Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
May-June 2018: 1–15

DOI: 10.1177/1729881418770011
journals.sagepub.com/home/arx
SAGE

Robot Operating System 2: The need for a holistic security approach to robotic architectures

Vincenzo DiLuoffo, William R Michalson and Berk Sunar

Abstract

It is no secret that robotic systems are expanding into many human roles or are augmenting human roles. The Robot Operating System is an open-source standard for the robotic industry that enables locomotion, manipulation, navigation, and recognition tasks by integrating sensors, motors, and controllers into reusable modules over a distributed messaging architecture. As reliance on robotic systems increases, these systems become high value targets, for example, in autonomous vehicles where human life is at risk. As Robot Operating System has become a de facto standard for many robotic systems, the security of Robot Operating System becomes an important consideration for deployed systems. The original Robot Operating System implementations were not designed to mitigate the security risks associated with hostile actors. Robot Operating System 2, the next generation of the Robot Operating System, addresses this shortcoming, leveraging Data Distributed Services for its messaging architecture and Data Distributed Services security extension for its data protection in motion. This article provides a systematic review of Robot Operating System 2 and identifies potential risks for this new robotic system paradigm. A Robot Operating System 2 robotic system is viewed as a series of layers from the hardware that include sensors, motors, and controllers to the software layers, which include the operating system, security services, protocols, messaging, and the cognitive layer for observation, learning, and action. Since Robot Operating System 2 and security are new considerations for robotics systems as they move into mainstream, many questions emerge. For example, can some portions be secure and other portions be non-secure? Does everything need to be secure? What are the trade-offs between, security, performance, latency and throughput? What about real-time robotic systems? This article provides an overview of the Robot Operating System 2 paradigm and represents a first step toward answering these questions.

Keywords

Robotics, Robot Operating System (ROS2), Data Distributed Services, DDS Security, vulnerability analysis, machine learning

Date received: 15 September 2017; accepted: 4 March 2018

Topic: Special Issue – Distributed Robotic Systems and Society
Topic Editor: Anis Koubaa
Associate Editor: Eduardo Castello

Introduction

Robot Operating System (ROS) 2 was introduced in 2014 but was first available as alpha code in 3Q2015. ROS 2 has taken a different approach in its messaging layer from its predecessor and now employs the industry standard called Data Distributed Services (DDS), from the Object Management Group (OMG). A new DDS

Robotics, Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, Massachusetts, USA

Corresponding author:
Vincenzo DiLuoffo, Robotics, Electrical and Computer Engineering, Worcester Polytechnic Institute, 100 Institute Rd, Worcester, MA 01609, USA.
Email: vdiluoffo@wpi.edu

security specification extension was released later in 2016, for secure messaging.

ROS 2 is still in its early stages of development with Beta 1 released in December 2016, Beta 2 in July 2017, Beta 3 in September, and first release in December 2017 called Ardent Apalone. The implementation of security functionality is being pushed out beyond the first release according to the ROS 2 road map.[1] This research is based on the ROS 2 Beta 2/3 code base and the early access release 5.3 of the Real-Time Innovations (RTI) implementation of DDS with security extensions enabled for data protection in motion (DIM). This article applies to Ardent Apalone release as well, since the underlining DDS implementation is based on RTI 5.3.

As robotic systems are becoming more prevalent in today's society where autonomous systems are interacting with humans, the need for securing these systems is becoming paramount. Historically, industrial robots were mostly used in the manufacturing environment where they were protected by walls and closed networks. As robotic systems evolve, they are moving away from closed environments and toward open networks. Trend Micro published a research paper on vulnerabilities with industrial robots, where the network, controllers, and command/control were listed as attack vectors.[2] A class of autonomous robots with a large array of sensors and open connectivity that span land, water, and air will be the most susceptible to vulnerabilities. So, how are these systems different from computers or mobile devices and can the same techniques be used to secure them?

Distributed computer security is well known using Transport Layer Security or internet protocol security for securing communications and access control for enforcing data protection using well-defined labels. The computer security model is extended into the cloud where Gholami and Laure expand on cloud security, virtualization and container management for keeping sensitive data secure.[3] Computers in the distributed environment are mostly managed by system management run by a corporation or updated by the operating system (OS) and/or hardware manufacturer.

As for mobile device security, only two prime players, Android and Apple, provide a single point for how applications are distributed and the cellular providers have tight controls over the International Mobile Equipment Identity and International Mobile Subscriber Identity management. These devices are under system control once the device is turned on by making a call home and nearest cell tower connections. With these controls in place, Jover[4] is still able to construct attacks on LTE and location aware exploits using software-defined radio.

In both cases, computers and mobile devices have standards and are managed by an entity to assist in the security model. Robots, on the other hand, haven't had the maturity or the standardization, so security capabilities and system management are new concepts. The array of sensors and the cognitive layer also set robotics apart in comparison to mobile devices and computers.

The contribution of this article explores the systematic security model for a new ROS 2 robotic system, including the potential risks associated with the cognitive layer. A detailed analysis is performed on the DDS Security standard related to performance versus security models and how security is incorporated with ROS 2. Since ROS 2 and explicit attention to security issues are relatively new to robotics, there is a need to analyze the risks and benefits of incorporating security into a robotic system and how the inclusion of security impacts the system design.

The rest of the article is divided as follows. We provide some background of ROS 2 in the "ROS 2 security background" section and the "Systematic security review" section provides a systematic review of the ROS programming paradigm and the incorporation of security features. The "Advanced threats" section identifies several possible advanced attacks on a robotic system. The "Performance versus security models" section presents some initial results showing how performance is impacted by several available security models and shows how the choice of a security model can be critical in real-time systems. We conclude in the "Conclusion" section.

ROS 2 security background

ROS 2 abstracts the complexities of the interface description language (IDL) used by the DDS implementation, while preserving a familiar ROS-based application programming interface to robot applications. In Figure 1,[5] the left portion of the diagram is a simple layering showing the DDS implementation at the bottom, followed by the mapping layer that performs the conversion between ROS messages and DDS IDL. Portable C can be used for different clients as well as C++, Python, and native C. Other languages can be adapted by wrapping the client library and the specific application language logic. The right-hand side of the diagram shows a more detailed view of both static and dynamic paths that can be taken to generate the IDL. The dynamic path uses introspection that is not supported in Beta 2/3 but was in previous releases. For this reason, this article uses only the static path referenced for the ROS 2 layer.

Currently in the Beta 2/3 release, the C++ development tools are more mature than Python or C for developing ROS clients. ROS 2 targets Linux (main development platform), macOS, and Windows as its supported platforms. In this article, the ROS 2 system model is described using Ubuntu 16.04, ROS 2 Beta 2/3, and RTI 5.3 DDS with DDS Security. The underlying system has no additional security configurations.

Figure 2 illustrates a more detailed view of the RTI DDS implementation with security. The static and dynamic paths represent different mechanisms for invoking security. The static path utilizes the IDL and create_function_ptr and the dynamic path uses XML for both code generation and security enablement.

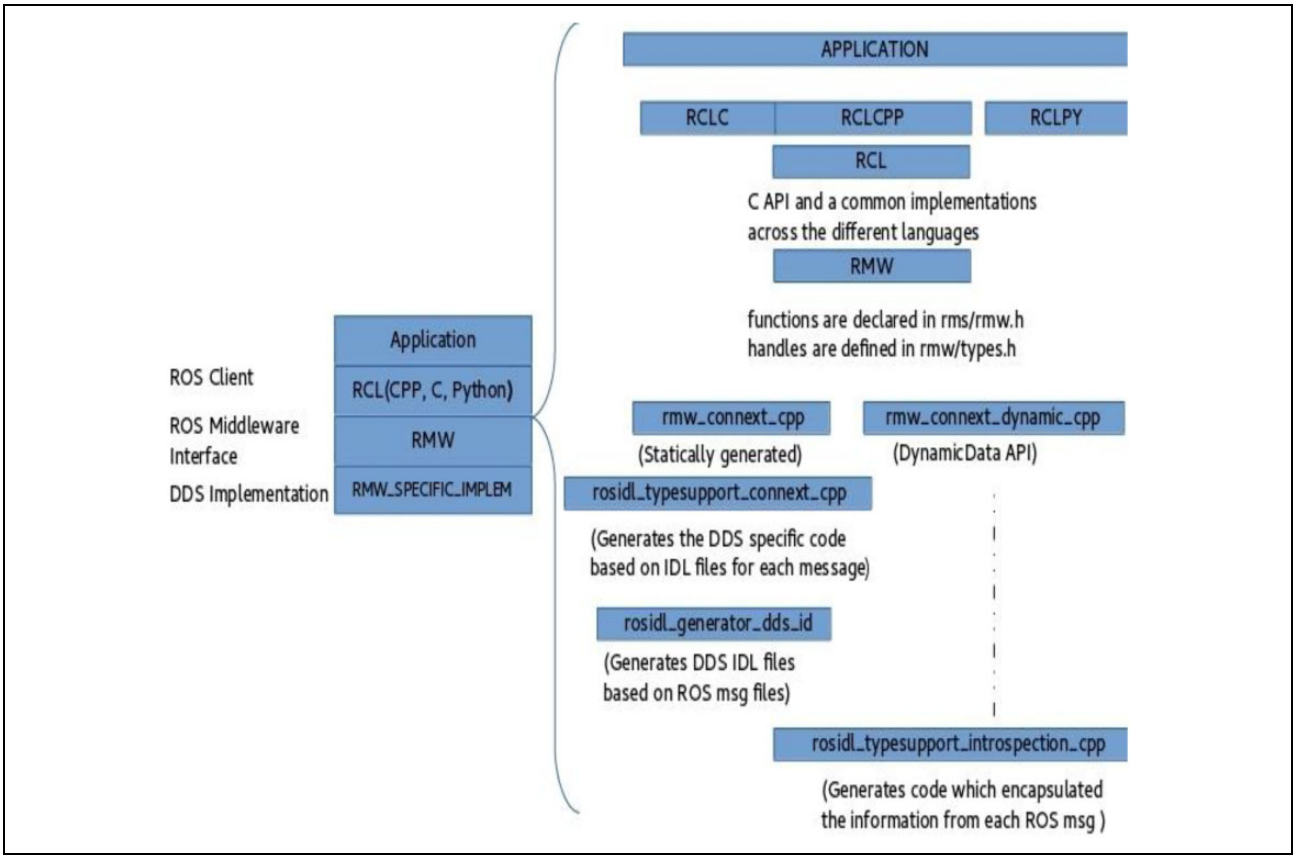

Figure 1. ROS 2 software architecture including the layering to DDS. ROS: Robot Operating System; DDS: Data Distributed Services.

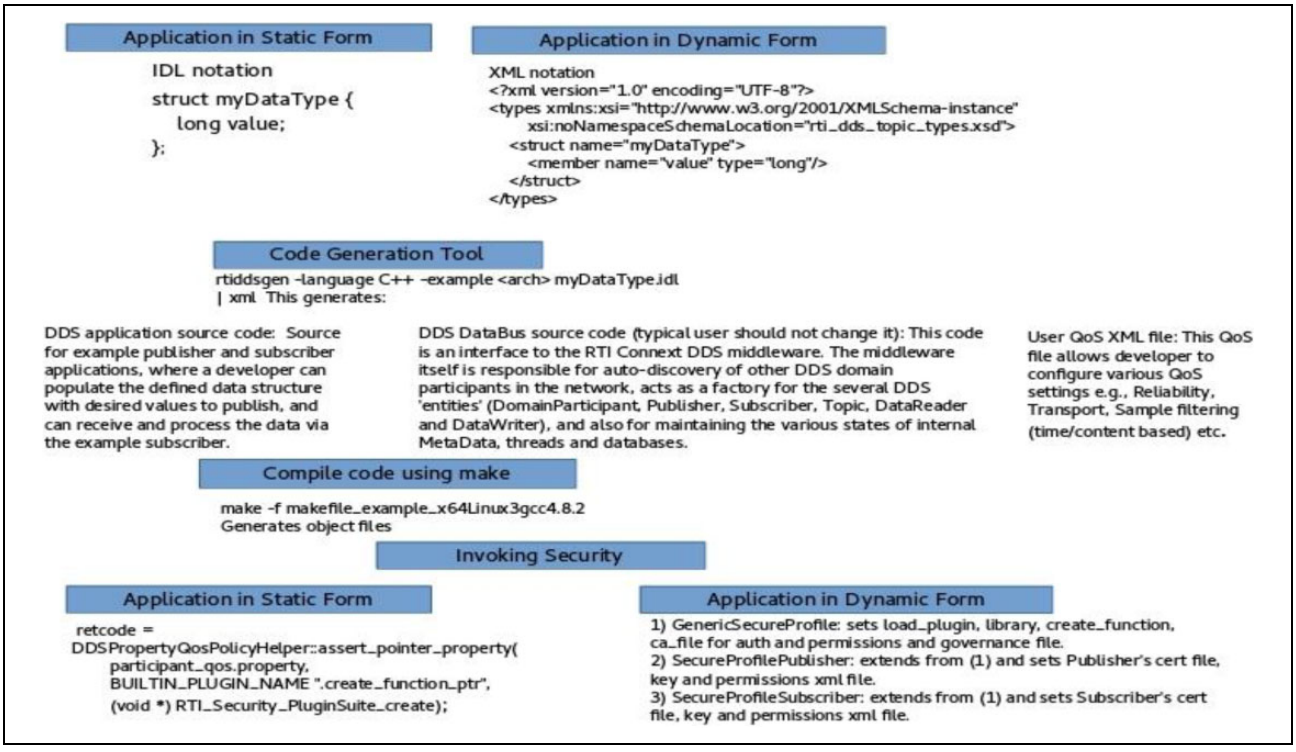

Figure 2. RTI software layers including security. RTI: Real-Time Innovation.

The new DDS Security extension provides ROS 2 with the capability to protect data in motion. The DDS Security extension relies on the public key infrastructure (PKI) utilizing hash, symmetric, and asymmetric cryptography. A couple of definitions related to security for background knowledge:

Symmetric key cryptography[6]-An example of a symmetric key algorithm is the Advanced Encryption Standard (AES), mostly used for bulk data encryption. The same key is used for cryptographic operations, encryption, and decryption of varying data length. Both block (chucks of data) and streaming (byte by byte or bits by bits) are types of symmetric ciphers. AES also provides different modes such as the Galois Counter Mode (GCM) used for streaming and Cipher Block Chaining (CBC) for block transfers. Other modes are defined in NIST Recommendations SP 800-38A (CBC)[7] and SP 800-38D (GCM and GMAC).[8]

Asymmetric or public key cryptography[9]-Examples of asymmetric algorithms are Rivest, Shamir, and Adleman (RSA) and Elliptical Curve (EC) algorithms, mostly used for authentication, digital signature, or key exchange. A pair of keys are generated, one portion is public (shared with whoever data is being exchanged with) and the other portion is private (only the owner has access to it). In sign/verify operation, the private key is used to sign and the public key is used to verify. In an encrypt/decrypt operation, the public key is used to encrypt and the private key is used to decrypt.

A Hash function[10] is logic that takes a message and reduces it into a digest where it becomes a one-way function and can't be reproduced without having the same data and algorithm. An example of a hash function is called the Secure Hash Algorithm (SHA-2). Hash functions are used to protect the integrity of the data from being altered. Using a Hash Message Authentication Key Code[11] falls under this category also, except that it uses a key as part of the hash algorithm.

PKI[10] is the set of systems/polices that provides the lifecycle management of the digital certificate. This includes the capability to issue, reissue, revoke, and store certificates. The certificate provides the authenticity that a public key is tied to a private key of the owner using the asymmetric key pairs. Components of the PKI are as follows:

X509 certificate-A standard structure for data about the issuer, validation period, distinguished name (DN), and role that the public key will be used for.

Registration authority (RA)-This can be performed automatically or in person depending on the policy for issuance. An example of automated registration is supplying an email address and receiving a certificate, meaning this is the lowest form of validation about the person's identity. In person registration, such as receiving a passport, is a more involved and more credible way of ensuring a person's identity.

Certificate authority (CA)—Performs the actual digital signature function, once the correct data have been authorized by the RA. Once the CA has digitally signed the certificate, the certificate can be stored in a repository for others to obtain similar to a phone directory.

Directory-A location where certificates and certificate revocation lists[12] are stored.

Table 1. Governance policy elements.

Element	T/F	None	Sign	Encrypt
Domain				
allow_unauthenticated_participants	X			
enable_join_access_control	X			
discovery_protection_kind		X	X	X
liveliness_protection_kind		X	X	X
rtps_protection_kind		X	X	X
Topic				
enable_discovery_protection	X			
enable_read_access_control	X			
enable_write_access_control	X			
metadata_protection_kind		X	X	X
data_protection_kind		X	X	X

DDS security extension defines two policies[13]

Domain governance policy—The governance policy defines how the domains are enforced. Elements are defined for the Domain and Topics within the domain in an XML file structure. A Boolean of true or false is used to turn the feature on/off or enabling security using none for no security, sign for integrity using (AES128 GMAC or AES256 GMAC), and encryption for confidential protection using (AES128 GCM or AES 256 GCM).[8] This is shown in Table 1. Enabling the security features using the Boolean values, for example, provides protection from unauthenticated participants gaining access, checks to see if a participant is authorized to gain access, enables the discovery protection on topics, and it enforces authorization protection to publisher or subscriber topics. The protection kinds are explained in more detail in the "Performance versus security models" section. The governance policy should be digitally signed by a CA to protect parameters from being altered.

Participant policy-This defines the permissions for the domain participant and for binding the subject to objects using the DN found in the certificate. The policy defines the domains that can be joined and controls read and write operations on topics and for data tag access. A set of permission rules for each topic to "allow" (determine who can read and/or write), "deny" (determine who isn't allowed to read and/or write), and "catch all using default" (allow or deny) can be defined. The participant policy should also be digitally signed by a CA.

The DDS Security standard calls for two separate CAs. One is for identity or participant credentials and the other is to digitally sign the policies as shown in Figure 3. There are five plug-ins defined by the standard: authentication, access, cryptographic, logging, and data tagging.

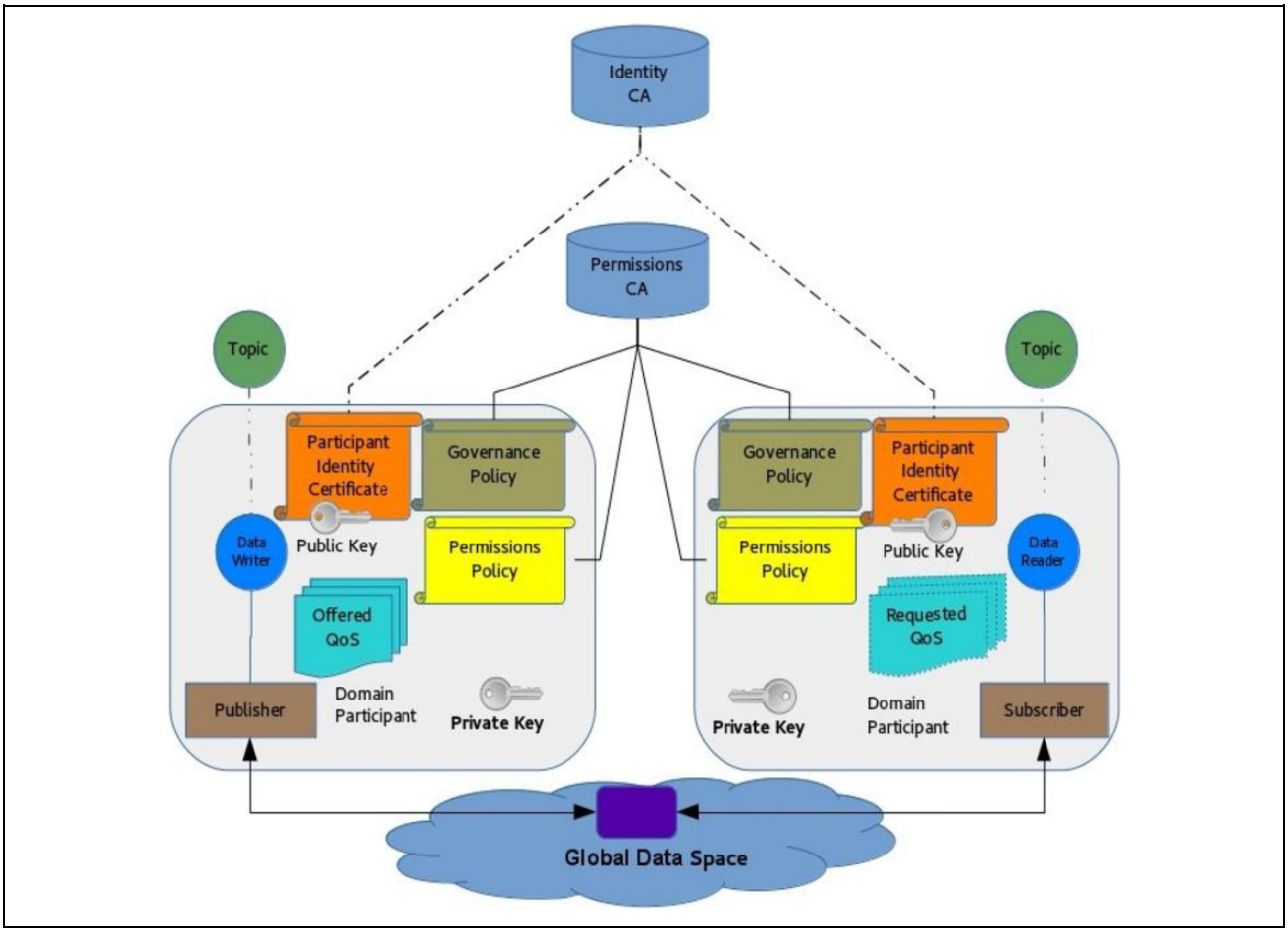

Figure 3. Data Distributed Services security deployment model.

Depending on the vendor's implementation of the security plug-ins, data tagging may be optional. These five plug-ins are used to support the DDS secure messaging, authorizing participant's actions, and logging.

Authentication—Identifies the participant (person or process) in the global data space using the X.509 certificate issued by the identity CA. Asymmetric algorithms that can be used are RSA[14] 2048 bits or Elliptical Curve Digital Signature Standard (ECDSA)[15] 256 bits. The Diffie–Hellman Key Agreement[16] is used to exchange symmetric keys using public keys. The DH key is 2048-bits modular exponential Group with 256-bits Prime as stated in the standard.[17]

Access—Provides the enforcement controls for what a participant can perform. Subject and Object controls using both policies as the access control list during DDS operations.

Cryptographic—Provides the cryptographic operations for performing encryption/decryption, sign/verify, hashing, and key generation. Example algorithms have been mentioned in the policy section and user authentication section above.

Logging—Provides the event tracking related to security operations and tasks being executed by a participant. Each event is time stamped.

Data Tagging—The capability to add additional tagging information onto data samples from the data writer.

Figure 3 shows the overall architecture of a secure DDS deployment model between two domain participants. The structure of the two CAs is shown in the center where one CA is for authenticating identity and the other used to digitally sign polices. Having two different CAs allows flexibility in the issuance process for identity and software signing. A validation process for issuing identity certificates is different from signing software, the RA process may request a face–to-face interview for identity issuance versus an auto RA to sign software. The certificates are shown with the key image and Governance/Participant polices are connected to the Permissions CA. DDS supports a set of discovery services that allows publishers and subscribers to dynamically discover each other. A domain participant allows an application to join the global data space and each topic is a string that addresses the objects in the same space. Each object is identified by a key where data writer and data reader are supported by pools of resources called publisher and subscriber. A data writer declares the intent to publish a topic and provides type-safe operations to write or send data.

A data reader declares the intent to subscribe to a topic and provides type-safe operations to receive data. An example of a publisher is a module that sends commands to a controller and a subscriber would be a module that receives those commands, this is referred to as pub-sub messaging.

How each participant discovers and shares information is enforced by the Governance and Participant security policies mentioned above and the usage of security plug-ins. The transfer of data between two participants in the global data space is performed using a Real-Time Publish Subscribe Protocol (RTPS).[18] The use of quality of service (QoS) profiles allows the RTPS communication layer to provide reliability and support for real-time environments where critical processes are under time constraints to complete. QoS profiles also enable the security parameters for the DDS Security deployment model. When security controls are turned on, discovery (who is publishing, sequence numbering, and in-line QoS), reliability (heartbeat, ack, nack) metadata, and payload data can be encrypted, providing the highest security protections but may result in latency and poor efficiency.

Systematic security review

System requirements should be fully understood in terms of what needs to be protected with respect to performance goals. The focus of using ROS 2 with DDS security turned on within a robotic system is a very new topic. What needs to be secure and how to design effective security while still achieving performance goals are questions that need to be addressed. In the "Performance versus security models" section, we demonstrate the effect that the choice of security model has on several performance measurements. A holistic security system approach is to understand the layering of the system and interactions with external entities. Vulnerability analysis should be performed in the same manner to expose the areas that pose higher risk and mitigate them. Thus, it is important when developing a system to determine what portions require what level of security. While adding DDS Security provides protection for participants at the distributed network messaging layer, this alone is not a holistic robotics security model since portions of the system may be overprotected (resulting in lower performance) or under-protected (resulting in unexpected vulnerabilities).

Since adding DDS Security is new in the context of robotic systems, it presents security (and potential performance) concerns due to the large amount of message traffic that results from ROS using the pub–sub paradigm as well as other concerns related to compromise at both hardware and software system elements. By taking a holistic security approach to developing ROS 2-based systems using the new DDS Security standard, a number of concerns may be identified as potential risks. In Figure 4, the building blocks of a typical robotic system are shown with nodes, indicating that the potential security risks that have been discussed in the past are being analyzed presently or may fall into future research areas. The DDS security standard addresses the DIM model for data security but falls short in other areas. Security vulnerabilities are represented by *non-invasive* (observe/manipulate data but no physical harm to device), *invasive* (any method to acquire data), and *semi-invasive* (in between the non-invasive and invasive cases) for hardware and *application programming interface (API) misuse* (insertion, evasion, denial of service), *unvalidated input* (parameter and scheme validation), *race conditions* (denial of service), side channel), *flow control* (buffer overflow and garbage collections issues), and *security issues* (access control, authentication, authorization, and cryptographic) for software.

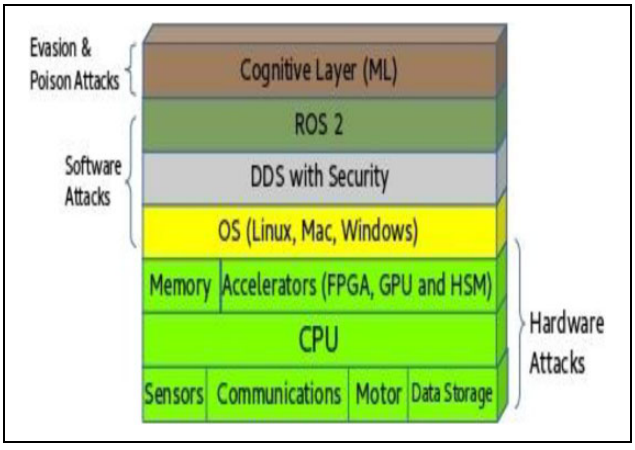

Figure 4. Vulnerability analysis of a ROS 2 robotic system. ROS: Robot Operating System.

A QoS profile is used to set paths to the CA and participant credentials as shown in Figure 3. Once the governance and participant policies are defined and credentials have been issued, security is enabled. As part of the policies, each topic must be known or a wild card can be used to express all topic behavior. In the ROS 2 layer, topic names will be used for either a publisher or subscriber. Using authentication, cryptographic and access plug-ins are to perform the function of establishing a secure session and of checking that each participant is allowed to have data access in either a write or read capacity. In the global data space, each participant must first be authenticated to the domain and then discovered by the services. Identity and permission tokens are used during this discovery step. This allows the participants access to the specified domain within the global data space. A secure session is dependent on the policies and on how the RTPS protocol is protected to move data within the specified domain.

It is important to understand that robots take on different forms that are dependent on the functions they perform. For example, soft robots for manipulating delicate objects and autonomous vehicles operating on public roads have drastically different operating environments; consequently,

they have drastically different security requirements as well. Depending on the type of robot and its functions, a security policy should be put in place for the PKI components and for how the individual validation processes are conducted. DDS security supports the concept of a plug-in framework but doesn't follow the logic of the Java Cryptographic Architecture or cryptographic application interface, where each service provider needs to be digitally signed to ensure that it has been vetted.

The DDS Security standard specifically calls out algorithms or cryptographic modes for the implementation to support. These algorithms or cryptographic modes may not be the best practice choices. The security plug-ins provide the mechanism to authenticate the participants and protect the session data being exchanged between them using the Governance and Participant security policies. We identify the areas that might use the best practice approaches versus those being specified in the standard.

For the authentication plug-in, a participant is issued a certificate based on one of the following types of algorithm/key definitions, RSA 2048 or ECDSA 256 bits. The authors of SafeCurve states that using prime256v1 curves is not safe due to elliptic-curve discrete logarithm problem being difficult and the gap of implementing elliptic-curve cryptography (ECC) security, exposing data to side channel attacks.[19] Other curves are offered to circumvent these shortcomings.

For the cryptographic plug-in, AES_GCM and AES_GMAC are used for sign and encrypt functions, which are symmetric key operations. As discussed earlier, processing symmetric key operations are low latency, especially when cryptographic modes are combined into an atomic operation. A number of published papers have investigated the exploits using AES_GCM[20] including forgery,[21,22] key recovery and timing attacks,[23] and nonce replay attacks.[21] AES_GCM is mostly discussed in the papers, but GMAC is a mode of GCM in which no plain text is supplied and the output is the authenticated field.

Another concern is the use of MD-5 as stated in the standard for key hash on the data and datafraq of the RTPS encrypted packets. MD5 and SHA-1 from a collision set of attacks have been vulnerable, but from preimage attacks the standard states that no known vulnerabilities have been found. The paper called "Finding Preimages in Full MD5 Faster than Exhaustive Search" details a cryptanalytic preimage attack on the full MD5[24]; also the National Institute of Standards and Technology (NIST) in 2005 published that SHA-1 shouldn't be used on future systems, which is a predecessor of MD5. NIST guidelines are pushing for new algorithms to be used while stating SHA-2 is still safe but with the release of SH-3 in 2014. SHA-3 might be a better alternative that could be considered.

The DDS Security standard mentions the use of PKCS#11, which is a standard for interfacing with hardware security modules (HSM) for credential storage and cryptographic operations. The DDS Security standard mentions the use of PKCS#11 for credential storage, but not cryptographic operations, because hardware accelerations would help in latency issues and data.

For the access control plug-in, a subject is the participant and an object is the topic. This is similar to the role-based access control model found in common OSs. The enforcement of the permissions is done by checking the polices as discussed earlier. Authentication and cryptographic plug-ins are used to establish keys used by the publisher and subscriber and using the permission token for the enforcement check. The standard doesn't discuss the role of who creates the policies and who submits them to be digitally signed by the CA. It seems that this role should be called the Security Officer.

For the logging plug-in, this is associated with the authentication and access control plug-ins for its activation. Once a participant is authenticated, logging will begin. The following structure is for the built-in logging function:

```
<topic_rule>
  <topic_expression>DDS:Security:LogTopic</topic_expression>
  <enable_discovery_protection>FALSE</enable_discovery_protection>
  <enable_read_access_control>TRUE</enable_read_access_control>
  <enable_write_access_control>FALSE</enable_write_access_control>
  <metadata_protection_kind>SIGN</metadata_protection_kind>
  <data_protection_kind>ENCRYPT</data_protection_kind>
</topic_rule>
```

Basically, if a participant is able to join the domain, it can write to the log file, but in order to read the log data a participant must have read access to the built-in topic name in its permissions policy. The built-in log data is protected by encryption.

For the data tagging plug-in, this provides the capability to add additional labels on data; for example, security classification that can be used for access control. The concern here is the potential for misclassification of data by the writer, since this is where the tags are being generated and associated with data.

The DDS Security standard states that, before authentication and access control can begin, the RTPS protocol is initialized with a sequence number that may be susceptible to prediction number attacks.[25] Randomizing can't be implemented using RTPS, since it's data centric. The authentication and access plug-ins need to check the sequence numbering for each of the messages being received or implement their own mechanism to mitigate prediction number attack. The RTPS specifications support endpoint checks, but no DDS built-in exists to access the underlining RTPS implementation for these checks. DDS built-ins are a predefined set of services supported by the vendor's implementation to perform functions, like discovering other participants on the network. So, in the case of DDS built-ins to check for prediction number attacks, this hasn't made it into a supported feature.

An area of security concern is the system management console, where an administrative security panel is enabled and the administrator can view the network topology and potentially see sensitive data being sent over the network. This could be considered an insider threat where the administrator has rights, but not a need to know. Other services may have similar access to sensitive data and these threats need to be explored in the OS layer.

Advanced threats

We define several adversarial models for the robot threat classification from software, cognitive, sand hardware (spoofing and side channels) attacks as shown in Figure 4. A robot system is constructed by hardware components such as sensors, motors (including optical encoders), controllers (open or closed loop), communications capabilities, and newer designs are adding accelerators (field-programmable gate arrays (FPGA), graphic processing unit (GPU), HSMs), all of which may be entry points into the system. In fact, the problem is worse because each of the components in the hardware listed above may contain one or more (CPUs, memories, or data storage), increasing the number of potential entry points into the system. A modern robot is truly a distributed system. The software stack for ROS 2 starts at the OS running DDS with security and integrating with the ROS 2 layer. The cognitive layer may run on top of the ROS 2 or as a node within the distributed system. The cognitive layer is a subscriber to sensor data and potentially consumes larger amounts of this data. It then interprets against its local knowledge base and takes action once a decision has been made. The action state may lead to publishing data to a topic, for example, position and velocity for locomotion. The cognitive layer creates a new potential entry point for an attack vector as it effects the decision-making and learning processes about the environment in which a robot operates.

Software

The OS supports the application logic, which are the ROS 2, DDS, DDS Security, and all the device drivers for the robot hardware. As mentioned earlier, all the software threats can be exploited for potential attacks at this level. The IDL is type-safe, which helps with buffer overflow and memory vulnerabilities, but Security Technical Implementation Guide (STIG) should be considered to help with risks. STIGs are a set of guidelines/checklist for securely configuring a OS that helps mitigate against adversary attacks or can be extended to cover the network equipment including configuration of firewalls, routers, etc.

In software attacks, the security controls are abused to place malicious code on a platform, which may lead to hijacking or eavesdropping. For example, cache attacks are a form of eavesdropping. Since CPUs are denser with logic (multiple cores), the use of caches is expanding at different levels. As data is being processed on a computer system that data may be placed into multiple caches over time and may still be available once the processing is completed. The exploit is achieved by introducing a spy routine to capture the side channel timing variations of the high-speed interconnect fabric and exposing the leaked data from the cross-core communications. Data cache attacks have been exploited at the inter processor connected fabric, Irazoqui et al.[26] discuss how information is leaked. The DDS security standard does not provide protection for the hardware components listed above and the different types of attacks described.

Cognitive layer

Robots are performing more complex tasks with the use of machine learning (ML) algorithms. The cognitive layer is truly the brain of a robot like a human. Robots use the sensors to see, touch, hear, smell, and taste. The input data from the sensors are fed into the ML engines to perform tasks such as navigation, object avoidance, object tracking, and path planning; in other words, these sensors form the basis upon which a robot makes its decisions. More specifically, they use their sensors to construct a model of the world around them and to form the basis for decisions about locomotion and/or manipulation. Thus, attacking what's sensed may cause the robot to build an erroneous world model and then make bad decisions. These types of vulnerabilities are not covered by simply adding DDS Security to the ROS 2 platform and protecting the messaging layer.

Adversarial attacks such as evasion and poison on ML are currently being researched, but no known solutions have been published. Perturbation of data is difficult to detect when large amounts of data are being compared between a known sample called trained data set and real data. The evasion attack[27] is geared toward misclassifying data as legit data and bypassing the detection. In the case of poisoning,[28] the training data is contaminated, so that the classifier is less reliable in the determination of the outcome. A black box technique was used on a number of remote ML services to poison the data that was trained locally, then substituted for the target.[29] Since the field of adversarial ML is new and a difficult problem to protect against contaminated data versus legitimate data, more research needs to be conducted.

Side channel

The sensors of a robotic system collect data from the environment and are processed by the cognitive layer. Thus, sensor error directly affects the decision-making process of the robot. Unlike system software, sensors are susceptible to spoofing attacks that include injecting fake signals. Most of these attacks fall into the non-invasive category. The paper "An emerging threat: eve meets a robot"

provided one example where the use of an adversarial frequency pulse was sent to the sensor to spoof the navigation of the differential drive. Anomalous data might be due to "normal" sensor data, but it could also be due to an attack. From the robot's perspective, it really doesn't matter—the same mechanism that provides reliability also enhances security in some cases. The countermeasure for this was a fault tolerant sensor to ensure the data were correct. Akdemir et al.[30] presented a number of both passive and active attacks on sensors but also provide countermeasures that can be used to aid in risk mitigation.[30]

Similarly, motors and controllers can be affected by communication faults that may fall into the non-invasive category, since spoofing by timing, injection, and monitoring are the major grouping for these attacks. Jiménez-Naharro et al.[31] provided glitch attacks on an I2C-based communications bus as an example for timing, injection, and monitor attacks, but also discussed a countermeasure using a frequency detector to mitigate the risk. The Luenberger Observer was used to reconstruct the state of the robot where the sensors were spoofed with injected false signals.[32] Non-invasive attacks on both camera and Light Imaging, Detection, and Ranging (LiDAR) that help autonomous vehicles navigate were used to spoof (blinding and timing) the sensors.[33] Petit et al. applied bright light to the camera, where the sensor was blinded and incapable of capturing a recognizable image. A countermeasure to blinding was placing multiple cameras as a fault-tolerance vision system. Timing attacks on LiDAR signals were performed to spoof the sensor into unknown states. The countermeasure for the timing attacks is possibly using redundancy or acquiring neighbor vehicle data in vehicle-to-vehicle communications. Altering the LiDAR's capture speed or signal pulses may aid in countermeasures against the timing attacks.

The classical CPU, memory, and data storage are well known for hardware vulnerabilities using non-invasive attacks such as electromagnetic interference (EMI), for CPU data leakage. EMI attacks can be viewed as acquiring CPU state by monitoring radiation signals or in the case of semi-invasive/ invasive attacks, EMI could be used to induce CPU errors. Attacks on memory range from noninvasive to semi-invasive, a simple example is to remove the memory from the computer and put it into a different computer to read out its contents. Freezing memory prolongs data retention when moving from one system to another. Attacks on data storage range from utilities to extract or recover data from a hard drive or reading data directly from the media.

Hardware accelerators are being used in robotics more frequently where off-loading processing logic is performed. FPGAs are a universal catch-all device that can be programmed using gate-level logic or IP-cores to create custom System on Chip processors. GPUs are gaining traction with the use of ML algorithms for processing large amounts of data in parallel. Depending on the type of hardware acceleration, a number of potential risks can make the system susceptible to attacks that may fall into all three hardware categories. By analyzing both EMI and Power signals, a number of techniques can be used to acquire sensitive data, this includes the simple power analysis, differential power analysis, differential electromagnetic analysis, and simple electromagnetic analysis.[34,35]

Performance versus security models

In order to get a better perspective of how security protection features effect system performance metrics, we focus on a simple system involving internode communications and examine what is being protected by the Governance policy related to the DDS_RTPS messaging as shown in Figure 5.

The blue and colored boxes represent elements of the DDS domain and some of these boxes have a 1:N relationship. The orange boxes are the policy protection elements and the red boxes are a subset of the message box that can provide finer protection control on the metadata (sub-message header and elements) and/or data protection (sub-message element that includes a serialized payload).

Applying protection around the orange and red boxes may be important, because in some environments, these data elements are considered sensitive data and can be exposed to threats via side channels or directly. Table 1 showed that the orange and red boxes can have a protection value of NONE, SIGN, or ENCRYPT on each element. The *Discovery_protection_kind* and *Liveliness_protection_*kind are related to the participant and endpoint blue boxes in Figure 5.

Discovery provides a complete picture of the domain, including other participants, readers, and writers. It helps to configure the communications with other writers and readers using transport, address, and port data. Discovery is supported by two protocols called Simple Participant Discovery Protocol (SPDP) and Simple Endpoint Discovery Protocol (SEDP). SPDP supports how participants are found and once found, they use SEDP to exchange data about the endpoints. From the discovery data, an adversary could create a network topology and in effect provide location maps. Again, deepening on security requirements having the SIGN value provides only integrity protection and not confidentiality, so the data is still in the clear. The ENCRYPT value provides confidentiality protection where data is ciphered then integrity protected (Encrypt then MAC) operation.

Liveliness provides the mechanism for checking if participants are alive and the communication path is working. The alive check is performed by entity builtins using the Liveliness QoS and Globally unique entity identifiers (GUID). Liveliness QOS defines how and when to test the communication paths between participants. A heartbeat is sent to each participant to ensure that they are still active and within the messaging a history cache is kept to determine what data has changed. A writer's history cache is used to store and manage data objects, this also incorporates the change cache where created, modified or deleted data records are kept. A reader must have the most to update information related to the data-object and the

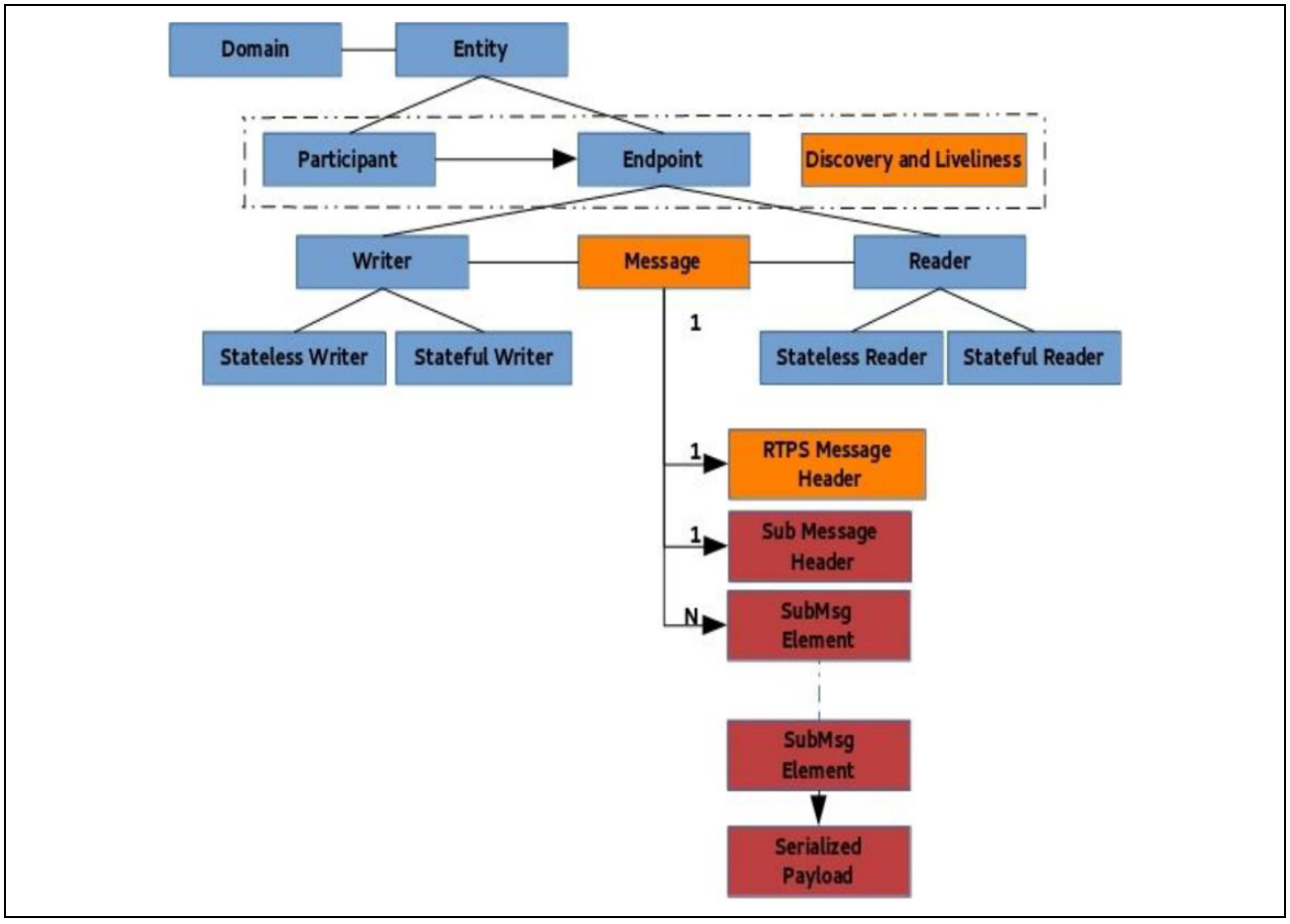

Figure 5. The Governance policy protection elements related to DDS_RTPS messaging. RTPS: Real-Time Publish Subscribe Protocol; DDS: Data Distributed Services.

liveliness mechanism provides that synchronization of the data between both writers and readers. The *Liveliness_protection _kind* can be either Sign or Encrypt depending on the security requirements. A potential risk of exposing the GUID in the clear is that it could allow an adversary to masquerade itself to collect data. With access to history and change data, an adversary could simply make changes in the data without any checks.

The *rtps_protection_kind* is related to the orange box called message. The *rtps_protection_kind* provides protection on the entire message including the message header, sub-message header, and all the sub-messages elements. Instead of protecting the entire message using *rtps_protection_kind*, a finer control can be achieved, shown in the red boxes with *metadata_protection _kind* and *data_protection_kind* as two independent operations.

The *metadata_protection_kind* provides protection on the sub-message header and the sub-message element that includes the GuidPrefix, EntityId, SequenceNumber, SequenceNumberSet, FragmentNumber, ragmentNumberSet, VendorId, ProtocolVersion, LocatorList, Timestamp, Count, and ParameterList elements. The *data_protection_kind* is only protecting the serialized payload sub message, so depending on the security requirements different levels of protection can be achieved using the Governance policy. A description of each element can be found in the RTPS specification.[36]

In order to analyze the performance of the example application using different security models, a utility was used to capture latency, throughput, and speed. Linux utilities such as the time, perf-test, and top commands provide the execution time, CPU utilization, or cache utilization. However, the performance data captured only relate to one publisher or subscriber client and there is no support to capture network or security enabled metrics. Therefore, since ROS 2 is a pub/sub-messaging platform, the common tools in Linux were unable to obtain the data results that were desired.

A toolset for determining messaging performance in a pub/sub environment should be able to measure the time a publisher sends a message and the time a subscriber sends a reply, this is referred to as latency. The proper throughput measurement tool measures the number of packets sent/received in a specific time. The measurement rate that packets are received is called the speed.

Having a utility to capture performance metrics for latency, throughput, and speed while also providing configuration for security is difficult, but RTI Perftest provides these features.[37] RTI Perftest is a command line utility that provides latency, throughput, and speed measurement data using RTI DDS messaging and the DDS Security for security enablement/configuration. RTI Perftest supports different configurations for messaging types, data block size, and security options. Four security options are defined as (1) *secureEncryptDiscovery*, (2) *secureSign*, (3) *secureEncryptData*, and (4) *secureEncryptSM*. These four options correspond to the governance policy protection kinds as described earlier. Both discovery and liveliness kinds are combined into the *secureEncrytDiscovery* and *secureSign* is used for the *rtps_protection* to only support the sign value. The *secureEncryptData* is the data protection for the serialized payload and *secureEncrptSM* is the metadata protection.

A number of experiments were conducted on a Lenovo W541 running Kubuntu 16.04 with RTI DDS 5.3, DDS Security, and the OpenSSL 1.0.2g. Two terminals windows were used, one to execute a publisher and the other to execute the corresponding subscriber. The block size used was 63KB, because if the block size was larger RTI Perftest would switch over to an asynchronous mode. Since RTPS supports UDP transport, its maximum datagram size is 64KB, so 63KB was chosen as the largest block size that could be handled without effecting the messaging mode and transport. The allowed execution time was kept at 100 s to be consistent across the experiments. The latency measurements are one way, so these data results would need to be doubled to calculate a round-trip estimate. We also wanted to observe the differences in the encryption algorithm and in key size, both RSA key length of 2048 bits and ECC 256 bits using prime256v1 curves were used.

Figure 6 shows the performance matrix related to latency, throughput, and speed. Figure 6(a) shows the latency performance results between no security enabled in the blue color, the magenta color represents the EC 256 bits, and the green color represents RSA 2048 bits. With no security enabled the latency on average was about 257 µs, EC was at 1385 µs and RSA was at 1343 µs.

Approximately 700 packets were transmitted in 100 s with no security enabled compared with 160 and 143, respectively, for EC and RSA encrypted packets. This means that the overhead to enable security is 137% in latency performance and 132% in number of packets transmitted.

Figure 6(b) and (c) shows the results for the throughput and speed performance tests comparing no security, EC, and RSA. The throughput rate for no security enabled was 69,769 packets/s at a speed of 35163.7 Mbps compared with 14522 packets/s at a speed of 7319.3 Mbps for EC and 14241 packets/s at a speed of 7177.7 Mbps for RSA. This means that the overhead to enable security is 132% in throughput and speed.

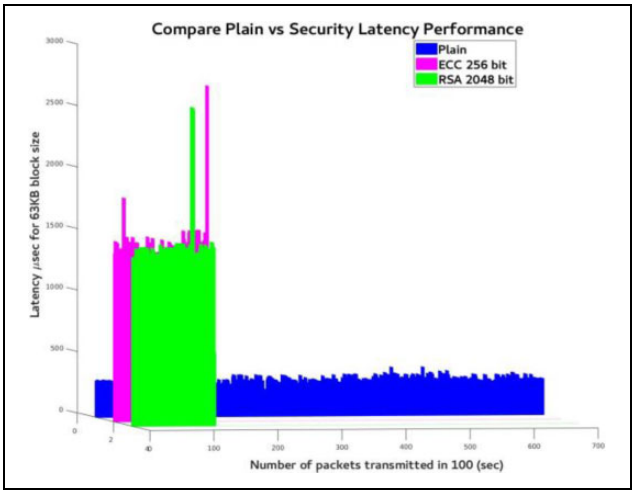

Figure 6a. A comparison between plain versus full security enabled using RSA 2048 bits and ECC 256 bits.

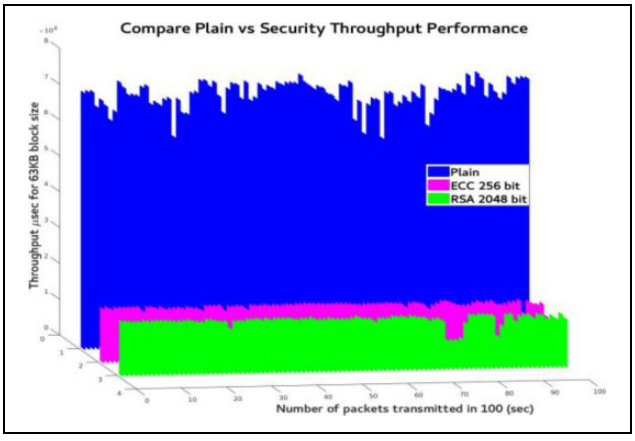

Figure 6b. A comparison between plain versus full security throughput enabled using RSA 2048 bits and ECC 256 bits.

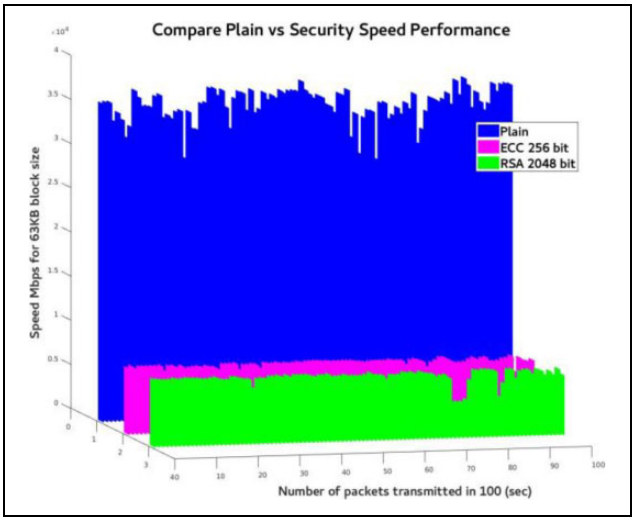

Figure 6c. A comparison between plain versus full security speed enabled using RSA 2048 bits and ECC 256 bits. RSA: Rivest, Shamir, and Adleman.

Table 2 shows the different permutations from the four security options related to ECC and RSA implementations. The governance file was changed, so that all tests had the same elements configured:

allow_unauthenticated_participants = false,
enable_join_access_control = true,
enable_discovery_protection = true,
enable_read_access_control = true,
enable_write_access_control = true
RSA μsec = RSA Latency μsec
RSA p/sec = RSA Throughput packets/sec
RSA Mbps = RSA Speed Mbps
ECC μsec = ECC Latency μsec
ECC p/sec = ECC Throughput packets/sec
ECC Mbps = ECC Speed Mbps
SEDis = secureEncryptDiscovery
SS = secureSign
SEData = secureEncryptData
SSM = secureEncryptSM

Table 2. Performance security options for RSA and ECC.

RSA μsec	RSA p/sec	RSA Mbps	ECC μsec	ECC p/sec	ECC Mbps	SEDis	SS	SEData	SSM
1343	14241	7177.7	1385	14522	7319.3	T	T	T	T
929	21035	10602.1	935	20619	10392.4	T	T	T	F
1138	16456	8294.1	1082	17506	8823.5	T	T	F	T
535	37665	18983.2	547	36811	18552.9	T	T	F	F
1200	16729	8431.4	1214	16397	8264.1	T	F	T	T
656	27305	13762.1	710	27291	13754.9	T	F	T	F
761	26250	13230.2	755	26711	13462.7	T	F	F	T
320	59381	29928.4	317	60807	30647.2	T	F	F	F
1420	14265	7189.6	1361	13687	6898.7	F	T	T	T
761	26250	13230.2	1086	20493	10328.7	F	T	T	F
1011	19906	10033.1	1069	18818	9484.5	F	T	F	T
552	37070	18683.3	598	34035	17153.7	F	T	F	F
1180	16749	8441.6	1212	16784	8459.3	F	F	T	T
752	26734	13474.2	758	26054	13131.7	F	F	F	T
614	28125	14175.1	656	28608	14418.5	F	F	T	F
257	69769	35163.7	264	71775	36174.7	F	F	F	F

RTI 5.3 only supported the *rtps_protection_kind* to have the value NONE or SIGN. All other protection_kinds in the governance policy support NONE or ENCRYPT. Depending on the security requirements enabling the five protection-kinds provides a higher level of security protection, but a performance penalty is incurred. The difference between using ECC versus RSA showed a slight improvement in latency, but in throughput and speed the difference was more pronounced. Having the discovery and liveliness protections enabled showed a minimum increase over having no security enabled. Having the encrypted data, discovery and liveliness protections with a signed RTPS message was an increase in latency of 113% compared with no security. Adding the meta protection increases the latency by 36%. The trade-offs between performance and security models can be drawn from Table 2 where the last line in the table represents no security and the first line being full security enabled.

An additional set of experiments were conducted using a remote machine over a wireless network. A Lenovo L450 was used with the same software stack as the W541. The wireless router used was the AUSIS RT-AC66U with support for (802.11AC). We looked at the latency, throughput, and speed for no security, full security using either RSA or ECC with the same key material as the above experiment. In order to get a better prospective on performance, we adjusted the packet data block size from 1K to 63K in powers of 2^n. In Figure 7, the X, Y, and Z coordinates relate to latency, throughput, and speed and the color bar represents the block size. Table 3 shows the values for Figure 7.

Table 3. Remote data transfer measurements.

Measurements	Plain data	Encrypt using ECC 256 bits	Encrypt using RSA 2048 bits
Block size (B)	1K	→	→
Latency (μs)	14,613	18,751	19,588
Throughput (packets/s)	2422	1742	1987
Speed (Mbps)	19	13	16
Block size (B)	2k	→	→
Latency (μs)	23,255	39,139	22,366
Throughput (packets/s)	1497	933	1122
Speed (Mbps)	23	15	18
Block size (B)	4k	→	→
Latency (μs)	53,349	54,328	61,270
Throughput (packets/s)	686	566	1122
Speed (Mbps)	21	18	17
Block size (B)	8k	→	→
Latency (μs)	65,816	91,531	104,085
Throughput (packets/s)	374	318	306
Speed (Mbps)	23	20	20
Block size (B)	16k	→	→
Latency (μs)	158,481	430,446	90,079
Throughput (packets/s)	159	151	28
Speed (Mbps)	20	19	4
Block size (B)	32k	→	→
Latency (μs)	52,947	123,340	130,292
Throughput (packets/s)	98	79	129
Speed (Mbps)	24	20	33
Block size (B)	63k	→	→
Latency (μs)	208,883	280,883	290,675
Throughput (packets/s)	29	29	48
Speed (Mbps)	18	13	24

RSA: Rivest, Shamir, and Adleman.

With block sizes, less than 8K the grouping was toward the right with a lower latency and centered for throughput and speed. As the block size increased so did the movement toward longer times and lower throughput and speed. The results were linear, but in some cases it took longer than 100 s to complete the test, because of the reliability QoS being used. This QoS ensures that no packets were lost in the communications, thus resulting in extending the latency duration and reducing the throughput.

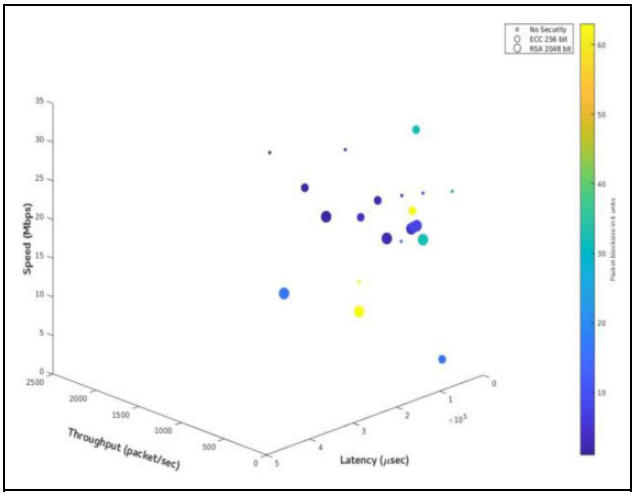

Figure 7. A comparison between plain versus security performance using a remote system and varying packet block size (1–63K).

As a result of the experiments performed, we concluded the following observations. While conducting the remote experiment if we took the single machine configuration and just ran the command statements as is, the RTPS/DDS would try each of the network interfaces and that prolonged the time for discovery. The "-nic" flag was used to direct the flow over a given IP address range for the network, for example, 192.168.1.*. To keep the measurements between single machine and two machine configurations the same, we didn't add additional flags to improve the performance gain. Our goal was to observe the behavior of performance-related security models. The ROS layer wasn't included, since this is only translation from ROS messages to DDS and most of the performance characterization was related to network communications and enabling security/cryptographic features. Future work to be considered is to add additional network traffic, enable logging and data tagging, and configure additional participants. Another focus area could be performance tuning and seeing if no security vs full security can be driven down under the 137% overhead.

Recommendations

The following areas are not covered by the standard but should be addressed during the product life cycle. From an overall system view starting at the hardware level, a root of trust should be incorporated so that both hardware and software are initialized to a known state. A self-test should be reflected in upstream layers and should be tied back to attestation measurements for both hardware and software. Another concern is the supply chain of where sourced components are acquired, this may introduce backdoor exposure. Trusted providers would mitigate some risks. Moving to the software side, the OS is open to a number of attacks where DDS and DDS security are running, so protecting the process and data should be considered at deployment. A potential open threat not addressed by DDS is attacks stemming from spy processes. A malicious spy process may be smuggled into the software ecosystem using an untrusted source. The spy processes would exploit leakages from other processes and even cryptographic libraries to recover sensitive information such as encryption keys. The cognitive layer has a number of known attacks that are not addressed and future research can be looking at ROS 2 security with ML.

Conclusion

Robotic security is a new and increasing concern and is not as mature as security for computers and mobile devices, thus there are many potential holes in the security architecture of a robotic system that need to be analyzed with respect to the threats unique to robotic systems. In this article, we covered the ROS 2 security model using the DDS Security extension and have identified potential security concerns that are not mitigated by DDS Security standard. A system vulnerability analysis was described using a taxonomy for both the hardware and software components of robotic systems. The OMG provides the specifications for the DDS[38] and RTPS[36] and a number of vendors are supporting both standards. ROS 2 and DDS Security extension is new and introduces a number of security concerns. The five security plug-ins were discussed with respect to potential concerns. The performance versus security model applied shows that adding security features to protected data in motion performance degradation is evident from a no security to a full security configuration. Table 4 is a comparison of measurements performed when security is fully enabled versus not enabled at all. There is a considerable difference between the two and further analysis should be conducted regarding performance versus the security model. Algorithm and key size made little difference compared to data protection features.

Table 4. Security measurement comparison.

Security enabled	Latency (average μs)	Throughput (average packets/s)	Speed (average Mbps)
Plain	260	70,772	35,669
Full	1363	14,382	72,485

An advanced set of threats were described related to hardware and the cognitive layer using ML. The DDS Security standard does not cover the advanced type of threats.

The ROS 2 security model is flexible in segmenting domains and participants to topics, so inherently ROS 2 allows being selective about what security techniques are applied to various portions of the robotic system. Thus, the question of applying security to everything or nothing in a robotic system can be addressed using two-level enforcement for access control, that is, using the governance and

permissions policies. However, it is essential to perform vulnerability analysis to determine risks and how they should be mitigated in a specific robotic architecture. The communications vulnerabilities of the robotics system that were raised in earlier papers can be mitigated using ROS 2 with DDS security; however, this is only a part of developing a secure robotic system.[39–41] We note that future work can be done to address the overall security of robotic systems. ROS 2 security is a start, but as shown from a holistic point of view many levels remain exposed.

Acknowledgement

The authors would like to thank Real-Time Innovation (RTI) for supporting the research with early access product and support.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

References

1. Arguedas M. ROS 2 raodmap. GitHub, https://github.com/ros2/ros2 (accessed 18 April 2017).
2. Quarta D, Pogliani M, Polino M, et al. Rogue robots: testing the limits of an industrial robot's security - wp-industrial-robot-security.pdf, https://documents.trendmicro.com/assets/wp/wp-industrial-robot-security.pdf (accessed 17 May 2017).
3. Gholami A, and Laure E. Security and privacy of sensitive data in cloud computing: a survey of recent developments. In *Academy & Industry Research Collaboration Center (AIRCC)*, 2015, http://www.airccj.org/CSCP/vol5/csit54711.pdf (accessed 15 February 2018).
4. Jover RP. LTE security, protocol exploits and location tracking experimentation with low-cost software radio. ArXiv160705171 Cs. 2016 Jul 18, http://arxiv.org/abs/1607.05171 (accessed 15 February 2018).
5. Thomas D. ROS 2 middleware interface. (Modified diagram), http://design.ros2.org/articles/ros_middleware_interface.html (accessed 18 April 2017).
6. Delfs H, and Knebl H. *Introduction to cryptography: principles and applications*. Springer Science & Business Media, 2007, p. 372.
7. Dworkin M. Recommendation for block cipher modes of operation. methods and techniques—nistspecialpublication800-38a.pdf. Report no. NIST Special Publication 800-38A, 2001. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf (accessed 24 April 2017).
8. Dworkin M. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC - nistspecialpublication800-38d.pdf. NIST Special Publication 800-38D, http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf (accessed 18 April 2017)
9. Stallings W. *Cryptography and network security: principles and practice*. London, United Kingdom: Pearson Education, 2016, p. 772.
10. Katz J, and Lindell Y. *Introduction to modern cryptography*, 2nd ed. CRC Press, 2014, p. 598.
11. Ferguson N, Schneier B, and Kohno T. *Cryptography engineering: design principles and practical applications*. Indianapolis, Indiana: John Wiley & Sons, 2011, p. 533.
12. Cooper D. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile, https://tools.ietf.org/html/rfc5280 (accessed 18 April 2017).
13. Specification OMG. DDS Security. Object Managing Group formal/2016-08-01. 2016, http://www.omg.org/spec/DDS-SECURITY/1.0
14. Rivest RL, Shamir A, and Adleman L. A method for obtaining digital signatures and public-key cryptosystems. *ACM* 1978; 21(2): 120–126.
15. Information Technology Laboratory. Digital Signature Standard (DSS). National institute of standards and technology. Report no. NIST FIPS 186-4, July 2013. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf (accessed 13 April 2017).
16. Rescorla E. Diffie-Hellman key agreement method, https://tools.ietf.org/html/rfc2631 (accessed 18 April 2017).
17. Lepinski M, and Kent S. Additional Diffie-Hellman groups for use with IETF standards, https://tools.ietf.org/html/rfc5114 (accessed 18 April 2017).
18. Specification OMG. The real-time publish-subscribe protocol (RTPS) DDS interoperability wire protocol specification. Object Manag Group Pct07-08-04. 2007, http://www.omg.org/spec/DDSI-RTPS/2.2
19. Daniel J. Bernstein and Tanja Lange. SafeCurves: choosing safe curves for elliptic-curve cryptography, http://safecurves.cr.yp.to/ (accessed 9 August 2017).
20. Ferguson N. Authentication weaknesses in GCM. Comments Submitt NIST Modes Oper Process. 2005, http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf (accessed 18 April 2017).
21. Böck H, Zauner A, Devlin S, et al. Nonce disrespecting adversaries: practical forgery attacks on GCM in TLS, https://www.usenix.org/sites/default/files/conference/protected-files/woot16_slides_bock.pdf (accessed 18 April 2017).
22. Mattsson J, and Westerlund M. Authentication key recovery on galois/counter mode (GCM). In: *International conference on cryptology in Africa*, Fes, Morocco, Africa, 2016. pp. 127–143. Springer, http://link.springer.com/chapter/10.1007/978-3-319-31517-1_7 (accessed 18 April 2017).
23. Gueron S, Langley A, and Lindell Y. AES-GCM-SIV: specification and analysis. 2017, http://eprint.iacr.org/2017/168.pdf (accessed 18 April 2017).
24. Sasaki Y, and Aoki K. Finding preimages in full MD5 faster than exhaustive search.pdf, https://www.iacr.org/archive/eurocrypt2009/54790136/54790136.pdf (accessed 18 April 2017).

25. Bellovin SM, and Gont F. Defending against sequence number attacks, https://tools.ietf.org/html/rfc6528 (accessed 18 April 2017).
26. Irazoqui G, Eisenbarth T, and Sunar B. Cross processor cache attacks. In: *Proceedings of the 11th ACM on Asia conference on computer and communications security*, 2016, pp. 353–364. New York, NY: ACM, http://dl.acm.org/citation.cfm?id=2897867 (accessed 25 April 2017).
27. Biggio B, Corona I, Maiorca D, et al. Evasion attacks against machine learning at test time. In: *Joint European conference on machine learning and knowledge discovery in databases*. 2013, pp. 387–402. Heidelberg, Berlin: Springer, http://link.springer.com/chapter/10.1007/978-3-642-40994-3_25 (accessed 18 April 2017).
28. Biggio B, Nelson B, and Laskov P. Poisoning attacks against support vector machines. ArXiv12066389 Cs Stat. 2012 Jun 27, http://arxiv.org/abs/1206.6389 (accessed 18 April 2017).
29. Papernot N, McDaniel P, Goodfellow I, et al. Practical black-box attacks against machine learning. In: *ACM Press*, 2017, pp. 506–519, http://dl.acm.org/citation.cfm?doid=3052973.3053009 (accessed 18 April 2017).
30. Akdemir KD, Karakoyunlu D, Padir T, et al. An emerging threat: eve meets a robot. In: *International conference on trusted systems*, 2010, pp. 271–289. Heidelberg: Springer, http://link.springer.com/10.1007%2F978-3-642-25283-9_18 (accessed 21 April 2017).
31. Jiménez-Naharro R, Gómez-Bravo F, Medina-García J, et al. A smart sensor for defending against clock glitching attacks on the i2c protocol in robotic applications. *Sensors* 2017; 17(4): 677.
32. Shoukry Y, and Tabuada P. Event-triggered projected Luenberger observer for linear systems under sparse sensor attacks. In: *2014 IEEE 53rd annual conference on decision and control (CDC)*, 2014, pp. 3548–3553. IEEE, http://ieeexplore.ieee.org/abstract/document/7039940/ (accessed 21 April 2017).
33. Petit J, Stottelaar B, Feiri M, et al. Remote attacks on automated vehicles sensors: experiments on camera and LiDAR, https://www.blackhat.com/docs/eu-15/materials/eu-15-Petit-Self-Driving-And-Connected-Cars-Fooling-Sensors-And-Tracking-Drivers-wp1.pdf (accessed 18 April 2017).
34. Sun S, Yan Z, and Zambreno J. Experiments in attacking FPGA-based embedded systems using differential power analysis. In: *2008 EIT 2008 IEEE international conference on electro/information technology*, 2008, pp. 7–12. IEEE, http://ieeexplore.ieee.org/abstract/document/4554259/ (accessed 25 April 2017).
35. De Mulder E, Buysschaert P, Ors SB, et al. Electromagnetic analysis attack on an FPGA implementation of an Elliptic curve cryptosystem. In: *2005 EUROCON 2005 The international conference on computer as a tool*, 2005, pp. 1879–1882. IEEE, http://ieeexplore.ieee.org/abstract/document/1630348/ (accessed 25 April 2017)
36. DDSI-RTPS, http://www.omg.org/spec/DDSI-RTPS/ (accessed 18 April 2017).
37. rtiperftest: RTI Perftest is a command-line application that measures the Latency and Throughput of very configurable scenarios that use RTI Connext DDS middleware to send messages. RTI Connext DDS Community, 2017, https://github.com/rticommunity/rtiperftest (accessed 30 July 2017).
38. Object Management Group Specification. Data Distribution Service (DDS). 2007, Document Number: formal/2015-04-10 Standard, http://www.omg.org/spec/DDS/1.4
39. Denning T, Matuszek C, Koscher K, et al. A spotlight on security and privacy risks with future household robots: attacks and lessons. In: *Proceedings of the 11th international conference on ubiquitous computing*, 2009, pp. 105–114. New York, NY: ACM, http://dl.acm.org/citation.cfm?id=1620564 (accessed 18 April 2017).
40. Bonaci T, Herron J, Yusuf T, et al. To make a robot secure: An experimental analysis of cyber security threats against teleoperated surgical robots. ArXiv Prepr ArXiv150404339. 2015, https://arxiv.org/abs/1504.04339 (accessed 8 April 2017).
41. Gil S, Kumar S, Mazumder M, et al. Guaranteeing spoof-resilient multi-robot networks. Auton Robots. 2017 Feb 28, http://link.springer.com/10.1007/s10514-017-9621-5 (accessed 8 April 2017).