This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAES.2019.2914519, IEEE Transactions on Aerospace and Electronic Systems

MAIDENS: MIL-STD-1553 Anomaly-Based Intrusion Detection System Using Time-Based Histogram Comparison

Sébastien J.J. Généreux, Alvin K.H. Lai, Craig O. Fowles, Vincent R. Roberge, Guillaume P.M. Vigeant, and Jeremy R. Paquet

Abstract—**Aircraft are becoming increasingly connected to other systems or networks, introducing attack vectors and posing a threat to their operational effectiveness. Many military aircraft function using the MIL-STD-1553 protocol in order to communicate information across its systems. Previous intrusion detection research on this type of data bus is unable to detect spoofing attacks where only data words are manipulated. This paper presents the design of a MIL-STD-1553 anomaly-based intrusion detection system (MAIDENS) able to identify those attacks through the use of a novel histogram comparison approach and time-based features. The approach includes optimizations for improved time resolution in threat identification. Results demonstrate the effectiveness of the approach in identifying threats quickly and accurately.**

Index Terms—***Military Aircraft, Data buses, MIL-STD-1553, Anomaly Detection, Intrusion Detection***

I. Introduction

The concept of operation within which this research was conducted involves recording bus traffic during operation of the aircraft and subsequently analyzing identified anomalies post-flight. MAIDENS is implemented on a recording device, known as a bus monitor. Airworthiness implications and in-flight personnel limitations for some platforms make a post-mission analysis more appropriate.

Paper submitted for publication on 24 April 2019. We gratefully acknowledge the support of the Directorate of Technical Airworthiness and Engineering Support who funded this research.

S.J.J. Généreux, is with National Defence, ON K1A 0K2 (e-mail: sebastiengenereux@outlook.com).

A.K.H. Lai, is with National Defence, ON K1A 0K2 (e-mail: alvinkhlai@yahoo.ca).

C.O. Fowles, is with National Defence, ON K1A 0K2 7N2 (e-mail: craigf@execulink.com).

V.R. Roberge, is with the ECE Department of RMC, ON K7K 7B4 (e-mail: Vincent.Roberge@rmc.ca).

G.P.M. Vigeant, is with National Defence, ON K1A 0K2 (e-mail: guillaume.vigeant@gmail.com).

J.R. Paquet, is with National Defence, ON K1A 7B4 (e-mail: Jeremy_paquet@rocketmail.com).

MIL-STD-1553 is a military bus communications protocol published in 1973 by the United States Department of Defense. It was originally produced to be used with military avionics, but in the following decades, it was adopted into all branches of the armed forces, as well as spacecraft and commercial avionics [1]. The MIL-STD-1553 protocol uses a multipoint topology of terminals connected via a dual redundancy data bus. Terminals interface avionics subsystems with the data bus.

One terminal is assigned the role of the bus controller (BC) which initiates all communication on the bus. A terminal assigned the task to record certain information from the bus is known as a bus monitor (BM). All other terminals that are not a BM or the BC are called remote terminals (RT). Fig. 1 depicts a simple bus topology showing the BC and two RTs. There can be up to 31 RTs on the bus, in addition to one address reserved for broadcast messages.

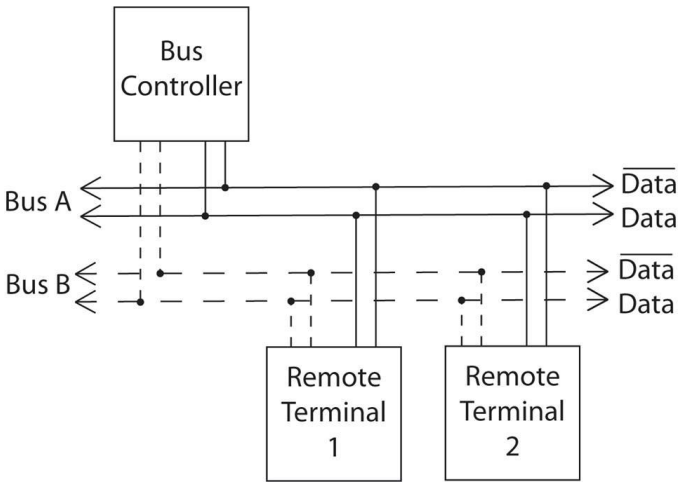

Fig. 1. Simple MIL-STD-1553 bus topology consisting of a BC and two RTs connected by a dual redundancy data bus.

Transfer of information is accomplished through three types of 20 bit words. Fig. 2 shows the per bit breakdown of each word.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAES.2019.2914519, IEEE Transactions on Aerospace and Electronic Systems

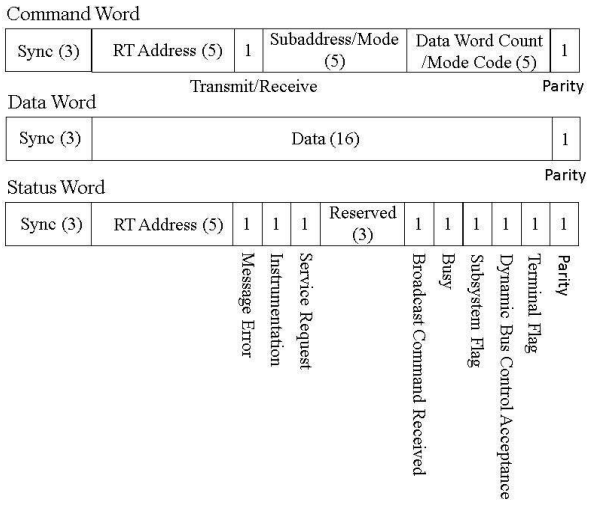

Fig. 2. Per bit breakdown of command, data and status words for the MIL-STD-1553 protocol.

Command words are sent by the BC to initiate communications on the bus. They indicate the intended recipient address, the type of communication and the size. In cases where the sub-address is 11111 or 00000, the data word count is interpreted as a mode command that the BC can use to trigger actions such as a RT reset. Data and status words are sent to transfer data and current subsystem status respectively.

Words are combined to form messages, of which there are 2 main types. Data transfers involve a receive or transmit command being sent, followed by data and status words. This can be a BC to/from RT or RT to RT transfer. Fig. 3 shows two such transfers, both initiated by the BC. Note the delays between command and reply (status) words. These delays have strict timing bounds, meaning the BC rejects any response past a certain delay and proceeds to the next command. The other type is mode commands that allow the BC to trigger specific actions. Both of these types of messages have broadcast equivalents where all RTs are addressed at once. This is done by indicating 11111 in the RT address field of a command.

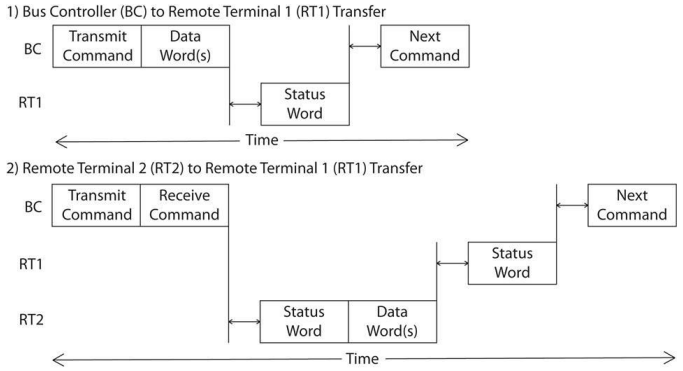

Fig. 3. Bus message examples demonstrating 1) transfer of data from the BC to an RT and 2) transfer of data between two RTs.

Reliability and data integrity are ensured by having very strict timing requirements when sending and receiving messages. MIL-STD-1553 was designed specifically for data integrity in information critical systems, not security. It was assumed at the time that aircraft were protected by the air gap between them and the ground. This however, is no longer the case with increasing wired and wireless connectivity, frequent software updates, and complex supply chain structures within larger corporations. The security of a network which was once thought to be isolated can no longer be taken for granted. The prevalence of MIL-STD-1553 makes it impractical to replace, so it will continue to be used now and in the future. It is then imperative that measures be taken to secure this protocol against potential exploits.

Other domains such as building automation and control networks [2], as well as SCADA networks [3], were also once thought to be protected through separation. These are also becoming increasingly connected to the outside world. It is this connectivity that prompted the creation of intrusion detection systems for such domains. MIL-STD-1553 however, is quite unlike a traditional TCP/IP network, in that it is extremely deterministic in the sequence of messages that are sent and received. Systems using this protocol iterate through a fixed order of messages, called a schedule, to keep track of the status of each remote terminal. This determinism paired with an untouched cyber battlespace, is what convinced some researchers of the suitability of an anomaly-based solution to this problem.

Recent research has revealed potential attacks that could compromise the integrity and availability of a MIL-STD-1553 communications bus and proposes an approach to intrusion detection [4]. Further research has provided evidence that a time-based approach to intrusion detection is especially suitable for identifying attacks on deterministic networks with strict timing requirements [5]. The contribution of this paper is twofold. Firstly, a novel approach to anomaly detection on the MIL-STD-1553 bus is presented. The approach makes use of histograms to model the timing characteristics of the MIL-STD-1553 protocol, and optimizations to improve time resolution when detecting threats. Secondly, the paper proposes a design and implementation of a modular framework in which the detection approach is used. 5 different threat scenarios are used to validate the system. Results show the system can identify the start and end of threat presence on the bus in under one second in most cases.

Section II discusses previous work in the fields of intrusion detection and MIL-STD-1553. Section III goes on to describe the proposed approach in detail, followed by Section IV which outlines the design and implementation of MAIDENS. Section V then defines and presents the means used to validate the approach and system. Finally, Section VI concludes.

II. Previous Work

Intrusion detection systems can be generally characterized by how they identify intrusions. Either the system is aware of what benign behaviour looks like, known as anomaly-based, or it knows what malicious behaviour looks like, known as signature-based. An intrusion is then characterized as either matching the pattern of malicious activity, or exceeding the normal behaviour of the system by a certain tolerance [6].

Signature-based intrusion detection systems have seen much success in traditional anti-virus applications, with additional research demonstrating its suitability for alternate environments such as on Android devices [7] and wireless ad hoc networks [8]. Recent research in this field focuses on minimizing memory requirements and the use of parallel processing in signature-based systems [9].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAES.2019.2914519, IEEE Transactions on Aerospace and Electronic Systems

Anomaly-based approaches have equally benefited from success, featuring low false negative and false positive rates. Much of this research however, has focused on non-deterministic traditional networking applications such as the web [10], which due to their complexity, require advanced detection techniques. These techniques vary greatly, ranging from deep learning [11], support vector machines [12], and Markov models [13], to hybrid approaches combining various techniques [3,14]. A commonality to many of these approaches is the need to select features with which to characterize the normal behaviour of the system, which is another topic of research in itself [15].

Research conducted by Losier, B. has shown that for deterministic networks such as those on aircraft using MIL-STD-1553, time-based features are an effective means for threat identification [5]. Losier, B.'s research is a proof of concept on the feasibility of using time-based features for intrusion detection on MIL-STD-1553, focusing on the features themselves, shown in Table I, and their relevance to intrusion detection. To validate the proof of concept, Losier, B. used a simple histogram approach which plotted the frequency of each value of a given feature. Histograms generated from known, intrusion-free data were referred to as the baseline, against which histograms generated from runtime data could be compared. If the runtime value was a certain percent over baseline value, then it would be considered anomalous. This approach is effective to validate the feature selection, however performs poorly in terms of the accuracy in time with which an intrusion can be identified.

TABLE I. TIME-BASED FEATURES [5]

Feature	Description
Response Time	Time for an RT to respond to a command
Inter-message Gap	Time from end of one message to the start of the next
Periodicity	Mean time difference in start times of messages over a given time period
Bus Utilization	Percent use of the bus over a given time period
Data Throughput	Number of data words sent over the bus over a given time period

The only other research dealing with MIL-STD-1553 security found was conducted by Stan, O. et al. which presents a comprehensive study of the threats faced by systems that use MIL-STD-1553, including possible attacks vectors and consequences [4]. They present a sequence-based anomaly detection method that uses Markov models defined using command word and timing features. Their results indicate great success in identifying anomalous bus activity stemming from a range of attack scenarios, including different types of spoofing and DDoS attacks. An important contribution of their research is the ability to classify messages as periodic or aperiodic to detect anomalies on each type separately [4]. A limitation identified in their paper is the inability to trigger anomalies when a corrupt RT is spoofing the BC or another RT, but only manipulates the data words and not the command word. The approach presented in this paper addresses this limitation by taking advantage of different time-based features selected by Losier, B. [5].

This paper proposes a novel approach, inspired by Losier, B.'s histogram comparison approach, that introduces different anomaly detection criteria to minimize false positive rates and utilizes a process to maximize detection accuracy in terms of time.

III. PROPOSED APPROACH

Overall, the approach entails comparing two histograms which plot the value of a feature from Table I against its density. These features were selected due to their success in identifying anomalies in Losier, B.'s research [5]. The first of these histograms is what is called the baseline profile, with there being one for each feature representing the normal behaviour of the bus. The runtime histogram is generated either from directly monitoring a live bus, or extracting values from a bus recording.

Given the highly deterministic nature of the bus, the simplest possible approach thought of was selected to improve processing speeds. Although rudimentary in comparison to modern anomaly-based detection techniques, this paper shows histogram comparison to be effective.

A. Parsing

Parsing involves extracting each feature from the bus and collecting them into buffers. Feature value are either extracted directly from the messages on the bus or as a function of time. The timestamp of the first and last messages in each buffer are saved as the start and end times for that buffer. The term window is used to describe the set of feature values in a buffer.

Windows are extracted in one of two ways. Either an entire window's worth of data is buffered each time, or a certain number of values from the previous window are kept in the subsequent window, known as a sliding window. The sliding window method can produce far more buffers, leading to a greater time resolution, however is much more computationally intensive. The regular buffering method could be sufficient depending on the accuracy of intrusion detection required by the application. This parsing process is identical for both building the baseline profile and profiles of runtime traffic.

Special care must be taken at this stage to account for the range of MIL-STD-1553 message and schedule types. These vary in the presence or lack of certain data or status words. Information on this can be found in the MIL-STD-1553 protocol definition document [16].

B. Building the Baseline Histogram

In order to build the baseline, each buffer is divided into bins of equal width based on the values contained in that buffer. These bins represent the density of values that fall between the low and high bounds of the respective bin. Density is calculated by dividing the number of values in a bin by the total number of values in the buffer, whereby the sum of all the bins' densities equals one. A baseline can be created when more than two buffers are received. For each bin, the minimum and maximum densities encountered across all of the buffers are stored. This represents the boundaries of normal bus traffic behaviour. The average of the densities is also calculated so that the user can determine whether most densities were closer to the maximum or minimum bounds, but this metric is not used in the detection scheme. An example of a complete baseline profile is displayed in Fig. 4 with the blue dots representing the average in the bin and the error bar as the range between the low and high bounds.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAES.2019.2914519, IEEE Transactions on Aerospace and Electronic Systems

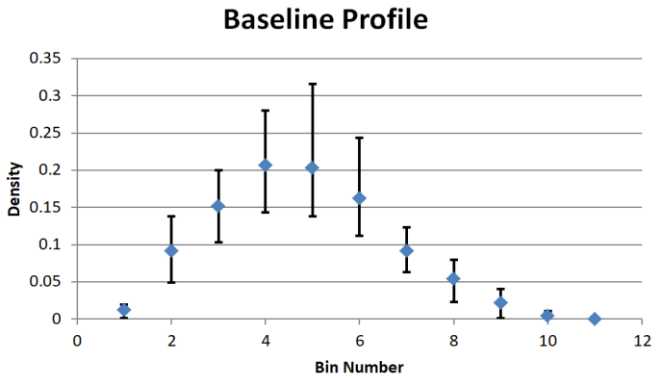

Fig. 4. A baseline profile with 10 bins demonstrating the average density of parsed buffers, alongside the maximum and minimum densities for each bin.

There is the option of applying engineering tolerance margins to each of the baselines. Two percentages, one for the low bound and one for the high bound, can be defined for all of the bins for a specific detection feature. The goal of the margins is to decrease the sensitivity of the system by decreasing the low bound or increasing the high bounds. For example, if a bin's low bound is 0.10 and the tolerance margin is set at 20%, a value will only be considered an anomaly if it is less than 0.08. Likewise, if the bin's high bound is 0.40 and the margin is set at 10%, the actual threshold would be set at 0.44.

C. Detecting an Anomaly

When analyzing traffic, each buffer is examined individually. It is divided into bins of the same width as the baseline profile that corresponds to the buffer's feature. The density of each bin from the runtime buffer is compared with the minimum and maximum from the baseline post-application of the engineering tolerance margins. This differs from the approach presented by Losier, B., as anomalies were identified based on a percent difference from the mean for each bin [5]. In addition, since a density is used instead of a frequency, the analyzed buffers can have a larger window size than the baseline buffers to decrease processing time. Fig. 5 is a graphical representation of the comparison, with the error bar chart as the baseline. The green dots in bins 1, 5, 6 and 11 are within the boundaries. The red dots are the anomalies that would be detected, with bins 2, 3 and 4 being under the minimum and bins 7, 8, 9, 10 being over the maximum.

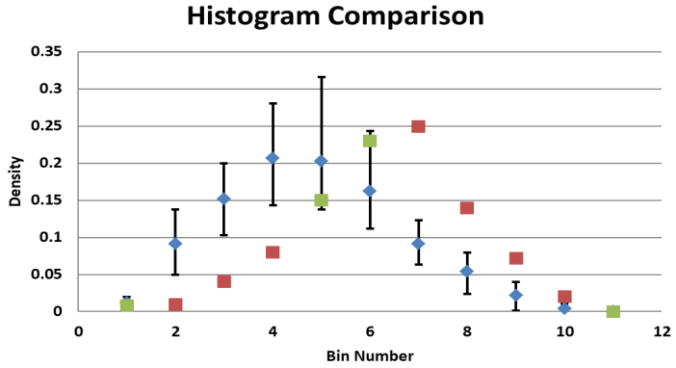

Fig. 5. Anomaly detection histogram that superimposes a runtime profile (squares) over a baseline profile (diamonds and bars) to identify anomalies.

It can now be understood why this method addresses the limitation identified in previous research [4]. A corrupt RT can take the address of another RT and since there is no authentication in this protocol, the bus controller will not be able to differentiate between the corrupt RT and the intended RT. This allows the corrupt RT to modify the data field of the message being returned to the BC, which could, for example, change navigational information. This approach can identify such an occurrence by building a baseline for each RT's response time and then comparing runtime histograms against that baseline. Any difference in delay between the intended RT and the corrupt RT's response times would be noticed and an anomaly generated. Messages travel as manchester encoded signals over twinaxial cables, meaning there exists a delay in the propagation of that message through the cable. A corrupt RT that is further than the intended RT from the BC would then experience a greater delay. Delays also occur due to different latencies in the RT itself. It is highly unlikely for two RTs to have identical latencies and distance from the BC, the only condition where the corrupt RT would not be identified.

D. Optimizations

Two optimization algorithms were applied to this approach to improve upon Losier, B.'s research [5]. Optimizing the bin width ensures that the histogram is the best representation of bus behaviour and establishing a minimum window size increases the time resolution for anomaly detection.

1) Bin Width Optimization

During the creation of a new baseline, the bin width for a feature is determined by the first buffer that is received from the parsing activity. Therefore, it is imperative that this buffer is representative of the system being profiled and that it is divided into bins correctly. Once the bin width is determined, it cannot be changed as the underlying data is not saved along with the baselines. The determination of a proper bin width is accomplished through a cost function expressed by:

$$C(\Delta) = \frac{2k - v}{\Delta^2} \quad (1)$$

where k is the mean, v is the variance and delta is the bin width of the data set. Equation (1) is modeled after a simplified version of the Mean Integrated Squared Error technique [17].

The cost function is iteratively calculated for bin widths that would result in a varying number of bins. The difference between the lowest and highest values in a buffer is divided by the number of bins desired in order to find the bin width. The bin width that is associated with the lowest cost, which could potentially be negative, represents the optimal bin width for the data. This process ensures that the resulting histogram properly captures the underlying data.

2) Minimum Window Size

The amount of information required to capture the normal behaviour of the system is unique to the specific feature, aircraft, remote terminal, mode of operation and even sub-address. This amount of information is what dictates the window size to be used in parsing. This window should be as small as possible in order to increase the time resolution of each window. Smaller windows result in less data required for analysis, leading to a faster response time in a real-time

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAES.2019.2914519, IEEE Transactions on Aerospace and Electronic Systems

application. Algorithm 1 below describes the proposed procedure to identify the minimum window size possible while still capturing the full range of normal behaviour of the system. Since bin width optimization is only applied to the first buffer, any subsequent buffers containing values for a certain feature that are not within the bins previously defined, will create additional bins on either end of the histogram. The addition of bins indicates that the first buffer was not representative of the full range of normal behaviour of the system and that a larger window size must be used.

Algorithm 1 Minimize window size

```
1 procedure minimizeWindowSize
2     windowSize = dataSize
3     n = maxOptimizedBins
4     numBins = 0
5     while numBins < n
6         numBins = CreateBaseline(windowSize)
7         windowSize - -
8     return windowSize
```

The procedure consists of reducing the window size until more bins than the number optimized for is observed. The minimum window size is identified as the lowest window size where no new bins were added. This is graphically shown in Fig. 6 below, where the red point is the minimum size. In this case, a window size of 30 values would be found to be the minimum window size that still captures the normal behaviour of the bus.

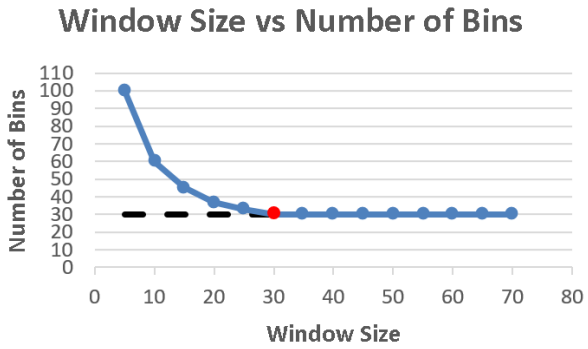

Fig. 6. Minimum window size plot graphically showing the iterative process of reducing window sizes until an increase in the number of bins is observed. The point in red is the minimum window size that captures the normal behaviour of the bus traffic.

E. Limitations

Being a learned system, the foremost limitation of MAIDENS is the requirement that the data used to teach the system and build baselines is representative of the full range of normal behaviours of the system. This could make data acquisition for baselining complex, requiring a comprehensive understanding of the use of the system. Also, any change to the system, such as the addition of a remote terminal or a significant software update, would require that a new baseline be created for the affected remote terminals.

This approach as implemented and validated, does not account for aperiodic events. Although the majority of bus traffic is periodic and deterministic, aperiodic events can be triggered by the operator. If these events are statistically well behaved they could be included in the teaching data, however some of them may very well not be part of the normal behaviour of the system. One cannot, for example, predict the number of times a given weapon system will be utilized.

IV. Design and Implementation

The proposed approach was implemented as a module within an overall system architecture referred to as MAIDENS. MAIDENS prioritizes flexibility to cater to different bus recording file formats. The system also facilitates further research into this approach if a new detection engine is ever created.

A. Software Components

MAIDENS was designed with the software components found in Fig. 7 in order to implement the approach specified in Section I. The major components are the parser, detection engine and graphical user interface, while the other components in Fig. 7 provide supporting services to the system. The parser is responsible for the retrieval and pre-processing of the MIL-STD-1553 traffic. The pre-processing consists of calculating the values for features and dividing sets of these values into windows. Features are referred to as characteristics in the MAIDENS design since they are extracted to capture the normal behaviour of the traffic. The analysis can be conducted when these windows are passed to the detection engine. Probability density information is calculated by iterating through all the values in the window. The detection engine also compares runtime traffic to the baseline in order to determine whether it is anomalous. The graphical user interface receives commands from the user and invokes the parser and detection engine functionalities as required. It is also tasked with visually rendering the analyzed data to the user.

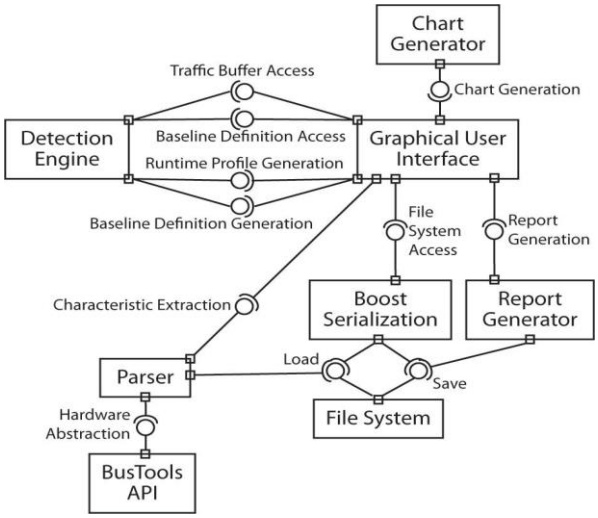

Fig. 7. Component diagram describing the relationships among software modules in terms of services provided or used.

B. Software Architecture

The data structures in Fig. 8 show the composition of the top-level Analysis object. The characteristics being examined are broken down into two categories depending on how they are calculated. Each buffer of values captures a specific window of time of the traffic. This time information is necessary so that any alerts that are generated can be tracked temporally. Each characteristic has a baseline profile, and the collection of these baseline profiles constitutes a baseline definition. In a general sense, each profile has a collection of bins, with the baseline

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAES.2019.2914519, IEEE Transactions on Aerospace and Electronic Systems

consisting of bins that contain a few extra data fields to capture information about the boundaries. Within the runtime profiles specifically, the alerts generated by anomalies are also saved.

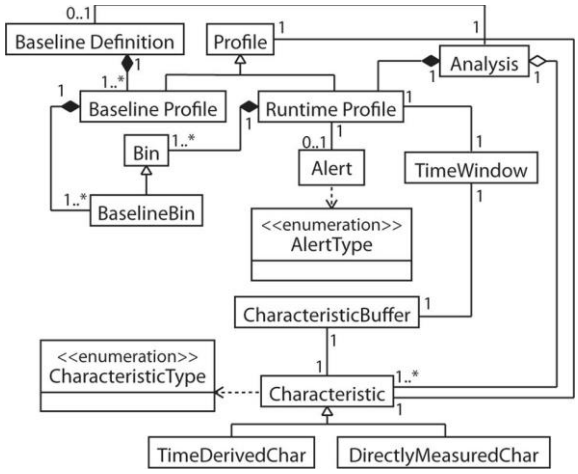

Fig. 8. Data structures class diagram showing the object-oriented breakdown of the overall Analysis object.

Fig. 9 highlights the dual mode of operation for the detection engine module. During the creation of a baseline definition, it is learning what the normal traffic should look like. During this stage, the accumulator sub-module is utilized in order to aggregate all of the buffers from the parser. The bin width optimization also resides in this sub-module, searching between two and thirty bins inclusively. It would be impractical for there to only be one bin and so the lower limit was set to two. The upper limit of bins was chosen so that each bin would be discernible on the graphical user interface. The analyze mode of operation is used to compare runtime traffic with a baseline, which uses the comparator sub-module.

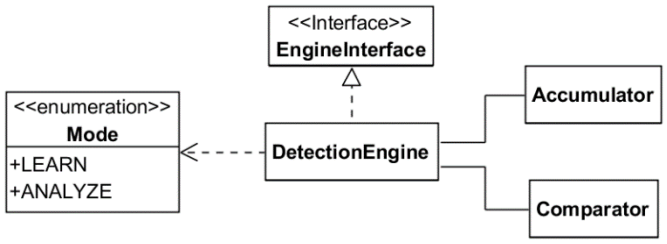

Fig. 9. Detection engine class diagram showing its two distinct modes of operation, when creating baselines (Learn) and when analyzing runtime traffic (Analyze).

C. Supporting Services

Components that enable the functionality of the core modules are shown in Fig. 7. BusTools API is required to capture traffic from a live bus. The C++ Boost library is used to serialize the baselines so that they can be reused in subsequent analyses. The chart generator provides a set of functions that can be used to create the error bar charts for the graphical user interface. The report generator is a C# plugin that retrieves the baselines and analysis results to populate Microsoft Excel and Word documents. This allows for further investigations of the bus traffic in greater detail. The file system is included in the diagram to signify that data is read from and written to the computer's disk.

V. Validation

Validation entailed verifying the efficacy of the intrusion detection approach. This is described in detail, focusing on the validation environment, the evaluation metrics, and finally the results for each metric.

A. Validation Environment

The approach was validated on a desktop computer with a quad-core Intel i5-4590 and 8GB of memory. Bus recordings were obtained from a test bench with real and simulated remote terminals in the case of accuracy validation and from completely simulated environments in the case of performance validation. The simulated environment consisted of the GE BusTools software running on a computer which either had the R15-USB or R15-EC mounted. These devices allow for the simulation of entire MIL-STD-1553 networks with custom schedules, frame rates and data.

B. Evaluation metrics

1) Performance: Time to parse and analyze a large bus recording compared to the time span of the recording.

2) Accuracy: Ability to identify when a threat occurs on the bus.

C. Performance

An aircraft's schedule iterates over a major frame, which is a sequence of messages passed between specific remote terminals. This major frame has a frequency normally no greater than 20Hz. However, to assess the performance of the approach, a schedule with a major frame rate of 500 Hz was created and recorded. This is the highest stable frame rate supported by the GE BusTools software [18].

The recording contained 300 000 messages, recorded over two minutes and thirty seconds. MAIDENS was able to parse and analyze the recording against its baseline in 34.171s, approximately a fifth of the time of the actual recording. It is important to note that the approach was implemented as a single threaded program, although parallelization could be used to further improve performance. Since Losier, B. manually carried out the histogram comparison, this approach vastly improved on the processing time for detecting an anomaly [5].

D. Accuracy

The MAIDENS was validated for accuracy using data from a full CP-140 Aurora test bench, consisting of real and simulated remote terminals. Baseline data was first recorded, followed by one recording for each of the five scenarios in the classified threat model. Each recording was taken over twenty minutes, and contain the number of messages indicated in Table II. Scenarios 3 through 5 are targeted at a specific remote terminal, and so the minimum window size process described by Algorithm 1 was repeated to get the 50 values per window as opposed to 525 for the scenarios targeting all bus activity. The baseline file recordings capture the normal behaviour of the relevant RTs. Scenarios 1/2 and 3/4 share the same baselines since they involved the same remote terminals.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAES.2019.2914519, IEEE Transactions on Aerospace and Electronic Systems

TABLE II. THREAT MODEL VALIDATION DATA

Scenario	Baseline File (number of messages)	Test File (number of messages)	Number of values per window
1	753 898	738 863	525
2	753 898	545 463	525
3	56 804	88 958	50
4	56 804	68 141	50
5	63 970	67 355	50

The five test file recordings each contain three occurrences of the threat scenario, with each occurrence lasting for approximately one minute. The exact times when the scenarios start and end were known. These recordings were then analyzed using the approach presented in Section III against the baseline data. The resulting anomalies allow the identification of the periods of time over which the scenarios occur. Table III below shows the time relative to the start of the recording when the attack occurred and when the attack was identified by the MAIDENS.

TABLE III. ABSOLUTE DETECTION TIMES

Scenario	Threat Occurrence	Actual Time (minutes:seconds)		Identified Times (minutes:seconds)	
		Start	End	Start	End
1	1	3:18.21	4:25.11	3:18.03	4:39.62
	2	9:24.48	11:17.10	9:23.88	11:40.52
	3	14:06.50	17:37.10	14:06.15	19:20.50
2	1	2:49.12	3:45.59	2:48.28	3:46.29
	2	7:46.42	9:45.89	7:46.06	9:46.93
	3	13:01.53	15:46.50	13:01.38	15:47.44
3	1	5:01.36	6:03.96	5:01.29	6:04.84
	2	9:59.68	12:06:24	9:59.63	12:07.40
	3	15:00.44	18:00.23	14:59.33	18:00.53
4	1	2:17.086	3:18.28	2:16.79	3:18.38
	2	7:20.37	9:46.17	7:18.70	9:47.93
	3	12:18.14	15:19.14	12:17.68	15:24.05
5	1	3:38.77	4:42.41	3:37.72	4:42.96
	2	8:42.37	10:35.99	8:41.94	10:36.16
	3	13:38.14	16:14.53	13:36.19	16:14.72

To measure accuracy, the differences in time between the actual and identified times were calculated. The difference in time is impacted by the window size used as well as how busy the bus is. A bus with little traffic will have fewer messages per unit of time and therefore possibly a greater time difference than a busy bus. Time differences for all three occurrences were then averaged to yield the results in Table IV below. Importantly, all the anomalies identified were directly as a result of the threat scenarios, thus having no false positives.

TABLE IV. DETECTION TIME DIFFERENCES

Scenario	Average Start Time Difference (seconds)	Average End Time Difference (seconds)
1	0.38	47.11
2	0.45	0.89
3	0.41	0.78
4	0.81	2.26
5	1.14	0.30

Both this and Losier, B.'s approach were able to identify all anomalies in their respective validation datasets [5]. However, Losier, B.'s results only indicated the presence of anomalies and did not specify the time in the bus recording where they were detected. This approach's ability to pinpoint the time of anomaly occurrence demonstrates an improvement in accuracy.

VI. CONCLUSION AND FUTURE WORK

This paper presents a novel approach to intrusion detection using an anomaly-based histogram comparison method. A software implementation of Losier, B.'s proposed histogram comparison method in [5] was implemented and improved upon with the addition of a sliding window parser to narrow time resolution, a bin width optimization algorithm to better represent traffic characteristics, a minimum window size algorithm to find the smallest representative traffic sample, and an alternate anomaly detection criteria. To prove the validity of this method, tests were performed to measure the performance and accuracy of the software. The results demonstrate the success of the approach, specifically in identifying the time periods over which the intrusions occurred. In addition to introducing a new method of intrusion detection on a MIL-STD-1553 bus, this approach addresses a limitation clearly identified in previous work [4].

As implemented, multiple modes of operation of an aircraft are not yet considered, however this could be accounted for if a baseline is generated for each mode. The normal definition of traffic on the MIL-STD-1553 bus is very different during takeoff compared to during manoeuvres or while landing. To be able to compare traffic to the correct baseline and identify anomalies, a collection of baseline definitions to describe the normal operation of the aircraft need to be created, and the traffic must be compared to each baseline until either a match is found or the traffic is concluded to be anomalous.

The execution time of the program can also be greatly improved. Currently, this approach has only been implemented on a single thread, which is sufficient for the speed of the MIL-STD-1553 bus. If the approach is to be applied to modern, deterministic protocols that operate at a higher frequency, it may be necessary to accelerate the intrusion detection by adding parallelism on a graphics processing unit [19] or creating an application-specific integrated circuit [20]. There is also potential in designing an embedded solution that would reside on the aircraft itself, possibly feeding real-time information to a terminal, or recording anomalies for post flight analysis [21].

Additional features could be considered to determine their applicability to this histogram comparison approach. For example, the number of data words per message, mode codes or broadcasts over a given period of time could be explored [4]. Other content-related features could consist of the number of status words or a particular message type.

REFERENCES

[1] MILSTD1553.com. (n.d.). *Where is MIL-STD-1553?*. [Online] Available at: https://www.milstd1553.com/applications/ [Accessed Jun. 23, 2018].

[2] Z. Pan, S. Hariri and Y. Al-Nashif, "Anomaly based intrusion detection for Building Automation and Control networks," *2014 IEEE/ACS 11th International Conference on Computer Systems and PPLications (AICCSA)*, Doha, 2014, pp. 72-77.

[3] I. Ullah and Q. H. Mahmoud, "A hybrid model for anomaly-based intrusion detection in SCADA networks," *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, 2-17, pp. 2160-2167.

[4] O. Stan, Y. Elovici, A. Shabtai, G. Shugol, R. Tikochinski, and S. Kur. "Protecting Military Avionics Platforms from Attacks on MIL-STD-1553 Communication Bus." arXiv:1707.05032v1 [cs.CR]. Jul. 17, 2017.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAES.2019.2914519, IEEE Transactions on Aerospace and Electronic Systems

[5] B. Losier, R. Smith and V. Roberge, "Design of a Time-Based Intrusion Detection Algorithm for the MIL-STD-1553", Royal Military College of Canada, Kingston, Project number DTAES-8 2102, Jan. 2019. [Online]. Available: http://roberge.segfaults.net/joomla/files/publications/Project_Report_2102.pdf. [Accessed: Apr. 12, 2019].

[6] H. Holm, "Signature Based Intrusion Detection for Zero-Day Attacks: (Not) A Closed Chapter?," *2014 47th Hawaii International Conference on System Sciences*, Waikoloa, HI, 2014, pp. 4895-4904.

[7] M. Ghorbanian, B. Shanmuguam, G. Narayansamy and N. B. Idris, "Signature-based hybrid Intrusion detection system (HIDDS) for android devices," *2013 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC)*, Langkawi, 2013, pp. 827-831.

[8] F. Anjum, D. Subhadrabandhue and S. Sarkar, "Signature based intrusion detection for wireless ad-hoc networks: a comparative study of various routing protocols," *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat.No.03CH37484)*, 2003, pp. 2152-2156 Vol.3.

[9] A. H. Almutairi and N. T. Abdelmajeed, "Innovative signature based intrusion detection system: Parallel processing and minimized database," 2017 International Conference on the Frontiers and Advances in Data Science (FADS), Xi'an, 2017, pp. 114-119.

[10] C. Torrano-Gimenez, A. Perez-Villegas and G. Alvarez, "An Anomaly-Based Approach for Intrusion Detection in Web Traffic," 2010 Journal of Information Assurance and Security, 5, pp. 446-454.

[11] N. T. Van, T. N. Thinh and L. T. Sach, "An anomaly-based network intrusion detection system using Deep learning," 2017 International Conference on System Science and Engineering (ICSSE), Ho Chi Minh City, 2017, pp. 210-214.

[12] K. Ghanem, F. J. Aparicio-Navarro, K. G. Kyriakopoulos, S. Lambotharan and J. A. Chambers, "Support Vector Machine for Network Intrusion and Cyber-Attack Detection," 2017 Sensor Signal Processing for Defence Conference (SSPD), London, 2017, pp. 1-5.

[13] Y. Du, H. Wang and Y. Pang, "A hidden Markov models-based anomaly intrusion detection method," Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No.04EX788), 2004, vol. 5, pp. 4348-4351.

[14] K. Qazanfari, M. S. Mirpouryan and H. Gharaee, "A novel hybrid anomaly based intrusion detection method," 6th International Symposium on Telecommunications (IST), Tehran, 2012, pp. 942-947.

[15] I. Ullah and Q. H. Mahmoud, "A filter-based feature selection model for anomaly-based intrusion detection systems," 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, 2017, pp. 2151-2159.

[16] Department of Defence. (1975, April, 30). AIRCRAFT INTERNAL TIME DIVISION COMMAND/RESPONSE MULTIPLEX DATA BUS [Online]. Available: https://snebulos.mit.edu/projects/reference/MIL-STD/MIL-STD-1553B.pdf.

[17] H.Shimazaki and S. Shinomoto. "A Method for Selecting the Bin Size of a Time Histogram." *Neural Computation*, 2007, vol. 19, pp. 1503-1527.

[18] "BusTools/1553-API Reference Manual." GE Intelligent Platforms. Goleta: GE Intelligent Platforms Embedded Systems, Inc. Oct, 2012.

[19] N. T. T. Van and T. N. Thinh, "Accelerating Anomaly-Based IDS Using Neural Network on GPU," 2015 International Conference on Advanced Computing and Applications (ACOMP), Ho Chi Minh City, 2015, pp. 67-74.

[20] C. Tran, T. N. Vo and T. N. Thinh, "HA-IDS: A heterogeneous anomaly-based intrusion detection system," 2017 4th NAFOSTED Conference on Information and Computer Science, Hanoi, 2017, pp. 156-161.

[21] E. Viegas, A. O. Santin, A. França, R. Jasinski, V. A. Pedroni and L. S. Oliveira, "Towards an Energy-Efficient Anomaly-Based Intrusion Detection Engine for Embedded Systems," in IEEE Transactions on Computers, 2017, vol. 66, no. 1, pp. 163-177.

Sébastien J.J. Généreux was born in Lorretteville Québec, Canada in 1996. He received a BEng degree in computer engineering from the Royal Military College of Canada, Kingston, Ontario in 2018.

He joined the Canadian Armed Forces as a Communications and Electronics Engineering Officer in 2014. He is now studying at the Canadian Forces School of Communications and Electronics in Kingston, Ontario, Canada.

Alvin K.H. Lai was born in North York, Ontario, Canada in 1996. He received a BEng. degree in computer engineering from the Royal Military College of Canada, Kingston, Ontario, Canada in 2018.

He joined the Canadian Armed Forces as an Aerospace Engineering Officer in 2014. From 2014-2018, he was a student at the Royal Military College of Canada. Currently, he is undergoing training at the Canadian Forces School of Aerospace Technology and Engineering in Borden, Ontario, Canada.

Craig O. Fowles was born in London, ON, Canada in 1996. He received a BEng. in computer engineering from the Royal Military College of Canada, Kingston, Ontario in 2018.

He is an Acting Sub-Lieutenant in the Royal Canadian Navy posted to CFB Esquimalt, Esquimalt, Canada.

Vincent R. Roberge received his BEng, MASc and PhD in Computer Engineering from the Royal Military College (RMC), Kingston, Canada, in 2005, 2011 and 2016, respectively.

He worked as an Aerospace Engineering Officer in the Canadian Armed Forces from 2000 to 2016. In July 2011, he joined the Department of Electrical and Computer Engineering at RMC as a military lecturer. He retired from the military at the rank of Major and is currently an Assistant Professor at RMC.

Dr. Roberge's current research interests include cyber security, digital forensics, parallel computing, optimization and real-time applications.

Guillaume P.M. Vigeant was born in Montreal, Québec on May 24th 1983, Guillaume Vigeant received the B.Eng degree and the MA.Sc degree in computer engineering from the Royal Military College of Canada in 2005 and 2011 respectively. Since 2014, he is a Ph.D. candidate within the department of electrical and computer engineering at the Royal Military College of Canada.

After completing the Aerospace Engineering Officer Occupational Basic Course in May 2006, he worked at the CF-18 Weapon System Manager Detachment in Mirabel, Québec as the Sniper Pod Integration Technical Lead. From 2011 to 2018, he has been an assistant Professor with the Electrical and Computer engineering department at the Royal Military College of Canada where his research focused primarily on the computer security challenges in hard real-time embedded systems.

Jeremy R. Paquet received his BEng and MASc in Electrical and Computer Engineering from the Royal Military College of Canada (RMC) in 2009 and 2014 respectively.

He is currently employed as an Aerospace Engineering Officer at the rank of Captain in the Canadian Forces. In this

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAES.2019.2914519, IEEE Transactions on Aerospace and Electronic Systems

role he was worked with General Dynamics conducting software verification and validation, and taught at RMC as an assistant professor within the Electrical and Computer Engineering department.

Mr. Paquet's current interests include cyber security, wireless communications, distributed systems, and communication standards.