Attacking OMG Data Distribution Service (DDS) Based Real-Time Mission Critical Distributed Systems

Michael James Michaud[1], Thomas Dean[2], Sylvain P. Leblanc[1]
[1]*Royal Military College of Canada,* [2]*Queen's University*
Michael.Michaud@forces.gc.ca, Thomas.Dean@queensu.ca, Sylvain.Leblanc@rmc.ca

Abstract*—Object Management Group's Data Distribution Service for Real-Time Systems (DDS) middleware standard is a popular technology that forms the core of many mission-critical distributed real-time, data-centric systems, including command and control systems, Air Traffic Control (ATC) systems and critical infrastructure systems. This paper shows how DDS can be manipulated to support malicious activity. We focus on client-side attacks by modelling and demonstrating five attacks in self-contained and isolated environments and by validating them using an end-to-end demonstrative scenario. This research enables further work in detecting and defending against cyberattacks on ATC systems, control systems or any other DDS-based critical infrastructure system.

***Index Terms*—distributed computing, Data Distribution Service, DDS, middleware, mission critical, OpenDDS, Quality of Service, QoS, real time, RTI, RTPS, SCADA and security.**

I. INTRODUCTION

Object Management Group's (OMG) Data Distribution Service (DDS) [1, 2] defines a standard for middleware that lies between the application software and the network transport layers. DDS middleware frees the developers of large distributed systems from the need to specify and develop custom data distribution architectures, networks and supporting software. DDS allows for reliable and timely data collection and delivery using a publish-subscribe mechanism that permits dynamic node discovery, Topic-based data distribution and time-space de-coupling of data streams.

The publish-subscribe mechanism frees the application developer from specifying the details of communication with other applications in the system. In other words, a publishing application does not need to specify where the data is going; it only needs to specify the specific data type (i.e. the "Topic") being communicated. A subscribing application does not need to specify how or where the data is coming from; it only needs to specify the specific Topic that it wants to receive[3]. Configurable Quality of Service (QoS) parameters allow DDS to ensure that all participating applications adhere to the proper data communication requirements (e.g. integrity and timely delivery of data) to send or receive a given data Topic.

While the flexible publish-subscribe mechanisms of DDS provide developers maximum flexibility when building their system, these mechanisms give rise to potential security issues which could be taken advantage of by a malicious DDS-based application. These issues include the ease of access to all data in the system, loss of control of data routing and communication pathways, and the ability for entities to nefariously join the DDS network.

II. BACKGROUND

The DDS specification defines a middleware standard that allows for reliable and timely data collection and delivery using a publish-subscribe mechanism that permits dynamic node discovery, topic-based data distribution and time-space de-coupling of data streams. DDS includes QoS standards that ensure the integrity and timely delivery of data, which is critical for information management, surveillance and control systems.

DDS' publish-subscribe architecture differs from client-server and message passing architectures. Communication is based on the data that is transmitted, rather than the source and destination of the data. RTI Connext DDS™ and OpenDDS are examples of publish-subscribe architectures that are based on OMG's DDS standard [13].

A. DDS Architecture

As depicted at Figure 1, the DDS standard is composed of the DDS Interoperability Wire Protocol (DDSI) layer, which sits above the network transport layer, and the DDS layer, which specifies Data-Centric Publish-subscribe (DCPS)[9] behavior and the DDS API [7], used by the application.

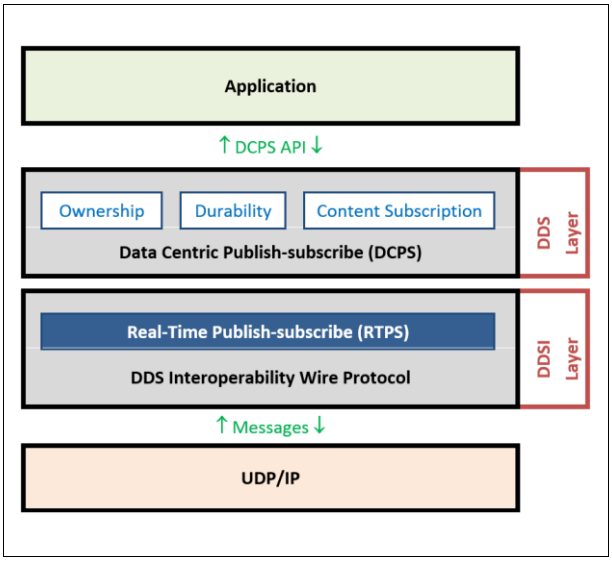

Figure 1: The OMG DDS standard architecture – stack view (reproduced from [8]).

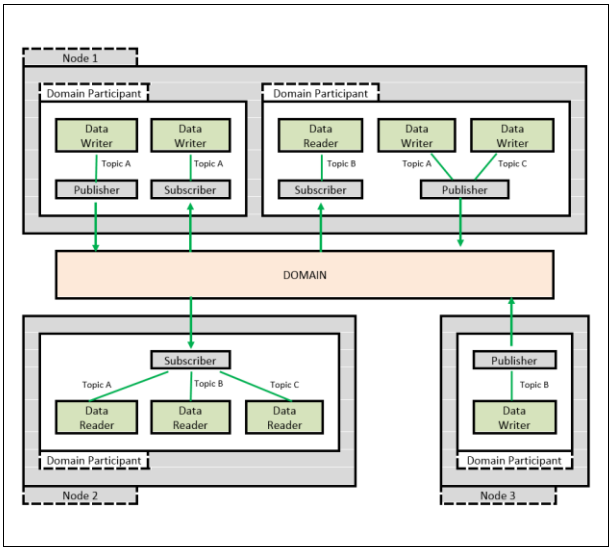

Figure 2: DDS Entities and communication flow (reproduced from [10]).

1) *Data-Centric Publish and Subscribe (DCPS)*

Applications communicate in a Global Data Space by publishing or subscribing to information given by a Topic. DCPS handles the data distribution and enforces QoS restrictions and policies. Figure 2 describes the following concepts:

- **Global Data Space**: is the top-most abstraction of DDS, as it stores all the data published in domains;
- **Domain:** is a virtual data space within the Global Data Space. Entities must be in the same domain to exchange data.
- **Domain Participant:** is the DDS Entity that contains one or more DataReaders and/or DataWriters.
- **DataWriter:** marshals data of a single Topic from the client application to be sent by Publisher on the Domain.
- **Publisher:** disseminates (publishes) data.
- **Subscriber:** receives published data and makes it available to the DataReader associated with the Topic;
- **DataReader**: receives data of a particular Topic from the Subscriber and decodes it for the client application.
- **Topic**: A triple consisting of a unique key, a data type and QoS.

The DDS DCPS specification also defines QoS policies that indicate minimum service levels required for communication and associated constraints [11]. Categories of DCPS QoS policies include:

Ownership: specifies whether the data can be published by multiple entities at the same time (`SHARED` ownership), or if data can only be published by a single entity at a time (`EXCLUSIVE` ownership). *Ownership Strength* specifies the relative priority of multiple data publishers when ownership is `EXCLUSIVE`;

- **Liveliness:** specifies whether an expected entity in the system needs to be active, or if it can have intermittent connectivity;
- **Reliability:** specifies whether or not data can be dropped or late;
- **Lifespan:** specifies the time period in which published data is valid (i.e. before it expires);
- **History:** specifies the desired action when data expires prior to communication to Subscribers; and
- **Resource Limits:** specifies which DDS resources the service can use to meet other QoS requirements.

2) *Real-Time Publish and Subscribe Protocol (RTPS)*

The RTPS specifies a wire protocol. It can be used on top of multicasting and best-effort transport layer protocols such as UDP/IP. RTPS is tailored[8] to support the real-time publish-subscribe requirements needed by DCPS by including timing parameters and properties.

RTPS allows automatic discovery of DataWriters and DataReaders within a Domain, using two discovery protocols:

- **Simple Participant Discovery Protocol (SPDP):** is used to discover all Domain Participants within a Domain; and
- **Simple Endpoint Discovery Protocol (SEDP):** is used for matching DataReaders and DataWriters (i.e. based on Topic) on the Domain.

B. DDS Implementations and Applications

While several commercial and open-source DDS middleware vendor implementations have been developed, the two most important are RTI and OpenDDS.

RTI Connext DDS™ is sold by Real-Time Innovations (RTI). RTI Connext DDS™ is used in many mission critical applications such as ATC, SCADA, machinery control, military C4ISR applications and naval C2 systems.

OpenDDS is an open-source implementation maintained by Object Computing Incorporated (OCI). However, it has not enjoyed the popularity of or seen as much use in large real-time mission critical systems as the other commercial DDS vendor implementations [12].

C. DDS Security

DDS was designed to provide maximum flexibility when building DDS-based system. The host provides a *trusted zone* of execution and relies on physical security measures and network isolation from other systems and outside users. These measures are likely to be imperfect, and it is posited that a host compromise is a likely method of attack and exploitation by DDS-based malware

III. ATTACK METHOD ANALYSIS, DEMONSTRATION AND VALIDATION

Our analysis is based on the following broad categories of threats that could impact DDS-based systems [5]:

- Unauthorized subscription;
- Unauthorized publication;
- Tampering and replay; and
- Unauthorized access to data.

A. DDS Analysis

We posit that the main threat to DDS-based systems is compromising via a client-side attack. We assume that the system has already been compromised through access to either the physical system, the software development environment, or the supply chain. This research therefore focusses on the actions of an attacker to exploit security issues *inside* a DDS-based system.

Analysis of DDS was done via investigation of the DDS specification documentation [1, 2] and DDS vendor implementation documentation and software [15, 16]. We explored the functionality and data structures of DDS to see how it interacts with surrounding elements of a system. Figure 3 shows the key DDS components. The elements in Red indicate potential attack surfaces.

B. Selection of Candidate Attack Methods

The attack methods were selected to ensure coverage of a broad set of attack types including examples of DoS, Hijacking and Network configuration alteration, as well as Data Deletion, Modification, Insertion, Redirection

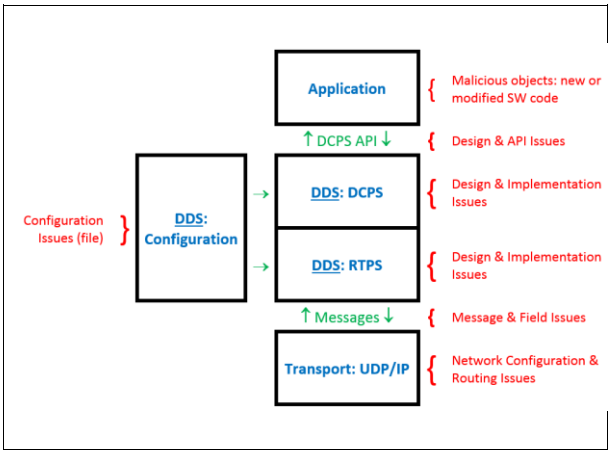

Figure 3: Example DDS Stack and locations of potential security issues (in red).

and Omission. The selected methods covered a broad set of DDS functionality, including: DCPS, RTPS, Discovery mechanisms; and vendor implementation-specific features. The issues selected are relatively easy to implement and are likely to be employed by a malicious actor.

C. Self-Contained Demonstration Environment

Each attack method was demonstrated in a self-contained environment using the DDS Shapes Demo application from RTI Connext DDS™ (v5.2.0), running on several Lubuntu (16.04) Virtual Machines (VMs) in a virtual network using Parallels (v11.2.1) on a MacBook. The DDS Shapes Demo application provided simple methods of specifying unique data types as shapes (squares, triangles and circles) with different qualities, such as color, size and movement rate. The Shapes Demo was run as both publisher and subscriber on the same and on different hosts.

D. Validation & Proof of Concept Environment

The validation scenario demonstrates the attack methods in an end-to-end representative scenario using a DDS-based ATC application. We used the "Vehicle Tracking" example application [17] provided by RTI, running in a host environment identical to the self-contained demonstration environment.

The "Vehicle Tracking" example application uses several DDS Entities to simulate an ATC environment above the San Francisco bay area. DDS Entities include Radar sensors and Operator Display workstations. Radar sensors publish the "Track" Topic to Operator Display workstations subscribers.

IV. ATTACK METHODS DISCUSSION

The initial state of the environments is shown in Figure 4. It illustrates the architectural view of a DDS stack

of DDS Entities (each on its own host), which demonstrates the interaction between DDS and its surrounding elements (configuration and application) on a host and communication (via the Transport) to other hosts on the network.

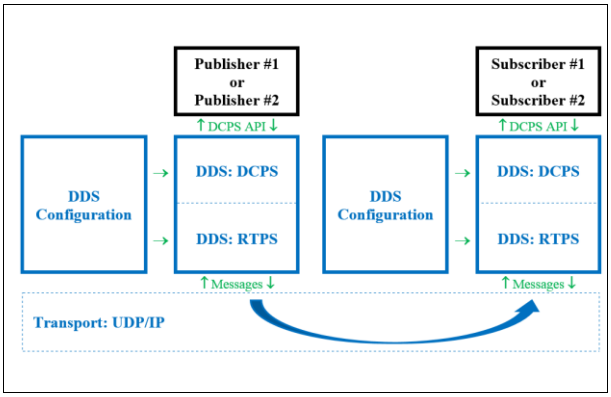

Figure 4: Self-contained Demonstration Environment and Validation Environment: Initial state architectural stack view.

A. *DDS Attack Method #1: Misuse of Anonymous Subscribe and Republish Functionality*

This DDS Attack Method involves the anonymous nature of the publish-subscribe mechanism that is central to DDS. The goal is to show that a new malicious DDS Entity could subscribe to existing data in the system and republish multiple, altered copies of the data, thus acting as a man-in-the-middle. If a DDS Entity that subscribes to this data displays the results to a human operator, the additional data can confuse or overload the operator, especially if the new data is subtly altered to make it indistinguishable from the original data. Figure 4illustrates the initial state of the relevant DDS Entities prior to demonstration of the DDS Attack Method (green arrow lines indicate original data flow).

1) *DDS Attack Method #1: Architecture and Implementation*

Multiple data publication is possible in DDS when the `OWNERSHIP_KIND` QoS policy is set to `SHARED` (vice `EXCLUSIVE`), which is the default and most commonly used data ownership configuration. In order to subscribe to or publish data in DDS, one need only know the data type and the QoS policies used. In the demonstration environment, this was easily found via examination of RTPS packets that contained valid Shapes data as the Shapes Demo source code was not available.

A new malicious application containing a subscriber and publisher was created, which specified the same data type and QoS policies of the target data (i.e. Green Triangle and `SHARED` ownership). The new malicious application multiplies the original data and alter the position of the copied data such that they could be observed by the user.

Figure 5 illustrates the architectural view of the self-contained demonstration (green line indicates original data flow; malicious entity and malicious data flow is shown in red).

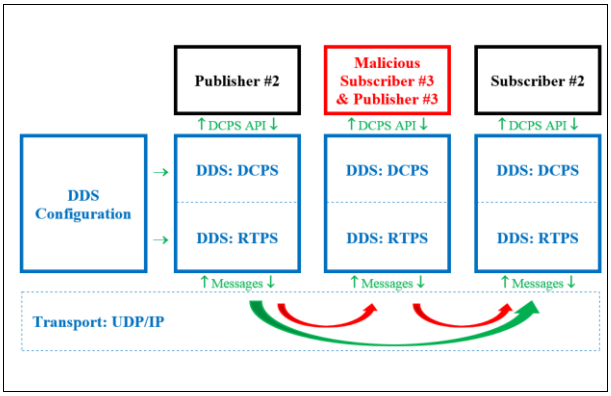

Figure 5: Self-contained Demonstration #1: Architecture stack view (Data Multiplication).

This self-contained demonstration successfully shows that this DDS Attack Method can be used in a data multiplication attack (data modification and data insertion). The new malicious application successfully subscribed to Green Triangle shapes (not shown here for brevity) and republished four copies of them. The original subscriber (Subscriber #2) subscribed to all of them (one original Green Triangle and four new Green Triangles) and is not able to distinguish which one of the five was the original Green Triangle.

2) *DDS Attack Method #1: Validation & Proof of Concept Results*

This attack method is replicated in the validation environment and achieves a "track multiplication and falsification" effect to distract and confuse ATC operators. A new malicious DDS Entity is executed which subscribes to aircraft tracks and republishes multiple copies of slightly altered aircraft tracks. This results in the Operator Display receiving the new aircraft tracks as well as the existing aircraft tracks.

Multiple new tracks, each containing subtle differences from the original track can easily distract, confuse an operator who will find it difficult to determine which tracks are legitimate. This can endanger real aircraft and other friendly assets. Figure 6 illustrates the resulting state of Display #2, which displays aircraft tracks from both Radar #1 and the malicious.

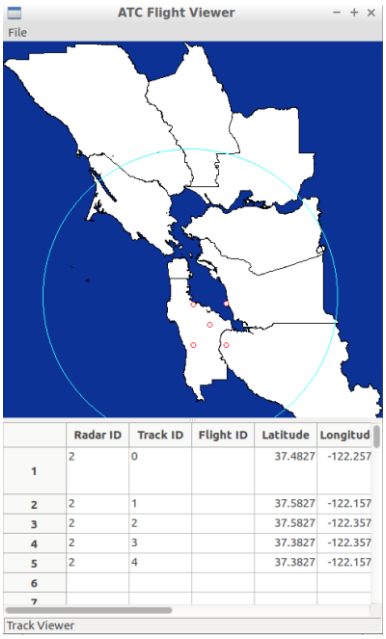

Figure 6: Post-exploit state of Display #2 (Data Multiplication).

B. *DDS Attack Method #2: Misuse of `OWNERSHIP_STRENGTH` QoS Policy and `EXCLUSIVE` Data Ownership*

This Attack Method uses the DDS `OWNERSHIP_STRENGTH` QoS Policy to determine which publisher can publish data for a Topic when `EXCLUSIVE` ownership is used. If multiple publishers publish data for a Topic with `EXCLUSIVE` ownership, only the publisher who currently has the highest `OWNERSHIP_STRENGTH` may publish the data.

This attack allows a malicious DDS publisher to hijack data publication from an existing publisher. Subscribers will receive data from the malicious publisher and no longer receive data from the original publisher, without any indication of change of communication flow. This change in configuration can cause issues for all subscribers to this data, especially if the subscribing application displays the results to a human operator.

1) *DDS Attack Method #2: Architecture and Implementation*

This DDS Attack Method involves exploitation of built-in DDS functionality that allows prioritization of unique data coming from multiple sources. This feature may be effectively used in situations where timely processing is needed and one sensor updates frequently but with lower-quality data, while another sensor updates less frequently but with higher-quality data – the subscribing application would only want the data from the source with the highest quality (i.e. highest ownership strength), when available. `EXCLUSIVE` ownership, which ensures only one publisher at a time can publish data within an update period, can handle multiple publishers using different `OWNERSHIP_STRENGTH` values. In order to publish malicious data, all that is required is knowledge of the data type, the QoS policies used in the communication flow and determination of an `OWNERSHIP_STRENGTH` value that would be higher than what is currently used. In the demonstration environment, this is easily discernable via examination of network traffic of valid Shapes data.

A new Shapes Demo DDS Entity, Publisher #4 is launched and published Blue Squares with a higher value of `OWNERSHIP_STRENGTH` (i.e. a value of `10` versus the original value of `9`). The Blue Squares published by Publisher #4 contains an additional "cross-hatch" `Fill` attribute that does not affect subscription but allows easier observation by the user demonstrating the DDS attack method.

Figure 7 illustrates the architectural view of the entities involved in the demonstration of this attack method. DDS Entities included unaltered DDS Entities (Publisher #1 and Subscriber #1), and the new malicious DDS Entity (Publisher #4).

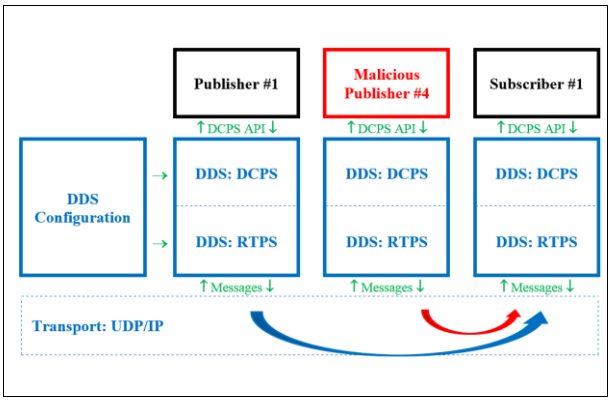

Figure 7: Self-contained Demonstration #2: Architecture stack view (Data Hijacking).

2) *DDS Attack Method #2: Validation & Proof of Concept Results*

This attack method was replicated in the validation environment and achieved a "track hijacking" style of attack to take over publication of aircraft tracks from one Radar to a new malicious Radar.

A new malicious DDS-based application is executed from a new host during normal operation of the ATC system. This results in an Operator Display displaying illegitimate and untrustworthy tracks from the new malicious application. Trusting invalid or inaccurate aircraft track data can put the safety of the real aircraft and other friendly assets in danger.

C. *DDS Attack Method #3: Misuse of `OWNERSHIP_KIND` QoS Policy and `SHARED` Data Ownership*

This DDS Attack Method involved the use of DDS `OWNERSHIP_KIND` QoS Policy which determines how data is distributed. The `OWNERSHIP_KIND` for data of a Topic must match on both the publisher and the subscriber in order for communication to occur.

A malicious change in the configuration of a DDS publisher can redirect the data from one set of subscribers to another set of subscribers. The original subscribers no longer receive the data from the publisher without any indication of change in communication flow. This change in configuration could cause issues for the original subscriber, especially if it displays the results to a human operator – the operator would no longer see the data and would not necessarily know that any data was missing. In addition, if the new subscriber to this data also displays the results to a human operator – the operator might be confused or overloaded by this new unexpected data.

1) *DDS Attack Method #3: Architecture and Implementation*

The `OWNERSHIP_KIND` of a Topic can be either `SHARED` or `EXCLUSIVE`, and it must match on both the publisher and subscriber in order for communication to occur. When the `OWNERSHIP_KIND` is set to `SHARED`, all DDS Entities can publish and subscribe to the data; when set to `EXCLUSIVE`, only one DDS Entity at a time can publish data, and only DDS subscribers with an `OWNERSHIP_KIND` set to `EXCLUSIVE` can subscribe to the data.

This demonstration changes the `OWNERSHIP_KIND` for the Triangle Topic on Publisher #1 from `EXCLUSIVE` to `SHARED`. This reroutes data away from Subscriber #1, whose `OWNERSHIP_KIND` is `EXCLUSIVE` and sends the data to Subscriber #2, whose `OWNERSHIP_KIND` is `SHARED`.

The alteration of the `OWNERSHIP_KIND` QoS Policy on Publisher #1 involves changing two configuration files. The Shapes Demo application hard-coded the location of a configuration file, which contains one line that sets the default QoS Policy values to those found in the RTI DDS library. Since this file is designed to be user modifiable (and given that the DDS philosophy is to allow customization to occur in the QoS files [1, 2]), it is first altered to remove the reference to the default QoS Policy values in the RTI DDS library. Then, a new file is created with maliciously altered QoS Policy values. This file is then placed into the same directory as the script that executes the application. Note that applications built using the DDS libraries automatically look for and read a file with the correct name on startup, which makes an ideal target to inject malicious QoS Policy settings.

Figure 8 illustrates the architectural view of the DDS Entities involved in this attack method. DDS Entities included unaltered DDS Entities (Subscriber #1 and Subscriber #2), and DDS Entities with maliciously altered configuration (Publisher #1). Note that only the configuration of Publisher #1 is altered – the executable remained unaltered.

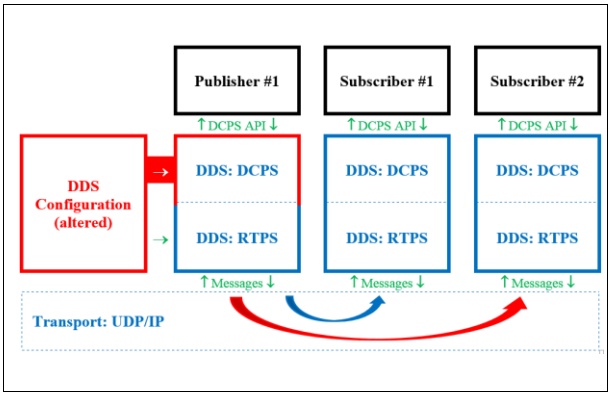

Figure 8: Self-contained Demonstration #3: Architecture stack view (Data Redirection)

2) *DDS Attack Method #3: Validation & Proof of Concept Results*

This attack method is replicated in the validation environment and achieves a "track hiding & redirection" attack to hide aircraft tracks from some Operator Displays and redirect it to other Operator Displays. This involves exploiting a periodic restart of a Radar that reads a maliciously-altered configuration file upon startup. Changes to the configuration includes a change to the `OWNERSHIP_KIND` of published aircraft track data from `EXCLUSIVE` to `SHARED`.

This malicious configuration change resulted in some Operator Displays, now receiving the new aircraft tracks from the maliciously altered Radar in addition to the previous aircraft tracks from the Radar that it normally receives tracks. It also resulted in other Operator Displays that normally received tracks from the maliciously-altered Radar to no longer receive aircraft tracks from that Radar.

D. *DDS Attack Method #4: Misuse of `LIFESPAN` QoS Policy Causing Immediate Data Expiration*

This DDS Attack Method involved the use of DDS `LIFESPAN` QoS Policy which determines the length of time data is valid. The `LIFESPAN` for a data Topic determines the length of time before the data is deleted – deletion can occur at the subscriber or even prior to publication by the publisher if the `LIFESPAN` is short enough.

A malicious change in the configuration of a DDS publisher can cause valid data to either not be sent to subscribers or not be processed by subscribers. Specifically, a low value for the LIFESPAN QoS Policy for a specific data Topic may cause these effects. This change in configuration could cause issues for a subscribing application, especially if it displays the results to a human operator – the operator would no longer see the data and would not know that any data was missing.

1) *DDS Attack Method #4: Architecture and Implementation*

This DDS Attack Method involves the alteration of the LIFESPAN QoS Policy for the Green Triangle Topic data on a publisher. Depending on the LIFESPAN value and processing speed, data can be deleted at one of three points: while in the history cache of the publisher prior to publication; while in the history cache of the subscriber after it was received but prior to processing by the subscribing application; and after initial processing and display by the subscribing application.

The demonstration involves specifying a LIFESPAN value of 100ms for the Triangle Topic the publisher. A value of 100ms resulted in RTPS messages sent from the publisher, and occasional Green Triangles displayed at the subscriber. This was due to the data being automatically deleted while in the subscriber's history cache – the application was able to use some of the data prior to DDS automatically deleting the data.

The alteration of the LIFESPAN QoS Policy on involves the same methods as were described in DDS Attack Method #3 – refer to section IV.C.1) for details.

Figure 9 illustrates the architectural view of the DDS Entities involved in self-contained demonstration of this DDS Attack Method (blue line indicates original data flow; malicious entity is shown in red; and dotted green line indicates intermittent, altered data flow). DDS Entities included unaltered DDS Entities (Publisher #1 and Subscriber #2), and DDS Entities with maliciously altered configuration (Publisher #2). Note that only the configuration of Publisher #2 was altered – the executable remained unaltered.

This demonstration successfully shows that this DDS Attack Method can be used in a data omission or data deletion attack. The altered configuration of Publisher #2 results in the subscriber receiving intermittent data (Green Triangles) from Publisher #2. Existing data (Blue Triangles) from Publisher #1 is not affected.

The value of the LIFESPAN QoS Policy for the Triangle Topic on Publisher #2 greatly affects the results.

Several LIFESPAN values were tested with the following general results:

LIFESPAN < 80ms: resulted in no RTPS messages sent from Publisher #2, and therefore no Green Triangles were displayed at Subscriber #2 (i.e. they were automatically deleted while in the Publisher's history cache);

LIFESPAN 80ms - 120ms: resulted in RTPS messages sent from Publisher #2, and occasional Green Triangles were displayed at Subscriber #2 (i.e. before they were automatically deleted while in the Subscriber's history cache); and

LIFESPAN >= 120ms: resulted in RTPS messages set from Publisher #2, and Green Triangles were displayed at Subscriber #2 (i.e. they were not deleted).

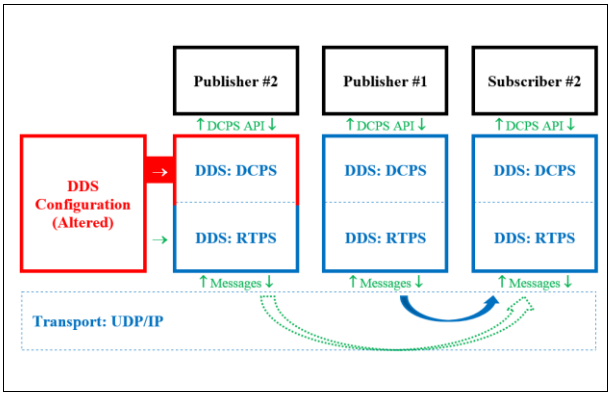

Figure 9: Self-contained Demonstration #4: Architecture stack view (Data Deletion).

The RELIABILITY QoS Policy specifies that reliable transmission of the data is required when it is set to RELIABLE; DDS will continue to retransmit data until receipt confirmation (from the subscriber) is received. Otherwise, DDS uses the default of BEST_EFFORT, which specifies that the packet is transmitted once, with no requirement of receipt confirmation. The DDS Shapes Demo used the default value of BEST_EFFORT, but when set to RELIABLE, the above results were reduced by a factor of 10 (i.e. 8ms was used instead of 80ms to achieve the same results).

2) *DDS Attack Method #4: Validation & Proof of Concept Results*

This attack method is replicated in the validation environment and achieves a "track deletion" style of attack to hide aircraft tracks from all Operator Displays. This involves exploiting a periodic restart of a Radar that reads a maliciously-altered configuration file upon startup. Changes to the configuration included a reduction to the LIFESPAN of published aircraft track data such that it was short enough that it either did not get published, or it expired and was deleted upon receipt at any Operator Display.

This resulted in a Display that normally received tracks from this Radar to no longer receive them, although it still received aircraft tracks from other Radars.

Note that different values of LIFESPAN were used with the ATC Example application versus the Shapes Demo application (discussed previously). These values are listed below:

- **LIFESPAN < 0.8ms**: resulted in aircraft track information being automatically deleted prior to being sent;
- **LIFESPAN 0.8ms – 1.2ms**: resulted in occasional aircraft track information being automatically deleted upon receipt, prior to being displayed; and
- **LIFESPAN >= 1.2ms**: resulted in aircraft all track information being displayed (i.e. not automatically deleted).

The LIFESPAN values were lower than those used by the Shapes Demo application by a factor of 100 for two reasons:

- The ATC Example application used RELIABILITY QoS Policy equal to RELIABLE versus BEST_EFFORT, as was used in the Shapes Demo application. This accounted for a reduction in the LIFESPAN value by a factor of 10; and
- The ATC Example application executed more efficiently and with a lower load than the Shapes Demo application. This accounted for another reduction in the LIFESPAN value by a factor of 10.

E. DDS Attack Method #5: Misuse of the LocatorList Environment Variable Causing Domain Misdirection During Participant Discovery

This attack method involves the use of the DDS LocatorList environment variable which specifies the IP address contacted by a new DDS Entity to initiate the DDS Simple Participant Discovery Protocol (SPDP). If the LocatorList environment variable specifies a different IP address from what is used by the other DDS Entities, and this IP address is malicious, it could be used to direct the newly started DDS Entity onto a different DDS Domain or network.

1) DDS Attack Method #5: Architecture and Implementation

This DDS Attack Method exploits the ability to conveniently specify a network address in which the SPDP process looks for other DDS Participants. In many cases, this involves a multicast IP address to which all DDS Entities register, although a centralized discovery scheme can be implemented using a broker at a unicast address. This is less common and is less resilient as it requires a centralized node.

When a new DDS Entity attempts to join the DDS network, it sends a message to this multicast address to advertise its presence, which is then received by all existing DDS Entities that have registered to that address. Information contained within this initial message contains information which is used to negotiate communication flow details with the existing DDS SEDP uses different IP addresses from the initial SPDP address specified by the NDDS_DISCOVERY_PEERS environment variable. Malicious alteration of the first step in the discovery process (i.e. SPDP) had the effect of hijacking all DDS-RTPS communication involving that DDS Entity.

This self-contained demonstration involved specifying a value of NDDS_DISCOVERY_PEERS equal to 239.200.0.1 for both the malicious publisher and the hijacked subscriber. This value was specified in the startup script that ran the application containing the DDS Entity (note that it could also have been specified in any startup script that the operating system uses to set global environment variables). Benign DDS Entities used the default value of 239.255.0.1, which involved no alteration of configuration files. All DDS Entities used the same Shapes Demo application – the only difference was the change in NDDS_DISCOVERY_PEERS for both the malicious Publisher #5 and the hijacked Subscriber #1.

The new Shapes Demo DDS Entity, malicious Publisher #5, was launched prior to the configuration change and subsequent reboot of Subscriber #1. Malicious Publisher #5 published Blue Squares with an additional "cross-hatch" Fill attribute that did not affect subscription, but it allowed easier observation by the user demonstrating the DDS attack method.

Figure 10 illustrates the architectural view of the DDS Entities involved in self-contained demonstration of this DDS Attack Method. DDS Entities include unaltered DDS Entities (Publisher #1), malicious DDS Entities (Publisher #5), and DDS Entities with maliciously altered configuration (Subscriber #1). Note that only the configuration of Subscriber #1 was altered – the executable remained unaltered.

2) DDS Attack Method #5: Validation & Proof of Concept Results

This attack method is replicated in the validation environment and achieves a "host hijacking" style of attack to redirect an Operator Display onto a malicious DDS Domain and then receive false aircraft track data from a malicious Radar. This involves exploiting a periodic restart of a Display with a maliciously altered startup script file. This altered DDS Discovery configuration, as shown in Figure 10, redirected the Display to join a malicious DDS network (which contained a new malicious Radar).

This caused the Display to receive the aircraft tracks from the malicious Radar on the malicious DDS Domain. As discovered in the self-contained demonstration environment, after several minutes the Display also started to receive valid aircraft tracks from the legitimate Radar,

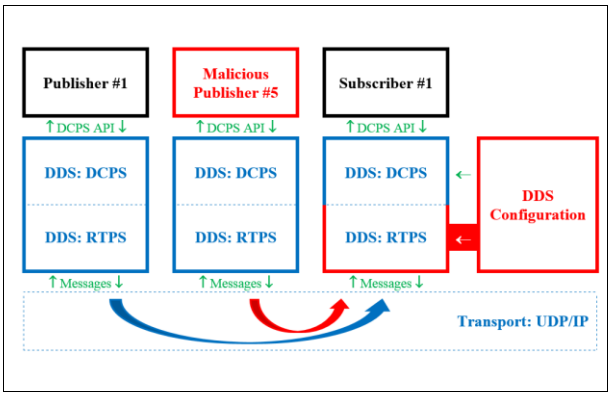

Figure 10: Self-contained Demonstration #5: Architecture stack view (Host Hijacking).

due to the fact that the legitimate Radar still knew the address of the Display.

Multiple new tracks from a malicious Radar could easily distract, confuse and overload an operator who would find it difficult to determine which aircraft tracks were legitimate. This could put the safety of the real aircraft and other friendly assets in danger.

In addition, since new aircraft tracks were inserted from a DDS Entity that is not on the original DDS Domain, there would be little indication of malicious activity. Anomalous behavior introduced by an undetected malicious DDS Entity could also put the reliability and credibility of the DDS-based ATC system into question.

V. SUMMARY AND DISCUSSION

This paper details five potential attack methods which could be used in a *client-side* attack against DDS-based systems. These were based on five DDS Security Issues (selected from 60 DDS Security Issues [18]). These five attack methods are modelled and demonstrated in a self-contained environment and then combined into an end-to-end validation scenario. A DDS-based ATC application was used in the validation scenario to provide context and a real-world example of the seriousness of DDS security issues. The validation scenario demonstrated the same effects as are demonstrated in the self-contained environments.

A. Current DDS Security Work

There have been several efforts to improve the security of DDS-based systems, by adding security features to the DDS middleware. These have included several interim extensions to the middleware that have addressed specific customer issues and the recently released OMG DDS Security Specification 1.0 [5].

1) Secure Transport (ST) and Discovery Service

Pradhan, et al, propose an approach to handling DDS security issues on an OpenDDS-based system that manages distributed satellite clusters [14]. Their approach consists of adding a new transport mechanism called Secure Transport (ST) and a new discovery service.

The new ST mechanism enforces information partitions based on security classifications/labels. The new discovery service authorizes and establishes ST information flows between matching Publishers and Subscribers based on their security labels.

This approach still does not adequately address DDS security issues because it only addresses issues that may arise from new DDS entities. It provides no protection against existing DDS entities that have been compromised. In addition, this approach relies on a new DDS service which itself may be compromised via a client-side attack.

2) Adaptive Discovery Framework

Existing DDS vendor implementations use multicast mechanisms based on a predefined static network definition. While this allows dynamic reconfiguration within smaller networks, it does not work with Wide Area Networks (WANs). Large DDS-based systems including systems implementing Net-Centric Operations and Warfare (NCOW) as proposed by the US DoD, operate over WANs. Therefore Wang, et al, propose an Adaptive Discovery Framework for WAN-based networks that dynamically reconfigures the underlying multicasting mechanisms used by DDS [9].

The proposed Adaptive Discovery Framework also addresses Information Assurance (IA) issues, including authentication, access control, information integrity and encryption. IA functionality has been placed into lowest level DDS node discovery services to ensure that mandatory security mechanisms (including those that provide the bootstrap services for DDS nodes), cannot be circumvented or tampered with.

3) DDS Security Specification

A new DDS Security Specification [5] has been published and RTI sells a product that implements the new standard called RTI Connext DDS Secure™ [6]. It covers five specific security areas: Authentication, Access Control, Encryption, Logging and Data.

B. Future Work

This research work only models, demonstrates and validates five of the 60 DDS security issues [18]. The same process we have shown can be used to explore, demonstrate and validate the 55 issues and result in more numerous, and potentially more effective, methods of compromising a DDS-based system.

The new DDS Security Specification [5] is designed to be backward-compatible with legacy "non-secure" implementations of DDS. Potential vulnerabilities may exist at the points in which legacy DDS-based systems communicate with secure DDS-based systems. Modelling and self-contained demonstration of the five DDS attack methods revealed in this research on a DDS-based system that has secure entities communicating with legacy non-secure entities may have interesting results. It may show that these security issues are not resolved, or it may reveal new security issues.

Further research is also needed to determine the security of different vendor implementations of DDS. Real-world evaluation and testing are required to determine the ease with which client-side attack can compromise DDS-based systems built from different vendor implementations. As well, an assessment of the specific damage that can be done if a DDS-based system is compromised is also required.

VI. CONCLUSION

This research explores 5 of 60 DDS security issues through the analysis of the DDS protocol and several DDS vendor implementations. It models and demonstrates five of these security issues in a self-contained and isolated environment that they could be used as attack methods on a DDS-based system. This research demonstrated that these attack methods can be exploited by a malicious actor in a client-side attack on a DDS-based system.

We are aware of no other published analysis showing how a DDS-based system may be manipulated in order to support a *client-side* attack. Much research exists concerning how to compromise a system from the outside, but we have found no public research work showing what methods a *client-side* attack on a DDS-based system might use. To the best of our knowledge, this research is unique in exposing these DDS security issues.

Knowledge of DDS security issues and how an attacker might use them to compromise a DDS-based system is the first step in being able to detect and defend against a cyberattack on an ATC system or any other DDS-based critical infrastructure system.

VII. REFERENCES

[1] "OMG DDS Specification | DDS 1.4." [Online]. Available: http://www.omg.org/spec/DDS/1.4/. [Accessed: 20-Jun-2016].

[2] "OMG DDS Specification | RTPS 2.2." [Online]. Available: http://www.omg.org/spec/DDSI-RTPS/2.2/. [Accessed: 20-Jun-2016].

[3] J. M. Schlesselman, G. Pardo-Castellote, and B. Farabaugh, "OMG data-distribution service (DDS): architectural update," in *2004 IEEE Military Communications Conference, 2004. MILCOM 2004*, 2004, vol. 2, pp. 961–967 Vol. 2.

[4] "OMG DDS Specification | Security 1.0." [Online]. Available: http://www.omg.org/spec/DDS-SECURITY/1.0/. [Accessed: 14-Feb-2016].

[5] "RTI Connext DDS Secure." [Online]. Available: https://www.rti.com/products/secure.html. [Accessed: 01-Dec-2015].

[6] J. H. van't Hag, "'Data-centric to the max', the SPLICE architecture experience," in *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings*, 2003, pp. 207–212.

[7] J. Yang, K. Sandstrom, T. Nolte, and M. Behnam, "Data Distribution Service for industrial automation," in *2012 IEEE 17th Conference on Emerging Technologies Factory Automation (ETFA)*, 2012, pp. 1–8.

[8] N. Wang, D. C. Schmidt, H. van't Hag, and A. Corsaro, "Toward an adaptive data distribution service for dynamic large-scale network-centric operation and warfare (NCOW) systems," in *IEEE Military Communications Conference, 2008. MILCOM 2008*, 2008, pp. 1–7.

[9] C. Eryigit and S. Uyar, "Integrating agents into data-centric naval combat management systems," in *23rd International Symposium on Computer and Information Sciences, 2008. ISCIS '08*, 2008, pp. 1–4.

[10] "OpenDDS - Articles - Introduction to OpenDDS." [Online]. Available: http://www.opendds.org/Article-Intro.html. [Accessed: 01-Dec-2015].

[11] A. M. Kulkarni, V. S. Nayak, and P. V. R. R. B. Rao, "Comparative study of middleware for C4I systems Web Services vis-a-vis Data Distribution Service," in *2012 International Conference on Recent Advances in Computing and Software Systems (RACSS)*, 2012, pp. 305–310.

[12] C. Esposito, S. Russo, and D. Di Crescenzo, "Performance assessment of OMG compliant data distribution middleware," in *IEEE International Symposium on Parallel and Distributed Processing, 2008. IPDPS 2008*, 2008, pp. 1–8.

[13] S. Pradhan *et al.*, "Establishing Secure Interactions across Distributed Applications in Satellite Clusters," in *2014 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 2014, pp. 67–74.

[14] "OpenDDS Documentation | Developers Guide." [Online]. Available: http://opendds.org/documents/. [Accessed: 20-Jun-2016].

[15] "RTI Connext Documentation | Core Libraries User Manual." [Online]. Available: https://community.rti.com/documentation. [Accessed: 20-Jun-2016].

[16] "RTI Use Case - Track and Monitor Vehicles and Assets." [Online]. Available: https://www.rti.com/resources/usecases/vehicle-tracking. [Accessed: 02-Sep-2016].

[17] M. J. Michaud, "Malicious Use of OMG Data Distribution Service (DDS) In Real Time Mission Critical Systems", MASc, Royal Military College of Canada, 2017.

[18] M. J. Michaud, S. P. Leblanc, "Vulnerability Analysis of the OMG Data Distribution Service (DDS)", ECE-2017-01 Royal Military College of Canada Computer Security Laboratory, 2017.