words, while figure 3.5 presents usual message exchanges on a 1553 databus.

Figure 3.2: 1553 MIL-STD-1553B Command Word bit usage [1]

Figure 3.3: 1553 MIL-STD-1553B Status Word bit usage [1]

Figure 3.4: 1553 MIL-STD-1553B Data Word bit usage [1]

Figure 3.5: 1553 Information Transfer Formats [2]

Each data transfer made through the Command word originates from the BC regardless in which one of the four formats it be: BC-RT/RT-BC, RT-RT, mode code transmission, and broadcast transmission. The BC-RT format represents the BC instructing the RT to receive a set of data. The RT-BC format represents the BC instructing the RT to transfer a certain amount of data. The RT-RT format is to initiate the data transfer from one RT to another. When a command word is sent from the BC to an RT, it can not only be either receive or transfer, but also be a mode code transmission or a broadcasted command. The broadcast command can be used with RT addresses 0x00 or 0x1F (31), which will broadcast a single command to all RTs, while suppressing the need for a status word response. Mode codes are a specific type of Command word. They can be sent by the BC to a/all RT by setting the sub-address (SA) field to either 0x00 or 0x1F (31) and by setting the word count to the desired Mode code. This allows the BC to send specific command words as presented in figure 3.6. The most useful Mode codes in the scope of this research are: 0x02(2)-Transmit Status words, 0x03(3)-Initiate

self test, 0x08(8)-Reset RT and 0x13(19)-Transmit Built-in Test Word. Hence, a standard command word will be constructed as follow, 5bits for the RT address, 1bit for transmit or receive (depending on the command word), 5bits of the SA set to either 0x0 or 0x1F (meaning command word) and 5bits to choose the command word. In the scope of this research, these command words are especially important as they will differ from device to device and might help in the classification process. Timing wise, all 1553 command and status words are 20 μ S, at a rate of 1 bit/ μ S.

T/R Bit	Mode Code	Function	Assoc. Data Word	Broadcast Command Allowed
1	00000 (0)	Dynamic Bus Control	No	No
1	00001 (1)	Synchronize (without Data)	No	Yes
1	00010 (2)	Transmit Status Word	No	No
1	00011 (3)	Initiate Self-Test	No	Yes
1	00100 (4)	Transmitter Shutdown	No	Yes
1	00101 (5)	Override Transmitter Shutdown	No	Yes
1	00110 (6)	Inhibit Terminal Flag Bit	No	Yes
1	00111 (7)	Override Inhibit Terminal Flag Bit	No	Yes
1	01000 (8)	Reset Remote Terminal	No	Yes
1	01001 (9)	Reserved	No	TBD
.	.	.	.	.
1	01111 (15)	Reserved	No	TBD
1	10000 (16)	Transmit Vector Word	Yes	No
0	10001 (17)	Synchronize (with Data)	Yes	Yes
1	10010 (18)	Transmit Last Command	Yes	No
1	10011 (19)	Transmit BIT Word	Yes	No
0	10100 (20)	Selected Transmitter Shutdown	Yes	Yes
0	10101 (21)	Override Selected Transmitter Shutdown	Yes	Yes
0 or 1	10110 (22)	Reserved	Yes	TBD
.	.	.	.	.
0 or 1	11111 (31)	Reserved	Yes	TBD

Figure 3.6: 1553 MIL-STD-1553B Mode Codes [1]

After receiving a message, a RT has up to 12 μ S to send an initial response message before being labeled late or else the BC will timeout at 14 μ S. Additionally, there is a minimum requirement of 4 μ S delay between all message. This means that the status words following each command word sent from an RT has between 4 μ S and 12 μ S to reach the BC. Furthermore, 1553 is scheduled based and much of the devices that are used on a 1553 databus must be in constant communication with other devices to operate properly. These devices are given a frequency at which they will appear in

the schedule. Which is why there is exhaustive research on 1553 intrusion detection systems using real-time communication as presented in [4],[21],[40].

There is, However, some scenarios that will lead to a lack of response or appropriate response without there being any mistakes in the communications. Such scenarios include the RT being busy or unavailable. In this case, when the RT receives a command, rather than answering with a status word showcasing the effects of the command word, it will send a status word with the "BUSY" flag set to the BC. Since the BC is schedule bound, the presence of "BUSY" RTs doesn't stop the execution of the BC agenda. The BC will hence continue with its next valid operation, assuming the information in the previous message wasn't necessary for the next message. If it was necessary, then a branching path will be taken to return to normal execution and the message that failed will be tried again in the next iteration of the schedule. This "BUSY" flag is especially interesting when combined with specific Mode Codes. In the case of this research, if an RT is executing a task such as a reset or built-in tests (BIT) and is probed, it will send back a status word with the "BUSY" flag set and the BC keeps executing its schedule normally. This means it is possible to measure execution time by continuously probing an RT until a message that is different than "BUSY" is obtained. Therefore, while IATs are not directly obtainable in 1553, it is possible to obtain device specific times using command words.

3.2 Cyber Attack Techniques

As mentioned in chapter 1, cyber attacks come in many forms and leverage a wide variety of vectors and threat agents. Additionally, they will have different impact or threats depending on the type of attack. As research in the field of cybersecurity for the 1553 infrastructure is limited, following is a list of vectors and threats that apply to 1553 as presented in [4], [21] and [40].

3.2.1 Attack Vectors

Attack vectors are defined as a path or mean for a hacker to penetrate a system. Most attack vectors will require physical access to the databus or devices that can remotely interact with the system. This can be achieved either by accessing the aircraft or its component. Remote access and social engineering can be used as means to attain the vectors. Currently three attack vectors are considered possible within the 1553 aircraft as presented in [4] and [8]:

- Data injection: The act of injecting fake or corrupted data into RTs through the use of rogue or tampered subsystems, sensors or devices that maintain a communication link with an outside network.
- Code injection and manipulation: The act of inserting malware or changing the software contents of an RT or component of an RT. This can be achieved in any of the phase of the component's life cycle, i.e development, maintenance and deployment, by the use of software updates, external devices, compact disks, diagnostic tools, USB devices, etc. [4].
- Physical tampering: The act of physically accessing devices or components attached to the 1553 databus in an effort to change the wiring and electronics to cause failure or unavailability.

3.2.2 Threats

While threats will differ for each aircraft, there is a few attack scenarios that are relevant to infrastructure itself. In [4] and [41], the authors argue that 3 types of threats are more likely, given the lack of encryption and lack of authentication process.

- Denial of service: The act of stopping a certain device from being available. This can be achieve by stopping the communication or by flooding the network,

causing errors. As the BC is the central command for all communication on 1553, it opens the possibility for a spoofed BC. Since there's nothing that separates a BC from an RT other than its behavior, a rogue agent could introduce a fake BC and high-jack the databus. Alternatively, a rogue RT could send messages each time it detects a message from the BC hence causing collisions and errors on the RTs response.

- Data leaks: Since the information is freely propagated through the databus, its is possible for a rogue RT to record all the traffic and possibly obtain clear text information from it. Provided an attacker has inside knowledge about an aircraft's configuration, a compromised system could also access information from other RTs by manipulating messages to control the routing as well as transmitting sensitive data [4].
- Integrity violation: Insertion of invalid or incorrect data into a component leading to its failure or inability to operate correctly. This data could be inserted via an internal component like an RT, a sensor, USB device, etc., but it could also be the result of manipulation of an insider threat like a rogue technician physically tampering with a device. As mentioned in [42], the presence of malicious data transmitted by a threat agent or within a trusted component can lead the system to fail, behave abnormally or operate incorrectly.

3.3 Classification algorithms

Throughout this thesis multiple algorithms were used and compared. While they all have their own quirks, they have also been proven to be effective classification tools for IoT devices. Since, 1553 RTs are not IoT, it is expected that the accuracy will be impacted. However, since the RTs behave in similar fashion to IoT devices,

operating on a network based on communication exchanges, it is expected they will yield comparable accuracy to other research in which the following algorithms have been used:

- ANN: Artificial neural network Neural networks are a way to interconnect inputs by giving each a weight and bias, then propagating the information to the outputs. It can be used both for supervised and unsupervised learning but its supervised learning capabilities make it a suitable candidate for pattern recognition [43]. We chose this algorithm due to its simple setup and usage as well as its proven accuracy as documented in [44] where it achieved 96.5% accuracy in detecting IoT based communication attacks. This algorithm was also used in [34] where it achieved the best detection rate with high number of features.
- SVC: Support vector Classification SVC which is a branch of Support Vector Machines (SVM), has good effectiveness when dealing with a high number of dimensions and is a versatile algorithm that uses the one-vs-one classification mechanism. Due to its classification method, which is based on vectors, this supervised learning method has no built-in probability estimates and must use external tools to output scores and probabilities[45]. It can classify groups by distinguishing one from the other by using dimensionality reduction. While SVC can be inaccurate compared to other models, in some specific cases, such as seen in [26], SVC can be used to achieve 90% in fingerprinting IoT devices and classifying them.
- MLP: Multi layer perceptron The MLP is a supervised learning subset of neural networks that is trained by using back propagation. Doing so allows the MLP to accurately approximate non-linear functions. In turn, MLP does have some

issue with attaining the best possible accuracy due to its non convex loss function, which requires to test with different testing data to attain the real loss minimum and not just the local minimum [46]. In their research [44], Fatayer et al. also compared the effectiveness of MLP to ANN finding it to execute more slowly but having better accuracy at 99.6%. We chose to include MLP to examine if we could replicate the same results.

- **CNN:** Convolution neural network CNN is a partially connected feed forward neural network. It leverages simple patterns and filters in order to create a bigger and more complex composite of information which is then passed to the decision algorithm. Hence, CNN are traditionally used in image recognition and anomaly detection [47]. While being more complicated to configure, CNN can effectively be used to classify image like matrices to pickup patterns in IoT communication. Based on the work of Aneja et al. in [30], which looked at the IAT of Iot devices, it is expected to obtain accuracy ratings comparable to the 86.7% obtain in [30].
- **KNN:** K nearest neighbour KNN is a simple classification algorithm that bases its prediction on the surrounding data points. Due its need for spatial proximity KNN can fall prone to the so-called "curse of dimensionality" when dealing with high-dimensional datasets. As such, KNN usually requires dimension reductions through the use of feature selection in order to get neighbours that have a meaningful distance to the predicted data. The disposition, and training data can greatly impact the results as such it's important to choose the best possible scenario [48]. In this sense, Yubo et al. in [26] did present KNN as a comparative to SVC where it obtained an 89% accuracy in classifying IoT devices.

3.4 Methodology

With these in mind, OMAP proposes a classification ANN using active probing based on memory mapping and device specific timings. The goal of OMAP is to probe each device, map its memory content and location, measure the time taken to execute standard tasks and presenting them in a matrix fit for pattern recognition based ANN. The following section describes the MIL-STD-1553 standard and its key components that impact the data acquisition process of OMAP.

The present work uses the tensorflow package used within the Google Collab environment to achieve classification using ANN for device type classification and KNN for version classification. We then compared the obtained results with other classification algorithm in order to leverage the best possible accuracy for further implementation of OMAP. Using the Google Collab environment, the OMAP-Parser and OMAP-Classificator were both written using Python 3.6.7. The BUSTOOLS API was then used to create the OMAP probe, which is a C++ probing script that effectively gathers the data from the RTs. The obtained data was examined to find the boundaries of the data as well as map their contents. It was then supplemented by an emulation of RT data based on these boundaries. The boundaries of the RT radio and example RTs of the BUSTOOLS were used as guidelines to supplement the dataset to obtain a dataset with 5 device types with multiple version each and at least 100 entries for each.

Chapter 4

OMAP

4.1 Feature Selection

The process of choosing features can be done in many ways. Whether a research uses genetic algorithm such as presented in [29] or hand picks features such as in [25] comes down to whether or not the effectiveness of features is known or not. In the case of 1553, *Losier et al.* in [40] and *Lai et al.* in [21] have proven that time-based features can effectively be used for detecting abnormal traffic and intrusions. Using these past research as a starting point, it is safe to assume that these time-based features could effectively be used for classification as well. The following table displays examples of messages and commands as seen from the BM which will help in illustrating what information can be leveraged from 1553 communications.

As can be seen from the message types example table 4.1, unlike UDP and TCP protocols, 1553 messages contain a relatively scarce amount of information. Each message starts with the number of the message, followed by the timestamp. Following those two is a description of the message, containing the used databus, whether or not the message is broadcasted to all RTs, the inter message gap, the command to be executed (comprised of: destination, transmit/receive, destination SA, number of words), the response time and status followed by the transferred data if any. It is

Broadcast Command Word	Msg:1 Time Tag:(162)17:04:22.699342 Bus:A Mode Code BROADCAST Gap:16 us Cmd=03-TX-31-08 <FFE8> Reset RT Message Time: 18 us
Command Word	Msg:12 Time Tag:(162)17:04:22.699342 Bus:A Mode Code Gap:16 us Cmd=03-TX-00-03 <1C03> Init. Self Test Rsp=5.0 us Sts=<0800> Message Time: 41 us
RT->BC Message	Msg:6 Time Tag:(1/1)00:09:07.638825 Bus:A RT->BC Gap:16 us Cmd=01-TX-30-32 <0FC0> Rsp=5.0 us Sts=<0800> 0001 0000 ffff 0000 0000 0000 0000 0000 0000 0000 0000 0000 Message Time: 681 us

Table 4.1: 1553 Message Types examples

also possible to leverage the timestamps of specific command words to obtain further information from these messages. The table of exploitable timing features describes these features.

Response Time (Rsp)	Elapsed time between the mid point of the synchronise signal of the BC request and synchronise signal of the response.
Inter-Message Gap (Gap)	Time period between the Synchronise signal of the final status word closing a response and the next command to be sent by the BC.
Data Throughput	Total amount of data words sent during a 50ms. time-frame.
Bus Utilization	Percentage of the time that the databus is being used during a 50ms time-frame.
Periodicity	Average time between each BC message to a single RT.
Reset Time	Time taken for a single RT to execute a reset. Smaller than 500ms as per [1].
Built-in Test Time	Time taken for a single RT to execute all its built-in tests. Smaller than 500ms as per [1].

Table 4.2: 1553 exploitable Timings

While some of the timings may be used for intrusion detection they may not be suitable for classification. As IDS use the passive detection approach they can leverage periodicity, throughput and bus utilization more effectively than an active approach. Hence, since this work aims to use active probing of RTs, we must find standard messages which comprise timing features that can be exploited for classification. The following distinguishes between the available features and explain why each features has been chosen.

- Response time: This period of time is the time taken by the RT to process the

request and transfer the information back to the BC. For a transmit command without a specified length, the RT will send the complete memory space (32 x 2bytes) of the targeted RT. Furthermore, the response time obtained using Bustools represents the total time taken to send all 64 bytes. While it currently seems impossible to get the IAT of those messages the transmission speed should be a good indicator of the technology that is used.

- Restart time: This period of time is obtained by transmitting a reset command word to an RT. However, 1553 will not allow the RT to send any response when it becomes operational after the reset. To circumvent this, a succession of status request command words are sent to the RT while it is still resetting. This allows us to measure the reset time by using the time tag present in both the initiating message and the first status word that isn't "BUSY". Unfortunately, this could not be done in an experimental manner due to the experimental setup that used emulation and device simulation. However, it is safe to assume that it is possible to measure the restart time. The granularity of the time measure could easily be adjusted to obtain significant difference in restart times by adjusting the BC message time cycle down to 4 μ S, which is the minimal value of the internal clock of the BC, which governs the time between periodic messages like the ones OMAP is dealing with.
- Built-in test time and word: By using the command word 3 (initiate self test) and 18 (transmit built-in test word) the BC can obtain device and manufacturer specific data. As no standard is in place for these self tests, the manufacturer is in full control of the RTs behavior. In turn, this leads to different execution time from device to device as well as different software version of a single device. Furthermore, the built-in test word is a reflection of the test results. Hence, for a same RT it is expected that different software versions will yield different

built-in test word. These differences, both in value and execution time, may prove to be key elements of the classification, the same way IAT are used in the IoT device classification presented in [30].

- Active and inactive SA: By requesting the transmission of a memory space to an RT, it is possible to assess whether it is a used or unused sub-addresses as used sub-addresses will typically have values that differentiate from the NULL value, ergo 0, while an unused sub-addresses will only contain zeroes in its memory. Typically, an RT is designed to have only specific sub-addresses that are active, though nothing is stopping the manufacturer from internally using sub-addresses despite the BC never using it. These can then be used as key metrics in identifying a specific device.
- Memory anomalies: As a difference rises between the data-sheet of an RT and the way it truly executes, memory anomalies shows. These anomalies are the manifestation of the manufacturer in the memory space. While normal utilization of an RT may lead to identifying which sub-addresses are used following the data-sheet, actively probing the entire memory space of an RT might lead to discovering memory spaces that aren't being used according to the data-sheet but that still contain information. This information may be there for internal RT calculations, branding or simply as an artefact. While these anomalies cannot be deemed as such without the use of the data-sheet, they can nevertheless be used to differentiate between two RTs and possibly their versions as well. The presence of such memory anomaly in an RT that doesn't usually have any might indicate the presence of integrity issues with the RT.

4.2 OMAP classification process

The classification process that is done by the presented framework, OMAP, is a multi-step approach. First, the raw data must be obtained from the BM by the use of a custom BC. This step is called the "OMAP-Probe" as it mimics an adversarial probing attack but with legitimate intentions. Then follows the "OMAP-Parser", which parses the responses for data. Finally, the "OMAP-classifier" which predicts the type and version of an RT. This multi-step approach can be observed in the flow chart presented in figure 4.1.

Figure 4.1: OMAP flow chart

The "OMAP-Probe" is effectively an initialized BC that uses one of the two available databus in order to query each and every RT. The queries are the same for every RT but their response is what is monitored. The queries themselves, that are then repeated for all RTs, are as follows:

- Initiate built-in test request: This will send the request from the BC to a single RT causing it to initiate the BIT sequence. During this period of time, which

can last up to 500ms, a series of status word request are sent. While it is self-testing an RT can either send a "Busy" status word, no response or a valid status word when completed. The BC continues to query for the full period of 500ms as it has no way to distinguish the responses. This task falls under the "OMAP-Parser's" duty. After the full 500ms the BC tasks the RT to transfer the BIT word for monitoring.

- Sub-Addresses' content requests: After having completed its tests, the RT is then ready to be queried and the BC will request the contents of all Sub-Addresses one by one. The response from the RT can either be a valid data-word, indicating an active SA, a lack of response or a status word with errors set both indicating an inactive SA.
- Initiate reset request: Much like the BIT request the reset request tasks the RT to reset. The RT then has 500ms to complete this task. As the BC continues executing during this reset, the RT will not send any response to say it is now available. Instead, like during the BIT request, the BC will continuously request a status word from the RT for the whole period of time. The parser will then be able to find the time taken for the reset by examining which status word first showed signs of being available.

The pseudo code that is presented in Algorithm 1 is the representation of the above taken steps from the "OMAP-Probe".

To use our "OMAP-Probe" we must use the BC to send our probing messages. This means either replacing the task list of the BC entirely or integrating the probing sequence into the original BC schedule. Effectively, this means probing the RTs while the aircraft is either "Online" or "Offline". "Online" means that we must find moments when the data bus is not already in use to actively probe RTs through the use of what is known as asynchronous messages. These messages are sent within gaps in

Algorithm 1: OMAP-Probe

Result: Initialize the BC of a 1553 network as well as establish the message schedule (BIT,Memory Content,Reset)

Setup *BC*;

Setup *BC* message interrupt, To stop the cyclic schedule;

for *RT* in range 32 **do**

 Transmit "command word: initiate built-in test command" to *RT*;

for *Timer* in range *Threshold* **do**

 └ Transmit "command word: send status word" to *RT*;

for *Subadr* in range 32 **do**

 └ Transmit "send: contents of SA[*Subadr*]" to *RT*;

 Transmit "command word: reset" to *RT*;

for *Timer* in range *Threshold* **do**

 └ Transmit "command word: send status word" to *RT*;

the cyclic schedule of the BC and their response must not interfere with normal message exchange. By doing "Online" probing, real-time protection is enabled. On the other hand, "Offline" probing allows the user to replace the BC schedule completely and as such does not require thorough study of each aircraft type BC schedule. Doing so allows us to use the BC as the probing mechanism. By replacing the BC schedule, an "Offline" approach can execute its probing much faster and without concern to the overheads impact on normal execution compared to the "Online" approach. Effectively, this means that we cannot send probing messages asynchronously in time gaps smaller than the time taken to send a probe and receive a response, since failure to send information to an RT, which requires data every fixed amount of time, might cause it behave erratically. While response time and message time depend on the contents, they also directly impact the speed at which the BC can send out messages due to response overhead. Additionally, for each probing message the ensued response will vary in its response and incidentally in the time it takes to send the response message. While the status-word can be sent to the BC by the RT with a 20 bit overhead, 32 data-words need to be sent entirely to transmit the contents of

an RTs single SA, meaning over 640 bit overhead per SA probe, hence a total of 960 total messages or 19200 overhead bits, only for data contents. Essentially, these 32 data-words are the same length as all other messages (20 bits each). However, due to the sheer number of messages that are to be sent to complete a probing sequence, it would be difficult to obtain all 32 data-words consecutively in an "Online" approach. These 32 data-words would need to be separated into 32 individual messages instead making the probing sequence even longer. To counter this overhead and maintain optimal probing time, we agglomerate messages with significant overhead together as is displayed in the "OMAP-Probe" algorithm. By doing so, we can have low overhead for Reset and BIT probing allowing us to have high probing frequency while having high overhead for data transfer where probing frequency won't be as impactful. This is however not possible in the "Online" scenario since only a few milliseconds might be available per cycle of the BC task list. Considering the overhead the probing process puts onto the BC, an "Online" approach is impractical, which is why we chose an "Offline" approach.

The second step in the OMAP classification process is the "OMAP-Parser". This parser collects the raw data created by the "OMAP-Probe" from the BM present in the 1553 network. It then goes through the entire messages exchange in order to extract the appropriate data. It then takes said data and shapes it into matrices which can then be fed to the "OMAP-Classificator" for evaluation. Mainly, the "OMAP-Parser" is looking for active sub-addresses of an RT and its contents. It can also measure reset time and BIT time by using the timestamps present in all messages. It parses the data for the initial command word, then looks for an appropriate response and then measures the elapsed time. The process is detailed in the pseudo code of Algorithm 2 below.

As Algorithm 2 presents, the "OMAP-Parser" goes through the whole message stream and relies on the fact that the "OMAP-Probe" follows a specific probing

Algorithm 2: OMAP-Parser

Result: Converts string vector of the probe into usable dataset for NN classification. Measures elapsed time for reset and BIT self test. Converts memory transfer into usable integers.

```

Initialize MsgList, Collected messages from the OMAP-Probe;
Initialize Datamap(32, 32, 32), Device Data of network for classification;
Initialize RESETtimelist(32), Reset times list for all 32 addresses;
Initialize BITtimelist(32), BIT times list for all 32 addresses;
for msg in MsgList do
    CMDindex = Find "Cmd" in msg;
    CMDword = msg[CMDindex + 4 : CMDindex + 15];
    RTadr = CMDword[0 : 2];
    RTsubadr = CMDword[6 : 8];
    CMD = CMDword[9 : 11];
    DATAindexbegin = Find "Sts" in msg;
    DATAindexend = Find "Message Time" in msg;
    if DATAindexbegin > 0 then
        DATA = msg[DATAindexbegin + 10 : DATAindexend];
        DATAlist = split DATA every " " and remove " ";
        for position in DATAlist do
            Datamap[RTadr, RTsubadr, position] = DATAlist[position];
        end
    end
    if CMD = 'Initiate self test' & First for RT;
    then
        BIT = TRUE;
        BITtime = 0;
        MSGtimebegin = Find "Time Tag:" in msg;
        MSGtimeend = Find "BUS:" in msg;
        TimeTag1 = msg[MSGtimebegin + 5 : MSGtimeend];
    else
        CMD = 'transmit Status word' & BIT = TRUE;
    end
    if Status != "BUSY" & Status != "NO RESPONSE" then
        BIT = FALSE;
        TimeTag2 = msg[MSGtimebegin + 5 : MSGtimeend];
        BITtimelist[RTadr] = TimeTag2 - TimeTag1;
    end
    if CMD = 'reset' then
        reset = TRUE;
        BITtime = 0;
        MSGtimebegin = Find "Time Tag:" in msg;
        MSGtimeend = Find "BUS:" in msg;
        TimeTag1 = msg[MSGtimebegin + 5 : MSGtimeend];
    end
    if CMD = 'transmit Status word' then
        if Status != "BUSY" & Status != "NO RESPONSE";
        then
            reset = FALSE;
            TimeTag2 = msg[MSGtimebegin + 5 : MSGtimeend];
            RESETtimelist[RTadr] = TimeTag2 - TimeTag1;
        end
    end
end
end

```

pattern. This pattern, is "Self-test" then "Memory contents" then "Reset". The "OMAP-Parser" looks for the RT address first and then for the bit pattern meaning "Initiate BIT". At that point, it stores the begin of the BIT time by positioning its cursor at the correct index. From there, it looks for the next message that will not contain BUSY. When found, it measures and stores the BIT time. After obtaining the BIT time, the "OMAP-Parser", parses the message vector for a "RT-BC" message with content. It stores all subsequent content from "RT-BC" messages within a memory content matrix before calculating the Reset time in a similar fashion as the BIT time. Having now 2 timing features and 960 memory content features, it converts them into a coma separated values dataset for classification. The dataset that is obtained from the parser is then passed to the classification algorithm. During the course of this research project, multiple avenues were explored to classify the device and their version. This process is described in section 4.3.

4.3 Device and Version Identification

As presented in [37] there is an argument to use more classes when using decisions trees. Alternatively [49], argues that the more classes are present when using supervised learning the more the accuracy will drop. It is hence necessary to look at the two possible solutions and adapt the classification to which approach will yield better results. As such, the present thesis examined the accuracy of a single ANN classifying all device with different version as different class, compared to a multistage classification process using an ANN to classify device type and then a KNN to classify the version. The second approach also has a second layer of redundancy against spoofing. Having the possibility to separate the features and use them to determine whether an RT is spoofed twice might end up being beneficial for the security of a 1553 network. However, adding a spoofed class for every device will result in a rapidly expanding

number of device class when dealing with real devices. Hence, it might be possible to regroup all types of spoofing within a single class. The next chapter will discuss this in more details.

For this research thesis, an ANN image classification algorithm is used. Once the features are extracted, they are converted to an image matrix before being passed to the classification ANN. The conversion is done by bringing all values to a base 255 such as grey-scale pixel values. As mentioned above, each one of the 30 sub-addresses contains 32 memory spaces each with 2 bytes worth of information. By considering each memory space has a maximum value of 65535 (0xFFFF), by dividing each value by 257 before, it is possible to bring the values of each memory space between 0 and 255, the default values of a gray-scale image. These new values are then disposed as presented in figure 4.2 to obtain a visual representation of the information contained within the RT. As for time values, they are given their own respective row each for later usage. The chosen ANN is then trained upon a fraction of these RT fingerprints before being tested. For the purpose of this study, a 0.67/0.33 training/testing ratio has been chosen as per customary values in Neural Network research. Furthermore, considering the size of the matrix passed to the ANN (30x32), a deeper network has been favored as it yielded better experimental results which is reinforced by the work presented in [35]. The chosen ANN is a network comprised of 5 layers, 3 of which are hidden layers. The first layer is 963 nodes for inputs. 960 of these inputs are the values contained within an RT's SA (32 values for 30 SA, since SA 0 and 31 are broadcasts). The other 3 inputs are the mean, maximum and minimal values contained within the SAs. The second layer which is hidden is a reduction to 600 nodes using the rectified linear activation function, since it is a computationally efficient activation function. The third layer is reducing the nodes even further to 300 nodes also using rectified linear activation function. The fourth layer is the same as the other hidden but with 100 nodes. The final layer is the output layer which uses a soft maximum

activation to output the predicted class. To achieve classification the data was scaled linearly between 0 and 1, hence timings and data values were scaled independently as their max values are different. While Z-score might have been used to scale the data like in [28], the non normal distribution of data present in the device would render this scaling less than effective. Furthermore, the network was trained with a series of different training set. By doing so it was possible to only keep the model which yielded better accuracy throughout the iterations for all 16 scenarios that are described in chapter 5.

Version classification

Alternatively, rather than classifying devices and versions at once, it could be done in a two step process as presented in section 5.3.1, where the device type is predicted first, followed by a type specific version prediction which can be done through a different classifier such as the KNN algorithm. Through the same iterative process as the ANN this second classifier was also subjected to different training sets. Furthermore, the number of neighbours was also iterated to see what the best K value was. From experimentation, 11 was the maximum value for K that yielded a best accuracy. When dealing with up to 39 neighbours, by using only odd numbers, the accuracy would sometimes be over 90% but it never yielded the best accuracy for the loop. Alternatively, low number of neighbours gave much better accuracy ratings with 1, 3 and 5 being the top contenders for best accuracy. Since a single neighbour puts a lot of stress and uncertainty on the choice if an outlier is present, the final choice of K was put at 3. It is expected that as the number of device classes expands through further research this value will need to be adjusted. Furthermore, due to having a high number of features the reduction of dimension is necessary when trying to distinguish between versions. Additionally, since the memory content of RTs should not have much variation between versions, timing features should be favored as features

Figure 4.2: Example of RT memory contents

for version classification. Combined with an unknown number of version for a single device type, KNN was favored over SVC due to its simplicity of interaction with previous factors, though section 5 will show that SVC should be favored for its accuracy. By separating type and version classification, different features are leveraged to obtain the best possible accuracy for each specific classification step. This two step classification process can also add redundancy for anomaly detection. Doing so does, however, require the creation of a version classifier for each RT type, which is selected by the prediction of the RT type.

Chapter 5

Experimental Evaluation

5.1 Dataset

As 1553 devices are typically used in the military and avionics sphere their devices are not readily available. However, in partnership with the Royal Military College of Canada, it was possible to experiment with simulated devices through the Bustools software by GE. This software in use by the Royal Military College presented simulated devices each with differences in their used memory space and boundaries for the stored values. Hence, the data set includes copies of those simulated devices in which values have been randomized within their respective boundaries. Data has also been transposed and stretched in a similar fashion presented in [30]. On top of those manipulation, a standard error deviation was also added to each time value. In total, five device types were used with multiple different version for each type. The resulting dataset is 100 emulations of each RT memory contents and measured time features. Considering all versions, 11 different devices are contained within the dataset. As the simulated devices relied on the same computer for device specific times, further steps were taken to observe the time features effect on the classification behaviour. By creating more versions with different deltas added or subtracted to timing features as well as adjusting the similarity of both BIT times and reset times it was possible

to test the extent of the effectiveness of the classification algorithm. The result is 16 datasets containing manipulated time features to be used for version classification.

Two different factors were modified to obtain these datasets, D for Delta value and R for Resemblance. The delta value ranges from 1 to 12 in four steps distributed in an arbitrary manner (1,3,6,12), keeping in mind that the precision of the Reset and BIT time depends on the messaging speed of the BC, which can be adjusted. These value ranges allow us to artificially change the timings by adjusting their values. It is to note that while real devices may lead to real time values, timing bins were considered for this study for a matter of simplicity. These timing bins also correlate with the work done in [21], which regrouped their timings in bins prior to fingerprinting. Nevertheless, the delta was added to observe just how much the impact of the added uncertainty to the time features will have. On the other hand, Resemblance is an indicator of similarity between each version of these times and its spoofed counterpart. Four different cases were studied which were:

- R0 - BIT times and Reset times are similar regardless of version for legitimate and spoofed;
- R1 - BIT times are similar, Reset times aren't;
- R2 - BIT times and Reset times are both different;
- R3 - Reset times are similar, BIT times aren't.

These four different resemblance scenarios serve the purpose of counterbalancing the lack of more real devices. While it wasn't possible to fingerprint a wide variety of devices, adding these four scenarios allowed the present thesis to observe the impact of similarity for the classification algorithm. This allowed to artificially test the effectiveness of the proposed approach. The delta is a measure to determine how different the measurements should be to obtain a good accuracy for predictions. Together they

Figure 5.1: Experimental setup of 1553 databus network

create 16 case studies that will allow the research to mimic possible outcomes if more real devices were available for testing.

5.2 Experiment setup

To test this proposed approach a standard 1553 network was created. Leveraging the devices created by Abaco Systems and the GE Bustools API it was possible to put in communication two computers running different scripts. One would run the OMAP-Probe acting as a BC as well as a BM. The other computer runs the RT version of the API. A configuration file is given to this second API to allow the emulated RT to respond to the messages sent by the BC. Since these are however not real devices, their timings are entirely bound to the executing computer. They are also not able to execute BIT as BIT are created by a manufacturer. It is however possible to postpone the response of an RT to mimic the act of resetting and executing BIT. Figure 5.1 presents the setup used to obtain the dataset. Through this setup, it was possible to create a dataset comprised of 5 RT types (Radio, Radio Spoof, Random, Radio Nav data, MCK). On top of these types, multiple versions of each type were created with modified contents and/or timing features to test the version classification. In total,

assuming each different version is its own class, 11 different classes of RT were probed. Furthermore, each of these 11 classes were probed 100 times each to construct a single dataset. Hence, each data set was comprised of 1100 tuples each with 962 features. From there, the dataset was given to the algorithms to execute classification. For these algorithms, the Google Collab environment was used to run the Python 3.0 scripts. The online virtual machines used to run these scripts was comprised of two Intel(R) Xeon(R) CPU running at 2.00GHz and about 13Gb of RAM.

5.3 Experimental result

5.3.1 One step classification vs Two step Classification

While many factors influence the classification of 1553 devices, using experimentation it was possible to obtain an accuracy rating between 85% and 90%, depending on time deltas, with a neural network using 3 hidden layers. However, this is considering that each different version of a same RT is a different class as presented in the confusion matrix figure 5.2. It is possible to note that the presence of similar values in each RT for different versions causes the classification algorithm to have difficulties in distinguishing between versions. As can be observed in figure 5.2, the ANN classifier could not distinguish between the RT radio versions 1,2 and 3. Similarly, other classes exhibited the same issues. Hence, even when integrating deterministic and distinctive data into the dataset, the high number of features prevents the algorithm from attaining an accuracy rating over 90%. As such, ANN cannot be recommended to implement both type and version classification in a one step manner, since its accuracy is rather low, sitting around 54%. Furthermore, assuming the time values of both the reset and BIT are deterministic, it would be worth while to classify the input per device type prior and then doing some version control afterwards. Figure 5.2 and

Figure 5.2: Confusion Matrix for the OMAP classification

Figure 5.3: Confusion Matrix comparison for the OMAP version classification: D0R0 vs D2R2

Appendix D show that by agglomerating all versions of all RT it is possible to obtain an accuracy rating that is better than having a different class for all versions, reaching over 95% when the time delta of the reset and BIT are higher than the standard deviation for each version. In turn, this points to the fact that the standard deviation has a big impact on the accuracy of the version classification as can be observed in figure 5.3, where we compare the accuracy for two key scenarios (D0R0: Delta smaller than standard deviation and similar time values regardless of version. D2R2: Delta bigger than standard deviation and different time values for each versions). In 5.7, we can see that the scenario D0R0 has an accuracy of 84% for version classification compared to 99% for D2R2. The key difference between these scenarios is the delta in the timing features between the versions, where D0 means a timing delta within the standard deviation bounds and D2 means a timing delta that puts each version outside the standard deviation of the other. Hence, it is possible to advance that as long as the difference between two version is more dominant than the standard deviation of the features, the accuracy will be over 95%.

Spoofed RTs: Joined vs Separated

When dealing with two step classification, we are left with the choice to assign spoof detection to either one of the classification algorithm or to detect spoofs during both steps. In an effort to study the impact of both scenarios on the total accuracy of the proposed approach, we created two additional scenarios: Joined Spoof and Separated Spoof. "Joined Spoof" is the scenario in which the RT type classification does not detect spoofs as attacks since they are considered to be a part of the legitimate RT class. Hence it falls upon the version classification to detect the spoof. Comparatively, "Separated Spoof" is the scenario where spoofs and legitimate are both their own class. As can be seen in figure 5.4, by grouping a class with its spoofed counterpart, it is still possible to obtain an accuracy rating nearing 100% much like when spoofs and legitimate classes are distinct. Hence, if the number of classes was ever to be big enough to impact the accuracy of the classification, it would be possible to regroup spoofed and legitimate device and let the second layer of classification determine if the device is spoofed or not. Doing so does take away the redundancy of the spoofing detection but could limit accuracy losses. Alternatively all spoofed devices could be regrouped within a single device class though this is out of scope and should be discussed in further research. However, by combining a legitimate device and its spoofed counterpart an extra label would need to be added to flag to the end user when an RTs integrity is questionable, whereas by keeping them separated memory integrity is a main part of the distinction of the two classes (legitimate and spoofed).

When we combine spoofed and legitimate data into one class and only task the ANN to classify RT types, as seen in figure 5.4, it's possible to obtain an accuracy nearing 100%. This is the case regardless of the scenarios as these scenarios mainly impact version control through reset and BIT times.

Comparatively, when the spoofed RT and its legitimate counterpart are separate

Figure 5.4: Confusion Matrix for the OMAP classification no spoofing detection

Figure 5.5: Confusion Matrix for the OMAP classification with spoofing detection

Figure 5.6: Accuracy Matrix for the OMAP device classification

classes, as can be observed in figure 5.5, the accuracy can be maintained nearing 100%. From this point on, we can safely assume that the detection of spoofed RT through memory integrity checks can be detected with high accuracy by the ANN.

As presented in the matrix of figure 5.4, figure 5.5 and figure 5.3, by classifying the device type and then its version, it is possible to obtain an accuracy between 84% and 100% for version classification. More confusion matrix are presented in appendix A, B and C where appendix A are the matrix for joined spoofed and legitimate for RT type, appendix B are separated and appendix C are version classification. Furthermore, figure 5.7 illustrates the impact of each scenarios on the accuracy. As can be observed, when the time delta for the BIT and reset value is nearing the standard deviation, such as in the four scenarios of D0, the accuracy for version control drops considerably.

Figure 5.7: Accuracy Matrix for the OMAP version classification

One step vs Two step Classification: Analysis

Figure 5.7 illustrates this phenomenon clearer by presenting the resulting accuracy obtained from running OMAP with the different scenarios. As mentioned above, having a delta value that is within standard deviation is not beneficial. This represents the case where the difference between a legitimate RT and a spoofed RT is not enough to counter the error margin of the measurement combined with the standard deviation of the hardware. It is to note that a standard error of 1 is by default applied to every time value upon creation of the data. Hence, having a delta that is within that range makes the classification process less reliable. Having delta values that are too high also seem to negatively impact the accuracy as can be observed in the D3 row of figure 5.7. Therefore, it will be important during experimentation to find a BC message cycle time that allows to get good distinction between the time values. On the other hand, resemblance can also greatly impact the accuracy. When both Reset

and BIT times are similar between the legitimate and spoofed RT (scenario R0) it is possible to see that the accuracy is lower on average. On the other hand, when both are different, the accuracy is greater as can be witnessed in the case of R2D1. Fluctuation of accuracy can also be due to the training of these different scenarios. However, throughout experimentation, the R2 column always performed better than the 3 other scenarios while the row D1 and D2, with deltas of 3 and 6 respectively, consistently had higher accuracy ratings. Regardless of timings, the accuracy of device type classification made in the first layer was always nearing 100% as can be observed in figure 5.6, proving the effectiveness of giving specific tasks to each classification model through a reduced number of classes. Spoof detection should hence be its own step and the first layer should contain as few class as possible for the best accuracy.

5.3.2 Comparative Evaluation on RT Type Classification

Joined Spoof comparison

As demonstrated above, different scenarios can have a dramatic impact on the accuracy of the model and that proper calibration during the scan will have a direct impact on the classification process. However, different models and algorithms will behave differently and might be more resistant to change. As such the following table details comparative metrics for the ANN, SVC, CNN and MLP models when using 100 iterations. In this case the classifiers were subjected to a dataset in which the spoofing of an RT is solely detected through the version control. Furthermore, the best case scenario was used for comparison basis between the algorithms. As such, the D2R2 scenario was used as a testing dataset for the following comparisons.

From the table 5.1, it is possible to see that ANN, SVC and MLP all attain an accuracy nearing 100%. Meanwhile, the CNN boasts an accuracy much lower near

RT Type Classification comparison no spoof detection					
Model	ANN	SVC	CNN (100)	CNN(300)	MLP
Training Accuracy	1.0000	0.9948	0.8061	0.9325	0.9948
Training Loss	6.557e-4	0.0193	1.0486	0.2935	0.0449
Testing Accuracy	0.9972	0.9909	0.8060	0.9121	0.9909

Table 5.1: Accuracy comparison RT Type (4 classes)

a) This table uses the D2R2 dataset with 4 classes where legitimate RTs and spoofs are joined for a total of 1100 tuples. The models classify each tuple based on the following features: Memory content, BIT time, Reset time. All models trained for 100 iterations except for the CNN (300).

80%. While ANN, SVC and MLP are all relatively simple, CNN requires custom filters to characterize every classes. This means that the user must input specific patterns as filters to obtain good accuracy. This extra step of configuration is not necessary in ANN, SVC and MLP making them more efficient and more accurate in a more timely manner. This also means that every new RT type class that would be introduced might require additional filters for the CNN, making its configuration burden even bigger. However, this does not mean that a CNN based model could not be used to adequately classify 1553 RTs but rather that it should only be used with deeper understanding of specific RT types and how to distinguish one from another rather than letting the model find its own distinguishable features from the dataset. Furthermore, CNN is based on distinguishing and assembling patterns for classification. In the case of 1553 RTs, it is possible that multiple RTs will possess similar data disposition but at entirely different scales which might pose a problem for CNN. On the other hand, ANN and MLP can both easily distinguish a pattern with different intensity more easily due to the intrinsic weight they give to each input nodes rather than the pooled inputs. On the other hand, SVC is able to attain accuracy levels that are similar to ANN and MLP with some minor configuration, making it a prime candidate for further studies with more RT classes considering its ease of

use. From table 5.1, it is also possible to see that despite the dataset, none of the presented display any sign of over-fitting. Regardless of the model, the accuracy is maintained from training to testing. Additionally, loss is nearing zero for all models that have an accuracy nearing 100%. It is possible to note that when CNN is given more time for training it can achieve good accuracy in this 4 classes classification, reaching about 90% accuracy, though later sections will discuss the implication of reaching such accuracy ratings.

Separated Spoof comparison

Comparatively, when given a dataset in which the RT class does not contain its spoofed counterpart, the additional class does severely impact the accuracy. As the following table shows, the ANN performed with the best accuracy though its accuracy dropped significantly during the testing phase. Comparatively, CNN was very stable once its training period was over. Finally, MLP had the worst drop in accuracy for the studied models when facing the test data. However, as can be seen in appendix D, most of the drop of accuracy for the MLP is the mislabeling of RT-spoofed as RT-genuine. As such, it does identify that they are similar but can barely distinguish one from another, mismatching a RT-spoof over half of the times. Contrary to the MLP, SVC and CNN most often mistook RT-spoofed as the random class. In this sense, MLP cannot be recommended as an integrity detection tool and should be used in a grouped dataset where it must predict the RT type or what it tries to mimic.

Furthermore, not all models are as efficient as the other. When given more time to train most models achieved similar accuracy and loss to what is presented in table 5.1 and table 5.2. However, when the CNN model was given over 300 iterations, it was possible for it to achieve an accuracy of 94.24% with a loss of 0.3731.

RT Type Classification comparison with spoof detection					
Model	ANN	SVC	CNN (100)	CNN (300)	MLP
Training Accuracy	1.0000	0.9974	0.8939	0.9454	0.998701
Training Loss	7.9617e-05	0.1008	1.1009	0.3731	0.0097
Testing Accuracy	0.9696	0.9515	0.8787	0.9424	0.8818

Table 5.2: Accuracy comparison RT Type (5 classes)

a) This table uses the D2R2 dataset with 5 classes where legitimate RTs and spoofs are separated for a total of 1100 tuples. The models classify each tuple based on the following features: Memory content, BIT time, Reset time. All models trained for 100 iterations except for the CNN (300).

Figure 5.8 shows the decrease of the loss for 3 of the 4 studied models, since the SVC model calculates its loss based on the hinge loss method. As can be observed, the CNN model had a much slower learning rate than both the ANN and MLP. As such, even after 100 iterations, the loss of the CNN is still over 0.5 which is achieved within the first few iterations of the ANN and MLP. Comparatively, the MLP learning curve is slower but seems more stable as the ANN exhibits loss peaks when reaching near minimal loss values. Furthermore, from table 5.1 and figure 5.8, it is safe to say that no model exhibits heavy over fitting. The same cannot be said when using 5 classes. As can be seen in table 5.2, the ANN, SVC and MLP all have significant drop of accuracy between training and testing exhibiting over fitting issues.

5.3.3 Comparative Evaluation on RT Version Classification

While the RT type classification process presented in this thesis relied on the memory contents of each RT and as such had well over 900 features for each observation, it was decided to classify the versions of each RT based on their response to active probing, much like Nmap would [17]. This effectively reduces the amount of features used to classify the versions down to only the timing features. In the future we recommend to

Figure 5.8: Studied models loss during training period

a) This graph shows the evolution of the loss for the ANN, CNN and MLP models with a dataset containing 5 classes during training.

further the present thesis by using automatic feature selection and feature reduction when the models will be presented with data taken from real RT devices including timing measurements based on the active scan. Nevertheless, KNN and SVC were tested against the obtained timings and the results can be observed below in table 5.3.

From the accuracy of the two models presented in table 5.3, it is possible to see that KNN is heavily dependent on its training data. While the use of KNN reduces the risk for the model to over fit, the dependency of the model on the training data causes it to have high training accuracy but rather low testing accuracy, exhibiting signs of over fitting. Alternatively, the SVC model is very stable between training and testing, with an accuracy nearing 88%. As such, while it does have higher loss than KNN, SVC is recommend for further use. It's accuracy is higher upon testing and is maintained through the iterations. Comparatively, KNN yielded some results

RT Version Classification comparison		
Model	KNN	SVC
Training Accuracy	0.9972	0.8885
Training Loss	0.0612	0.4872
Testing Accuracy	0.8042	0.8733

Table 5.3: Accuracy comparison RT version

a) This table uses the D2R2 dataset with 3 RT versions and spoofs for a total of 4 classes and 500 tuples. The models classify each tuple based on the following features: BIT time, Reset time.

with low accuracy when the training dataset was comprised of features that were not distinctive enough one from another. Additionally, KNN requires training data that is hand picked to attain its best accuracy, since class imbalance will heavily impact the number of neighbours and as such the accuracy of the model.

5.3.4 Comparative execution time analysis

While it was shown that the CNN model can achieve good accuracy and loss, it does requires meticulous tuning and triple the iterations to achieve comparable accuracy. To compare, the whole OMAP classification process (ANN and KNN) takes 44 seconds to execute; SVC executes in under 2 seconds and MLP trains in 7 seconds. Meanwhile, to attain similar accuracy and loss to the top accuracy presented in this thesis, the CNN training period takes over 13 minutes. While this is not a problem, for the time being, it might become one if OMAP is deployed and must scan multiple aircraft and their systems. It is, however, possible to shorten the execution time by augmenting the batch size of the CNN model. While doing so results in a quicker execution time the resulted loss was higher than when using a smaller batch size and the achieved accuracy was below than the smaller batch accuracy. Again, these times were taken

when each algorithm is given 1100 tuples with 962 features for classification whether it be type or version. The classification process was done on the online Google Collab environment which runs Python 3.0 scripts using two Intel(R) Xeon(R) CPU running at 2.00GHz and 13Gb of RAM. Figure 5.9, displays the execution time in seconds for all classification algorithms displayed on a logarithmic scale.

Figure 5.9: Studied models execution time

a) Time taken to achieve accuracy depicted in table 5.2 displayed on a logarithmic scale.

Chapter 6

Conclusion and Future Work

6.1 Conclusions

Assuming that both reset and BIT times are deterministic, it was possible for the proposed approach to obtain an accuracy nearing 100% for device classification and between 80% and 100% for version classification. It is important to note that because the applied time delta value presented in chapter 5 section 1 has a big impact on version classification, it will be mandatory to choose an internal clock speed for the BC message rate that allows to properly distinguish between versions as illustrated by figure 5.7. It is expected that finding the appropriate messaging clock speed for version granularity would not be challenging, considering that transmission of 1553 messages can be configured with some leniency as pointed out by the guidelines seen in the designers manual [1]. Choosing the right message frequency through the BC clock speed will effectively impact the time delta values between two responses to the same probe for two RT version. Since the frequency of the probe message dictates the interval between each probing message, having a higher frequency will lead to knowing exactly when the RT finishes its task, whereas a lower frequency might lead the RT to finish its task between two probing messages. Hence, the division and granularity of the timings features are a direct result of the chosen clock speed. Since

this thesis used simulated RTs, the response timers have been assigned to each RT and we were able to obtain multiple granularity in the form of scenarios. Results also show that when a significant granularity is achieved, such as in the case of scenarios for D2, the combination of ANN and KNN has proven to be an effective solution for the classification of 1553 devices and their versions but not entirely the best. If the achieved accuracy can be maintained on real 1553 devices the same way it was achieved with emulated devices, OMAP could become a corner stone for defensive cyber operations in the MIL-STD-1553 community. On the one hand, the approach presented in this thesis dealt with the cybersecurity side of 1553 and gave good theoretical results. As the dataset used was created from an emulation of 1553 devices, within standard operation procedures, due to the scarce public 1553 devices and their overwhelming cost for consumers, we hope that the obtained results could be replicated with real devices that operates within the standard operation procedures for MIL-1553 aircraft. On the other hand, the process presented in this thesis shows that separating the device type and version classification task between the two chosen algorithm, ANN and KNN, ends up giving a better accuracy rating. While this does give two label for each output value (type,version), the trade-off of dataset complexity is well worth the gained accuracy. It also shows that a time delta value for timing, between versions and spoofed for BIT and reset time, that is within standard deviation will not result in optimal accuracy. Hence, fine tuning of the probing clock cycle is critical. As such, OMAP is a promising first step for the creation of a network discovery and device classification tool for MIL-STD-1553. Our results show that both ANN and KNN are able to adequately detect spoofing and integrity attacks. While we decided to only use KNN to detect spoofing in the end, it was shown that when an RT and its spoofed counter part are considered as 2 different device type, the ANN algorithm of OMAP was able to adequately detect integrity attacks that relied on changing the contents of an RT's memory hence classifying them as different

device types. It was shown, that by training the ANN model with a dataset in which good granularity is achieved, the ANN model was able to detect spoofed RTs with a detection rate of 84.51%, where spoofed RTs had signs of memory tampering. Hence this shows that ANN is well suited for detection of memory integrity attacks.

In this thesis, ANN and KNN were both compared to alternative algorithms as was presented in section 5.3.2 and 5.3.3. This comparison did show that many models were able to obtain an accuracy similar to the ANN, which attained 100% accuracy for training and 99.72% accuracy for testing when using a joined spoof and legitimate dataset. SVC and MLP were both able to achieve low loss and high accuracy through training and testing within a time frame under a minute for RT type classification. Furthermore, we recommend using the MLP in the future instead of ANN because of the high risk the ANN model over fitted around specific RT types, but only if the type classification does not detect spoofs. If the choice is made to no join spoofs and legitimate, then ANN should be maintained. However, if real tested RTs exhibit specific patterns that are humanly distinguishable, CNN might become a viable option. For the time being, however, the intensive configuration and extensive training period of the CNN model does not make it an adequate choice for 1553 devices. SVC has also proven to be a valid choice for both RT type and version classification, with accuracy ratings of 99% and 88% respectively. It is expected, however, that with more classes and more similar data disposition SVC will suffer and will require intensive feature selection and dimensionality reduction to maintain its RT type classification accuracy. This is not the case with RT version classification, which is why we recommend the use of SVC for the classification of 1553 RT versions based on timing properties, since it has good accuracy with minimal loss through training and testing, while exhibiting less signs of over fitting compared to KNN which, in this case, had much lower accuracy in testing accuracy, dropping from 99.72% to 80.42%. Through these steps, it was possible to create an approach which probed a 1553 network, parsed its response

and then classified them in order to predict each device type with an accuracy between 95% and 100% as well as their version with an 88% accuracy. By classifying these devices, we paved the way for proper device memory integrity validation as a second hand effect.

6.2 Future Work

Further analysis of RT's memory could be integrated with the passive approach presented in MAIDENS [21] to establish whether an RT has been compromised. The classification made by OMAP using two step classification could be used in combination with the MAIDENS project to obtain a good authentication of MIL-STD-1553 devices during real-time operations and offline maintenance. Together they could achieve both similarity and anomaly based classification in an active and pro-active way. Where similarity based classification will give a confidence score that the classified device is truly what it was labeled and where anomaly based classification will give a confidence score that a device is a threat. These two scores could then be adjusted with a threshold to signal to the user or technician of a problem within the aircraft. The threshold for detection of anomalies and classification as threat for both algorithms could easily be merged and adjusted together as they are both based on the messages of the 1553 devices. Hence, creating a shared fingerprint dataset could lead to an elevated integrity, authentication and anomaly detection.

It is know that Nmap utilizes different techniques depending on its first scan results, similarly, it will be desirable to experiment whether combining the classification process of OMAP with some device specific probing scripts would gain deeper version control on top of the KNN algorithm. A second round of probing could be added to OMAP once device specific behavior is known. This however will require a deeper knowledge of each specific device and their behavior which is outside of the scope of

this thesis.

While the methods used in this thesis yielded accuracy ratings over 95% for device type and 88% version classification, it would be interesting to compare them to more algorithms. For example, random forest or decision tree instead of KNN for version classification. Alternatively, multiple methods could be applied to obtain multiple predictions and then using a final decision algorithm which would use the confidence score of every algorithm as metrics. While we compared algorithms for the two steps of classification, it was tested on the best case scenario: D2R2. Hence, it might be beneficial to test the algorithms on all the scenarios to evaluate how they react with data that is less distinctive between each classes.

The next step for development of OMAP resides in the scanning and creation of fingerprints for real 1553 devices. Once a fingerprint database has been created it will be possible to incorporate it in the learning phase of OMAP to begin applied testing. It would also be possible to build a database of device fingerprint to be shared with cybersecurity personnel working on 1553. With this new data, real time values as well as message transfer times could also be integrated in the classification process. While these times may not be as precise as IAT are for IoT devices they could be an interesting feature to consider. At that point, considering the amount of extra features, it might be necessary to add some form of automatic feature selection process, perhaps using genetic algorithms.

Finally, while the present research aimed at detecting integrity attacks, an argument could be made to tell the user the location of the integrity fault within the memory map. This could help in correlating which device caused the fault through the help of the BM recordings. Nevertheless, OMAP has proven to be effective in itself, but much like Nmap, it will require a plethora of extra tools and tricks to achieve the statute of all purpose defensive cyber tool.

Appendix A

Accuracy and Confusion matrix for device classification joined spoof and legitimate

Figure A.1: Accuracy Matrix for the OMAP device classification

(a) Confusion Matrix Device classification D0R0

(b) Confusion Matrix Device classification D0R1

(c) Confusion Matrix Device classification D0R2

(d) Confusion Matrix Device classification D0R3

(a) Confusion Matrix Device classification D1R0

(b) Confusion Matrix Device classification D1R1

(c) Confusion Matrix Device classification D1R2

(d) Confusion Matrix Device classification D1R3

(a) Confusion Matrix Device classification D2R0

(b) Confusion Matrix Device classification D2R1

(c) Confusion Matrix Device classification D2R2

(d) Confusion Matrix Device classification D2R3

(a) Confusion Matrix Device classification D3R0

(b) Confusion Matrix Device classification D3R1

(c) Confusion Matrix Device classification D3R2

(d) Confusion Matrix Device classification D3R3

Appendix B

Accuracy and Confusion matrix for device classification separated spoof and legitimate

Figure B.1: Accuracy Matrix for the OMAP device classification separate

(a) Confusion Matrix Device classification D0R0 separate

(b) Confusion Matrix Device classification D0R1 separate

(c) Confusion Matrix Device classification D0R2 separate

(d) Confusion Matrix Device classification D0R3 separate

(a) Confusion Matrix Device classification D1R0 separate

(b) Confusion Matrix Device classification D1R1 separate

(c) Confusion Matrix Device classification D1R2 separate

(d) Confusion Matrix Device classification D1R3 separate

(a) Confusion Matrix Device classification D2R0 separate

(b) Confusion Matrix Device classification D2R1 separate

(c) Confusion Matrix Device classification D2R2 separate

(d) Confusion Matrix Device classification D2R3 separate

(a) Confusion Matrix Device classification D3R0 separate

(b) Confusion Matrix Device classification D3R1 separate

(c) Confusion Matrix Device classification D3R2 separate

(d) Confusion Matrix Device classification D3R3 separate

Appendix C

Accuracy and Confusion matrix for version classification

Figure C.1: Accuracy Matrix for the OMAP version classification

(a) Confusion Matrix Version classification D0R0

(b) Confusion Matrix Version classification D0R1

(c) Confusion Matrix Version classification D0R2

(d) Confusion Matrix Version classification D0R3

(a) Confusion Matrix Version classification D1R0

(b) Confusion Matrix Version classification D1R1

(c) Confusion Matrix Version classification D1R2

(d) Confusion Matrix Version classification D1R3

(a) Confusion Matrix Version classification D2R0

(b) Confusion Matrix Version classification D2R1

(c) Confusion Matrix Version classification D2R2

(d) Confusion Matrix Version classification D2R3

(a) Confusion Matrix Version classification D3R0

(b) Confusion Matrix Version classification D3R1

(c) Confusion Matrix Version classification D3R2

(d) Confusion Matrix Version classification D3R3

Appendix D

Confusion matrix for device Type classification separated spoof and legitimate comparison

Figure D.1: Accuracy Matrix for the ANN 5 Types classification

Figure D.2: Accuracy Matrix for the SVC 5 Types classification

Figure D.3: Accuracy Matrix for the CNN 5 Types classification - 100 iterations

Figure D.4: Accuracy Matrix for the CNN 5 Types classification - 300 iterations

Figure D.5: Accuracy Matrix for the MLP 5 Types classification

List of References

- [1] “Mil-std-1553 tutorial and reference.” [Online]. Available: <https://www.altadt.com/support/tutorials/mil-std-1553-tutorial-and-reference/>
- [2] “Designer’s guide - mil-std-1553.” [Online]. Available: <https://www.milstd1553.com/wp-content/uploads/2012/12/MIL-STD-1553B.pdf>
- [3] “552nd air control networks squadron creates first qualification training for mission defen,” Nov 2019. [Online]. Available: <https://www.acc.af.mil/DesktopModules/ArticleCS/Print.aspx?PortalId=92&ModuleId=12972&Article=2012383>
- [4] O. Stan, A. Cohen, Y. Elovici, and A. Shabtai, “Intrusion detection system for the mil-std-1553 communication bus,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 4, p. 3010–3027, 2020.
- [5] J. Wiggen, “The impact of covid-19 on cyber crime and state-sponsored cyber activities,” Jun 2020. [Online]. Available: https://www.kas.de/documents/252038/7995358/The+impact+of+COVID-19+on+cyber+crime+and+state-sponsored+cyber+activities.pdf/b4354456-994b-5a39-4846-af6a0bb3c378?version=1.0&_t=1591354291674
- [6] “Flipper zero - multi-tool device for hackers. lite version based on stm32.” [Online]. Available: <https://flipperzero.one/>
- [7] S. J. Merrifield, “Mil-std-1553 fpga network traffic emulator.” [Online]. Available: <http://stevenmerrifield.com/1553/index.html>
- [8] O. Stan, Y. Elovici, A. Shabtai, G. Shugol, R. Tikochinski, and S. Kur, “Protecting military avionics platforms from attacks on mil-std-1553 communication bus,” Jul 2017. [Online]. Available: <https://arxiv.org/abs/1707.05032>

- [9] V. C. Hu, D. F. Ferraiolo, and D. R. Kuhn, “Assessment of access control systems - nist,” Sep 2006. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir7316.pdf>
- [10] E. Barker, L. Chen, and D. Moody, “Recommendation for pair-wise keyestablishment schemes using integer factorization cryptography,” Sep 2014. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br1.pdf>
- [11] M. Scholl, “An introductory resource guide for implementing the health ...” Oct 2008. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-66r1.pdf>
- [12] W. L. Ross, “Risk management framework for information systems and ...” Dec 2018. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf>
- [13] M. Nieves, K. L. Dempsey, and V. Y. Pillitteri, “An introduction to information security,” Nov 2018. [Online]. Available: <https://www.nist.gov/publications/introduction-information-security>
- [14] “Cyber attack - what are common cyberthreats?” Apr 2020. [Online]. Available: https://www.cisco.com/c/en_ca/products/security/common-cyberattacks.html
- [15] “What is snort?” [Online]. Available: <https://www.snort.org/>
- [16] “Wireshark.” [Online]. Available: <https://www.wireshark.org/>
- [17] “Nmap.” [Online]. Available: <https://nmap.org/>
- [18] M. Zamot, “Finding rogue devices in your network using nmap,” Dec 2020. [Online]. Available: <https://www.redhat.com/sysadmin/finding-rogue-devices>
- [19] A. Moshirnia, “No security through obscurity: Changing circumvention law to protect our democracy against cyberattacks,” 2018. [Online]. Available: <https://brooklynworks.brooklaw.edu/blr/vol83/iss4/3/>
- [20] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, “Unmanned aircraft capture and control via gps spoofing,” *Journal of Field Robotics*, vol. 31, no. 4, p. 617–636, 2014.

- [21] S. J. J. Génereux, A. K. H. Lai, C. O. Fowles, V. R. Roberge, G. P. M. Vigeant, and J. R. Paquet, “Maidens: Mil-std-1553 anomaly-based intrusion detection system using time-based histogram comparison,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 1, pp. 276–284, 2020.
- [22] “Sans institute,” Apr 2021. [Online]. Available: <https://www.sans.org/blog/what-you-need-to-know-about-the-solarwinds-supply-chain-attack/>
- [23] R. Changalvala and H. Malik, “Lidar data integrity verification for autonomous vehicle using 3d data hiding,” *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019.
- [24] Changalvala and Malik, “Sensor data integrity verification for autonomous vehicles using spread 3d dither qim,” *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, 2020.
- [25] H. Noguchi, M. Kataoka, and Y. Yamato, “Device identification based on communication analysis for the internet of things,” *IEEE Access*, vol. 7, p. 52903–52912, 2019.
- [26] Y. Song, Q. Huang, J. Yang, M. Fan, A. Hu, and Y. Jiang, “Iot device fingerprinting for relieving pressure in the access control,” in *Proceedings of the ACM Turing Celebration Conference - China*, ser. ACM TURC ’19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: <https://doi.org/10.1145/3321408.3326671>
- [27] N. Ammar, L. Noirie, and S. Tixeuil, “Autonomous identification of iot device types based on a supervised classification,” *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020.
- [28] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, “Inferring iot device types from network behavior using unsupervised clustering,” *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, 2019.
- [29] A. Aksoy and M. H. Gunes, “Automated iot device identification using network traffic,” *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019.
- [30] S. Aneja, N. Aneja, and M. S. Islam, “Iot device fingerprint using deep learning,” *CoRR*, vol. abs/1902.01926, 2019. [Online]. Available: <http://arxiv.org/abs/1902.01926>

- [31] A. S. Uluagac, S. V. Radhakrishnan, C. Corbett, A. Baca, and R. Beyah, "A passive technique for fingerprinting wireless devices with wired-side observations," in *2013 IEEE Conference on Communications and Network Security (CNS)*, 2013, pp. 305–313.
- [32] K. Yang, Q. Li, and L. Sun, "Towards automatic fingerprinting of iot devices in the cyberspace," *Computer Networks*, vol. 148, p. 318–327, 2019.
- [33] B. Bezawada, I. Ray, and I. Ray, "Behavioral fingerprinting of internet-of-things devices," *WIREs Data Mining and Knowledge Discovery*, vol. 11, no. 1, 2019.
- [34] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Communications Surveys; Tutorials*, vol. 21, no. 1, p. 686–728, 2019.
- [35] N. Chockwanich and V. Visoottiviset, "Intrusion detection by deep learning with tensorflow," *2019 21st International Conference on Advanced Communication Technology (ICACT)*, 2019.
- [36] C. Gupta, S. Wang, U. Dayal, and A. Mehta, "Classification with unknown classes," *Lecture Notes in Computer Science Scientific and Statistical Database Management*, p. 479–496, 2009.
- [37] A. Hoffmann, R. Kwok, and P. Compton, "Using subclasses to improve classification learning," *Machine Learning: ECML 2001 Lecture Notes in Computer Science*, p. 203–213, 2001.
- [38] A. Mandelbaum and D. Weinshall, "Distance-based confidence score for neural network classifiers," *CoRR*, vol. abs/1709.09844, 2017. [Online]. Available: <http://arxiv.org/abs/1709.09844>
- [39] "Mil-std-1553b in avionics: where data networking has been and where it's going." [Online]. Available: <https://www.intelligent-aerospace.com/commercial/article/16544804/milstd1553b-in-avionics-where-data-networking-has-been-and-where-its-going>
- [40] B. Losier, "Design of a time-based intrusion detection algorithm for the mil-std-1553," Jan 2019. [Online]. Available: http://roberge.segfaults.net/joomla/files/publications/Project_Report_2102.pdf
- [41] D. De Santo, C. Malavenda, S. Romano, and C. Vecchio, "Exploiting the mil-std-1553 avionic data bus with an active cyber device," *Computers Security*, vol. 100, p. 102097, 2021.

- [42] B. Hamid, B. Gallina, A. Shabtai, Y. Elovici, and J. Garcia-Alfaro, “On the security of mil-std-1553communication bus,” *Security and Safety Interplay of Intelligent Software Systems ESORICS 2018 International Workshops, ISSA 2018 and CSITS 2018, Barcelona, Spain, September 6-7, 2018, Revised Selected Papers*, 2019.
- [43] “Artificial neural network,” Jul 2021. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network
- [44] T. S. Fatayer and M. N. Azara, “Iot secure communication using ann classification algorithms,” *2019 International Conference on Promising Electronic Technologies (ICPET)*, 2019.
- [45] “1.4. support vector machines.” [Online]. Available: <https://scikit-learn.org/stable/modules/svm.html#classification>
- [46] “1.17. neural network models (supervised).” [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron
- [47] “Convolutional neural network,” Jul 2021. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network
- [48] “sklearn.neighbors.kneighborsclassifier.” [Online]. Available: <https://scikit-learn.org/stable/modules/neighbors.html>
- [49] B. Kazan, “Additional classes effect on model accuracy using transfer learning,” Ph.D. dissertation, DiVA, 2020.