MALICIOUS USE OF OMG DATA DISTRIBUTION SERVICE (DDS) IN REAL-TIME MISSION CRITICAL DISTRIBUTED SYSTEMS

UTILISATION MALVEILLANT DU SERVICE DE DISTRIBUTION DE DONNÉES *DDS* D'OMG DANS DES SYSTÈMES DISTRIBUÉS CRITIQUES DE MISSION EN TEMPS RÉEL

A Thesis Submitted

to the Division of Graduate Studies of the Royal Military College of Canada

by

Michael James Michaud, B.Eng.,

Lieutenant (Navy)

In Partial Fulfillment of the Requirements for the Degree of

Master of Applied Science in Electrical and Computer Engineering

March, 2017

Dedicated to wife Tanya who worked tirelessly to make sure I was fed and clothed, and also dedicated to my two children Sarah and Samantha, who have endured the many days and nights away from their "Waddy".

Acknowledgements

I would like to acknowledge the tremendous support and assistance of my supervisors and of the Electrical and Computer and Engineering departments at both Queen's University and the Royal Military College of Canada. Their support and consideration made this an experience that I will look back on with great fondness.

Abstract

Michaud, Michael James. M.A.Sc. Royal Military College of Canada, March, 2017. *Malicious Use of OMG Data Distribution Service (DDS) in Real-Time Mission-Critical Distributed Systems.* Supervised by S.P. Leblanc, Ph.D. and T. Dean, Ph.D.

Object Management Group (OMG)'s Data Distribution Service for Real-Time Systems (DDS) middleware standard is a popular technology used in mission-critical distributed real-time, data-centric systems, including Air Traffic Control (ATC) systems. DDS is the core integrating technology for an ATC system to track and guide aircraft. DDS is also used in many command and control systems and critical infrastructure systems therefore it is critical that these DDS-based systems function properly.

This research investigated how DDS could be misconfigured or manipulated to support malicious activity within a DDS-based system. Since many DDS-based systems are well isolated from other systems and from outside users, the focus of the research was on the methods involved in a client-side attack on a DDS-based system and the effects of the compromise. This research revealed sixty DDS security issues through the analysis of the DDS protocol and several DDS vendor implementations. Five of these DDS security issues were modelled and demonstrated in a self-contained and isolated environment to demonstrate how they could be exploited to compromise a feature of DDS. These DDS security issues were then validated using an end-to-end attack scenario using a different DDS-based system. This research demonstrated that these DDS security issues could be effectively exploited by a malicious actor in a client-side attack on a DDS-based system.

This research addresses a little-researched area of DDS security: the identification of DDS security issues and how they might be exploited by a client-side attack to compromise a DDS-based system. The result of this research work enables further work to be done in being able to detect and defend against a cyberattack on an ATC system, control system or any other DDS-based critical infrastructure system.

Keywords: distributed computing, Data Distribution Service, DDS, middleware, mission critical, Object Management Group, OMG, OpenDDS, Quality of Service, QoS, real time, RTI, RTPS, SCADA and security.

Résumé

Michaud, Michael James. M.Sc.A. Collège militaire royal du Canada, March, 2017. *Utilisation malveillant du Service De Distribution De Données DDS d'OMG dans des systèmes distribués critiques de mission en temps réel*. Thèse dirigée by S.P. Leblanc, Ph.D. et T. Dean, Ph.D.

Le standard de middleware DDS (service de distribution de données pour les systèmes temps réel) du Groupe de gestion des objets (OMG) est une technologie populaire utilisée dans les systèmes informatiques répartis en temps réel, y compris les systèmes de contrôle de la circulation aérienne (ATC). DDS est la principale technologie d'intégration d'un système ATC pour suivre et guider les avions. DDS est également utilisé dans de nombreux systèmes de contrôle industriels et systèmes d'infrastructure critiques, il est donc essentiel que ces systèmes basés sur DDS fonctionnent correctement.

Cette recherche a étudié comment DDS pourrait être mal configuré ou manipulé pour soutenir l'activité malveillante dans un système basé sur DDS. Étant donné que de nombreux systèmes basés sur DDS sont bien isolés des autres systèmes et des utilisateurs extérieurs, la recherche a porté sur les méthodes impliquées dans une attaque côté client sur un système basé sur DDS et les effets du compromis. Cette recherche a révélé soixante problèmes de sécurité DDS grâce à l'analyse du protocole DDS et plusieurs implémentations de fournisseurs DDS. Cinq de ces problèmes de sécurité DDS ont été modélisés et démontrés dans un environnement isolé pour démontrer comment ils pourraient être exploités pour compromettre une fonctionnalité de DDS. Ces problèmes de sécurité DDS ont ensuite été validés en utilisant un scénario d'attaque de bout en bout utilisant un autre système basé sur DDS. Cette recherche a démontré que ces problèmes de sécurité DDS pourraient être efficacement exploités par un acteur malveillant dans une attaque côté client sur un système basé sur DDS.

Cette recherche porte sur un domaine peu documenté de la sécurité DDS: l'identification des problèmes de sécurité DDS et la façon dont ils pourraient être exploités par une attaque côté client sur un système basé sur DDS. Le résultat de ce travail de recherche permet de poursuivre les efforts pour détecter et se défendre contre une cyberattaque sur un système ATC, un système de contrôle ou tout autre système d'infrastructure critique basé sur DDS.

Table of Contents

List of Tables

List of Figures

List of Acronyms

A-IDS Anomaly-Based Intrusion Detection System

API Application Programming Interface

ATC Air Traffic Control

AVL Automatic Vehicle Location

C2 Command and Control

C4ISR Command, Control Communications and Computer Intelligence, Surveillance and Reconnaissance

CA Certificate Authority

CORBA Common Object Broker Architecture

CST Composite State Transfer

CVIS Cooperative Vehicle Infrastructure Systems

DCOM Distributed Common Object Model

DCPS Data-Centric Publish and Subscribe

DDS Data Distribution Service for Real-Time Systems

DDSI DDS Interoperability wire protocol layer

DLRL Data Local Reconstruction Layer

DoD Department of Defence (US)

DSA Digital Signature Algorithm

EJB Enterprise JavaBeans

ENISA European Union Agency for Network and Information Security

GUI Graphical User Interface

IA Information Assurance

IDS Intrusion Detection System

IEC International Electrotechnical Commission

IETF Internet Engineering Task Force

IoT Internet of Things

IP Internet Protocol

JMS Java Message Service

MLS Multi-Level Security

NCOW Net-Centric Operations and Warfare

NFC Near-Field Communication

OCI Object Computing Incorporated

OMG Object Management Group

OWASP Open Web Application Security Project

PKI Public Key Infrastructure

POS Point of Sale

PS Publish-Subscribe protocol

PSR Primary Surveillance Radar

QoS Quality of Service

RFC Request For Change

RTI Real-Time Innovations

RTPS Real-Time Publish Subscribe protocol

SCADA Supervisory Control and Data Acquisition

SEDP Simple Endpoint Discovery Protocol

SM Security Model

SOAP Service Oriented Architecture Protocol

SPDP Simple Participant Discovery Protocol

SPI Security Plugin Interface

SSR Secondary Surveillance Radar

ST Secure Transport

TCP/IP Transmission Control Protocol / Internet Protocol

UDP/IP Unreliable Datagram Protocol / Internet Protocol

VM Virtual Machine

WAN Wide Area Network

Chapter 1 Introduction

1.1 Introduction

OMG's DDS middleware standard [1, 2] defines a standard for middleware that lies between the application software and the network transport. 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 manages the details of data distribution. 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. More importantly, DDS includes Quality of Service (QoS) standards that ensure the integrity and timely delivery of data, which is critical for information management, surveillance and control systems.

The publish-subscribe model that DDS uses supports an "information-centric" or "data-centric" communications pattern. This supports a distributed architecture and allows distributed applications to be designed based on the data type and specified QoS qualities like reliability, persistency, ownership, lifetime, latency and data order. A "Data-centric" architecture allows communication that is custom tailored to the specific requirements of a distributed system, without having to specify a particular network architecture or messaging scheme [3].

In other words, DDS removes the burden from the application developer, of needing to know details of how to communicate with the other applications in the system. A data publishing application does not need to know about whom or where the data is going to; it only needs to know about the specific data type that is being communicated. A data subscribing application does not need to know about how or where data is coming from; it only needs to know about the specific data type (i.e. the "Topic") that it wants to receive [3]. Configurable Quality of Service (QoS) requirements allow DDS to ensure that the proper data communication requirements are adhered to by all participating DDS Entities that access the data Topic.

The flexible publish-subscribe mechanisms of DDS that make it ideal for application developers, give rise to many potential security issues. 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 easily join the DDS network. These issues are compounded by the fact that

the current DDS standard [1, 2] lacks common security measures including access control, authentication, information partitioning and encryption.

1.2 Motivation

DDS is ubiquitous in mission-critical distributed real-time, data-centric systems, which include machinery control systems, energy control systems, ATC systems and military Command and Control (C2) systems used in aircraft and ships [4]. DDS is also used in Canadian ATC systems to track and guide aircraft safely through Canadian airspace. An ATC system's ability to operate correctly and provide accurate information to an air traffic controller fully depends on the proper functioning of DDS.

DDS was designed to provide a developer maximum flexibility when building their DDS-based system and to expedite system development and deployment. DDS removes the burden from the application developer, of needing to know details of how to communicate with the other entities in the system. A malicious DDS-based application could take advantage of this communication flexibility built-in to DDS. A new or existing DDS application could be developed or modified easily with malicious intent.

While there have been some attempts to secure DDS-based systems, these security measures and improvements have focused on securing the data [6], and little attention has been directed to the host running the DDS-based applications. Securing the host containing the DDS-based applications has relied on physical security measures and network isolation from other systems and outside users. This has been done to provide a *trusted zone* where DDS could execute. It is posited that the main threat would be from a compromised host on which the DDS-based system runs, via a myriad of client-side attacks.

DDS security research has been done before, however the research has focused on how to add functionality to DDS in order to address specific requirements of a particular customer. OMG has developed a new DDS Security specification [5] in order to resolve DDS security issues in a wide-scale and coherent manner. Real Time Innovations (RTI) is a DDS vendor that has implemented this new specification, which is called RTI Connext DDS Secure™ [6]. This solution still relies on the notion that the hosts upon which DDS applications execute, are a *trusted zone*. The assumption that the hosts are a *trusted zone* does not hold and this solution does not address any security concerns that may arise from a compromised host. Therefore, any

DDS-based system developed with the new DDS Security features is still vulnerable to effects of a client-side attack.

DDS security research has been done in preparation of the new DDS Security specification [5] by OMG and RTI, but little of this proprietary research has been made public. Therefore, our research work is extremely important as no comprehensive DDS security analysis has been done in a forum accessible to developers of Canadian DDS-based products. In addition, no public DDS security research exists that would be useful in developing DDS threat signatures for use in an anomaly-based Intrusion Detection System (IDS) such as the DDS-IDS being developed by Queen's University.

Furthermore, there is also no published analysis of how a DDS-based system may be compromised via a client-side attack on altered software, configuration or DDS libraries. There also has been no published research on possible indications of this type of DDS compromise, or what possible outcomes would result from a host-based DDS compromise.

1.3 Motivating Scenario

ATC systems are used to manage information provided by Radars and other surveillance systems, and provide this information to a human air traffic controller, who uses this information to guide aircraft safely through the air. Many ATC systems are multiple host, real-time, data distributed systems based on DDS and operate within the confines of physically secure facilities with little or no direct communication with other systems. The isolated nature of how DDS is used in these systems means that an outside network-based attack is very unlikely. It is more likely that malicious activity would involve client-side access to the system through the same hardware, or consist of altered software, configuration files or libraries via a compromised software development environment or software delivery process. The architecture and operating environment of DDS-based ATC systems are very similar to other DDS-based systems in power plants, machinery control systems and other critical infrastructure systems. Therefore, for simplicity this thesis will only describe the environment and situations in which ATC systems operate.

Safe operation of an ATC system relies upon reliable, timely and accurate aircraft track data being transmitted, processed and displayed to an operator. Manipulation of the data to alter current sensor reports, to insert new sensor reports or to cause sensor reports to be ignored or

deleted could have dire consequences. For example, if aircraft track data was changed (e.g. position or altitude), then an operator might make a decision that leads to a mid-air collision and loss of life, or leads to the false-determination of an aircraft's intent (i.e. is the aircraft hostile or is it reporting an emergency?). If new invalid aircraft tracks were inserted, it might lead to operator overload or incorrect decisions by the operator. If aircraft track data was manipulated or deleted such that it did not appear on an operator display, it could lead to a mid-air collision due to lack of situational awareness. It is therefore paramount that any malicious activity within the DDS-based system be detected such that corrective measures be taken prior to any loss of life.

Malicious activity could come from altered software or configuration of the DDS host that interfaces to the sensor (e.g. a Radar) or the DDS host that consumes or displays the data (e.g. an operator workstation display). Alternatively, a new malicious software process could be created to achieve the same result. The inherent flexibility of DDS makes these effects easy to implement.

ATC systems (and many other DDS-based systems) are isolated from other networks and systems, they operate in physically isolated in access-controlled environments, and details of their implementation (and sometimes existence) tend to not be well-known. Therefore, the main threat would likely be a host compromise via a client-side attack developed with somewhat detailed knowledge of the system, its use and its architecture and implementation.

As discussed above, it is quite possible for a malicious actor to alter software or configuration of the DDS-based system which could cause severe damage or loss of life. It is therefore imperative that methods of detecting both malware and malicious activity within a DDS-based system be developed. Knowledge of how DDS might be compromised to support malicious activity on a DDS-based system is the first step in being able to defend against a cyberattack on an ATC system or any other DDS-based critical infrastructure system.

1.4 Research Aim

This research aims to investigate DDS security issues and develop and validate DDS threat models that could be used by a malicious actor to compromise a host via a client-side attack on a DDS-based system. Investigation will include analysis of the DDS standard and vendor implementations to find potential security issues. Threat models will be developed based on discovered DDS security issues that have been demonstrated in a self-contained environment,

such that they can be combined into an end-to-end validation scenario that demonstrates their effectiveness in compromising a DDS-based system.

1.5 Thesis Outline

This thesis document details DDS security research and analysis, including self-contained demonstration and validation of several DDS security issues. Chapter 2 describes the DDS middleware technology, including a review of current DDS security research. Chapter 3 contains a methodical analysis of DDS security issues from the DDS specification [1, 2] and several vendor implementations. Chapter 3 also details how a small number of the DDS security issues discovered in the analysis were modelled and demonstrated in a self-contained and isolated environment, in order to compromise features of DDS. Chapter 4 describes how an end-to-end validation scenario made up of a combination of the previously modelled and demonstrated security issues could be used in a client-side attack to compromise hosts on a DDS-based system. Chapter 5 discusses the implications of this research and future work that may be done.

Chapter 2 Literature Review

2.1 Introduction

The previous chapter described the motivation for analyzing DDS. This chapter provides an overview of DDS, which is required to understand this research work. This chapter includes a description of how DDS works, different DDS vendor implementations, a broad overview of security concerns with DDS middleware technology and a review of current DDS security research.

2.2 Data Distribution in Real-Time Systems

Since their inception, computer systems have successfully been used for data collection and distribution, decision-making and control. Computer control of machinery based on collection and distribution of real-time data has allowed computers to reliably perform complex tasks such as: SCADA (Supervisory Control and Data Acquisition) for power and machinery control systems, military C4ISR (Command, Control Communications and Computer Intelligence, Surveillance and Reconnaissance), and Air Traffic Control (ATC).

2.2.1 Middleware and Distributed, Real-Time, Mission Critical Computing

Traditional software systems approaches to building distributed computing environments for data distribution and control have included centralized and client-server architectures, however these architectures are fast becoming limited as distributed computer systems evolve. Key issues that have been encountered include the fact that hardware and networking protocols are becoming more heterogeneous, and the fact that distributed computer systems are more dynamic and spontaneous in nature, requiring loosely coupled and asynchronous infrastructures. Both issues are not well handled by traditional client-server architectures [8].

Real-time mission-critical systems can consist of up to many thousands of networked devices (nodes) that can include platforms, sensors, machinery actuators, weapons systems and decision nodes. Together these communicate with each other in order to enable collaborative decision making, and effect changes in other devices, with physical effects (e.g. light bulb, machinery control, weapons deployment), or logical effects (e.g. creation or modification of information) [4].

Real-time, mission-critical systems require that middleware such as those based on the DDS standard support the following:

- **Common real-time view of data:** the ability to view and share the current view of the data with all nodes, including users and other connected systems in real-time;
- **Data integrity and delivery:** enforcement of QoS (Quality of Service) requirements to ensure that the right data gets to the right place at the right time [4];
- **Heterogeneous environment support:** support for integration of equipment of different age, design philosophy, and protocol. Portability to different environments and interoperability with legacy equipment is a common requirement; and
- **Dynamic reconfiguration:** components of the network constantly change as users (data producers and consumers) join and leave various network domains. This includes dynamically formed or modified collections of nodes that need to share a common view of the data and need the ability to exchange data seamlessly upon reconfiguration or loss of any node.

Object Management Group (OMG)'s Data Distribution Service for Real-Time Systems (DDS) middleware standard has become a very popular technology used in distributed real-time, data-centric systems. 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. More importantly, DDS includes QoS (Quality of Service) standards that ensure the integrity and timely delivery of data, which is critical for information management, surveillance and control systems.

The DDS standard has therefore become an enabling technology [4] for implementation of the following applications:

- SCADA and Industrial Machinery control systems [9, 18];
- Banking and financial trading systems [19];
- Internet of Things (IoT) and Home Automation systems [20, 21];

- Distributed data gathering, surveillance and intelligent decision making systems including traffic data collection [22] and vehicle-to-vehicle control systems [23];
- Commercial and military ATC systems;
- Satellite control systems [24];
- Military C2 systems within the context of a single platform (ship, aircraft or field unit) [16]; and
- Multi-unit, theatre-level C4ISR systems supporting "data-centric" warfare [12, 16].

2.2.2 DDS vs. Legacy Architectures

Mission-critical real-time middleware has traditionally followed a client-server architecture, but it has evolved into using message-passing architectures and publish-subscribe architectures.

Client-server architectures access data on remote objects using function calls and methods which are very specific to the remote object's software and data structure. Client-server architectures work well when the data sources are specified and centralized and the network configuration is relatively static. Client-server architectures generally use connection-oriented protocols and are not dynamically scalable. Examples of client-server middleware include Common Object Broker Architecture (CORBA), Distributed Common Object Model (DCOM) and Enterprise JavaBeans (EJB).

Message-passing architectures extend the client-server model by simplifying the exchange of information between nodes on the network by using queues of messages. Message-passing architectures still require connection-oriented protocols, rarely specify standard QoS characteristics (i.e. QoS policies are static and hard-coded between applications exchanging messages), and are therefore limited by the messages that are passed and the methods that handle them. Examples of message-passing middleware include the CORBA Event Service and Notification Service and the Java Message Service (JMS).

Publish-subscribe architectures differ from client-server and message passing architectures in that communication is based on the data that is transmitted, rather than the source and destination of the data. The message becomes a generic container for the actual information published. Message passing requirements therefore consist of a "Topic" designation (to which a Subscriber will subscribe), and Quality of Service (QoS) parameters that designate generic requirements for

its delivery. Publish-subscribe architectures are not constrained to have a connection-oriented transport protocol, as data publication and subscription can both be anonymous - any requirement for retransmission is handled by the middleware based on the QoS parameters. RTI Connext DDS™ and OpenDDS are examples of publish-subscribe architectures, and are based on OMG's DDS standard [17].

As a connectionless, "data-centric", publish-subscribe middleware architecture standard, DDS extends the traditional publish-subscribe architecture by increasing the QoS parameters available to distributed system developers.

2.2.3 Data-Centric vs. Message Centric

DDS middleware has become popular as it is based on a publish-subscribe model where the data is more important than the network topology or the devices used. This allows an "Information-centric" or "Data-centric" paradigm that offers "the right information at the right place at the right time" [8].

"Data-centric" communications allow the distributed architecture and distributed applications to be designed based on the data type that has a specified QoS (i.e. qualities like dependability, survivability, scalability, security and confidentiality). A "Data-centric" architecture allows communication that is custom tailored to the specific requirements of a distributed system, without having to specify a specific network architecture or messaging scheme [3].

This is in contrast to client-server and other "Message-centric" architectures that dictate the message types, formats, protocols and require a specific, defined network topology. Although some client-server and "Message-centric" implementations have had DDS-like architectural and QoS capabilities, none have included all features required by real-time, mission critical distributed systems, and none have eased implementation as the communications architecture cannot be isolated from the designer, as is the case with DDS [4].

"Data-centric" communications standards such as DDS remove the burden of how to communicate with the other applications in the environment. A data publisher does not need to know about whom or where the data is going to; they only need to know about the specific data type that is being communicated. Likewise, a data subscriber does not need to know about how or where data is coming from; they only need to know about the specific data type they want to receive [3].

2.3 DDS Technology

DDS was created by OMG as a specification of a publish-subscribe middleware for distributed systems. The DDS specification was designed to allow efficient distribution of data in a distributed system, whereby participants share information in a global data space in which participants can read and write. Data access is controlled through the use of Topics to which participants publish or subscribe. DDS shields developers and applications from the inner workings of the communication mechanism – applications only need to specify the data used in the communication (i.e. the Topic).

DDS is ideal for implementation of mission-critical, real-time systems as it supports the following capabilities:

- Seamless access to information from any DDS-enabled source running over potentially heterogeneous hardware and software platforms and networks;
- Abstraction of details of the particular transport layer used (i.e. information delivery integrity, aggregation, filtering and prioritization);
- Ability to continually adapt to changes in network topologies, changes to publisher and subscriber membership and ability to handle intermittent connectivity; and
- QoS policies and mechanisms to allow applications to specify communication constraints appropriate to the application, without specifying the underlying network architecture.

2.3.1 DDS Architecture

The OMG DDS standard is composed of the DDSI (DDS Interoperability Wire Protocol) layer, which specifies the layer above the network transport and the DDS layer, which specifies Data-Centric Publish-subscribe (DCPS) behaviour and the DDS API [8]. Figure 1 shows a stack view of the OMG DDS standard architecture [9].

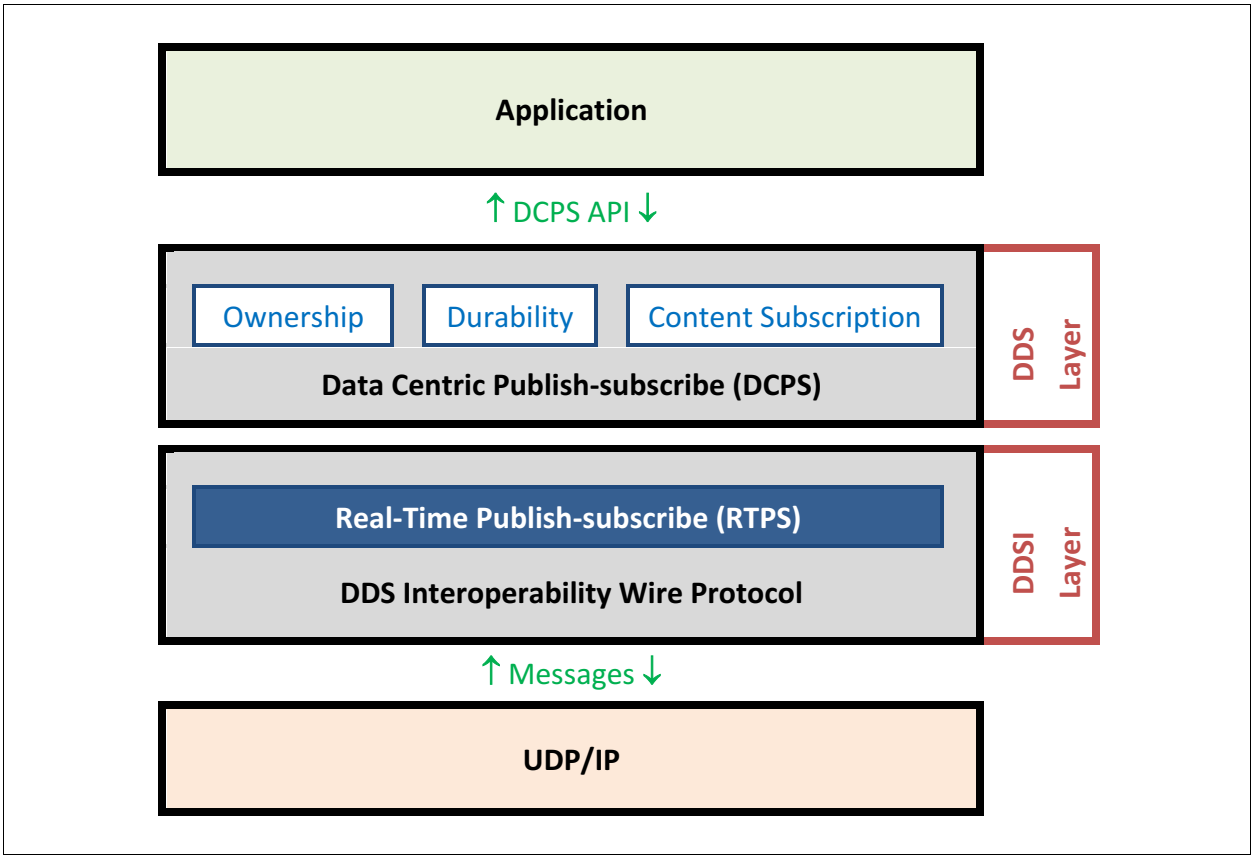

Figure 1: The OMG DDS standard architecture – stack view (reproduced from [9]).

The following sections detail the major portions of the DDS architecture that pertain to this research work.

2.3.1.1 Data-Centric Publish and Subscribe (DCPS)

The core of DDS is built around DCPS [12], which defines how applications communicate with each other. Applications communicate over a Global Data Space, where they publish information indicated by a Topic or subscribe to information indicated by a Topic. DCPS handles the data distribution and enforces QoS restrictions and policies, thereby freeing the applications from performing these tasks.

The following DCPS entities, as shown in Figure 2, are used in a DDS client application:

- **Global Data Space**: is the top-most abstraction of DDS, as it stores all the data published in the domain. Because the Global Data Space can be fully distributed, the system can be scalable and avoid single point of failure;

- **Domain:** is a virtual data space (within the Global Data Space), in which DDS client applications send and receive data. The Domain supports communication between Domain Participants, and it encapsulates all data and communications in the Domain from other Domains;
- **Domain Participant:** is the DDS Entity that contains one or more DataReaders and DataWriters, that allows communication over a Domain;
- **DataWriter:** prepares data (from the client application) to be sent to a Publisher on the Domain. A DataWriter prepares data (format and type) based on a single Topic specification;
- **Publisher:** disseminates (publishes) data onto the Domain, as prepared by one or more DataWriters (i.e. one per Topic);
- **Subscriber:** receives published data on the Domain, and makes the data available to the particular DataReader associated with the subscribed Topic;
- **DataReader**: receives data from the Subscriber based on a particular Topic, and makes it available to the client application. Within the DataReader, DDS supports two notification mechanisms to the client application [13]: *Listener-based notifications* which notify the client application of a relevant asynchronous event such as data arrival or QoS setting violation (this is the default); and *Condition-based notifications* which notify the client application when a specific condition occurs such as a status of an entity or the arrival of data; and
- **Topic**: is the data specification (i.e. unique key, a data type and a set of QoS parameters) that unambiguously links a DataWriter (via Publisher) with a DataReader (via Subscriber), over the Domain. Communication can only occur if the DataReader subscribes to the same Topic that the DataWriter has published. Note that communication only requires that the Topic be the same, however it is anonymous and transparent [3]. In other words, Publishers and Subscribers have no visibility over who creates or consumes the data. The data type specified in the Topic is usually defined by an OMG Interface Definition Language (IDL) structure developed for the particular application.

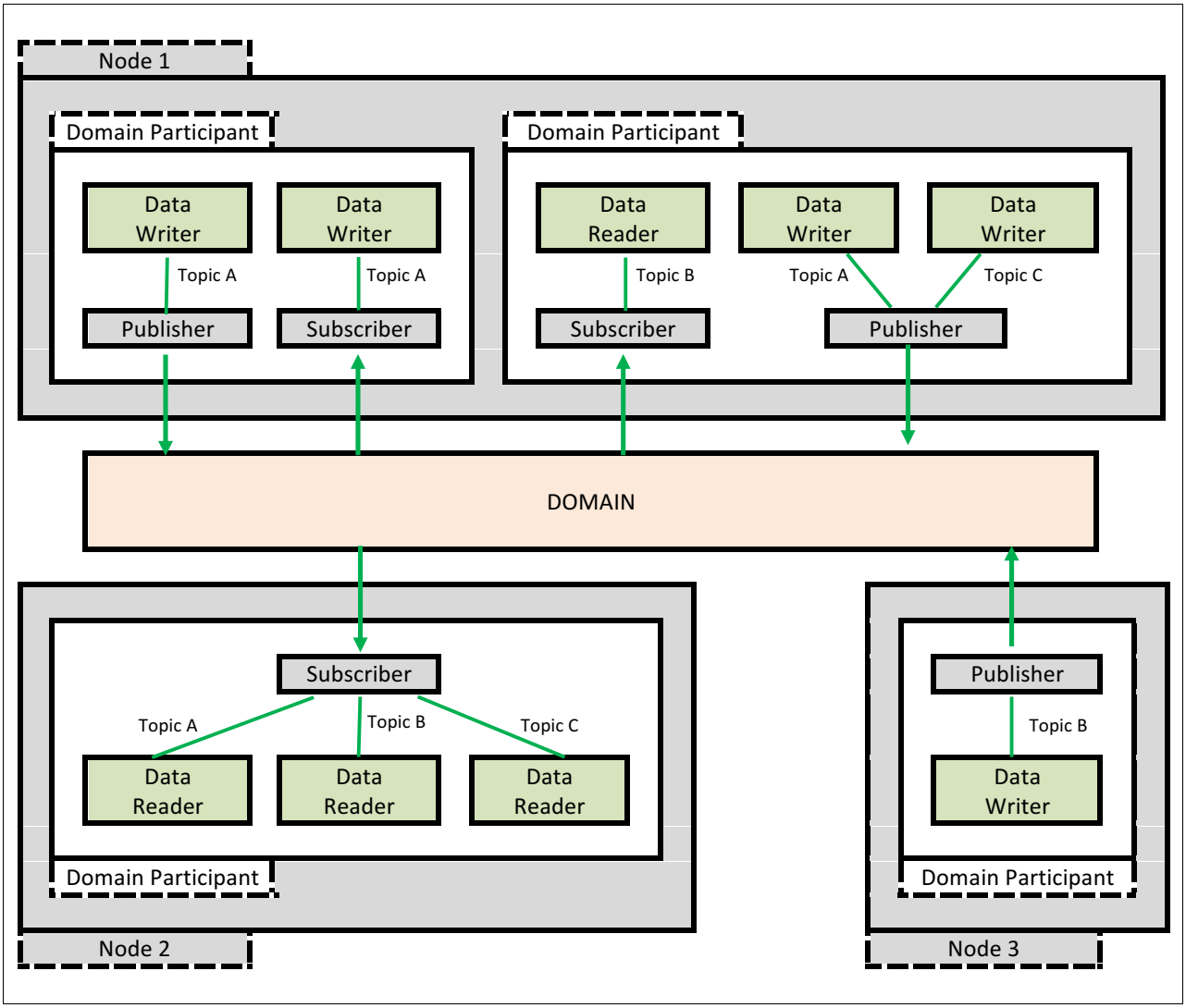

Figure 2: DDS Entities and communication flow (reproduced from [14]).

The DDS DCPS specification also defines QoS policies that indicate minimum service levels required for communication between DCPS entities, and constraints on the data communicated between DCPS entities [15]. 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; in other words, whether it is more important to be timely than complete (or vice-versa);
- **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 whether the data is to be retained if communication is broken or intermittent; and
- **Resource Limits:** specifies which DDS resources the service can use to meet other QoS requirements and what constraints are placed on those resources.

Domain Participants specify the QoS policies required for the various DCPS entities (Topic, DataWriter, DataReader, Publisher, Subscriber, or Domain Participant); the DDS vendor implementation will decide how to achieve these behaviors (i.e. whether or not RTPS or another transport is to be used). Publisher and Subscriber QoS policies can be set through an "offer-request" collaboration mechanism that works as follows.

- A Publisher offers a set of QoS policies to all subscribers;
- A Subscriber requests the set of QoS policies that it requires; and
- DCPS then attempts to match the requested policies with the offered policies (within the restrictions of the particular DDS vendor implementation). Communication between the Publisher and Subscriber only occurs when the QoS policies are consistent.

2.3.1.2 Real-Time Publish and Subscribe Protocol (RTPS)

The Real-Time Publish and Subscribe Protocol (RTPS) specifies a wire protocol (i.e. a protocol for services that exist at or above the transport layer) for DDS. It can be used on top of multicasting and best-effort transport layer protocols such as UDP/IP, although connection-oriented protocols can still be used (e.g. TCP/IP)

RTPS has been tailored to support the real-time publish-subscribe requirements needed by DCPS by including timing parameters and properties. As such, RTPS mainly consists of two communication protocols: the Publish-Subscribe (PS) protocol which is used for transferring

data and the Composite State Transfer (CST) protocol, which is used for communicating state information.

The main features which make RTPS very suitable for a DDS wire protocol include [9]:

- **Performance and QoS properties:** enable and enforce best-effort and reliable publish-subscribe communications in (near) real-time applications;
- **Fault tolerance:** since there is no requirement for a broker (as in CORBA), and since data is replicated across DDS Entities in the Global Data Space, DDS-based systems can be made more reliable and avoid single points of failure; and
- **Dynamic membership:** DDS Entities can be added and removed at any time due to the connectionless model upon which RTPS is based – RTPS will automatically adjust to changing memberships.

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.

2.3.2 DDS Implementations

The DDS specification does not specify a transport model and it handles reliable and ordered delivery of messages itself. Therefore, DDS can be adapted to different network transport models. Several commercial and open-source DDS middleware vendor implementations have been developed, which support multiple network-protocols for transparent forwarding of DDS-data and fault tolerant communication paths between nodes. Major implementations of DDS are listed below.

2.3.2.1 RTI Connext DDS™

As one of the two developers of the original DDS middleware software (the other being Thales Nederland), the American software company Real-Time Innovations (RTI) developed their own

implementation of DDS. Now called RTI Connext DDS™, RTI's implementation is a fully DDS-compliant implementation [16], with the following implementation nuances:

- It uses a completely decentralized architecture, where no single node contains an essential DDS service or function;
- Communication and configuration capabilities reside in same process as the client application itself on each node; and
- The middleware itself is made up of the software library, the distributed database and the client tasks that exist on each node.

RTI Connext DDS™ is used in many mission critical applications such as ATC, SCADA, machinery control, military C4ISR applications and naval C2 systems.

2.3.2.2 OpenDDS

The open-source software organization Object Computing Incorporated (OCI), has been developing a fully open-source version of DDS called OpenDDS. It uses a simpler but potentially less robust architecture which may be preferable in some applications:

- It supports a centralized architecture where key DDS functionality including the master data store can be configured to reside on a single node; and
- When configured to use the centralized architecture, this single information repository runs on a designated node which then controls all configuration for DDS participants on a single domain [17].

OpenDDS is developed and maintained by many volunteers in C++ and it continues to evolve. However, it has not enjoyed the popularity and has not had as much use in large real-time mission critical systems as the other commercial DDS vendor implementations [16].

2.4 Security in DDS

DDS was designed to provide a developer maximum flexibility when building their DDS-based system. This flexibility was built into DDS at the expense of internal resiliency and security, and little effort has been made to secure DDS from any manipulation of the software or configuration on the host. In other words, proper DDS functionality could be affected if the host running the

DDS software is compromised. Protection of the host (which provides a *trusted zone* of execution) relies on physical security measures and network isolation from other systems and outside users. Protection of the data handled by DDS-based systems has been the main focus of current DDS security efforts [6], which are described in the following sections.

2.4.1 Current State of DDS Security

The OMG DDS standard continues to evolve, and currently DDS 1.2 is the latest and most commonly used version in deployed products [1]. A new DDS security standard has been developed to augment the current DDS standard and potentially be incorporated into the next version of DDS [5]. The DDS Security Standard 1.0 is described in section 2.4.2.3.

2.4.1.1 Existing DDS Security Advantages

Even without the DDS Security Standard 1.0, DDS contains many features that lend itself to use within a secure environment [5].

As DDS does not require a broker to control message distribution and the DDS vendor implementation can use peer-to-peer or multicast messaging. This removes the potential of a vulnerable single broker being attacked allowing communication to continue between uninfected nodes.

In addition, since DDS does not specify an underlying transport mechanism, the DDS vendor implementation can use multicasting to send a single packet to any number of users that have expressed interest through subscription, reducing the possibility of data loss due to an infected machine. This also increases system robustness by being able to dynamically add another (backup) client node to handle multicast packets.

The DDS standard states that a central data store is not required and the DDS vendor implementations often implement this via a distributed data store. This reduces the potential of data corruption or system unavailability due to an infected server.

These features, along with DDS's automatic node discovery mechanism, allow a DDS-based system to continue operation if network segments become isolated, or if a single node needs to be replaced. When reconnected, a DDS-based system automatically rediscovers the reconnected node or network segment and continues to function.

The DDS standard includes many QoS policies that when applied, form custom "contracts" between Publishers and Subscribers in the Domain. Some key QoS polices pertaining to network security are:

- Data Lifetime;
- Frequency of Data Updates;
- Maximum Data Delivery Latency;
- Data Delivery Priority;
- Data Delivery Reliability (including attempts to retransmit); and
- Duration of Data Validity.

Since these QoS policies form the "contract" between Publishers and Subscribers, any violation can cause the communication to fail and trigger a security violation - this could be used as criteria for an Anomaly-based Intrusion Detection System (A-IDS).

2.4.1.2 Existing DDS Security Disadvantages

Without the DDS Security Standard 1.0, DDS, categories of threats impacting DDS-based systems are as follows [5]:

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

At stake is the confidentiality and integrity of the data within the Global Data Space and specific Domains. Specific shortcomings of DDS (without DDS Security Standard 1.0) are described in the following paragraphs.

Authentication of Domain Participants is required to prevent unauthorized access to the Domain (or Global Data Space), as any application could start publishing any information masquerading as another Publisher. In addition, *Authorization* of DataWriters and DataReaders

(as part of the Domain Participant) is required to perform certain actions (e.g. defining a new Topic, reading or writing a specific Topic).

DDS supports anonymous publish-subscribe communication, however it does not support information partitioning based on security classifications, which is a key requirement to secure many systems [24]. Attempting to use different Topics to enforce information partitioning based on security classifications does not fulfill this requirement for the following reasons: it places the burden on a Subscriber to only subscribe to Topics that are appropriate to their security level; and it places the burden on a Publisher to not publish data with a higher security classification to a Topic that can be read by a Subscriber with a lower security classification [24].

LAN-based DDS systems (including those in a single platform such as a tank or aircraft) may have *some* protection against attack due to strict physical protection to network infrastructure that the platform affords. In this case, data is still vulnerable because there is still possibility of data access from inside and encryption is not mandatory in DDS [12].

A key design feature of DDS is that communications can be established automatically upon node discovery [25]. In addition, automatic physical node and endpoint discovery can be done as per the RTPS wire protocol. If this feature is enabled without adequate security credential checks, it could present a significant security risk. In other words, an attacker could use this feature to join the Domain automatically and access or manipulate data.

2.4.2 DDS Security Enhancements and Research and Development

Since the DDS Security Standard 1.0 has only been recently finalized and published [5], several interim extensions to the middleware have been made, based on the existing DDS standards. These interim extensions, followed by the new DDS Security Standard 1.0 are described in the following sections.

2.4.2.1 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 [24]. 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. It uses Multi-Level Security (MLS) labels to represent security classifications as a set of hierarchical classification levels and a set of non-hierarchical need-to-know categories.

The new discovery service authorizes and establishes ST information flows between matching Publishers and Subscribers based on their security labels. The discovery service is based on a MLS policy and communication schema, and only privileged entities of the system can interact with it.

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.

2.4.2.2 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 [12].

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.

This approach is not practical to implement, as it uses a new, more complex and non-standard discovery method that is different from SPDP and SEDP discovery protocols built-in to DDS. Use of this approach would require all network participants to upgrade to this new framework, which would involve significant additional development effort by application developers.

2.4.2.3 DDS Security Standard 1.0

OMG's DDS Security Standard 1.0 defines Security Model (SM) and Service Plugin Interface (SPI) architecture extensions for DDS. The new standard emphasizes the use of new built-in SPI plug-ins that enable out-of-the box security features. RTI has already developed a new DDS vendor implementation based on this standard (see section 2.4.2.4) [5, 6].

The DDS Security Model contains five built-in SPI plug-ins used for Information Assurance (IA), which are as follows:

- **Authentication Service Plugin:** provides identity verification for a DDS Entity including mutual authentication (i.e. shared secret) between Domain Participants;
- **Access Control Service Plugin:** provides security policy enforcement for allowable operations on an authenticated DDS Entity (e.g. Domain membership changes, Topic publish-subscribe access, etc.);
- **Cryptographic Service Plugin:** provides cryptographic operations including encryption, decryption, hashing, digital signatures, message integrity checking, and protected key distribution;
- **Logging Service Plugin:** provides a method to log all DDS security-relevant events (i.e. audit trail), and to distribute them as required; and
- **Data Tagging Service Plugin:** provides a method to specify security metadata (e.g. classification level, etc.), in published data and a means to determine the access privileges required to subscribe to the data.

When incorporated into DDS vendor implementations, these new plugins address many of the security shortcomings in current DDS vendor implementations and may supersede the DDS security enhancements outlined in the previous sections.

While the OMG DDS Security Standard 1.0 addresses many security shortcomings of DDS, it still requires significant developer effort to upgrade existing DDS-based systems. Many of its security enhancements are optional and it supports backwards-compatibility with non-secure versions of DDS, but the data is only as secure as the security of the weakest DDS participant. In

addition, security loopholes could still exist within the mechanisms used to support backwards compatibility.

2.4.2.4 RTI Connext DDS Secure™

The American software company Real-Time Innovations (RTI) has developed its own implementation of a secure DDS based on the OMG DDS Security Standard 1.0. Marketed as "RTI Connext DDS Secure™", it is advertised to be "the world's first standards-compliant, off-the-shelf messaging platform that delivers the security, performance and safety required for deployment of the Industrial Internet of Things" [6]. Its standard capabilities include the five SPI plug-ins specified by the OMG standard (described in section 2.4.2.3). These capabilities are supported by X.509 Public Key Infrastructure (PKI) with a pre-configured shared Certificate Authority (CA); Digital Signature Algorithm (DSA) with Diffie-Hellman and RSA for authentication and key exchange; and AES128 and AES256 for encryption [6].

This approach shares the same shortcomings as the OMG DDS Security Standard 1.0 (described in section 2.4.2.3).

2.4.3 DDS Security Research and Development Shortcomings

All DDS security improvements described above have only focused on protection of the data handled by DDS-based systems [6]. All versions of DDS, including those based on DDS Security Standard 1.0, remain vulnerable to attacks on the hosts (i.e. the host provides a *trusted zone* of execution for the DDS libraries, configuration files and application software). Protection of the host relies on physical security measures and network isolation from other systems and outside users. These measures are never perfect and it is posited that a host compromise is a likely method of attack, and that the host would be a likely location for DDS-based malware.

Analysis of how DDS might be compromised would be very beneficial. This could include analysis of the DDS standards (DDS 1.4 and RTPS 1.2) and analysis of the different vendor implementations. In addition, analysis of the effects of a compromised DDS-based system would be extremely valuable. Analysis would lead to improved methods of detecting DDS malware artifacts and improved methods of detecting DDS malware execution.

2.5 Summary

This chapter provided an overview of DDS, including a description of how DDS works, different DDS vendor implementations, a broad overview of security concerns with DDS middleware technology and a review of current DDS security research. This chapter also described that further work in DDS security is needed to detect and counter threats to DDS-based systems, as malicious behavior is likely to originate from the DDS hosts. This information provides the needed background to understand the analysis of DDS security issues and how they could be used to compromise DDS-based system, as described in the chapters that follow.

Chapter 3 Identification and Characterization of Potential DDS Security Issues

3.1 Introduction

The previous chapter provided the background required to understand DDS security issues, including descriptions of the DDS standard, different DDS vendor implementations, a broad overview of DDS security concerns and a review of current DDS security research. This chapter describes the analysis of DDS security issues as well as the detailed investigation and self-contained demonstration of five threat models based on DDS security issues. This chapter describes the scope of the analysis, the process in which the analysis is done and the selection criteria used to identify DDS security issues for analysis. This includes a custom DDS taxonomy developed to categorize and analyze the identified DDS issues. This chapter also details the process in which each of the five DDS security issues was modelled and demonstrated in a self-contained and isolated environment to demonstrate how each one can compromise a DDS-based system.

3.2 Terminology

Terminology used in the remainder of the document is based on definitions in the Internet Engineering Task Force (IETF) Glossary contained in Request For Change 4949 (RFC 4949) [26]. For clarity, specific terms that have been tailored to reflect the focus of this research (DDS security), are described in Appendix A.

3.3 Selection and Identification of Areas of DDS to Study

DDS is commonly used in Industrial Control, SCADA, and Command and Control (C2) systems, which tend to be deployed in physically isolated, access-controlled environments and are often isolated from other networks and systems [4]. The operational environment in which DDS-based systems are deployed determines *what types of security issues* will be examined. The DDS technology and vendor implementations determine *what features of DDS* will be examined.

The following sections describe the criteria used to identify and select DDS issues for analysis. These include operational security issues in DDS-based systems, features of the DDS standard, and DDS vendor implementations.

3.3.1 Operational Security

DDS-based systems tend to be used in physically isolated, access-controlled environments that are isolated from other networks and systems [4], and therefore it is posited that an attack originating outside of the security perimeter of a DDS-based system would be unlikely. It is also posited that the main threat would likely be a host compromise developed by a person who has a somewhat detailed knowledge of the system, its use, its architecture and its implementation (i.e. a disgruntled worker, insider, organized crime, or a state-sponsored actor). In other words, the main threat would be by a host compromise via a myriad of client-side attack techniques developed by a malicious actor. For the purposes of this research, it is assumed that access to the DDS-based system has already been gained via access to either the physical system, the software development environment, or the supply chain. The client-side malware can tamper with the application that uses DDS or the DDS libraries, and upload malicious payloads to the DDS-based system. This thesis does not examine techniques required to perform those tasks.

Therefore, this research focusses on the actions of an attacker once they have a foothold and persistence is gained on the target system. In other words, this research focusses on how an attacker could exploit security issues *inside* a DDS-based system – not the methods of infection.

3.3.2 DDS Architecture, Features and Software Security

This research focusses on how a host compromise might affect the functionality of the DDS-based system through tampering of the application software, DDS libraries or DDS configuration. Techniques by which this might be achieved were explored by analyzing the DDS software and architecture. Analysis was done via investigation of the DDS specification documentation [1, 2] and DDS vendor implementation documentation and software [27, 28].

The DDS standard and DDS vendor implementations were analyzed by exploring the functionality and data structures of DDS to see how it interacts with surrounding elements of a DDS-based system. Figure 3 below shows a DDS “stack” on one DDS host, which contains DDS layers and the surrounding elements (the elements in Red indicate potential attack surfaces).

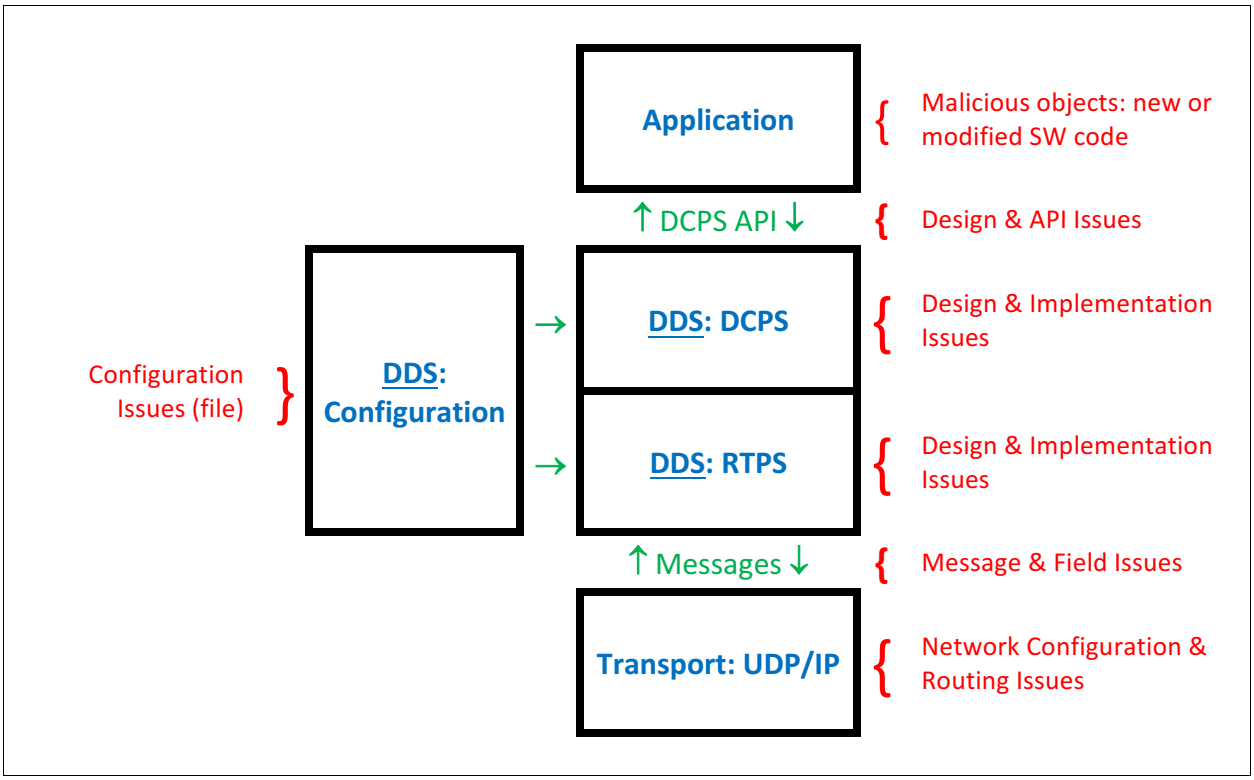

Figure 3: Example DDS Stack and locations of potential security issues (in red).

The DDS Specification [1, 2] was analyzed to determine design flaws in DCPS which could be abused. Analysis of the API provided by DCPS to the host application was done to reveal methods and effects of malicious use of (or inadvertent misuse of) the DCPS API. This included examination of the vendor-implemented debugging API built into DCPS by two DDS vendors (OpenDDS [27] and RTI Connext DDS™ [28]). In addition, vendor-specific implementation flaws of DCPS were investigated to determine other methods of exploitation and their potential effects.

A similar analysis of RTPS was done, but it did not include any API analysis (i.e. RTPS does not provide an external API). Additional analysis of RTPS messages was done, along with a detailed examination of the DDS Entity discovery protocols and mechanisms (i.e. SPDP and SEDP).

Analysis of transport mechanisms used in DDS was done, with a focus on design and vendor-implementation flaws when multiple transports are used. Note that DDS supports multiple, concurrent transport configurations using the same transport mechanism or different transport mechanisms.

RTPS is an optional part of DDS, but it was specifically designed to match DCPS requirements and it was designed to operate on best-effort protocols like UDP/IP. Since RTPS is used in almost all DDS-based systems, the analysis was done exclusively with RTPS instead of other wire protocols.

The analysis examined vendor implementation mechanisms used to configure DDS. Configuration mechanisms included methods of specifying configuration parameters at build-time (i.e. before the application or DDS libraries are compiled and executed), and methods of specifying configuration parameters at run-time (i.e. via configuration files that get read at application startup or dynamically during application execution).

3.4 Characterization and Analysis of DDS Security Issues

The focus of the analysis was to determine how DDS could be misconfigured or manipulated to support malicious activity within a DDS-based system. The previous section identified the operational context in which to examine DDS security issues, as well as the features of DDS to analyze (i.e. DCPS, RTPS, DDS Transport mechanisms and DDS Discovery mechanisms). This section describes the analysis process, the custom DDS taxonomy (used to guide the analysis and ensure coverage of each DDS feature), and the analysis results.

3.4.1 Analysis Process

The analysis was limited to the DDS areas of study identified in the previous section, however the focus of the analysis was on the potential security issues, attack types, attack methods and effects. To ensure thorough and focused analysis of each feature, a custom DDS Analysis Taxonomy was developed, which is described in section 3.4.2.

The analysis examined the following DDS artifacts:

- DDS specification documentation [1, 2];
- DDS vendor implementation documentation [27, 28];
- DDS vendor implementation source code (OpenDDS only);
- DDS vendor implementation sample configuration files and templates;
- DDS example application source code; and

- RTPS messages generated by execution of DDS example applications.

The analysis process was largely ad-hoc, however it generally followed the methodology described below. Analysis of each artifact consisted of a detailed study of the documentation artifacts, which were organized by DDS features and functionality. Where appropriate to the feature or functionality being analyzed, sample application source code and configuration files were analyzed to improve the understanding of the subject. Whenever possible, execution of sample DDS applications was done in order to analyze DDS behaviour and produce RTPS messages that could be analyzed in Wireshark. In addition, whenever available, previous analysis results and insights were revisited to confirm any new findings.

Analysis of vendor implementations focused on the behaviour of the vendor product. This behaviour was determined via analysis of the vendor design documentation and verified via the observation of sample DDS applications execution. Analysis did not include any examination of vendor-provided binary files for the purpose of determining security issues at the machine-code-level, nor did it include any discovery of security issues that may be exposed during execution via fuzzing techniques.

Note that the analysis process deliberately used two different vendor implementations to ensure broad coverage of potential DDS security issues. Certain types of security issues are much more easily exploited using OpenDDS than RTI Connext DDS™ by virtue of the fact that the OpenDDS source code is available. This could allow a malicious developer to rewrite DCPS or RTPS libraries for easy insertion into a target system. In addition, the configuration methods are different between the two DDS vendor implementations.

3.4.2 DDS Analysis Taxonomy

The analysis used a new DDS Malware Taxonomy developed during this research work, to ensure a focused and thorough analysis of security issues in each DDS feature. This new Malware Taxonomy was based on the work of Hansman and Hunt [29], Applegate and Stavrou [30], and the ENISA and OWASP web pages [31, 32]. The new taxonomy is summarized in Table 1, and is described in the paragraphs that follow.

Table 1: DDS Malware Taxonomy Summary

Taxonomy Dimension		Implication to DDS Analysis
1	**Attack Origin**	This analysis focused on client-side attacks on a DDS-based system.
2	**Attack Target(s)**	This analysis fixed the target to DDS components and host applications employing DDS.
3	**DDS Flaws**	Included Design, Implementation and Configuration flaws that might be exploited as part of an attack.
4	**Attack Types**	Included Denial of Service (DoS) attacks, network attacks, repudiation attacks and data attacks.
5	**Attack Methods**	Included exploitation of hardware and software availability, alteration of DDS software or configuration files or existing deficiencies.
6	**Informational Impacts**	Included the effects of an attack on data handled by DDS (e.g. Information Denial / Destruction / Disclosure / Discovery / Distortion).
7	**Detection**	This analysis only briefly explored methods of detecting DDS malware and DDS misuse to provide completeness in the analysis (i.e. detection was out of scope).

The **1st dimension** covers the main **means by which the attack first reaches the target** (i.e. how and where an attack source originates). The analysis focuses on client-side attacks on a DDS-based system therefore this *analysis does not focus on this dimension* (it is included for completeness).

The **2nd dimension** covers the **target(s) of the attack**. In our analysis, the target is fixed to the DDS (DCPS and RTPS) as well as to host application, configuration mechanisms, transports and RTPS messages as indicated in section 3.3.2.

The **3rd dimension** covers design, implementation and configuration **flaws** that result in a security issue that an attacker might exploit as part of an attack.

The **4th dimension** covers **attack types** as determined by the primary effect of the security issue. These include Denial of Service (DoS), network attacks (including spoofing, hijacking, routing traps), repudiation attacks and data attacks. Note that the attack types are not orthogonal; they can be combined to be used in an attack scenario.

The **5th dimension** covers **attack methods** used by an attack (i.e. how the attack is carried out). This analysis explores the specific *methods of exploitation* including: hardware and software availability, alteration of software (DDS-based application, DCPS libraries or RTPS libraries), alteration of configuration files and existing deficiencies.

The **6th dimension** covers attacks that have **information impacts (effects and consequences)** beyond the primary effect. Since DDS is data-centric, the data and informational impact of the attack on the target DDS-based system is categorized as follows (adapted from Applegate and Stavrou's Information Impact [30]): Denial of Information, Destruction of Information, Disclosure of Information, Discovery of Information and Distortion of Information.

The **7th dimension** covers indications that could be used in the **detection** of DDS malware and DDS misuse. Detection of DDS malware and misuse is out of scope of the analysis; however, the analysis briefly explores methods of detection of DDS malware and DDS misuse. This is done to provide completeness in the analysis of each DDS feature and provides quick sanity-check of the validity of each DDS security issue.

Further details of the DDS Malware Taxonomy are described in Appendix B. The DDS Security Analysis Template, based on the DDS Malware Taxonomy is also included in Appendix B, section B.10. This template was used to aid the analysis of each DDS security issue identified in section 3.3.2.

3.4.3 DDS Analysis Results

Analysis of DDS security issues resulted in the discovery of 60 separate DDS security issues in four key areas: DCPS, RTPS, Transport mechanisms and Configuration mechanisms. Detailed analysis of the 60 DDS security issues focused on the 3rd, 4th, 5th and 6th dimension of the DDS Malware Taxonomy described in section 3.4.2. The overall analysis results are listed in Appendix C. Analysis of each of the 60 DDS security issues was recorded by filling out a DDS Security Analysis Template (included in Appendix B, section B.10) for each issue. This analysis is recorded in [36].

Within the scope of this research, only five of these DDS security issues were further examined using self-contained demonstration environments and an end-to-end validation scenario. This further examination will be done in the sections that follow.

3.5 Modelling and Self-contained Demonstration of DDS Security Issues

The analysis of DDS security issues in section 3.4 revealed 60 DDS security issues in four key areas: DCPS, RTPS, Transport mechanisms and Configuration mechanisms. Five of the 60 identified DDS security issues were selected for modelling and self-contained demonstration, where the goal was to demonstrate its malicious effects in an isolated and controlled environment. Combination of the five DDS security issues (that had been modelled and demonstrated in self-contained environment), into an end-to-end DDS attack scenario was done in an end-to-end validation scenario (which is detailed in section 4.2).

3.5.1 Selection DDS Security Issues for Analysis, Modelling and Self-contained Demonstration

Many of the 60 DDS security issues identified during the analysis shared common DDS Flaws, Attack Types and Exploitation Techniques (i.e. the 3rd, 4th and 5th dimensions of the DDS Malware Taxonomy respectively). The analysis also covered a very broad range of DDS features and functionality including vendor implementation-specific DDS features and nuances. Selection of DDS security issues for modelling and self-contained demonstration ensured coverage of a broad set of attack types (as indicated in the 4th dimension of the DDS Malware Taxonomy). These attack types included examples of DoS, Hijacking and Network configuration alteration; as well as Data Deletion, Modification, Insertion, Redirection and Omission. In addition, selected DDS security issues covered a broad set of DDS functionality, including: DCPS functionality (i.e. publish-subscribe and QoS functionality); RTPS functionality (including messages); Discovery mechanisms; and vendor implementation-specific features (including configuration methods). Appendix E details how the selected DDS Issues provide coverage of attack types and DDS features.

Detailed analysis was performed for each of the five DDS security issues that was selected. The analysis results for each DDS security issue included filled-out DDS Security Analysis forms, which are listed in Appendix F.

3.5.2 DDS Self-Contained Demonstration Environment

Self-contained demonstration of each DDS security issue used the DDS Shapes Demo application from RTI Connext DDS™, running on several Lubuntu Virtual Machines (VMs) in a virtual network using Parallels on a MacBook (software and operating system versions are

detailed in Appendix D). A fully virtualized environment aided in the configuration and refinement of the self-contained demonstrations.

The DDS Shapes Demo application was developed by DDS vendors to test interoperability between different DDS implementations, and also to demonstrate how different QoS policies affect data flow within a DDS-based system. The Shapes Demo provides simple methods of specifying unique data types as shapes (squares, triangles and circles) with different qualities, such as colour, size and movement rate. The Shapes Demo can run as both publisher and subscriber, and multiple instantiations of the Shapes Demo allow it to simulate different DDS Entities on different hosts. The Shapes Demo provides a user interface that graphically displays the shapes to which the DDS Entity subscribes and publishes, including graphical configuration of QoS policy settings. This allows quick and easy of validation of behaviour. The following subsections detail the configuration and normal operation of a DDS-based system using the Shapes Demo application, which will be used as the initial state for each of the self-contained demonstrations detailed in the sections that follow.

3.5.2.1 DDS Self-contained Demonstration Environment: Initial State Overview

Demonstration of DDS security issues relied on the following QoS policies: Ownership, Ownership Strength, Reliability and Lifetime (see section 2.3.1.1 for details). The target DDS-based system consisted of four DDS Entities, each one consisting of a Shapes Demo application running on its own host as follows:

- **Publisher #1**: published Blue Squares and Blue Triangles with `EXCLUSIVE` ownership;
- **Subscriber #1**: subscribed to Blue Squares and Blue Triangles with `EXCLUSIVE` ownership;
- **Publisher #2**: published: Green Triangles with `SHARED` ownership; and
- **Subscriber #2**: published: Green Triangles with `SHARED` ownership.

Figure 4 illustrates the initial data distribution state of the target DDS-based system.

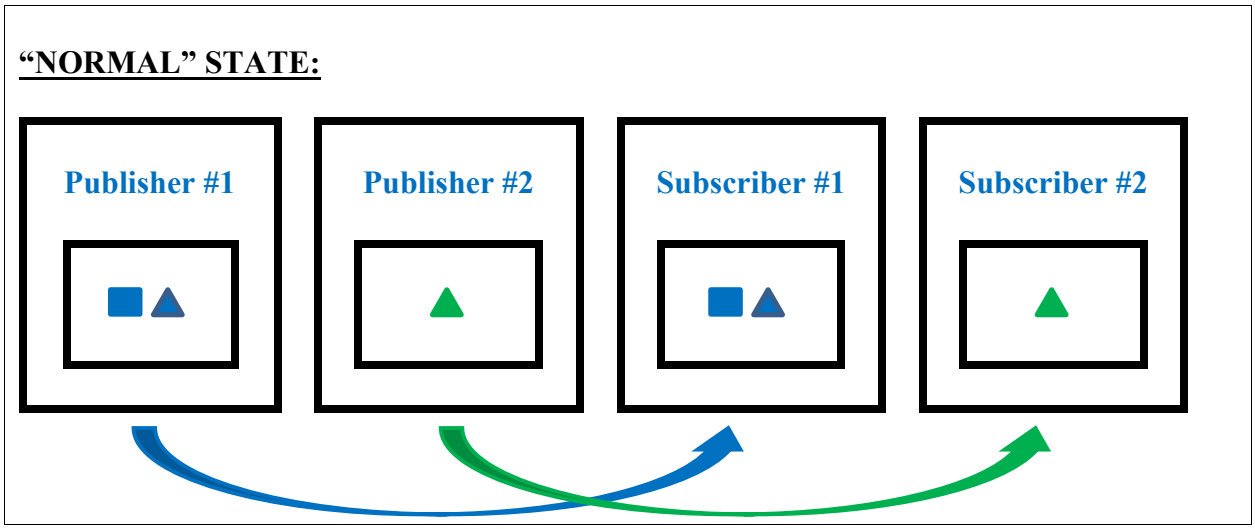

Figure 4: Self-contained Demonstration Environment: Initial state of DDS Entities and communication flow.

3.5.2.2 DDS Self-contained Demonstration Environment: Initial State Architectural View

Figure 5 illustrates the architectural view of the target DDS-based system in its initial state. It shows a DDS stack on two 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.

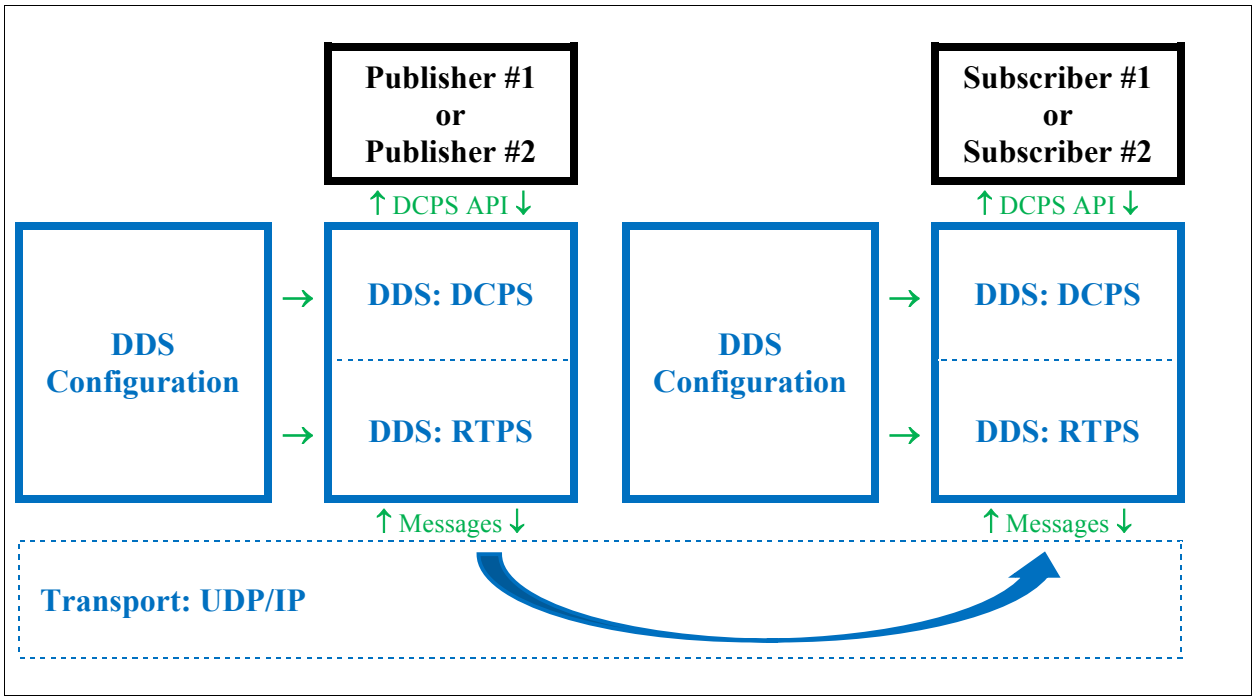

Figure 5: Self-contained Demonstration Environment: Initial state architectural stack view.

3.5.3 DDS Security Issue #1: Misuse of Anonymous Subscribe and Republish Functionality

This DDS security issue involved the anonymous nature of the publish-subscribe mechanism that is central to DDS, in order to subscribe to valid data and to republish multiple, altered copies of the original data. Within the context of the self-contained demonstration environment, this involved adding a new DDS Entity that would subscribe to a specific shape and republish multiple, altered copies of the shape. The original shape and the republished shapes would all be received by the original subscriber. Details of this self-contained demonstration are described in the sections that follow.

3.5.3.1 DDS Security Issue #1: Self-contained Demonstration Overview

The goal of this self-contained demonstration was to prove that a new malicious DDS Entity could be used to subscribe to existing data in the system and republish multiple, altered copies of the data. In other words, this new malicious DDS Entity could be used to generate misleading and confusing data based on valid data. If the DDS Entity that subscribes to this data displays the results to a human operator, the additional data could confuse or overload the operator, especially if the new data was subtly altered to make it indiscernible from the original data.

Two DDS Entities in the Shapes Demo test environment were used to demonstrate this DDS security issue. Figure 6 below illustrates the initial state of the relevant DDS Entities prior to demonstration of the DDS security issue (green arrow lines indicate original data flow).

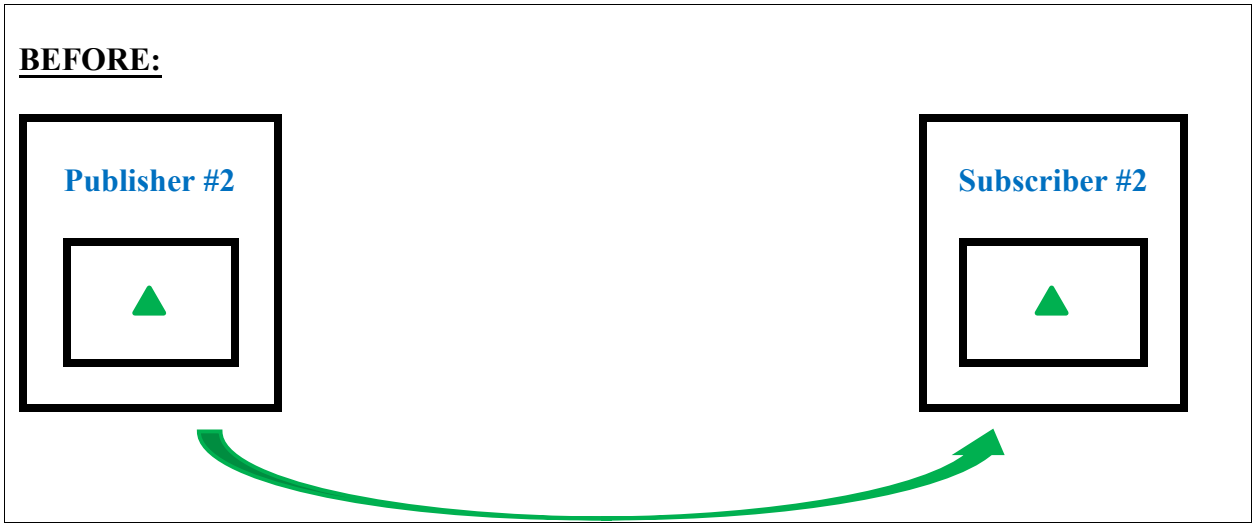

Figure 6: Self-contained Demonstration #1: Initial state of DDS Entities and communication flow.

To demonstrate this DDS security issue, a new malicious DDS Entity was developed, created and executed on another host on the network. This new application combined functionality of a subscriber, a data multiplier and a publisher (illustrated as Publisher #3 and Subscriber #3). This self-contained demonstration involved subscription to instances of the Green Triangle Topic, and published multiple copies of each instance of the Green Triangles back on the network. The published copies of the original were slightly altered in position, such that they would be discernable from the original, as seen by a subscriber (Subscriber #2 in this case). This is shown in Figure 7 below (green line indicates original data flow; malicious entity and malicious data flow is shown in red).

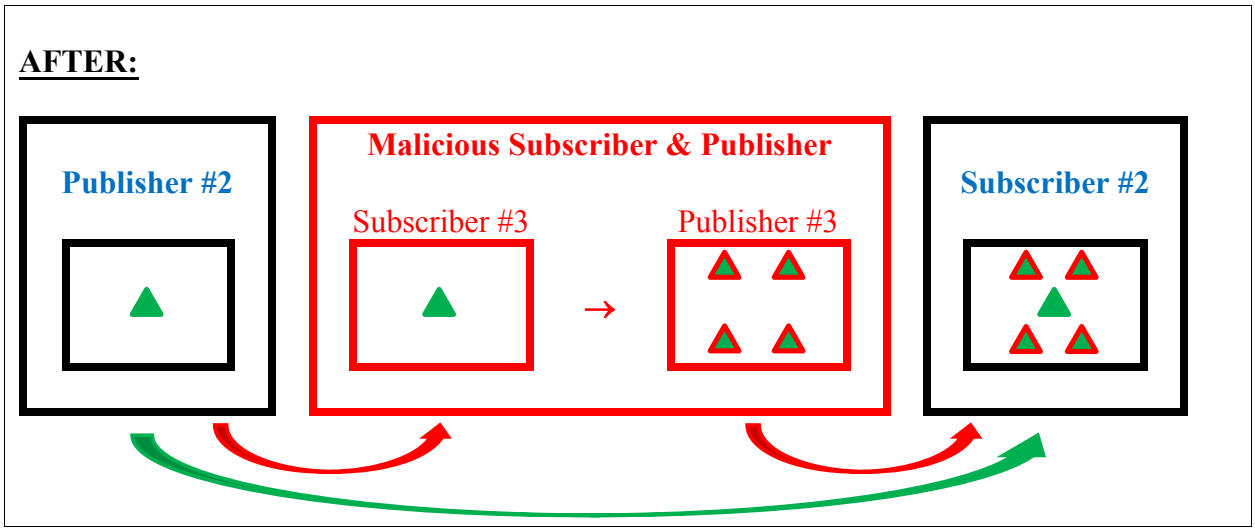

Figure 7: Self-contained Demonstration #1: Post-exploit state of DDS Entities (Data Multiplication).

3.5.3.2 DDS Security Issue #1: Architecture and Implementation

Self-contained demonstration of this DDS security issue involved exploitation of the anonymous nature of the publish-subscribe mechanism that is central to DDS. 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 only needs to know the data type and the QoS policies used in the communication flow. In the DDS self-contained demonstration environment, this was easily discernable via examination of RTPS packets that contained valid Shapes data using the Wireshark analysis tool. Note that Shapes Demo source code was not available, but if it was available this task would have been easier.

A new malicious application containing subscriber and publisher DDS Entities 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 contained logic that would multiply the original data and alter the position of the copied data such that they could be observed by the user demonstrating the DDS security issue.

Figure 8 below illustrates the architectural view of the DDS Entities involved in the self-contained demonstration of this DDS security issue (green line indicates original data flow; malicious entity and malicious data flow is shown in red). DDS Entities included unaltered DDS Entities (Publisher #2 and Subscriber #2), and new malicious DDS Entities (Subscriber #3 and Publisher #3).

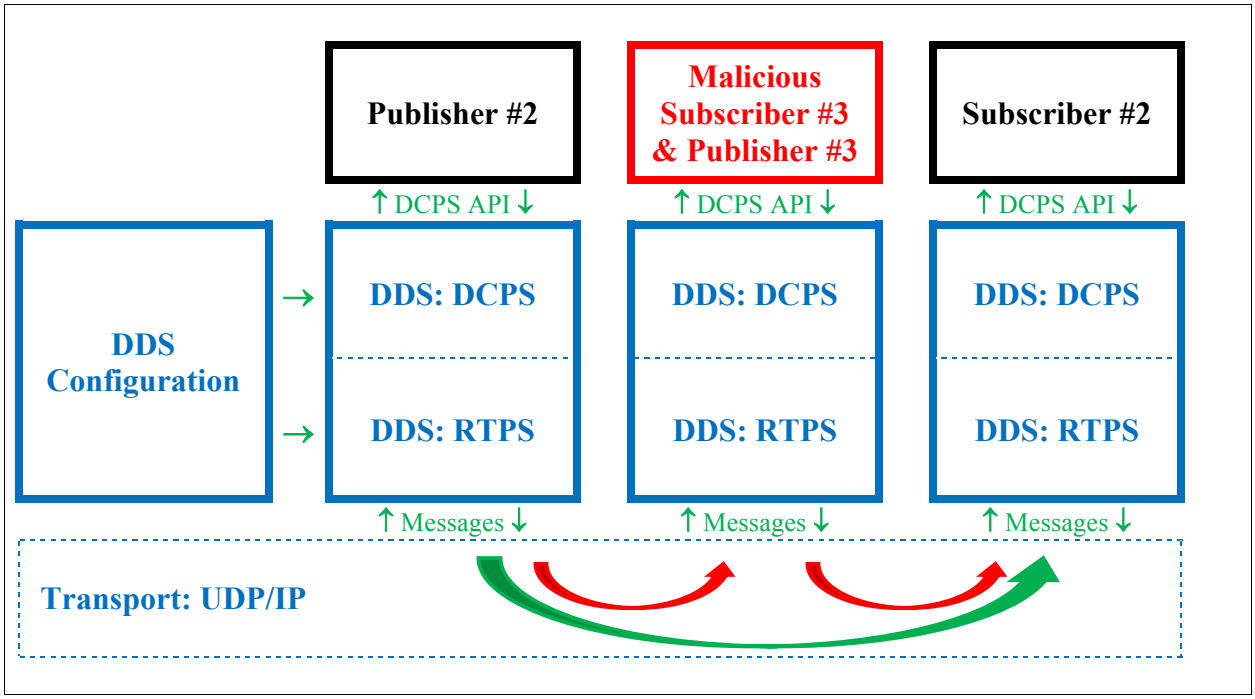

Figure 8: Self-contained Demonstration #1: Architecture stack view (Data Multiplication).

3.5.3.3 DDS Security Issue #1: Results and Discussion

Execution of this self-contained demonstration successfully showed that this DDS security issue can be used in a data multiplication attack (data modification and data insertion). The new malicious application successfully subscribed to Green Triangle shapes, 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 was not able to distinguish which one of the five was the original Green Triangle.

3.5.4 DDS Security Issue #2: Misuse of OWNERSHIP_STRENGTH QoS Policy and EXCLUSIVE Data Ownership

This DDS security issue involved the use of 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 self-contained demonstration involved launching a new malicious publisher that has a higher OWNERSHIP_STRENGTH than the existing publisher, causing the existing publisher to cease publishing data and the new malicious publisher to publish new data. In essence, the malicious publisher hijacked data publication for this Topic. The QoS Policy was changed via modification of the application's QoS policies configuration via insertion of a generic DDS configuration file which is higher in the QoS Policy configuration precedence. Details of this self-contained demonstration are described in the sections that follow.

3.5.4.1 DDS Security Issue #2: Self-contained Demonstration Overview

The goal of this self-contained demonstration was to prove that a new malicious DDS publisher can hijack data publication from an existing publisher. Subscribers would receive data from the new malicious publisher and no longer receive the data from the original publisher, without any indication of change of communication flow. This change in configuration could cause issues for all subscribers to this data Topic, especially if the subscribing application displays the results to a human operator – the operator might not notice that the data is invalid.

Three DDS Entities in the Shapes Demo test environment were used to demonstrate this DDS security issue. Publisher #1 published Blue Squares with OWNERSHIP_KIND set to EXCLUSIVE ownership, and Subscriber #1 subscribed only to Squares with OWNERSHIP_KIND set to EXCLUSIVE ownership. Figure 9 below illustrates the initial state of the relevant Entities prior to demonstration of the DDS security issue (blue arrow lines indicate original data flow).

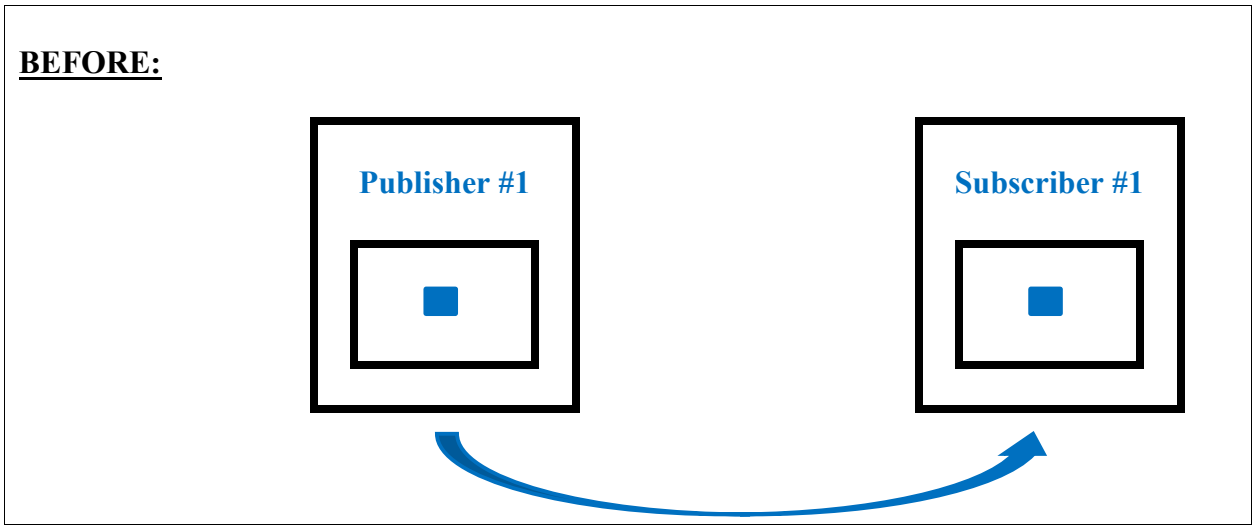

Figure 9: Self-contained Demonstration #2: Initial state of DDS Entities and communication flow.

To demonstrate this DDS security issue, Publisher #1 published Blue Squares with an `OWNERSHIP_STRENGTH` of `9`, and a new malicious Publisher #4 was configured to publish Blue Squares with an `OWNERSHIP_STRENGTH` of `10`. Once new malicious Publisher #4 joined the DDS network, without restarting existing DDS Entities, Subscriber #1 only received Blue Squares from malicious Publisher #4. This is shown in Figure 10 below (malicious entity and malicious data flow is shown in red).

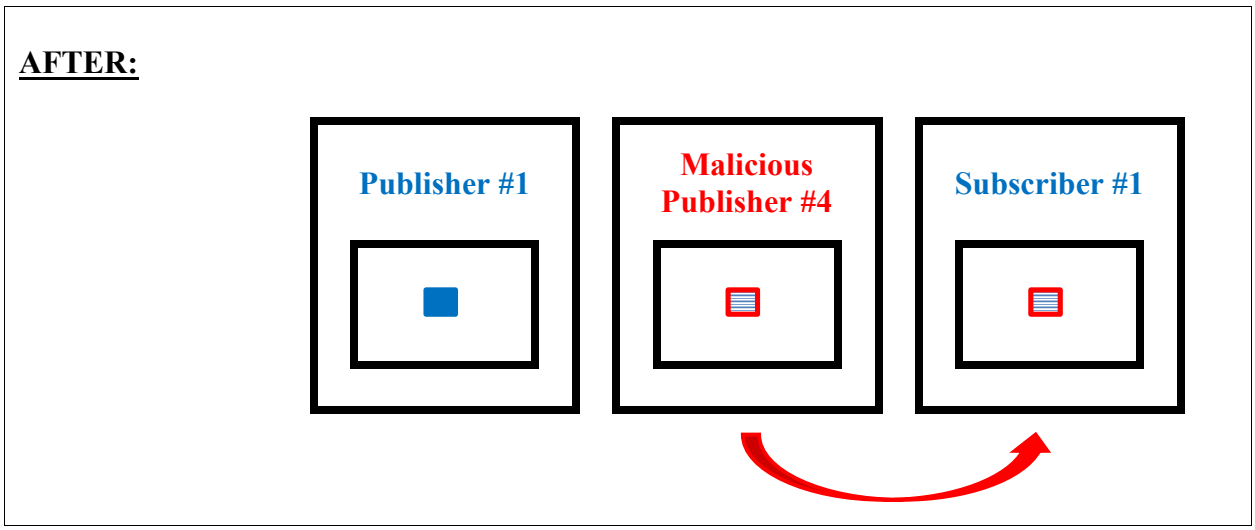

Figure 10: Self-contained Demonstration #2: Post-exploit state of DDS Entities (Data Hijacking).

3.5.4.2 DDS Security Issue #2: Architecture and Implementation

Self-contained demonstration of this DDS security issue involved exploitation of built-in DDS functionality that allows prioritization of unique data that comes 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, and another sensor updates less often 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 was required was 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 was currently used. In the DDS self-contained demonstration environment, this was easily discernable via examination of Wireshark packet dumps of valid Shapes data. Note that Shapes Demo source code was not available, but if it was available this task would have been easier.

A new Shapes Demo DDS Entity, Publisher #4 was 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 contained an additional "cross-hatch" `Fill` attribute that did not affect subscription but it allowed easier observation by the user demonstrating the DDS security issue.

Figure 11 below illustrates the architectural view of the DDS Entities involved in self-contained demonstration of this DDS security issue (blue line indicates original data flow; malicious entity and malicious replacement data flow is shown in red). DDS Entities included unaltered DDS Entities (Publisher #1 and Subscriber #1), and the new malicious DDS Entity (Publisher #4).

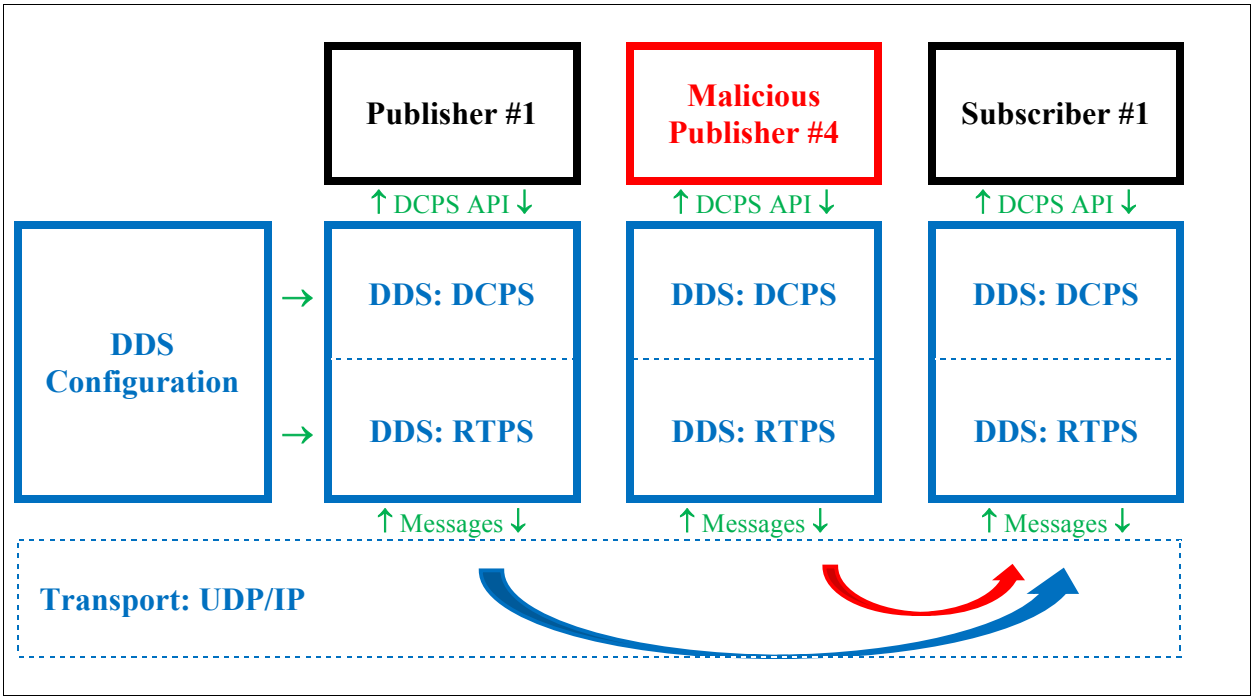

Figure 11: Self-contained Demonstration #2: Architecture stack view (Data Hijacking).

3.5.4.3 DDS Security Issue #2: Results and Discussion

Execution of this self-contained demonstration successfully showed that this DDS security issue can be used in a data hijacking attack (i.e. a combination of a data omission attack and a data insertion attack). The new malicious Shapes Demo DDS Entity successfully hijacked data publication of Blue Squares from Publisher #1. Subscriber #1 only received Blue Squares from the malicious Publisher #4 which were indistinguishable from valid Blue Squares except that the "fill" colour of the shape was cross-hatched (for ease of observability during testing). There was no indication at Subscriber #1 that any change in data distribution state had occurred.

3.5.5 DDS Security Issue #3: Misuse of `OWNERSHIP_KIND` QoS Policy and `SHARED` Data Ownership

This DDS security issue 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. This self-contained demonstration involved changing the `OWNERSHIP_KIND` of a publisher, such that data was no longer received by the intended subscriber – instead it appeared on another subscriber. The QoS Policy was changed via insertion of a generic DDS configuration file which is higher in

precedence (according to the RTI configuration mechanism). Details of this self-contained demonstration are described in the sections that follow.

3.5.5.1 DDS Security Issue #3: Self-contained Demonstration Overview

The goal of this self-contained demonstration was to prove that 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 of 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 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.

Four DDS Entities in the Shapes Demo test environment were used to demonstrate this DDS security issue. Publisher #1 published Blue Triangles with `OWNERSHIP_KIND` set to `EXCLUSIVE` ownership and Subscriber #1 only subscribed to Triangles with `OWNERSHIP_KIND` set to `EXCLUSIVE` ownership. Publisher #2 published Green Triangles with `OWNERSHIP_KIND` set to `SHARED` ownership and Subscriber #2 only subscribed to Triangles with `OWNERSHIP_KIND` set to `SHARED` ownership. Figure 12 below illustrates the initial state of the relevant DDS Entities prior to demonstration of the DDS security issue (blue and green arrow lines indicate original data flow).

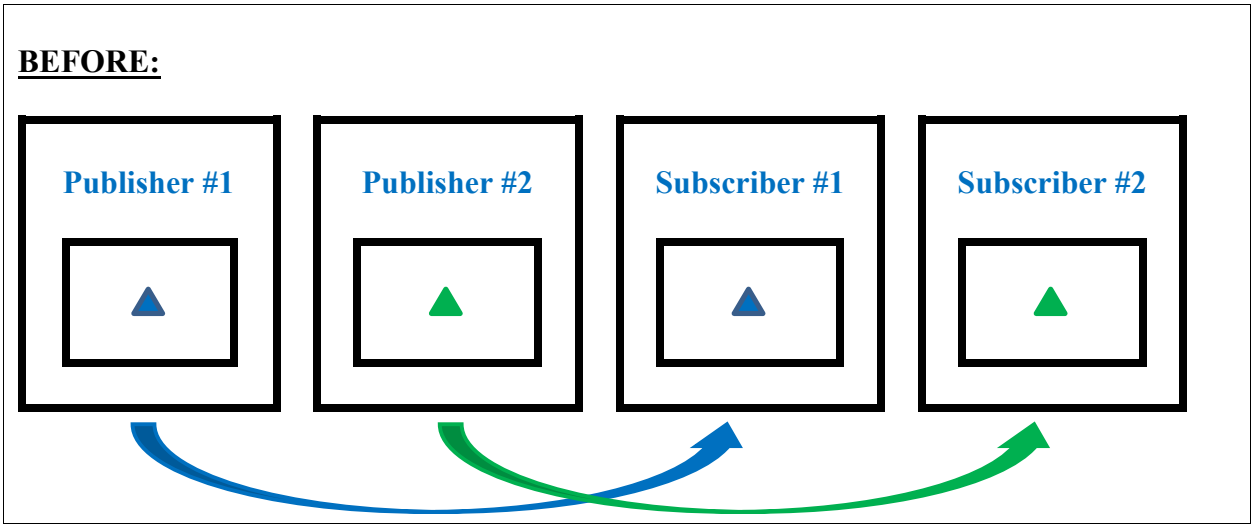

Figure 12: Self-contained Demonstration #3: Initial state of DDS Entities and communication flow.

To demonstrate this DDS security issue, the target publisher's (Publisher #1), configuration was altered such that the `OWNERSHIP_KIND` of the Topic (Blue Triangles) that it published was changed from `EXCLUSIVE` ownership to `SHARED` ownership. Once the DDS application on the Publisher #1 host was restarted, the original subscriber (Subscriber #1) no longer received any Triangles, as it only subscribed to Triangles with `EXCLUSIVE` ownership. Instead, Subscriber #2 received the Blue Triangles from Publisher #1, in addition to the Green Triangles previously received from Publisher #2 (i.e. Subscriber #2 subscribed to Triangles with `SHARED` ownership, regardless of colour). This is shown in Figure 13 below (green line indicates original data flow; malicious entity and malicious data flow is shown in red).

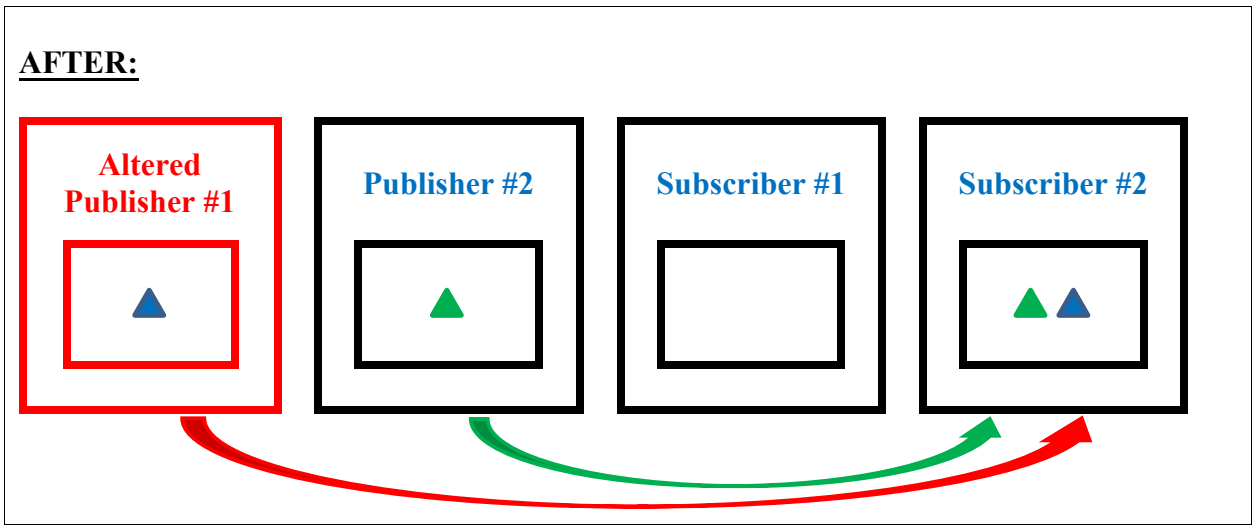

Figure 13: Self-contained Demonstration #3: Post-exploit state of DDS Entities (Data Redirection).

3.5.5.2 DDS Security Issue #3: Architecture and Implementation

Self-contained demonstration of this DDS security issue involved the alteration of the `OWNERSHIP_KIND` QoS Policy for the Topic data on one DDS Entity to change how data is distributed. The `OWNERSHIP_KIND` of a Topic can be either `SHARED` ownership or `EXCLUSIVE` ownership, and it must match on both the publisher and subscriber on order for communication to occur. When the `OWNERSHIP_KIND` is set to `SHARED`, all DDS Entities can publish and subscribe to the data. When the `OWNERSHIP_KIND` is 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 self-contained demonstration involved changing the `OWNERSHIP_KIND` for the Triangle Topic on Publisher #1 from `EXCLUSIVE` to `SHARED`. This had the effect of rerouting data away from Subscriber #1, whose `OWNERSHIP_KIND` for the Triangle Topic was set to `EXCLUSIVE`. Data was then sent to Subscriber #2, whose `OWNERSHIP_KIND` for the Triangle Topic was set to `SHARED`.

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

Figure 14 below illustrates the architectural view of the DDS Entities involved in self-contained demonstration of this DDS security issue (blue line indicates original data flow; malicious entity and malicious data flow is shown in red). 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 was altered – the executable remained unaltered.

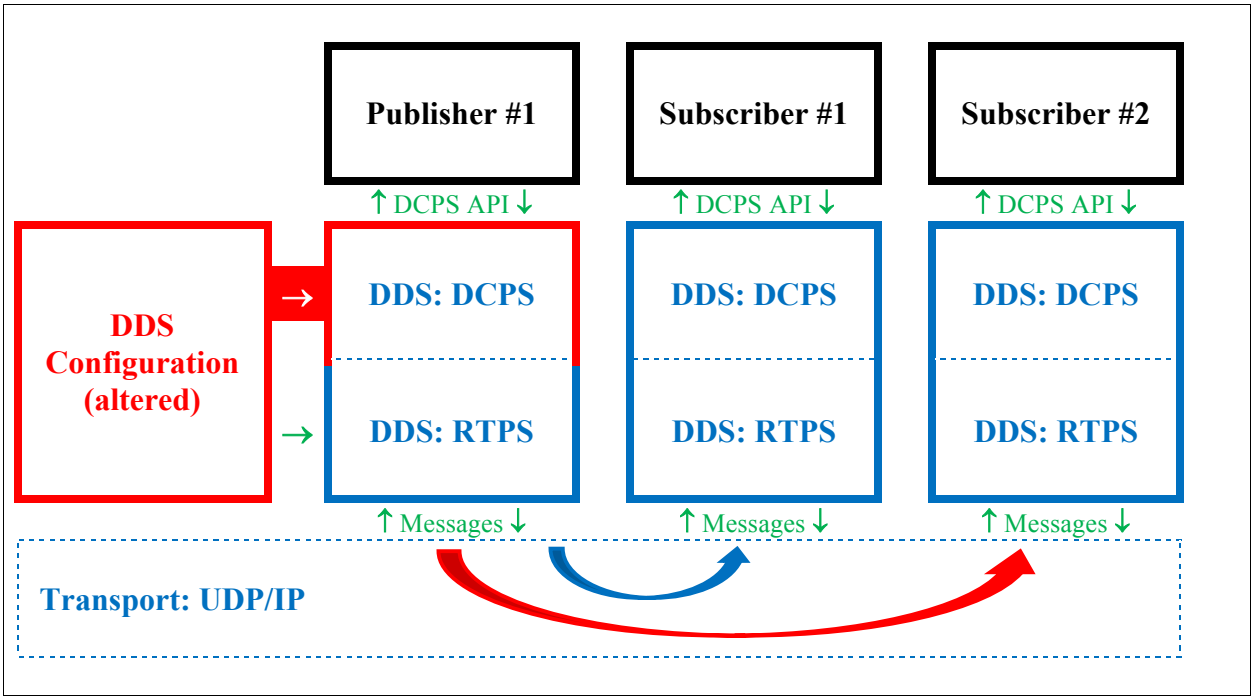

Figure 14: Self-contained Demonstration #3: Architecture stack view (Data Redirection)

3.5.5.3 DDS Security Issue #3: Results and Discussion

Execution of this self-contained demonstration successfully showed that this DDS security issue can be used in a data redirection attack. The altered configuration of Publisher #1 resulted in Subscriber #1 no longer receiving any data (Blue Triangles) from Publisher #1. Instead, Subscriber #2 received the data (Blue Triangles) that were originally intended for Subscriber #1. There was no indication at either Subscriber #1 or Subscriber #2 that any change in data distribution state had occurred.

Using the Shapes Demo application required that one configuration file be altered and one configuration file be added on Publisher #1. This was due to two reasons: one, that the Shapes Demo application hard-coded a specifically-named QoS configuration file, and two, this specific configuration file contained syntax specifying that default QoS parameters be taken from configuration files deep in the protected DDS library file structure. This was not encountered in any other DDS application during this research work, which involved examination of many demonstration and sample DDS applications provided with OpenDDS and RTI Connext DDS™.

The most commonly encountered method of configuring QoS policies was the use of the "`USER_QOS_PROFILES.xml`" file. This method uses built-in DDS mechanisms which looks for the presence of this file in the same directory in which the DDS application was launched

(highest precedence), or in the directory where the executable resides (lowest precedence). This method was also used for the Shapes Demo application, but it required the alteration of the hard-coded configuration file specific to the Shapes Demo application (as described in the previous paragraph).

3.5.6 DDS Security Issue #4: Misuse of `LIFESPAN` QoS Policy Causing Immediate Data Expiration

This DDS security issue 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. This self-contained demonstration involved changing the `LIFESPAN` of a data Topic at the publisher, such that the data would expire prior to be displayed at the subscribing application. The QoS Policy was changed via insertion of a generic DDS configuration file which is higher in precedence (according to the RTI configuration mechanism). Details of this self-contained demonstration are described in the sections that follow.

3.5.6.1 DDS Security Issue #4: Self-contained Demonstration Overview

The goal of this self-contained demonstration was to prove that 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.

Three DDS Entities in the Shapes Demo test environment were used to demonstrate this DDS security issue. Publisher #1 published Blue Triangles, Publisher #2 published Green Triangles and Subscriber #2 subscribed to Triangles (`OWNERSHIP_KIND` was set to `SHARED` ownership for all DDS Entities). Figure 15 below illustrates the initial state of the relevant Entities prior to demonstration of the DDS security issue (blue and green arrow lines indicate original data flow).

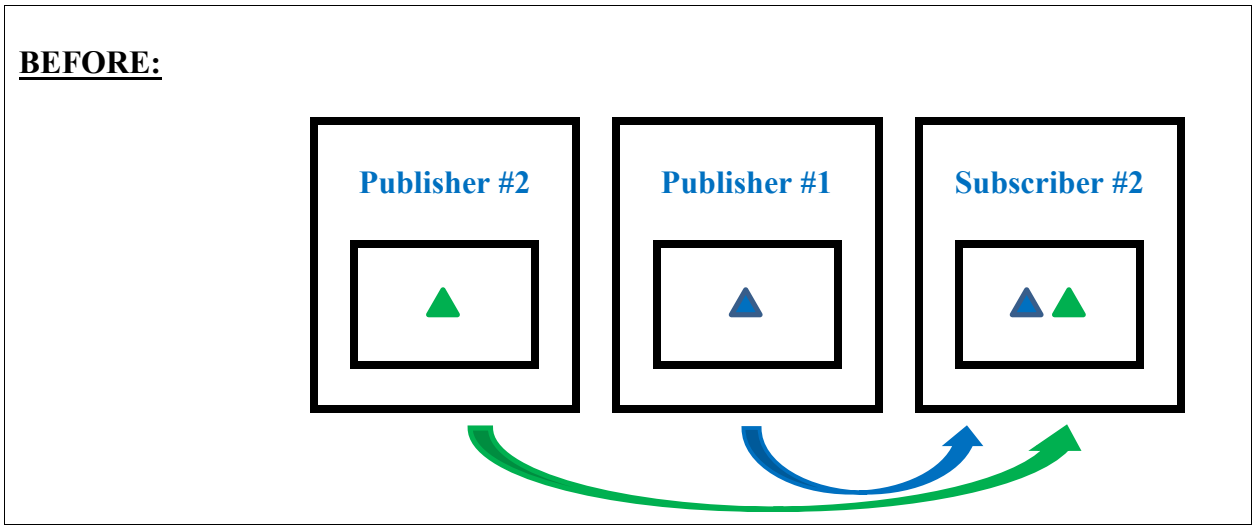

Figure 15: Self-contained Demonstration #4: Initial state of DDS Entities and communication flow.

To demonstrate this DDS security issue, the target publisher's (Publisher #2), configuration was altered such that the `LIFESPAN` of the Topic (Green Triangles) was reduced to a value which caused Green Triangles to disappear from Subscriber #2. Once the configuration of Publisher #2 was altered and the Publisher #2 DDS application on that host was restarted, Subscriber #2 no longer received any Green Triangles. The disappearance of Green Triangles was dependent on both the `LIFESPAN` value and the processor load, as certain `LIFESPAN` values caused intermittent display of Green Triangles (see discussion in subsection 3.5.6.3 below). This is shown in Figure 16 below (blue line indicates original data flow; malicious entity is shown in red; and dotted green line indicates intermittent, altered data flow).

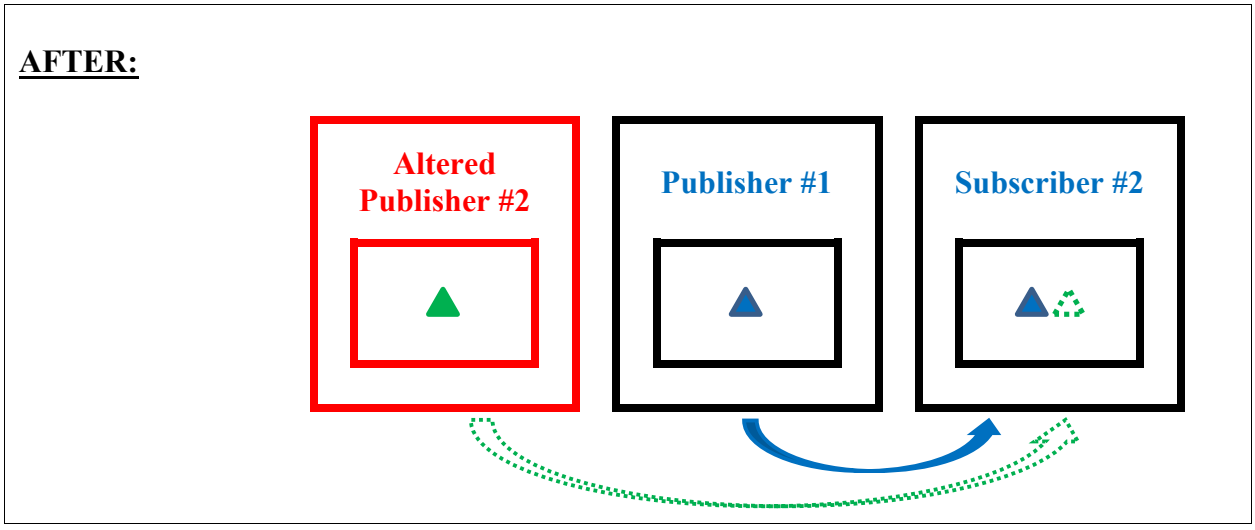

Figure 16: Self-contained Demonstration #4: Post-exploit state of DDS Entities (Data Deletion).

3.5.6.2 DDS Security Issue #4: Architecture and Implementation

Self-contained demonstration of this DDS security issue involved the alteration of the `LIFESPAN` QoS Policy for Green Triangle Topic data on Publisher #2. The `LIFESPAN` QoS Policy of a Topic determines when data becomes invalid and when it is automatically deleted by DDS. 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.

This self-contained demonstration involved specifying a `LIFESPAN` value of `100ms` for the Triangle Topic on Publisher #2. A value of `100ms` resulted in RTPS messages sent from Publisher #2, and occasional Green Triangles displayed at Subscriber #2. 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 Publisher #2 involved the same methods as were described in DDS Security Issue #3 – refer to section 3.5.5.2 for details.

Figure 17 below illustrates the architectural view of the DDS Entities involved in self-contained demonstration of this DDS security issue (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.

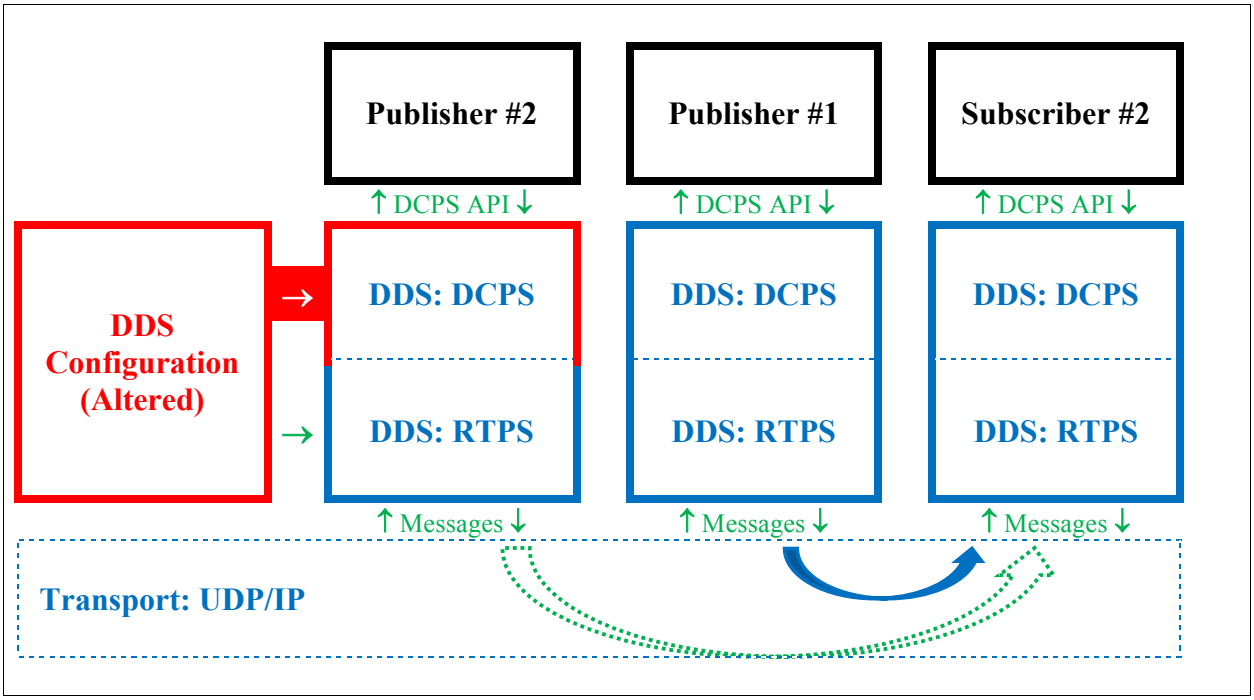

Figure 17: Self-contained Demonstration #4: Architecture stack view (Data Deletion).

3.5.6.3 DDS Security Issue #4: Results and Discussion

Execution of this self-contained demonstration successfully showed that this DDS security issue can be used in a data omission or data deletion attack. The altered configuration of Publisher #2 resulted in Subscriber #2 receiving intermittent data (Green Triangles) from Publisher #2. Existing data (Blue Triangles) from Publisher #1 was not affected.

The value of the `LIFESPAN` QoS Policy for the Triangle Topic on Publisher #2 greatly affected the results. Several `LIFESPAN` values were used within the virtual environment (described in Appendix D) 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).

The `RELIABILITY` QoS Policy specifies if reliable transmission of the data is required. Reliable communication is specified when the `RELIABILITY` QoS Policy is set to `RELIABLE`, and 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 as above).

The time in which data travels from one DDS Entity to another is highly dependent on the system under which it is executed. Therefore, any changes to the virtual environment, or deployment on a physical system would require adjustment of the `LIFESPAN` values to achieve the same results. If the goal was to stop the data flow, then the most reliable way would be to set the `LIFESPAN` value to `1ns`. If the goal was to have intermittent data display on the subscribing DDS Entity, then finer tuning would be required, and the results would still be sensitive to network load or the load of the host running the subscribing DDS Entity.

Implementing this self-contained demonstration using the Shapes Demo application required the same mechanisms that were employed in DDS Security Issue #3. Discussion of those mechanisms are described section 3.5.5.3.

3.5.7 DDS Security Issue #5: Misuse of the `LocatorList` Environment Variable Causing Domain Misdirection During Participant Discovery

This DDS security issue involved the use of 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. This self-contained demonstration involved changing the `LocatorList` environment variable and restarting the subscriber, redirecting it to join a malicious DDS-based system and subsequently subscribe to maliciously created data. The `LocatorList` environment variable was changed

via modification of the script that launches the application containing the subscriber. Details of this self-contained demonstration are described in the sections that follow.

3.5.7.1 DDS Security Issue #5: Self-contained Demonstration Overview

The goal of this self-contained demonstration was to prove that a malicious change in `LocatorList` environment variable specified in the startup script of a VM on which a DDS subscriber executes can cause the subscriber to join a different DDS domain or DDS network. Specifically, a different IP address specified by the `NDDS_DISCOVERY_PEERS` environment variable (which is the RTI Connext DDS™ implemented name for the `LocatorList` environment variable), causes a new or restarted subscriber to contact that IP address first to discover other DDS Entities, as part of the RTPS SPDP process. If other malicious DDS Entities (on a parallel malicious DDS-based system) respond at this address, the subscriber will establish contact to these malicious DDS Entities instead of DDS Entities on the legitimate DDS network. 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 valid data, and would not know that the data displayed was invalid.

Three DDS Entities in the Shapes Demo test environment were used to demonstrate this DDS security issue. The benign DDS-based system comprised of Publisher #1 which published Blue Squares, and Subscriber #1 which subscribed to Squares. Figure 18 below illustrates the initial state of the relevant DDS Entities prior to demonstration of the DDS security issue (blue arrow line indicates original data flow). A malicious DDS Publisher #5 (not shown in Figure 19) which published Blue Squares was started with `NDDS_DISCOVERY_PEERS` set to `239.200.0.1` (i.e. not the default value used by the legitimate DDS Entities at `239.255.0.1`).

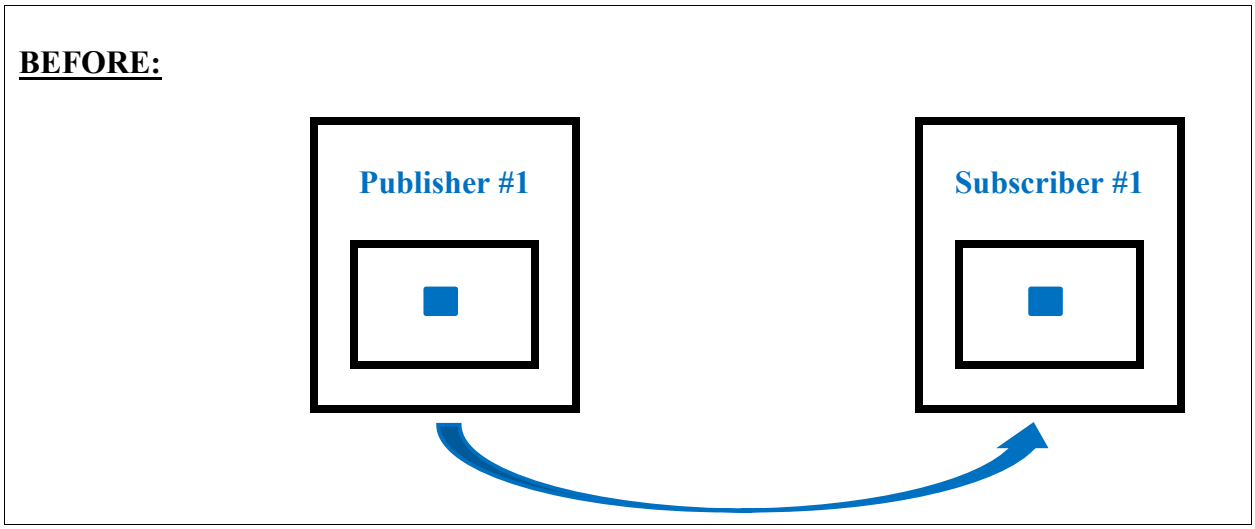

Figure 18: Self-contained Demonstration #5: Initial state of DDS Entities and communication flow.

To demonstrate this DDS security issue, the target subscriber's (Subscriber #1), startup script was altered such that the `NDDS_DISCOVERY_PEERS` environment variable specified `239.200.0.1`, matching the value that malicious Publisher #5 used. Once the host containing Subscriber #1 was restarted, the subscriber's SPDP process initiated contact with the malicious Publisher #5, as it was registered to that multicast IP address. Endpoint communication, using the Simple Endpoint Discovery Protocol (SEDP) process, was established between malicious Publisher #5 and Subscriber #1 based on the initial discovery of the malicious Publisher #5 via SPDP. Subscriber #1 did not receive any Blue Squares (from Publisher #1), instead it received Blue Squares from malicious Publisher #5. This is shown in Figure 19 below (malicious entity and malicious data flow is shown in red).

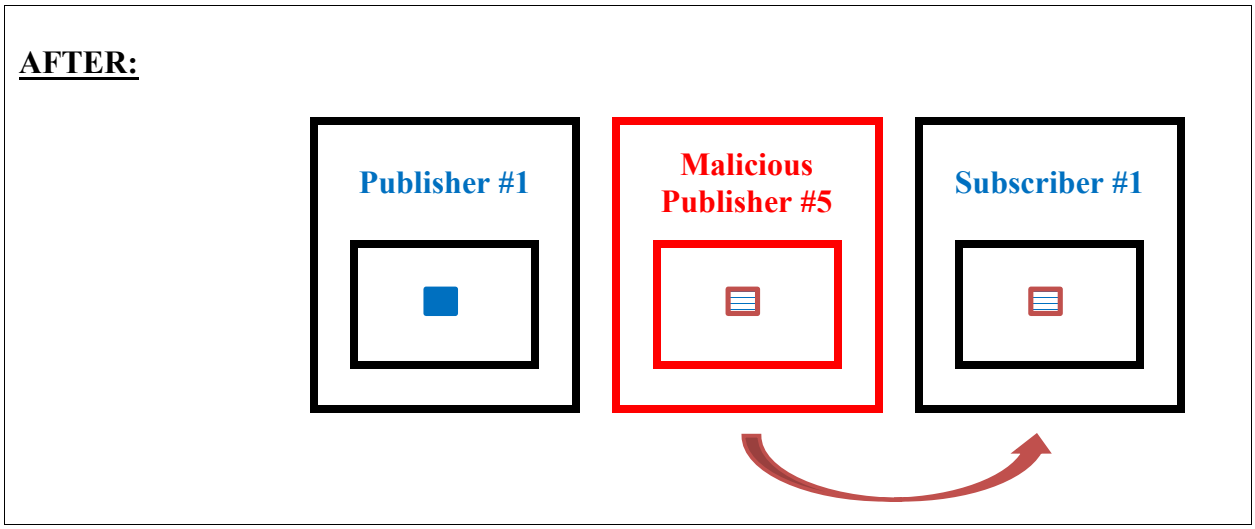

Figure 19: Self-contained Demonstration #5: Post-exploit state of DDS Entities (Host Hijacking)

3.5.7.2 DDS Security Issue #5: Architecture and Implementation

Self-contained demonstration of this DDS security issue exploited 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 Participant Discovery (SPDP) RTPS 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 Entities as part of the Endpoint Discovery (SEDP) process (i.e. which is based on the results of the SPDP process). 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 #5 and the hijacked Subscriber #1. 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 security issue.

Figure 20 below illustrates the architectural view of the DDS Entities involved in self-contained demonstration of this DDS security issue (blue line indicates original flow; malicious entity and malicious replacement data flow are shown in red). 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.

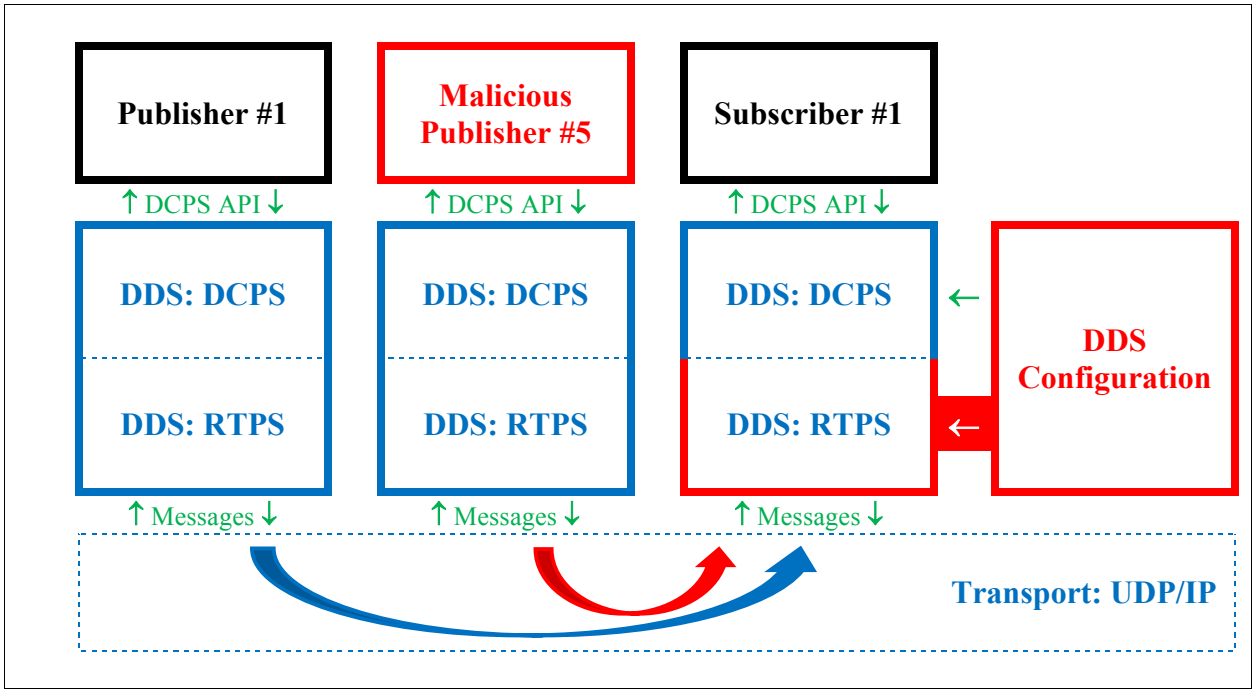

Figure 20: Self-contained Demonstration #5: Architecture stack view (Host Hijacking).

3.5.7.3 DDS Security Issue #5: Results and Discussion

Execution of this self-contained demonstration successfully showed that this DDS security issue can be used in a host hijacking attack (a combination of a data omission attack and a data

insertion attack). Subscriber #1 was successfully redirected to the malicious DDS network, which only comprised of malicious Publisher #5 (for simplicity). Subscriber #1 only received Blue Squares from the malicious Publisher #5, which were indistinguishable from valid Blue Squares except that the "fill" colour of the shape was cross-hatched (which was only added for ease of observability during testing). There was no indication at Subscriber #1 that any change in data distribution state had occurred.

Several minutes after execution of the self-contained demonstration, it was observed that the maliciously altered Subscriber #1 started to receive regular Blue Squares again from Publisher #1, along with the malicious Blue (cross-hatched) Squares from malicious Publisher #5. This was due to the fact that the existing benign DDS network (comprised of Publisher #1), had previously completed the SPDP discovery process with Subscriber #1, and therefore knew at which endpoint network address to re-establish communication – there was no need for it to re-initialize communication via the SPDP process (which would have used the default non-altered addressed specified by its `NDDS_DISCOVERY_PEERS` environment variable). This meant that this security issue could not be used to completely redirect a node onto a new DDS network or DDS Domain, without either changing the network address of the redirected host, or by waiting a sufficiently long time before restarting the target node in order for it to "expire" from the DDS network (i.e. for connection information to be deleted from the DDS Entities on the DDS network).

While this result was not expected at first, it does provide an interesting method of inserting data into a compromised subscriber with altered configuration, without a malicious publisher being part of the DDS network, or being seen by entities on the DDS network. No other DDS Entities were aware of, or had any means of discovering the malicious Publisher. Therefore, this DDS Security issue could be used to implement a stealthier and less easily detectable method of implementing a data insertion attack on a DDS network.

3.6 Summary

This chapter provided described the scope and selection process used to identify DDS security issues for analysis. The analysis was done using a custom DDS taxonomy developed to categorize and analyze the identified DDS issues. The analysis revealed 60 DDS security issues logically grouped in four key areas: DCPS, RTPS, Transport mechanisms and Configuration

mechanisms. Five of the 60 identified DDS security issues were modelled and demonstrated in a self-contained environment, which showed the effectiveness of each one in producing the malicious effects as described in its individual analysis. Each self-contained demonstration was performed in an isolated and controlled environment where only the effects of that particular security issue were produced. Validation of the DDS security issues that were modelled and demonstrated in a self-contained environment, will be done by combining them into an end-to-end DDS attack scenario, as detailed in the next chapter.

Chapter 4 Validation of Potential DDS Security Issues

4.1 Introduction

The previous chapter described the analysis of DDS security issues, which revealed 60 DDS security issues. Five of these security issues were modelled and demonstrated in a self-contained environment that showed the effectiveness of each one in producing the malicious effects in an isolated and controlled environment. This chapter details how these DDS security issues were combined into an end-to-end DDS attack scenario which used a different DDS application to validate the results found in the self-contained demonstration environment. The validation scenario used a DDS-based ATC example application to provide context and demonstrate the real-world implications of exploitation of DDS security issues.

4.2 Validation of 5 DDS Security Issues

The DDS security issues that were modelled and demonstrated in a self-contained environment from section 3.5 were combined into one end-to-end validation scenario. The end-to-end validation scenario was constructed to demonstrate various Attack Types as described in the 4th dimension of the DDS Malware Taxonomy (see section 3.4.2) as was demonstrated in a self-contained environment. These attack types included DoS attacks, network attacks, repudiation attacks and data attacks.

The DDS security issues included in the validation scenario produced the Information Impacts as described in the 6th dimension of the DDS Malware Taxonomy (see section 3.4.2). Otherwise known as the 5 D's [30], the following Information Impacts are used in the validation scenario:

- **Denial of Information**: This involved preventing communication between legitimate users, which in the validation scenario involved preventing a publisher or subscriber from sending or receiving data;
- **Destruction of Information**: This involved illegitimate deletion of information, which in the validation scenario involved expiry of data during the publication process, followed by its automatic deletion;

- **Disclosure of Information**: This involved illegitimate access to information, which in the validation scenario involved both unauthorized subscription and unauthorized publication of data;
- **Distortion of Information**: This involved manipulation and corruption of information (i.e. violation of data integrity), which in the validation scenario involved publication of new and altered data that subscribers would receive as 'correct'; and
- **Discovery of Information**: This involved gathering and storage of ancillary information that could be extracted and analyzed for future use. This information could have included network topology and communication flows (i.e. network mapping and reconnaissance methods that could reveal intelligence that the attacker could exploit in future attacks or operations). Note that even though the validation scenario did not result in discovery of information, previous inside knowledge of the network topology was used in the validation scenario to demonstrate the previous 4 types of Information Impacts (i.e. the previous 4 D's).

4.2.1 Validation Scenario Application Environment

The validation scenario was developed to demonstrate the above-mentioned Attack Types and Information Impacts using a DDS-based Air Traffic Control (ATC) application running on several host VMs. The application that was used was the "Vehicle Tracking" example application [35] provided by RTI, running on several Lubuntu VMs in a virtual network using Parallels on a MacBook (software and operating system versions are detailed in Appendix D).

The "Vehicle Tracking" example application used several DDS Entities to simulate an ATC environment above the San Francisco bay area. DDS Entities included Radar sensors and Operator Display workstations. Radar sensors were publishers on the DDS-based system and the Operator Display workstations were the subscribers. Aircraft tracks published by the Radar sensors use the DDS Topic of "Track" and individual unique aircraft tracks were Instances of the Topic "Track".

4.2.2 Validation Scenario Overview

Real-world ATC systems and other C2 systems have many of the same requirements, in that they need to track objects and display information about the tracked objects. The use of an ATC

example application with Radar sensors (publishers) and Operator Display workstations (subscribers) running on multiple hosts is an ideal analog to both ATC systems and other C2 systems. It is reasonable to assume that typical operation of these systems involves periodically (often daily) restarting or rebooting Radars and Operator Displays for maintenance and to satisfy certification and regulatory requirements. Many of the changes required to exploit DDS security issues can be done during these restarts, and restarting DDS Entities was easily done in this validation environment.

The same DDS Security Issues that were modelled and demonstrated in the self-contained environment using the Shapes Demo application (see section 3.5), were included in the end-to-end validation scenario using the ATC example application.

4.2.3 Validation Scenario Execution

Details of how each DDS security issue was integrated then executed in the validation scenario are described in the sections that follow.

4.2.3.1 Validation Scenario: Initial State

The initial state of the ATC validation scenario used four DDS Entities – two Radars executing an application called "`track_generator`" and two Operator Display workstations executing an application called "`track_gui`", each running on its own VM as follows:

- **Radar #1**: published and updated aircraft track instances with `EXCLUSIVE` ownership. This represents a situation where only *one Radar at a time* is permitted to publish data for aircraft tracks. This is analogous to a tracking Radar or a data fusion processor that correlates and fuses multiple tracks and track information into one aircraft track, thereby producing a higher quality aircraft track for publication;
- **Display #1**: subscribed to aircraft tracks with `EXCLUSIVE` ownership. This represents Operator Displays that interact with aircraft tracks *from one Radar* (or data-fusion processor) – aircraft tracks that have already been correlated and fused with data from other sources. Aircraft tracks displayed at these Operator Displays would be considered to have a higher quality and trustworthiness, and an ATC operator would use them to directly control an aircraft;

- **Radar #2**: published and updated aircraft tracks with `SHARED` ownership. This represents Radars that publish incomplete or lower-quality aircraft track data or other information. This type of Radar publication represents a Primary Surveillance Radar (PSR) or a Secondary Surveillance Radar (SSR) that publishes partial aircraft track information that could be used by an operator or data fusion processor to perform track correlation and data fusion (e.g. Display #2, described below); and
- **Display #2**: subscribed to aircraft tracks with `SHARED` ownership. This represents Operator Displays that interact with *data from multiple Radars* such as Radars represented by Radar #2 (above). The role of this type of Operator Display would be to perform track correlation and data fusion, thus combining information from multiple Radars (or other sources) into one higher-quality aircraft track. This is done semi-automatically by this type of Operator Display and requires some interaction from the ATC operator to resolve conflicts or fill-in missing information.

The data distribution state in the validation environment matched that of the initial state described in the self-contained demonstration environment (see section 3.5.2). Figure 21 illustrates the initial states of the two Operator Displays, each one displaying the appropriate aircraft tracks as published by the two Radars. Aircraft tracks are displayed as red dots on the "map" portion of the GUI and are also listed on the tabular portion of the GUI.

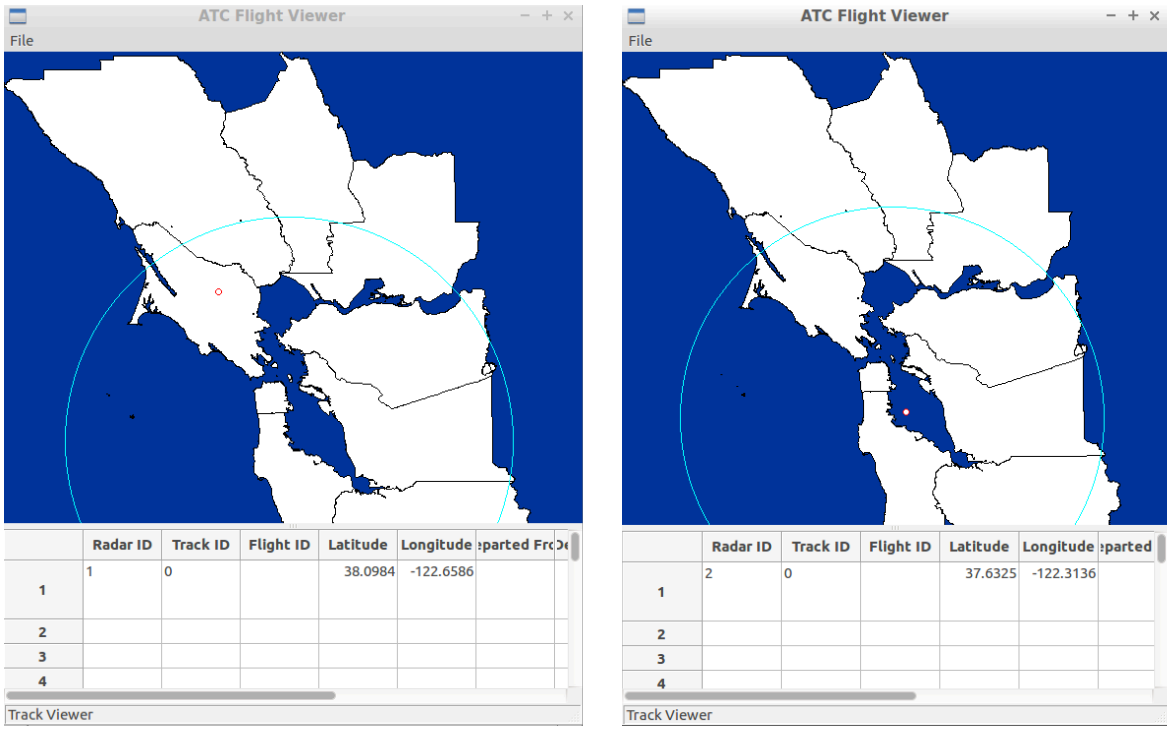

Figure 21: Validation Scenario: Initial state of Display #1 and Display #2.

4.2.3.2 Validation Scenario: Stage #1

This stage of the validation scenario validated *DDS Security Issue #1: "Misuse of Anonymous Subscribe and Republish Functionality"*, which was modelled and demonstrated in a self-contained environment in section 3.5.3. The goal of this stage was used to perform a "track multiplication and falsification" style of attack to distract and confuse the ATC operators who use the Operator Displays. This involved execution of a new malicious DDS Entity which subscribed to aircraft tracks and republished multiple copies of slightly altered aircraft tracks.

At this stage in the validation scenario, the state of the relevant DDS Entities was as follows:

- **Radar #2**: published and updated aircraft tracks with `SHARED` ownership; and
- **Display #2**: subscribed to aircraft tracks with `SHARED` ownership.

In this context, Display #2 received what it interprets to be incomplete or low-quality aircraft track data *from multiple Radars* (such as Radar #2) to aid the ATC operator to perform track correlation and data fusion, thus combining multiple pieces of information into one higher-quality aircraft track.

The data distribution state was the same as was modelled and demonstrated previously in the self-contained environment (see section 3.5.3). Figure 22 illustrates the initial state of Display #2, which displayed aircraft tracks from Radar #2.

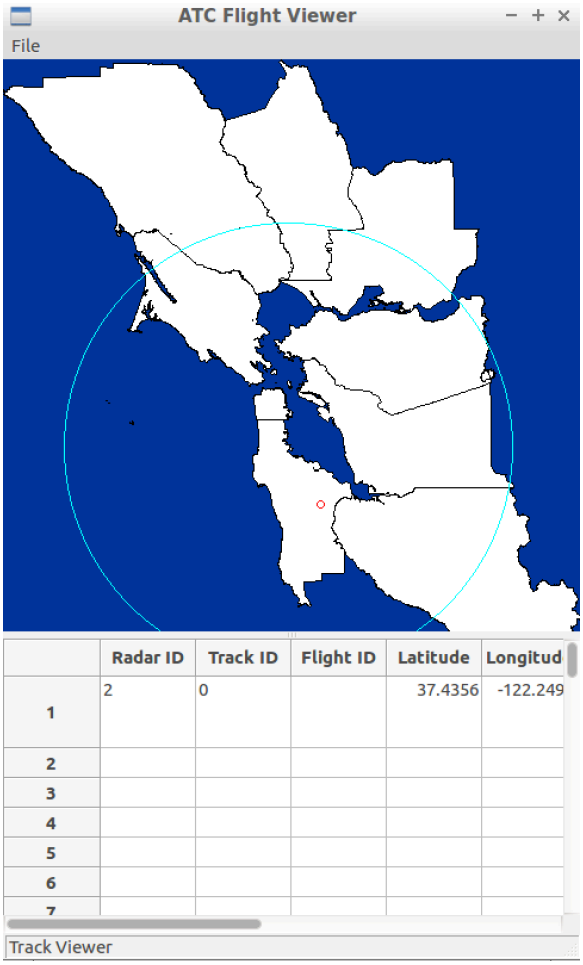

Figure 22: Stage #1: Initial state of Display #2.

In this stage, a new malicious DDS-based application was executed from a new host during normal operation of the ATC system. This application contained two malicious DDS Entities – a subscriber and a publisher which subscribed to aircraft tracks (with `SHARED` ownership) and republished four copies of slightly altered aircraft tracks (their positions were altered to make it look like a cloud of aircraft tracks). All Operator Displays (including Display #2) that subscribed to aircraft tracks with `SHARED` ownership, received the new aircraft tracks as well as the existing aircraft tracks.

This new malicious application was used to perform a "track multiplication and falsification" style of attack to distract and confuse ATC operators at the Operator Displays. In this case, the

role of Display #2, was to support semi-automatic track correlation and data fusion performed by the ATC operator. Multiple new tracks, each containing subtle differences from the original track could easily distract and overload an operator who would find it difficult to determine which tracks were legitimate. This could put the safety of the real aircraft and other friendly assets in danger. Figure 23 illustrates the resulting state of Display #2, which displayed aircraft tracks from both Radar #1 and the malicious Radar after the execution of the malicious application.

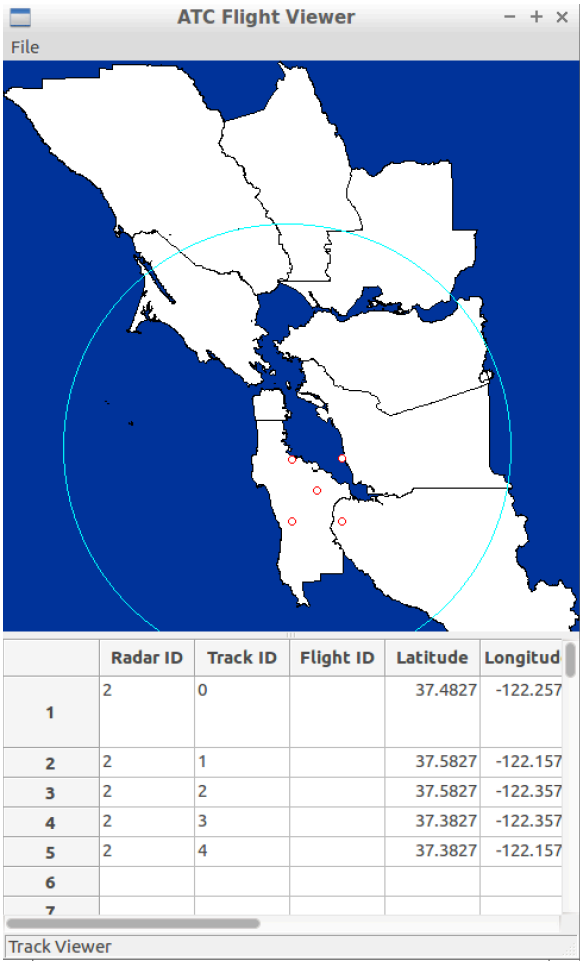

Figure 23: Stage #1: Post-exploit state of Display #2 (Data Multiplication).

4.2.3.3 Validation Scenario: Stage #2

This stage of the validation scenario validated *DDS Security Issue #2: "Misuse of* `OWNERSHIP_STRENGTH` *QoS Policy and* `EXCLUSIVE` *Data Ownership"*, which was modelled and demonstrated in a self-contained environment in section 3.5.4. The goal of this stage was to perform a "track hijacking" style of attack to take over publication of aircraft tracks

from one Radar (Radar #1) to a new malicious Radar (Radar #4). This involved execution of a new application containing the same executable as Radar #1, but with a configuration that had a higher value of `OWNERSHIP_STRENGTH` of the track data than was published by the legitimate Radar #1, thereby hijacking the aircraft track data publication.

At this stage in the validation scenario, the state of the relevant DDS Entities was as follows:

- **Radar #1**: published and updated aircraft track instances with `EXCLUSIVE` ownership; and
- **Display #1**: subscribed to aircraft tracks with `EXCLUSIVE` ownership.

In this context, Display #1 received aircraft tracks *from one Radar* such as Radar #1, and an ATC operator would use them to directly control an aircraft. The aircraft tracks from Radar #1 were considered to have a higher quality and trustworthiness, as they would have already been correlated and fused with data from other sources (Radar #1 could represent a tracking Radar or a combined PSR/SSR Radar that performs automatic correlation). In this situation, only one Radar at a time is permitted to publish data for a particular aircraft track.

The data distribution state was the same as was modelled and demonstrated previously in the self-contained environment (see section 3.5.4). Figure 24 illustrates the initial state of Display #1, which displayed aircraft tracks from Radar #1.

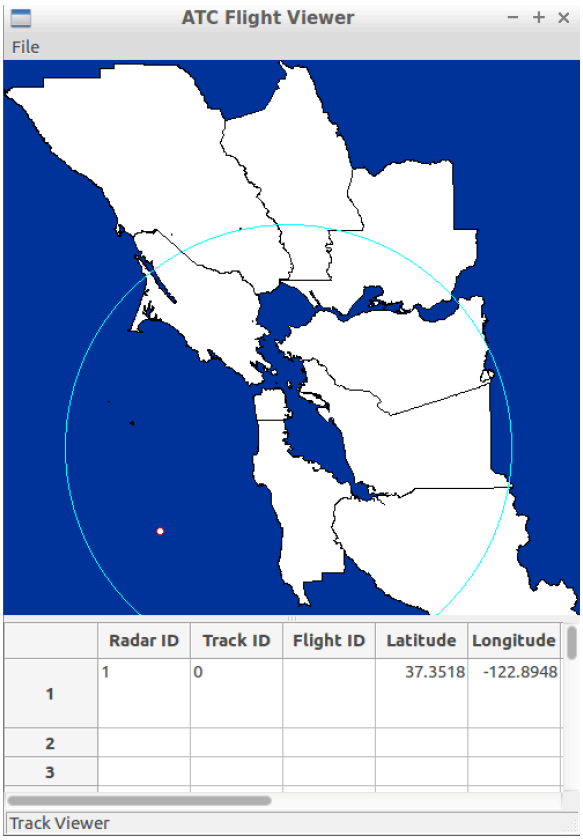

Figure 24: Stage #2: Initial state of Display #1

In this stage, a new malicious application called Radar #4 was executed from a new host during normal operation of the ATC system. This application was the same as Radar #1, but it had a value of `OWNERSHIP_STRENGTH` that was higher than Radar #1. Publication of aircraft tracks from malicious Radar #4 replaced the aircraft tracks from Radar #1, as displayed on all Operator Displays (including Display #1) that subscribed to aircraft tracks with `EXCLUSIVE` ownership – aircraft tracks from Radar #1 were no longer seen at any Operator Displays.

This new maliciously configured application (i.e. malicious Radar #4) was used to perform a "track hijacking" style of attack to take over publication of aircraft tracks from Radar #1. In this case, the role of Display #1 was to interact with high-quality and trustworthy aircraft tracks (i.e. that were already correlated and fused from other data sources). Instead, Display #1 displayed illegitimate and untrustworthy tracks. Trusting invalid or inaccurate aircraft track data could put the safety of the real aircraft and other friendly assets in danger. Figure 25 illustrates the resulting state of Display #1, which previously displayed aircraft tracks from Radar #1 and now displays aircraft tracks from malicious Radar #4 after the execution of the malicious application

(note the change in position of the aircraft track on the "map" portion of the GUI and the different coordinates listed in the table).

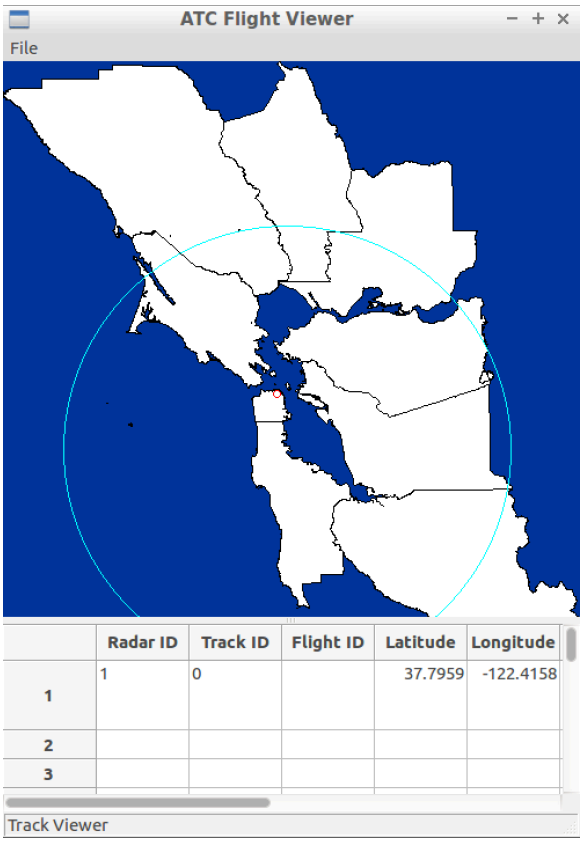

Figure 25: Stage #2: Post-exploit state of Display #2 (Data Hijacking)

4.2.3.4 Validation Scenario: Stage #3

This stage of the validation scenario validated *DDS Security Issue #3: "Misuse of* `OWNERSHIP_KIND` *QoS Policy and* `SHARED` *Data Ownership"*, which was modelled and demonstrated in a self-contained environment in section 3.5.5. The goal of this stage was to perform a "track hiding & redirection" style of attack to hide aircraft tracks from certain Operator Displays (who specified `EXCLUSIVE` ownership), and redirect it to other Operator Displays (who specified `SHARED` ownership). This involved exploiting a periodic restart of the Radar #1 with an altered configuration such that the ownership of the published aircraft track data changed from `EXCLUSIVE` to `SHARED`.

At this stage in the validation scenario, the state of the relevant DDS Entities was follows:

- **Radar #1**: published and updated aircraft track instances with `EXCLUSIVE` ownership;

- **Display #1**: subscribed to aircraft tracks with `EXCLUSIVE` ownership;
- **Radar #2**: published and updated aircraft tracks with `SHARED` ownership; and
- **Display #2**: subscribed to aircraft tracks with `SHARED` ownership.

In this context, Display #1 received higher quality and trustworthy aircraft tracks *from one Radar* such as Radar #1, and an ATC operator at Display #1 would use them to directly control an aircraft. In addition, Display #2 received incomplete or low-quality aircraft track data *from multiple Radars* such as Radar #2, in order to aid the ATC operator to perform semi-automatic track correlation and data fusion.

The data distribution state was the same as was modelled and demonstrated previously in the self-contained environment (see section 3.5.5). Figure 26 illustrates the initial state of Display #1, which displays aircraft tracks from Radar #1, and the initial state of Display #2, which displays aircraft Tracks from Radar #2.

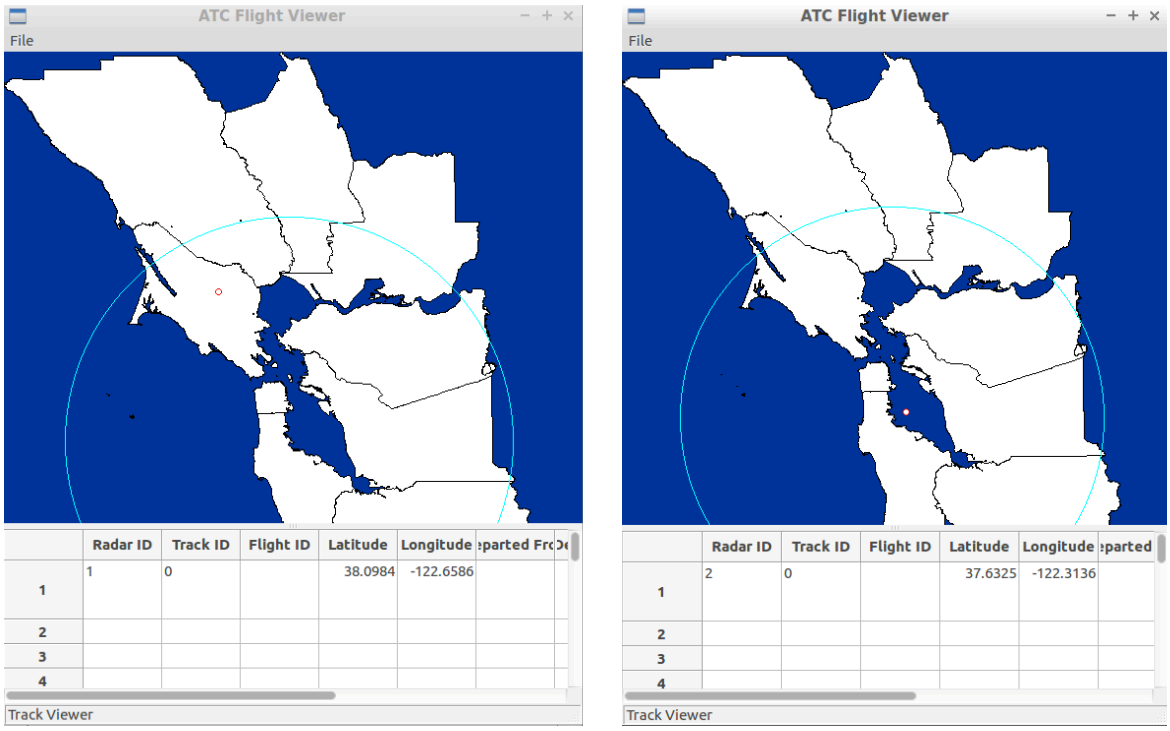

Figure 26: Stage #3: Initial state of Display #1 and Display #2

In this stage, a periodic restart of Radar #1 was exploited such that a maliciously altered configuration file was read by Radar #1. The configuration modification of Radar #1 changed publication of aircraft tracks from `EXCLUSIVE` ownership to `SHARED` ownership. All Operator

Displays that subscribed to aircraft tracks with `SHARED` ownership (including Display #2), now received the aircraft tracks from maliciously altered Radar #1, as well as the previous aircraft tracks from Radar #2. Display #1 no longer received aircraft tracks from Radar #1

This was used to perform a "track hiding & redirection" style of attack to hide aircraft tracks from Display #1 and redirect them to Display #2. Multiple new tracks from maliciously altered Radar #1 could easily distract and overload an ATC operator at Display #2 who would find it difficult to determine which tracks were the ones that they were supposed to handle. In addition, the operator at Display #1 would not receive any aircraft tracks from Radar #1, which might not be noticed especially if Display #1 also displayed aircraft tracks from another source. This could put the safety of the real aircraft and other friendly assets in danger. Figure 27 illustrates the resulting state of Display #1 and Display #2, after the restarting Radar #1 with a maliciously-altered configuration. Note that even though the tabular portion of the GUI at Display #1 listed an aircraft track from Radar #1, it was not updated, nor was the aircraft track displayed in the "map" view of the GUI – this indicates a "stale" aircraft track, that is eventually deleted automatically from the subscriber's history cache at Display #1.

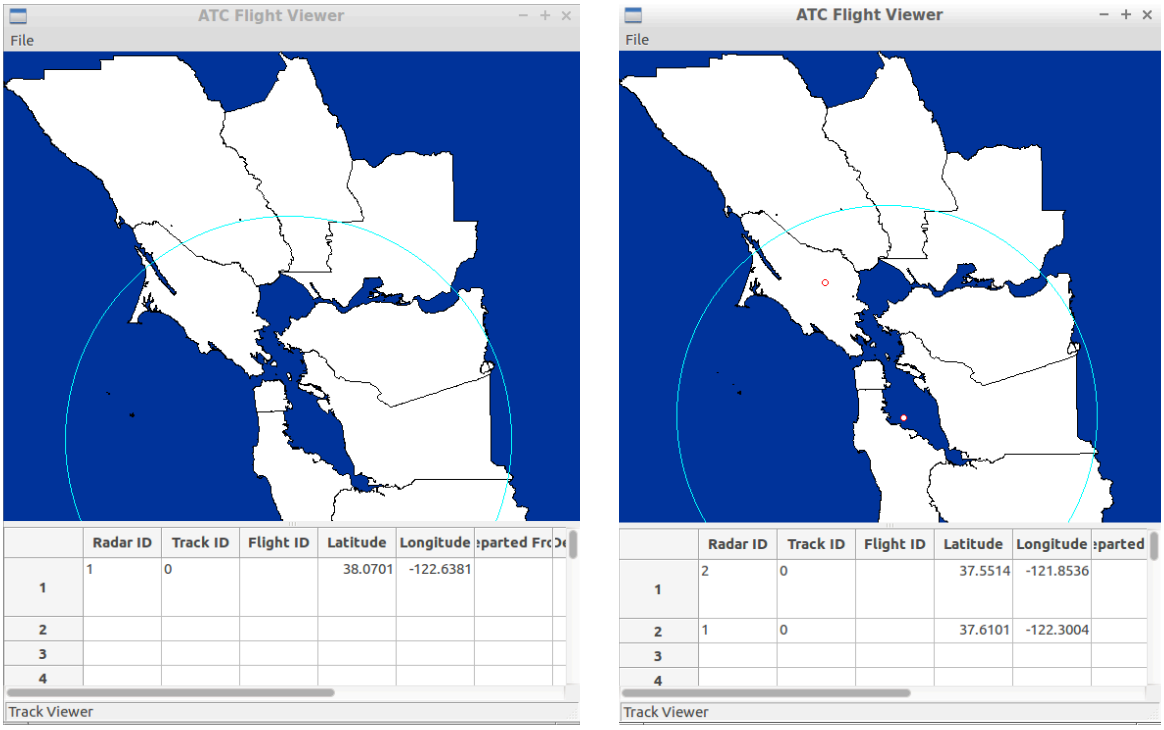

Figure 27: Stage #3: Post-exploit state of Display #1 and Display #2 (Data Hiding & Redirection)

4.2.3.5 Validation Scenario: Stage #4

This stage of the validation scenario validated *DDS Security Issue #4: "Misuse of `LIFESPAN` QoS Policy Causing Immediate Data Expiration"*, which was modelled and demonstrated in a self-contained environment in section 3.5.6. The goal of this stage was to perform a "track deletion" style of attack to hide aircraft tracks from all Operator Displays (with `SHARED` ownership). This involved exploiting a periodic restart of the Radar #2 with an altered configuration such that `LIFESPAN` of the published aircraft track data was short enough that it either did not get published, or it expired and was deleted upon receipt at any Operator Display (represented by Display #2).

At this stage in the validation scenario, the state of the relevant DDS Entities was follows:

- **Radar #1**: published and updated aircraft track instances with `SHARED` ownership (i.e. the result of stage #3 in this scenario);
- **Radar #2**: published and updated aircraft tracks with `SHARED` ownership; and
- **Display #2**: subscribed to aircraft tracks with `SHARED` ownership.

In this context, Display #2 received incomplete or low-quality aircraft track data *from multiple Radars* such as Radar #2 (and also Radar #1, as a result of the previous stage), in order to aid the ATC operator to perform semi-automatic track correlation and data fusion.

The data distribution state was the same as was modelled and demonstrated previously in the self-contained environment (see section 3.5.6). Figure 28 illustrates the initial state of Display #2, which displayed aircraft tracks from Radar #1 and Radar #2.

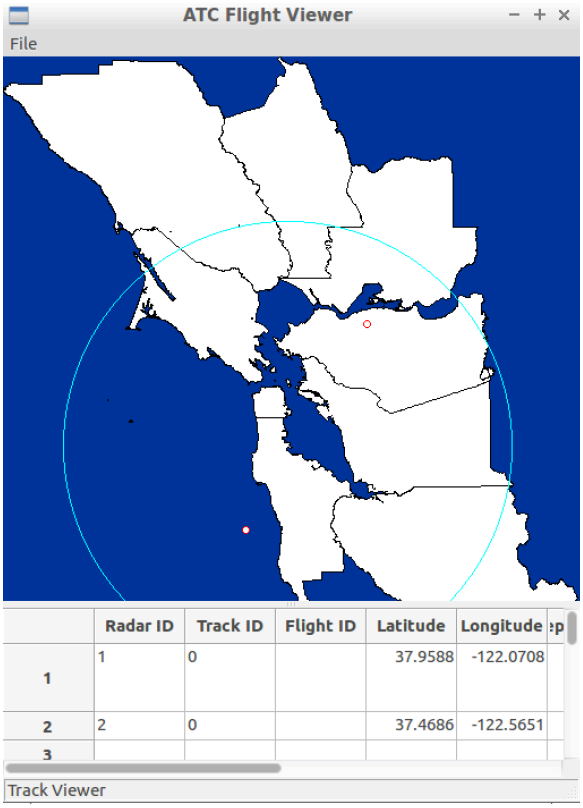

Figure 28: Stage #4: Initial state of Display #2

In this stage, a periodic restart of Radar #2 was exploited such that a maliciously altered configuration file was read by Radar #2. The configuration change of Radar #2 changed the `LIFESPAN` of published aircraft tracks to a duration that caused the DDS infrastructure to delete them prior to being received by Display #2. Therefore, Display #2 no longer received aircraft tracks from Radar #2, although it still received aircraft tracks from Radar #1.

The role of Display #2 was to assist the ATC operator to perform track correlation and data fusion semi-automatically. If the ATC operator at Display #2 does not receive aircraft tracks from Radar #2 that was needed for track correlation and data fusion (i.e. combining this information with aircraft information from other Radars and other sources), the operator might generate a "higher-quality" track that is in fact incorrect or inaccurate. This could put the safety of the real aircraft and other friendly assets in danger. Figure 29 illustrates the resulting state of Display #2, after a restart of the application with a maliciously-altered configuration. Note that even though the tabular portion of the GUI at Display #2 listed an aircraft track from Radar #2, it was not updated, nor was the aircraft track displayed in the "map" view of the GUI – this

indicates a “stale” aircraft track, that is eventually deleted automatically from the subscriber’s history cache at Display #2.

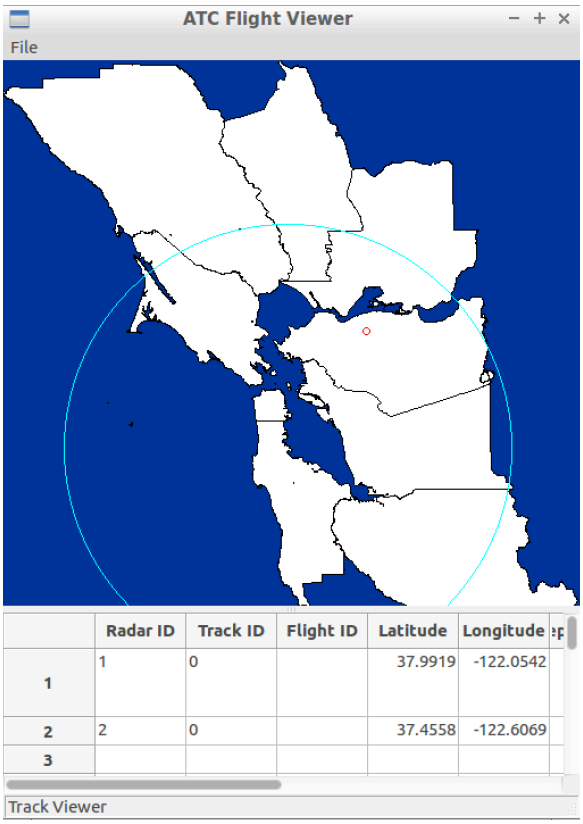

Figure 29: Stage #4: Post-exploit state of Display #2 (Data Deletion)

Note that different values of `LIFESPAN` were used with the ATC Example application versus the Shapes Demo application discussed in section 3.5.6. These values are listed below:

- **`LIFESPAN < 0.8ms`**: resulted in no RTPS messages sent from Radar #2, and therefore no aircraft tracks were displayed at Display #2 (i.e. aircraft tracks were automatically deleted while in Radar #2’s publisher history cache);
- **`LIFESPAN 0.8ms – 1.2ms`**: resulted in RTPS messages sent from Radar #2, and occasional aircraft tracks were displayed at Display #2 (i.e. before they were automatically deleted while in Display #2’s subscriber history cache); and
- **`LIFESPAN >= 1.2ms`**: resulted in RTPS messages set from Radar #2, and aircraft tracks were displayed at Display #2 (i.e. aircraft tracks were 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, as described in section 3.5.6.3; 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.

4.2.3.6 Validation Scenario: Stage #5

This stage of the validation scenario validated *DDS Security Issue #5: "Misuse of the `LocatorList` Environment Variables Causing Domain Misdirection During Participant Discovery"*, which was modelled and demonstrated in a self-contained environment in section 3.5.7. The goal of this stage was to perform a "host hijacking" style of attack to redirect an Operator Display onto a malicious DDS Domain, and receive false aircraft track data from a malicious Radar. This involved exploiting a periodic restart of Display #1 with an altered DDS Discovery configuration such that it joined the malicious DDS network.

At this stage in the validation scenario, the state of the relevant DDS Entities was follows:

- **Radar #1**: published and updated aircraft track instances with `SHARED` ownership (i.e. the result of stage #3 in this scenario);
- **Display #2**: subscribed to aircraft tracks with `SHARED` ownership.

In this context, Display #2 received incomplete or low-quality aircraft track data *from multiple Radars* such as Radar #2 (and also Radar #1, as a result of the stage #3 of the scenario). Note that as a result of stage #4 in the scenario, Radar #2 no longer sent aircraft tracks to Display #2. Therefore, the only remaining communication flow at this stage of this scenario was between Radar #1 and Display #2.

The data distribution state was the same as was modelled and demonstrated previously in the self-contained environment (see section 3.5.7), except that the results of the previous stages (stage #3 and stage #4 of the scenario) remained in effect (i.e. there was only communication

between Radar #1 and Display #2). Figure 30 illustrates the initial state of Display #2. (i.e. Display #2 displays aircraft tracks from Radar #1).

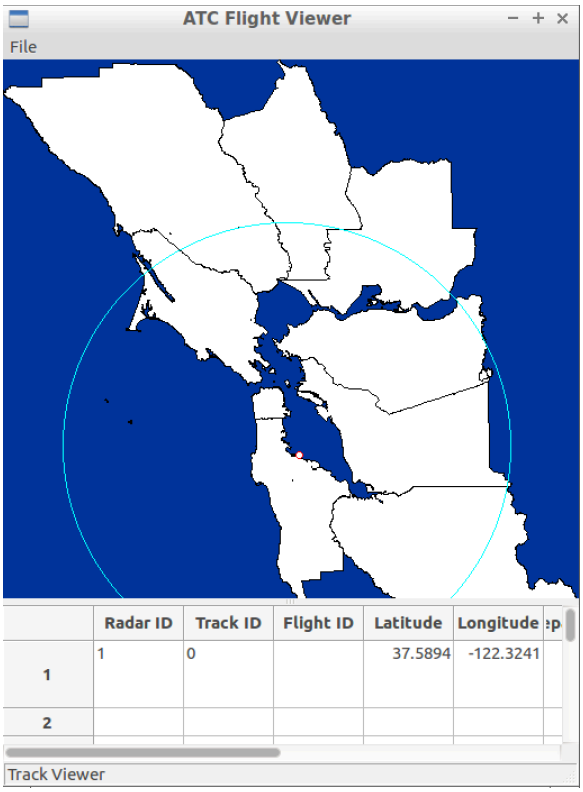

Figure 30: Stage #5: Initial state of Display #2.

In this stage, a periodic restart of Display #2 was exploited such that a maliciously altered startup script file changed an environment variable used by Display #2. The altered environment variable redirected the DDS discovery process such that Display #2 joined a malicious DDS network, represented by malicious Radar #5. Display #2 then received the aircraft tracks from malicious Radar #5 on the malicious DDS Domain. Note that after several minutes, Display #2 started to receive valid aircraft tracks from the legitimate Radar #1, due to the fact that Radar #1 still knew the address of Display #2 (i.e. it did not need to re-initiate the SPDP discovery process – this phenomenon is described in in section 3.5.7.3).

This was used to perform a "host hijacking" style of attack to redirect a Display #2 onto a malicious DDS Domain, and receive false aircraft track data from malicious Radar #5. In this case, the role of Display #2, was to aid the ATC operator to perform track correlation and data fusion, thus combining multiple pieces of information into one higher-quality aircraft track. Multiple new tracks from malicious Radar #5 could easily distract, confuse and overload an

operator (e.g. at Display #2) who would find it difficult to determine which aircraft tracks were legitimate and which aircraft tracks could be correlated into a higher-quality aircraft track. 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 (i.e. indication of the presence or activity of a malicious DDS Entity). Anomalous behaviour introduced by an undetected malicious DDS Entity could put the reliability and credibility of the DDS-based system into question. Figure 31 illustrates the resulting state of Display #2, after restarting the application with a maliciously-altered configuration (i.e. Display #2 only displayed aircraft tracks from malicious Radar #5).

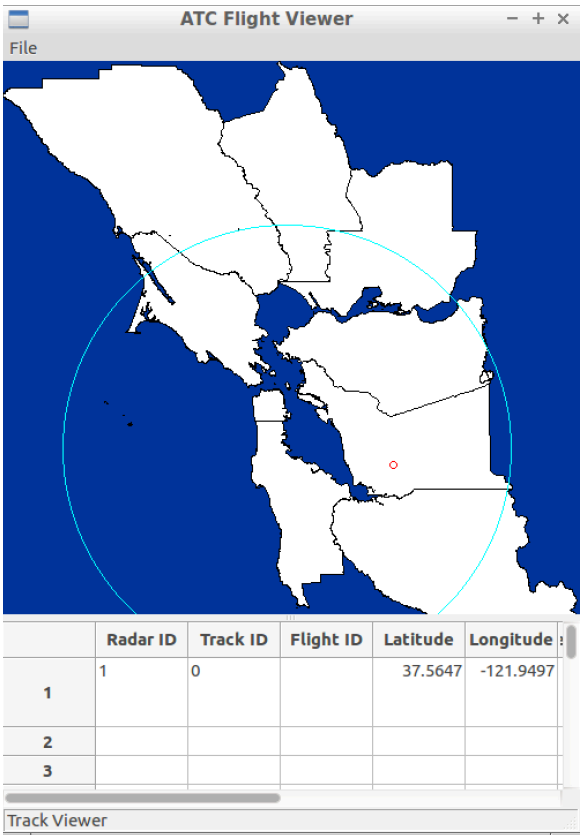

Figure 31: Stage #5: Post-exploit state of Hijacked Display #2 (Host Hijacking).

4.2.4 Validation Scenario Conclusion

The end-to-end validation scenario was built using DDS security issues that were modelled and demonstrated in a self-contained environment (section 3.5). The end-to-end validation scenario demonstrated the same attack types (i.e. the 4th dimension of the DDS Malware Taxonomy

described in section 3.4.2) as was demonstrated in a self-contained environment, including DoS attacks, network attacks, repudiation attacks and data attacks. Even though the validation scenario was constructed from a different DDS application, it produced the same effects as the self-contained demonstrations in section 3.5. These effects included the Information Impacts (as described in the 6th dimension of the DDS Malware Taxonomy described in section 3.4.2), including Denial of Information, Destruction of Information, Disclosure of Information and Distortion of Information. This is summarized in Table 2.

Table 2: DDS Security Issue Validation Matrix

#	DDS Issue	Attack Type	Informational Impact
1	Misuse of Anonymous Subscribe and Republish Functionality	DoS Attack; and Data Attack, involving: - Data Insertion; and - Data Modification.	Track multiplication and falsification, involving: - Disclosure of Information; and - Distortion of Information.
2	Misuse of `OWNERSHIP_STRENGTH` QoS Policy and `EXCLUSIVE` Data Ownership	Data Hijacking Attack, involving: - Data Deletion; and - Data Insertion.	Track hijacking, involving: - Denial of Information; - Destruction of Information; and - Distortion of Information.
3	Misuse of `OWNERSHIP_KIND` QoS Policy and `SHARED` Data Ownership	Data Attack, involving: - Data Redirection.	Track hiding & redirection, involving: - Disclosure of Information; and - Denial of Information.
4	Misuse of `LIFESPAN` QoS Policy Causing Immediate Data Expiration	Data Attack, involving: - Data Deletion; and - Data Omission.	Track deletion, involving: - Denial of Information; or - Destruction of Information.
5	Misuse of the `LocatorList` Environment Variable Causing Domain Misdirection During Participant Discovery	Network Attack, involving: - Network Configuration Corruption.	Host hijacking, involving: - Denial of Information; and - Distortion of Information.

4.3 Validation of Research Aim

The aim of this research was to investigate DDS security issues and develop and validate DDS threat models that could be used by a malicious actor to compromise a host via a client-side attack on a DDS-based system (as described in section 1.4). Analysis of the DDS standards and implementations revealed 60 DDS security issues (as described in section 3.4.3), and five of them were modelled and demonstrated in an isolated and controlled environment using the DDS Shapes Demo application (as described in section 3.5). Self-contained demonstration revealed that the five selected DDS security issues could be exploited to compromise information in a DDS-based system on an individual basis. We have no reason to suspect that the remaining 55 security issues would be any less problematic, but we did not explore them further because of the level of effort required. The five DDS security issues that were modelled and demonstrated in a self-contained environment were then combined into an end-to-end attack scenario that used a different DDS-based application designed to represent an ATC system (as described in section 4.2). The validation scenario demonstrated the same effects as the self-contained demonstrations, and ATC information was compromised in the same fashion.

The DDS-based application used in the validation scenario was specifically chosen to be analogous to a real ATC system to bring operational context to the subject of DDS security. Specific informational effects of a compromised DDS-based ATC system included: Denial of aircraft information to an operator who needs it, Destruction of aircraft information prior to use, Disclosure of aircraft information to unauthorized entities and Distortion or corruption of aircraft information. In the worst-case scenario, these types of data compromises could lead to loss of life.

The two DDS-based applications used in the self-contained demonstrations and the validation scenario were very different in functionality and used different programming philosophies. Use of one application to reproduce the results from the other successfully validated the DDS security issues that could be exploited by a malicious actor in an attack on a DDS-based system.

The aim of this research was achieved through successful validation of the modelled DDS security issues. This was successfully done through demonstration of how each DDS security issue was used by a malicious actor to compromise a host via a client-side attack on a DDS-based system.

4.4 Discussion of DDS Security Issues

This research work focused on how the DDS protocol and the software used to implement it could be exploited by a client-side attack to compromise a DDS-based system. This research work covered situations where the intent was either malicious or benign (i.e. sloppy programming), but the effect was still harmful. Therefore, this research work could be used to either increase DDS security (via new standards or better vendor implementations), or to guide DDS application developers to develop more secure software.

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], but we do not believe that it addresses the core security issues discussed here. It only covers five specific security areas: Authentication, Access Control, Encryption, Logging and Data Tagging and there is no published research that indicates its effectiveness in handling the effects of a client-side attack. In addition, because it only covers a subset of all DDS security issues, the research work described in this thesis could be used to spawn additional research into this "secure" version of DDS, and potentially expose further security issues in it.

In addition, we found no published DDS vulnerability research that has developed and validated DDS threat models which could be used to develop methods of DDS malware detection. Use of the DDS security issues that were modelled, demonstrated and validated in this research work followed by modelling, demonstrating and validating the remaining 55 DDS security issues (revealed in this research) could be used to provide threat models to be used in the DDS-IDS being developed by Queen's University.

There is also no published analysis of how a DDS-based system may be manipulated in order to support an attack. This includes, but is not limited to, manipulation of the software and configuration used to implement the DDS middleware layer in order to attack other DDS Entities in the system. This research may be the only academic work that has exposed this aspect of DDS.

4.5 Summary

This chapter provided details of how five DDS security issues that were previously modelled and demonstrated in self-contained environment were 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 were demonstrated in the self-contained environments. Therefore, the research aim of investigating DDS security issues and developing and validating DDS threat models was successfully met, as the research demonstrated that these security issues could be exploited by a malicious actor to compromise hosts on a DDS-based system.

Chapter 5 Discussion and Conclusion

5.1 Discussion

The previous chapter provided details of how five DDS security issues that were demonstrated in self-contained environments were combined into an end-to-end validation scenario. An ATC DDS application was used to validate these DDS security issues, and it provided context and a real-world example of the seriousness of these DDS security issues. The following subsections describe the implications and importance of the results, and outlines potential future work.

5.1.1 Results of the Work

The validation scenario proved that exploitation of these five DDS security issues demonstrated in the self-contained environments could be carried out against a different DDS application with the same results. This also demonstrated the ease and effectiveness that these five DDS security issues could be used to successfully compromise a DDS-based system.

Since the analysis revealed another 55 DDS security issues, the same process used in this research work could be extended to cover these 55 issues and result in more numerous, and potentially more effective, methods of compromising a DDS-based system. This reinforces the need to continue analysis, demonstration and validation of the remaining DDS security issues discovered in this research. In addition, more research and refinement of the DDS Security specification [5] can be done such that DDS becomes less exploitable.

The results of this research work were not limited to a particular DDS vendor implementation – the results could be reproduced on different DDS vendor implementations with minimal change.

5.1.2 Importance of the Work

This research work focused on how the DDS protocol and the software used to implement it could be manipulated or modified to cause harm to a DDS-based system. This research work covered situations where DDS could be altered causing harm, regardless of intent – whether it was malicious or benign (i.e. sloppy programming). This is important because security and resiliency of DDS-based system relies on the notion that the hosts upon which DDS applications

execute, reside in a *trusted zone* and are therefore protected from misuse or abuse. The assumption that the hosts are a *trusted zone* does not hold – this is DDS' Achilles-heel.

Many organizations who deploy DDS-based systems provide physical security to prevent the host from being compromised; they also deploy DDS-based systems in such a way as to make them isolated from other networks. Physical security and network isolation do not prevent the myriad of client-side attacks from compromising a DDS-based system via alteration of the host application, configuration files or via alteration of the DDS libraries. While this was not within the scope of this research, a DDS-based system is also vulnerable if any of its host is infected from an attack that originates from outside the network. The new DDS Security specification [5], and the RTI Connext DDS Secure™ [6] implementation of this new specification still relies upon the notion that the host upon which DDS applications execute, is a *trusted zone*. The assumption that the hosts are a *trusted zone* does not hold. *Therefore, all versions of DDS ("secure" or not), are still vulnerable to physical attacks, compromised software development environment and compromised supply chain*.

There is also no 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 flaws.

5.1.3 Future Work

This research work only modelled, demonstrated and validated five of the 60 DDS security issues revealed in the analysis. Further work to model, demonstrate and validate the remaining 55 DDS Security Issues could result in more, and potentially more effective methods of compromising a DDS-based system. In addition, the process used in this research could also be applied to another middleware product which may not be DDS-based.

The new DDS Security Specification [5], and the associated RTI Connext DDS Secure™ [6] vendor implementation only cover five specific security areas: Authentication, Access Control, Encryption, Logging and Data Tagging. There is no published research that indicates its effectiveness in addressing any DDS security issues than originate from inside a DDS-based system. Application of the methods and processes used in this research work, could expose

security issues with this “secure” version of DDS. Specifically, application of the five self-contained demonstrations and the validation scenario to a DDS-based system built with a “secure” DDS vendor implementation could expose the inability of “secure” DDS to handle these types of DDS security issues.

In addition, the new DDS Security Specification [5] is also 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 security issues revealed in this research on a DDS-based system that has “secure” DDS entities communicating with legacy “non-secure” DDS Entities may have interesting results. It may show that these DDS security issues are not resolved or it may reveal new DDS security issues.

Further research is also needed to determine the security of different vendor implementations of DDS. Real-world evaluation and testing is 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. Without results of either of these activities (i.e. the ease of attack and the magnitude of the damage caused by the attack), it is unlikely that current operators of DDS-based systems will expend financial resources to upgrade their systems to the latest security standard or migrate from one DDS vendor implementation to another more secure DDS vendor implementation.

Intrusion Detection is another area in which more research and development needs to occur. Many commercial “signature-based” Intrusion Detection Systems (IDS) exist and research continues in this area. Since these systems can only detect security events which have been detected on similar systems, their effectiveness may be limited by the malware signatures encountered thus far in DDS-based systems. “Anomaly-based” IDSs may be more valuable and effective in DDS-based systems, as they may be able to detect contextual situations that DDS was not designed to prevent, but are dangerous nonetheless.

A larger issue that requires exploration is the question of how to handle a security violation in a DDS-based system once it occurs. Research and testing of particular deployed DDS-based systems is needed to reveal the potential severity of an attack and to develop an intelligent IDS capable of alerting the operator to the type of attack and its severity. This would give the

operator the information needed to determine whether to continue operating the system, or shut it down while dealing with an intrusion. Many DDS-based systems are mission-critical and safety-critical; any interruption in service could result in mission failures or loss of life. Alternatively, leaving the system in a compromised and infected state may also lead to the same severe situation. Therefore, more research and testing is required to assess the effect of a certain types of attacks and breaches in order to allow the operators of the system to make informed decisions about how to proceed with the use of the system.

5.2 Conclusion

This research revealed 60 DDS security issues through the analysis of the DDS protocol and several DDS vendor implementations, it modelled and demonstrated five of these DDS security issues in a self-contained and isolated environment, and validated their effectiveness in compromising DDS-based systems by combining them into an end-to-end attack scenario. This research demonstrated that these DDS security issues could be exploited by a malicious actor in a client-side attack on a DDS-based system, therefore validating the research aim of investigating DDS security issues and developing and validating DDS threat models that could be used by a malicious actor to compromise a host via a client-side attack on a DDS-based system.

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.

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] D. L. R. Layer, "Addressing the Challenges of Tactical Information Management in Net-Centric Systems With DDS," 2008.

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

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

[7] "The STRIDE Threat Model." [Online]. Available: https://msdn.microsoft.com/library/ms954176.aspx. [Accessed: 21-Jun-2016].

[8] 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.

[9] 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.

[10] G. Pardo-Castellote, "OMG Data-Distribution Service: architectural overview," in *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings*, 2003, pp. 200–206.

[11] "Data Distribution Service - Wikipedia, the free encyclopedia." [Online]. Available: https://en.wikipedia.org/wiki/Data_Distribution_Service. [Accessed: 01-Dec-2015].

[12] 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.

[13] J. Hugues, L. Pautet, and F. Kordon, "A framework for DRE middleware, an application to DDS," in *Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, 2006. ISORC 2006*, 2006, p. 8 pp.–.

[14] 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.

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

[16]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.

[17]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.

[18]I. Calvo, F. Perez, G. de Albeniz, and I. Etxeberria-Agiriano, “Towards a OMG DDS communication backbone for factory automation applications,” in *2011 IEEE 16th Conference on Emerging Technologies Factory Automation (ETFA)*, 2011, pp. 1–4.

[19]I. Etxeberria-Agiriano, I. Calvo, and F. Perez, “Providing soft real-time capabilities to business applications,” in *2012 7th Iberian Conference on Information Systems and Technologies (CISTI)*, 2012, pp. 1–6.

[20]P. Peniak and M. Franekova, “Open communication protocols for integration of embedded systems within Industry 4,” in *2015 International Conference on Applied Electronics (AE)*, 2015, pp. 181–184.

[21]“Data-Centric IoT Messaging.” [Online]. Available: http://www.eejournal.com/archives/articles/20140811-dds. [Accessed: 01-Dec-2015].

[22]B. Almadani *et al.*, “AVL and Monitoring for Massive Traffic Control System over DDS, AVL and Monitoring for Massive Traffic Control System over DDS,” *Mob. Inf. Syst. Mob. Inf. Syst.*, vol. 2015, 2015, p. e187548, Oct. 2015.

[23]F. Ben Cheikh, M. A. Mastouri, and S. Hasnaoui, “Implementing a Real-Time Middleware Based on DDS for the Cooperative Vehicle Infrastructure Systems,” in *2010 6th International Conference on Wireless and Mobile Communications (ICWMC)*, 2010, pp. 492–497.

[24]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.

[25]U. Lang and R. Schreiner, “Managing Security in Intelligent Transport Systems,” in *2015 IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, 2015, pp. 48–53.

[26]R. W. S. <rwshirey@uwalumni.com>, “Internet Security Glossary, Version 2.” [Online]. Available: https://tools.ietf.org/html/rfc4949. [Accessed: 28-Sep-2016].

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

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

[29]S. Hansman and R. Hunt, “A taxonomy of network and computer attacks,” *Comput. Secur.*, vol. 24, no. 1, pp. 31–43, Feb. 2005.

[30]S. D. Applegate and A. Stavrou, “Towards a Cyber Conflict Taxonomy,” in *2013 5th International Conference on Cyber Conflict (CyCon)*, 2013, pp. 1–18.

[31]“Threat Risk Modeling - OWASP.” [Online]. Available: https://www.owasp.org/index.php/Threat_Risk_Modeling. [Accessed: 21-Jun-2016].

[32]“Existing taxonomies — ENISA.” [Online]. Available: https://www.enisa.europa.eu/topics/csirt-cert-services/community-projects/existing-taxonomies. [Accessed: 21-Jun-2016].

[33]J. D. Howard, “An analysis of security incidents on the internet 1989-1995,” Carnegie Mellon University, 1997.

[34]G. Meng, Y. Liu, J. Zhang, A. Pokluda, and R. Boutaba, “Collaborative Security: A Survey and Taxonomy,” *ACM Comput Surv*, vol. 48, no. 1, pp. 1:1–1:42, Jul. 2015.

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

[36]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.

Appendix A Terminology

A.1 Terminology

The following definitions are used in this document. Many are based on definitions in the Internet Engineering Task Force (IETF) Glossary contained in Request For Change 4949 (RFC 4949) [26], but they have been tailored to reflect the subject of this thesis (i.e. DDS-based systems):

- **Application**: a program that interacts with the user and the *DDS Vendor Implementation*; which is effectively isolated from the *Network Infrastructure*.
- **Attack**: an intentional act by which an actor attempts to evade security services and violate the security policy of a *Host* or *DDS-Based System*.
- **Client-side Attack**: an *Attack* that is initiated by an actor from a position inside the system's security perimeter that employs techniques that compromise *DDS Entities* and hosts on *DDS-Based System.*
- **DDS-Based System**: a system using a combination of *Applications*, an existing *DDS Vendor Implementation* and a *Network Infrastructure*.
- **DDS Entities**: a logical functional block consisting of an *Application* and *DDS Vendor Implementation* libraries, that function as data publisher or data subscriber. *DDS Entities* include DCPS DataWriters and DCPS DataReaders and their corresponding RTPS Writers and RTPS Readers.
- **DDS Vendor Implementation**: the software used to develop a *DDS-based system* based on the DCPS (and optionally, the RTPS) specification(s).
- **Flaw**: an error in the design, implementation, configuration or operation and management of a *DDS-Based System* or *DDS Vendor Implementation*. A *Flaw* may result in a *Vulnerability*.
- **Host**: a computer that runs an *Application* and *DDS Vendor Implementation* libraries that implement *DDS Entities* and contains *Network Infrastructure* in order to communicate with other *DDS Entities* on the same *Host* or other *Hosts* (in a *DDS-Based System*).

- **Malicious Logic**: Any hardware, firmware, or software intentionally introduced into a *DDS-Based System* to perform an unauthorized function or to control execution of a service in an unauthorized manner (i.e. a subtype of *Misuse*).
- **Misuse**: the use of a component or a feature of a *DDS-Based System* for other than authorized purposes, that causes a system component to perform a function or service that is detrimental to system or data security (see *Malicious Logic* and *Tampering*).
- **Network Infrastructure**: Network services from the Network Layer to the Physical Layer in the OSI model. In the context of this thesis, this would include all network services that RTPS uses.
- **QoS**: Quality of Service parameters specified via configuration file or programmatically (in the source code), and expressed in RTPS messages, that affect the behavior of *DDS Entities* and communication flows.
- **Tampering**: deliberately altering a *DDS-Based System* logic, data, or configuration to cause the system to perform unauthorized functions or services (i.e. a subtype of *Misuse*).
- **Vulnerability**: a *Flaw* or weakness in a system's design, implementation, configuration or operation and management that could be exploited to violate the security policy of a *DDS-Based System*.

Appendix B DDS Security Taxonomy

B.1 Introduction

The analysis in section 3.4 used a new DDS Malware Taxonomy specifically developed for this research work, to ensure a focused and thorough analysis of security issues in each DDS feature. This new Malware Taxonomy was based on the work of Hansman and Hunt [29], Applegate and Stavrou [30], and the ENISA and OWASP web pages [31, 32]. The following subsections explain its development and provide more detail of how it is applicable to malware affecting DDS and other middleware products.

B.2 DDS Malware Taxonomy Origin and Development

Hansman and Hunt [29] proposed a four-dimension taxonomy to categorize malware threats. The first four dimensions were as follows:

- The **1st dimension**: covers the main *means by which the attack first reaches the target*;
- The **2nd dimension**: covers the *target(s) of the attack*;
- The **3rd dimension**: covers *flaws* and exploits that the attack uses; and
- The **4th dimension**: covers attacks that have payloads or *side-effects beyond the primary effect*.

The Hansman and Hunt [29] taxonomy has been customized for use in this thesis, in order to reflect the focus of on attacks on DDS and other middleware products. The number of dimensions has been expanded in order to more accurately categorize DDS malware types, methods and effects.

The new DDS Malware Taxonomy is as follows:

- The **1st dimension**: covers the main *means by which the attack first reaches the target*;
- The **2nd dimension**: covers the *target(s) of the attack*;
- The **3rd dimension**: covers *design, implementation and configuration flaws* that result in a security issue that an attacker might exploit as part of an attack;

- The **4th dimension**: covers *attack types* as determined by the primary effect of the security issue;
- The **5th dimension**: covers *attack methods* used by an attack (i.e. how the attack is done);
- The **6th dimension**: covers *information impacts (effects and consequences)* beyond the primary effect; and
- The **7th dimension** covers indications that could be used in the *detection* of DDS malware and DDS misuse.

The following subsections detail each of the seven dimensions in the DDS Malware Taxonomy.

B.3 DDS Malware Taxonomy 1st Dimension: Initial Attack Vector

The 1st dimension of the DDS Malware Taxonomy covers *the main means by which the attack first reaches the target*. Note that this research work focused on client-side attacks on a DDS-based system; we therefore *did not focus on this dimension*, but include it here for completeness.

The following list indicates potential methods in which a piece of DDS malware might be placed into a DDS-based system:

- Viruses;
- Worms;
- Buffer overflows;
- Denial of Service attacks from external sources;
- Network Attacks from an external source or network;
- Software access and modification:
    - Privilege Escalation;
    - Authentication Violation; and
    - Sabotage of software, including corruption of files, weaponization of software [32].
- Physical Attacks, including those occurring at the factory, software and hardware maintenance facilities, and at the actual deployed location:

    - Sabotage of HW: including corruption of firmware, weaponization of hardware and rewiring network connections [32];
    - Theft of hardware or backups (i.e. to "weaponize" the devices) [32];
    - Unauthorized physical access to premises (required for above) [32]; and
    - Coercion, extortion or corruption (required for access to the facility) [32].
- Password Attacks; and
- Information Gathering, including Reconnaissance and Sniffing.

B.4 DDS Malware Taxonomy 2nd Dimension: Attack Targets

The 2nd dimension of the DDS Malware Taxonomy covers *the targets(s) of the attack*. Note that this research work focused on DDS (DCPS and RTPS) as well as the host application, configuration mechanisms, transports and RTPS messages. This dimension would be useful in describing malware that target different portions of DDS or of other systems.

The following list indicates potential targets of an attack on any critical infrastructure system (i.e. not just DDS):

- Hardware:
    - network cabling; and
    - DIP switches, code plugs, etc.
- Application Software:
    - Server applications and services (including applications accessing DDS-DCPS Publisher middleware API);
    - User applications and services (including applications accessing DDS-DCPS Subscriber middleware API);
- Middleware:
    - the DDS-DCPS middleware services (including API presented to the applications);
    - the DDS-RTPS middleware services (used to access UDP/IP network layer).
- Host Operating Systems;

    - Libraries;
    - File systems; and
    - Command interpreters (shells).
- Network and Protocols:
    - TCP, UDP and IP;
    - DDS-DCPS (i.e. higher-level DDS protocol); and
    - DDS-RTPS (i.e. optional DDS wire protocol).

B.5 DDS Malware Taxonomy 3rd Dimension: DDS Flaws

The 3rd dimension of the DDS Malware Taxonomy covers *design, implementation and configuration flaws* that result in a security issue that an attacker might exploit as part of an attack. Howard [33] details three broad categories of vulnerabilities which also apply to DDS malware. These include *design, implementation and configuration flaws*.

The following list indicates *flaws* that result in a security issue that an attacker might exploit as part of an attack on a DDS-based system. These are enumerated for ease of cross-referencing in the DDS Security Analysis Template, described in Appendix B, section B.10:

- **(F1) Flaw in design**: The fundamental design is flawed, such that potential vulnerabilities would exist in a perfect implementation. Note that within the context of this research work, this corresponds to issues in the DDS 1.4 and RTPS 1.2 specifications [1, 2].
- **(F2) Flaw in implementation**: The fundamental design is secure, but it has been poorly implemented (i.e. poor toolset or libraries, or poor interpretation of design goals), and it introduces its own bugs and potential vulnerabilities. Note that within the context of this research work, this corresponds to issues with the actual OpenDDS or RTI Connext DDS™ middleware products.
- **(F3) Flaw in configuration**: The design and implementation are both secure, but it was incorrectly configured, therefore becoming vulnerable (e.g. open ports, backdoor, debugging interfaces, etc.). This may also include the ability to modify a secure configuration such that it becomes insecure. Within the context of this research work, this

corresponds to issues with the default DDS configurations and changes to "safe" configurations.

B.6 DDS Malware Taxonomy 4th Dimension: Attack Types

The 4th dimension covers *attack types* as determined by the primary effect of the DDS security issue. Note that the attack types are not mutually exclusive; they are often used concurrently in an attack scenario.

The following list indicates *attack types* that result from exploitation of a DDS security issue. These are enumerated for ease of cross-referencing in the DDS Security Analysis Template, described in Appendix B, section B.10:

- **(A1) Denial of Service (DoS) Attacks** (i.e. internal to the DDS network):
    - **(A1.1) Host-based DoS**:
        - Resource hogs or resource depletion; and
        - Crashers.
    - **(A1.2) Network-based DoS**:
        - TCP flooding;
        - UDP flooding; and
        - ICMP flooding.
    - **(A1.3) Distributed DoS**:
        - Use of multiple nodes to accomplish a network-based DoS attack.
- **(A2) Network Attacks**:
    - **(A2.1) Spoofing and Identity Theft**:
        - Existing DDS Domain Participant (publisher or subscriber); and
        - New DDS Domain Participant (publisher or subscriber).
    - **(A2.2) Session hijacking**
    - **(A2.3) Replay of messages**
    - **(A2.4) Network configuration corruption**:

        - Routing table corruption;
        - IP address corruption; and
        - DDS discovery configuration corruption.
    - **(A2.5) Routing Traps**: whereby a node that was supposed to transfer or forward packets does not function properly, therefore impeding or denying service to the associated nodes [34]:
        - **(A2.5.1) Sinkhole Attacks**: a node establishes itself as a relay node with goal of routing all traffic to itself and controls distribution or data flow (i.e. selectively stops communication flow, or manipulates data and forwards it);
        - **(A2.5.2) Blackhole Attacks**: a node establishes itself as a relay node and stop all traffic (i.e. zero packet forwarding); and
        - **(A2.5.3) Greyhole Attacks**: a node establishes itself as a relay node and stops all traffic (like a Blackhole), except that it still forwards packets involved in the establishment and maintenance of routes.
    - **(A3) Repudiation Attacks**: occur when a user performs a malicious action and other parties lack any method to prove the action was malicious [7]. Non-repudiation refers to the ability of a system to counter repudiation threats by receiving proof that the action was action was not malicious (note that very few non-repudiation techniques are effective – ACK/NACK methods are not sufficient [7]).
- **(A4) Data Attacks**:
    - **(A4.1) Data Deletion:**
        - Automatic or manual deletion of data; and
        - Causing data to not be published.
    - **(A4.2) Data Modification**:
        - Value manipulation;
        - Timestamp or sequence number manipulation; and
        - Change of update frequency or delay of transmission.

- **(A4.3) Data Insertion:**
    - Addition of new data with no alteration of existing data.
- **(A4.4) Data Redirection**:
    - Data sent to both the original destination and the new destination (i.e. snooping); or
    - Data only sent to the new destination (i.e. true redirection or data theft).
- **(A4.5) Data Omission**:
    - Data does not appear to the entity that requested it.

B.7 DDS Malware Taxonomy 5th Dimension: Attack Methods

The 5th dimension covers *attack methods* (i.e. how the attack is carried out). Note that the attack methods are not mutually exclusive; they are often used concurrently to exploit a security issue in an attack scenario.

The following list indicates *attack methods* used to exploit a DDS security issue. These are enumerated for ease of cross-referencing in the DDS Security Analysis Template (described in Appendix B, section B.10):

- **(M1) Hardware or Software Availability Attacks**:
    - **(M1.1) Outage**: Exploits the time in which no service is available (temporary or permanent); and
    - **(M1.2) Failover**: Exploits the mechanism to fail-over from one node to another compromised node.
- **(M2) Alteration of Application Software**:
    - Alteration or insertion of malicious logic in the application software that uses DDS services.
- **(M3) Alteration of Middleware (DDS)**:
    - Alteration or insertion of malicious logic in DDS libraries, in order to defeat DDS security measures.

- **(M4) Alteration of Configuration, File or File System**:
    - Alteration of configuration files; or
    - File System changes (including altered files, links, etc.).
- **(M5) Existing Deficiencies**:
    - **(M5.1) Bugs**: that could be exploited to perform malicious actions (i.e. in the DDS protocol, software, network, hardware, etc.); and
    - **(M5.2) Debugging interfaces and APIs**.
- **(M6) Alteration of Hardware**:
    - Defective or damaged hardware; and
    - Alteration of DIP switches, code plugs, etc.

B.8 DDS Malware Taxonomy 6th Dimension: Informational Impacts

The 6th dimension covers attacks that have *information impacts (effects and consequences)* beyond the primary effect. The data-centric paradigm upon which DDS is based, means that the informational impact of a DDS attack or compromise is the most important. Applegate and Stavrou [30], included this Informational Impact as a subcategory of their Attack Vector taxonomy. Hansman and Hunt [29] also include a dimension in their taxonomy (the 4th dimension in their taxonomy) that covers "attacks that have payloads or side-effects beyond the primary effect" [29]. In the context of this research work, and the application of this taxonomy, this 6th dimension covers the impact an intrusion has on the victim's information.

The following list indicates *information impacts (effects and consequences)*, of an exploitation of a DDS security issue. These are enumerated for ease of cross-referencing in the DDS Security Analysis Template (described in Appendix B, section B.10):

- **(D1) Denial of Information**: Preventing communication between legitimate users, which on a DDS system would involve preventing a publisher/subscriber from sending or receiving subscribed information on a domain, or preventing access to the domain;
- **(D2) Destruction of Information**: Illegitimate erasure of information, which on a DDS system would involve the permanent deletion of executable files, configuration files or historical or persistent data;

- **(D3) Disclosure of Information**: Illegitimate access to information for which the subscriber does not have access to or interception and or disclosure of sensitive information (i.e. publishing this information on a new DDS Topic, or transmission via covert channel);
- **(D4) Discovery of Information**: Gathering and storage of intelligence that could be extracted and analyzed for future use (potentially giving the attacker intelligence for future attacks or operations). This could include network configuration information or information on internal data structures; and
- **(D5) Distortion of Information**: Manipulation or corruption of information such that legitimate users may receive incorrect information without any indication that it is incorrect. This might include violation of data integrity.

B.9 DDS Malware Taxonomy 7th Dimension: Detection

The 7th dimension covers indications and methods of *detection* of misuse or exploitation of DDS security issues (which could also be DDS malware). Note that detection of misuse or exploitation of DDS security issues was out of scope of this research work, the analysis briefly explored methods of detection of misuse or exploitation of each DDS security issue to provide completeness in the analysis of each DDS feature and provides quick sanity-check of the validity of each DDS security issue.

The following list covers indications and methods of *detection* of misuse or exploitation of DDS security issues. These are enumerated for ease of cross-referencing in the DDS Security Analysis Template (described in Appendix B, section B.10):

- **(T1) Code or File Change**: Alteration of software source code, including make-files and header files (i.e. at build-time); and alteration of configuration files, scripts and any persistent method of affecting environment variables (i.e. at run-time);
- **(T2) Process Change**: A change in the run-time state of a process including new process/thread, ended process/thread or a change in performance of a process/thread or host node;

- **(T3) Message (Field) Change**: A change in a field or portion of a packet, message or submessage used for DDS communication (note that the analysis focused on RTPS messages, but it is not limited to those messages);
- **(T4) Communication Flow Change**: A change in the pattern or flow of packets, messages or submessages used for DDS communication (note that the analysis focused on RTPS messages, but detection would not be limited to those messages); and
- **(T5) Performance Change**: A change in performance of a process, thread or host node.

B.10 DDS Security Analysis Template

The following template was used in the analysis of each feature that is implicated in a particular DDS security issue.

A. Background (of this feature)
<description>

B. Issues (with this feature)
<description of the issue>

C. DDS Flaws (used by the attack)		
F1: Flaw in design	**F2: Flaw in implementation**	**F3: Flaw in configuration**
<elaboration of the DDS flaw>		

D. Attack Types		
A1: Denial of Service (DoS) Attacks (internal): - **A1.1**: Host-based - **A1.2**: Network-based - **A1.3**: Distributed	A2: Network Attacks: - **A2.1**: Spoofing/Identity theft - **A2.2**: Session hijacking - **A2.3**: Replay of messages - **A2.4**: Network configuration corruption - A2.5: Routing Traps: o **A2.5.1**: Sinkhole Attacks o **A2.5.2**: Blackhole Attacks o **A2.5.3**: Greyhole Attacks	**A3**: Repudiation Attacks A4: Data Attacks: - **A4.1**: Deletion - **A4.2**: Modification - **A4.3**: Insertion - **A4.4**: Redirection - **A4.5**: Omission
<elaboration of the attack type>		

E. Methods of Exploitation		
M1: HW or SW Availability Attacks - **M1.1**: Outage - **M1.2**: Failover **M2**: Alteration of Application SW	**M3**: Alteration of Middleware (DDS) **M4**: Alteration of Configuration, File and/or File System	M5: Existing Deficiencies: - **M5.1**: Bugs - **M5.2:** Debugging interfaces/APIs **M6**: Alteration of HW
<elaboration of the methods used to perform the attack>		

F. Informational Impacts (Effects and Consequences)		
D1: Denial of Information **D2:** Destruction of Information	**D3:** Disclosure of Information **D4:** Discovery of Information	**D5:** Distortion of Information
<elaboration of the effects of the attack>		

G. Detection		
T1: Code/File Change **T2:** Process Change	**T3:** Message/Field Change **T4:** Communication Flow Change	**T5:** Performance Change
<elaboration of potential detection methods>		

Appendix C Analysis of DDS Security Issues

C.1 Introduction

Analysis of DDS security issues resulted in 60 separate DDS security issues. Each DDS security issue was analyzed and a DDS Security Analysis form (see section B.10), was filled-out for each one. Note that only the forms for the five DDS security issues that were demonstrated in a self-contained and isolated environment and validated, were included in this thesis (see Appendix F).

The following subsections summarize the 60 DDS security issues that are detailed in [36].

C.2 DCPS Protocol and API Issues

The following sixteen DCPS Protocol and API issues listed below were discovered in the analysis:

- Misuse of Listener Queues to Ignore Events;
- Modification of Conditions & Waitsets Expressions to Ignore Events;
- Misuse of Waitset Queue to Ignore Conditions;
- Misuse of Duplicate Conditions & Waitsets to Change Event Responses;
- Misuse of the Ignore DDS Entity API Method;
- Misuse of DDS Entity Mapping and Discovery API Methods;
- Misuse of the Delete All DDS Entities Method;
- Misuse of the Top-Level (Domain Participant) Default Listener;
- Misuse of Anonymous Subscribe and Republish Functionality;
- Modification of Content Filtered Topic Expressions;
- Modification of Multi-Topic Expressions;
- Access and Manipulation of Built-in Topics;
- Misuse of the `unregister()` vs `dispose()` API Methods in Redundant Publication of Data Instances;

- Misuse of the `unregister()` vs `dispose()` API Methods in Single Ownership of Data Instances;
- Modification of Conditional Read Expressions in Subscription Specifications; and
- Misuse of Zero-Copy Read Functionality and Un-returned Loans.

C.3 DCPS QoS Policy Issues

The following eleven DCPS QoS Policy issues listed below were discovered in the analysis:

- Modification of Individual QoS Policies After Entity Creation;
- Modification of a Profile That Contains QoS Policies, After Entity Creation;
- Modification of QoS Policy Default Values Before Entity Creation;
- Misuse of `DEADLINE` and `TIME_BASED_FILTER` QoS Policies Causing DoS;
- Modification of `PARTITION` QoS Policy and non-Repudiation;
- Misuse of `RELIABILITY` and `RESOURCE_LIMITS` DCPS QoS Policies causing DoS;
- Misuse of `LIFESPAN` QoS Policy Causing Immediate Data Expiration;
- Misuse of `ENTITY_FACTORY` QoS Policy Causing Creation of Disabled Entities;
- Misuse of `OWNERSHIP_KIND` QoS Policy and `SHARED` Data Ownership;
- Misuse of `OWNERSHIP_STRENGTH` QoS Policy and `EXCLUSIVE` Data Ownership; and
- Abuse of `EXCLUSIVE` Data Ownership and Host Failure.

C.4 DCPS Vendor-Implementation Issues

The following eight DCPS vendor-specific implementation issues listed below were discovered in the analysis:

- Misuse of OpenDDS Recorder and Replayer Debugging API;
- Misuse of OpenDDS Persistence Service External Product and Process;

- Modification of RTI Connext DDS™ `DISCOVERY` QoS Policy and Endpoint Authentication;
- Misuse of RTI Connext DDS™ Recorder and Replayer Debugging API;
- Misuse of RTI Connext DDS™ Persistence Service External Product and Process;
- Misuse of RTI Connext DDS™ Request-Reply Pattern and Wiretap;
- Modification of RTI Connext DDS™ DataWriter `MULTI_CHANNEL` QoS Policy; and
- Modification of RTI Connext DDS™ `PROPERTY` QoS Policy.

C.5 RTPS Protocol Issues

The following four RTPS Protocol issues listed below were discovered in the analysis:

- Misuse of RTPS Version Configuration Attributes and Host Miscommunication;
- Malicious Manipulation of RTPS History Cache Structure;
- Misuse of RTPS Global Unique Identifier (GUID) Configuration Attribute and Communication Path Alteration; and
- Misuse of RTPS `Locator` Attribute and Communication Path Alteration.

C.6 RTPS Message Issues

The following four RTPS Message issues listed below were discovered in the analysis:

- Manipulation of the `SubmessageLength` Field in RTPS Messages and Data Insertion or Loss;
- Manipulation of the `SubmessageElement` Structure in RTPS Messages and Data Insertion or Loss;
- Misuse of the `ParameterList` in RTPS Messages and Data Insertion;
- Misuse of the `Gap` Submessage in RTPS Messages and Data Integrity Violation;
- Changes to Submessage Combination and Optimization in RTPS Messages and Indications of Malicious Activity; and

- Manipulation of Inline Coherent Set Parameters in an RTPS Message Causing Data Loss.

C.7 RTPS Discovery Mechanism Issues

The following six RTPS Discovery Mechanism issues listed below were discovered in the analysis:

- Misuse of the `LocatorList` Environment Variables Causing Domain Misdirection During Participant Discovery;
- Misuse of `LocatorList` Fields in RTPS Messages Causing Participant Misdirection During Participant Discovery;
- Misuse of the `defaultLocatorList` Parameters used for Endpoint Discovery Causing Participant Misdirection During Endpoint Discovery;
- Disruption of Endpoint Discovery Messages and Non-Repudiation; and
- Misuse of Alternate Discovery Protocols Causing Alternate Communication Paths.

C.8 RTPS Vendor-Implementation Issues

The following three RTPS Vendor-specific implementation issues listed below were discovered in the analysis:

- Misuse of OpenDDS Configuration File to Manipulate Discovery Protocols;
- Alteration of the OpenDDS `InfoRepo` Centralized Discovery Mechanism to Disrupt or Redirect New Communication Flows; and
- Misuse of RTI Connext DDS™ `DISCOVERY` QoS Policies to Disrupt or Redirect New Communication Flows.

C.9 DDS Transport Mechanism Issues

The following three DDS Transport Mechanism issues listed below were discovered in the analysis:

- Misuse of Multiple RTPS Channels Causing Communication Flow Redirection or Covert Channels;

- Misuse of Multiple OpenDDS Mixed Transport Mechanisms Causing Communication Flow Redirection or Covert Channels; and
- Misuse of Multiple RTI Connext DDS™ Mixed Transport Mechanisms Causing Communication Flow Redirection or Covert Channels.

C.10 DDS Configuration Issues

The following 5 DDS Configuration issues listed below were discovered in the analysis:

- Misuse of OpenDDS Compliance Profile at Build Time Causing System Instability;
- Misuse of OpenDDS Safety Profile at Build Time Allowing Potentially Unsafe Functionality;
- Misuse of OpenDDS Run-Time Configuration File to Specify Unsafe Parameters or Behaviour;
- Misuse of RTI Connext DDS™ Run-Time Configuration File to Specify Unsafe Parameters, QoS Settings or Behaviour; and
- Misuse of RTI Connext DDS™ Run-Time Configuration File Load Order to Specify a New Configuration File.

Appendix D Tools and Infrastructure

D.1 Tools and Infrastructure

This research used the following tools to perform the analysis, modelling and self-contained demonstration of the DDS security issues, and execute the end-to-end validation scenario:

- OpenDDS 3.8: DDS libraries (including source code), tools and documentation. Note that OpenDDS was not used for the self-contained demonstration, nor was it used for the end-to-end validation scenario;
- RTI Connext DDS™ 5.2.0: DDS libraries (precompiled), tools and documentation;
- RTI Vehicle Tracking Sample Application [35];
- Ubuntu 16.04 (host environment for both DDS vendor implementations); and
- Parallels 11.2.1 (to virtualize the network environment).

The nature of DDS and its small footprint allowed research and modelling to be done using 2-10 VMs running on a MacBook containing an Intel core i5 with 8GB memory.

Appendix E Selection of DDS Issues Used for the Self-Contained Demonstration and the Validation Scenario

E.1 Introduction

The analysis of DDS resulted in the discovery of 60 DDS security issues, as recorded in [36]. Many of these DDS security issues shared common DDS Flaws, Attack Types and Exploitation Techniques (i.e. the 3rd, 4th and 5th dimensions of the DDS Malware Taxonomy respectively). These DDS security issues also covered a very broad range of DDS features and functionality including implementation/vendor-specific DDS features and nuances.

For the purposes of keeping this research work within the time-frame of a Master's Thesis, the number of DDS security issues selected for modelling and self-contained demonstration and the end-to-end validation scenario was reduced to five – an additional two issues were used by the five DDS issues, but they weren't modelled, demonstrated or validated on their own. These five (plus two) DDS security issues covered many of the attack types, DDS features and vendor-specific features discovered in the analysis.

E.2 Selection of DDS Issues

Selection of DDS security issues for modelling and self-contained demonstration ensured coverage of a broad set of Attack Types (as indicated in the 4th dimension of the DDS Malware Taxonomy). These attack types included examples of DoS; Hijacking and Identity theft; Network configuration; and Data Deletion, Modification, Insertion, Redirection and Omission. In addition, the selected DDS security issues covered a broad set of DDS functionality, including: DCPS functionality (i.e. publish-subscribe and QoS functionality); RTPS functionality (including messages); Discovery mechanisms; and implementation/vendor-specific features (including configuration methods).

The selection of specific DDS security issues for modelling and self-contained demonstration was refined to also include those issues that would fit into the end-to-end validation scenario (described in section 4.2). Table 3 illustrates coverage of the DDS security

issues selected for modelling and self-contained demonstration based on their Attack Type, DDS Feature Coverage and Exploitation method.

Table 3: DDS Security Issue Selection Matrix

#	DDS Issue	Attack Type	Feature Coverage	Exploitation Method
1	Misuse of Anonymous Subscribe and Republish Functionality	Data Attack: - Data Insertion - Data Modification DoS Attack: - Host-based DoS	- DDS-DCPS core functionality and API	- New DDS Entity created from a new DDS application
2	Misuse of `OWNERSHIP_STRENGTH` QoS Policy and `EXCLUSIVE` Data Ownership	Data Attack: - Data Deletion - Data Insertion (i.e. Data Hijacking)	- DDS-DCPS core functionality and API	- New DDS Entity created from an existing DDS application
3	Misuse of `OWNERSHIP_KIND` QoS Policy and `SHARED` Data Ownership	Data Attack: - Data Redirection	- DDS-DCPS core functionality and API - DDS Configuration	- Configuration change - Restart DDS application
4	Misuse of `LIFESPAN` QoS Policy Causing Immediate Data Expiration	Data Attack: - Data Deletion - Data Omission	- DDS-DCPS core functionality and API - DDS Configuration	- Configuration change - Restart DDS application
5	Misuse of the `LocatorList` Environment Variable Causing Domain Misdirection During Participant Discovery	Network Attack: - Network Configuration Corruption (i.e. Host Hijacking)	- RTPS Functionality - DDS Discovery Mechanisms - DDS Configuration	- Configuration change - Restart DDS application - New DDS Entity created from an existing DDS application

#	DDS Issue	Attack Type	Feature Coverage	Exploitation Method
6	Misuse of RTI Connext DDS™ Run-Time Configuration File to Specify Unsafe Parameters, QoS Settings or Behaviour	*This supports the attacks specified in DDS Issues #3, 4 and 5, and therefore was not used on its own.*	- DDS-DCPS core functionality - RTPS Functionality - RTPS Discovery - DDS Configuration	- Configuration file change - Environment variable change
7	Misuse of RTI Connext DDS™ Run-Time Configuration File Load Order to Specify a New Configuration File	*This supports the attacks specified in DDS Issues #3, 4 and 5, and therefore was not used on its own.*	- DDS Configuration	- Configuration file change

Appendix F Analysis of DDS Issues Used for the and Self-contained Demonstration and the Validation Scenario

F.1 Introduction

Analysis of DDS security issues started with understanding specific features of DDS, and how flaws in these features could lead to a potential security issue. Analysis continued with the type of attack that exploitation of the flaw could produce, followed by the methods in which the attack could be done. Finally, the effects of exploitation of the DDS security issue were examined, including the determination if any of these effects could lead to the detection of the DDS security issue.

F.2 Analysis of DDS Issues

To ensure thorough and focused analysis of each DDS feature, the custom DDS Analysis Taxonomy was used to guide the analysis. The analysis results were recorded using the DDS Security Analysis Template, (included in Appendix B, section B.10).

The following subsections detail the analysis of each of DDS security issues that were selected for modelling and self-contained demonstration and used in the end-to-end attack scenario. The highlighted items in the forms provide a quick method of identifying the key findings of each section of the analysis.

Note that two more DDS security issues were analyzed and listed below, but they were not individually modelled or demonstrated in a self-contained environment because they are employed by the previous five DDS security issues.

F.3 DDS Security Issue #1: Misuse of Anonymous Subscribe and Republish Functionality

A. Background (of this feature)
The DCPS model for publish and subscribe allows for any entity to publish Instances/Samples of a Topic onto a Domain allowing any entity to subscribe to these Instances/Samples of a Topic on that Domain. The only requirements are that the Domain ID, Topic Name, Data Type and QoS match. Therefore, if the QoS contract is fulfilled (which may not always be the case due to data ownership or data origin requirements), then anything can be recorded, altered and then replayed.

B. Issues (with this feature)
An application could be built with out-of-the-box DCPS methods, which would allow any one of the following to occur: • Recording data for retransmission on a covert channel (snooping); • Recording data and blindly replaying it on the same domain (DoS due to network traffic or extra processing by DataReaders); or • Recording data, altering the data (to various degrees), and then replaying it (to cause confusion, etc.); or • Recording data, make copies of it (with slight differences) and then replaying it (could cause confusion, operator overload etc.). The key difference between this and the OpenDDS and RTI Connext DDS™ Recorder and Replayer features, is that in this case, knowledge of the data structure is required, QoS must match absolutely, and is therefore very specific, obtrusive and requires more knowledge and programming effort to implement. Alternatively, analysis via Wireshark could be used to provide the same results.

C. DDS Flaws (used by the attack)		
F1: Flaw in design	**F2**: Flaw in implementation	**F3**: Flaw in configuration
This is a flaw in design, as this functionality is fully conformant to the DDS DCPS specification, and is built into all DCPS-compliant implementations.		

D. Attack Types		

A1: Denial of Service (DoS) Attacks (internal):
- **A1.1**: Host-based
- **A1.2**: Network-based
- **A1.3**: Distributed
	A2: Network Attacks:
	- **A2.1**: Spoofing/Identity theft
	- **A2.2**: Session hijacking
	- **A2.3**: Replay of messages
	- **A2.4**: Network configuration corruption
	- A2.5: Routing Traps:
	  o **A2.5.1**: Sinkhole Attacks
	  o **A2.5.2**: Blackhole Attacks
	  o **A2.5.3**: Greyhole Attacks
		A3: Repudiation Attacks
		
		A4: Data Attacks:
		- **A4.1**: Deletion
		- **A4.2**: Modification
		- **A4.3**: Insertion
		- **A4.4**: Redirection
		- **A4.5**: Omission

This type of attack can be classified as: • DoS (of all forms) depending on how much data was replayed on the network (based on network capacity), how much data was received by subscribing applications (host overloading), and playback could be done at multiple hosts (leading to a Distributed DoS); • Data Modification and Insertion attack, due to the ability to alter data, or insert new data; or • Redirection by recording data and replaying over another channel (covert).		

E. Methods of Exploitation		
M1: HW or SW Availability Attacks - **M1.1**: Outage - **M1.2**: Failover **M2**: Alteration of Application SW	**M3**: Alteration of Middleware (DDS) **M4**: Alteration of Configuration, File and/or File System	M5: Existing Deficiencies: - **M5.1**: Bugs - **M5.2:** Debugging interfaces/APIs **M6**: Alteration of HW
This flaw involves altering the application software.		

F. Informational Impacts (Effects and Consequences)		
D1: Denial of Information **D2:** Destruction of Information	**D3:** Disclosure of Information **D4:** Discovery of Information	**D5:** Distortion of Information

A DoS effect could prevent an application from receiving and processing all (or any) of the data required for correct operation.

Subscribed data could be replayed onto another channel, leading to disclosure of information.

A publication of altered or new data (based on original data) could force an application to use similar but not correct information and cause confusion or incorrect decisions and processing.

G. Detection		
T1: Code/File Change **T2:** Process Change	**T3:** Message/Field Change **T4:** Communication Flow Change	**T5:** Performance Change

This could be detected via:

- a tool which detects file (executable) changes;
- QoS contract changes communicated a part of DATA (p), DATA(r) and DATA(w) messages. As well, the new DataReader and new DataWriter entities would be indicated in these messages;
- a new communication flow between DataReaders and DataWriters which may not normally be associated with this Topic or Partition; and
- discovery of a covert channel (i.e. perhaps via discovery of a new domain or transport).

F.4 DDS Security Issue #2: Misuse of OWNERSHIP_STRENGTH QoS Policy and EXCLUSIVE Data Ownership

A. Background (of this feature)

When the OWNERSHIP_KIND is EXCLUSIVE ownership, the OWNERSHIP_STRENGTH QoS attribute is used to determine which (out of multiple) DataWriter objects can update the same Instance (i.e. publish Samples of the Instance).

Ownership of an Instance can change due to:

- a new DataWriter with a higher OWNERSHIP_STRENGTH attempts to modify the Instance;
- a change in OWNERSHIP_STRENGTH of an existing DataWriter that owns the Instance;
- a change in LIVELINESS of an existing DataWriter that owns the Instance; or
- a missed deadline (i.e. based on the DEADLINE QoS), for the DataWriter that owns the instance.

Note 1: There is no error or notification given to a DataWriter that modifies an instance it does not currently own.

Note 2: If multiple DataWriters with the same OWNERSHIP_STRENGTH attempt to modify the same Instance, only one will modify it. The choice is vendor implementation-specific (i.e. not specified by the DDS Specification [1, 2]).

B. Issues (with this feature)

The OWNERSHIP_STRENGTH of each Instance can be changed at any time. Therefore, any DataWriter with a higher OWNERSHIP_STRENGTH could gain ownership and publish samples. This could be done by raising the OWNERSHIP_STRENGTH of a malicious DataWriter, or by reducing the OWNERSHIP_STRENGTH of the existing owner DataWriter.

C. DDS Flaws (used by the attack)		
F1: Flaw in design	**F2**: Flaw in implementation	**F3**: Flaw in configuration

The ability to change the `OWNERSHIP_STRENGTH` via configuration file at startup and at any time during execution is a configuration flaw. The fact that this post-compilation change could change fundamental behavior of compiled code makes this a design flaw.

D. Attack Types		

A1: Denial of Service (DoS) Attacks (internal):
- **A1.1**: Host-based
- **A1.2**: Network-based
- **A1.3**: Distributed
	A2: Network Attacks:
	- **A2.1**: Spoofing/Identity theft
	- **A2.2**: Session hijacking
	- **A2.3**: Replay of messages
	- **A2.4**: Network configuration corruption
	- A2.5: Routing Traps:
	  o **A2.5.1**: Sinkhole Attacks
	  o **A2.5.2**: Blackhole Attacks
	  o **A2.5.3**: Greyhole Attacks
		A3: Repudiation Attacks
		
		A4: Data Attacks:
		- **A4.1**: Deletion
		- **A4.2**: Modification
		- **A4.3**: Insertion
		- **A4.4**: Redirection
		- **A4.5**: Omission

This data attack could cause:

- New and potentially harmful incorrect data to be sent to the subscribing application, and the "real" data would never be published (eventually deleted by the "real" DataWriter); or
- If a "fast" DataWriter is involved and publishes a lot of Samples, DataReaders could be overwhelmed by the amount of data and slowdown or fail.

E. Methods of Exploitation		
M1: HW or SW Availability Attacks - **M1.1**: Outage - **M1.2**: Failover **M2**: Alteration of Application SW	**M3**: Alteration of Middleware (DDS) **M4**: Alteration of Configuration, File and/or File System	M5: Existing Deficiencies: - **M5.1**: Bugs - **M5.2:** Debugging interfaces/APIs **M6**: Alteration of HW

This involves:

- Alteration of `OWNERSHIP_STRENGTH` QoS configuration either by default QoS settings (in configuration file or source-code); or
- Programmatically altering the `OWNERSHIP_STRENGTH` QoS (i.e. malicious code).

F. Informational Impacts (Effects and Consequences)		
D1: Denial of Information	**D3:** Disclosure of Information	**D5:** Distortion of Information
D2: Destruction of Information	**D4:** Discovery of Information	

This would prevent an application from receiving and processing "real" valid data required by operations, the "real" owner no longer "owns" the Instance.

G. Detection		
T1: Code/File Change	**T3:** Message/Field Change	**T5:** Performance Change
T2: Process Change	**T4:** Communication Flow Change	

This could be detected via:

- a tool which detects file (executable or configuration file) changes; or
- QoS contract changes communicated a part of DATA(w) and DATA(r) messages.

It could also be detected by:

- A lack of messages between the "real" DataWriter and existing DataReader(s); and
- An abundance of new messages between the new DataWriter and existing DataReader(s) – most notably the existence of new communication flows between Publisher and Subscriber that do not normally have communication flows.

F.5 DDS Security Issue #3: Misuse of OWNERSHIP_KIND QoS Policy and SHARED Data Ownership

A. Background (of this feature)
The OWNERSHIP_KIND QoS policy is used to control whether multiple DataWriter objects can update the same Instance (i.e. publish Samples of the Instance). If the kind is SHARED ownership, there is no unique ownership of the Instance, and multiple DataWriters can update the Instance. If the kind is EXCLUSIVE ownership, only one DataWriter can update the Instance, which is decided by the OWNERSHIP_STRENGTH, LIVELINESS and DEADLINE QoS Policies (contract) between DataWriter and DataReaders.

B. Issues (with this feature)
A change from EXCLUSIVE ownership to SHARED ownership could allow any DataWriter to publish samples (even if they have a lower OWNERSHIP_STRENGTH, which would have disqualified them previously). This could allow a malicious entity to publish incorrect data. In addition, many malicious DataWriters could publish a lot of Samples in a distributed DoS attack.

C. DDS Flaws (used by the attack)		
F1: Flaw in design	**F2**: Flaw in implementation	**F3**: Flaw in configuration
The ability to change the OWNERSHIP_KIND via configuration file at startup is a configuration flaw. The fact that this post-compilation change could change fundamental behavior of compiled code (i.e. allow invalid publication), makes this a design flaw.		

D. Attack Types

A1: Denial of Service (DoS) Attacks (internal):
- **A1.1**: Host-based
- **A1.2**: Network-based
- **A1.3**: Distributed
	A2: Network Attacks:
	- **A2.1**: Spoofing/Identity theft
	- **A2.2**: Session hijacking
	- **A2.3**: Replay of messages
	- **A2.4**: Network configuration corruption
	- A2.5: Routing Traps:
	  o **A2.5.1**: Sinkhole Attacks
	  o **A2.5.2**: Blackhole Attacks
	  o **A2.5.3**: Greyhole Attacks
		A3: Repudiation Attacks
		
		A4: Data Attacks:
		- **A4.1**: Deletion
		- **A4.2**: Modification
		- **A4.3**: Insertion
		- **A4.4**: Redirection
		- **A4.5**: Omission

This data attack could cause:

- New and potentially harmful incorrect data to be sent to the subscribing application, obscuring the "real" data; or
- If multiple DataWriters are involved and each publishes a lot of Samples, DataReaders could be overwhelmed by the amount of data and slowdown or fail.

E. Methods of Exploitation

M1: HW or SW Availability Attacks - **M1.1**: Outage - **M1.2**: Failover **M2**: Alteration of Application SW	**M3**: Alteration of Middleware (DDS) **M4**: Alteration of Configuration, File and/or File System	M5: Existing Deficiencies: - **M5.1**: Bugs - **M5.2:** Debugging interfaces/APIs **M6**: Alteration of HW

This involves:

- Alteration of `OWNERSHIP` QoS configuration either by default QoS settings (via altered configuration file or source-code); or
- Programmatically altering the `OWNERSHIP` QoS (i.e. malicious code).

F. Informational Impacts (Effects and Consequences)

D1: Denial of Information **D2:** Destruction of Information	**D3:** Disclosure of Information **D4:** Discovery of Information	**D5:** Distortion of Information

This could prevent an application from receiving and processing "real" valid data required by operations, because of the number of invalid Samples.

If multiple DataWriters are involved and each publishes a lot of Samples, DataReaders could be overwhelmed by the amount of data and slowdown or fail.

G. Detection		
T1: Code/File Change **T2:** Process Change	**T3:** Message/Field Change **T4:** Communication Flow Change	**T5:** Performance Change

This could be detected via:

- A tool which detects file (executable or configuration file) changes; or
- QoS contract changes communicated a part of DATA(p) messages.

This could also be detected by an abundance of new messages between new DataWriter(s) and existing DataReader(s) – most notably the existence of new communication flows between publisher and subscriber that do not normally have communication flows.

F.6 DDS Security Issue #4: Misuse of LIFESPAN QoS Policy Causing Data Deletion

A. Background (of this feature)

A data Sample is valid as long as the LIFESPAN duration QoS attribute for the Topic or Instance has not been exceeded. A data Sample's "life" starts when it is added to the DataWriter's history cache, and the DataWriter or DataReader will delete it from the history cache if the LIFESPAN is exceeded.

The LIFESPAN duration QoS attribute is set by the DataWriter for the Topic or Instance, and can be changed at any time, and values can range from 1 nanosecond to 1 year, or DDS_INFINITE.

B. Issues (with this feature)

If the LIFESPAN duration QoS attribute was modified to the minimum value (1 nanosecond), for a Topic or Instance, samples could potentially "expire" immediately. They would be automatically deleted before transmission by the DataWriter or before they are able to be read from the DataReader cache by the subscribing application. This would have the effect of deleting data (permanently) prior to transmission.

C. DDS Flaws (used by the attack)

F1: Flaw in design	**F2**: Flaw in implementation	**F3**: Flaw in configuration

This is primarily a flaw in configuration, as DDS requires this flexibility in the design (i.e. to account for really fast systems with samples of very short LIFESPAN). This QoS attribute can be changed at any time; it could be specified at compile-time or run-time (programmatically or via configuration file) or during execution.

D. Attack Types		
A1: Denial of Service (DoS) Attacks (internal): - **A1.1**: Host-based - **A1.2**: Network-based - **A1.3**: Distributed	A2: Network Attacks: - **A2.1**: Spoofing/Identity theft - **A2.2**: Session hijacking - **A2.3**: Replay of messages - **A2.4**: Network configuration corruption - A2.5: Routing Traps: o **A2.5.1**: Sinkhole Attacks o **A2.5.2**: Blackhole Attacks o **A2.5.3**: Greyhole Attacks	**A3**: Repudiation Attacks A4: Data Attacks: - **A4.1**: Deletion - **A4.2**: Modification - **A4.3**: Insertion - **A4.4**: Redirection - **A4.5**: Omission
This data attack could cause required data to never get to the subscriber and the data would be lost permanently.		

F. Informational Impacts (Effects and Consequences)		
M1: HW or SW Availability Attacks - **M1.1**: Outage - **M1.2**: Failover **M2**: Alteration of Application SW	**M3**: Alteration of Middleware (DDS) **M4**: Alteration of Configuration, File and/or File System	M5: Existing Deficiencies: - **M5.1**: Bugs - **M5.2:** Debugging interfaces/APIs **M6**: Alteration of HW
This involves: • Alteration of `LIFESPAN` duration QoS attribute either by default QoS settings (in configuration file or source-code); or • Programmatically altering the `LIFESPAN` QoS (i.e. malicious code); or		

F. Effects		
D1: Denial of Information **D2:** Destruction of Information	**D3:** Disclosure of Information **D4:** Discovery of Information	**D5:** Distortion of Information
This could cause a publishing application with a maliciously altered configuration, to prevent new data from being received by other DataReaders – the data would "expire" and be deleted immediately by the DataWriter or DataReaders.		

G. Detection		
T1: Code/File Change **T2:** Process Change	**T3:** Message/Field Change **T4:** Communication Flow Change	**T5:** Performance Change

This could be detected via:

- A tool which detects file (executable or configuration file) changes;
- QoS contract changes communicated a part of DATA(r) and DATA(w) messages; or
- A decrease in messages between a DataWriter and DataReaders normally associated with this Topic or Instance.

F.7 DDS Security Issue #5: Misuse of the `LocatorList` Environment Variable Causing Domain Misdirection During Participant Discovery

A. Background (of this feature)

When the initial Participant Discovery process (i.e. SPDP) sends the initial DATA(p) messages on the network to announce the presence of a new DomainParticipant, it does so using pre-configured and pre-specified addresses and ports. These defaults are specified in configuration files or environment variables corresponding to the following (vendor implementation-dependent):

	Entity Locators
`SPDP using Unicast`	*`SPDPbuiltinParticipantReader.unicastLocatorList`* *`SPDPbuiltinParticipantWriter.readerLocators`*.
`SPDP using Multicast`	*`SPDPbuiltinParticipantReader.multicastLocatorList`* *`SPDPbuiltinParticipantWriter.readerLocators`*.

These default locators are used by the local `SPDPbuiltinParticipantWriter` Built-in Endpoint to advertise itself, and a remote `SPDPbuiltinParticipantReader` Built-in Endpoint to listen for the advertisements.

Note that in the RTI Connext DDS™ vendor implementation, the name for the `LocatorList` environment variables is changed to "`NDDS_DISCOVERY_PEERS`".

After this step, the actual discovered locators used by the rest of the discovery process (i.e. the remainder of the SPDP process and the SEDP process) are stored in the following data structures of the Built-in Endpoints:

- `metatrafficUnicastLocatorList`
- `metatrafficMulticastLocatorList`

B. Issues (with this feature)

Changes to these parameters (via environment variable or configuration file), could be done to either:

- Disrupt Participant (SPDP) discovery, leading to communication loss;
- Hijacking newly discovered Participants onto a separate Domain; or
- Enable a covert channel or Domain.

C. DDS Flaws (used by the attack)

F1: Flaw in design	**F2**: Flaw in implementation	**F3**: Flaw in configuration

These parameters can be set easily, which by design, simplifies and speeds up DDS-based system development and configuration, but they present an easily exploitable flaw in design and configuration.

D. Attack Types

A1: Denial of Service (DoS) Attacks (internal): - **A1.1**: Host-based - **A1.2**: Network-based - **A1.3**: Distributed	A2: Network Attacks: - **A2.1**: Spoofing/Identity theft - **A2.2**: Session hijacking - **A2.3**: Replay of messages **A2.4**: Network configuration corruption - A2.5: Routing Traps: ○ **A2.5.1**: Sinkhole Attacks ○ **A2.5.2**: Blackhole Attacks ○ **A2.5.3**: Greyhole Attacks	**A3**: Repudiation Attacks A4: Data Attacks: - **A4.1**: Deletion - **A4.2**: Modification - **A4.3**: Insertion - **A4.4**: Redirection - **A4.5**: Omission

This attack could cause DDS Network Configuration corruption, causing:

- Participants being isolated from an existing DDS-based system;
- Late-discovered Participants being redirected/hijacked onto a new DDS-based system;
- Routing Traps which could rewrite or redirect the remainder of the SPDP and SEDP mechanisms resulting in Sinkholes, Blackholes and Greyholes; and
- RTPS might not be able to communicate with other RTPS entities (partially or entirely), leading to data loss (deletion or omission).

Note that depending on the sophistication of the attack and combination of other security issues, all types of data attack are possible if this DDS security issue is exploited.

E. Methods of Exploitation		
M1: HW or SW Availability Attacks - **M1.1**: Outage - **M1.2**: Failover **M2**: Alteration of Application SW	**M3**: Alteration of Middleware (DDS) **M4**: Alteration of Configuration, File and/or File System	M5: Existing Deficiencies: - **M5.1**: Bugs - **M5.2:** Debugging interfaces/APIs **M6**: Alteration of HW
This involves changing the configuration file or environment variable prior to the startup of the DomainParticipant, or prior to restarting the process or host that runs the DomainParticipant DDS process.		

F. Informational Impacts (Effects and Consequences)		
D1: Denial of Information **D2:** Destruction of Information	**D3:** Disclosure of Information **D4:** Discovery of Information	**D5:** Distortion of Information
This could prevent an application from sending, receiving and processing data required by the for the DDS-based system to function. This could also lead to establishment of an alternate DDS-based system (i.e. a covert channel or an entire covert system).		

G. Detection		
T1: Code/File Change **T2:** Process Change	**T3:** Message/Field Change **T4:** Communication Flow Change	**T5:** Performance Change
This could be discovered by a tool that detects changes to configuration files and scripts (i.e. within DDS software or in the operating system files that control environment variables). Detection of a new or different locators in RTPS messages could be used to detect new malicious RTPS entities on the network. It could also be detected by a new RTPS communication flow that is not used by the current DDS-based system (i.e. covert channel or new DDS-based system).		

F.8 DDS Security Issue #6: Misuse of RTI Connext DDS™ Run-Time Configuration File to Specify Unsafe Parameters, QoS Settings or Behaviour

A. Background (of this feature)

RTI Connext DDS™ provides several mechanisms to configure the run-time behavior of a DDS-based system:

- **Configuration File**: global options, discovery, transport and options applicable to specific publishers and subscribers;
- **Command Line**: limited executable options to control interaction with the host operating system or host application, including specification of the "`.xml`" configuration file to read; and
- **Debugging API**: e.g. The DDS Monitor feature.

This analysis will focus on human-readable "`.xml`" configuration file(s). They contain xml strings that specify global settings, discovery settings, transport settings, and DDS Entity QoS Policy values. In RTI Connext DDS™, configuration settings use the QoS mechanism to set their values and store them internally in the DDS vendor implementation. These "configuration QoS" settings are referred to as "QoS DDS Extensions", and the same methods of programmatically (i.e. via the API used by the application) changing regular "Entity" QoS settings are available, as well as using "`.xml`" files.

Note that default QoS (including "configuration QoS" settings) are built-in to the DDS libraries, but RTI Connext DDS™ provides the "`.xml`" files to application developers to use in order to override the values.

B. Issues (with this feature)

Since "`.xml`" files can be read on DDS Startup on each DDS host, there are many ways to alter the configuration file, and ensure that it is read on startup.

In addition, since the default values are provided as "`.xml`" file templates by RTI, and they are in unencrypted, human-understandable plain-text, they are easy to understand and alter.

Changes to these files could be used in conjunction with other vulnerabilities to cause many different malicious effects within the DDS-based system.

C. DDS Flaws (used by the attack)

F1: Flaw in design	**F2**: Flaw in implementation	**F3**: Flaw in configuration

This run-time configuration flaw is only present for the RTI Connext DDS™ vendor implementation.

D. Attack Types

A1: Denial of Service (DoS) Attacks (internal): - **A1.1**: Host-based - **A1.2**: Network-based - **A1.3**: Distributed	A2: Network Attacks: - **A2.1**: Spoofing/Identity theft - **A2.2**: Session hijacking - **A2.3**: Replay of messages - **A2.4**: Network configuration corruption - A2.5: Routing Traps: o **A2.5.1**: Sinkhole Attacks o **A2.5.2**: Blackhole Attacks o **A2.5.3**: Greyhole Attacks	**A3**: Repudiation Attacks A4: Data Attacks: - **A4.1**: Deletion - **A4.2**: Modification - **A4.3**: Insertion - **A4.4**: Redirection - **A4.5**: Omission

Although this isn't really an attack, use of this technique could enable other attacks (outlined in this analysis) that use the altered configuration parameters.

E. Methods of Exploitation

M1: HW or SW Availability Attacks - **M1.1**: Outage - **M1.2**: Failover **M2**: Alteration of Application SW	**M3**: Alteration of Middleware (DDS) **M4**: Alteration of Configuration, File and/or File System	M5: Existing Deficiencies: - **M5.1**: Bugs - **M5.2:** Debugging interfaces/APIs **M6**: Alteration of HW

This involves changing a run-time configuration file.

The values could be (re)loaded via restarting the host or DDS processes (outage or failover), or at points specified by the application.

Note that it RTI Connext DDS™, the "`.xml`" files are also read upon creation of a new DDS Entity from the DomainParticipant (i.e. process running on the host).

F. Informational Impacts (Effects and Consequences)		
D1: Denial of Information **D2:** Destruction of Information	**D3:** Disclosure of Information **D4:** Discovery of Information	**D5:** Distortion of Information
Although this isn't really an attack, use of this technique could enable other attacks (outlined in this analysis) that use the newly enabled functionality, and cause the effects produced by them.		

G. Detection		
T1: Code/File Change **T2:** Process Change	**T3:** Message/Field Change **T4:** Communication Flow Change	**T5:** Performance Change
This could be discovered by a tool that detects changes to configuration files. It could also be detected at run-time by a tool that detects changes to the host process or thread execution (i.e. a crash, stop, reboot or failover).		

F.9 DDS Security Issue #7: Misuse of RTI Connext DDS™ Run-Time Configuration File Load Order to Specify a New Configuration File

A. Background (of this feature)

RTI Connext DDS™ uses "`.xml`" files to specify configuration settings and QoS Policy values. There are several mechanisms in which it does this, and there is a specific load order (first to last) in which it looks for the "`.xml`" files:

- **File**: "`$NDDSHOME/resource/xml/NDDS_QOS_PROFILES.xml`" is loaded automatically if it exists (it does not exist by default). This file contains the default QoS values that will be used for all entity kinds; (First to be loaded)
- **Environment Variable**: "`$NDDS_QOS_PROFILES`", which contains URL locators separated by semicolons. These files are loaded automatically if they exist;
- **File**: "`<working directory>/USER_QOS_PROFILES.xml`" is loaded automatically if it exists;
- **QosPolicy Attribute**: the "`url_profile`" attribute in the `PROFILE` QosPolicy, which contains URL locators. These files are loaded automatically if they are specified and exist;
- **QosPolicy Attribute**: the "`string_profile`" attribute in the `PROFILE` QosPolicy, which contains XML strings. These strings contain the QoS policy name and new value. (Last to be loaded).

Note that this configuration mechanism is executed whenever the following DomainParticipantFactory operations occur (i.e. not just at process startup) – it is not necessary to explicitly call the `load_profiles()` method. QoS Profiles are loaded when anytime a new DomainParticipant is created, or via the "`set_*`", "`get_*`" or "`reload_*`" functions are called (where * is "`library`", "`qos`", or "`profile`").

B. Issues (with this feature)
Since "`.xml`" files can added or altered on the host, and RTI Connext DDS™ supports several different methods using a "load order", this mechanism can be abused to add or alter the DDS configuration.

C. DDS Flaws (used by the attack)		
F1: Flaw in design	**F2**: Flaw in implementation	**F3**: Flaw in configuration
This run-time configuration flaw is present for the RTI Connext DDS™ vendor implementation.		

D. Attack Types		
A1: Denial of Service (DoS) Attacks (internal): - **A1.1**: Host-based - **A1.2**: Network-based - **A1.3**: Distributed	A2: Network Attacks: - **A2.1**: Spoofing/Identity theft - **A2.2**: Session hijacking - **A2.3**: Replay of messages - **A2.4**: Network configuration corruption - A2.5: Routing Traps: o **A2.5.1**: Sinkhole Attacks o **A2.5.2**: Blackhole Attacks o **A2.5.3**: Greyhole Attacks	**A3**: Repudiation Attacks A4: Data Attacks: - **A4.1**: Deletion - **A4.2**: Modification - **A4.3**: Insertion - **A4.4**: Redirection - **A4.5**: Omission
Although this isn't really an attack, use of this technique could enable other attacks (outlined in this analysis) that use the altered configuration parameters.		

E. Methods of Exploitation		
M1: HW or SW Availability Attacks - **M1.1**: Outage - **M1.2**: Failover **M2**: Alteration of Application SW	**M3**: Alteration of Middleware (DDS) **M4**: Alteration of Configuration, File and/or File System	M5: Existing Deficiencies: - **M5.1**: Bugs - **M5.2:** Debugging interfaces/APIs **M6**: Alteration of HW
This involves adding or changing parameters in one of the files, changing parameters in one of the files specified by a locator, or by changing the locator itself (i.e. redirect to a new locator). The values could be (re)loaded via restarting the host or DDS processes (outage or failover), or at points specified by the application. Note that it RTI Connext DDS™, the "`.xml`" files are also read upon creation of a new DDS Entity from the DomainParticipant (i.e. the process running on the host).		

F. Informational Impacts (Effects and Consequences)		
D1: Denial of Information **D2:** Destruction of Information	**D3:** Disclosure of Information **D4:** Discovery of Information	**D5:** Distortion of Information
Although this isn't really an attack, use of this technique could enable other attacks (outlined in this analysis) that use the newly enabled functionality, and cause the effects produced by them.		

G. Detection		
T1: Code/File Change **T2:** Process Change	**T3:** Message/Field Change **T4:** Communication Flow Change	**T5:** Performance Change
This could be discovered by a tool that detects changes to configuration files. It could also be detected at run-time by a tool that detects changes to the host process or thread execution (i.e. a crash, stop, reboot or failover).		