OD1NF1ST: True Skip Intrusion Detection and Avionics Network Cyber-attack Simulation

MICHAEL MAXIMILIAN WRANA, MARWA ELSAYED, KARIM LOUNIS, ZIAD MANSOUR, STEVEN DING, and MOHAMMAD ZULKERNINE, Queen's University, Canada

MIL-STD-1553 is a communication bus that has been used by many military avionics platforms, such as the F-15 and F-35 fighter jets, for almost 50 years. Recently, it has become clear that the lack of security on MIL-STD-1553 and the requirement for internet communication between planes has revealed numerous potential attack vectors for malicious parties. Prevention of these attacks by modernizing the MIL-STD-1553 is not practical due to the military applications and existing far-reaching installations of the bus. We present a software system that can simulate bus transmissions to create easy, replicable, and large datasets of MIL-STD-1553 communications. We also propose an intrusion detection system (IDS) that can identify anomalies and the precise type of attack using recurrent neural networks with a reinforcement learning true-skip data selection algorithm. Our IDS outperforms existing algorithms designed for MIL-STD-1553 in binary anomaly detection tasks while also performing attack classification and minimizing computational resource cost. Our simulator can generate more data with higher fidelity than existing methods and integrate attack scenarios with greater detail. Furthermore, the simulator and IDS can be combined to form a web-based attack-defense game.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; • **Security and privacy** → **Intrusion detection systems**; • **Computer systems organization** → **Embedded and cyber-physical systems;**

Additional Key Words and Phrases: Machine Learning, reinforcement learning, simulation software, anomaly detection, avionics networks

ACM Reference format:
Michael Maximilian Wrana, Marwa Elsayed, Karim Lounis, Ziad Mansour, Steven Ding, and Mohammad Zulkernine. 2022. OD1NF1ST: True Skip Intrusion Detection and Avionics Network Cyber-attack Simulation. *ACM Trans. Cyber-Phys. Syst.* 6, 4, Article 33 (December 2022), 27 pages.
https://doi.org/10.1145/3551893

All the authors are with School of Computing, Queen's University, Canada.
Dr. Zulkernine and Dr. Ding acknowledge funding by the Innovation for Defence Excellence and Security (IDEaS) Program for the 6 months period of work. Under the Competitive Projects Element, Dr. Zulkernine and Dr. Ding have been awarded funding to support the development of a possible solution to the Defence Challenge: Spot The Hack: Intrusion detection systems for avionics networks and bus technologies.
Authors' address: M. M. Wrana, M. Elsayed, K. Lounis, Z. Mansour, S. Ding, and M. Zulkernine, Queen's University, 557 Goodwin Hall, Kingston, Ontario, Canada, K7L 2N8; emails: {16mmw, marwa.afifi, karim.lounis, steven.ding, mz}@queensu.ca.

2378-962X/2022/12-ART33
https://doi.org/10.1145/3551893

1 INTRODUCTION

The MIL-STD-1553 protocol is a standard for communications that was developed by the United States Department of Defense mainly for use in avionics platforms [13]. MIL-STD-1553 is intended to allow for communication between different devices physically connected through a bus. Since its creation, the communication bus has been installed on many avionics platforms in service with the United States Air Force and Navy, NATO coalition aircraft in Europe, Russian fighter jets, and satellites operated by NASA.

The original specification focuses heavily on fault tolerance and reliability due to the intended military and aerospace applications. The MIL-STD-1553 is synchronous (using a leader/follower topology), halfduplex, and deterministic. The bus uses a multi-point topology to link devices (terminals) through a shared backbone (bus). Despite the care taken to design MIL-STD-1553's reliability system, security protocols were merely an afterthought. In the 1970s, when MIL-STD-1553 was first developed, our modern notion of cybersecurity did not exist. Since first being installed in the F-16 Fighting Falcon, MIL-STD-1553 has spread to billions of dollars worth of internet-connected military hardware across the globe. However, the system is not robust toward modern cyber attacks like **denial of service (DoS)** or **man-in-the-middle (MITM)** [20]. Redesigning MIL-STD-1553 from scratch has been tried but is impractical due to the bus' wide-reaching installations [38]. An alternative solution is to augment the existing protocol with protection.

The best form for such an augmented protection device is an **intrusion detection system (IDS)** [34]. An IDS is not designed to stop attacks from being possible but rather to identify when an intrusion has happened and notify the relevant parties and can provide the required security to avionic and satellite platforms while being practical to apply to existing installations of MIL-STD-1553. Recent studies have proposed several data-driven IDS solutions for MIL-STD-1553 [6, 16, 21, 40, 46, 47]. Existing work has adopted a co-simulation method, whereby communication traffic is directly observed from the physical bus while data being transmitted is algorithmically generated as synthetic messages. The results described are promising, however, they suffer from several common issues:

- I1: Short simulation duration and synthetic communication messages. Much of the existing work simulates benign traffic and cyber attacks in a very limited timeframe of 1–20 min. The generated data cannot effectively evaluate how the proposed IDS adapts to changing environments in different flight scenarios. Additionally, the data is collected by algorithmically generated messages rather than actual flight communications.
- I2: Limited number of attack scenarios and their combinations. Most of the existing work is implemented and independently simulated with only 1–3 attacks.
- I3: Only consider one type of communication message due to computational limits. Existing methods only analyze a subset of transmitted words (command words). However, based on our analysis of the protocol, some attacks can be successfully executed without using any command words.

I1 and I2 are caused by the lack of a comprehensive and flexible full-stack simulation system that is able to (1) repetitively simulate bus traffic in a long run, (2) simulate cyber attacks of various vectors for many runs, and (3) simulate normal behaviors with high-fidelity flight data. I3 was caused by the lack of a learning system that is able to scale with the number of messages being communicated. Furthermore, current IDS' only provide binary anomaly detection and cannot explain the type of attack occurring.

Due to the lack of existing datasets for MIL-STD-1553 and the inconsistency created by I1 and I2, we first built a complete open-source simulation system of the MIL-STD-1553 communication protocol that enables the implementation of custom and pre-defined advanced cyber attacks over an arbitrary simulation platform. Our simulator allows the collection of high-fidelity data over long time periods and facilitates the creation of an IDS for this platform. The main contributions of this article are as follows:

- We propose a new neural network architecture that provides a data skipping mechanism specifically designed for IDS'. It can identify whether a message is considered anomalous and specify the precise type of attack with greater performance compared to other methods.
- We experimentally demonstrate the effectiveness of our IDS through comparison with existing state-of-the-art neural network models designed for avionics platforms.

2 RELATED WORKS

System Security. Here, we summarize the main issues with security on MIL-STD-1553 and describe some solutions to these problems.

Increased inter-connectivity between the aircraft control domain and outside world has opened new attack vectors in secure avionics networks. Consider the Airbus A350-900-XWB, which uses MIL-STD-1553. An FAA report indicates that the A350 network architecture allows for increased connectivity between the aircraft control domain and external network sources [4]. Carpenter and Cerchio et al. describe an increased cyber-attack risk on A350 aircraft due to the use of TCP/IP for communication of flight critical avionics, which creates a bridge between MIL-STD-1553 and the outside world [8, 11]. He et al. and McGraw et al. describe the risks associated with MIL-STD-1553 installations on wireless spacecraft networks [20, 36].

Existing work has described the weaknesses of MIL-STD-1553 and given examples of both uplink and downlink flow attacks [19, 33, 43]. In an uplink flow attack, the malicious party could modify information appearing in cockpit displays causing the pilot to take action based on false information [19, 33]. In a downlink flow attack, the malicious party would target hardware components causing malfunctions such as the ejection seat sending the pilot into the air [19, 33].

We will summarize the security of MIL-STD-1553 based on five cybersecurity categories: Authentication, Confidentiality, Integrity, Availability, and Non-repudiation. The authentication and non-repudiation systems provided by MIL-STD-1553 are non-existent. Once an intruder has breached the network they can send words to the bus posing as any device at any time. The assumption made during development was that the military use-case would provide these services. This also contributes to compromising the confidentiality, integrity, and availability of the overall system [45].

Maintaining confidentiality on MIL-STD-1553 is crucial due to the sensitive and easily exploitable nature of secure military communications. There is no consideration of this paradigm in MIL-STD-1553, as all messages are publicly available in plain text to every device. Appliances have been designed that can break system confidentiality without direct installation [12]. Some systems have been developed to encrypt data on the bus, but are not widely implemented due to cost and space constraints [26].

MIL-STD-1553 is somewhat robust toward integrity attacks as words cannot be intercepted or modified after transmission [49]. Attacks such as MITM that modify words are theoretically possible although have infinitesimal chances of success due to the small inter-message time gap [20]. Some IDS' have been designed that can detect specific types of integrity attacks on the platform [16, 21].

MIL-STD-1553 is vulnerable to many availability attacks as collisions between words are possible if a malicious party is sending messages at unexpected times [20]. Most existing IDS' can successfully detect some form of availability attacks [6, 16, 21, 47].

Table 1. Overview and Comparison of Existing Intrusion Detection Systems for MIL-STD-1553

Author	Dataset	IDS Type	IDS Method	Security
Stan et al.	semi-physical	Baseline	Markov Chain	A
Genereux et al.	software	Baseline	Histogram	I, A
Losier et al.	semi-physical	Baseline	Histogram	—
He et al.	semi-physical	Classification	Naïve Bayes	I, A
Onodueze et al.	software	Classification	Deep Learning	I, A

Data Generation. One of the major challenges associated with this research area is appropriate dataset generation. An overview of existing approaches can be found in Table 1. There are no publicly accessible databases containing real message sequences. As a result, different authors have created independent datasets to prove the effectiveness of their IDS.

Yahalom et al. have created three publicly available datasets to be used for IDS testing [54]. Networks struggled to train on this dataset due to the limited attack scenarios (two) and lack of malicious messages [40].

Most authors used a semi-physical setup involving a physical bus emulator and software IDS [14]. Stan et al. created their dataset and attack scenarios using a multi-PC physically simulated environment where one machine is benign and another malicious [47]. Losier et al. and He et al. created benign data with a physical simulation and then digitally inserted attack scenarios to later be detected by the IDS [6, 21]. Genereux et al. used a purely software-based simulation where data and attacks are generated digitally [16].

Intrusion Detection Systems. Intrusion Detection Systems have been used to secure similar embedded systems [5, 17]. A summary of existing MIL-STD-1553 IDS' is in Table 1. Stan et al. generated a baseline using a Markov Chain statistical model and evaluated its effectiveness against some pre-planned attack scenarios [47]. Genereux et al. used a similar system by generating a baseline histogram of message timings and then compared the graph during pre-calculated attack scenarios [16]. Losier et al. suggested a similar histogram approach, however, the detailed attack scenarios are classified and thus cannot be compared [6]. This approach and IDS design has proven to be effective, but the scope and generalizability to the real world were somewhat limited. Another IDS proposed by He et al. instead used a Naive Bayes classifier to identify if a message is anomalous or not [21].

Controller Area Network (CAN) intrusion detection is a similar problem space surrounding cyber attacks on vehicular networks. Zhang et al. propose a two-layer system using rule-based detection followed by a deep neural network [55]. Nam et al. show how bi-directional neural networks can be effective intrusion detection tools in vehicular networks [39]. Javed et al. describe a two-layer neural network with convolution layers followed by an RNN [24]. However, there are some fundamental differences that make CAN solutions not directly applicable. Certain attacks such as fuzzing and replay attacks are possible on CAN [48] while impossible on MIL-STD-1553 and vice versa. Furthermore, message timing is more stringent on MIL-STD-1553 leading to different network architecture needs.

Machine learning-based anomaly detection has been shown to be effective in cyber-physical systems [15, 25, 30, 53]. Some machine learning-based IDS' have also been tried on MIL-STD-1553 [40]. Onodueze et al. indicated that the imbalanced public dataset [54] and lack of attack scenarios resulted in poor performance. Furthermore, any practical IDS for MIL-STD-1553 also must function with the limited computing power that is available on older avionics platforms [52]. Some solutions to minimizing the computational efficiency for **Recurrent Neural Network (RNN)**-based anomaly detectors include the skip-RNN [7] and leap-LSTM [23]. Furthermore, shortcut

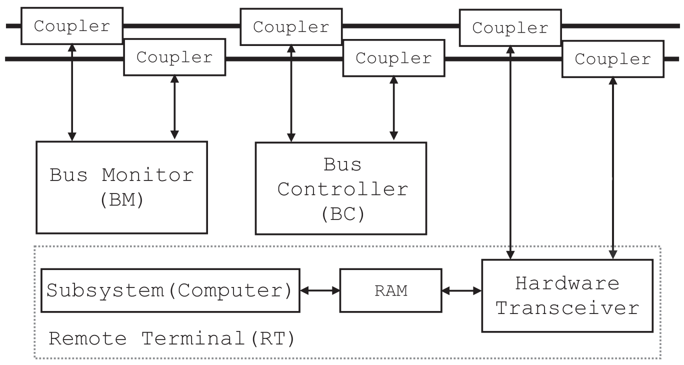


Fig. 1. MIL-STD-1553 architecture and its primary components.

paths [27], residual networks [22], and highway paths [44] have been used to improve the gradient calculation of LSTMs. These models use a similar idea as the skip-RNN and leap-LSTM to improve computational efficiency.

3 ATTACK MODEL

Here, we will summarize the attack model for MIL-STD-1553, including system architecture, security objectives and assets, trusted computing base, and adversary model.

3.1 Bus Architecture

First, we will provide an overview of the MIL-STD-1553 protocol in terms of its components, word types, and message protocols. MIL-STD-1553 consists of a **Bus Controller (BC)** controlling multiple terminals (RTs) linked by a physical bus that provides a single data path between the controller and associated terminals. Every device linked to the data line can read every word (sent in plain text) that is transmitted. The system also supports double and triple redundancy by coupling each device to multiple data lines. Figure 1 shows a dual-redundant system.

3.2 Bus Components

There are three main devices connected to a MIL-STD-1553 system, which are summarized below and in Figure 1.

3.2.1 Bus Controller (BC). The bus controller is the master device and manages all communications over the network. It can initiate data transmission between devices.

3.2.2 Remote Terminal (RT). The remote terminal can be used to connect the data bus to a computing subsystem or link one MIL-STD-1553 to another. RTs are followers and can only send words through the bus when commanded by the BC. There can be a maximum of 30 RTs on a single network.

3.2.3 Bus Monitor (BM). The bus monitor records words that are transmitted through MIL-STD-1553. Depending on available storage, every word or some specified subset may be recorded. This component is critical for an IDS, as the data collected by the BM can be used to train models and evaluate a system for intrusions.

3.3 Word Types

There are three unique types of messages (words) that can be sent by devices on MIL-STD-1553, which are detailed in Figure 2. Each word consists of 20 bits. The first 3 are for synchronization

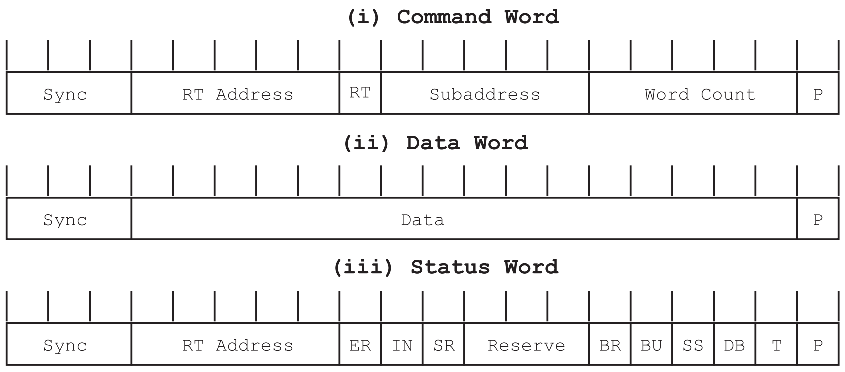


Fig. 2. Contents of (i) Command, (ii) Data, and (iii) Status words in MIL-STD-1553 protocol.

and contain a Manchester code [32] that allows for clock and data information to be sent on the same wire. The next 16 contain a payload that differs depending on the message type. The final bit is for error/parity verification.

3.3.1 Command. Command words are used by the BC to instruct a specific device on the bus to take an action.

3.3.2 Data. Data words are used to send information from one device to another. The 16 bits of payload contain the data being transmitted over the bus.

3.3.3 Status. Status words are used by devices to respond to the BC before data transmission or after data reception.

3.4 Message Protocols

There are four types of message protocols that can be initiated by the BC on MIL-STD-1553, which are summarized in Figure 3. The standard does not specify a particular order of command sequences, this is left to be programmed by each individual user. On most military aircraft the BC has a preset cyclic schedule of data transfers to be executed.

3.4.1 RT-BC. This protocol allows for one-way data transfer from an RT to the BC. First, the BC sends a `transmit` word specifying an RT, which will be sending data. Next, the targeted RT will respond with a `status` word indicating any issues before transmission. Immediately following the status word, the RT will send a number of `data` words as specified in `transmit` to the bus.

3.4.2 BC-RT. This protocol facilitates one-way data transfer from the BC to an RT. First, the BC sends a `receive` indicating the RT that will be receiving data. Immediately following this, the BC will send a number of `data` words as specified in `receive` to the bus. Following reception of the `data` words, the targeted RT will respond with a `status` word indicating any issues with the transmission.

3.4.3 RT-RT. This protocol provides one-way data transfer from an RT to another RT. First, the BC sends `receive` to the bus indicating RT1, which is receiving data. Next, the BC sends a `transmit` specifying RT2, which is sending data. RT2 responds with a `status` word indicating any problems immediately followed by the requested number of `data` words. RT1 finishes the protocol by sending a `status` word indicating any issues with the transmission.

3.4.4 Broadcast. There are two defined types of broadcast protocols that allow one device to send data to every other device. First, the BC broadcast begins with the BC sending `receive` to all

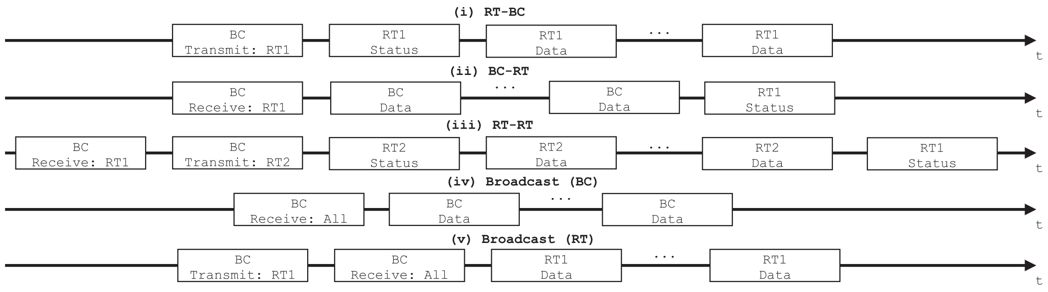


Fig. 3. Data transfer protocols on MIL-STD-1553 for (i) RT-BC, (ii) BC-RT, (iii) RT-RT, (iv) Broadcast (BC), and (v) Broadcast (RT).

Table 2. Security Objectives for MIL-STD-1553 Systems

Category	Objective
Confidentiality	Prevent leakage or extraction of data from MIL-STD-1553 systems. Data should not be extracted through eavesdropping on words sent through the bus. Electromagnetic-based attacks such as Tempest [1, 2, 31] should not be possible. It should be possible to use encryption techniques on the bus and connected subsystems [1].
Integrity	Prevent data words on the bus from being manipulated or modified in any way. The network must be protected from corruption of data or insertion of attacker-generated data into the system. [31]
Availability	Ensure the bus can be used to communicate important information at any time as required by the BC. This includes protection from both physical tampering of the hardware components and software-based denial of service. Do not allow RTs or their associated computer systems to be disabled or disconnected from the network.
Authentication	Guarantee that all devices connected to MIL-STD-1553 are authorized to read each data word within the network. Devices should only be able to read classified data with the correct permission level [1].This includes physically linked RTs and the computer systems associated with each RT.

devices followed by the specified number of data words. No RTs will respond following after the data words have been transmitted. Alternatively, the BC can first send a `transmit` word indicating a specific RT. The targeted RT will send data to the bus and every device will interpret it.

3.5 Security Objectives

The security objectives of a user of MIL-STD-1553 can be divided into four categories: confidentiality, integrity, availability, and authentication. A detailed list of our objectives can be found in Table 2. Any MIL-STD-1553 system should be secure from: leakage or extraction of data, information being manipulated or modified, communication interruptions or corruption, and unauthorized devices connecting to the network.

The designer's notes state that usage of encryption systems (i.e., cryptographic keys, zeroize techniques) should be possible, although provide no guidelines for how to achieve this. Authentication is briefly considered with regard to classification levels [1]. Each device connected to the network should only have access to data that it is classified to see, and data transmitted at a higher level should be inaccessible [1].

3.6 Asset Identification

3.6.1 Hardware Assets (HW). Hardware assets for the MIL-STD-1553 include the components of the aircraft that are connected to the bus (e.g., GPS, HUD, Gyroscope), the components of the bus itself (e.g., chipset, wires, couplers), and I/O ports on RTs connected to the bus (e.g., USB).

Compromising any piece of hardware associated with MIL-STD-1553 would severely violate the security objectives of confidentiality, availability, and authorization, while weakening integrity [45]. An attacker could eavesdrop on communications through an electromagnetic signal reader, or connect directly and extract data from the bus. The availability would be compromised through physical tampering [12] or connecting to the network and spamming the bus with words. Since no encryption or message signing mechanisms are present on MIL-STD-1553, the authentication could be violated by physical connection. Breaking integrity is made easier with physical intrusion but would require additional information assets.

The extensive worldwide supply chains associated with modern aircraft like the A350 present numerous opportunities to compromise a flight-critical component [10, 51]. Stan et al. explain how even a tested component can be compromised by code that waits to be in a real aircraft environment [45]. Miller et al. survey critical infrastructure incidents, some of which were caused by compromised supply chains [37].

3.6.2 Information Assets (Info). There are two main categories of information assets that an intruder may require to compromise MIL-STD-1553: RT addresses and the contents of words. This information alone cannot be used to violate a security objective, however, it is required in addition to a software or hardware asset to intrude on the network.

To breach the system's confidentiality and authentication, the attacker would require details about the contents of command and status words. This includes targeted RT addresses, mode-code bits, and word timings from the legitimate BC. An attacker would additionally need information from data words such as formatting and context to violate system integrity.

3.6.3 Software Assets (SW). Software assets for MIL-STD-1553 include the operating systems and programs contained in any of the RTs connected to the network. These could be breached by an attacker through remote or physical malicious software update injection. Breaking into the software assets of MIL-STD-1553 could violate all the security objectives, although it is mostly focused on integrity, availability, and authentication [47].

Once given access to a software system within a MIL-STD-1553 network, an attacker could violate availability by spamming words on the bus. The malicious party could (with additional information assets) breach the integrity and authentication of the system by intercepting transmission protocols through BC impersonation.

3.7 Trust Model

The main assumption about trusted devices is that all hardware connected to and part of the bus itself is legitimate. This is reasonable due to the highly secure nature of MIL-STD-1553 applications. Any malicious party would need physical access to a military base or interception of the production supply chain to compromise any hardware asset. As such, all the simulated attacks focus on exploits that can be performed with only software and information assets.

3.8 Adversary Model

Our software package implements ten different MIL-STD-1553 exploits in eight categories. Here, we will describe each attack in detail and how it takes advantage of the communication protocol. The full implementation of each of these attack scenarios can be found on our Github. A summary of the scenarios can be found in Table 3.

3.8.1 Random-word Generation. This is a relatively simple Denial of Service attack [35]. The attacker waits for any word to be sent on the bus and attempts to also send a word containing random bits at the same time. The messages will collide on the bus and the benign messages are

Table 3. Summary of MIL-STD-1553 Vulnerabilities Exploited by OD1NF1ST

Attack Type	Objective Violated	Assets Required	Protocol Exploited	Attack Indicators
Random Word Generation	A	SW	None	Interrupt Bus
Man in the Middle	All	SW, Info	All	Intercept Protocol
Status Word Manipulation	I, A	SW, Info	RT-BC, BC-RT, RT-RT	Interrupt Bus
Command Invalidation	A, Auth	SW, Info	RT-BC, RT-RT	Interrupt RT
TX Shutdown	A, Auth	SW, Info	All	Interrupt RT
Desynchronization	I, A, Auth	SW, Info	All	Interrupt RT
Data Trashing	A	SW, Info	BC-RT, RT-RT	Intercept Protocol
Data Word Corruption	I	SW, Info	RT-BC, RT-RT	Intercept Protocol

rendered unreadable. Alternatively, the attacker can simply constantly spam words onto the bus causing collisions. This attack has two different styles where either the currently communicating RT or any coupled RT can be targeted by the attacker.

3.8.2 Man-in-the-middle (MITM). In MITM, the attacker seeks to insert a data transfer between two targeted RTs: RT0 and RT1. The attacker poses as a BC and sends fake `command:receive (address0)` and `command:transmit (address1)` words to the bus. The RTs have no way to identify the message source and immediately comply. If successful, then the attacker can either gain access to classified data or send manipulated data to a device on the network. Although MITM is extremely dangerous, it requires the real BC to send no words during the entire operation, as any command words transmitted would cause a message collision rendering the attacker's data unreadable. Due to the cyclic nature of MIL-STD-1553, MITM attacks will most likely become ineffective DoS attacks.

3.8.3 Status Word Manipulation. To perform status word manipulation, the attacker impersonates an RT and attempts to intercept communication between two devices by sending a fake status word to the BC before the real one. This can be used to force a device into idle mode waiting for a word that never arrives. The timing is extremely precise and the attacker only has 14 μs to generate the fake word. Typically, it takes 20 μs to send messages to the bus, however, this does not render the attack impossible. Instead, the RT will process the fake status word as a command word (since command and status words have the same three Manchester code sync bits). This type of attack can target either `receive` or `transmit` commands.

As an example, consider a typical benign RT-BC protocol. First, the BC sends a `command: transmit (address)` word targeting the RT. Upon detection of this word, the attacker will transmit a malicious status word to the bus. This fake word has two purposes: the BC will read it as `status:OK` from the RT and await data, while the RT will read it as `command` and return to idle. The final result of status word manipulation is a frozen bus while the BC awaits data. Although this individual attack is not particularly threatening, it opens the bus to previously improbable attacks such as MITM.

3.8.4 Command Invalidation. In command invalidation, the attacker sends fake command words to the bus immediately following a benign one. This can be used to prevent regular data transmission and force an RT into idle mode. To implement this attack, the attacker must insert their word during the extended delay between command and status words. This means command invalidation is only possible during specific protocols. The legitimate BC may try and re-establish connection with the targeted RT due to the missing status word, so the attacker must use a command word that does not require any response such as `command:receive`.

As an example, consider a typical benign RT-RT protocol. First, the BC sends a `command: transmit (address0)` and `command: receive (address1)` for RT0 to send data to RT1. Following the second command word, the attacker inserts a false command word resetting RT1 to idle mode. Next, RT0 will send a `status` word followed by the `data` words. RT1 will not correctly process this data, because it has been reset by the attacker.

3.8.5 TX Shutdown. In a TX Shutdown attack, the attacker sends a `command:shutdown (address)` word to the main bus, disabling a particular RT. In a system that is not double or triple redundant, this attack simply disables the target. When multiple bus lines are active, the attack is more difficult, since the `status` word response on the backup bus may reactivate the target. In this case, the attacker must use `command:shutdown (all)` on both buses. Although that disables RTs on the network, it is impossible to go undetected, since communications will cease. The currently active BC will simply begin to reactivate devices as is necessary during normal operation. Alternatively, the attacker can target a particular RT on a backup bus and disable its connection. A backup bus TX shutdown can easily go undetected and completely disable an RT in the case of partial bus or system failure.

3.8.6 Desynchronization. During a desynchronization attack, the attacker sends a `command: resync (address)` word to the targeted RT, resetting its clock. This results in devices sending words at incorrect times causing message collisions and limiting the bus' transmission capabilities. The attack can be extended if the attacker follows the malicious `command:resync (address)` word with a `data` word containing new synchronization information. The targeted RT would no longer need to respond with a `status` word. By providing precise synchronization information, the attacker can control the flow of data on the compromised network. Furthermore, the attacker can send a `command:resync (all)` word, thus resetting the clock of every RT in the network.

3.8.7 Data Trashing. In a data trashing attack, the attacker inserts a command word during routine data transfer between two RTs. This results in the receiving RT trashing all the transmitted data words without processing them due to a state change upon reading the malicious command word. To implement this attack, the attacker waits for an RT-RT communication to enter the `data` transmission phase. A fake `command` is then inserted by the attacker before the final `data` can be sent. If the malicious command word does not require the reception RT to respond with a status word, then the BC will detect the error and re-attempt the data transfer.

As an example, consider a benign data transfer protocol between two devices, RT0 and RT1. The BC initiates the transfer by sending `command:receive (address1)` and `command:transmit (address0)`. Before the final data word from RT0 is sent, the attacker inserts a fake `command:transmit (address1)`. RT1 will switch to transmission mode, and the BC will assume the transfer was successful upon reception of a `status:OK` from RT1. The BC will think the transfer was successful, however, the `status:OK` was actually a response to the malicious command word.

3.8.8 Data Word Corruption. In a data word corruption attack, the attacker attempts to insert fake data words during a benign transmission and have them processed by the receiving device. If the attacker has an intimate understanding of the purpose for each RT, then data word corruption can be used to manipulate sensors and their display readings. The attack exploits the time gap between a `transmit` command and the `status` word response. The attacker attempts to insert a fake `status:OK` followed by manipulated `data` words. If successful, then the targeted RT will receive and process whatever data the attacker wants, and the system will be unaware of the intrusion.

As an example, consider a benign data transfer protocol between two devices, RT0 and RT1. The BC initiates the transfer by sending `command:receive (address1)` and `command:transmit` `(address0)`. The attacker inserts a false `status:OK` before RT0 can respond normally. RT0 will enter `idle` mode, and the attacker is free to send the manipulated data words. RT1 will process them normally and respond with its own `status:OK` word. The BC will assume the transfer was successful and continue normal operations.

3.8.9 Attack Indicators. There are two categories of attack indicators on MIL-STD-1553, the message timing and contents. Using timing as an indicator presents several issues. Different avionics platforms have distinct configurations that are network specific. As such, each system has varying harmonic rate groups and polling rates, which makes using this feature, in general, more difficult. Furthermore, even within the same platform installing a different system or weapon would greatly affect the network communication protocols. Finally, many MIL-STD-1553 networks use manually configured communication cycles, which vary greatly from system to system.

There are three types of message content indicators that can be used by an IDS to identify an ongoing attack scenario: interrupted bus communications, intercepted protocol, and interrupted RT functionality.

Random word generation, status word manipulation, and incorrectly executed MITM attacks cause interrupted bus communications. This either takes the form of message collisions as a result of multiple words being transmitted at the same time or freezing with no word transmission. An IDS can identify this scenario primarily based on reoccurring messages with similar signatures (i.e., multiple words from the same RT address, no words on the bus).

Data word corruption, data trashing, and MITM attacks work to intercept an existing protocol to insert data without stopping the operation of MIL-STD-1553. These attacks can be identified by the IDS through either a pattern anomaly or the content of the data being transmitted. When an attacker seeks to modify data words maliciously, the content will differ significantly (i.e., non-pilot RT sending fire weapon command) from observed patterns.

Command invalidation, TX shutdown, and desynchronization interrupt the functionality of a specific RT without affecting the bus as a whole. This attack can be identified by an IDS based on the content of the suspicious words. Malicious commands and genuine ones differ based on the addresses of devices involved and the command itself (i.e., RT0 should never be shutdown) being transmitted.

4 SIMULATION SYSTEM

Creating an effective simulator for MIL-STD-1553 has the following challenges. First, the system should be able to fully represent the maximum capacity of the network (i.e., 31 RTs). Simulating this with purely hardware components is difficult and often prohibitively expensive. Second, it must be able to accurately represent attack scenarios of different types and categories. This allows for existing as well as yet undiscovered attacks to be implemented in the simulator and tested. Finally, it should reflect real-world data that might be sent through the system while installed on an avionics platform. Message contents are a robust indicator of attacks that are resilient toward varying network configurations and communication cycles. The simulator component of OD1NF1ST presents significant advantages over existing semi-physical and software systems designed to replicate the communication bus.

(1) We can produce communication data over nearly unlimited time periods with greater detail compared to existing methodologies. The system provides high-fidelity data containing exact timestamps and word contents without the need for user interaction.
(2) The software could also be used to facilitate bus scheduling simulations and assist future research toward optimizing transmissions on aircraft using MIL-STD-1553. Furthermore,

it could be used to analyze the performance of certain devices (e.g., oil engine sensor reading).

(3) Our software package allows for the effortless implementation of advanced cyber-attack scenarios, and we provide exploits of 10 different MIL-STD-1553 vulnerabilities covering all the three system security categories. This is an improvement over other methods that only integrate 2–4 intrusion scenarios and focus on specific subsets of system security paradigms. Furthermore, as research continues in this area, new attacks can be easily integrated into the existing system. This provides researchers with a future-proof way to collect up-to-date data for training any MIL-STD-1553-based IDS.
(4) The simulator can be simply linked with other software to create an efficient and simple pipeline. OD1NF1ST can begin with raw flight simulation data and end with real-time classification of word sequences.

Next, we will provide an overview of the simulation package in terms of its architecture, implementation, and pipeline integration.

4.1 Simulation Software

OD1NF1ST has two main modes of operation: terminal-based data collection, and web-based interactive flight and attack simulation. A repository containing the terminal-based mode, interactive display, sample datasets, and attack scenarios can be found on the project Github.[1] An overview of the system architecture is described in Figure 4. The simulator consists of three main components:

(1) The transmission layer replicates the raw data transfer between devices connected to MIL-STD-1553.
(2) The event system controls how devices coupled to the bus react based on the information being transmitted.
(3) The Interactive Display package can be used to host a live interactive website and simulate real-time attack and defense scenarios.

4.2 Transmission Layer

Our software simulator uses two different systems to imitate the data transmission (physical and message layers) of MIL-STD-1553. The physical layer is replicated using the **User Datagram Protocol (UDP)** computer networking protocol. UDP allows devices to send datagrams to other hosts on an IP network without the need for prior communications to create a data line. The message layer is simulated using a software-based proxy manager. The manager controls the simulator's proxy servers along with writing and implementing their policies and filtering resource requests.

4.2.1 The Proxy Manager. The proxy manager is a process on the simulator used to administrate and control any communications sent through UDP. It contains objects that represent the physical bus itself as well as any devices that would be coupled to it. The default system configuration contains a maximum of 32 devices: 30 RTs, 1 BC, and 1 BM. Backup BCs or additional BMs to replace a faulty BC can be added on demand. Connection between multiple MIL-STD-1553s through communication-configured RTs is also supported.

4.2.2 Device. Each device runs as an independent thread and contains a proxy to its respective device object. A single unified class allows for easy conversion of a backup BM to BC or BC to RT (as is possible on MIL-STD-1553) without reconnecting to the bus. Each object contains device information such as:

[1]OD1NF1ST Repository. https://github.com/MichaelWrana/OD1NF1ST.

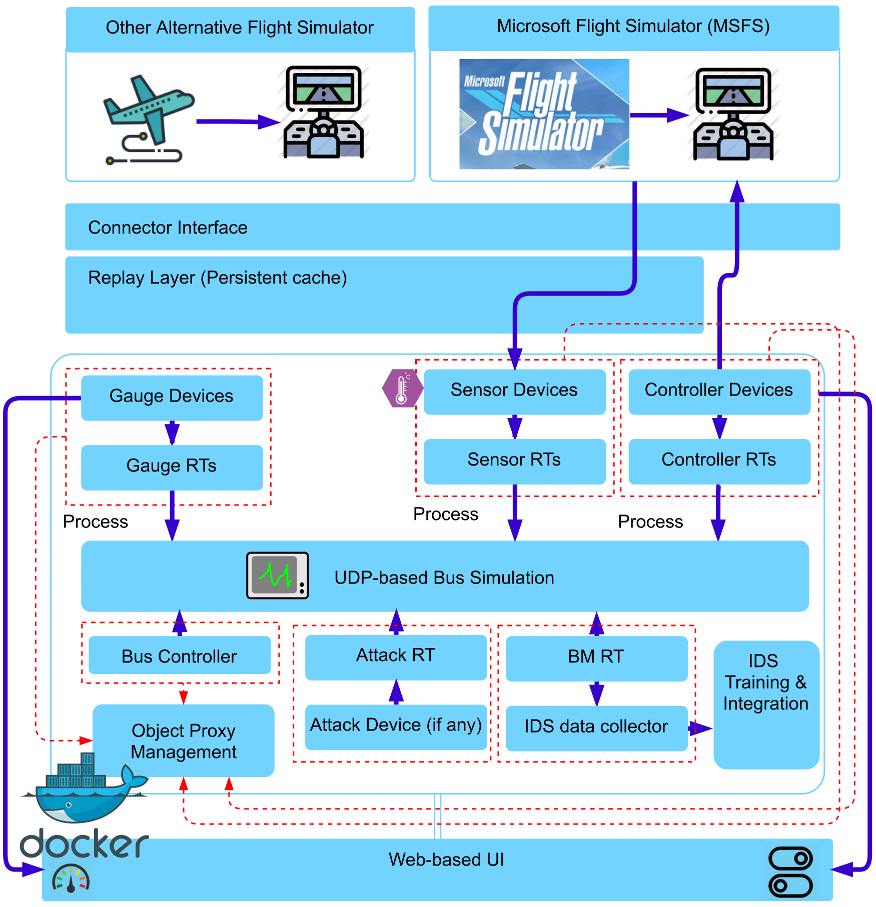


Fig. 4. Overview of integrated pipeline to process OD1NF1ST data.

- Mode: `RT | BC | BM`
- State: `transmit | receive | idle | off`
- A reference to bus objects this device is coupled to
- A queue of messages transmitted over the network but not yet processed

Each device proxy also contains two UDP sockets: one for message reception and another for transmission. Every word transmitted to the bus is read by every device through its reception socket and added to its individual message queue. The devices process words one by one in their queue and react accordingly using the event system.

4.2.3 Bus. This object represents the main data line and every device is coupled to it through UDP sockets. Multiple buses can be created and coupled to each device to simulate doubly and triply redundant systems. The bus object contains system information such as:

- A proxy for communication over the UDP network
- A dictionary of references to all devices connected to the bus.
- Methods for sending data to all other devices and receiving transmissions from another device.

Table 4. Overview of the OD1NF1ST Event System and Related Communication Protocols

Event Name	Description	Protocols
RT_REC_CMD	RT reads a `transmit` command word and `address == this.address`	BC-RT RT-RT Broadcast
RT_TRS_CMD	RT reads a `receive` command word and `address == this.address`	RT-BC RT-RT Broadcast
RT_REC_DATA	RT reads a data word and `this.state==receive`	RT-RT BC-RT Broadcast
BC_REC_DATA	BC reads a data word after sending `command:transmit` to an RT	RT-BC Broadcast
BC_REC_STS	BC reads a status word	BC-RT

4.3 Event System

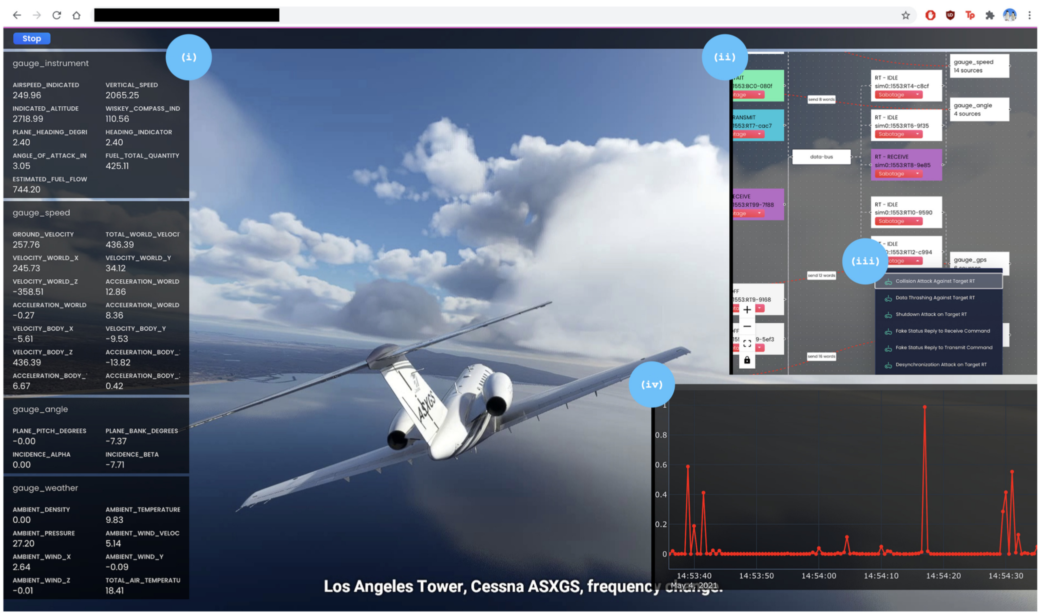
When a message is transmitted through the UDP network and received by a device, it must be processed and the device should react appropriately based on the MIL-STD-1553 protocol. Every word is always read by each device connected to the network, however, the devices will behave differently depending on their current state and mode. To replicate this, OD1NF1ST uses an event system that doubles as a way to monitor and train the IDS. An example of some events in the system are described in Table 4.

4.4 Flight Simulator Pipeline

The OD1NF1ST simulator package can also be integrated with flight simulation software and an IDS to create a full data pipeline. Here, we will summarize our system for linking the simulator to its inputs and outputs. An overview of the pipeline can be seen in Figure 4.

- We first collect flight data so the content of each word being transmitted through the simulated bus is realistic. The main tool used to collect data is **Microsoft Flight Simulator 2020 (MSFS)**[2] and the **SimConnect (SC)**[3] Python package. MSFS creates realistic aircraft flight scenarios and SC allows for real-time collection of data with matching video.
- The MSFS recording can feed directly into the simulator, which has a pre-programmed cyclic communication scenario similar to ones found on military aircraft. The communication pattern can be altered by the user if required. The simulator produces either real-time recordings (for the interactive display) or packs them into a **JavaScript Object Notation (JSON)**[4] file for saving and processing. Each file contains objects representing words with a timestamp, payload details, and anomaly labels. Each file contains an attack type label but is not guaranteed to have an intrusion. The length of recordings for each JSON is user-dependent with a default of 3–5 min.
- Next, we process the JSONs and convert them to integer sequences to feed the IDS. The data can be split for cross-validation and performance analysis or used exclusively for training. Once trained, the network can be used to detect intrusions on real-time simulator data

[2]MSFS Website. https://www.flightsimulator.com/.
[3]SimConnect API. https://pypi.org/project/SimConnect/.
[4]JavaScript Object Notation. https://www.json.org/json-en.html.


[redacted]

gauge_instrument

AIRSPEED_INDICATED 249.96
VERTICAL_SPEED 2065.25
INDICATED_ALTITUDE 2718.99
WISKEY_COMPASS_IND 110.56
PLANE_HEADING_DEGRI 2.40
HEADING_INDICATOR 2.40
ANGLE_OF_ATTACK_IN 3.05
FUEL_TOTAL_QUANTITY 425.11
ESTIMATED_FUEL_FLOW 744.20

gauge_speed

GROUND_VELOCITY 257.76
TOTAL_WORLD_VELOCI 436.39
VELOCITY_WORLD_X 245.73
VELOCITY_WORLD_Y 34.12
VELOCITY_WORLD_Z -358.51
ACCELERATION_WORLD 12.86
ACCELERATION_WORLD -0.27
ACCELERATION_WORLD 8.36
VELOCITY_BODY_X -5.61
VELOCITY_BODY_Y -9.53
VELOCITY_BODY_Z 436.39
ACCELERATION_BODY_ -13.82
ACCELERATION_BODY_ 6.67
ACCELERATION_BODY_ 0.42

gauge_angle

PLANE_PITCH_DEGREES -0.00
PLANE_BANK_DEGREES -7.37
INCIDENCE_ALPHA 0.00
INCIDENCE_BETA -7.71

gauge_weather

AMBIENT_DENSITY 0.00
AMBIENT_TEMPERATURE 9.83
AMBIENT_PRESSURE 27.20
AMBIENT_WIND_VELOC 5.14
AMBIENT_WIND_X 2.64
AMBIENT_WIND_Y -0.09
AMBIENT_WIND_Z -0.01
TOTAL_AIR_TEMPERATU 18.41

Los Angeles Tower, Cessna ASXGS, frequency change.

Fig. 5. OD1NF1ST interactive web display containing (i) Real-time flight data, (ii) Bus simulation overview, (iii) Attack launching panel, and (iv) Intrusion detection system. One can launch attacks on arbitrary system components and observe the IDS' response in real-time.

through the interactive display component (Section 4.5). Details on the conversion process and our IDS can be found in Sections 6 and 7.

4.5 Interactive Display

The interactive display is an appealing feature of OD1NF1ST. In this system, a live simulation of a pre-planned flight route is hosted on a website, as seen in Figure 5. The transmissions of MIL-STD-1553 along with raw flight data are displayed to the user in Figures 5(i) and 5(ii). Here, the user acts as the "hacker" and can launch any of the pre-defined attack scenarios against the bus or any connected device at any time in Figure 5(iii). The impact of such an attack can be visualized in the device communications and data display. An IDS is actively running and will attempt to detect the intrusions launched by the user. The identification of malicious messages is also shown to the user to see if their attacks were effective in beating the IDS in Figure 5(iv). Such a system can be hosted and used to crowd-source attack scenarios that defeat the IDS. This can be used to further improve both the complexity and capability of the attacks as well as the defense mechanisms.

5 INTRUSION DETECTION SYSTEM

In this section, we will summarize the components of our IDS architecture including runtime reduction, the true-skip mechanism, and REINFORCE policy optimization.

Many existing IDS' rely chiefly on word timings to classify messages, which presents some potential issues. A system trained on a specific timing pattern or network structure will become inaccurate when configuration changes occur. This is common on aviation platforms, for example, the F-35 supports over 10 different types of missile systems [18]. Furthermore, if the system remains the same but the harmonic rate groups change, then the IDS will likely become ineffective.

Therefore, OD1NF1ST is designed around using exclusively word contents and metadata for classification, which presents advantages over timing-based systems. OD1NF1ST does not rely on platform-specific behaviors or configurations to train and is capable even during the quickly changing situations present in aircraft flight. Furthermore, changes to polling rates or harmonic groups will have minimal impact on its performance.

5.1 Runtime Reduction

One factor that defines the effectiveness of an IDS for MIL-STD-1553 other than raw performance is runtime. Many avionics platforms operate with limited computing power and every acceptable model must abide by strict resource constraints. Although RNNs are well-suited to the task of sequential anomaly detection [28], they use significantly more resources per data point compared to other network architectures. This problem is well-known and methods exist to reduce the computational cost of RNNs such as the skipRNN and leapLSTM [7, 23].

In the case of a skipRNN, instead of simply computing the LSTM or **gated recurrent unit (GRU)** output an update is applied at each timestamp t using the following formula:

$$s_t = u_t \cdot S(s_{t-1}, x_t) + (1 - u_t) \cdot s_{t-1},$$

where s_t is the state, S is the LSTM/GRU output function, x_t is the data, and u_t is the binary update gate. Although the state updates are skipped, the runtime of this model actually increases due to the additional calculations needed to evaluate this equation. The original LSTM/GRU output must be calculated (even if not used) to make the network differentiable for training.

In the case of leapLSTM, data points are skipped rather than state updates. The output of the LSTMs at each timestamp t are calculated using the following formula:

$$h_t = \begin{cases} S(h_{t-1}, x_t) & \text{if } d_t = 0, \\ h_{t-1} & \text{if } d_t = 1, \end{cases}$$

where h_t is the hidden state, S is the LSTM output function, x_t is the data, and d_t is the binary update gate. Although a leapLSTM allows for true skipping of the computational cost of processing data points, the method for calculating d_t is very expensive. For each sequence the output of a reversed RNN, LSTM$_r$ and three-layer CNN are passed through a two-layer multi-layer perceptron (MLP) using the following formulae:

$$s_t = \text{ReLU}(W_1[x_t; \mathbf{f}_{follow}(t); \mathbf{f}_{precede}(t)] + b_1),$$

$$\pi_t = \text{softmax}(W_2 \cdot s_t + b_2),$$

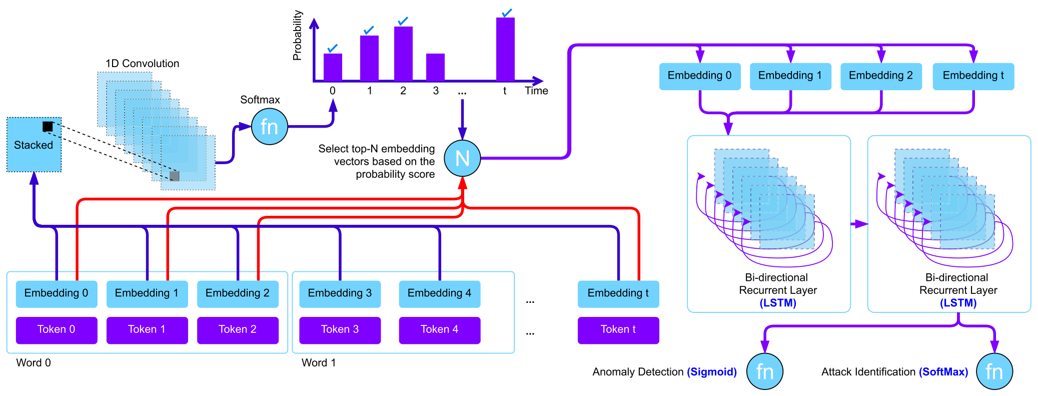
where s_t is the MLP node output, W_i are the node weights, b_i is the bias, and π_t is the probability of setting d_t to 0 or 1. $\mathbf{f}(t)$ is computed according to the following equation applied to the data preceding and following t:

$$\mathbf{f}(t) = \begin{cases} [\text{LSTM}_r(t+1); \text{CNN}(t+1)] & \text{if } t < T, \\ \mathbf{h}_{end} & \text{if } t = T, \end{cases}$$

where T are the total number of timestamps and h_{end} is the features when the sequence reaches the end.

5.2 True-skip Mechanism

To address the critical resource issues, we propose a network that consists of three components: skip mechanism, recurrent network, and word classification. A diagram of the system architecture can be found in Figure 6.



1D Convolution
Stacked
Softmax
fn
Probability
Time
0
1
2
3
...
t
Select top-N embedding vectors based on the probability score
N
Embedding 0
Embedding 1
Embedding 2
Embedding t
Bi-directional Recurrent Layer (LSTM)
Bi-directional Recurrent Layer (LSTM)
Embedding 3
Embedding 4
Token 0
Token 1
Token 2
Token 3
Token 4
Token t
Word 0
Word 1
Anomaly Detection (Sigmoid)
Attack Identification (SoftMax)

Fig. 6. OD1NF1ST intrusion detection system architecture. Given a list of words being observed in a large sliding window, the IDS first converts each word into sub-tokens and maps each token to its respective embedding vector. Next, a 1D Convolution operation coupled with the softmax activation function estimates a probability $\in [0, 1]$ for each token. Then, a top-N operation selects the embedding vectors based on their respective probability score (in this example $N = 4$). Finally, a two-layer Bi-Directional LSTM model is used for final predictions. It is noted that the top-N operation is non-differentiable and thus uses a REINFORCE Algorithm for training.

The network takes as input a sequence of configurable but fixed length containing processed words from the OD1NF1ST simulator. The sequence is embedded using a layer with 8 nodes, which is sufficient to include the 256 words in the MIL-STD-1553 dictionary, since each integer is between 0 and 255 ($2^8 = 256$).

The sequences are then passed through a convolutional layer, which attempts to predict the words that are most likely to be malicious. This layer provides a rough guess as to whether messages could be part of an attack scenario. Rapidly identifying possible intrusions is important for MIL-STD-1553 data, since words are transmitted very quickly (typically 140–160 μs between messages). A large amount of data must be analyzed efficiently by the IDS and the runtime of such a simple network is very low. The output at each layer is computed as

$$C_k^{(l)} = \text{softmax}\left(\sum_c W_k^{(l),c} \cdot X^{(l-1),c} + B_k^l\right),$$

where k is the filter size, l is the layer, c denotes channel number, X is the data, W is the weight, and B is the bias. In our network, this can be simplified, since we use a single layer and filter size of 1 to produce a single prediction for each word:

$$C = \text{softmax}\left(\sum_c W^c \cdot X^c + B\right).$$

Next, we select the top n words with the highest probability of being intrusions and pass them to the next layer. Since only n words from the whole sequence are passed to the recurrent network, we save major computational resources in this step. Word selection is non-differentiable and thus cannot be trained. To solve this problem, we apply a reinforcement learning algorithm (described in Section 6.2) to support backpropagation. REINFORCE trains the network on if its prediction about whether a message is anomalous or not is correct. We only train top-n using binary anomaly labels.

Next, we use two Bi-directional LSTM layers, one with 32 nodes followed by another with 128. This part of the IDS is used as a more exhaustive predictor to produce the most accurate guess possible. The architecture of the inner model was chosen for two main reasons.

(1) LSTMs are well suited to prediction tasks for sequential time-series data [41, 50]. The ability for nodes to remember past information despite time gaps of unknown duration is critical for any MIL-STD-1553 IDS.
(2) Bi-directional RNNs are generally more efficient and have better accuracy than single direction models for intrusion and anomaly detection [29, 42]. It simulates a reverse-lookup learning process.

The recurrent network is trained using both binary anomaly detection and attack classification. The output of each layer for the recurrent portion of the network is computed as

$$h_t = o_t \circ \sigma(f_t \circ c_{t-1} + i_t \circ \tilde{c}_t),$$

where h_t is the hidden state, $o_t, f_t, i_t, \tilde{c}_t$ are the output gate, forget gate, update gate, and cell input activation vector, respectively. $\circ$ is the Hadamard product. Each gate's output is calculated as

$$g_t = \sigma(W_g x_t + U_g h_{t-1} + b_g),$$

where W_g, U_g, and b_g are the weight matricies and bias vector paramters learned during training. x_t is the data and h_{t-1} is the hidden state.

The output of the recurrent layers are passed to a single fully-connected layer with ReLU activation followed by two output layers. One layer produces binary anomaly detection using Sigmoid activation and the other guesses the precise attack class with Softmax activation.

5.3 REINFORCE for Policy Optimization

To make the model parameters trainable, we apply a REINFORCE algorithm to the top-n selection based on the convolutional layer output. The objective function of our network is based on the sum of loss in three different categories:

$$\mathcal{L}_{\text{OD1N}} = \mathcal{L}_{\text{ANOM}} + \mathcal{L}_{\text{ATTK}} + \mathcal{L}_{\text{TOPN}},$$

where $\mathcal{L}_{\text{ANOM}}$ is from binary anomaly detection, $\mathcal{L}_{\text{ATTK}}$ is from attack classification, and $\mathcal{L}_{\text{TOPN}}$ is from the most relevant word selection. To compute the loss for binary anomaly detection and categorical attack classification, we use cross-entropy:

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log\,(1 - \hat{y}_i),$$

where θ are the model parameters, N is the output size, $\hat{y}_i$ is the ith scalar value in the model output, and y_i is the corresponding target value. To find $\mathcal{L}_{\text{TOPN}}$, we use reinforcement learning and provide a binary reward to the network if the selected words are actually anomalies. The objective function for policy gradients is defined as

$$J(\theta) = \mathbb{E}\left[\sum_{t_l=0}^{T_l-1} r_{t_l+1}\right],$$

where θ is the model parameters, t_l and T_l are the reinforcement learning current and terminal time, respectively. r_{t+1} is the reward for performing an action (a_t) in a state (s_t) based on the reward function (R). So, $r_{t+1} = R(a_t, s_t)$. We optimize the policy by taking the gradient ascent using the partial derivative with respect to θ. Thus, the policy gradient function is

$$J(\theta) = \nabla_\theta \sum_{t_l=0}^{T_l-1} \nabla_\theta \log \pi_\theta (a_{t_l} | s_{t_l}) G_t,$$

where θ are the model parameters to our policy algorithm, π_θ. ∇_θ represents the gradient of loss, a_{t_l} and s_{t_l} represent the action and state, and G_t is the discounted cumulative reward. Although each of the words is associated with a time, the probability of each one being malicious is calculated in parallel. Thus, there is only a single timestamp in the reinforcement algorithm. Additionally, by assuming that a correct anomaly prediction will yield a reward of 1 and 0 and vice versa, the objective function can be simplified to

$$J(\theta) = \begin{cases} \nabla_\theta \log \pi_\theta (a|s) \times 1, & \text{if anomaly prediction is correct,} \\ \nabla_\theta \log \pi_\theta (a|s) \times 0, & \text{otherwise.} \end{cases}$$

The combined loss equation allows the network to train on its predictions about which words are relevant in each sequence and which sequences contain malicious activity.

6 EXPERIMENT

To evaluate OD1NF1ST, we generated a dataset using the simulation package and compare our IDS to other state-of-the-art methods. We also perform cross-validation on the attack types to verify our IDS' ability to independently detect each different type of intrusion. We additionally evaluate the network's ability to detect intrusions without training. Finally, we test our network on a Raspberry Pi to simulate the embedded systems used on avionics platforms.

6.1 Data Processing and Environment

To generate data for the network, we followed the pipeline as described above in Section 4. First, using MSFS, two flight paths were established. The plane used for these trips was a Cessna Citation CJ4. The first was from (Anonymous) to (Anonymous) and takes approximately 8:30 h to complete. The second was from (Anonymous) to (Anonymous) and takes approximately 1:30 h to complete. The simulator was run twice to obtain two unique CSV files containing detailed flight information for the whole journey.

Next, the CSV files were processed through the OD1NF1ST simulator to generate two sets of JSON files. The first, labeled cruise, is larger and contains attacks while the planes were in-flight. The second, labeled takeoff, is shorter and attacks were attempted during takeoff and landing. Overall, the dataset contains 42 h of continuous MIL-STD-1553 communications, which is 250× longer than any previously published method.

The data was passed through our simulator based on a system architecture and transmission schedules summarized in Figure 7. We simulate the network with 12 RTs, 1 BC, and 1 BM. There are three groups of producers, each polled at a rate of 120 Hz. The engine control group contains flight data associated with the aircraft's engines such as acceleration, speed, fuel consumption, and engine status. The sensor group represents external data collection from the aircraft such as the GPS, Thermometer, and Altimeter. The flight control group contains directional information such as Gyroscope data, angle of attack, angle of incidence, and heading. This information is periodically funneled to the cockpit group, which represents the aircraft's electronic gauges and pilot **Heads-up-display (HUD)**.

In the cruise dataset there are a total of 1,662,752 words with an attack presence of 1.08%. In the takeoff dataset there are a total of 256,544 words with an attack presence of 1.18%. A detailed summary describing the presence of each type of attack can be found in Table 5. We included a

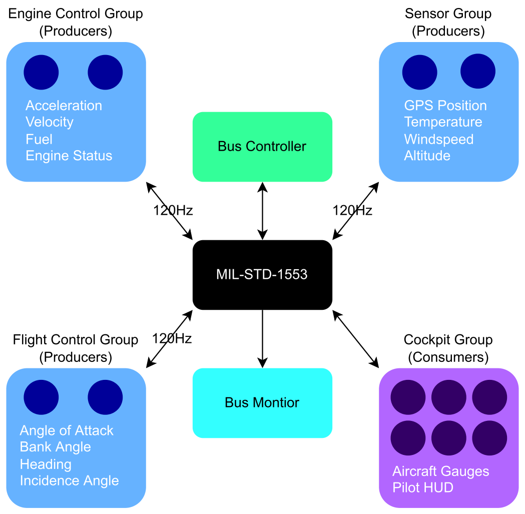


Fig. 7. System architecture simulated using OD1NF1ST. There are three producer groups each with 2 RTs being polled at a rate of 120 Hz. They represent the engines, flight controls, and external sensors of the aircraft. There is one consumer group of 6 RTs representing the aircraft gauges and pilot HUD. The network also has an associated BM and BC.

Table 5. Summary of the Attacks in Our Datasets

Attack Type	Instances (Cruise)	Instances (Takeoff)
Random Word Generation (RT)	7,900	1,398
Random Word Generation (Bus)	40	47
Status Word Manipulation (Transmit)	5,784	298
Status Word Manipulation (Receive)	2,765	948
Man in the Middle	416	122
Command Invalidation	17	26
TX Shutdown	57	46
Desynchronization	168	59
Data Trashing	616	43
Data Word Corruption	170	53
Total Malicious	17,933	3,040
Total Benign	1,662,752	256,544
Presence	1.08%	1.18%

similar number of each intrusion in the dataset, however, the number of words required to execute each type varies significantly. Thus, there is significant deviation in the number of anomalies associated with each attack type.

We implemented our proposed solution in Python relying on various libraries including: Pandas[5] and Numpy[6] for processing the bus traffic data; combined with Scikit-Learn,[7] TensorFlow,[8]

[5]Pandas API. https://pandas.pydata.org/.
[6]Numpy API. https://numpy.org/.
[7]Scikit-Learn API. https://scikit-learn.org/.
[8]TensorFlow API. https://www.tensorflow.org/.

and Keras[9] for applying the deep learning techniques. Two hardware devices were used in this experiment. A Windows 10 machine with: Intel 8-core i7-7700k CPU @ 4.20 GHz, Nvidia GeForce GTX 1080Ti GPU 11 GB VRAM, and 64 GB RAM was used to generate flight data and simulate MIL-STD-1553. A Docker platform deployed on a VM running Ubuntu 16.04.6 with: 4-core Intel Xeon Gold 2.3 @ 3.90 GHz, 200 GB RAM was used for data processing and the IDS.

6.2 Evaluation Methods

We implemented a cross-validation approach for the out-of-sample evaluation of our solution and used a 10% training, 90% validation split. Using a small proportion of training data avoids overfitting the network. There was no guarantee of a specific number of intrusions in either set to better approximate real-world scenarios, where they would not be guaranteed or at regular intervals.

Our proposed model is compared with the following networks. (i), (ii), and (iii) standard RNN, LSTM, and GRU, respectively; (iv) and (v) skipLSTM and skipGRU, respectively [7]; (vi) leapLSTM [23], (vii) Naive Bayes [21], (viii) Markov Chain [47], and (ix) Histogram Comparison [16]. For network (i), we use two Recurrent layers each with 128 nodes. Networks (ii) and (iii) contain a single layer with 256 nodes. Networks (iv), (v), and (vi) all contain one layer with 128 nodes.

To evaluate each model's effectiveness, we use seven metrics: (i) Precision, (ii) Recall, (iii) F1, (iv) **Area Under ROC Curve (AUC)**, (v) Sparse Categorical Accuracy (SCA), (vi) Runtime (seconds), and (vii) Proportion of messages analyzed.

The first three metrics (precision, recall, F1) focus on the mistakes made by the network, and how impactful they are. Precision measures false positives (i.e., model detects an attack when there is none), recall measures false negatives (i.e., model fails to detect an ongoing attack), while F1 is the harmonic mean of precision and recall. The dataset is highly imbalanced with very few malicious messages, so these metrics are chosen over raw model accuracy. We also consider AUC, the area under the **receiver operatic characteristic (ROC)** curve created from the false positive rate and true positive rate. This metric measures how well the model can distinguish between the benign and anomalous message classes. Both F1 score and AUC must be considered for binary anomaly detection performance. Due to the large number of benign messages compared to malicious, a network that simply guesses benign every time can achieve F1 scores >0.98 [40]. In the case of random guessing, AUC will reveal the actual performance of the network. To evaluate each model's performance in the attack classification task, we use SCA, which measures the number of correct guesses compared to the total. The three existing MIL-STD-1553 IDS' [6, 21, 47] evaluated for comparison do not perform attack classification, so no SCA metric is reported.

These types of intrusion detection systems provide no overhead to the operation of the communication bus, so we evaluate the efficiency of each method based on what proportion of messages could be classified during the real-time operation of MIL-STD-1553. This percentage is based on the estimated inter-message time gap [9]. It is also important to consider the model's performance relative to runtime. A network that achieves a slightly higher accuracy over a longer time is not as applicable to MIL-STD-1553.

6.3 Results

We compare the performance metrics as applied to the validation dataset for each network on both the long and short flight datasets in Table 6. It showed that for the cruise dataset, OD1NF1ST displayed excellent precision (0.95) that was comparable to other IDS' (0.91–0.98), with the exception of Leap-LSTM, which was moderate (0.8087), and naïve Bayes, which was poor (0.7089). The recall was overall lower and much more variable, however, GRU and OD1NF1ST outperformed

[9]Keras API. https://keras.io/api/.

Table 6. Performance and Runtime Comparison of OD1NF1ST with Other IDS' on Takeoff and Cruise Datasets

		Evaluated Deep Learning Solutions						Existing IDS Solutions for MIL-1553			
Solution		RNN	LSTM	GRU	skip-LSTM [7]	skip-GRU [7]	leap-LSTM [23]	He [21]	Genereux [16]	Stan [47]	OD1NF1ST
	Precision	0.9824	0.9822	0.9182	0.9676	0.9429	0.8087	0.7089	0.9837	**0.9896**	0.9530
	Recall	0.6500	0.6564	0.7530	0.6228	0.6692	0.3786	**0.9958**	0.5132	0.3779	0.7579
Dataset 1	F1	0.7824	0.7869	0.8274	0.7578	0.7828	0.5158	0.7925	0.6707	0.5469	**0.8443**
(Cruise)	AUC	0.8944	0.9568	0.9622	0.8988	0.9613	0.8574	0.9549	0.5817	0.6303	**0.9648**
	SCA	0.9349	0.9347	0.9381	0.9330	0.9366	0.9126	—	—	—	**0.9476**
	Runtime (sec)	500	399	385	1382	1388	1666	405	555	504	379
	Proportion	53.78%	67.40%	69.85%	19.46%	19.37%	16.14%	66.40%	48.45%	59.10%	**71.11%**
	Precision	0.6977	0.9821	0.9437	0.8108	0.9592	0.9844	**0.9982**	0.9741	0.9809	0.9322
	Recall	0.3626	0.7185	0.7788	0.6286	0.7166	0.7013	0.7830	0.4830	0.3919	**0.8018**
Dataset 2	F1	0.4772	0.8299	0.8533	0.7412	0.8204	0.8191	0.8605	0.6388	0.5587	**0.8621**
(Takeoff)	AUC	0.7489	0.9557	0.9526	0.9361	0.9508	0.9518	0.7907	0.5304	0.5393	**0.9590**
	SCA	0.8696	0.9305	0.9307	0.9081	0.9296	0.9305	—	—	—	**0.9398**
	Runtime (sec)	67.8	57.8	56.6	178.1	172.4	263.7	70.3	84.3	65.7	56.0
	Proportion	57.43%	67.37%	68.79%	21.86%	22.59%	14.77%	55.39%	49.27%	59.27%	**69.53%**

the other IDS' (0.753 and 0.7579, respectively), while OD1NF1ST displayed the highest F1, AUC, and SCA scores (0.84, 0.96, and 0.95, respectively). OD1NF1ST also had the lowest per-message classification time and would be able to analyze 71% of real-time data. On the Takeoff dataset, OD1NF1ST displayed similar performance with top F1, AUC, and SCA scores (0.8621, 0.959, and 0.94, respectively).

Although our network did not independently achieve the highest precision and recall, we found the best balance between the two. In a real-world scenario, poor precision could cause major operational setbacks as ground crews try to find and solve a problem that does not exist. Poor recall might result in a compromised system going un-detected for longer periods of time, causing information leaks or physical damage and loss of life (i.e., an unauthorized rocket launch). OD1NF1ST achieving the highest AUC demonstrates that the model did not guess benign for every sequence (high F1 scores without correctly performing the task). Furthermore, OD1NF1ST scored the highest in SCA compared to other deep learning solutions. This metric is important, because it provides actionable intelligence to ground and flight crews who can act based on the type of attack believed to be occurring in the system. Overall, OD1NF1ST had the highest overall F1, AUC, and SCA scores, indicating excellent performance, accurate predictions, and valuable information.

Most of the networks achieved high precision but struggled with recall in both scenarios, which may reflect the sparse intrusions present in the dataset. Furthermore, existing recurrent time-saving mechanisms did not translate well to a dataset with very few anomalous messages, which resulted in poor performance and runtime on the skip and leap networks.

Of the comparison models, traditional RNN performed adequately in the Cruise dataset, albeit with the poorest F1 score. However, RNN performance markedly declined on the smaller Takeoff dataset, suggesting that pre-trained weights should be passed to all networks in a real-world scenario. If the model is trained on almost purely benign data, then it will become too sensitive and detect attacks when none are present. This is also confirmed by Onodueze et al. [21] who found that a network only trained using benign messages was incapable of correctly identifying anomalies.

Finally, while OD1NF1ST outperformed the other neural networks, they generally achieved, with the exception of the RNN, acceptable performance on both datasets. This demonstrates that even when faced with a limited number of malicious messages, machine learning is an effective strategy for MIL-STD-1553-based intrusion detection.

6.4 Choice of *n*

The optimal value of *n* varies greatly depending on the data used, model configuration, and desired output. To provide some heuristics for choosing an appropriate *n*, we evaluated model performance

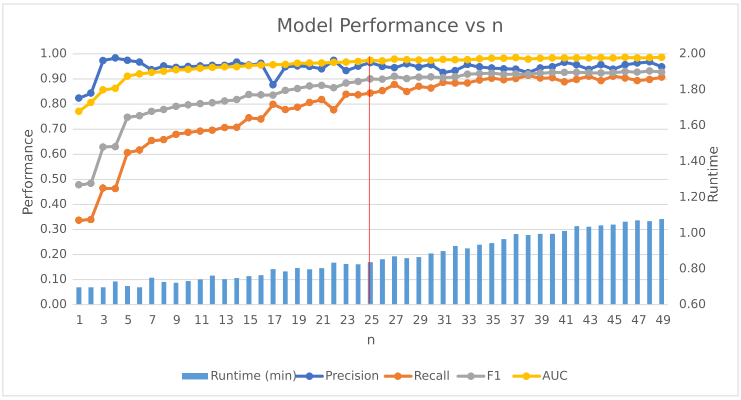


Fig. 8. Comparison of (i) Runtime, (ii) Precision, (iii) Recall, (iv) F1, and (v) AUC with top n selection of malicious messages. Our chosen optimal value of n is highlighted in red.

versus different values of n. We trained and tested OD1NF1ST on Dataset 2 (Takeoff) with all other parameters remaining consistent between $n = 1$ and $n = 50$. We compared runtime with Precision, Recall, F1, and AUC and the results from this test are in Figure 8.

We found that as n increased, the model runtime increased linearly, while the other evaluation metrics increased logarithmically. As such, there is a middle point where the linear increase in runtime outweighs the logarithmic increase in performance. We recommend three values of n based on the goals of optimizing runtime, performance, or a balance between the two:

$$n = \begin{cases} n = 0.15l, & \text{for minimal runtime,} \\ n = 0.35l, & \text{for balanced results,} \\ n = 0.4l, & \text{for best performance,} \end{cases}$$

where l is the total sequence length after embedding. However, in general, it is best to determine the optimal value for each deployed platform independently.

6.5 Attack Cross-validation

As another method to evaluate the effectiveness of our IDS, we partitioned the dataset into individual attack types and compared the performance. This allowed us to test if OD1NF1ST can identify the anomalous behaviors associated with each individual attack type. Table 7 shows the results of attack cross-validation on the cruise dataset. For training and evaluating the model, we only consider binary anomaly detection metrics, since each dataset only contains one type of attack.

The cross-validation table shows that our network can independently detect each of the different proposed intrusions. The most difficult attacks to detect overall were command invalidation, TX shutdown, desynchronization, and data trashing. TX Shutdown and command invalidation are both attacks with very few examples in the generated dataset (<0.01%), which may result in poor training and validation capabilities.

TX shutdown and desynchronization are challenging to identify, because a malicious party sending command words to the targeted RT is no different from the genuine BC transmitting the same message. Data trashing and command invalidation both share a similar attack pattern, whereby the attacker inserts a malicious command word to alter the state of a device on the network. These are difficult to detect, because no collisions occur and messages are sent in the same way during a regular transmission. The main difference between these attacks and benign protocols is the way

Table 7. Attack Cross-Validation Using OD1NF1ST

Attack Type	Precision	Recall	F1	AUC
Random Word Generation (RT)	0.9737	0.9038	0.9374	0.9786
Random Word Generation (Bus)	0.9752	0.7979	0.8777	0.9507
Status Word Manipulation (TR)	0.9027	0.6896	0.7818	0.9639
Status Word Manipulation (REC)	0.8698	0.7873	0.8265	0.9789
Man in the Middle	0.9853	0.7417	0.8463	0.9185
Command Invalidation	0.9833	0.4103	0.5790	0.8639
TX Shutdown	0.9508	0.4142	0.5771	0.9041
Desynchronization	0.6666	0.7059	0.6587	0.8565
Data Trashing	0.9683	0.6025	0.7428	0.9481
Data Word Corruption	0.9441	0.8787	0.9102	0.9676

Table 8. Detection of Attacks without Training Using OD1NF1ST

Attack Type	Precision	Recall	F1	AUC
Random Word Generation (RT)	0.7305	0.7951	0.7614	0.9590
Random Word Generation (Bus)	0.8748	0.9519	0.9117	0.9905
Status Word Manipulation (TR)	0.8557	0.6667	0.7494	0.9186
Status Word Manipulation (REC)	0.9918	0.6365	0.7753	0.8942
Man in the Middle	0.7205	0.6947	0.7073	0.9089
Command Invalidation	0.4107	0.0885	0.1456	0.8029
TX Shutdown	0.6933	0.1130	0.1943	0.8301
Desynchronization	0.5647	0.4237	0.4841	0.8511
Data Trashing	0.8715	0.5787	0.6955	0.8455
Data Word Corruption	0.8686	0.7668	0.8145	0.9522

words are interpreted by devices, not the messages they send. The minimal differences between benign and malicious words creates a challenge for our IDS to successfully detect them.

6.6 Unknown Attack Detection

To further analyze OD1NF1ST's ability to detect attacks in a real-world scenario, we divided the training and testing datasets based on attack type and evaluated them independently. This allowed us to determine if OD1NF1ST can identify attack scenarios that it has not been specifically trained on. Table 8 shows the results from attempting to detect each attack type without training on it beforehand. To evaluate the model, we only consider binary anomaly detection performance, since the network cannot categorize an attack outside of a given list.

Overall, we observed a significant performance decrease when detecting attacks the model has not been trained on. Furthermore, some attack types were rendered completely undetectable. Most attacks ended with an F1 score between 0.7 and 0.8 with the exception of random word generation, command invalidation, TX shutdown, and desynchronization. The AUC scores remained consistent between 0.80 and 1.0 for all attacks.

Random word generation was noticeably easier to detect compared with all other attacks. Attacking the system by sending a large number of messages to the bus is so disruptive that anomalies are extremely easy to detect, even with no training. Command invalidation, TX Shutdown, and Desynchronization are all attacks that fall under intercepted RT functionality. Without specific training and attack scenarios, the system cannot identify these three types of attacks with useful accuracy.

6.7 OD1NF1ST Installation on Raspberry Pi

We have simulated running OD1NF1ST on a Raspberry Pi 4 to further demonstrate the system's capabilities on resource-constrained hardware. The specs of the Raspberry Pi were as follows: Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz, 8 GB LPDDR4-3200 SDRAM. The Raspberry Pi was connected to: a USB mouse and keyboard for I/O, cat5e ethernet for downloading software updates, micro-HDMI to 1920 × 1080 monitor, 120 V wall outlet, with a 20 W adapter.

We loaded a subset of the Takeoff dataset and evaluated it using OD1NF1ST. The total runtime of the test was 9.17 s, which when extrapolated to the entire dataset would be approximately 91.7 s (61% the speed of the prior test). Such a platform could realistically classify 1 word per millisecond. During execution, the Raspberry Pi consumed approximately 5 W, which is well within the provided electric current on modern military aircraft. For comparison, the BAE RAD750 embedded flight computer is designed to use between 6.0 and 11.5 W when connected to DC current [3].

7 CONCLUSION

In conclusion, we present a software system for simulation of the MIL-STD-1553 platform that can be integrated in real-time with an IDS to create a web-based attack-defence game. The OD1NF1ST simulator can collect high-resolution data over significantly longer recording periods compared to any existing method. Furthermore, it allows for future-proof integration of advanced cyber-attack scenarios on our MIL-STD-1553 simulator. Our IDS can detect anomalies and classify attacks with greater performance compared to other state-of-the-art methods with limited computational resources while achieving comparable performance to more expensive approaches.

The feasibility and capability of the attack scenarios are proven by their implementation in this simulation system as well as the theoretical basis for their existence as described by He et al. [20]. The data from the simulation environment can be pre-processed in different ways than described in Section 7 to provide even greater detail to the IDS. The generalizability of the system is provided by MSFS, which can simulate a custom flight scenario anytime, anywhere, and in a multitude of weather conditions. Unseen attacks can also be implemented on-demand using the framework to evaluate future discoveries of MIL-STD-1553 vulnerabilities.

There are two possible directions for future work in this research area: improvement in the simulation system and refinement of the intrusion detection system. The simulation system can be further enhanced to generate new datasets for testing and training different models. The IDS could be further improved and modified to be more generalizable to new attack scenarios as they are developed or new datasets generated from different flight scenarios.

REFERENCES

[1] U.S. Department of Defense. 1978. MIL-STD-1553 designer's guide. (Sep. 1978).

[2] 1990. Electromagnetic pulse (EMP) and tempest protection for facilities. *U.S. Army Corps of Engineers, Publications Depot* 1110, 3 (Dec. 1990).

[3] 2000. RAD750 hardware specification. *BAE Systems* (Aug. 2000), 43–44.

[4] 2014. Special conditions: Airbus model A350-900 airplane; electronic system-security protection from unauthorized external access. *Federal Aviation Administration* 79, FR (Sep. 2014), 53128–53129.

[5] M. Al-Subaie and M. Zulkernine. 2007. The power of temporal pattern processing in anomaly intrusion detection. In *Proceedings of the IEEE International Conference on Communications*. Institute of Electrical and Electronics Engineers, Glasgow, UK, 1391–1398. https://doi.org/10.1109/ICC.2007.234

[6] Blaine Losier, Ron Smith, and Vincent Roberge. 2019. Time-based intrusion detection on MIL-STD-1553. Royal Military College of Canada, Project 2102 Technical report, 1–38.

[7] Victor Campos, Brendan Jou, Xavier Giró-i-Nieto, Jordi Torres, and Shih-Fu Chang. 2017. Skip RNN: Learning to skip state updates in recurrent neural networks. Retrieved from http://arxiv.org/abs/1708.06834.

[8] Paula Carpenter. 2013. Administration of aircraft & related ground support network security programs. *Civil Aviat. Advis. Publ.* 232A, 1 (Mar. 2013), 4–9.

[9] Condor Engineering. 2000. *MIL-STD-1553 Tutorial.* Condor Engineering, Santa Barbara, CA.

[10] Mocenco Daniela. 2015. Supply Chain management risks: The A350 development program. In *Proceedings of the 5th Eastern European Economic and Social Development Conference.* 38.

[11] Raymond De Cerchio and Chris Riley. 2011. Aircraft systems cyber security. In *Proceedings of the IEEE/AIAA 30th Digital Avionics Systems Conference.* 1C3–1–1C3–7. https://doi.org/10.1109/DASC.2011.6095969

[12] D. De Santo, C. S. Malavenda, S. P. Romano, and C. Vecchio. 2021. Exploiting the MIL-STD-1553 avionic data bus with an active cyber device. *Comput. Secur.* 100 (2021), 102097.

[13] Department of Defense. 1978. *Aircraft Internal Time Division Multiplex Data Bus.* Department of Defense.

[14] Leroy Earhart. 1982. MIL-STD-1553: Testing and test equipment. In *Proceedings of the Papers of the 2nd AFSC Avionics Standardization Conference*, Vol. 1. 349–365.

[15] Marwa A. Elsayed and Mohammad Zulkernine. 2020. PredictDeep: Security analytics as a service for anomaly detection and prediction. *IEEE Access* 8 (2020), 45184–45197. https://doi.org/10.1109/ACCESS.2020.2977325

[16] Sebastien J. J. Genereux, Alvin K. H. Lai, Craig O. Fowles, Vincent R. Roberge, Guillaume P. M. Vigeant, and Jeremy R. Paquet. 2019. Maidens: Mil-std-1553 anomaly-based intrusion detection system using time-based histogram comparison. *IEEE Trans. Aerospace Electron. Syst.* 56, 1 (2019), 276–284.

[17] Pierre-Francois Gimenez, Jonathan Roux, Eric Alata, Guillaume Auriol, Mohamed Kaaniche, and Vincent Nicomette. 2021. RIDS: Radio intrusion detection and diagnosis system for wireless communications in smart environment. *ACM Trans. Cyber-Phys. Syst.* 5, 3, Article 24 (Apr. 2021), 1 pages. https://doi.org/10.1145/3441458

[18] Douglas M. Hayward, Andrew Duff, and Charles Wagner. 2018. F-35 weapons design integration. In *Proceedings of the Aviation Technology, Integration, and Operations Conference.* https://doi.org/10.2514/6.2018-3370

[19] Daojing He, Yun Gao, Xiaoxia Liu, Sammy Chan, Yao Cheng, Xiaowen Liu, Baokang Zhao, and Nadra Guizani. 2021. Design of attack and defense framework for 1553B-based integrated electronic systems. *IEEE Netw.* 35, 4 (2021), 234–240. https://doi.org/10.1109/MNET.011.2000517

[20] Daojing He, Xuru Li, Sammy Chan, Jiahao Gao, and Mohsen Guizani. 2019. Security analysis of a space-based wireless network. *IEEE Netw.* 33, 1 (2019), 36–43.

[21] Daojing He, Xiaoxia Liu, Jiajia Zheng, Sammy Chan, Sencun Zhu, Weidong Min, and Nadra Guizani. 2020. A lightweight and intelligent intrusion detection system for integrated electronic systems. *IEEE Netw.* 34, 4 (2020), 173–179.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. Retrieved from http://arxiv.org/abs/1512.03385.

[23] Ting Huang, Gehui Shen, and Zhi-Hong Deng. 2019. Leap-LSTM: Enhancing long short-term memory for text categorization. Retrieved from http://arxiv.org/abs/1905.11558.

[24] Abdul Rehman Javed, Saif ur Rehman, Mohib Ullah Khan, Mamoun Alazab, and Thippa Reddy G. 2021. CANintelliIDS: Detecting in-vehicle intrusion attacks on a controller area network using CNN and attention-based GRU. *IEEE Trans. Netw. Sci. Eng.* 8, 2 (2021), 1456–1466. https://doi.org/10.1109/TNSE.2021.3059881

[25] Gonçalo Jesus, António Casimiro, and Anabela Oliveira. 2021. Using machine learning for dependable outlier detection in environmental monitoring systems. *ACM Trans. Cyber-Phys. Syst.* 5, 3, Article 29 (July 2021), 30 pages. https://doi.org/10.1145/3445812

[26] Jemti Jose. 2013. Design of manchester II bi-phase encoder for MIL-STD-1553 protocol. In *Proceedings of the International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s'13).* IEEE, 240–245.

[27] Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. 2017. Residual LSTM: Design of a deep recurrent architecture for distant speech recognition. Retrieved from http://arxiv.org/abs/1701.03360.

[28] Tae-Young Kim and Sung-Bae Cho. 2018. Web traffic anomaly detection using C-LSTM neural networks. *Expert Syst. Appl.* 106 (2018), 66–76. https://doi.org/10.1016/j.eswa.2018.04.004

[29] Fanhui Kong, Jianqiang Li, Bin Jiang, Huihui Wang, and Houbing Song. 2021. Integrated generative model for industrial anomaly detection via bi-directional LSTM and attention mechanism. *IEEE Trans. Industr. Inform.* (2021), 1. https://doi.org/10.1109/TII.2021.3078192

[30] Liwei Kuang and Mohammad Zulkernine. 2008. An anomaly intrusion detection method using the CSI-KNN algorithm. In *Proceedings of the ACM Symposium on Applied Computing (SAC'08).* ACM, New York, NY, 921–926. https://doi.org/10.1145/1363686.1363897

[31] Markus G. Kuhn and Ross J. Anderson. 1998. Soft tempest: Hidden data transmission using electromagnetic emanations. In *Information Hiding*, David Aucsmith (Ed.). Springer, Berlin, 124–142.

[32] V. Lalitha and S. Kathiravan. 2014. A review of manchester, miller, and fm0 encoding techniques. *SmartCR* 4, 6 (2014), 481–490.

[33] Efrat Levy, Nadav Maman, Asaf Shabtai, and Yuval Elovici. 2022. Anomili: Spoofing Prevention and Explainable Anomaly Detection for the 1553 Military Avionic Bus. Retrieved from https://arxiv.org/abs/2202.06870.
[34] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. 2013. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* 36, 1 (2013), 16–24.
[35] Karim Lounis and Mohammad Zulkernine. 2020. Exploiting race condition for Wi-Fi denial of service attacks. In *Proceedings of the 13th International Conference on Security of Information and Networks (SIN'20)*. ACM, New York, NY, Article 2, 8 pages. https://doi.org/10.1145/3433174.3433584
[36] Robert M. McGraw, Mark J. Fowler, David Umphress, and Richard A. MacDonald. 2014. Cyber threat impact assessment and analysis for space vehicle architectures. In *Sensors and Systems for Space Applications VII*, Vol. 9085. SPIE, 131–141.
[37] Bill Miller and Dale Rowe. 2012. A survey SCADA of and critical infrastructure incidents. In *Proceedings of the 1st Annual Conference on Research in Information Technology*. 51–56.
[38] J. K. Murdock and James R. Koenig. 2000. Open systems avionics network to replace MIL-STD-1553. In *Proceedings of the 19th Digital Avionics Systems Conference. Proceedings (DASC'00)*, Vol. 1. IEEE, 4E5–1.
[39] Minki Nam, Seungyoung Park, and Duk Soo Kim. 2021. Intrusion detection method using bi-directional GPT for in-vehicle controller area networks. *IEEE Access* 9 (2021), 124931–124944. https://doi.org/10.1109/ACCESS.2021.3110524
[40] Francis Onodueze and Darsana Josyula. 2020. Anomaly detection on MIL-STD-1553 dataset using machine learning algorithms. In *Proceedings of the IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'20)*. 592–598. https://doi.org/10.1109/TrustCom50675.2020.00084
[41] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. 2018. A comparison of ARIMA and LSTM in forecasting time series. In *Proceedings of the 17th IEEE International Conference on Machine Learning and Applications (ICMLA'18)*. 1394–1401. https://doi.org/10.1109/ICMLA.2018.00227
[42] S. Sivamohan, S. S. Sridhar, and S. Krishnaveni. 2021. An effective recurrent neural network (RNN) based intrusion detection via bi-directional long short-term memory. In *Proceedings of the International Conference on Intelligent Technologies (CONIT'21)*. 1–5. https://doi.org/10.1109/CONIT51480.2021.9498552
[43] Matthew John Squair. 2007. Safety, software architecture and MIL-STD-1760. In *Proceedings of the 11th Australian Workshop on Safety Critical Systems and Software*. 93–112.
[44] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. Retrieved from http://arxiv.org/abs/1505.00387.
[45] Orly Stan, Adi Cohen, Yuval Elovici, and Asaf Shabtai. 2019. On the security of MIL-STD-1553 communication bus. In *Security and Safety Interplay of Intelligent Software Systems*, Brahim Hamid, Barbara Gallina, Asaf Shabtai, Yuval Elovici, and Joaquin Garcia-Alfaro (Eds.). Springer International Publishing, Cham, 153–171.
[46] Orly Stan, Adi Cohen, Yuval Elovici, and Asaf Shabtai. 2020. Intrusion detection system for the MIL-STD-1553 communication bus. *IEEE Trans. Aerospace Electron. Syst.* 56, 4 (2020), 3010–3027. https://doi.org/10.1109/TAES.2019.2961824
[47] Orly Stan, Yuval Elovici, Asaf Shabtai, Gaby Shugol, Raz Tikochinski, and Shachar Kur. 2017. Protecting military avionics platforms from attacks on mil-std-1553 communication bus. Retrieved from https://arXiv:1707.05032.
[48] Shahroz Tariq, Sangyup Lee, Huy Kang Kim, and Simon S. Woo. 2020. CAN-ADF: The controller area network attack detection framework. *Comput. Secur.* 94 (2020), 101857. https://doi.org/10.1016/j.cose.2020.101857
[49] R. Ben Truitt, Edward Sanchez, and Michael Garis. 2004. Using open networking standards over MIL-STD-1553 networks. In *Proceedings of the AUTOTESTCON*. IEEE, 117–123.
[50] Waseem Ullah, Amin Ullah, Ijaz Ul Haq, Khan Muhammad, Muhammad Sajjad, and Sung Wook Baik. 2020. CNN features with bi-directional LSTM for real-time anomaly detection in surveillance networks. *Multimedia Tools Appl.* 80, 11 (2020), 16979–16995. https://doi.org/10.1007/s11042-020-09406-3
[51] John Villasenor. 2013. *Compromised by Design?: Securing the Defense Electronics Supply Chain*. Center for Technology Innovation at Brookings.
[52] Chris Wiegand. 2018. F-35 air vehicle technology overview. In *Proceedings of the Aviation Technology, Integration, and Operations Conference*. 3368.
[53] Di Wu, Hanlin Zhu, Yongxin Zhu, Victor Chang, Cong He, Ching-Hsien Hsu, Hui Wang, Songlin Feng, Li Tian, and Zunkai Huang. 2020. Anomaly detection based on RBM-LSTM neural network for CPS in advanced driver assistance system. *ACM Trans. Cyber-Phys. Syst.* 4, 3, Article 27 (May 2020), 17 pages. https://doi.org/10.1145/3377408
[54] Ran Yahalom, David Barishev, Alon Steren, Yonatan Nameri, Maxim Roytman, Angel Porgador, and Yuval Elovici. 2019. Datasets of RT spoofing attacks on MIL-STD-1553 communication traffic. *Data Brief* 23 (2019), 103863.
[55] Linxi Zhang and Di Ma. 2022. A hybrid approach toward efficient and accurate intrusion detection for in-vehicle networks. *IEEE Access* 10 (2022), 10852–10866. https://doi.org/10.1109/ACCESS.2022.3145007

Received 9 September 2021; revised 19 May 2022; accepted 1 July 2022