Internal Cognitive Assurance Model for Autonomous Robotic Systems (ICAMARS)

by

VINCENZO DILUOFFO

A Dissertation
Submitted to the Faculty
of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

In

Robotics Engineering

May 2021

APPROVED:

_____
Name: Professor William R. Michalson. Robotics Engineering Advisor

_____
Name: Professor Berk Sunar Electrical and Computer Engineering CO-Advisor

_____
Name: Paul Pazandak, Ph.D.

ACKNOWLEDGMENTS

First and foremost, I am extremely grateful to my supervisors, Bill Michalson and Berk Sunar for their invaluable advice, continuous support, and patience during my PhD study. I am grateful for Professors Gregory Fisher, John Nafziger, and Alice Squires' robotics classes. Thank you to Teaching Assistants, Matt Bowers, Erik Skorina and the rest of the RBE faculty for their support.

My PhD journey was inspired by Marco Pistoia and Keith Jarvis after they planted the seed of their own blazed trail. My journey into Robotics has been a lifelong passion since my childhood and I would like to thank RTI for their software and support. I would like to thank Paul Pazandak for being part of the defense committee. This long endeavor would not be possible without the unwavering support of my wonderful wife Alice, who read all my papers and made sacrifices throughout. I am very appreciative also of the support of my children, Ryan, Thomas, and Julia, with support from Lauren, Victoria, and Lupin.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABSTRACT

It is no secret that robotic systems are expanding into or augmenting human roles. We have seen an increase in the number of autonomous vehicles, ride services, aerial and maritime vehicle companies. As these robotic systems are deployed in an essentially unbound environment, they are therefore more susceptible to adversarial attacks. Artificial Intelligence is enabling the progression of autonomous systems as we witness this with self-driving cars, drones, deep sea and space exploration. The increased level of autonomy provides new security exposures, which are different from conventional ones. As the Robot Operating System (ROS) has become a de facto standard for many robotic systems, the security of ROS becomes an important consideration for deployed systems. The original ROS implementations were not designed to mitigate the security risks associated with hostile actors. This shortcoming is addressed in the next generation of ROS, ROS 2 by leveraging DDS for its messaging architecture and DDS security extensions for its protection of data in motion. However, ROS 2 security only addresses a subset of the overall system and does not address new security consideration necessary for autonomous robotic systems. As a result, many questions emerge and can be categorized into performance tradeoffs, vulnerability analysis, and determining if trust metrics/solutions exist. Upon investigating a number of these questions, the results advocate for a holistic approach.

Therefore, our focus is on a holistic approach for assessing system trust which requires incorporating system, hardware, software, cognitive robustness, and supplier level trust metrics into a unified model of trust. While there are extensive writings related to various aspects of robotic systems such as, risk management, safety, security assurance and so on, each source only covered subsets of an overall system and did not consistently incorporate the relevant costs in their metrics. This study was motivated by a need to address the demand for a holistic security architecture for autonomous systems. In this research, we have defined trust metrics for each of the layers in an autonomous robotic architecture. The resulting internal assurance model utilizes a Bayesian Network for scoring each subsystem based on security-enabled features. This Bayesian Network is used to determine the internal trust of an autonomous robotic system before it can be extended to an external entity. While this model is for static assessment, our future work looks to extend the base model approach to a dynamic operational one in which the defenders kill chain is introduced

into the model. While utilizing a Dynamic Bayesian Network model, mitigation strategies can be applied to reduce security risks and provide platform resiliency.

CHAPTER 1 

1 INTRODUCTION

As robots increase their capabilities and expand into different markets, the question of security needs to be addressed. One of the first papers related to robotics, security, and privacy experimented with toys [1]. Historically, industrial robots were mostly used in the manufacturing environment where they were protected by physical barriers: walls and closed networks. However, autonomous robots, with a large array of sensors and open connectivity that span land, water, and air have no such physical barriers. These systems will be the most susceptible to security vulnerabilities. This research provides an overview of the current state of ROS 2 security, evidence of exploits, a survey of trust metrics and a potential approach to a holistic security architecture for autonomous robotic systems. The need for a holistic approach led to this dissertation work that produced the following papers.

The paper titled “Robot Operating System 2: The need for a holistic security approach to robotic architectures” [2] provides an overview of ROS 2 security, a security assessment and the tradeoffs between performance vs security for real-time systems. The paper discusses many attack vectors and system layers within a robotics system and the need for a holistic approach to security must be considered during the lifecycle of the system.

In a second paper titled “Credential Masquerading and OpenSSL Spy: Exploring ROS 2 using DDS security”, we provide sound evidence that exploits can be achieved with ROS 2 on a Linux environment using RTI 5.3 DDS [3]. Two exploits are presented, the first uses the OpenSSL library to dump sensitive data and the second manipulates a configuration file, so that masquerading credentials can be used. This paper also presents a mitigation strategy against the exploits, but also makes the point that data in motion protection is only a small part of the overall system that needs to be protected.

In the third paper titled “A Survey on Trust Metrics for Autonomous Robotic Systems”, we searched for other solutions that addressed the holistic security need for autonomous robotic systems. This holistic security approach needs to cover the system, hardware, software, cognitive/AI robustness, and supplier layers. Our research concluded that while solutions were presented, they only addressed small portions of the holistic model or did not address costs for reward or damage inflicted by adversaries. This led to our definition for trust metrics that covers each of the layers, based on standards and/or well-known values.

In the final paper titled “Internal Cognitive Assurance Model for Autonomous Robotic Systems (ICAMARS)”, a new holistic security approach is defined. This paper introduces a scoring system for each of the system layers and components that depend on the system, hardware, software, cognitive/AI robustness, and suppliers’ layers using a Bayesian Network model for static security assessment. These scores are then used to reason about the posture of the system when interacting with external entities. This internal model allows the capability to further analyze and decouple from the normal external request, authenticate, authorize, and fulfill. We extend the model from static

assessment to a dynamic concept where threats can be handled in real-time using a Dynamic Bayesian Network (DBN) model.

1.1 Background

The Autonomous Robot Market is expected to rise from its initial estimated value of USD 6156.78 million in 2018 to an estimated value of USD 17748.47 million by 2026, registering a CAGR of 14.15% in the forecast period of 2019-2026 [4]. Of the autonomous robot market, ROS is enabling a number of platforms and security should be a focus area for the expected deployments of these systems.

ROS is an open source project maintained by Open Robotics, that has expanded into industrial, military, agricultural and automotive robotics. Other work was underway for hardware base models to connect building blocks together at the physical level and be ROS compliant, but no funding was acquired to move forward [5]. Figure 1-1 shows the progression for the global robot market using robot operating systems (ROS) reached a volume of 4.4 million units in 2018 [6].

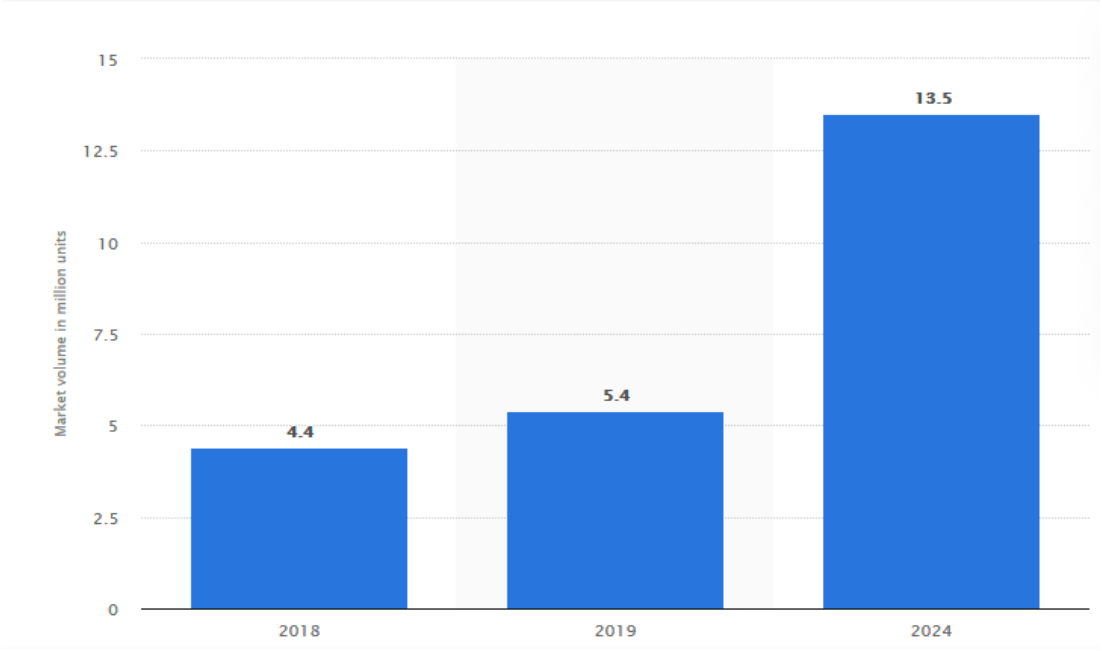


Figure 1-1: Global ROS-based robot market volume 2018-2024

The Robot Operating System (ROS)

The original version of ROS, now referred to as ROS 1, was developed as a distributed architecture using publisher/subscriber messaging between nodes. Figure 1-2 is an example of a ROS messaging between two nodes using topics “Talker” as a publisher and “Listener” as a subscriber. The ROS Master sets up the topics to find each other. This is called a central message broker model where communication initialization is started and then each of the topics can communicate directly with each other. The parameter server is a service run on the Master to support global value storage. An example of global values are configuration parameters where they can be set or retrieved from the ROS network. ROS 1 provides no means of security, since this was a research platform to enable the building blocks for locomotion, manipulation, and navigation for robots. Indeed, vulnerabilities have been identified as the master node provides open ports that are scannable over the Internet [7].

The Robot Operating System 2 (ROS 2)

The second version of ROS, ROS 2, was introduced in 2014 but was first available as alpha code in the third quarter of 2015. ROS 2 has taken a different approach in its messaging layer and now employs industry standard Data Distributed Services (DDS), from the Object Management Group (OMG) to provide secure messaging. The coupling of a real-time transport, Quality of Service (QoS) and security are gained from adopting the DDS standard within the ROS 2 framework.

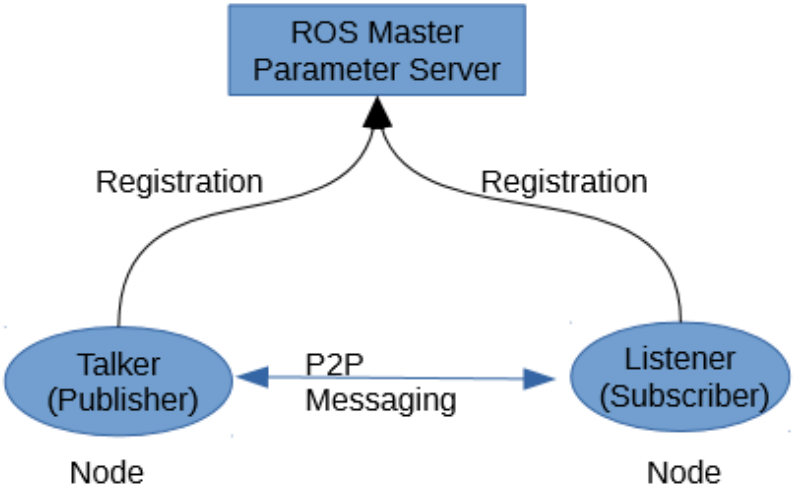


Figure 1-2: ROS 1 Messaging using topics

Figure 1-3 [8] shows that ROS 2 uses a similar model to ROS 1, but eliminates the central broker model component. DDS supports a set of discovery services that allows publishers and subscribers to dynamically discover each other without the name server bottleneck caused by a central broker. 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 writers and data readers are supported by pools of resources called publishers and subscribers. 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 read or receive data. DDS uses the Real-Time Publish Subscribe

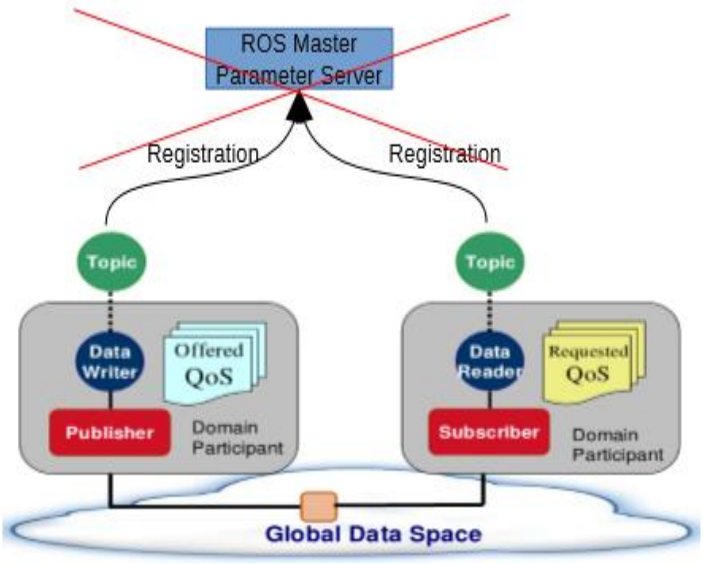


Figure 1-3: Data Distributed Service (DDS) Software Model

(RTPS) protocol for its data transportation. The RTPS protocol is designed to run over multicast and connectionless best-effort transports such as UDP/IP. The use of QoS profiles allows the RTPS communications layer to provide reliability in a lossy environment like wireless/cellular and provides support for real-time environments where critical processes are under time constraints to complete.

ROS 2 Security Model

In ROS 2 the DDS security model is utilized to protect the data in motion as shown in Figure 1-4 where the publisher and subscriber exchange data over a secure RTPS connection. As part of the model there are policies that control the behavior for how each entity in the global space interacts. Each entity, may it be a publisher or

subscriber is supported by a security library that is made up of five plugins called authentication, access, cryptographic, logging and data tagging.

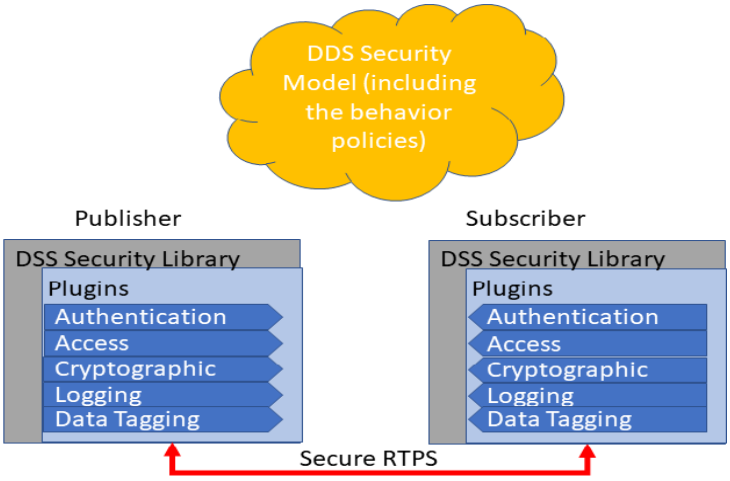


Figure 1-4: DSS Security Model

A detailed technical description is provided in Chapters 2 and 3 which covers the secure messaging supported in the ROS 2 DDS implementations.

1.2 Problem Statement

Depending on the type of autonomous robotic system, these systems may enable processing of sensitive data and/or pose human risk if compromised. For example, autonomous vehicles are one of the fastest growing segments, where autonomy classification levels range from 1 to 5, where level 5 is fully autonomous. As level 5 is still a reach goal, we are witnessing more related accidents caused by the system not correctly detecting objects or having the reasoning maturity to deal with the uncertainties that humans encounter. As more and more of these vehicles push the envelope toward reaching level 5, adversaries will always be on hand to exploit the technology for reward.

There are a number of problems related to securing autonomous robotic systems from adversarial threats. First, security is often an afterthought where products are augmented with security features that are more or less band aides for exploits. For example, password enforcement techniques like password length, complexity in pattern and expiration period. These stovepipe type solutions only cover bits and pieces of the overall systems, so protecting only a portion of the system will leave the other areas exposed.

Second, autonomous robotic systems are often in the wild where they are susceptible to physical threats. Physical threats can be categorized as invasive, semi-invasive and non-invasive that exploit side channels. These physical threats are discussed in Chapter 1. This is an important difference between autonomous robotic systems and conventional computer systems in which part of the security depends on the computer being protected within a physical construct and physical safeguards are implemented. Another difference between conventional computer systems and autonomous robotic systems is the use of a plethora of sensors that autonomous systems use to interact with their environment. These sensors are prone to spoofing attacks.

Finally, autonomous robotic systems are driven by cognitive/AI algorithms which pose different types of threats such as, using tainted training data or spoofing a classifier to detect the wrong object. This topic is touched upon in Chapter 2, but details are discussed in Chapter 4 and 5. By examining these problem areas in a holistic security architecture model, this can reduce the number of threats depending on the implemented security features of the system.

1.3 Research Statement

Since autonomous robotic system security is still maturing as opposed to conventional computer security, a number of questions have begun to emerge throughout the stages of this research. *These questions first start with ROS 2 related to performance vs security and if that would affect the real-time constrains imposed by robotic systems. Is there a difference between using different algorithms and/or DDS features related to performance, throughput, and speed? By ROS 2 supporting DDS security what system elements were exposed or vulnerable in a robotic system?*

Another set of questions were related to DDS security and if that provided adequate security for the system. Within the software stack what components provided the most protection but allowed an adversary complete control of the system?

Since DDS security is only protecting data in motion, does a solution exists that would cover the rest of the system? This was the key question that was the steppingstone to asking the following, what trust metrics are needed to define a holistic security architecture? This drove the next question; can a BN be used to support all the trust metrics and represent an internal security model. What are the benefits from creating an internal/external security model? Another question, is a static model sufficient to assess an autonomous robotic system? From this static model, can it be extended to a dynamic model?

1.4 Contribution of Research

As the number of autonomous robotic systems being deployed increases, security should be considered up front. There has been limited literature related to robotic security that encompasses the holistic security architecture approach. This

research work contributes a security assessment of ROS 2, defines trust metrics and a means to assess an autonomous robotic system's internal security posture using BN. Through this research, an internal holistic security model was defined to incorporate trust metrics for the system, hardware, software, AI robustness and supplier components. We explain how the static model can be extended to a dynamic model, where the dynamic model will allow the autonomous robotic system to react to threats in real-time. These reactions to threats can be viewed as both offensive/defensive postures so that resiliency can be achieved.

1.5 Outline

The following chapters address the research questions and background needed to answer these questions and achieve the research goal. In Chapter 2, the current state of security related to ROS 2 is discussed with a focus on robotic system security assessment and the comparison of performance versus security for different configurations of DDS. In Chapter 3, exploits are discussed as well as mitigation techniques. This chapter also address the need for a holistic security approach. In Chapter 4, we search for a holistic security approach for autonomous robotic systems and start to construct a definition for a holistic security architecture. We survey the trust metric space for viable values for our holistic security architecture. Chapter 5, defines a solution using BN and a set of trust metrics to assess the internal static assurance. Chapter 6, discusses the future work related to extending these techniques to a dynamic model. Finally, the challenges and conclusions are provided in Chapter 7.

CHAPTER 2 

2 THE CURRENT STATE OF ROBOT SECURITY

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

2.1 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 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 [9]. 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 chapter applies to the 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 [10]. 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 [11]. 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. When the phone is turned on, it authenticates with the network at which point all activities are governed by the subscriber's carrier policies. Despite these controls in place, Jover[12] was able to construct attacks on LTE and created location aware exploits using a 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, have not had this

level of maturity or the standardization since the incorporation of security capabilities and system management is not well developed. The array of sensors and the cognitive layer also set robotics apart in comparison to mobile devices and computers.

Section 2.3 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 chapter 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. Upon completing the analysis of the systematic security model for a new ROS 2 robotic system we present our conclusion in the "Conclusion" section.

2.2 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 2-1 [13], 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 uses only the static path referenced for the ROS 2 layer.

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 chapter, 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 2-2 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.

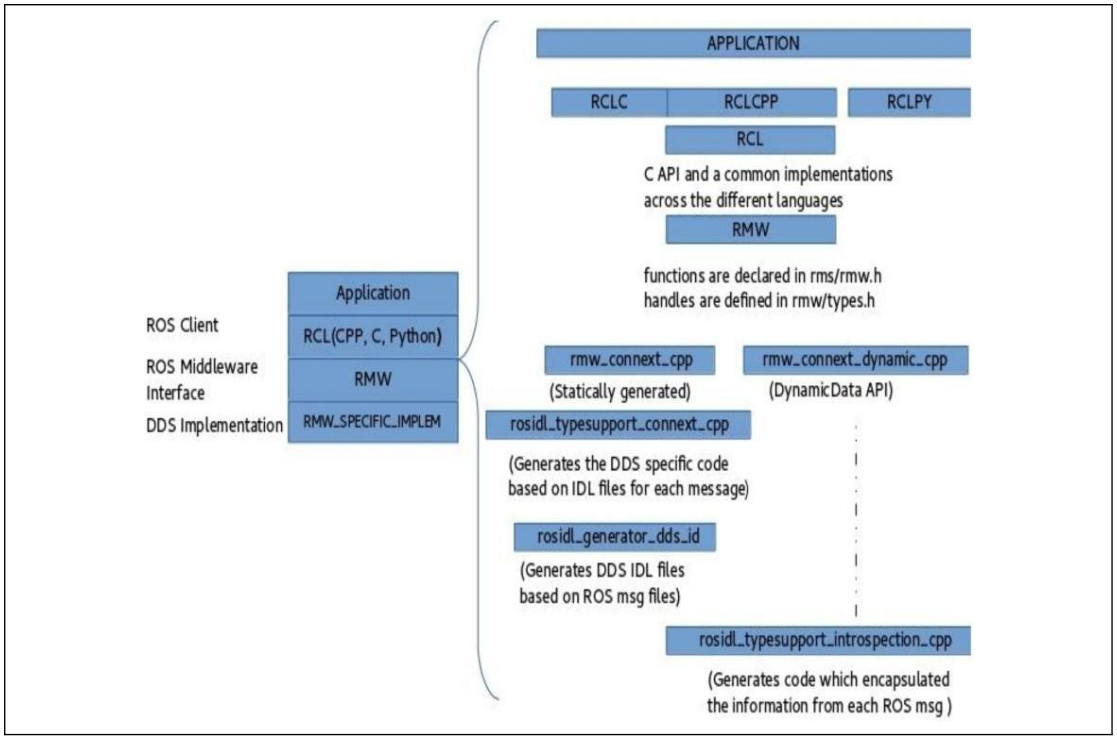


Figure 2-1: ROS 2 software architecture including the layering to DDS. ROS: Robot Operating System; DDS: Data Distributed Services.

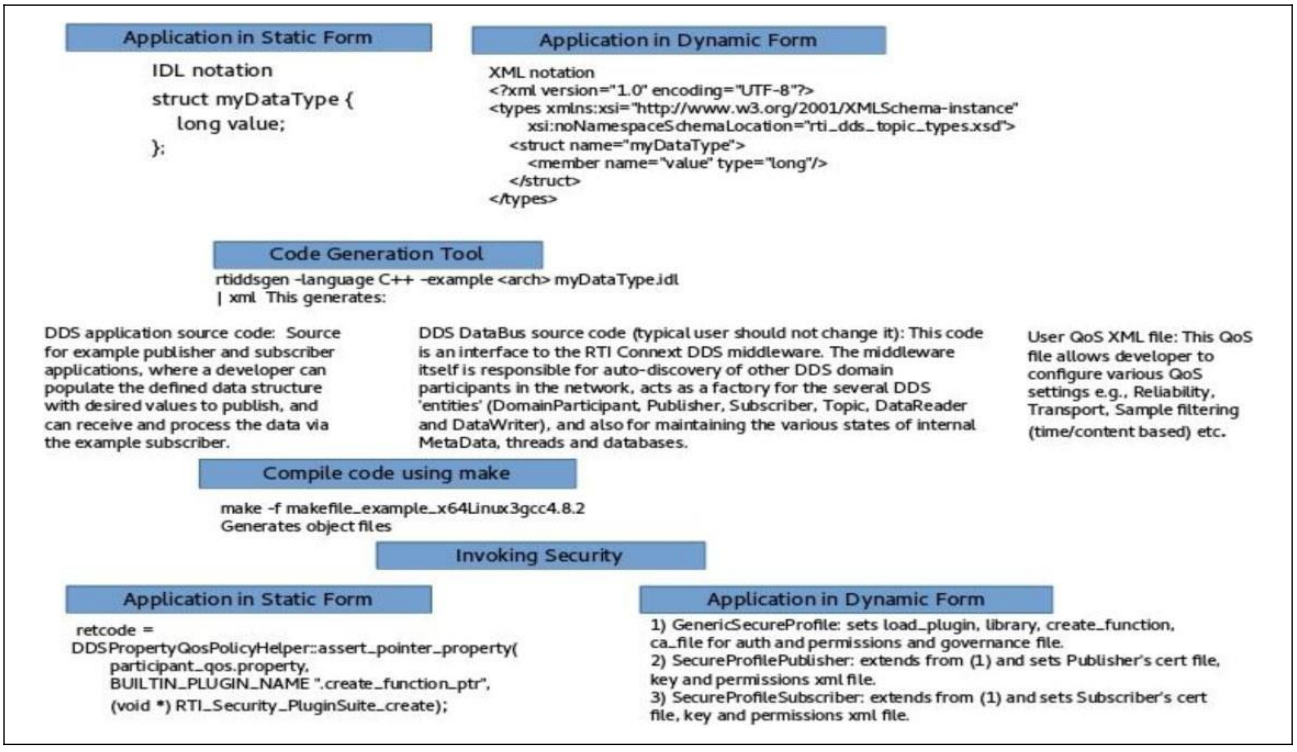


Figure 2-2: RTI software layers including security. RTI: Real-Time Innovation.

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 [14] - 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) [15] and SP 800-38D (GCM and GMAC) [16].

Asymmetric or public key cryptography [17] - 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 whomever 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 [18] is logic that takes a message and reduces it into a digest where it becomes a one way function and cannot 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 [19] falls under this category also, except that it uses a key as part of the hash algorithm.

PKI [18] 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 credible way of ensuring a person's identity.

Certificate authority (CA) - Performs the actual digital signature function, once the correct data has 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, similarly to a phone directory.

Directory - A location where certificates and certificate revocation lists [20] are stored.

TABLE 2-1: 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

The DDS security extension defines two policies [21] .

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) [16]. This is shown in Table 2-1. Enabling the security features using

the Boolean values 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 is not 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 2-3. There are five plug-ins defined by the standard: authentication, access, cryptographic, logging, and data tagging.

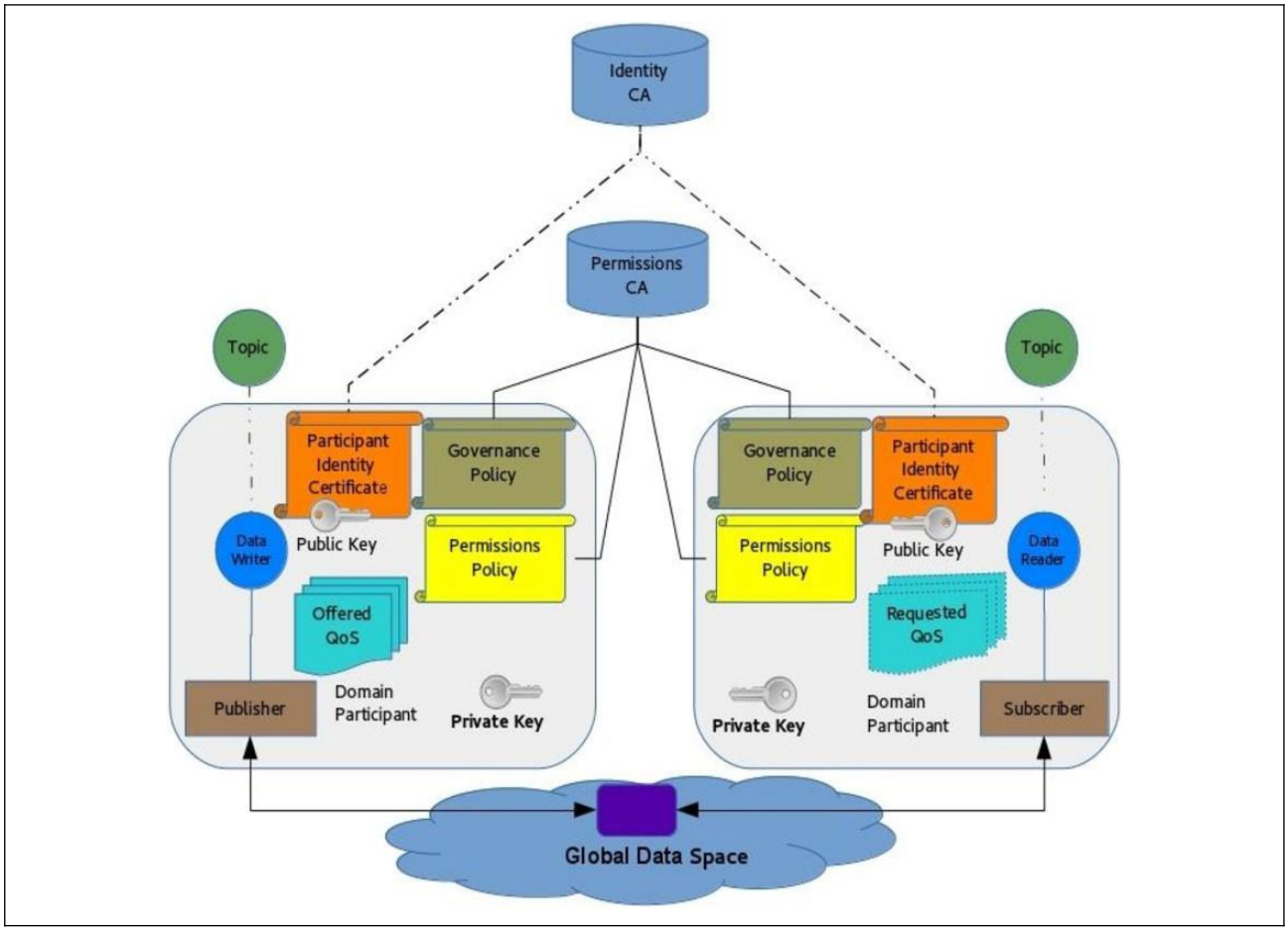


Figure 2-3: Data Distributed Services security deployment model.

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 [22] 2048 bits or Elliptical Curve Digital Signature Standard (ECDSA) [23] 256 bits. The Diffie–Hellman Key Agreement [24] 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 [25].

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 2-3 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 writers and data readers are supported by

pools of resources called publishers and subscribers. 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) [26]. 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.

2.3 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. Determining 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 2-4, 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.

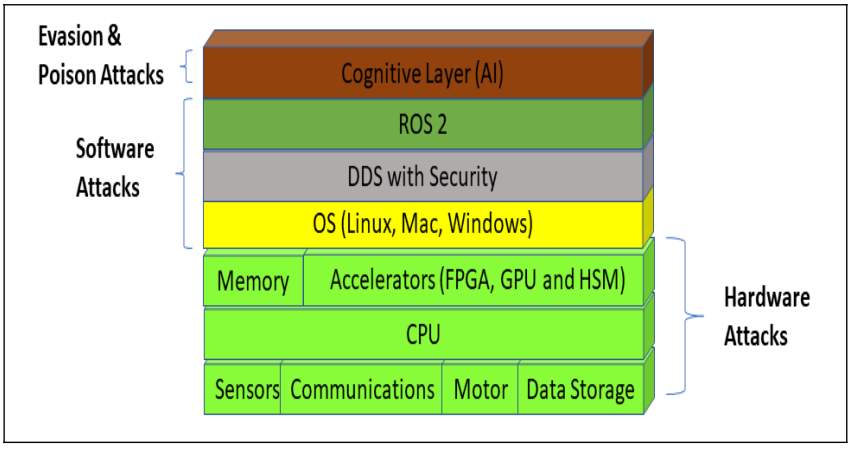


Figure 2-4: Vulnerability analysis of a ROS 2 robotic system. ROS: Robot Operating System.

A QoS profile is used to set paths to the CA and participant credentials as shown in Figure 2-3. 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 establish a secure session and checks 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 does not 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 state that using prime256v1 curves is not safe due to the elliptic-curve discrete logarithm problem being difficult and the gap of implementing elliptic-curve cryptography (ECC) security, exposing data to side channel attacks [27]. 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 [28] including forgery, [29] [30]recovery and timing attacks, [31] and nonce replay attacks [29]. 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 [32]; also the National Institute of Standards and Technology (NIST) in 2005 published that SHA-1 should not 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 does not 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 [33]. Randomizing cannot be implemented using RTPS, since it is 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 underlying 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 has not 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.

2.4 Advanced threats

We define several adversarial models for the robot threat classification from software, cognitive, and hardware (spoofing and side channels) attacks as shown in Figure 2-4. A robotic 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 units (GPUs), 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 devices, 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 acts 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 an 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 highspeed 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.[34] 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 is 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 [35] is geared toward misclassifying data as legitimate data and therefore bypassing the detection. In the case of poisoning [36], 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 [37]. 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 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 does not 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 was correct. Akdemir et al.[38] presented a number of both passive and active attacks on sensors but also provide countermeasures that can be used to aid in risk mitigation [38].

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. Naharro et al.[39] 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 [40]. 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 Petit et al.[41], applied bright light to the camera, where the sensor was blinded and incapable of capturing a recognizable image. A countermeasure to blinding was configuring multiple cameras to provide a fault-tolerant vision system. Timing attacks on LiDAR signals were performed to spoof the sensor into unknown states. A countermeasure for timing attacks may involve adding redundancy or acquiring neighbor vehicle data in vehicle-to-vehicle communications. Dithering a LiDAR's capture speed or signal pulses may also aid in providing countermeasures against timing attacks.

The classical CPU, memory, and data storage are well known for hardware vulnerabilities using non-invasive techniques, such as monitoring electromagnetic radiation (EMI) that can be used to detect 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 non-volatile 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 [42] [43].

2.5 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 2-5. The blue 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 2-1 shows 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 2-5.

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, depending 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 built-ins 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 determining 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 up to date information related to the data object and the liveliness mechanism provides

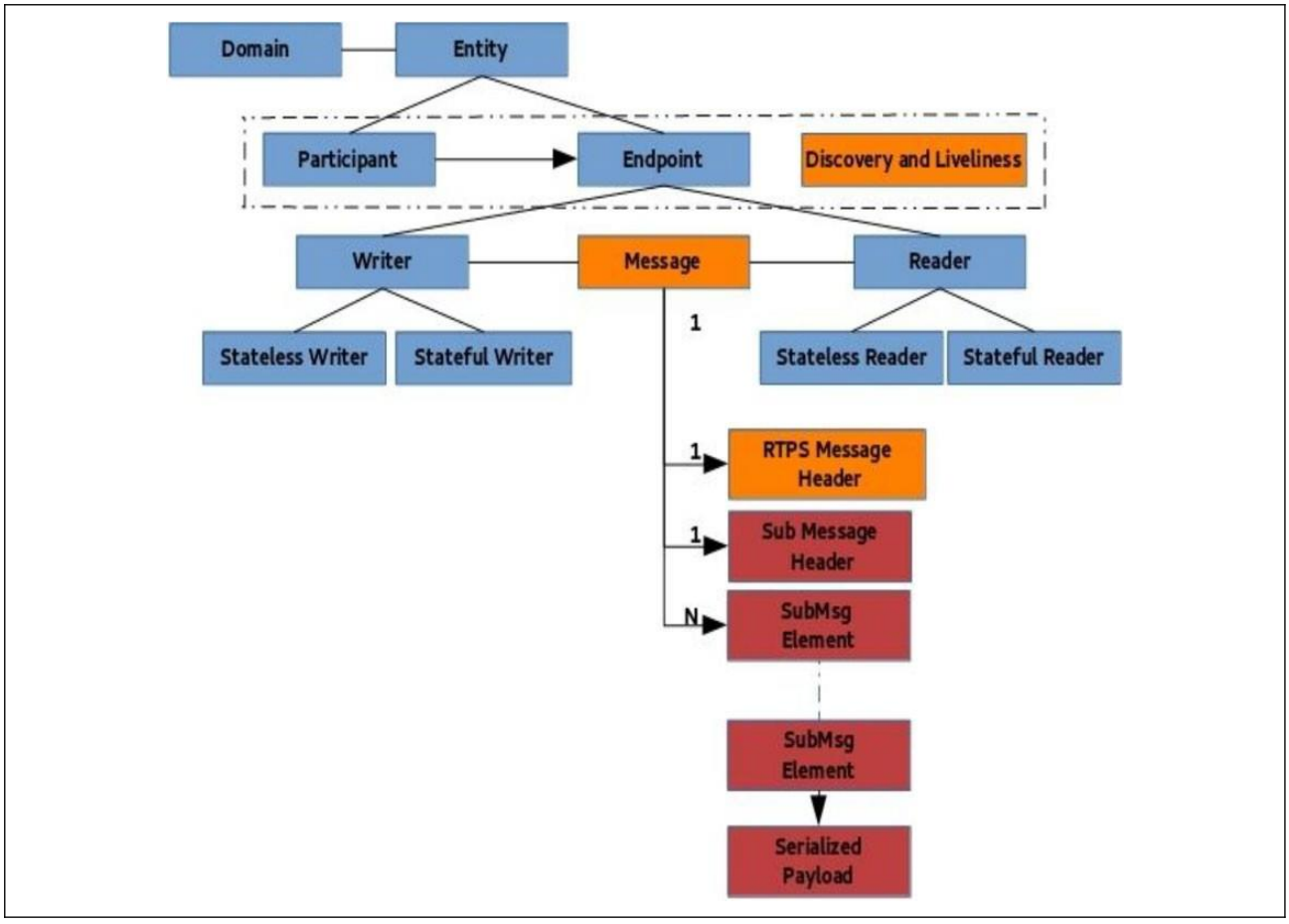


Figure 2-5: The Governance policy protection elements related to DDS_RTPS messaging. RTPS: Real-Time Publish Subscribe Protocol; DDS: Data Distributed Services.

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, FragmentNumberSet, 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 [44].

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, perftest, 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 [45]. 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 terminal 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 seconds to be consistent across the experiments. The latency measurements are one way, so these data results would need to be doubled to calculate a roundtrip 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 2-6 to Figure 2-8 shows the performance matrix related to latency, throughput, and speed. Figure 2-6 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 seconds with no security enabled compared with 160 and 143, respectively, for EC and RSA encrypted packets.

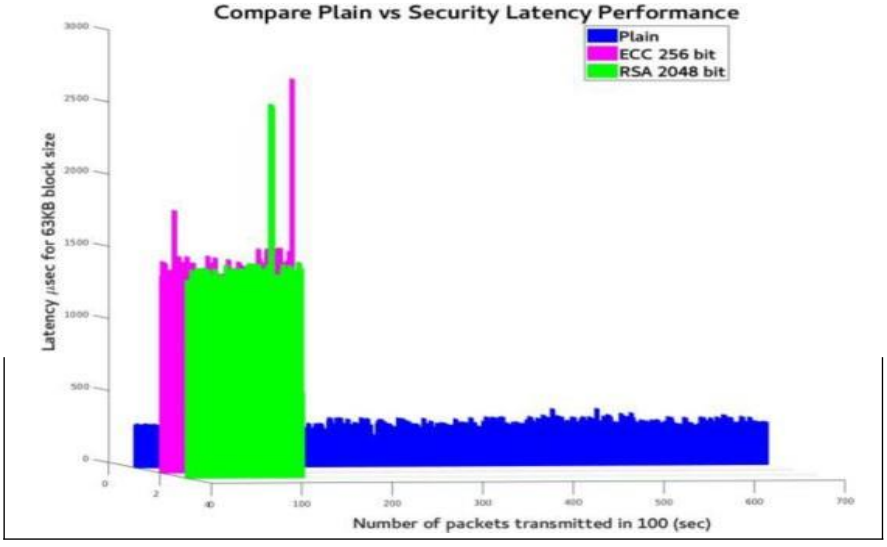


Figure 2-6: A comparison between plain versus full security enabled using RSA 2048 bits and ECC 256 bits.

This means that the overhead to enable security is 137% in latency performance and 132% in number of packets transmitted.

Figure 2-7 and Figure 2-8 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

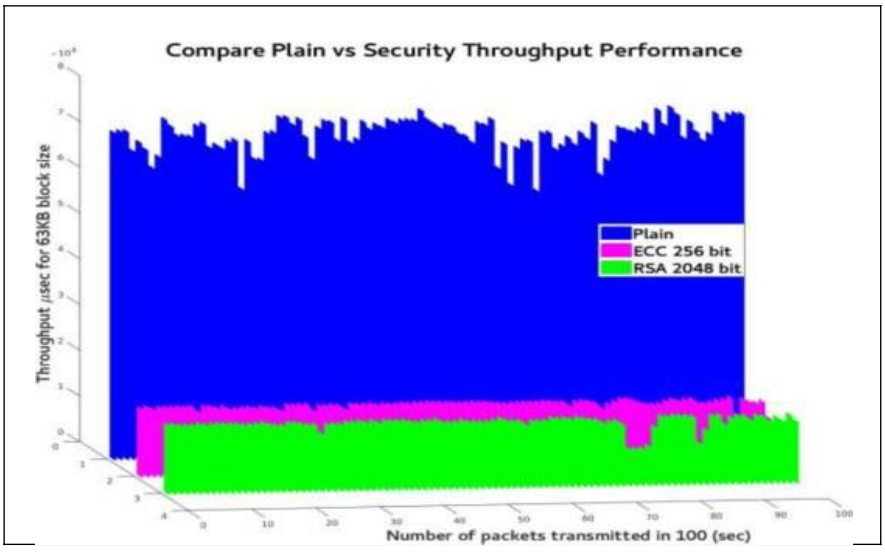


Figure 2-7: A comparison between plain versus full security throughput enabled using RSA 2048 bits and ECC 256 bits

7177.7 Mbps for RSA. This means that the overhead to enable security is 132% in throughput and speed.

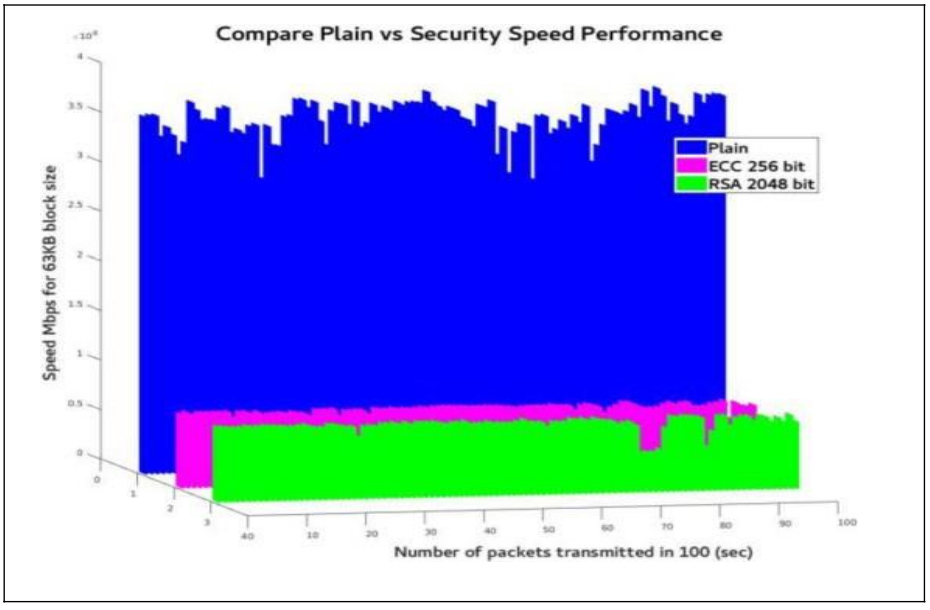


Figure 2-8: A comparison between plain versus full security speed enabled using RSA 2048 bits and ECC 256 bits. RSA: Rivest, Shamir, and Adleman.

Table 2-2 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 2-2: 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	177.7	385	4522	319.3	T	T	T	T
29	1035	0602.1	35	0619	0392.4	T	T	T	F
138	6456	294.1	082	7506	823.5	T	T	F	T
35	7665	8983.2	47	6811	8552.9	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	3462.7	T	F	F	T
320	59381	29928.4	317	60807	30647.2	T	F	F	F
1420	14265	189.6	1361	13687	898.7	F	T	T	T
761	26250	3230.2	1086	20493	0328.7	F	T	T	F
1011	19906	10033.1	1069	18818	484.5	F	T	F	T

RSA µsec	RSA p/sec	RSA Mbps	ECC µsec	ECC p/sec	ECC Mbps	SEDis	SS	SEData	SSM
552	37070	8683.3	598	34035	7153.7	F	T	F	F
1180	16749	8441.6	1212	16784	8459.3	F	F	T	T
752	26734	13474.2	758	26054	3131.7	F	F	F	T
614	28125	4175.1	56	8608	4418.5	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 tradeoffs between performance and security models can be drawn from Table 2-2 where the last line in the table represents no security and the first line represents 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 perspective on performance, we adjusted the packet data block size from 1K to 63K[1] in powers of 2^n. In Figure 2-9, the X, Y, and Z coordinates relate to latency, throughput, and speed and the color bar represents the block size. Table 2-3 shows the values for Figure 2-9.

TABLE 2-3: REMOTE DATA TRANSFER MEASUREMENTS

Measurements	Plain Data	Encrypt using ECC 256 bits	Encrypt using RSA 2048 bits
Block size (B)	1K	→	→
Latency (µsec)	14613	18751	19588
Throughput (packets/sec)	2422	1742	1987
Speed (Mbps)	19	13	16
Block size (B)	2k	→	→
Latency (µsec)	23255	39139	22366
Throughput (packets/sec)	1497	933	1122
Speed (Mbps)	23	15	18
Block size (B)	4k	→	→
Latency (µsec)	53349	54328	61270
Throughput (packets/sec)	686	566	1122
Speed (Mbps)	21	18	17
Block size (B)	8k	→	→
Latency (µsec)	65816	91531	104085
Throughput (packets/sec)	374	318	306
Speed (Mbps)	23	20	20

1 Overhead that prevented 64k

Block size (B)	16k	→	→
Latency (µsec)	158481	430446	90079
Throughput (packets/sec)	159	151	28
Speed (Mbps)	20	19	4
Block size (B)	32k	→	→
Latency (µsec)	52947	123340	130292
Throughput (packets/sec)	98	79	129
Speed (Mbps)	24	20	33
Block size (B)	63k	→	→
Latency (µsec)	208883	280883	290675
Throughput (packets/sec)	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.

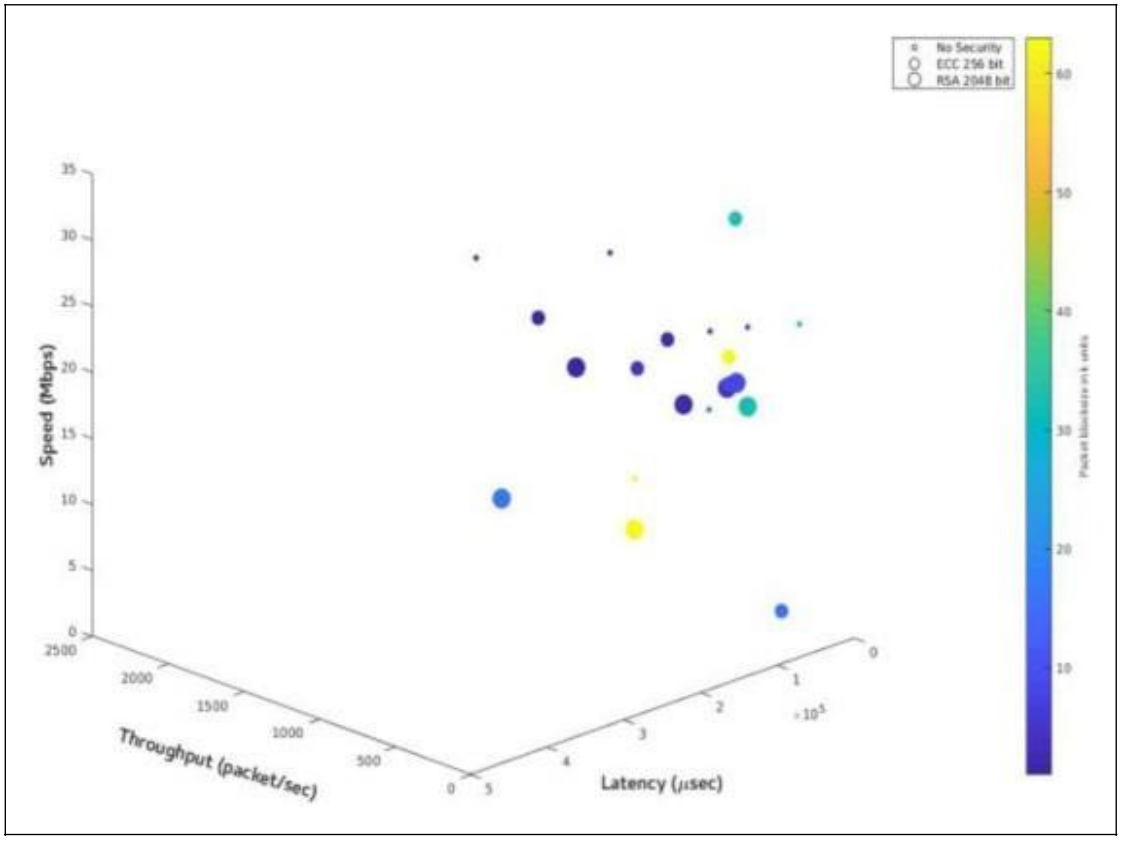


Figure 2-9: A comparison between plain versus security performance using a remote system and varying packet block size (1–63K).

The following observations were made as a result of the experiments performed. 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 which 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 the single and dual machine configurations the same, we did not add additional flags to improve the performance gain. Our goal was to observe the behavior of performance related security models. The ROS layer was not

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 determining if no security vs full security can be driven down under the 137% overhead.

2.6 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 (This is further discussed in Chapter 4 under the hardware and supply chain sections). 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 are 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. In Chapter 3, we provide two exploit examples where the first is an OpenSSL spy (dumps sensitive data) and the second, a credential masquerading using a configuration file.

2.7 Conclusion

Robotic security is a new and increasing concern and is not as mature as security for computers and mobile devices. 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 chapter, 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 [46] and RTPS [44] 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 degradation was evident when adding security features to protected data in motion as demonstrated in the performance versus security model. Table 2-4 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 2-4: SECURITY MEASUREMENT COMPARISON

Security Enabled	Latency (average µsec)	Throughput (average packets/sec)	Speed (average Mbps)
Plain	260	70772	35669
Full	1363	14382	72485

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 types 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 [1] [47] [48]. 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.

What we learned from Chapter 2 is that ROS 2 and DDS security do not provide adequate security coverage for an autonomous robotic system and enabling security has adverse effects on performance. Chapter 2 also covered a number of advanced attacks that need to be addressed in a holistic security architecture. In the next chapter, we provide examples of exploits and mitigation techniques around the ROS 2 and the DDS security model. This research provides evidence that only securing one element of the system does not protect the rest of the system and a holistic approach is needed.

CHAPTER 3 

3 THE ISSUES WITH ROBOTIC SECURITY

Credential Masquerading and OpenSSL Spy: Exploring ROS 2 using DDS security

3.1 Introduction

Modern robots are constructed with sensors, controllers, communications, motors, hardware accelerators as well as software forming a cognitive layer for processing and controlling the robot. Autonomous robots are often fully autonomous, putting their software and hardware all in one location, providing an adversary with a complete system with little, if any, physical security.

This makes physical attacks on robots much easier than attacks on corporate managed computers, since systems are typically under system management and are physically protected by the building that houses them. As robots move from factory floors into society this physical protection is removed making systems more vulnerable.

To address security in robotic systems, ROS 2 with DDS security allows online data in-motion encryption with access control protection. DDS security is dependent on the OpenSSL library and on a security configuration file that specifies sensitive data location. DSS Security assumes that the underlying Operating System (OS) is secure and that the dependencies are consistent, but ongoing integrity checks are not performed. However, off-line and on-line exploits can involve software or hardware

attacks, especially when robots are out in the wild. Research is in the early stages of investigating autonomous vehicle security [49] [50], artificial intelligence and robotics [51] while others are looking at the performance related to security [52] and creating isolation containers from memory restrictions using ARM Trustzone [53]. However, these approaches tend to focus on individual security threats and ignore viewing the threat environment as a whole; that is, taking what we refer to as a holistic approach to autonomous robot security. An example vulnerability analysis can be found in [51].

Operating systems are generally known for being susceptible to various types of exploits; a common one being the exploitation of privilege escalation [54] [55] [56] to gain access. The use of secure and trusted boot code mitigates some of these potential threats, but only up to the OS layer. In Linux, the OpenSSL library executes in the user space, so the application space is vulnerable to many exploits. These exploits include buffer overflow, timing attacks, and injected malware, just to name a few.

This chapter identifies exploits in the user/ application space related to OpenSSL and the property configuration file for the ROS 2/DDS security usage model. Several technologies are examined that potentially mitigate these vulnerabilities.

We describe four different scenarios where these types of attacks affect the behavior of the system.

- *In use case one, we have a swarm of drones that are performing a surveillance mission around a building. Each night at the same time the drones are dispatched. We will call this the spy intercept scenario. The drones have been infected by the adversary's altered OpenSSL library and now the data is being streamed to a remote server where all the data is being dumped. The adversary now has the capability to review the mission, data captured from the drones and can determine where potential schedule gaps of surveillance exists.*
- *In use case two, an adversary has placed an altered OpenSSL library on the platform and is able to siphon data from a sensor. In this example, a camera*

mounted on the robot can send data to a remote server for the adversary to view, this is called stealing services.

- *In use case three, an autonomous vehicle can be repurposed. An adversary has altered the OpenSSL library on a vehicle and now has access to the keys. This means that control of the vehicle can be achieved for late night runs when owner is asleep. The car can be returned without being noticed.*
- *In use case four, the configuration file is altered by the adversary to change the credentials in the property file, this will enable the adversary to take control of the robot platform without being noticed. The property configuration file and OpenSSL library manipulation can occur on the same platform to provide additional control to the adversary.*

The above use cases are a set of examples of OpenSSL or property file exploits. The list of exploits can grow and will enable an adversary to have control of an autonomous platform.

3.2 Summary of Results

Exploiting the usage of OpenSSL

In this paper, we show that a compromised OpenSSL library can intercept sensitive information while victim participants believe they are sending secure publisher and subscriber messages. In ROS 2/DDS, security on a Linux system is enabled by having an interface between the vendor's security plugins and the OpenSSL library. The Governance and Participant policies define the security behavior within the domain. These behaviors also define the protection kinds to be used and therefore, the cryptographic algorithms associated with any communications. The vendor's security plugin implementation is proprietary and how or what functions are being called from within the OpenSSL library is also unknown. However, by replacing the original OpenSSL library with an altered "spy" OpenSSL library, the interactions between participants become visible. Using this approach, we demonstrate that information can

be captured on a victim machine. While the demonstration presented here was done locally, a remote server can be utilized where all information is sent to it for post processing or data manipulation, as in a Man in Middle attack. As autonomous robots tend to have many, if not all, of their nodes on a single platform this type of exploit can be a severe threat. To thwart this type of attack, we identify mitigation techniques that may be applied for detecting unknown spy processes in the Linux user space software stack.

Manipulating the configuration file for misuse

Another exploit we explore is the misuse of the ROS 2/DDS security property file where security credentials are configured. This attack involves an adversary manipulating the property data using masquerading credentials. A single unprotected, configuration file supports the credentials of the CA's Certificate (authentication and document sign), participant's certificate and private key, as well as the signed Governance and Participant's policies. These configuration files point to locations on the file system where the sensitive data is located, and changes to these files are undetected; meaning that an adversary can substitute their own credentials. Depending on the CA's issuing policy, an email can be the simplest form of authentication for issuance, so an adversary can get their own credentials from the initial CA versus just replacing the configuration file with adversary data. The private keys are base 64 encoded using a privacy enhanced mail (PEM) format [57], depending on the security policy these can be password protected. However, there is no difference between how the security plugin checks for an illegal set of configuration parameters versus a

legitimate set; therefore, an adversary can change the credential parameters without detection.

3.3 Background

The Robot Operating System (ROS) 2 uses the Data Distribution Service (DDS) [58] as the transport layer. An extension to DDS is the DDS Security Standard, which provides the data protection layer on the Real Time Publisher Subscriber (RTPS) [59] data. DDS security [60] is implemented by a vendor's security plugin. The standard specifies five Security Plugin Interfaces (SPIs): Authentication, Access Control, Cryptographic, Logging and Data Tagging.

Authentication and Cryptographic operations are defined in the standard, but the implementation is dependent on the vendor. There are no test vectors or compliance tests for the security plugin implementation. The standard simply defines specific algorithms for authentication and cryptographic operations – OpenSSL [61] is a common cryptographic library for providing these algorithms.

When ROS 2 is built on top of a DDS implementation it must have all its dependent paths configured. The build process must include a path to OpenSSL or another cryptographic library. In the case of using RTI 5.3 DDS with support for DDS security on Linux, the security plugin is configured to use OpenSSL native.

An XML or YAML (Yet Another Markup Language) structured file is used to define the parameters for the security plugin. These parameters are the Certificate Authority's public keys, for identity and policies, signed Governance and Permission policy files, and the participant's certificate and private key. The file path for each of the parameters is also set in the property configuration file. Each of the policies

(Governance and Permission) enables a protection kind on the message data. Available protection kinds are discovery, liveliness, RTPS, metadata and data. An explanation of these protection kinds can be found in [51] [60]. The policies also define the access control for the domain, which includes the nodes, as well as determining who has write or read authorization.

To use a DDS implementation with ROS 2, the environment is setup by configuring the paths for the DDS implementation; on Linux this is done in the users bashrc file. The DDS configuration includes the path to the OpenSSL library. Next, the DDS environment and ROS 2 are compiled with the DDS. Since the OpenSSL library can be updated by upstream providers for Linux, this library lives in user space. In most cases, the OpenSSL library is used as a dynamic shared library so that changes to the OpenSSL library are independent of the application. In the case where the OpenSSL library is a static shared library both DDS and ROS 2 must be recompiled to take advantage of any new patches or updates.

How each participant discovers and shares information is enforced by the Governance and the Participant security policies. The transfer of data between two participants in the global data space is performed using the Real Time Publish Subscribe protocol (RTPS) [59]. The use of 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 or property files 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. The usage of the protection kinds corresponds to what

types of cryptographic algorithms are called. In order to exchange data between participants in a secure manner, each participant is initially issued an identity certificate using a public key algorithm. The authentication plugin performs the initial setup for key exchange and that is followed by security controls performed by the cryptographic plugin. The following set of cryptographic equations are given as reference (DDS Security Standard) and are discussed in the attack overview section.

For the authentication plugin, a participant is issued a certificate based on one of the following types of algorithm/key definitions, RSASSA-PSS 2048 or ECDSA 256 bits.

RSASSA-PSS is defined as [62]:

$$n = p * q \tag{3-1}$$

Where n = is the modulus, p and q = are prime numbers, and d = is the private exponent. The modulus n specifies the key length in bits. The message m, is encrypted using the private key to produce a digital signature, s.

$$RSASSA - PSS - SIGN = ((n, d), m) \tag{3-1a}$$

To verify the digital signature, the public key, e = is the public exponent is used to decrypt the message.

$$RSASSA - PSS - VERIFY = ((n, e), m, s) \tag{3-1b}$$

ECDSA is defined as [63]:

$$Qa(x, y) = d * G \tag{3-2}$$

where d = private key, G = field of points, and Q (x, y) = public key curve point

Sign operation:

$$r = x \bmod n \quad \text{(3-2a)}$$

where r is part of the signature pair, n = integer order of G

$$s = k^{-1}(z - rd) \quad \text{(3-2b)}$$

where k= random integer, z = left most bit of the hash

Verify operation

$$w = s^{-1} \bmod n \quad \text{(3-2c)}$$

$$u_1 = z \cdot w \bmod n \text{ and } u_2 = r \cdot w \bmod n \quad \text{(3-2d)}$$

$$(x, y) = u_1 \times G + u_2 x\, Qa \quad \text{(3-2e)}$$

where r = x mod n

Key Agreement is used to exchange symmetric keys using public keys. The DH key is 2048-bits MODP Group with 256-bits Prime or ECDH + prime 256 v1 as stated in the standard [60]. These are considered ephemeral keys which are temporary and only for the session.

ECDH is defined as [64]:

$$(x_k, y_k) = d_a Q_b \quad \text{(3-3)}$$

where Q_b = public key of user2

$$(x_k, y_k) = d_b Q_a \quad \text{(3-3a)}$$

where Q_a = public key of user1

x_k= shared secret between user1 and user2 is used to exchange symmetric keys.

For the cryptographic plugin, AES_GCM and AES_GMAC are used for authenticated encryption and decryption functions that are symmetric key operations. Symmetric key operations are low latency, especially when cryptographic modes are combined into an atomic operation. AES_GCM is a popular algorithm, but GMAC is a mode of GCM in which no plain text is supplied and the output is the authenticated field. GCM -128 and GCM -256-bit keys are specified in the standard.

Authenticated Encryption is defined as [65]:

$$C = P \oplus MSB(E(K, Y)) \quad \text{(3-4)}$$

where C = ciphertext, P = plaintext, MSB = Most Significant Bit, E = encryption of Y using K = key

$$T = MSB(GHASH(H, A, C) \oplus E(K, Y)) \quad \text{(3-4a)}$$

where T = tag, H = hash, A = additional authenticated data, C= ciphertext, IV = nonce

Authenticated Decryption is defined as [65]:

$$T = MSB(GHASH(H, A, C) \oplus E(K, Y)) \quad \text{(3-4b)}$$

$$P = C \oplus MSB(E(K, Y)) \quad \text{(3-4c)}$$

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 HMAC falls under this category also, except that it uses a key as part of the hash algorithm.

SHA-2 is defined as [66]:

$$\text{SHA-2} = (M) \quad \text{(3-5)}$$

HMAC is defined as [67]:

$$HMAC = (K, M) \quad \text{(3-6)}$$

The OpenSSL library has two functions, one is for the cryptographic algorithm operations and certificate support, the second is for client and server support for the secure socket layer / transport secure layer. The OpenSSL command line executable is in the `/usr/bin` directory on Ubuntu Linux. The directory `/usr/lib/ssl` points to `/etc/ssl/`openssl.cnf, which defines how certificates are created. Under the ssl directory are the certs, misc, and private directories. The certs directory has several certificates, the misc directory handles the certificate generation and private is owned by root as the key store. The header files are in the `/usr/include/openssl` directory and the shared, libssl and libcryto libraries are in the `/lib/x86_64-linux_gnu` directory. The OpenSSL software layers are illustrated in Figure 3-1: OpenSSL Architecture, starting with a set of abstracted higher level APIs, followed by

the lower level APIs and supporting utilities and the hardware interface being at the lowest [68].

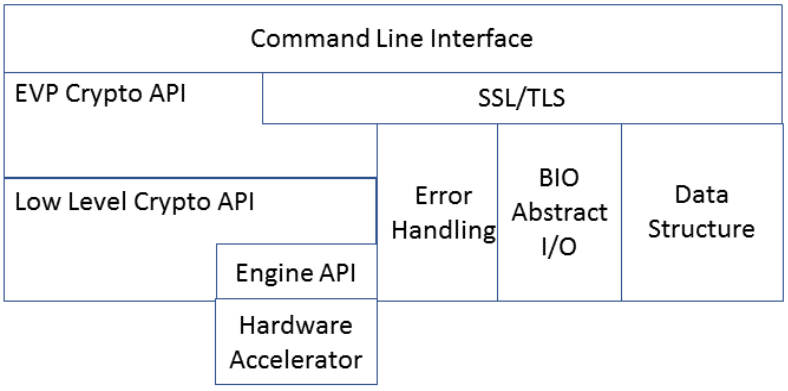


Figure 3-1: OpenSSL Architecture

The RTPS protocol standard defines the message structure for data exchange and the DDS Security Standard defines the new RTPS message wrappers that conform to the RTPS messages[60]. The wrappers provide the protection on the message structure using encryption, message authentication and/or digital signatures.

When security is enabled the RTPS messages are transformed with special wrappers and still conform to the protocol standard. Figure 3-2: Secure Transform of RTPS messages [60] shows a regular message stack on the left and the secure transformation on the right. Depending on the protection kind being specified the secure wrappers can be applied at the RTPS message, meta sub-message and/or at the payload sub-message levels.

The *rtps_protection_kind* provides protection on the entire message including the message header. Instead of protecting the entire message, a finer control can be achieved using the *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 elements 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, so depending on the security requirements different levels of protection can be achieved using the Governance policy.

As shown in Figure 3-2: Secure Transform of RTPS messages, the SRTPS_PREFIX is used to wrap a complete RTPS message and the sub-messageId is set to 0x33. This is followed by SEC_PREFIX, that is used to wrap a RTPS sub-message and the sub-messageId is set to 0x31. The SecurePayload has a sub-messageId set to the value 0x30. The counterparts to the prefixes are the postfix messages that provide a method to validate the authenticity of the RTPS sub-messages. The SEC_POSTFIX has a sub-meesageId of 0x32 and the SRTPS_POSTIX has the sub-messageId of 0x34. The SecureDataHeader contains the cryptographic transformation information and is followed by the information that authenticates the result from the cryptographic transform.

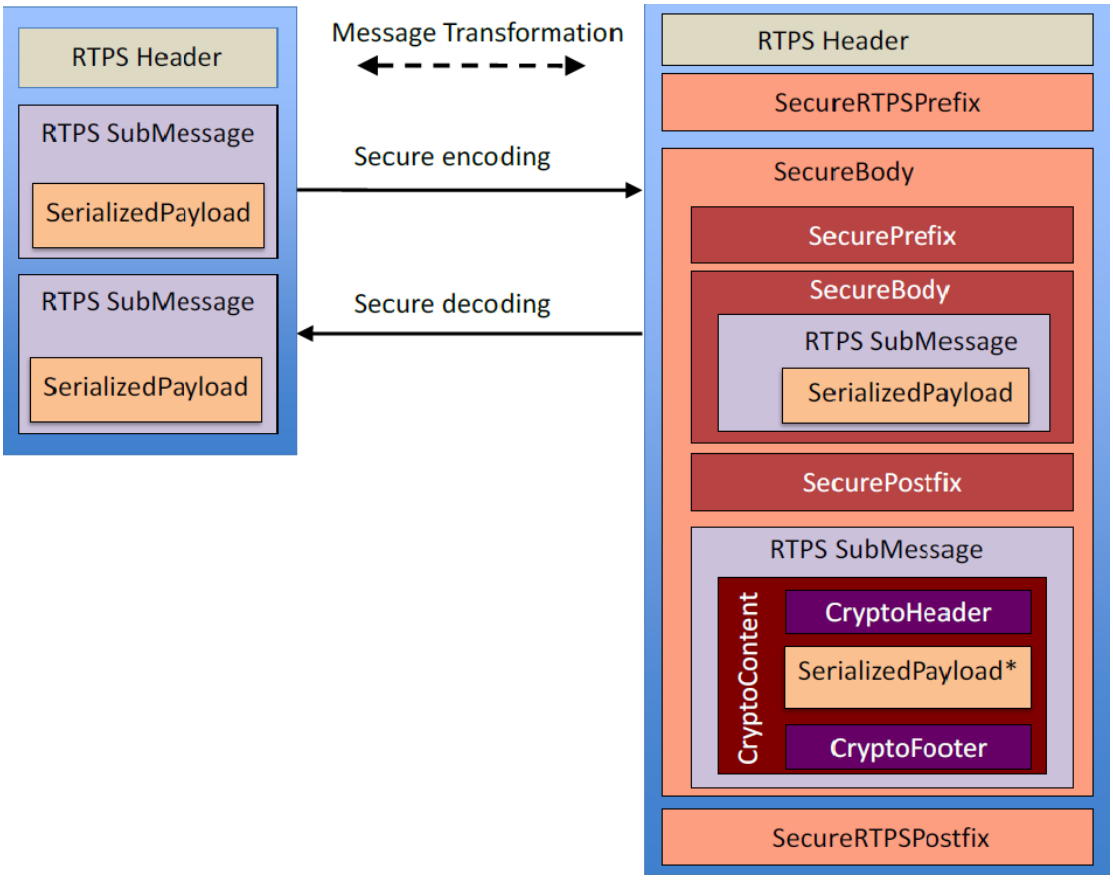


Figure 3-2: Secure Transform of RTPS messages

The set of defined built-ins in the DDS Security Standard are used to enable the interoperability between vendors. Section 7 of the DDS Security Standard defines the mappings of Entity Id values for the secure built-in data writer and data reader. Mappings for built-in participants, CryptoTransformationKind and CryptoTransformKeyId are defined in section 8, and key material is defined in section 9 [60].

Table 3-1 shows the list of partial values that are transmitted during the RTPS message exchange.

TABLE 3-1: MAPPING VALUES USED IN THE SECURE TRANSACTIONS

Mapping Name	Name	Value
Entity Ids Mapping	SEDPbuiltinPublicationSecureWriter	{{ff,00,03}, c2}
	SEDPbuiltinPublicationSecureReader	{{ff,00,03}, c7}
	SEDPbuiltinSubscriptionSecureWriter	{{ff,00,04}, c2}
	SEDPbuiltinSubscriptionSecureReader	{{ff,00,04}, c7}
	BuiltinParticipantMessageSecureWriter	{{ff.20,00}, c2}
	BuiltinParticipantMessageSecureReader	{{ff.20,00}, c7}
	BuiltinParticipantStatelessMessageWriter	{{00.20,01}, c3}
	BuiltinParticipantStatelessMessageReader	{{00.20,01}, c4}
	BuiltinParticipantVolatileMessageSecureWriter	{{ff,02,02}, c3}
	BuiltinParticipantVolatileMessageSecureReader	{{ff,02,02}, c4}
	SPDPbuiltinParticipantsSecureWriter	{{ff, 01,01}, c2}
	SPDPbuiltinParticipantsSecureReader	{{ff,01 01}, c7}
Member	PID_IDENTITY_TOKEN	0x1001
	PID_PERMISSION_TOKEN	0x1002
	PID_PARTICIPANT_SECURITY_INFO	0x1005
	PID_PROPERTY_LIST	0x0059
Cryptographic	CRYPTO_TRANSFORMATION_KIND_NONE	{0, 0, 0, 0}
	CRYPTO_TRANSFORMATION_KIND_AES128_GMAC	{0, 0, 0, 1}
	CRYPTO_TRANSFORMATION_KIND_AES128_GCM	{0, 0, 0, 2}
	CRYPTO_TRANSFORMATION_KIND_AES256_GMAC	{0, 0, 0, 3}
	CRYPTO_TRANSFORMATION_KIND_AES256_GCM	{0, 0, 0, 4}

3.4 Attack Overview

As discussed in Chapter 3, the five security plugs are provided by the vendor as object code that treats the OpenSSL library as a black box. Thus, to understand how the security services use the OpenSSL library, we examined the OpenSSL library to see what cryptographic functions were being called. The cryptographic functions are

defined in the security standard, but since OpenSSL has two different levels of API, as shown in Figure 3-1: OpenSSL Architecture, we needed to add dump routines to save data that is normally internal to OpenSSL library operation into files and use that data to identify the use of specific OpenSSL function calls. This process is not trivial as for some cryptographic functions there are initial, update and final sets of API calls to complete a single cryptographic operation.

Our first step was to add these dump routines into the cryptographic functions, specifically gcm128, hmac, e_aes, evp_digest, and ech_key. Since the property file already had the identity key values in the local directory there was no need to add additional dump routines in the code. The dump routine used was the hexdump function found in the OpenSSL library test code but modified for writing to a local file. In most cases the output from the individual routines were modified to match the hexdump function parameters. For example, *static void hexdump (File *f, const char *title, unsigned char * s, int l*) was the modified dump routine and for working with big numbers the format was converted to *BN_bn2hex().* The basic function call was *hexdump (stdout, "title", variable, variable length).* This call was placed into each of the above files and within each initial, update and final API function. Figure 3-3 shows the flow of cryptographic operations performed during authentication between two parties along the horizontal path and how data protection is performed is shown in the vertical path.

As part of the authentication, a common shared secret is established using a key exchange algorithm (equations 3-3 and 3-3a), this value is hashed and used within the CryptoKeyFactory, CryptoKeyExchange, CryptoTransform and LogOptions data structures for identity and authentication token exchange (see tables 36, 37, 38 and 39 of [60]). As a final step, the participant will digitally sign the data using his/her private key (all equations in 3-1 or 3-2 depending on algorithm and equation 3-5 for the hash operations). As part of the dump routines this data is captured to reveal the sensitive data, such as shared secret, message token parameters and private keys used to digitally sign the token data.

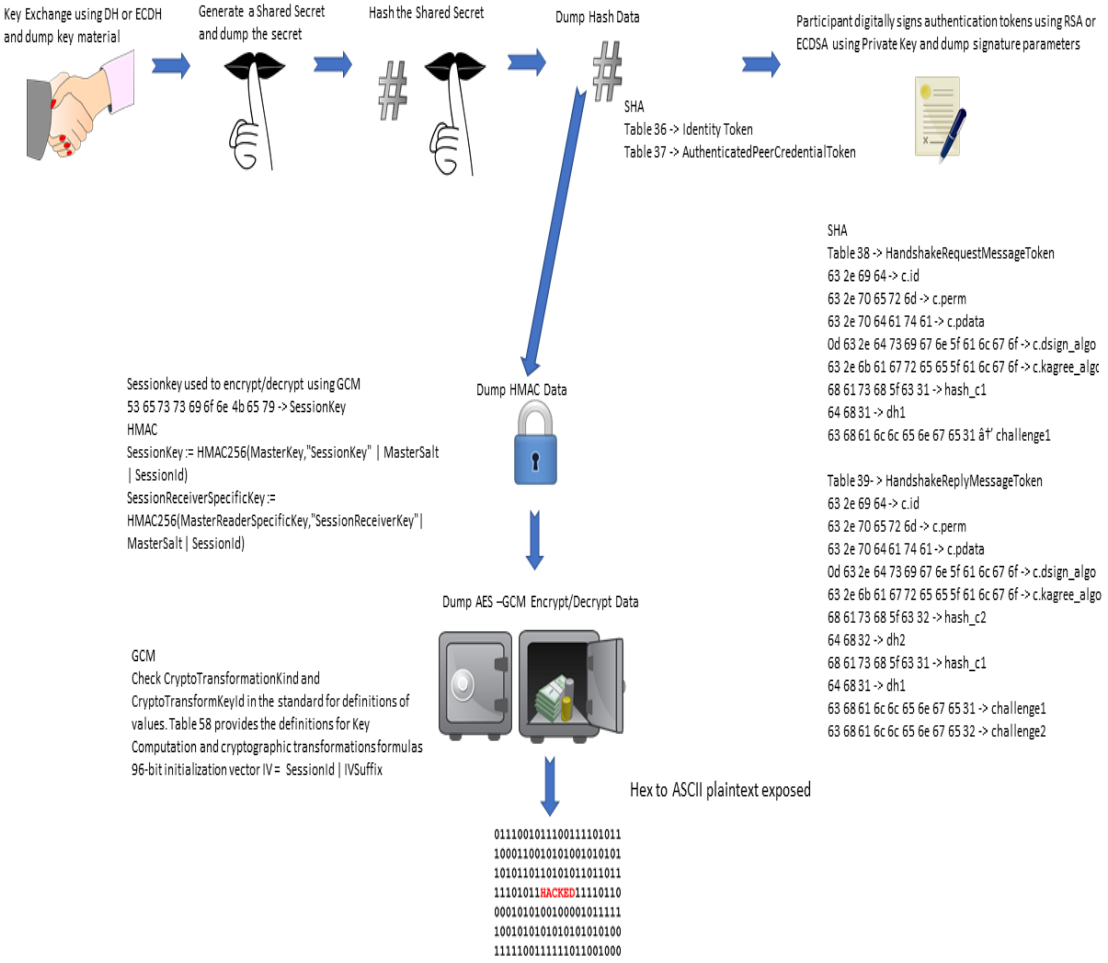


Figure 3-3: DDS security data dump flow

The vertical path is used to perform integrity and confidentiality data protection. From the hashed data branch to applying an integrity check using HMAC (equation 3-6), the data is then transformed using AES_GCM (all equation in 3-4). The key computation and cryptographic transformations formulas (see table 73) in the DDS Security Standard [60] provides key convolutions and transforms. The dump routines reveal all the data and keys used during these cryptographic operations. The output data from each of the dump routines must be converted from hex to ASCII to show the human readable data being exposed. Even though some of the sensitive data is transient, an adversary can still manipulate the data and expose a threat as discussed in the use cases above.

To confirm our data dump routines and modified OpenSSL Library performed as desired, we tested in two different environments. The first test environment was configured using a Lenovo W541 computer with Linux Ubuntu 16.04. The OpenSSL library used was 1.0.2m, but newer releases can also be used. The Real-Time Innovation (RTI) 5.3 DDS with security and RTI Perftest 2.4 [69] were used to compile and run the tests. Wireshark 2.4.6 was used to capture the RTPS packet network traffic. The OpenSSL source files were downloaded from OpenSSL.org, modified, and compiled. The files were extracted in the /usr/src/opensll-1.0.2m directory. All the changes were done in the /crypto directory and gcm128 was under the /crypto/modes directory.

To verify that the modified OpenSSL libraries were still working correctly, outputs were compared to the downloaded OpenSSL test vectors. The following commands were used to compile and test:

1. /usr/src/openssl-1.0.2m$. /configure shared (creates lib* and openssl files)
2. /usr/src/openssl-1.0.2m$ sudo make (compile)

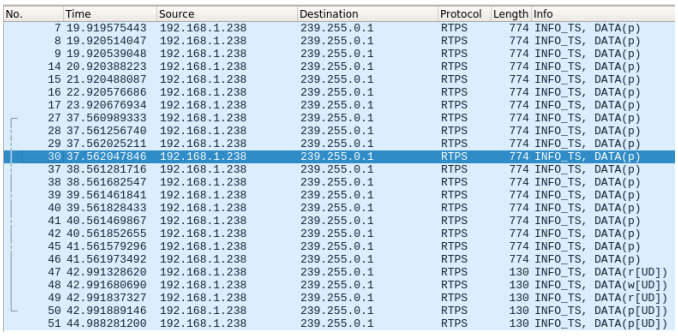
3. /usr/src/openssl-1.0.2m$ sudo make test (test)

Each of the dump routines were checked against the test vectors or against the standards being followed by the OpenSSL developers. For example, RFC 7027 for the brain pool curve test vectors. The matching of the dump routine to the test vectors gave confidence that the dump routines and formatting were providing the same results from a known reliable source. Now that the OpenSSL library has been compiled and tested, we compiled against the RTIPerftest. For this to occur we first directed the path to our OpenSSL by creating a symlink: *ln -s /path/to file /path/to/symlink* and by changing the bashrc file RTI_OPENSSLHOME = /usr/src/openssl_1.0.2m. RTIPerftest compiled using: ./build.sh –platform x64Linux3gcc5.4.0 –secure –openssl-home /usr/src/openssl-1.0.2m.

Our next step was to collect data using Wireshark, saving only those transactions that used the RTPS protocol to reduce the amount of network traffic data collected. We ran the publisher and subscriber using no security to give us a baseline with the following commands in two console terminals:

1) ./bin/x64Linux3gcc5.4.0/release/perftest_cpp -pub -datalen 63000 -executionTime 1
2) ./bin/x64Linux3gcc5.4.0/release/perftest_cpp -sub -datalen 63000

In Figure 3-5 is the result of no security shown using Wireshark.



No.	Time	Source	Destination	Protocol	Length	Info
7	19.919575443	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
8	19.920514047	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
9	19.920539048	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
14	20.920388223	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
15	21.920488087	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
16	22.920576686	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
17	23.920676934	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
27	37.560989333	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
28	37.561256740	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
29	37.562025211	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
30	37.562047846	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
37	38.561281716	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
38	38.561682547	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
39	39.561461841	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
40	39.561828433	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
41	40.561469867	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
42	40.561852655	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
45	41.561579296	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
46	41.561973492	192.168.1.238	239.255.0.1	RTPS	774	INFO_TS, DATA(p)
47	42.991328620	192.168.1.238	239.255.0.1	RTPS	130	INFO_TS, DATA(r[UD])
48	42.991680690	192.168.1.238	239.255.0.1	RTPS	130	INFO_TS, DATA(w[UD])
49	42.991837327	192.168.1.238	239.255.0.1	RTPS	130	INFO_TS, DATA(r[UD])
50	42.991889146	192.168.1.238	239.255.0.1	RTPS	130	INFO_TS, DATA(p[UD])
51	44.988281200	192.168.1.238	239.255.0.1	RTPS	130	INFO_TS, DATA(p[UD])

Figure 3-4: Wireshark detailed view into RTPS protocol transaction using secureEncryptDiscovery enabled

We use the following commands to run with discovery and liveness protection kind using encryption:

1) ./bin/x64Linux3gcc5.4.0/release/perftest_cpp -pub -datalen 63000 -secureEncryptDiscovery -executionTime 1
2) ./bin/x64Linux3gcc5.4.0/release/perftest_cpp -sub -datalen 63000 -secureEncryptDiscovery

From our dump routines there should be several files that have been created in the specified directory paths, Figure 3-5 shows the output from Wireshark where the top blue highlight is one of the secure wrappers as shown in Figure 3-2. The lower blue highlight is the expanded view that represents the encoding of the SecurePayload with a value of 0x30 using a crypto_transformation with AES-GCM 256 that correlates to value {0, 0, 0, 4} as shown in Table 3-1. The payload is followed by a closing post sub message with a value of 0x32.

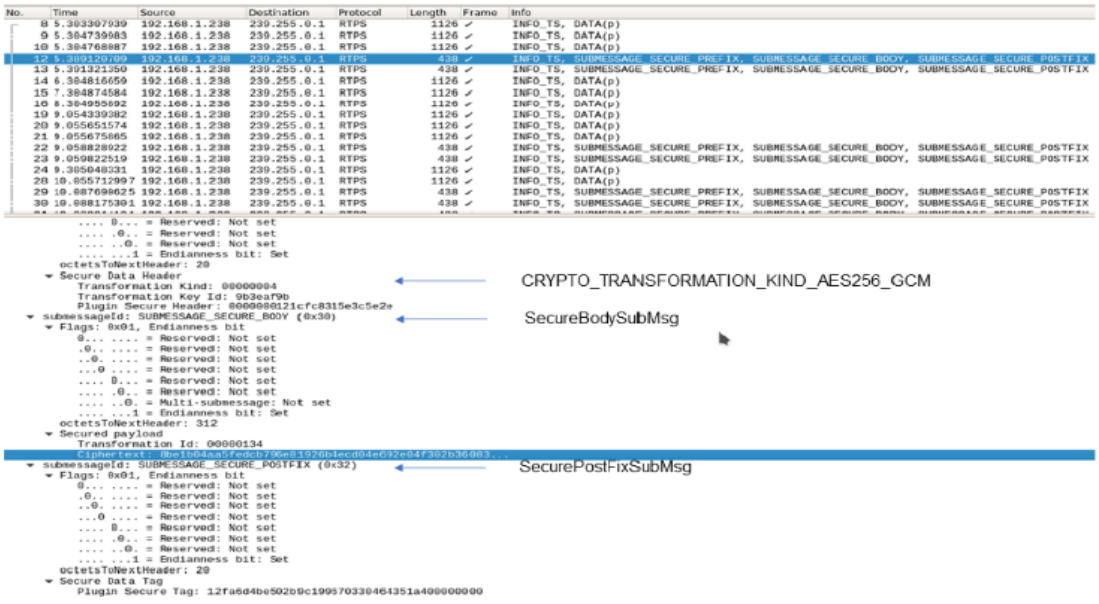


Figure 3-5: Wireshark plain run of publisher and subscriber transaction

Figure 3-7 shows the dump routine file that was generated showing the GCM key, iv and final tag. These values can be used to decrypt the messages if they are not already in the clear in another file, like the digest output file.

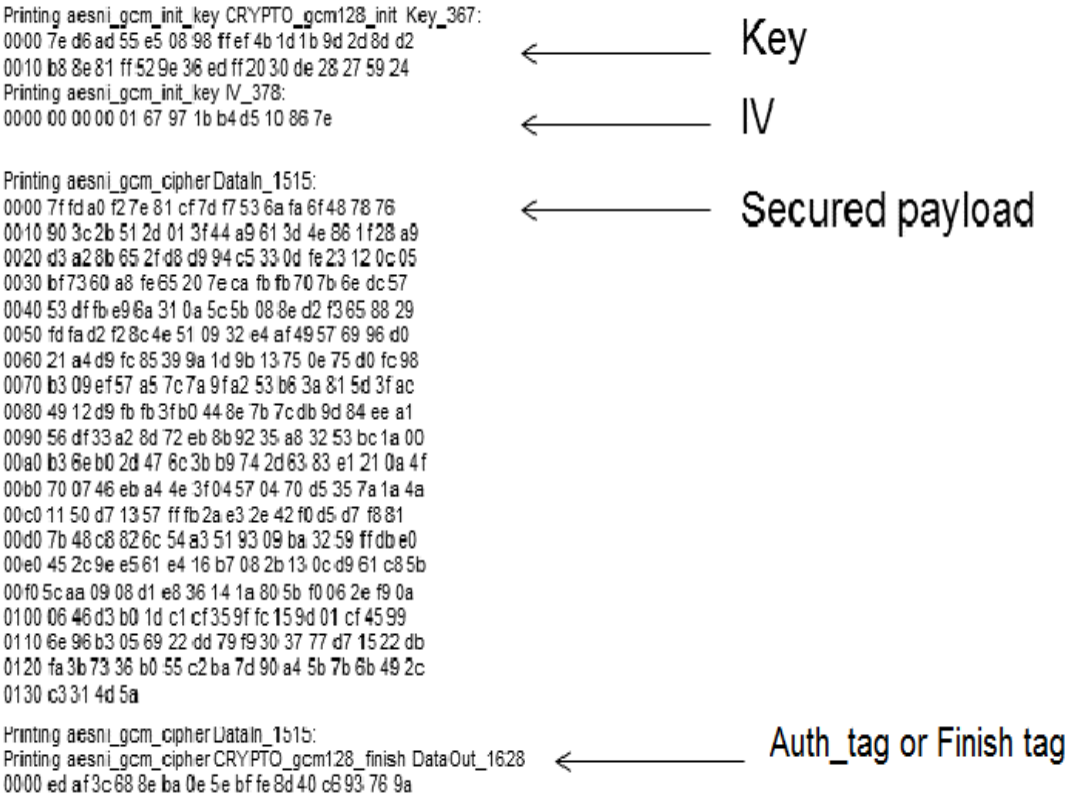


Figure 3-6: Output from dump routine that shows key, iv and final tag

Figure 3-7 shows the output from an AES_GCM 256 bit script to decrypt the data using the key, iv and final tag [70]. We show that by obtaining the relevant algorithm parameters that decryption is validated by the auth_ok_ = true and plain text being showed between the brackets in figure 7.

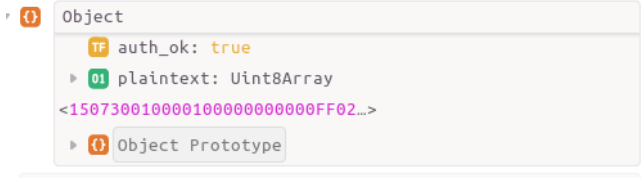


Figure 3-7: Output from gcm decrypt script

In the ecdh file this is the data captured from the key exchange between publisher and subscriber. Figure 3-8 shows the public, private and shared secret key being dumped.

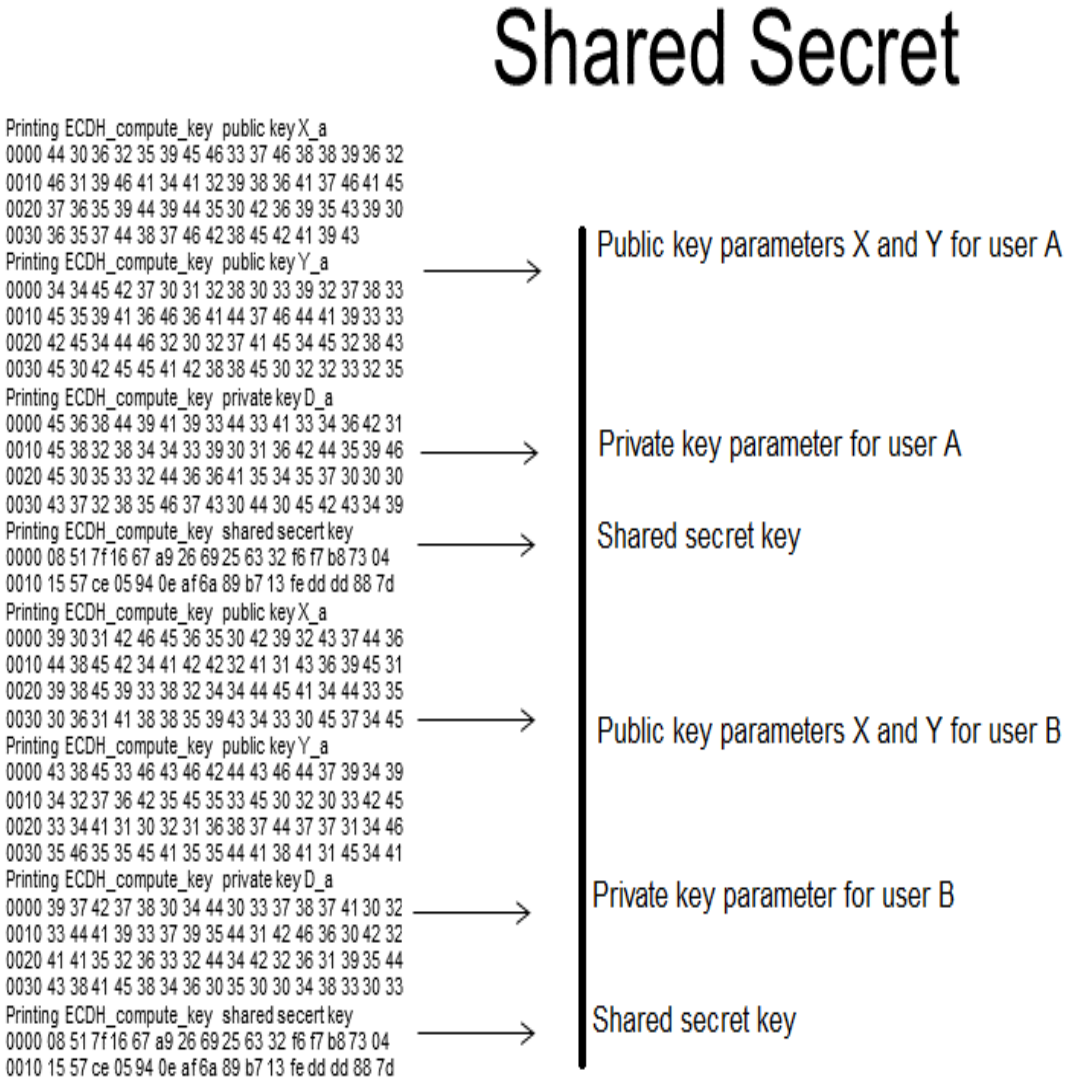


Figure 3-8: ECDH handshake between two participants to reveal the shared secret key

The second test environment was configured using a Lenovo L450 with Ubuntu 18.04, RTI DDS with security 5.3.1, ROS 2 Bouncy Bolson release and SROS enabled. We used the same OpenSSL Library as in the first test environment, release 1.0.2m that was modified. We copied the files into their respective locations. The results from our data dump routine revealed the plaintext for the “Hello World" example using SROS as shown in Figure 3-9, running the talker and listener nodes.

Figure 3-9: SROS example of talker and listener

From our output dump routines, we can exploit the DDS security and view the data being exchanged between two parties. An adversary can easily replace the libcrypto.so.* and libssl.so.* files in the /lib/x86_64-linux_gnu directory using an escalated privilege exploit. The dump routine can be easily extended to write the data to a remote server. Validation was performed using Ubuntu 16.04 on a ThinkPad® model L450, where the files libcrpto.1.0.0 and libssl.1.0.0 were in the /lib/x86_64 directory. These files were replaced with our spy libraries and symlinks were created to libcrypto.so and libssl.so from these two libraries.

libcrypto.so.1.0.0 -> libcrypto.so
libssl.so.1.0.0 -> libssl.so

We copied the openssl file into /usr/bin, which provided the configuration needed to capture the data being used by RTI DDS security plugin.

The basic concept of using DDS security is to have all participants use public key and have the policy rules digitally signed. This level of trust is established at the Certificate Authority since both identity and document certificates are issued by a single entity. Depending on the deployment model different CAs can issue identity and document signing operations. The validation in a two-party exchange is performed by having the chain of trust, this means that the issuing CAs public key must be available during a public key validation operation. When security is enabled, each participant must have their identity credentials and the signed Governance and Permission files on the same platform. Since autonomous systems are self-contained, the credentials and policy rules are also stored on the platform. A property file defines all the participants (publisher and subscriber) credentials, identity CA's public key file, document signed CA's public key, and signed policy rules (Governance and Permission) located within

a directory. An example of a property file is shown in Figure 3-10 where the credential locations are defined. The security plugin is defined at the top of the file, followed by the CA certificate and PEM data. The domain Governance file location is defined followed by the participants (publisher and subscriber). In each of those sections the permission file location is defined and as well as the private key locations. This example is for a dynamic linking security property file and a static method can be used but would need to be compiled into the code.

Figure 3-10: Property file example for a publisher and subscriber

Since changes in credentials and locations will occur, the static method is less flexible since this requires the program to be compiled for each change. In this example the CA's publisher, subscriber and signed files are known by the security plugins, since these files are parsed for the required information to perform authentication, authorization and cryptographic operations using the parameters with them.

However, no checks are performed on these files, so that manipulation of the parameters or the files themselves can be achieved by an adversary. An adversary can masquerade the credentials with their own set or change the policy rule files with system parameters / topic names to be self-signed. We see this as a serious vulnerability, since a property file can be altered and no checks are in place to detect the tampering or credentials being replaced.

3.5 Evaluate Mitigation Options

We will evaluate several technologies that might mitigate one of the two vulnerabilities and provide a recommendation for both. The first technology is the Trusted Platform Module, the second being the Security Services for DDS security plugins using ARM TrustZone and the recommendation is the combination of Integrity Measurement Architecture (IMA)/Extended Verification Module (EVM) with the TPM.

Several industries are using the Trusted Platform Module (TPM), part of the Trusted Computer Group (TCG) to establish root of trust for the PC/Server market. This is being extended into the mobile and Internet of Things as proposed by GlobalPlatform [71] and the Industrial Internet of Things Security Framework [72]. The introduction of using TPMs in robots is still a novel thought. Trusted boot is the process of taking

measurements during the boot process from firmware to Operating System (OS) and validating the measurements against a known good set of values by a 3rd party. The TPM is a small microprocessor like a smart card but has a different structure for how hash values are stored in platform configuration registers (PCR). Figure 3-11 shows the TPM 2.0 structure with support for newer algorithms including Elliptic Curve cryptography and SHA 256 bit.

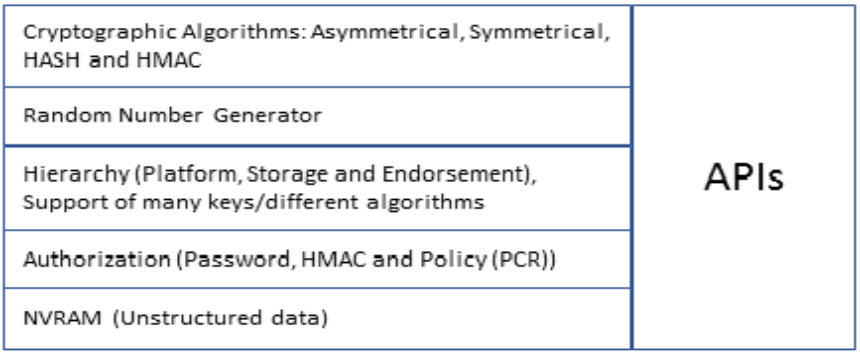


Figure 3-11: Trusted Platform Module 2.0

The TPM functionality can be implemented in software as well but eliminates the hardware protection features found in some manufacture's products. In either case, static or dynamic root of trust measurements are up to the OS executing. In the case of static case PCR (0 to 7) are used in the boot process from Power on Reset (PoR) to OS and in the dynamic case PCR (17 to 20) when the x86 instruction halts the processor into a known state [73]. Table 3-2 provides the layout structure for the TMP as shown below [73] [74].

TABLE 3-2: PLATFORM CONFIGURATION REGISTER LAYOUT

PCR Number	PCR Value
0	BIOS
1	BIOS Configuration
2	Option ROMs
3	Option ROM configuration

4	MBR (master boot record)
5	MBR configuration
6	State transitions and wake events
7	Platform manufacturer specific measurements
8 to 9	Static operating system
10	Integrity Measurement Architecture (IMA)
11 to 15	Static operating system
16	Debug
17	DRTM and launch control policy
18	Trusted OS start-up code (MLE)
19	Trusted OS (for example OS configuration)
20	Trusted OS (for example OS Kernel and other code)
21	as defined by the Trusted OS
22	as defined by the Trusted OS
23	Application support
24	

The purpose of the trusted boot using a TPM was for network admission, meaning that before the computing node was granted access to the network it was validated against Policy Enforcement Point and Policy Decision Point entities. The TMP is used to store the hash values collected during the boot process and then digitally sign a quote (all the hash values) that was sent within a Tunneled Network Connection protocol. The known good values or golden measurements were stored and validated by a 3rd party verifier and the result of the compare were sent to the PDP to remediate or allow the compute node to gain access. The remediation process for example, is to update the compute node with the latest software patches or image. Once updated the compute node would validate and gain network access. This remediation process only works for systems with network access. Other models are using the TPM for secure boot like, bitlocker in Windows. In TPM 1.2 the chip did not offer any physical protection and defense was weak. In some vendor implementations the TPM 2.0 has physical protection at the die level that adds a tamper resistant mesh, this helps with

hardware side channel attacks. The TPM can be used to validate specific application PCR values, but this needs to be implemented with custom software to generate and validate the values.

We have discussed the trusted boot process of using a TPM for establishing a root of trust during system bring up. We now move into a concept called Trusted Execution Environment. The ARM TrustZone provides the memory management to be partitioned and restricted for secure applications. ARM has been working toward Security Services for DDS security plugins using ARM TrustZone [53] [75]. Figure 3-12 shows a flow from normal to secure world where the SecureLib (new name libddssec) communicates to the security libraries running in restricted memory space. The architecture Cortex-A (applications 32/64-bit architecture) and for Cortex-M (embedded 32-bit architecture) the TEE layer might be removed with direct firmware communications. The exception levels, EL0 - user space application, EL1 - privileged OS, EL2 - Hypervisor, and EL3 - firmware/security monitor. Cortex-A supports TrustZone with Memory Management Unit and the Cortex-M supports TrustZone with Memory Protection Units (MPU). Physical memory is divided into Normal or Secure by setting NS bit within the Translation Lookaside Buffer (TLB) for all system memory on an A architecture, where the MPU is programmed for different regions. Physical memory size on an M architecture is 4GBytes, where on the A architecture it can grow by adding RAM. Secure world has access to Normal, but not the reverse. A context switch is performed from normal world to secure by a calling an API. The M35P is an interesting chip with claims for anti-tamping protection against side channel attacks.

While the trusted execution environment vendors make claims that they are secure, researchers who are continuously scrutinizing TEEs have discovered several

vulnerabilities. There is a position paper that points out limited functionality within the secure region for: no mechanism to verify execution code, no defense within the secure region, no detection mechanism, and no mitigation when compromised [76]. Other work has been focused on side channels, power management and cache timing attacks on ARM processors with TrustZone [77] [78] [79]. Intel has its version called Software Guard Extensions (SGX), but this has been compromised by leveraging the speculative execution bug [80]. Also, from a performance point of view, it was covered in [51] that enabling security added latency, as well as throughput and speed overhead to each of the transactions. By performing context switches between the normal and secure worlds, this will surely add additional performance penalties for saving the state data to registers, startup, teardown, and data validation process between the worlds; this is a consideration for real-time constraints. Since the SecureLib work that ARM is working on is to enable the security plugins to live in the secure world, this would help mitigate against the OpenSSL dump routine during execution time but does not cover the data at rest or off-line attacks, for example a cold boot attack [81].

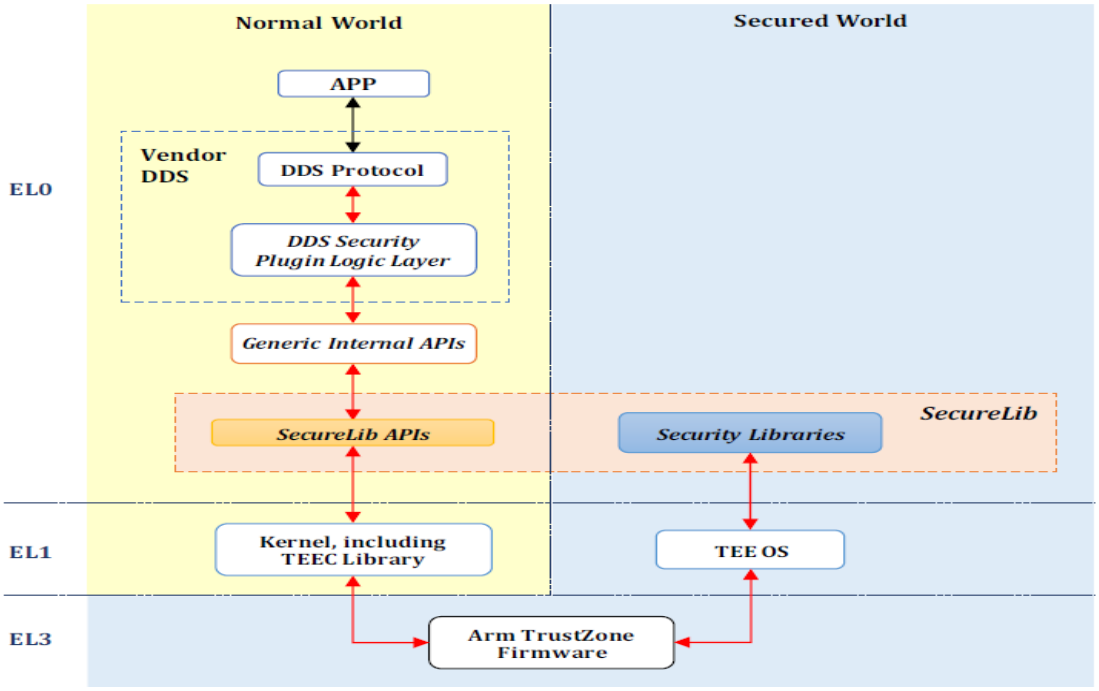


Figure 3-12: Security Services for DDS security plugins using ARM TrustZone

Our recommendation on Linux is to use IMA/EVM, since this technology has been up streamed and supported in the kernel [82]. IMA maintains a run time integrity list and is anchored to PCR 10 in the TPM. By extending the PCR 10 for each file listed in the policy, each measurement is aggregated into a value. This makes an attack difficult since all the sequences and values must be known to reconstruct the result. Features of IMA include [83]:

Collect – measure a file before it is accessed.
Store – add the measurement to a kernel resident list and, if a hardware Trusted Platform Module (TPM) is present, extend the IMA PCR
Attest – if present, use the TPM to sign the IMA PCR value, to allow a remote validation of the measurement list.
Appraise – enforce local validation of a measurement against a "good" value stored in an extended attribute of the file.
Protect – protect a file's security extended attributes (including appraisal hash) against off-line attack.
Audit – audit the file hashes.

Table 3-3 shows a full software stack that includes the platform trusted services used to request quotes from the compute node, trusted software stack used as the interface for TPMs, IMA/EVM and the rest of the trusted boot as mentioned above [83].

TABLE 3-3: FULL STACK OF PTS, TSS, TPM AND BOOT PROCESS USING TCG SPECIFICATIONS

Software Layer	Specification	Interface
Application	PTS	OpenPTS, TPM-Tools
Libraries	TSS	TrouSerS
Linux Kernel	TPM-2	IMA, EVM, TPM Driver
Boot	BIOS	GRUB-IMA, TBOOT
Hardware	TPM	Software TPM

By using the features of IMA, store and protect, the two templates below provide an example for each. Store provides a hash for the file to be generated, while protect adds a signature. An example of using an ima-ng template [83]:

PCR template-hash filedata-hash filename-hint
10 91f34b5c671d73504b274a919661cf80dab1e127 ima-ng
sha1:1801e1be3e65ef1eaa5c16617bec8f1274eaf6b3 boot_aggregate

Another example using an ima-sig template [83]:

PCR template-hash filedata-hash filename-hint file-signature
10 f63c10947347c71ff205ebfde5971009af27b0ba ima-sig
sha256:6c118980083bccd259f069c2b3c3f3a2f5302d17a685409786564f4cf05b3939
/usr/lib64/libgspell-1.so.1.0.0 0302046e6c10460100aa43a4b1136f45735669632ad ...

Additional information can be found in the kernel.org [84] about the IMA template management module. Protection against offline attacks is provided by the trusted boot and the IMA/EVM as well. Since IMA is a run time process, files are being checked constantly for changes. Having both OpenSSL and the property file under this type of control will mitigate several attacks, like the ones we demonstrated above.

3.6 Conclusion

We have presented two attack vectors related to ROS 2/DDS security and have identified four different use case scenarios that are plausible. In each use case the adversary was able to obtain control of data or direct manipulation of the platform using the modified OpenSSL library and/or the configuration file with credential masquerading. In the first and second use cases, the data was either decrypted by possessing the correct keys or directly reading the plaintext data from the dump routines. In the third and fourth use cases, either having the possession of the keys and/or credential masquerading could have achieved success by the adversary.

This paper presented the two attack vectors and compared technologies to help mitigate the risks. Even when files have been downloaded and checked with a hash, that one-time check does not provide the safeguards against ongoing threats. We believe that IMA/EVM can help mitigate against these threats, since it is a runtime and offline security set of features to protect files. The trusted boot does provide checks on the lower levels of the software stack to enable the OS to boot with a root of trust mechanism. While the SecureLib running in ARM TrustZone is an interesting protection mechanism, it only accounts for two of the five DDS security plugins. The need for a holistic security solution still needs to be considered for robotic architectures since the new movement is toward autonomous. Enabling a TPM within a robotic platform and how attestation will be performed are new concepts that need to be extended beyond the traditional mechanism of trusted boot.

We believe that research is still in early stages to using TPMs in robotic systems. Some robotics system TPM considerations are; how it will be managed, who will perform the attestation, who owns the golden measurements and what types of

remediation will be put in place are still unknown. Robots are not IoT devices like some industry vendors are claiming and that a one solution cannot be for all. Robots are moving into a cognitive learning phase where limited data is needed to learn and evolve within their environments. This is quite different from an IoT device.

The next chapter discusses a trust metric survey that was conducted to determine if a security solution already exists for autonomous robotic systems. This survey also helped to flush out the definition of what a holistic security architecture should look like. For the holistic architecture, each of the system, hardware, software, Cognitive/AI, and supplier layers are explored to determine what are meaningful trust metric values and what techniques can be used for modeling the security architecture.

CHAPTER 4 

4 SEARCHING FOR A SOLUTION

A Survey on Trust Metrics for Autonomous Robotic Systems

4.1 Introduction

We have surveyed the trust metric space related to evaluating systems, hardware components, software components, cognitive-layer robustness as well as vulnerabilities introduced in the supply chain and have come to realize that no current set of metrics for assessing system trust fully spans any system architecture, let alone autonomous robotic systems. The overall complexity of performing assessment and the complexity of identifying potential security problems are bad enough in a controlled environment; now add high-value targets in an unconstrained environment and they get much worse.

Defining trust metrics for system security is difficult [85] leading many practitioners to only define metrics for small portions of an overall system. This approach, while making the assessment of a system more tractable, can result in security vulnerabilities being undetected. Existing approaches to evaluating trust rapidly become computationally intractable due to the large number of interrelated variables that must be considered. Thus, there is a need to come up with a way to make evaluating system's trust more computationally feasible.

We also observed that many approaches tend to focus on a small stovepiped set of problems which are more-or-less tractable. While that body of work provides a useful

foundation, it omits several factors that are important to consider when taking a holistic approach to trust evaluation. For example, neither loss (the cost of damage from an exploit) nor reward (the benefit of exploit) costs were addressed for all facets of a system, even though these costs have an effect on the level of trust that can be assigned to a system. In our opinion, there is a need to have a secure base before trust can be extended with external entities by evaluating an internal representation. A secure base is the set of security features (hardware and software) supported by the platform. Once those features are understood, the system can internally analyze and evaluate how to react to external stimuli such as external requests received through network interfaces. In essence, this internal evaluation relies on a model of the security features and how those features interact to detect and/or mitigate requests that may have malevolent intent. Such a model requires metrics to rate the level of trust for the best possible security posture. The metrics used in this model help establish the chain of trust depending on the configuration and features of the autonomous robotic system.

Autonomous robotic systems, such as autonomous vehicles operating within a dense population environment, can be very complex. Such a robotic system is typically constructed from many individual components including hardware (CPUs, sensors, actuators, accelerometers, systems of systems), software (firmware, OS, services, cognitive layer, application specific logic), and AI components (learning algorithms). These components may be sourced through one or more supply chain vendors. The integration of all these components becomes a complex problem that significantly effects the trust model.

For example, components of a system may be open source, Commercial Off the Shelf (COTS), or custom built. An individual component may work as designed, but

when integrated with other components the behavior of the composite system may be unexpectedly altered in a way that exposes security side channels. We believe that a holistic trust metric that accounts for these complexities is beneficial to determining the security posture of a robotic system.

So, there are two main problems in the art that must be addressed. One is simply defining trust and determining ways of establishing the level of trust that can be attributed to a system and/or components of a system. The other is bounding the computational complexity of a system trust mode such that the problem can be solved in a reasonable amount of time with reasonable resources.

There are many definitions of the word "Trust" in the context of system security. One definition that captures the essence of the term is that "trust" is "the firm belief in the competence of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather it is subject to the entity's behavior and applies only within a specific context at a given time" Azzedin and Maheswaran [86].

In their earlier work, Castelfranchi and Falcone define the concepts of internal vs external, or global trust, with regard to the cognitive social trust model[87]. They further breakdown internal trust into two areas called reliance and disposition, where the former is the ability, competence, and self-confidence; the latter being the willingness, persistence, and engagement to fulfill a task. The external, or global, part of the trust model is to have the opportunity, for the quickest fulfilment of the resources and the success without having interferences and adversities. Other work validates the concepts of internal vs external trust as an example of work from internal trust Devitt's model [88] and Henshel, et al.[89] contribution validates the need to separate trust into two categories.

It is important to differentiate these two types of trust concepts when modeling the behavior of autonomous systems as further decomposition can be made in the security validation and protection architecture. Viewing trust from an internal and external perspective can enable the decision process to be different when analyzing the security capabilities of an autonomous system. A system that has more security capabilities might be better suited for requests being made from an external entity vs one that has less capabilities. This type of internal rational is different from today's computer security models where no introspection is performed on the decision being made from an external request. Today's computer security model is to authenticate first then authorize; once authenticated the process or application is granted access to fulfill the request. The term introspection is the capability to look inside and analyze or debug.

Most literature focuses on external communications from the perspective of an agent interacting with another, or a group of agents. This interaction builds a trust evaluation mechanism based on prior evidence and ratings from external entities, or recommenders. Hearing from multiple sources builds confidence about an outcome or decision that others have made. Therefore, that accumulated evidence can sway a choice or further aide in a choice. Some trust rating systems targeting the use of referrers, or recommenders, are discussed in [90], [91], [92], and [93]. However, it is necessary to be cautious when applying these evaluation techniques to an autonomous system.

Historically, the notion of trust is defined based on what is being trusted. For example, only focusing on the individual system architecture, hardware, or software layers of a system. In general, trust is assessed by comparing features of the thing being assessed to a set of guidelines, and this comparison may be done at a high (abstract) or low (detailed) level. High level assessments tend to be qualitative in nature, while lower

assessments tend to *appear* quantitative. However, one problem that arises from performing assessment at low levels is that an ad hoc picking of features or weights can result in even low-level assessment being fundamentally qualitative. Further, applying low level assessment techniques to large portions of a system can become expensive, both in terms of engineering time and computational resources.

Our motivation for this work is to construct a security assessment framework that first performs the equivalent functions of static analysis and later have the capability to be extended into the form of dynamic analysis for autonomous robotic systems. Security assessment can be performed at the system level using Common Criteria methodology as one approach, which is a static approach that has many paper/human tasks. Alternatively, software uses automated tools to perform static/dynamic analysis. By mimicking the automated analysis tools in the software model, this can be leveraged for our consideration.

The term static analysis is common in today's software development process since the analysis is performed during development/testing or during build cycles. These static analysis tools support many flavors of development languages, such as C++/C, Java, and Python. They work by scanning the source code for potential bugs or security vulnerabilities against a set of rules. These pattern rules can check for syntax issues, buffer overflow issues, and null pointer dereferences to name a few. Some tools can perform the same analysis on object code. Common tools like Coverity and Veracode perform these types of analysis. To extend the static analysis for software, there are dynamic tools that perform similar functions, but at runtime. These dynamic tools look for flow control failures, memory errors, and race conditions to name a few. A common tool like Valgrind performs these types of analysis. Taking the concept of

both static and dynamic analysis to determine potential security risks on an autonomous robotic system, needs to encompass the holistic architecture model, and not just software.

It is our belief, that developing a security analysis tool to perform a system-level trust assessment is a way that provides useful results in a tractable manner. A more comprehensive security posture can be created when internal and external interactions are separated, but not decoupled. A self-assessment can be performed on an external request or goal before being fulfilled. This internal checking may uncover potential vulnerabilities, misbehavior, or lack of ability to perform a task.

Representing an internal system model may require a large amount of data and may become computationally complex. A Bayesian Network (BN) is a probabilistic graphical model where nodes and arcs define the casual inference using probabilities. Using a BN is one potential solution, as it satisfies the local Markov property where the joint probability is reduced to a compact form, allows the combining of supporting evidence or negating hypotheses. This helps to reduce the computational complexity issue. We also bound the problem by limiting the number of parent nodes in the model.

The goal of this chapter is to survey the current techniques for evaluating trust, and to extend them in a structured way both to account for the complexities of autonomous robotic systems and to minimize subjectivity. Chapter 4 is structured as follows: section 4.2 discusses the search results; the trust metric evaluation for layers system, hardware, software, Cognitive/AI, and supply chain can found in sections 4.3, 4.4, 4.5, 4.6, and 4.7. Section 4.8 describes the assessment of the techniques used in security assessments and this is followed by a solution outline in section 9. We conclude our findings with developing new trust metrics for an autonomous robotic system.

4.2 Search Results

So, how does one evaluate or assess trust? In order to perform assessment, there needs to be a metric, or set of metrics, that correlate well with trust and a tractable methodology to evaluate the metric(s). This has generally been done by assigning weights to features and then combining those weights in some way to develop a score that is presumed to correlate with the level of trust assigned to the system being evaluated.

Trust evaluation can be viewed at different levels for computer systems. At the top is the system level architecture, which then breaks down into other individual categories. For purposes of this discussion, we focus on, without limitation, the hardware and software subsystems of the overall system. This structure is shown in Figure 4-1 where the three circles represent the different levels.

For evaluating trust in a computer system, the Trusted Computer Evaluation Criteria (TCEC) standard was created in the 80's which disclosed a methodology for evaluating computer systems known as the "Orange Book." The Orange Book defined

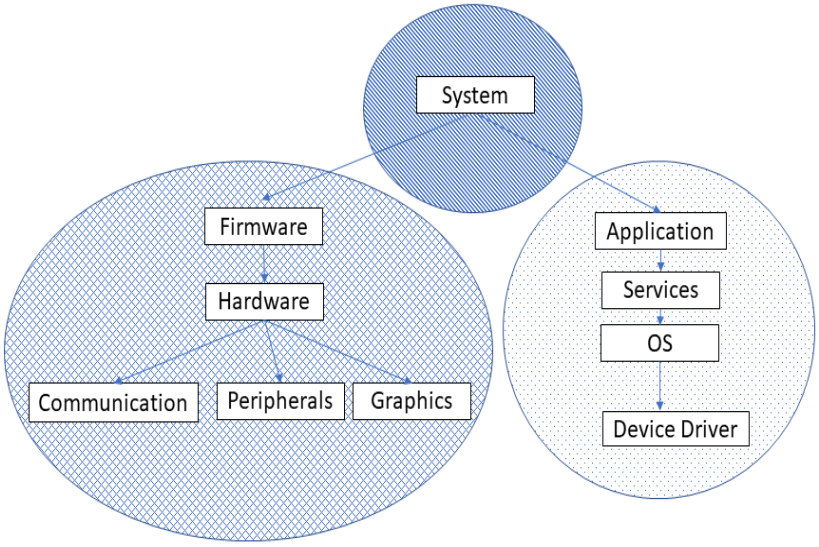


Figure 4-1: Different approaches to trust at system, hardware, or software levels

four primary levels, where A was the highest, followed by B, C and D (being no security). Around the same time another standard was introduced called the Information Technology Security Evaluation Criteria (ITSEC) which defined seven levels (E6, the highest to E0, no security). The levels of ITSEC were mapped to the Orange Book levels that included all the associated sub-levels. The Canadian government created their own version of a security standard and so, to minimize proliferation of standards, the International Standards Organization created the Common Criteria (CC) standard that tried to harmonize these different standards into one, called ISO-15408. The CC has levels ranging from E7 (highest) to E0 (no security). The higher levels of evaluation incorporate formal methods for system behavior verification, auditability of development artifacts and certified 3rd party lab testing.

Using these standards, systems are evaluated based on the architectural features they contain. A system architecture can be evaluated by reading the description of the features required by a trust level and comparing those features to the features of the system being evaluated. Conversely, if a system designer wanted to achieve a desired

level of trust, those descriptions provided a list of required features to achieve that level. Since this approach has some level of industry acceptance, it is desirable to capture elements of this method of scoring system-level trust and expanding on it to capture additional elements determined to be important as well.

As is the case at the system-level, similar approaches exist for evaluating trust in the software components of a system. When evaluating software systems, different methodologies have been proposed to ensure a secure software development lifecycle process (SDLC). This means that security tasks are included in the development process and not as an afterthought. These tasks may include code review, vulnerability analysis, white hat testing, and utilizing code scanning software for static and dynamic analysis. Incorporating these techniques in the development process is less expensive than finding security vulnerabilities after release. Some examples of SDLC are the software assurance maturity model (which is derived from the capability maturity models), the software security framework, the System and Software Integrity Levels from ISO-15026, and the Common Criteria (CC). Following an SDLC process requires tightly integrating the process into the organization's software development philosophy and training their software development resources. Depending on the level that each of these development processes use, a set of tasks are checked off to ensure that they were completed. Software is then evaluated based on these development processes and, as an artifact of the process, the results can be audited.

As we saw at the system level, existing methodologies for trusted software development also rely on formalizing a set of criteria that are applied to the software development process. Again, these criteria form an industry-accepted basis for creating a trust metric that is based on objective criteria.

Not surprisingly, hardware development is taking steps similar to those used in software development to ensure trust in the hardware development process. This is logical since, in modern hardware design, software tools are commonly used to synthesize (or at least customize) hardware components using a variety of hardware design/description languages and other computer aided design tools. Different methodologies are used at this level including the DARPA-led "Trust in Integrated Circuits" program where ICs are manufactured in certified labs to ensure their pedigree, as well as incorporating defined security tasks into the hardware development lifecycle. Indeed, some Electronic Design Automation (EDA) tools are now incorporating security tasks into their tool suites. For example, Mentor Graphics TrustChain provides a unique id that is embedded in the chip and enables tracking in the chip's lifecycle. These methodologies provide an evaluation mechanism for assessing how/if security considerations are being incorporated into the hardware lifecycle.

Thus, the existing art provides some examples for which trust evaluation occurs at the system, hardware, or software levels. While these previous approaches focus on evaluating on a specific level, this may create gaps in the overall security posture of a system. For example, GreenHills INTEGRITY separation kernel was evaluated at EAL 6+ (only software was evaluated) but included a configuration with an embedded PowerPC and PCI card. This evaluation only examined a small portion of an operating system that did not include other services like device drivers or application logic that would be part of a system architecture. Nor was hardware included that might expose side channels into the kernel itself. Traditionally, we have focused the evaluation of trust on just the hardware and software levels of the computer system, but with new autonomous robotic systems, this needs to change.

Autonomous robotic systems are different from computer systems in general since they may have multiple compute nodes, and are a system of systems, which may include sensors, actuators, and AI operating in unconstrained environments. So, what is a holistic approach to defining a set of security metrics for evaluation? It is our recommendation that incorporating metrics for the system, hardware, software, AI, and supplier, defines a good basis for evaluation. Further, by developing a computationally tractable methodology for trust evaluation that includes metrics spanning all these systems, we form a foundation that can be extended to include additional system aspects not discussed herein.

Figure 4-2 shows an example of an autonomous robotic system broken down into components. The left side of the figure represents the software layers including the cognitive or Artificial Intelligence (AI) layer, which is expressly identified as a component of the system. The right side represents the hardware layers and the "supply chain", which may influence any part of the system. Components like sensors and actuators are linked to IoT devices since these devices have similar trust issues. Several components did not have a direct mapping to a specific category and were placed into a major grouping (system, hardware, software, AI robustness or supply chain).

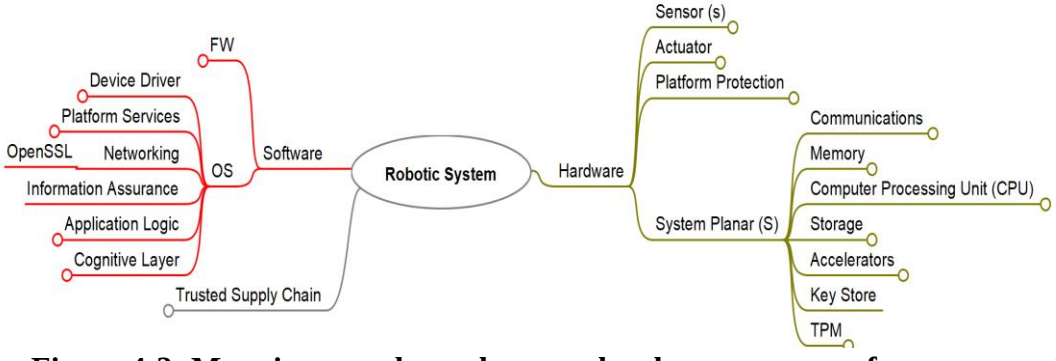


Figure 4-2: Mapping search results to each robot component for trust metrics

For each system component, we searched relevant terms to generally identify the existing art using terms like BNs, trust metrics, risk assessment, and vulnerability assessment for autonomous robotic systems. Other terms were added for each system specific aspect. We then categorized that art according to its primary contribution(s) to modeling the system architecture aspect that were conducive to our goals. The primary contributions were further analyzed to extract the main points that aligned to our objectives, while discarding unrelated findings in a second pass. The litmus tests that we targeted were related to the method(s) for utilizing trust metrics: how were the metrics created, were the metrics linked to standards or recognized bodies of work, and did the findings prove successful. Those that passed were incorporated into this document, which are further analyzed in section 4.3.

We began our search for trust metrics by attempting to identify those methodologies currently existing in the art that have already been proposed for evaluating trust, even if those methodologies only solved a portion of what we see as the overall problem. In that investigation, we specifically looked for trust evaluation methodologies applied at the system level, as well as methodologies applied at the hardware, software, cognitive/AI, and supply chain levels of a system design.

4.3 System Level Trust Evaluation

To identify methodologies used for evaluating trust at the system level, we searched for papers focused on trust metrics, autonomous systems, BNs, trustworthiness, system security, resilient systems, and security evaluation. From these parameters the results are broken down into four groupings: risk assessment, trust with

a human in the loop [94], a trusted boot scheme [95], and standards (that included frameworks and security evaluation methodologies [96] [97] [98] [99] [100] .

We summarize some of the system search results that are attractive to incorporate into our holistic security model. These findings include the need to: support attack paths, determine behavioral states of an autonomous system, and integrate the concepts of trust, resilience, and agility. The system model should also incorporate attributes for physical protection and safety, since these elements will be needed for autonomous robotic systems that are exposed to and interact with humans. In the remainder of this section, we briefly describe what these various techniques are, how they work, what they do well relative to our goals, and any shortcomings of the technique when applied to autonomous robotic systems.

In our research we found that techniques fell loosely into two categories: graph-based assessment techniques that incorporated metrics and system security assessment techniques. Exemplary graph-based techniques are found in the work of Shetty[101], Henshel et al.[102], and Cho et al.[103].

For example, Shetty presents a technique to categorize attack paths (a chain of exploits) utilizing a BN where exploit impact, cost of exploit, and the degree of difficulty is determined [101]. An attack graph is a visual representation of potential paths that an attacker can take or has taken to achieve a successful goal. It is important to autonomous robotics systems because defense strategies can be used to block the threat ahead of a potential attack.

What is it - Shetty's Cyber RIsk Scoring and Mitigation (CRISM) tool is described as interfacing with the Common Vulnerability Scoring System (CVSS) and the National Vulnerability Database (NVD) to create trust metrics. These metrics are

derived from the CVSS base score as low (0 to 4), middle (4 to 7), and high (7 to 10). The base score reflects the severity of a vulnerability based on its intrinsic characteristics like the exploitability (attack vector) or the impact to the data's confidentiality, integrity, and availability. A further explanation is given in the software section below.

How does it work – As a first step, an attack graph is generated from network scans, internal enterprise vulnerability tests, internal enterprise vulnerability database (s) for how the systems are connected and network topology. Essentially, this step is to create the topology of how components are connected in the network-based system architecture. The National Vulnerability Database (NVD) is a repository which provides CVSS base metrics on Common Vulnerabilities and Exposure (CVE) entries. Once a vulnerability is discovered, it is assigned a unique CVE Identifier. For example, CVE-2014-7173, a brief description (FarLinX X25 Gateway through 2014-09-25 allows command injection via shell metacharacters to the files sysSaveMonitorData.php, fsx25MonProxy.php, syseditdate.php, iframeupload.php, or sysRestoreX25Cplt.php) and a score of 9.8. The score is derived from the CVSS base metric, including the exploitability (attack vector) and impact (confidentiality, integrity, availability). Only system component vulnerabilities that are associated with the attack graph are reported. NVD assigns metrics that are based on CVSS base scores (low 0 to 4), middle (4 to 7), and high (7 to 10) but only the impact scores are used. A risk probability is calculated depending on the lifecycle of the vulnerability (not yet discovered, discovered, patch available, patch not applied, patch applied, etc.). By combining the risk probability and CVSS impact score, a set of values can be constructed using the information from the attack graph/topology where component

security metrics are categorized by attack paths with attributes that include the impact, cost, and degree of difficulty. A risk assessment tool is created that captures the attacker's exploits.

What does it do well – From CVSS base metrics (see Software Metric section) NVD assigns a ranking low, middle, and high for vulnerability risks. This provides an established set of data sources and metrics for known attack vectors and paths taken, which are then assessed using a BN. This concept of risk metrics can be extended into different layers of a robotic system using BNs.

Shortcomings - CVSS/NVD provides a good framework for systems where the attack vectors are known, but since autonomous systems are relatively new, the attack vectors are not yet completely known. Yet, many parts of a current robotic system can already use this information to at least assess parts of a system. Furthermore, CRISM provides a framework for including these attacks as they are discovered.

Henshel et al., presented a cyber security risk assessment model that characterized "Trust" as human (users, defenders, and attackers) behavior and all other as "Confidence" (hardware and software).

What is it - Trust in humans had two categories, the first being inherent (part of the individual that is further broken into behavioral and knowledge/skill characteristics) and the second being situational (external to the individual). These two categories define how mental states affect risk and impacts the levels of trust [89]. The human factor was being incorporated into a cybersecurity risk framework/model called Multi-Level Risk Assessment Parameterization Framework [102] using BNs.

How does it work - The risk assessment taxonomic parameterization framework (i.e., MulRAP Framework) is applied by first identifying the complex

system in question. The complex system is then deconstructed into its functional components and processes. The level of deconstruction is determined by the granular specificity of the risk assessment question. The functional components and processes (i.e., system/risk parameters) are then characterized based on the environmental context of the risk assessment question and the known vulnerabilities of the complex system in question. A BN allows the factors that contribute toward high-risk situations to be identified. These risks are identified as type of activity from a country, threat index for the country, risk prior to defense, risk after defense, potential access, and network component compromised. Two scenarios (risk of a database being compromised, one with low to medium risk level and the other with high) were given that demonstrated the outcomes from the BN.

What does it do well – Henshel et al., does a good job of identifying the various elements that go into evaluating trust assessments from human factors and incorporating them into an evidence-based model. It combines information from human factors, and it provides a model for capturing evidence-based causal relationships between elements.

Shortcomings – The focus of the paper was on the parameters for defining human characteristic in a cybersecurity model. The framework/model was discussed in another paper as referenced above, but while a direct mapping between humans and machines (human characteristics to machine characteristics) may be invalid, the methodology applied for assessing trust is beneficial. By breaking down human trust further between inherent and situational characteristics, this provides the basis for reducing risk.

Cho et al. presents a technique using Petri Nets that covers trust, resilience, and agility for multi-domain environments called TRAM. The focus is mainly toward human and machine conflict.

What is it - TRAM consists of trust (security and dependability), resilience (fault-tolerance, recoverability, and reconfigurability), and agility (Service Level Agreements (SLA)) elements for measuring system quality in a multi-domain environment. This multi-domain environment consists of hardware, software, network, human factors, physical environments, and the effects of behavior at a component or system level for an overall service.

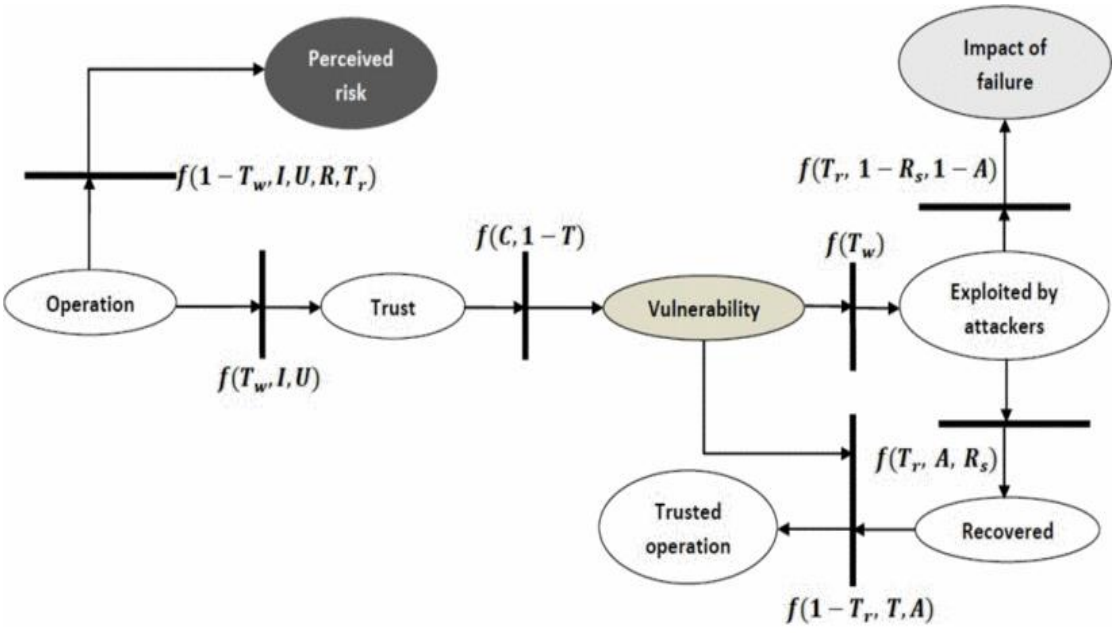


Figure 4-3: A Petri net representation of the relationships between metric attributes, assessments, and threats.

How does it work - A Petri network diagram in Figure 4-3 shows where metric values, assessments, and threats are links of relationships. T_w refers to trustworthiness of a system with $T_w = (T_r, T, R_s, A)$ where T_r is the degree of threat, T is perceived trust, R_s is resilience, A is agility of the system, and metrics range from [0,1] Cho et al.[103].

What does it do well – – The TRAM model ties the relationship of Trust to other variables like resiliency and agility. Autonomous robotic systems will need to support the concept of resilience and agility, since they may defend themselves from threats and still be able to function in a limited capacity.

Shortcomings – – While the paper claimed that the TRAM system included hardware and software assessment, no further definition was included nor was there an explanation about where these modules came from. In autonomous robotics systems, hardware can refer to a number of things including compute modules, sensors, and actuators. These are not simple extensions to a conventional computer system connected to a network. While regular Petri network diagrams can be applied to threat models, they do not provide a mechanism for evidence-based reasoning about uncertainty. Informed decisions or reasoning can take place when data is available to support a hypothesis; this is called Evidence-based reasoning, but reasoning about uncertainty is represented as probabilities. Therefore, the concept of linking the relationships between actors can be applied in a BN to provide a better security assessment in the presence of uncertainties.

System security assessment assures that the security requirements are met in the implemented system. The techniques for the assessment can range from a checklist of tasks to having a formal verification process driven by an independent lab. From our research we found that commonly used system security assessment techniques are Federal Information Processing Standard FIPS 140-3 and Common Criteria (CC). The additional benefit of CC is the mapping to safety standards, as discussed below.

An underlying assumption in many of the sources we reviewed was that computing devices are assumed to be housed within a controlled physical location and

managed. Since autonomous robotic systems are not in such an environment, and may be exposed to physical attacks (invasive attacks involving physical manipulations on semiconductors like microprobing, and non-invasive attacks where side channel leakage can occur like power analysis) [2], this assumption is invalid and this type of system will need system level protection. System level protection can range from the highest level of security, tamper protection with countermeasures, to lowest level, evidence of tampering. Two specifications that do address the higher level of protection requirements are FIPS and CC.

What is FIPS? – The Federal Information Processing Standard FIPS 140-3 is a standard (levels 1 to 4, level 4 being the highest) for approving cryptographic modules, but the development lifecycle methodology for achieving levels 3 and 4 are complex (fully documented and formal design including testing) and require fully vetted modules that undergo security analysis by independent labs. FIPS Level 1 is the minimum set of security requirements that uses at least one approved algorithm or security feature. Level 2 builds on one by adding the tamper evidence requirement, such as a seal or coating that when broken provides visual evidence of tamper. Security Level 3 is intended to have a high probability of detecting and responding to attempts at physical access, use or modification of the cryptographic module. Security Level 4 provides the highest level of security, the physical security mechanisms provide a complete envelope of protection around the cryptographic module with the intent of detecting and responding to all unauthorized attempts at physical access. Security Level 4 cryptographic modules are useful for operation in physically unprotected environments.

How does FIPS work – FIPS validation requires fully vetted modules that undergo security analysis by independent labs and the implementation of algorithms are tested against a known set of algorithm values. At higher levels, the certifying module undergoes a series of exploits by the lab to ensure that there are safeguards in the system that are present and that these safeguards work against the expected attack vectors. Attack vectors can range from environmental stress testing including temperature (low and above operational levels) and physical testing, which attempts to break into the tamper protected envelope by any means. If the module can successfully defend against these different scenarios, it is deemed as passing. The cost and time frame to achieve certification depends on a number of factors and of course the level being tested against. The timeframe can take from 6 to 24 months; Government recovery costs range from $4k for level 1 to $11k for level 4, but does not include lab, consultant, and own internal team and development costs. A process flow for FIPS begins with block 1 in a five-block serial process flow. In block 1, the implementation under test undergoes activities like contract with lab, identify what to validate, perform a gap analysis (FIPS requirements and current product), fix, update and extend to support FIPS algorithms, prepare documentation for certification, and prepare for testing can take place. Block 1 can take from 3 to 18 months. In Block 2, the request is submitted for review pending and this can take up to six weeks followed by Block 3. In Block 3, a review is performed where questions and answers can be exchanged for a duration of two to three weeks. Block 4 is the coordination period where corrections and revision of documents take place; this period can take from 1 to 6 months. Block 5 is the approval, which can take a week for the certificate.

What does FIPS do well – FIPS 140-3 standard provides a methodology for security evaluation depending on the level of features/functions that a module supports. This is a well-documented process with a number of certified labs that support the evaluation. The algorithms also support a set of known good results that are used during the implementation and testing phases. This methodology can be extended in the autonomous robotic system space for covering security modules and tamper protection.

Shortcomings of FIPS – This relies completely on the human-based vetting process which is time consuming and expensive. Another shortcoming is that the criteria are static, once assessed, always assessed. There is no way to account for "unexpected" attacks or for having the system reason about what may or may not be an attack. A final point is that the standard defines a set of requirements to achieve a level of evaluation that only targets the cryptographic modules and the small Target of Evaluation (TOE), but this may miss higher levels of side channels.

Like FIPS, the Common Criteria (CC) provides a framework for information technology security evaluation and defines a common evaluation methodology to achieve an international recognized certification.

What is it – CC defines the Evaluation Assurance Levels (EAL) are from 1 to 7, with EAL 7 being the highest level. EAL 1 (functionally tested), EAL 2 (structurally tested), EAL 3 (methodically tested and reviewed), EAL 4 (methodically designed, tested, and reviewed), EAL 5 (semiformal designed and tested), EAL 6 (Semiformal verified design and tested. Formal methods and systematic covert channel analysis required.), EAL 7 (Formally verified design and tested. This level requires more formal methods and systematic covert channel analysis required) [104]. Level 4 is the highest

level that is mutually recognized by the Common Criteria Recognition Arrangement (CCRA).

How does it work – CC defines a process flow for deliverables and certification. A first step in the process flow is to define the security product and market segment for the product. This will be matched to a Protection Profile (PP) that is used for the type of product or requirements for the needed security solution, which is like a request for proposal. The next step is to define a Target of Evaluation (TOE). This defines the security features that the product supports. A protection profile is created to document what the TOE is for the certification. Defining the Security Target (ST) is the next step, which defines the security properties of the TOE like functionality and assurance components; the ST can target multiple PPs. The ST and implementation documentation is provided to an accredited third-party lab. The lab evaluates if the ST meets the PP through testing and makes the decision for the evaluation. The lab performs two evaluation tests called Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs). ST establishes the SFR for the product evaluation depending on the individual security features and the SARs for assurance claims. The SAR describes the measures taken during development and evaluation to ensure that the security functionality claims comply. The lab tests determine if the claims being made are valid. The final step is based on the test findings where the lab assigns an evaluation assurance level. In the case of higher assurance levels, formal evidence must be submitted with the documentation. Depending on the EAL, the time and cost can range considerably. Dale, in 2006 presented a briefing that provided the following correlation between the levels: EAL 1 = 0, EAL 2 = $100 to $170k and 4 to

6 months; EAL 3 = $130 to $225K and 6 to 9 months; EAL 4 = $175 to $750k and 7 to 24 months; EAL 5 = $750 to $2M and 24 to 48 months [104].

What does it do well – CC has been recognized internationally as a (ISO - 15408) standard for information technology. This standard can be applied to a number of different IT devices, which helps with autonomous robotic systems. A fully documented development lifecycle is required, and at higher assurance levels will undergo an analysis (protection mechanisms) from an independent 3rd party certified lab. Other standards for safety have been mapped to these assurance levels, as is discussed below.

Shortcomings- Currently CC is being utilized toward IT equipment and will need to include complex systems like autonomous robotics systems where not only the OS will be considered, but sensors, actuators and AI must be part of the certification process. Similar to FIPS, CC may only define a small TOE part of the system and potentially expose other parts of the system. Also, like FIPS, this captures a static assessment of the TOE and does not account for uncertainty.

Both FIPS and CC have evolved from a need to assess cryptographic modules and are early examples of approaches for scoring certain systems and system components. Another linkage is the Orange Book assessment guide for higher levels systems.

FIPS is geared toward cryptographic modules but addresses the tamper protection mechanisms. CC, on the other hand, is targeted toward system level assurance where tamper protection is included in the higher assurance levels. There is a continuous effort in mapping safety and security specifications, since each have similarities in the development process, but more importantly security is becoming a

requirement. A general specification on, IEC 61508, spans into the medical, machinery, automotive, rail, process industry and nuclear domains. The European Union is taking steps toward creating workgroups to address the mapping of safety and security [105]. Like FIPS and CC, such specifications aid in quantitatively assessing systems and components, as they provide a basis for scoring systems and components.

Since autonomous robotic systems can be incorporated into a number of industries, the current safety standards are used to map back to a CC standard. Schmittner and Ma presented a paper that discusses the mapping of Automotive Safety Integrity Level (ASIL) to CC EAL levels [106] as shown in Table 4-1.

TABLE 4-1: COMPARISON OF INTEGRITY AND ASSURANCE LEVELS

ASIL		EAL
ASIL A	~	EAL3
ASIL B	~	EAL4
ASIL C	~	EAL5
ASIL D	~	EAL 6

Table 4-2 shows the layering of the different safety specifications (automotive, general grouping, aviation and rail) and the mapping to Common Criteria for security [106] [107] [108]. At the top row is the automotive standard ASIL that Schmittner and Ma mapped to CC as shown in Table 4-1, where Quality Management (QM) represents that risk is not unreasonable and no safety measure is needed. The mapping also had ASIL-D being the highest degree of hazard injury and highest degree of rigor applied in the assurance. Mapping ASIL to General and Rail safety standard, Safety Integrity Level (SIL) where SIL 1 is the lowest and SIL 4 being highest. These safety standards are then mapped to aviation standard D0-178, where Design Assurance Level DAL implies A (Catastrophic), B (Hazardous and Sever -Major), C (Major), D (Minor), and E (No Effect). These safety standards are aligned with FIPS 140-3 and CC EALs. CC

supports in between levels by adding the plus symbol, which means that partial categories were fulfilled for a specific category. All these techniques are, at root, doing the same thing. The problem is that they do not really capture the details of how a system actually operates (complexity/granularity rapidly becomes intractable using this methodology). The problem is that the assessment is a bit divorced from reality for an autonomous system. If it is available, it is beneficial, but it will not be available for everything. Using it as evidence supporting trust makes sense, but it is only part of the solution.

TABLE 4-2: MAPPING SAFETY AND SECURITY SPECIFICATIONS

Domain	Domain Specific Assurance Levels						
Automotive (ISO 26262)	QM	ASIL-A	ASIL-B	ASIL-C	ASIL-D	ASIL-+	
General (IEC-61508)	-	SIL-1	SIL-2	SIL-3	SIL-4		
Aviation (DO-178/254)	DAL-E	DAL-D	DAL-C	DAL-B	DAL-A		
Railway (CENELEC 50126/128/129)	-	SIL 1	SIL 2	SIL 3	SIL 4		
FIPS 140-3	L1	L2	L3	→	L4	→	→
CC (ISO 15408)	EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7

The main value in these approaches is their identification of areas that need to be assessed in a comprehensive model for a robotic system, and their proposed methodology for assigning qualitative "weights" based on component (system) characteristics. Using CC EAL levels 1 to 5 can be looked at for a base set of system level metrics to cover the security and potential safety assurance. We also need to consider resiliency and how the system will react when threats are encountered. Cho et al. discussed resiliency in their TRAM research as these concepts should be considered.

4.4 Hardware Level Trust Evaluation

A robot has many hardware components from processing data to sensing the environment. The processing of data is performed by a microprocessor and other accelerator devices. Sensors and actuators perform the locomotion, manipulation, and navigation functions in a robot. This integration between the hardware components can be defined as direct or decoupled communications. This means that in the case of direct communications, the data is exchanged between client and server with no blockers, so that a response is provided as a direct means of the request, which is similar to a command-response manner. In the case of decoupling communications, there is a broker as an intermediate step to exchange data for which the widely used protocol is a publisher/subscriber format. The broker is the man in the middle that needs to know all the entities in the publisher/subscriber network in order to route the messages between source and destination. This might impact trust differently as compared to the direct communications approach. Sensors provide the basic capabilities for autonomous systems to collect data from the environment in order to make decisions. Each hardware component is constructed from a single Integrated Circuit (IC) or a set of ICs that provide the functionality for data processing, sensing or actuator controller logic. ICs are made up of electronic circuits like, transistors, capacitors, resistors, etc., that make up the digital or analog functional logic. ICs that are designed for microprocessor, graphics engines, or digital processing logic can have many millions of transistors. In order to design these complex ICs, software used to design, develop, and test are called Electronic Design Automation (EDA) tools. EDA tools provide a number of functionalities like logic synthesis (converts Verilog or Very High-Speed Integrated Circuit Hardware Description Language (VHDL) to netlists), place and route (uses

netlist to find optimal locations for wires and transistors), simulation (takes the netlist as a description of the circuit logic and mimics its behavior), and verification (design rules for the netlists). So why is it important to have hardware trust?

In order to establish a secure foundational base for autonomous robotics systems a concept called the hardware root of trust is used to form a trusted base for all secure operations. Hardware root of trust is the concept to boot from an immutable point in the software stack used on computer systems today. This concept assumes that the hardware is free from potential vulnerabilities, but from our point of view hardware trust should also look back at the design origin of the ICs. This level of trust needs to be considered for the holistic security model, since underlying vulnerabilities like trojans and malware can be introduced in the IC design phase.

A robot's hardware is likely a combination of COTS and custom electronics from a variety of sources. The components on these electronic assemblies, in turn, come from a variety of sources. We generally assume that the things we buy do what they are supposed to do – i.e., an 802.11 chip does 802.11 wireless communications. If it does, we are happy. However, when evaluating the amount of trust that can be placed in a system, we also need to acknowledge that we have no idea what else it might do, and if any other possible functions may result in system vulnerabilities. Security at the chip level is taken into consideration and finding a set of trust metrics is one component to a holistic security architecture.

Each of these hardware components is constructed of Integrated Circuits (ICs) that may have single purpose or combined purpose where multiple functions are placed on a single die. An example of this structure is shown in Figure 4-4 which illustrates an example of a common robot sensor, called an inertial measurement unit (IMU). This sensor component includes an ARM-based CPU as well as a motion processing unit, or MPU chip. From the block diagram of the MPU, shown on the left of Figure 4-4, we see that the MPU is itself a complex system in addition to the ARM CPU. The arrow points to an exploded view of the internal design (consisting of two dies integrated into a single package). One die houses a 3-Axis gyroscope and a 3-Axis accelerometer. Another die houses a 3-Axis magnetometer, "the MPU-9250 is a 9-axis Motion Tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer and a Digital Motion Processor™ (DMP) all in a small 3x3x1mm

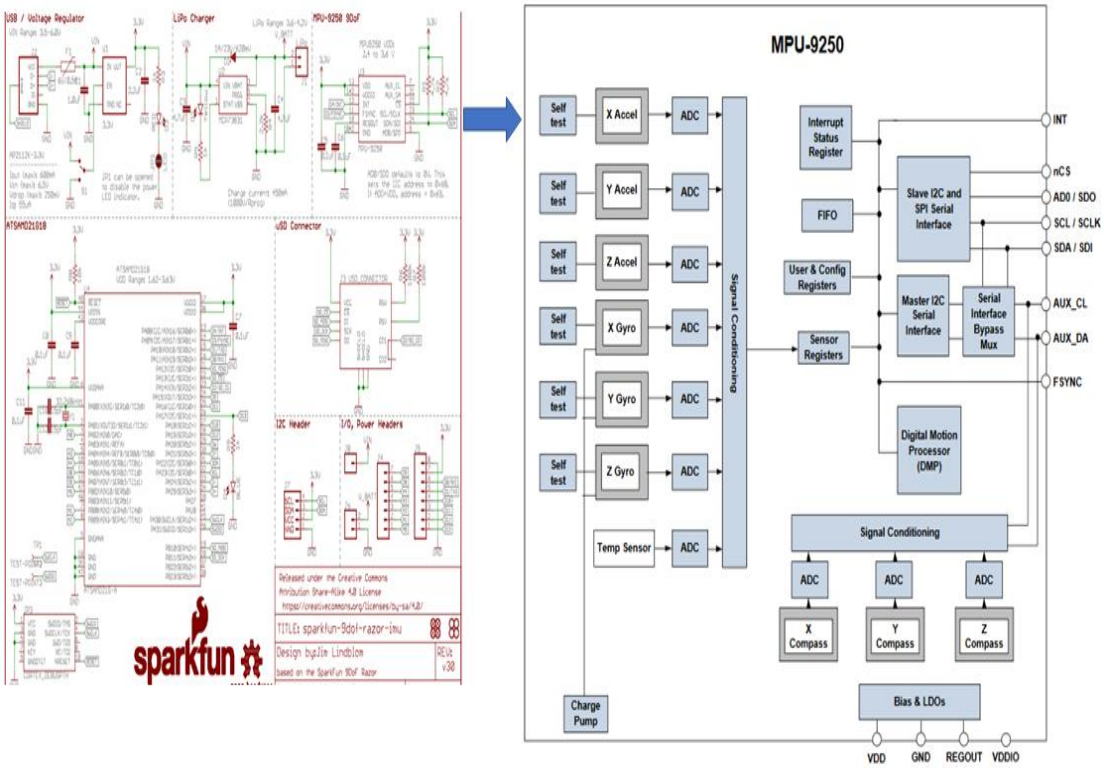


Figure 4-4: IMU board with a motion processing unit

package" [109] [110]. Thus, even in a simple sensor there may be a large amount of complexity in which there are several places where trust can be compromised even in a $35 COTS part.

The process to create an IC is very complex, starting with Electrotonic Design Automation (EDA) tools and ending in a foundry. Figure 4-5 [111] shows the top line as the design flow and the lower line being potential points in the process to alter or inject trojans in the design. Counterfeiting is the illegal forgery of an original part. While it poses an economic threat to the legitimate owner of the Intellectual Property (IP), and possibly a reliability threat for the component, a true forgery should not be any different electrically than the original part. However, since the authorized manufacturer of the part being counterfeited has no control over the counterfeit, this provides a counterfeiter an opportunity to insert logic that can compromise trust. These are only a couple of possible threats that can occur at different points in the IC lifecycle.

Acknowledging that there are potential vulnerabilities in the design and manufacturing process, there have been a number of initiatives related to trusted IC development including the Trusted Foundry—the proposed approach is to split the

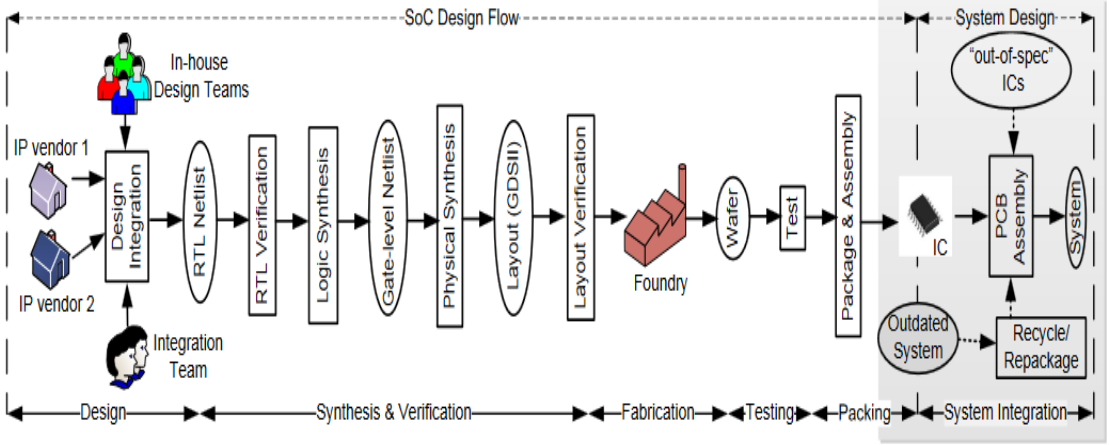


Figure 4-5: The top line is the IC design flow and the bottom line represents the injection points.

foundry functions into two [112]. One foundry would create the fine layer of transistors with detailed leads to connect them and the second vetted foundry would do the less fine wiring with connections to the outside world. This split benefits trust in two ways, 1) not having a single point of knowledge and 2) it creates a second validation point. We view the foundry as a supply chain trust metric and focus on the IC design in this section.

To identify methodologies used for evaluating trust at the hardware level, we searched for papers focused on BNs, trustworthiness, sensor trust, and IC trust. From these parameters the results were categorized into two groupings, the first being the IoT and Sensor where Mozzaquatro el al.[113] and Thamburu el al.[114] discuss IoT cybersecurity framework and management system for them. In addition to the first grouping is the work from Lu and Cheng [115] and Boudriga [116] that focused on identifying bad sensor behavior in a wireless network. The second grouping was related to hardware trust metrics, where Kimura presents a technique for determining a design integrity trust metric for hardware design by evaluating five different characteristics (signal, logical, power, function and structural integrity) [101]. We focus on Kimura's work, since this supports our discussion above with vulnerabilities in the design phase. This is important for autonomous system because hidden threats can be exposed at the design phase and in the supply chain section below, we cover the foundry portion of the IC process as a set of split metrics.

What is it- Kimura presents quantifying metrics for hardware designs called "Development of Trust Metrics for Quantifying Design Integrity and Error Implementation Cost". An evaluation of design integrity is accomplished by looking at five different characteristic domains (Logical Equivalence, Signal Activity Rate,

Structural Architecture, Functional Correctness, and Power Consumption) of the design and then aggregating their measured deviations from expected characteristics together to arrive at a single value Design Integrity (DI) metric. This technique can be leveraged in two ways, one being a metric for design integrity used by the IC provider to ensure that no unauthorized functions were introduced when the device progressed through foundry and also allows validating the output from the foundry. A consumer of the IC can use the design integrity metrics to determine the differences between actual and expected designs by obtaining the needed information. In each case the Design Integrity approach provides a metric to measure the quality of the design and a methodology to reduce potential exposure to IC threats. An Error Implementation Cost (EIC) measure is developed as a technique to quantify errors and to allow error ranking and rating. A final Trust Measure metric is calculated that includes an estimate of design's integrity and reference design quality or characteristics. In the case of the IC provider, they would all have content as an output of the product lifecycle, and in the case of the IC consumer, they would have limited information, so the reference quality is a variable to address the subcategory content.

How does it work- Figure 4-6 [117] shows the breakdown into the sub layers of the design. The Design Integrity (DI) trust metric accounts for the signal activity rate, logical equivalence, power consumption, functional correctness, and structural analysis [118]. The integrity of a design can be defined as the amount of deviation observed between reference and sample designs. For the case of black box IP, a reference specification is usually provided, such that a behavioral model can be constructed, or layout reverse engineering conversion tools can be used to derive netlists.

- Signal activity rate is the number of times the evaluated element changes state over the duration of a given test scheme. Signal rate evaluates data, I/O and logic signals using equation 4-1.

$$\mathrm{SR}_{\mathrm{integrity}} = \frac{\mathrm{SR}_{\mathrm{expected}} - \Delta\mathrm{SR}_{\mathrm{dist}}}{\mathrm{SR}_{\mathrm{expected}}} \qquad (4\text{-}1)$$

where $0 \leq SR_{integrity} \leq 1$ and the differences between actual and expected is the ΔSR_{dist} value. This can be used by either the provider or consumer of the IC.

- Logical equivalence is the degree to which the logic state points of the design can be compared to the original reference. This is best suited for the IC consumer case where they may have access to a netlist to ensure that they have a clean chip. A tool like Cadence Conformal can be used to check the difference between netlists and depending on the differences result, this is the deviation score. $LE_{integrity}$ can now be expressed as the ratio of Equivalent Points to the Total Comparison Points as shown in equation 4-2.

$$\mathrm{LE}_{\mathrm{integrity}} = \frac{\mathrm{Points}_{\mathrm{EQ}}}{\mathrm{Points}_{\mathrm{COMPARED}}} \qquad (4\text{-}2)$$

where $0 \leq LE_{integrity} \leq 1$

- Power consumption measures how closely the actual design aligns to the original reference from a power perspective. Power consumption is comparing the simulation tests to actual tests for each power test point using equation 4-3.

$$\mathrm{Pintegrity} = \frac{\mathrm{Pexpected} - \Delta\mathrm{Pdist}}{\mathrm{Pexpected}} \qquad (4\text{-}3)$$

where that $0 \leq Pintegrity \leq 1$ and the differences between actual and expected is the $\Delta Pdist$ value. This can be used by either the provider or consumer of the IC.

- Functional correctness is the difference between the actual and expected results in order to verify correct design. Tests can range from exhaustive testing to ones that only provide corner and basic coverage. *Fintegrity,* is evaluated by observing the number of errors that occur, *εobserved*, for a given verification test scheme and *TPtotal* is the total verification test points used for verifying the design functionality, this is shown in equation 4-4.

$$\mathrm{Fintegrity} = \frac{\mathrm{TPtotal} - \varepsilon\mathrm{observed}}{\mathrm{TPtotal}} \qquad (4\text{-}4)$$

where $0 \leq Fintegrity \leq 1$. This can be used by either the provider or consumer of the IC.

- Structural analysis looks at the architectural components regarding gate level net lists for comparison and/or leaf cells using equation 4-5,

$$S_{\text{integrity}} = 1 - \Delta S \qquad (4\text{-}5)$$

where ΔS is the differences between actual and expected number of modifications found. This can be used by either the provider or consumer of the IC.

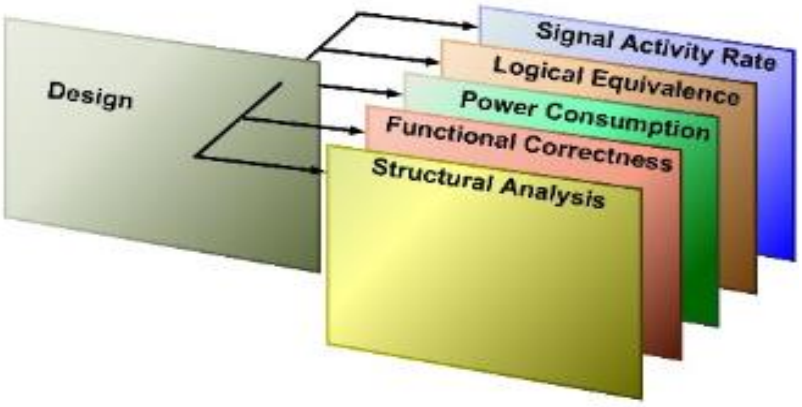


Figure 4-6: Breaking down the design into subcategories

The integrity of a design can be defined as the amount of deviation observed in a one-to-one mapping for each of the subcategories defined above where the actual vs expected is compared to the reference specifications, simulation, and testing results. In Kimura's [118] the subcategories are analyzed individually and normalized, then accumulated together, for a single set of metrics. The DI trust metric considers the design integrity plus the reference design quality. The reference design quality is the set of design artifacts that are obtained. These can range from fully synthesized behavioral models, being the highest reference confidence, to the data sheet, being the lowest. Figure 4-7 shows the trust metric ranges within 0 to .19 being the lowest range and with 1 the highest. Equation 4-6 is used to calculate the Trust Measure.

$$\text{TM} = \text{DI} * \text{R} \qquad (4\text{-}6)$$

where DI is the Design Integrity and R is the Reference Design Quality.

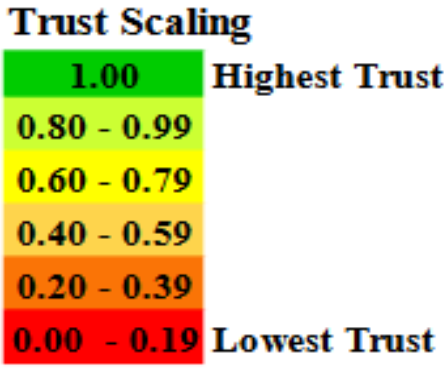


Figure 4-7: Trust scaling for hardware components

What does it do well – From the design validation point of view these subcategories represent the areas that are prone to the injection type of attacks and forensic analysis for counterfeits. Therefore, having a trust metric based on hardware design properties provides a degree of assurance that an IC has been validated between the expected design/simulation and the as built or actual. As shown in Table 4-3 [118] where the test cases of the five subcategories are quantified, the Design Integrity metric is calculated and the Figure of Deviation (FOD) is calculated. The top and bottom boxes are inductions that no malice hardware injected faults were detected, the orange one being all signal bits being inverted on the output and the two yellow ones being a counter added to trigger a XOR operation. The other yellow one being an add function with a bit-wise inverter. These were determined by having a test fixture where the expected results were compared to the actual ones.

TABLE 4-3: TEST CASES FOR THE DESIGN INTEGRITY ANALYSIS

Test Article	$SR_{integrity}$	$P_{integrity}$	$LE_{integrity}$	$S_{integrity}$	$F_{integrity}$	Design Integrity	FOD	Description of Error Insertion
TA1	1.00	1.00	1.00	1.00	1.00	5.00	0.00%	No malicious content added.
TA2	0.67	0.98	0.13	0.56	0.99	3.33	33.40%	Counter added to trigger XOR operation on output
TA3	1.00	0.99	0.00	0.97	0.87	3.83	23.40%	AND function with bit-wise invert
TA4	1.00	0.99	0.00	1.00	0.00	2.99	40.20%	All signal bits inverted at output
TA5	1.00	1.00	1.00	1.00	1.00	5.00	0.00%	Stray signal wire added

This furthers the need to have the foundry be represented in the supply chain metrics that cover the other types of attacks that were mentioned above. The focus of IC design and accelerators, like FPGAs, are most prominent in autonomous robotic systems. The use of trust metrics for design integrity is vital for the holistic security model.

Shortcomings - While the set of values shown in Figure 4-7 represents a Trust Metric for hardware design quality, it only covers the logic design part of the process – it does not address the foundry or the design flaw costs that were also discussed in Kimura[118]. As stated, before the IC provider has all the knowledge and artifacts from the development process and this methodology can be applied in house, the consumer of the IC will need to be more resourceful in obtaining knowledge about the design.

In general, relatively few entities are designing the actual components, and more are using COTS devices. Thus, the importance of obtaining components from sources with integrity increases. In the example of the IMU, a complete board is provided by Sparkfun that may contain components of unknown provenance or a potential risk for compromise in the signal paths. Another potential danger in using a quick to get, cheap and off the shelf component, is that it may work in the prototype phase as desired but

using the prototype component in a production phase will increase the risk of a flaw, like a Sparkfun board. This causes reliability issues and may expose inherent security flaws. As part of the hardware component metrics, the need to incorporate the loss and reward values must also be factored in an overall trust model. These two values are described in more detail in the software section. The trust scaling does provide a good set of metrics for the hardware components because it focuses on the design elements.

4.5 Software Level Trust Evaluation

To identify methodologies used for evaluating trust at the software level, we searched for papers focused on BNs, trustworthiness, and security. From these parameters the results are categorized into two groupings, standards and the usage of the Common Vulnerability Scoring System (CVSS). We highlight some of the software search results that are attractive to incorporate into our holistic security model.

Software in a robotic system extends from firmware to kernel/Operating System (OS) into middleware and application layers. Software can also be embedded into controllers or using device driver logic within the OS to enable hardware components and can be found embedded in sensors (such as the MPU-9250 mentioned above) or actuators. Software need not only be logically correct but may also need to support temporal constraints. This means that the OS supports a preemptive scheduling on time-bounded tasks, so that the task is completed within a time constraint. The time bound processing is called real-time, and this term can be further broken down into hard or soft, where in the case of hard real-time, there is no tolerance for a delayed response and in the case of soft, there are tolerances. Software has far reaching effects at different levels of the stack; therefore, security for it must be taken into consideration.

Software metrics provide a quantitative means to assess or measure the efficiencies in the software development lifecycle, where requirements, design, implementation, testing, and documentation are the different phases. We focus on the implementation since code is an output of this phase. The remaining phases will be considered as part of the supplier chain. The reason why software security metrics are important at the implementation phase is because the code exposes the underlying vulnerabilities. Metrics are used to assess the software assurance against vulnerabilities/weakness from an attack and finding a set of trust metrics is another component to a holistic security architecture.

As an example, consider that the software running on any given robot today is likely a combination of open source and COTS code modules obtained from a variety of public and private sources. For example, OpenSSL is a well-established cryptographic library that provides a number of data protecting functions that are embedded within many operating systems. If it works as expected in a system, we tend to think it is ok, providing the level of security the industry associates with OpenSSL. However, an important assumption underlying the security of communications using OpenSSL is an assurance of the physical security of trusted nodes. For example, in Chapter 3 I demonstrated a spy process that can capture encrypted data, as well as the necessary decryption keys, and send them to a remote service without the user knowing their system has been compromised. This demonstrates that even a well-known library, like OpenSSL, can be compromised if the physical hardware is compromised. Since in an autonomous robotic system nodes may be captured, an assumption of physical security may be invalid [3].

In our research results we came across a number of standards that define metrics for vulnerabilities, whether in misuse, misconfiguration, or in a weakness of implementation. These are described as follows Common Vulnerability Scoring System (CVSS), Common Configuration Scoring System (CCSS), Common Misuse Scoring System (CMSS), and Common Weakness Scoring System (CWSS). Since these are derived from the CVSS standard we will discuss these as a group and will compare them below.

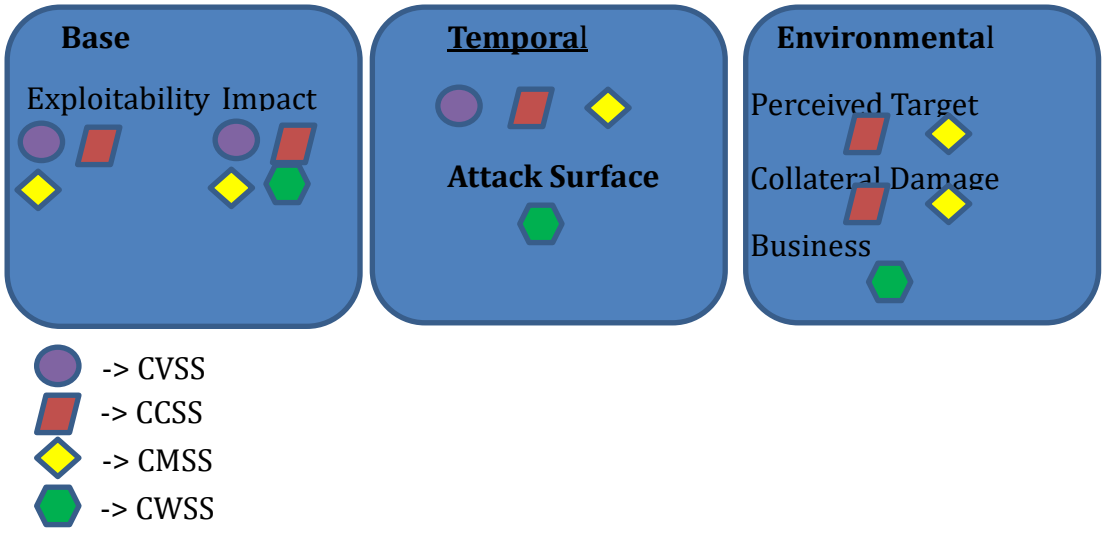


Figure 4-8: Comparison of CVSS, CCSS, CMSS and CWSS scoring specifications

What is it - We first start with CVSS, which is the main standard related to the National Vulnerability Database (NVD) that is a repository which provides Common Vulnerabilities and Exposure (CVE) entries. The Common Vulnerability Scoring System is a standard for assessing known vulnerabilities [119]. The next standard CCSS [120] is derived from CVSS but deals with the misconfiguration class of vulnerabilities. Vulnerabilities from misconfiguration can be described as a software feature that enables changing settings. For example, changing access control settings on a directory

where Read/Write permission is given to other/everyone via file manager can leave sensitive data exposed, if not used properly. This type of security exposure is deemed a configuration vulnerability. Like CCSS, CMSS [121] is derived from CVSS but deals with the misuse class of vulnerabilities. A misuse vulnerability is built into the software as a feature where the intended feature can expose a security vulnerability. For example, a web link can lead to a malicious site in a messaging application by clicking on the link. The software design feature was to allow a user to follow a link, but not protect against potential exploits. Our final standard, called CWSS is also derived from CVSS but deals with the weakness class of vulnerabilities. CWSS is a part of the Common Weakness Enumeration (CWE) project, co-sponsored by the Software Assurance program in the office of Cybersecurity and Communications of the U.S. Department of Homeland Security (DHS) [122]. An example of a weakness is given by CWE-290 (authentication bypass spoofing), which is caused by an improperly implemented authentication scheme. A spoofing weakness can be caused by only checking the IP address of a client that can be spoofed by the attacker and not using a DNS lookup as the source. A comparison between each of the outlined specifications is that CVSS is based on a vulnerability that is known and CWSS can be used as an assessment metric for software assurance.

How does it work – Basically each standard has three groups called Base, Temporal, or Attack Surface, which is unique to CWSS and Environmental. The Base Group has two metric subgroups: *exploitability* defines the complexity to achieve the exploit and the *impact* defines the result of the exploit. The Temporal group defines the time dependent attributes for the exploit in CVSS, CCSS and CMSS. In CWSS instead of having the Base/Exploitability and Temporal categories, some elements are placed

into an Attack Surface group. The Environment group defines the surrounding specific attributes for the exploit. In the case of CCSS and CMSS, both Perceived Target Value and Collateral Damage Potential are defined whereas in CWSS a business impact is specified in the Environmental Group. Figure 4-8 shows the comparison of the four different types of scoring specifications overlaid on to each other.

Each standard has a set of equations for calculating a score and an example of a known vulnerability using the CVSS scoring. As discussed earlier the National Vulnerability Database (NVD) is a repository which provides CVSS base metrics on Common Vulnerabilities and Exposure (CVE) entries. Once a vulnerability is discovered, it is assigned a unique CVE Identifier. In this example, the CVE-2019-15786 Detail ROBOTIS Dynamixel SDK through 3.7.11 has a buffer overflow via a large rxpacket is given with the following parameters.

Using the following vector /AV: N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H, a base score is calculated with a 9.8 result being critical. This vector is broken down into the following structure where:

- Attack Vector = Network
- Attack Complexity = Low, Privileges Required = None
- User Interaction = None
- Scope = Unchanged
- Confidentially = High
- Integrity = High
- Availability = H

A calculator can be used to obtain the same value score with these parameters. The specification, examples and calculator are located at the FIRST website [119], the calculator breaks down the three groups into a set of values where metrics are assigned

and an overall score is the result. The scores rank from 1 to 10, with 10 being a high severity. For CVSS, the base score is only used in the CVE and NDS entries.

What does it do well – CVSS provides a set of metrics for known vulnerabilities and is used in many types of analysis tools we found in our search. The base impact set of metrics are common values that were used in most of our research results. As for CCSS and CMSS, they provide a set of metrics for Collateral Damage Potential and the Perceived Target Value. These two metrics represent the damage caused by an attack and the reward. These values provide insight into the assessment of an attacker's goals and should be included as part of the trust metric for the different layers of an autonomous robotic system. Finally, CWSS provides a set of metrics for potential weakness in software and hardware that can be used for software assurance. In fact, a number of software tools are using CWE metrics for static and dynamic analysis and the NVD is also using this metric in conjunction with CVEs. CWSS provides a set of metrics that can be used in the software layers for the holistic security metrics.

Shortcomings – While CVSS and CWSS are targeted toward conventional known software vulnerabilities, autonomous robotic systems are a new topic for this standard. However, limited vulnerabilities have been found using this technique. As for CCSS and CMSS, while these contributing type vulnerabilities can be found in the CWSS and CVSS as the two complementary specifications, CCSS and MCSS maybe redundant.

From the number of categories and sub-categories from each of the scoring specifications, it seems that impact, collateral damage, and perceived target value are the three most important attributes for calculating a trust metric. The other categories

and sub-categories are attributes related to supporting/describing the exploit as to when and where.

From the CWSS specification, technical impact scoring provides a better set of definitions with their corresponding coefficients that are used in the equations to derive the category numerical value for assessment (Critical, High , Medium, low and None (1, .9, .6, .3 and 0) [122].

From CCSS and CMSS the collateral damage provides coefficient metrics that are: none: 1, low: 1.25, low-medium: 1.5, medium-high:1.75 and high: 2 [120] [121].

From CCSS and CMSS the perceived target value metrics are: low: 0.8, medium: 1.0, and high: 1.2 [120] [121].

Several articles were found that used attack graphs and utilized CVSS metrics for determining security flaw vulnerabilities. These articles picked a small portion of the system or looked at a network configuration that was software centric. Current approaches of combining CVSS scores have at least two limitations: they lack support for dependency relationships between vulnerabilities, meaning they may be ignored or modeled in an illogical way. The other limitation is that they focus on successful attack probabilities only (Cheng et al.)[123].

Frigault et al.[124] and Xie et al.[125], present two techniques where the CVSS scores are used as metric values in a BN. The CVSS scores and BN usage model are similar to the technique from Shetty as discussed earlier. Xie et al., extends the usage model by incorporating noisy gates to assist in uncertainty in real-time security analysis. Each node in a BN requires a distribution which is conditioned on its parents. To model the child node's effects from each independent parent node, a combined influence can

be achieved by using logic gates (e.g., AND, OR and etc.). The term noisy reflects the fact that the logic gate combination is probabilistic and not deterministic.

Frigault et al., presents a technique to measure the overall network security by combining CVSS, attack graph, and BN. This technique is further extended into a Dynamic BN where temporal events are represented in the model**.**

What is it – An overall metric is developed for network security using a set of CVSS base scores that are converted into probabilities which are assigned to the paths in an attack graph for an overall assessment value. Both the attack graph properties and probabilities are used in a BN and are extended into a DBN for time-based events.

How does it work – An annotated attack graph is first generated and then a BN is derived from the data in the attack graph. The nodes of the BN represent exploits and conditions derived from the attack graph. Each node represents a probability derived from the CVSS score. Conditional Probability Tables (CPT) represent the causal relationships between exploits and conditions. A BN inference model can reason about if an attacker can reach their goal by any condition. A DBN was also presented that utilized the temporal scores in CVSS for the vulnerability, which showed the time-based activities in exploit maturity, remediation level, and report confidence [124].

What does it do well – This approach combines the CVSS (for known attack vectors and paths taken) metrics into probabilities, so that CPT are constructed and utilized in the BN, then extended into a DBN. The use of causal inference is a good method to understand relationships and potential uncertainty.

Shortcomings – The scoring mechanism is based on a known attack graph and CVSS score metric but does not consider the attacker's experience/knowledge about the environment. Using CVSS as the only metric is one method of achieving the

construction of the BN. This is limited to one data point that states an existing exploit is already found in the NVD and new ones are unaccounted. Since robotic systems are a new domain, the entries in the NVD are limited.

Xie et al., presents a technique that extends the CVSS scores and BN usage model by incorporating noisy gates to assist in uncertainty in real-time security analysis. Real-time security analysis in this context is related to observations from an Intrusion Detection System (IDS) sensor providing false or negative readings or a file system integrity checker such as Tripwire that alerts to a file being changed. The term uncertainty is related to an attack being successful, the uncertainty of an attacker's path choice, and/or the uncertainty from imperfect IDS sensors.

What is it - The CVSS scoring metrics allow security analysis tools to define potential exploits in the pre-deployment phase, but how does one account for uncertainties? A BN model is presented that separates three uncertainties in real-time security analysis: the uncertainty on attack success, the uncertainty of attacker choice, and the uncertainty from imperfect IDS sensors.

How does it work – Using CVSS metric scores (Base and exploit in the temporal category), the CPT are structured from these values. An attack from an attacker is defined as the physical path (attacks can only occur by following network connectivity and reachability; this is the physical limit for attack) and the attack structure (attacks can only happen by exploiting some vulnerability, with pre-conditions enabling the attacks and post-conditions as the consequence (effect)). A tripwire node is created to sense the detection of an attack, for example from an IDS sensor. The use of the Noisy AND/OR logic are utilized for the conditional events that the attacker must take in order to achieve the vulnerability or goal [125].

What does it do well – The use of tripwire and Noisy AND/OR logic is used for the conditional events. The combination of these two logics enables switching for real-time events/analysis. These two logics can be extended into the autonomous robotic layers of the BN for triggering on time events.

Shortcomings - Real-time security analysis is a far more imprecise process than deterministic reasoning. We do not know the attacker's choices, thus there is the uncertainty from unknown attacker behaviors. Cyber-attacks are not always guaranteed to succeed, thus there is the uncertainty from the imperfect nature of exploits. The defender's observations on potential attack activities are limited, and as a result, we have the uncertainty from false positives and false negatives of IDS sensors [125].

Our search results showed that these techniques provide a basis for the assignment of scores, and in aggregate, they provide coverage of many things important to score. They miss a few things that can be easily added, including the need to incorporate loss and reward values into their security assessments. The discussion of asset value and loss were represented in Cheng et al.[126] with regard to security measurements for situation aware cyberspace. We view that loss and reward should be part of the risk calculations to provide a better qualitative assessment. By capturing the coefficients for impact, cost of loss and reward value, these will be utilized in a trust model that represents the software components related issues. For those things that need to be added, there is not enough experience with robotic systems to form a basis for assignment of scores, but the methodology is helpful.

4.6 Cognitive/AI Level Trust Evaluation

An autonomous robotic system may have one or more AI algorithms running on its system for it to operate within an environment that has many uncertainties. Often, the robot must learn and adapt to the stimuli from the environment. For example, having an autonomous robot in the household as a caregiver aide or assistant. An adversarial attack can alter the robot's behavior by mischaracterizing objects like medicine or producing wrong monitoring results for the patient. This is only a small window of the possibilities of scenarios where an AI is being deployed and having assurance about the implementation is needed, especially in critical and safety applications.

To identify methodologies used for evaluating trust at the Cognitive or AI level, we searched for papers focused on cognitive, trust metrics, BNs, and AI robustness. From these parameters the results were categorized into three groupings: standards [127] [128], techniques for detecting adversarial attacks [129]-[130] and accuracy vs adversarial training. We highlight some of the Cognitive/AI search results that are attractive to incorporate into our holistic security model. AI certainly is not new; it has been around since the mid-1950s when John McCarthy used the term Artificial Intelligence and invented the LISP programming language. The path of AI has taken different twists along the way with expert rule base systems as one form, to neural networks that mimic the brain, and currently the combination of these two called symbolic AI. In order to handle complex data for AI models to be useful the community is leveraging Convolution Neural Networks (CNNs). CNNs use a form of deep learning that is targeted toward image and natural language domains. Reinforcement learning is another form of Deep Neural Networks (DNN)s targeted toward path planning, while perception and motion are often associated with autonomous mobile platforms. The

new focus of research is AI adversarial attacks, the classification of data being poisoned, evasion attacks, and black box attacks to name a few. For every attack, a remedy may arise to counter, but this takes time and there needs to be a method to identify these types of attacks. In the case of autonomous mobile robots, an AI/learning layer opens up new types of attack strategies that more conventional attacks do not.

AI logic often comes from open or COTS sources and determining the robustness to adversarial attacks needs to be considered for the system trust model. If these AI implementations have not been subjected to some adversarial data training, the lack of testing with different types of data sets will lead to an untrustworthy implementation. The need for a robustness metric is required, so that certain implementations can be used in high assurance scenarios. Adversarial attacks have been discussed in the literature related to poison (tainting the training data), white/black/grey box (using physical and alternative training models), and evasion (misclassification by spoofing). A large portion of research has been geared toward image-based data sets; however, testing should be expanded into other domains. Since AI robustness and adversarial attacks are new, our research has come up with a mixed bag of results that are grouped into possible solutions: the first group is adding adversarial data to the training cycle, the second group is using a value or utility function to assist in a distance measurement functions, and the third group is using linear bounds as constraints. We first describe the types of attacks to provide an overview for why AI robustness metrics are needed, since this is an evolving field.

An example of a poison type of attack is in spam filters, where emails are filtered for characteristics of being spam. What looks like a normal email is used to train the spam filter's algorithm to include the extra words to be recognized in its knowledge

base. This process then causes the spam filter to flip what would normally be seen as a good email into a bad one.

White, black, and grey box attacks are related to an adversary having different levels of knowledge about the system. In the case of white box, the adversary would have full disclosure about the algorithm, including the weights and all data used during training. An example of a white box attack is in object recognition systems like handwritten symbols in an image where a 1 can be recognized to be a 7 with small modification of the image. A black box case is where the adversary has no knowledge of the algorithm or data. An example of a black box attack is subjecting the model being attacked to different inputs and acquiring the outputs, the adversarial knowledge that is obtained can be transferred into a substitute model for further exploitation. In the case of grey box attacks, the perturbation of a pixel can cause misclassification of an image like in the stop sign example discussed below.

A simple case of evasion attacks is to manipulate the test data so slightly that it does not change the classification boundary and in the spam filter case the email is obfuscated so that it passes detection. These are only a few examples of adversarial attacks.

To assist in adversarial defenses, DARPA has a new program called Guaranteeing AI Robustness against Deception (GARD) that was announced in early 2019. GARD goals are to come up with metrics that measure adversarial perturbation and the capability to defend against several attacks. Figure 4-9 shows a cycle of attacks and resulting defenses but points out that these defenses, do not generalize to new attacks. For example, each attack corresponds to a single defense, represented in Figure

4-9, where the bottom left-hand side starts with an attack and right-hand side is the related defense [131].

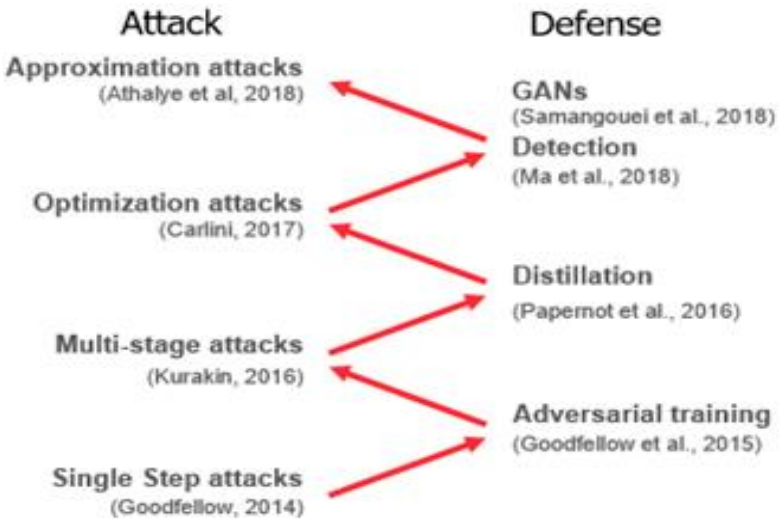


Figure 4-9: Single Attack vs Defense for Machine Learning

Machine learning classifiers are tuned to make decisions from training data, like an algorithm being able to recognize a stop sign from an image data set. There are different types of classifiers that span predictive, binary, nearest neighbor, support vector machine, and deep learning to name a few. These classifiers create boundary lines from the data they receive that helps recognize the object that it was trained for. Adversarial perturbation changes the boundary line, and this leads to

mischaracterization of the object. This is seen in Figure 4-10 where the tight boundary that distinguishes between the data types are shown [131].

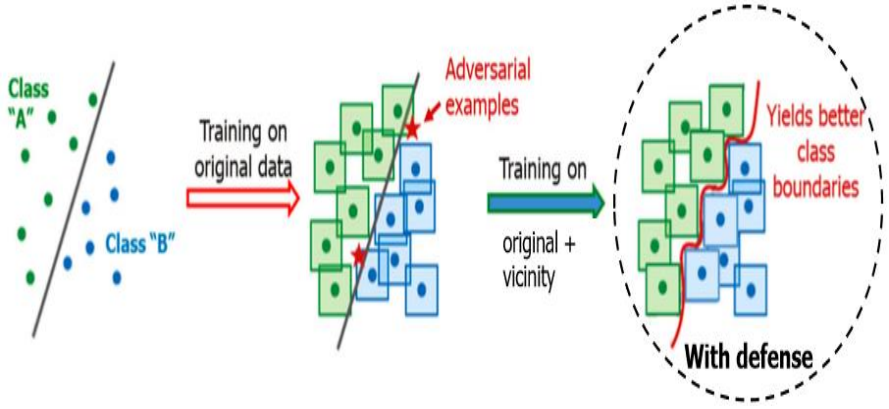


Figure 4-10: Creating defense boundaries

An adversarial example is recognizing a stop sign, but when the sign is slightly changed a mischaracterization occurs and, in some cases, causes grave danger. In Figure 4-11 a normal stop sign is seen on the left, but, on the right, slightly changing the text on the sign causes a resulting mischaracterization of the sign as indicating a 45-mph speed limit [132].

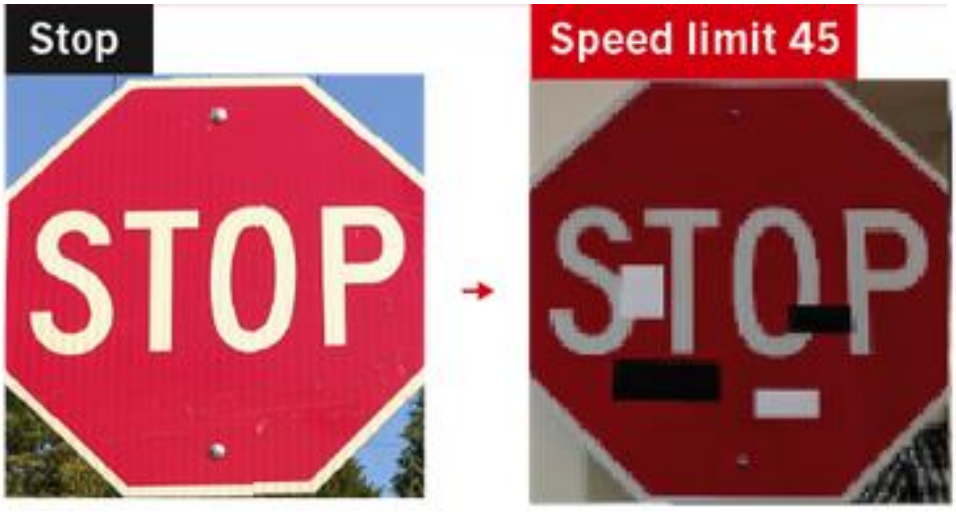


Figure 4-11: Altered text on stop sign

Another adversarial example is targeting neural network policies in reinforcement learning called adversarial policy. The policy is a set of action rules the agent can take as a function of state and environment. This policy causes the multi-agent environment to generate seemingly random and uncoordinated behavior. This behavior can be seen in Figure 4-12 where the victim (in blue) is against a normal opponent on the top and an adversarial opponent on the bottom [133].

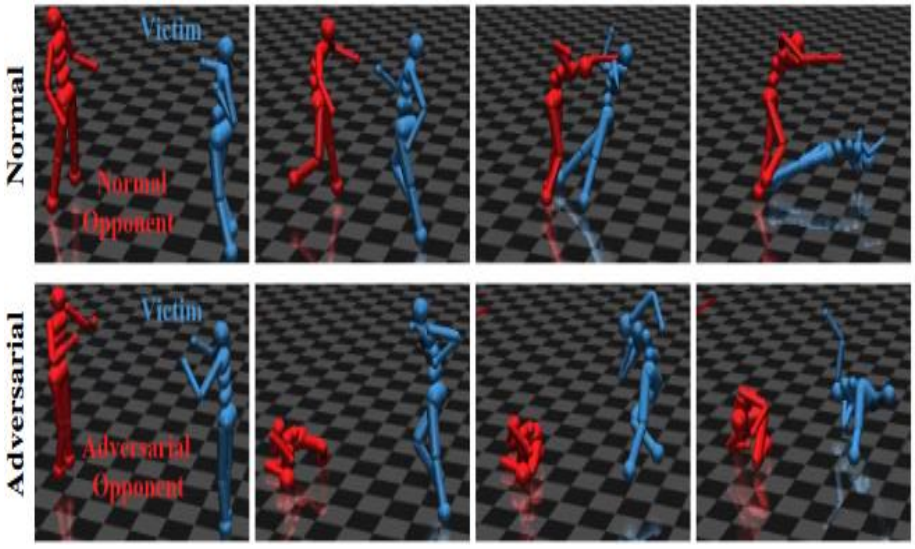


Figure 4-12: Adversarial policy in deep reinforcement learning

Based on the above we take a deeper look at these adversarial examples. In the stop sign example shown in Figure 4-11, the left-hand side is recognized as x and classified originally as target t = arg-max F(x). The right-hand side is a new desired target where t′ not equal to t. This is called x′ a targeted adversarial example if arg-max F(x′) = t′ and x′ is close to x given a distance metric [134]. Another type of adversarial example is called the Fast Gradient Method (FGM), which is a one-step algorithm that takes a single step in the direction of the gradient. $x' = FGM(x) = x + \in sign(\nabla x\ J_{(\theta,x,y)})$, where θ is the parameters of a model, x is the input to the model, y is the targets associated with x, J(θ,x,y) is the cost used to train the neural network and

$\in$ controls the step size taken [135]. The cost function can be linearized around the current value of θ and obtain an optimal max-norm constrained perturbation. To determine where boundary lines are placed, a number of techniques are used for distance metrics in order to differentiate between *x* and *x'*, original and perturbed data. We highlight three techniques, first is Minkowski's general formula, second is Lipchitz continuity and the third is an activation function for finding or determining distance functions.

The minimum distance of a misclassified nearby adversarial example to x is the minimum adversarial distortion required to alter the target model's prediction, which is referred to as the lower bound. A certified boundary guarantees the region around x that the classifier decision cannot be influenced from all types of perturbations in that region. In other words, the robustness is being able to detect perturbation as close to *x* as possible and in some cases, this is an approximation or an exact guarantee to determining the lower boundary point. In order to evaluate the distance, sometimes called distortion or error between x' and x, the generalized Minkowski's formula is used to calculate the distance metric within p-norm space. For p-norm when p=1, this is a Manhattan distance, when p=2 is the Euclidean distance, and p=∞ a Chebyshev distance. This is shown in Figure 4-13 below.

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

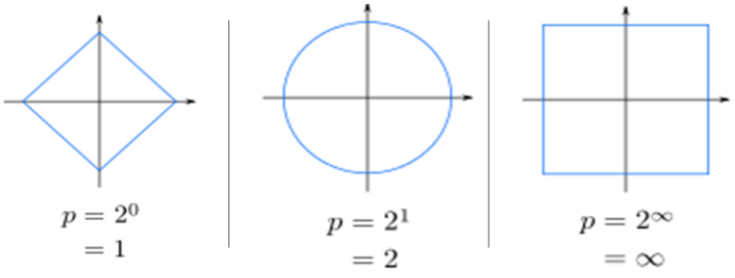


Figure 4-13: Minkowski's distance metric for p-norm

Another technique that is used is called the Lipchitz continuity for determining distortion between measured space. The equation of $| f(X_1 - f(X_2) | \leq K | X_1 - X_2 |$, where two metric spaces are given and if there exists a real constant where $K \geq 0$, for all x_1 and x_2 , K is referred to the Lipchitz constant.

The activation function technique is when a neuron or node fires in a Neural Network (NN) that allows the input data to pass (directly or transformed) to the output stage depending on the layer depth. Activation functions can be linear or non-linear, where the latter can handle more complex data types. Rectified Linear Unit (ReLU), hyperbolic tangent, sigmoid, and arctan are several types of activation functions. In relations to digital circuits, this is the rising edge that turns a gate on or off, but with different ranges and signal characteristics. NNs have building blocks that are used to transform data from input to output nodes or to different layers. A Pooling block reduces the number of parameters and computation in the network by reducing the spatial size of the data represented. A Residual block feeds the next layer but also supports a jump in layers from 2 to 3 hops away and a Batch normalization blocks provides a standardized technique for inputs to the next layer, so that a reduction in training cycles and learning process can be achieved. A newer technique is used by applying two linear bounds on each activation function where the output of each layer

is constrained by these two terms. We present some of our search results that utilize some of these techniques to determine an approach to finding a robustness metric in three different groupings.

One group tried to dwarf adversarial attacks by increasing the accuracy of the model using the p norms, so that both adversarial and training data combined would be used to train the model. But the findings from Tsipras et al.[136], Nakkiran[137], and Su et al.[138] pointed out the need to differentiate between the two types of training (classifier and adversarial) and not make the classifier so tuned to adversarial data, because the model's overall accuracy decreases.

In another group of findings, the authors Weng et al.[139] derived their work from Hein and Andriushchenko's[140] using Extreme Value vs Mean Value Theory and Lipchitz continuity for determining lower and upper bound approximation for adversarial detection. Weng et al, presents the Cross Lipschitz Extreme Value for nEtwork Robustness (CLEVER) for determining a robustness metric.

What is it – CLEVER is a technique for determining the lower bound of a neural network that achieves a robustness metric against adversarial attacks. In other words, it determines the minimal distortion level to achieve an adversarial attack from the original data.

How does it work – This is the intuitive explanation of how finding the lower bound is performed using Lipchitz continuity in the paper. In Figure 4-14 the function value $g(x) = fc(x) - fj(x)$ near point x_0 is inside a double cone formed by two lines passing $(x_0; g(x_0))$ and with slopes equal to $\pm$ Lq, where Lq is the (local) Lipschitz constant of $g(x)$ near x_0. This means that the function value of $g(x)$ around x_0, for example $g(x_0 + \delta)$ is bounded by $g(x_0)$ given δ (adversarial perturbation) and Lq

(Lipschitz constant). When $g(x_0 + \delta)$ is decreased to 0, an adversarial example is found and the minimal change of δ is the distortion difference between adversarial example and original data) is given by $g(x_0)/ Lq$ [139]. The cross Lipschitz constant is the cross terms fc(x) - fj(x) of the function.

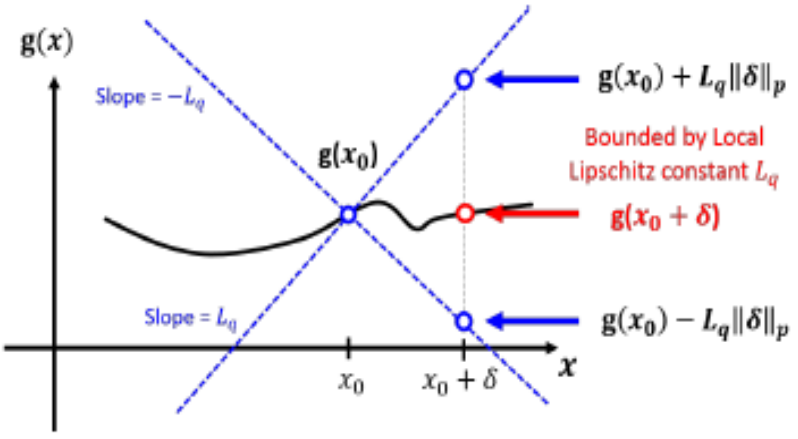


Figure 4-14: Lower bound explanation

There are two algorithms that are defined the first algorithm is for targeted attacks and the second is the same except for removing the target class that suites the non-targeted attacks. The goal of the targeted attack is to make the model misclassify by predicting the adversarial example as the intended target class instead of the true class. The untargeted attack does not have a target class, but instead it tries to make the target model misclassify by predicting the adversarial example as a class, rather than the original class. Targeted was defined as three different classes called random target, least likely, and top-2. The random target class is defined as randomly selecting a target. The least likely class is the lowest probability when predicting the original example. The top-2 class is the highest probability except for the true class, which is usually the easiest target to attack. Basically, sample a set of points within a circle boundary using

Lp=2 as in step 4, then find the max value and store that in S for each group of points or batch. In step 9, a maximum likelihood estimation is performed on the reverse of the Weibull distribution parameters (a Cumulative Distribution Function (CDF) has a finite right endpoint (denoted as a_w). The right endpoint reveals the upper limit of the distribution, known as the extreme value. This equation is the inverse Weibull distribution, where G(y) is the CDF of max y's or limit distribution and a_w, b_w and c_w are the location, scale and shape parameters, respectively.

$G(y) = \exp\left\{-\left(\frac{a_w - y}{b_w}\right)^{c_w}\right\}$, where $y < a_w$ and when $y > a_w$, it becomes 1.
The extreme value is exactly the unknown local cross Lipschitz constant [139].

Algorithm 1: `CLEVER-t`, compute **CLEVER** score for targeted attack

Input: a K-class classifier $f(x)$, data example x_0 with predicted class c, target class j, batch size N_b, number of samples per batch N_s, perturbation norm p, maximum perturbation R
Result: CLEVER Score $\mu \in \mathbb{R}_+$ for target class j

$S \leftarrow \{\emptyset\}$, $g(x) \leftarrow f_c(x) - f_j(x)$, $q \leftarrow \frac{p}{p-1}$.
for $i \leftarrow 1$ **to** N_b **do**
  for $k \leftarrow 1$ **to** N_s **do**
    randomly select a point $x^{(i,k)} \in B_p(x_0, R)$
    compute $b_{ik} \leftarrow \|\nabla g(x^{(i,k)})\|_q$ via back propagation
  end
  $S \leftarrow S \cup \{\max_k\{b_{ik}\}\}$
end
$\hat{a}_W \leftarrow$ MLE of location parameter of reverse Weibull distribution on S
$\mu \leftarrow \min(\frac{g(x_0)}{\hat{a}}, R)$

Algorithm 2: `CLEVER-u`, compute **CLEVER** score for un-targeted attack

Input: Same as Algorithm 1, but without a target class j
Result: CLEVER score $\nu \in \mathbb{R}_+$ for un-targeted attack

for $j \leftarrow 1$ **to** K, $j \neq c$ **do**
  $\mu_j \leftarrow$ `CLEVER-t`$(f, x_0, c, j, N_b, N_s, p, R)$
end
$\nu \leftarrow \min_j\{\mu_j\}$

To illustrate this further Figure 4-15 [139] is a two-dimensional space with three hyperplanes and the two balls (p norm) are bounded by a radius. The three hyperplanes $w_i x + b_i = 0$ divide the space into seven regions (with different colors). The red dash line

encloses the ball B2(x0;R1) and the blue dash line encloses a larger ball B2(x0;R2). A sampling is done within the two ball spheres as to finding the max y's.

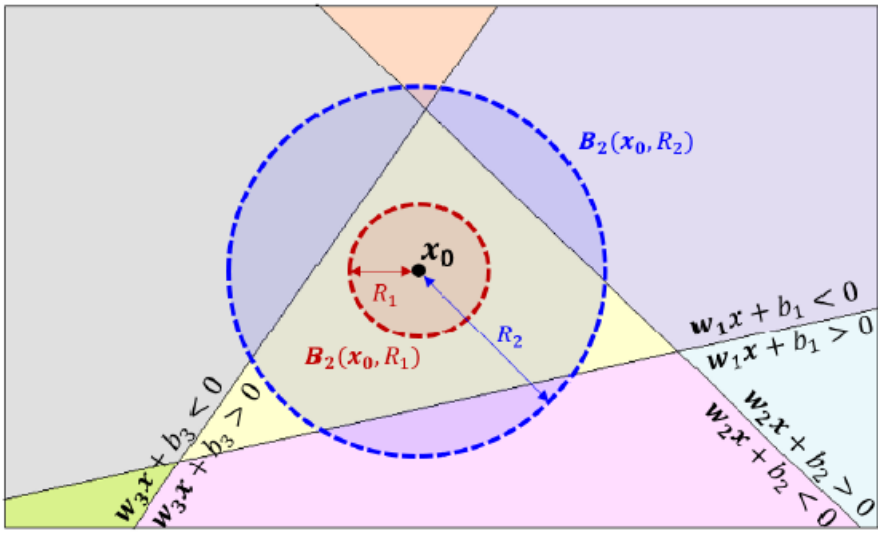


Figure 4-15: Illustration of the algorithm

What does it do well – CLEVER provides a means to approximate a formal guarantee on AI robustness by finding the lower bound on an implementation for detecting adversarial perturbation. This is useful in understanding the security limits of AI algorithms that run on autonomous robotic systems. CLEVER is attack-agnostic, works for large neural networks, and it is computationally feasible.

Shortcomings – CLEVER was validated using image data and tested against a few other adversarial examples. This small sampling of tests needs to be expanded into a larger test space for this to be considered a generalized solution. However, it does provide a step in the right direction to a qualitative metric for AI robustness against adversarial attacks.

A third group leveraged the work of CLEVER as an approximation for defining a lower bound on adversarial perturbation detection, where CROWN (another detection technique) extends to create an exact certification for general activation functions in

DNN. This same research group proposes the CNN-Cert, a certifying framework to accommodate general DNNs [141]. This technique focuses on using two linear bounds constrained on activation functions for each output layer.

What is it – Boopathy et al., presents a framework for certifying robustness in neural networks with guarantees on adversarial attack detection. This framework is called CNN-Cert for Convolution Neutral Networks, but the authors state that it can support a number of other architectures including max-pooling layers, batch normalization, and residual blocks, as well as general activation functions [141].

How does it work – Let f(x) be a neural network classifier function and x_0 be an input data point. They use σ (·) to denote the coordinate-wise activation function in the neural networks. Some popular choices of σ include ReLU:σ(y) =max(y,0), hyperbolic tangent: σ(y) = tanh(y), sigmoid: σ(y) = 1/(1+e−y) and arctan: σ(y) = tan−1(y). The symbol ∗ denotes the convolution operation and Φr (x) denotes the output of r-th layer building block, which is a function of an input x. Also, denote Φr−1 as the input of activation layer. The superscripts denote index of layers and subscripts to denote upper bound (U), lower bound(L). The general form of the equation is as follows:

$A_L^0 * x + B_L^0 \leq \mathbf{\Phi r(x)} \leq A_U^0 * x + B_U^0$, where $A_U^r, B_{U,}^r A_L^r, B_{L,}^r$ are constant tensors related to weights and bias as well as the corresponding parameters in the linear bounds of each neuron [141]. This general form is applied to the different residual blocks, pooling layers, and batch normalization layers by deriving the linear upper and lower bounds using the right-hand and left-hand sides of the general equation. CNN-

Cert was tested against different image datasets using MMIST and CIFAR (consists of 60000 32x32 color images in 10 classes) using 4 to 7 layers and different filters.

What does it do well – CNN-Cert provides a guarantee to finding a certifying robustness region or the minimal distortion detection using linear bounds on outputs. CNN-Cert was tested against different image datasets using MMIST and CIFAR using 4 to 7 layers and different filters with > 11% improvement in performance as compared to CLEVER. Since CNN is a form of DNN, this technique can be applied to different autonomous robotic algorithms to determine the certified region or lower bound.

Shortcomings – Similar to the comments above on CLEVER, since this is from the same research group.

From the two different approaches for determining a certified region for detecting adversarial perturbation, one being an approximation and the other being an exact guarantee in AI DNNs, we can derive a rating for AI implementations. By using these lower boundary techniques, a minimum distortion level is established. From this point we can define ranges for rating AI implementations against these known values. To better illustrate this concept the following Figure 4-16 shows a center region as equal to the certified region, and each subsequent ring is corelated to the rating or strength of the AI implementation using a distance function. Let x be the certified region and y be the AI implementation, we can use the p-norm distance equation to determine the differences for adversarial perturbation detection. We call R_m the AI Robustness Metric for rating the strength of an AI implementation.

$R_m = || x- y||$, *where* $y > 0$ *and* $0 < R_m \leq 1$

As one moves further away from the certified region the rating should decrease, within 1 being the lowest

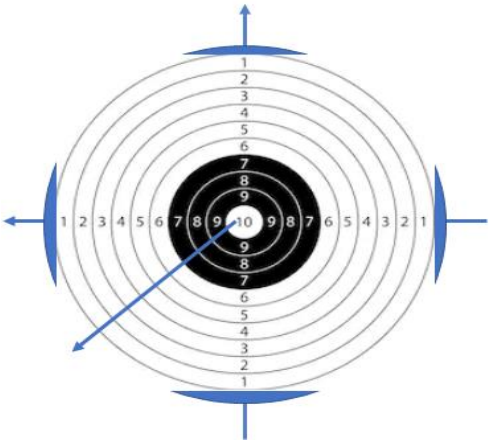


Figure 4-16: Robustness distance metric

The capability to detect the smallest perturbation distortion near *x* (original) provides a higher robustness metric score. In the case that CLEVER score is attack-agnostic, a higher score number indicates that the network is likely to be less vulnerable to adversarial examples; the same applies to CNN_Cert [142]. So, with finding the lower bound, one can start to formulate a way to express a rating system for how AI robustness metrics can be applied to different AI implementations. Since each defense technique has a distance/error from the certified area (boundary) where perturbations can be detected, we can consider these values as trust metrics in a continuous set of ranges between [0,1] as part of the holistic trust model.

Unlike some of the other measures we have seen so far, this is still an area that is less mature. However, in the spirit of the previous discussions, we can consider a system with no "defenses" against attack a more vulnerable system then one that defends itself against multiple types of attack. Therefore, for our purposes we can consider it the same way we are considering other metrics. AI trust metrics and costs

will need to be fused together for a set of metrics that will be considered for the holistic model.

Another important element to consider is the training and test data that is being used and how this is being protected for distribution. The supplier of AI components should have a repository of this data, but how is this protected and how is the authenticity of the data checked? The AI component vendor will need to be considered under the supplier trust metrics category as well, since the training and test data will need to be obtained for validation/certification.

4.7 Supply Chain Level Trust Evaluation

A supply chain is defined as set of resources and processes that upon placement of a purchase order begins with the sourcing of raw material and extends through the manufacturing, processing, handling and delivery of goods, and related services to the purchaser [143]. Supply chain vendors touch many components of an autonomous robotic system. Fundamentally, these vendors supply all the components used in a system, with individual vendors supplying anything from a single hardware or software component (including AI) to a complete system. Therefore, to ensure the trustworthiness of a system, each of the components or systems must come from entities that are reputable, provide reliable products, support services, and follow best security practices. The system, hardware, software, and AI layers in our holistic security architecture may present security risks from their suppliers. We believe that the supplier is an important entity to incorporate into the holistic security architecture.

A supplier can produce either a whole system or a system of systems as a component within a larger system. A security threat at the system level can be viewed

in this example as a drone manufacturer called DJI where backdoor exploits were found that could divert drone data to a remote server or control it. The problem with DJI's backdoors, however, is that hackers or government actors can install malware through these backdoors for cyberspying [144]. This is compounded by the fact that this technology likely has a Remote Access Trojan (RAT) embedded in it [144].

For the hardware level as we discussed in the evaluation of hardware section, security threats can come in the form of malicious insertions where add, delete or modifications of gates can occur at the foundry. For example, a hardware attack technique is called TrojanZero where the researchers were able to insert additional gates into the design without being detected, using standard power analysis at post manufacturing testing phase [145]. They were able to create low probability of triggering a detection with the additional gates as well as enabling backdoor entry. The success of TrojanZero was predicated on using 3rd party testing service for the IC, where the attacker at the foundry had knowledge about the design and testing methodology. A similar hardware attack example can be applied at the Printed Circuit Board (PCB) level where malicious insertion can happen at the raw card manufacturer as discussed by Russ and Gatlin[146].

A software supply chain has several attack vectors that may span from insider malware injection into the code itself, abusing the code signing mechanism, malware injection into the software update mechanism or service, attacking open source directly, and attacks the distribution service like application stores. Some of these examples are CCleaner, SimDisk and ShaowPad where state actors attacked the supply chain and for code signing abuse attacks like ShadowHammer, Naid/MCRat and BlackEnergy 3. Examples of update services being attacked were Flame, CCleaner and Adobe pwdum.

Several examples of opensource being affected were RubyGems backdoor and JavaScript backdoor. A survey was performed that outlined the software supply chain security threats that included the examples mentioned above called " Breaking trust: Shades of crisis across an insecure software supply chain" by Herr et al.[147]. As a supplier of software product or service, the software development methodologies must be reviewed with the emphasis on secure assurance engineering as discussed in Section 2.

Since the topic AI/Cognitive algorithms for autonomous robotic systems is new, we have to consider the attack vectors at the supply chain. AI is a little different from software since training data, adversarial training data, robustness data, and the models themselves need to be considered as part of the packing or hosting services and delivery of these components. Another consideration is 3rd party components being infected with malware or created potential backdoor exploits that are integrated into the base solution. There is a clear need to implement security controls in the supply chain assessment and formulating a supply chain trust metric will need to account for this feature.

To identify methodologies used for evaluating trust at the supply chain level, we searched for papers focused on supply chain trust metrics, vendor trust metrics, and supply chain risks. From the search parameters the results were categorized into two groupings: standards and different approaches to defining supplier trust metrics [148]. We highlight some of the supply chain search results that are attractive to incorporate into our holistic security model.

One standard, called ISO 28001, provides a guide for best practices for implementing supply chain security, assessments, and plans [143]. Figure 4-17 shows a high-level ISO 28000 security management system for a supplier and ISO 28001

provides an eight-step methodology for security risk assessment and development of countermeasures [149]. A description of the security management is first discussed and followed by the eight-step methodology for risk and countermeasures.

- A security management policy includes the organizations charter and goals toward security and the security controls framework. The goals may consist of the overall threat and risk management framework, comply with legislation, regulatory and statutory requirements aligned with the business segments, and clearly document the overall policy/goals and communicate to stakeholders both internal and external including 3rd party.
- The security planning and risk assessment is geared toward if an event occurs what are the steps and process for the crisis. An event can be a physical failure threats and risks (incidental damage, malicious damage or terrorist or criminal action) and an operational threats and risks (security controls, human factors, equipment safety and business impact to operations).
- The Implementation and operations are defined as the infrastructure to support the security controls for the business, create a chain of command, communicate and train personnel and fully documented the policies, objective, and procedures. The operational controls as well as data controls should be implemented with archiving or off-site capability in case of an emergency or security incident. Within the operational and data controls there should be key management controls.
- Checking and corrective action should be taken that affects the security performance measurement and monitoring of the systems, security related failures, controlling the records and audit.
- The organization shall have periodical reviews of the security management systems and implement improvements when necessary.

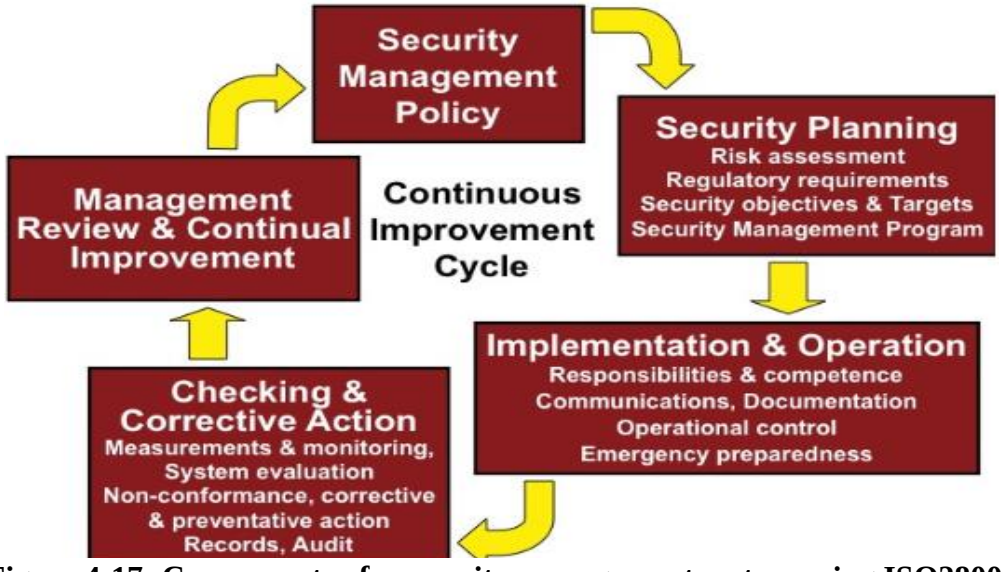


Figure 4-17: Components of a security management system using ISO28000

The eight-step methodology for security risk assessment and development of countermeasures is defined as a vulnerability assessment or analysis and the steps to perform the analysis are described below.

- Step one- is the consider the security threat scenarios, so that each scenario is broken down and analyzed
- Step two – From the analysis determine the consequences and classify them
- Step three – Rate each scenario and associated security incidents with a probability of success
- Step four – Create a security incident scoring for the business
- Step five – From the security scenarios and incident create countermeasures
- Step six – Take results from five and implement them
- Step seven – Evaluate step six
- Step eight – repeat the process

A supplier that does not support a security management system will be exposed to a number of vulnerabilities. Autonomous systems are complex, depending on the functions that they perform. For example a vehicle can have 70 to 100 embedded processors and a plethora of sensors for object detection/navigation, not including the communications network [150] [151]. Many of the components are COTS and open source software is commonly used. The following packages are the top open source projects used in autonomous vehicles: Autoware, Apollo, EB Robinos & EB Robinos Predictor, NVIDIA® DriveWorks, and OpenPilot [152]. A new autonomous operating system is open sourced based on ROS 2 called Apex [153]. Therefore, supply chain trust metrics are important, and several factors should be considered when selecting a set of quantitative values for assessing supply chain trust.

Combining the security best practices as mentioned above within the attributes that a supplier might be rated against are shown in Figure 4-18 [154]. In addition, the following items should also be included in the supplier assessment: the quality of the product/services, reliability, and maintainability of the product/deliverables. Finally,

supplier assessment should include the security of the item that also includes the organization security plan, security incidences (including history), any penalties that have been incurred, and financial stability. This combination provides a supplier trust metric that covers the autonomous robotic system layers at a system level or at the individual component level. A Supplier Assessment Management System (SAMS) provides objective measurement criteria for supplier performance in eight major categories.

Figure 4-18: Supplier Assessment Management System (SAMS): 8 Categories of Assessment

As a result of our research for supplier trust metrics, the SAP score card is presented as an example set of values to be used for metric values.

What is it – A description of a supplier assessment score cards are presented by Systems, Applications and Products (SAP) and Northrop Grumman (NG) [154] that utilizes a framework called the Supplier Assessment Management System (SAMS).

How does it work –The eight categories in the SAMS framework are rated with the following values, using SAP values: blue= 100 to 91, green=90 to 75, yellow=74 to 51 and red=50 to 0. NG Supplier Scoring on each category is performed using these metrics: Unsatisfactory < 2, Marginal 2 to 2.7, Satisfactory 3.75 to 2.76, and Excellent 4 to 3.76.

What does it do well – This provides a set of metrics that can be used for the supply chain vendors related to a system level or individual component level of an autonomous robotic system. By using the numbering of the SAP score card and reference labels from NG score card, the result is a combined set of trust metric values that can be used in our model.

Shortcomings – While this provides a good set of metrics for the supplier vendor, this will also need to cover the AI components that have been described above. The topic of security will need to be introduced to suppliers that are not familiar with it and the security management system.

We have covered the layers of an autonomous robotic system that included the system, hardware, software, and cognitive layers that will have a supply chain trust metric associated with its components in the holistic security model. By combining the architecture layers and supply chain trust metrics we capture the origins of the components and reveal the risk beforehand vs after system deployment.

4.8 Assessment Techniques

From our search results for each individual evaluation layer of the holistic architecture, several assessment techniques have been discussed. These assessment techniques have ranged from pen and paper assessment of items on a checklist to visualization/automating the assessment process using graphic tools in both a static and dynamic manner. For simple systems pen and paper works well but falls apart quickly as complexity (and accounting for system interactions) increases. The reason we are talking about assessment techniques is that having metrics is only one part of the puzzle. The next part is evaluating a system based on those metrics. Performing the evaluation can be done in a number of ways, as for visualization/automating there are different techniques that have been applied, including the use of Attack Graphs, Fault Trees, Petri Nets, BNs, and DBNs.

The first assessment technique is the Attack graph for visually representing the asset and target related to the security domain.

What is it: Attack Graphs/Trees are conceptual diagrams showing how an asset, or target, might be attacked. These graphs are generated by an analyst/automation that has obtained data from host scanning tools and network diagrams that includes connectivity between hosts. The paths are assessed and ranked from attacker to target using the connectivity connections.

How does it work: Attack trees are multi-leveled diagrams consisting of one root, leaves, and children. From the bottom up, child nodes are conditions which must be satisfied (true/false) to make the direct parent node true; when the attack condition at the node is true, the attack is complete. Each node may be satisfied only by its direct child node. An example of an Attack Graph/Tree is shown in Figure 4-19 [155] [156]

which demonstrates how an attacker is able to conduct a series of exploits on Secure SHell (SSH). The SSH is used to encrypt network services like remote command-line, login, and remote command execution. The sshd is the daemon service for SSH and the bof is the buffer overflow on a sequence of host computing devices. The sshd_bof exploit, acquires user privileges (user) on each box. This example shows the sequence of exploits and the preconditions (a set of system properties that must exist for an exploit to be successful and in this case the sshd being executed on the host) that are necessary for a successful attack.

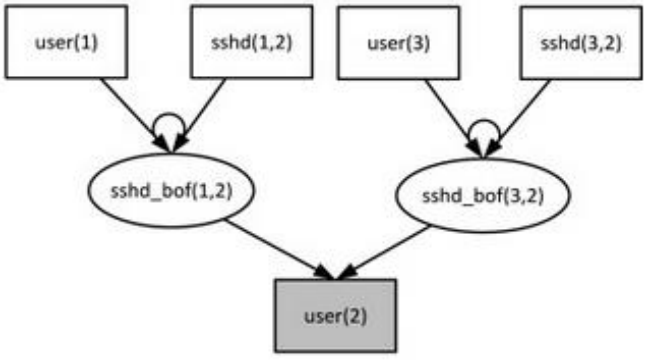


Figure 4-19: An example of an Attack Graph

What does it do well: An easy way to represent potential attack vectors is using a tree diagram that can be automatically generated using analysis tools. This will need to be extended for the autonomous robotic system security architecture. This will be considered as part of the NVD data being assessed for the robotic system when CVEs are available.

Shortcomings: An attack graph can be exhaustive (all possible outcomes must be known) for analyzing attack vectors. As the number of nodes grows, it becomes difficult both in resources and time to perform the analysis. However, the simplicity of these graphs makes it ideal for security assessment, when isolating the network paths and reduced nodes to analysis.

The next technique is similar to an Attack graph but is called a Fault tree.

What is it: Fault Tree is a top-down, failure analysis in which an undesired state of a system is analyzed using Boolean gates to combine a series of lower-level events [157]. One method of creating a Fault Tree is by following the steps for a Method Of Cut Sets (MOCUS) [158]. MOCUS provides a deterministic result that requires less resources to calculate the top event's probability, thus reducing error and improving performance.

1. Create a table where each row of the table represents a cut set, and each column represents a basic event in the cut set.
2. Insert the top event of the Fault Tree in the first column of the first row.
3. Scan through the table, and for each Fault Tree gate:
a. If the gate in an AND gate, then insert each of its input in a new column.
b. If the gate is an OR gate, then insert each of its input in a new row.
4. Repeat step 3 until all the gates in the Fault Tree is explored and the table only contains the basic events.
5. Use Boolean laws to remove all redundancies within the table.

How does it work: The Fault Tree Analysis method is mainly used in safety and reliability engineering to understand how systems can fail, to identify the best ways to reduce risk, and to determine/estimate failure event rates or a particular system level (functional) failure. In the case of security, the same example from the Attack graph

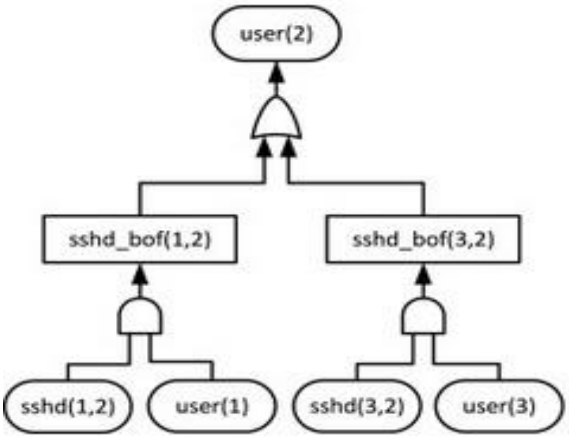


Figure 4-20: An example of the Fault Tree

above is reconfigured for a Fault Tree analysis as shown in Figure 4-20 [156] [157], where the gates (AND/OR) are used.

What does it do well: Fault Trees provide a visual graph of the system that represents relationships between events and their causes. While this is a common model for fault analysis, it can still be used for assessing trust in a system. As seen by the figures, the comparison between an Attack graph and a Fault tree is similar.

Shortcomings: In utilizing Fault Trees, a large tree needs to enumerate all possible sequences of failures in a complex system and has limitations. These limitations are a result of the exhaustive computational resources required to produce an output for all sequences of failure. Classic Fault Tree models that are combinatorial like, Attack and Fault Trees, do not support situations that have complex dependencies at the system or sub system levels. These dependencies are failure characteristics such as functional dependent events and priorities of failure events.

The Petri net technique is different from the first two, because it can represent dependencies transitions with arcs in behavioral control where in the other two cases this was not feasible.

What is it: A *Petri net* is a graphical tool for describing the control flow behavior of concurrent processes in systems, which was introduced in 1960's.

How does it work: Petri Net is a directed bi-graph and is defined as a 5-tuple N = (P, T, F, W, M0), where P is finite set of places, T is a finite set of transitions, F is a set of arcs, W is a weight function and M0 is an initial marking. Transitions (events that may occur) are represented by bars and places (conditions), which are represented by circles. The directed k-weighted arcs (a measure of the arc multiplicity) describe which places are pre- and/or postconditions for which transitions (signified by arrows).

The state of a process is modeled by tokens in places and a state is also called a marking, as shown in Figure 4-21 [159]. Petri nets are concurrent, nondeterministic models, meaning that they can support multiple transitions of events at the same time and the firing of the events can occur at different orders. This example can be shown in Figure 4-21 where T1 is splits into p2 and p3 paths.

What does it do well: PN's are intuitive and model concurrency (partial order) well. PNs have been adapted to supporting temporal events and probabilities by different extensions. By supporting these extensions this becomes a very useful modeling tool.

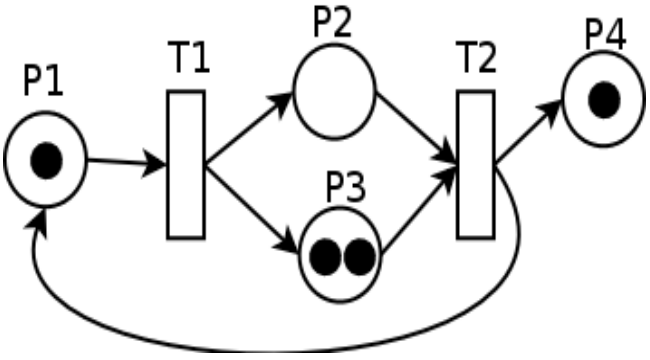


Figure 4-21: An example of a Petri Net

Shortcomings: The disadvantages of a PN are that the size of the nets for modelling very complex systems become difficult to validate because the number of reachable markings blows up, making it analytically intractable. In other words, PNs are considered state space models that allow complex interactions at the system and sub-system levels but are prone to the state space explosion problem. The state space explosion problem is the size of a state space with respect to the structural scale of a PN and has a tendency of growing exponentially.

The next technique is different from the others with regard to using probabilities and undirected paths for inference. BN provides the capability to reason about the domain once a joint distribution is constructed that includes prior evidence.

What is it: BN are Probabilistic Graphical Models (PGM) that represent causality inference using variable conditional dependence. A DBN extends a BN by relating variables to each other over adjacent temporal steps. BNs use the underlying Bayes Theorem as explained below that is followed by an example.

How does it work: To propagate the level of belief in a hypothesis that is put to the network, that level of belief can be formulated to indicate the level of trust that is placed in a system based on the evidence that supports (or negates) the hypothesis. For example, from Bayes Theorem:

$P(X|Y) = (P(Y|X) * P(X))$ / P(Y), which is the Posterior = (Likelihood * Prior) / Evidence). Posterior is after an observation has occurred, Likelihood is the probability that the event will happen, and Prior is the data before it is observed/evidence set where something is known.

The following example, shown in Figure 4-22, applies a Bayesian analysis to two different boards, the one on the left is a low-cost hobbyist processor board with an Arduino-like processor (overseas manufacturer and open source software) versus a TI MSP430 development board (US manufacturer, reputable suppliers, etc.). The microprocessor trust metric and supplier trust metric nodes have a causal relationship with the board. The conditional probability table is constructed by the first column being hardware trust metric using values (none, low, low to medium, medium to high and high), while the second column is the supplier trust metric with values for each row (unsatisfactory, marginal, satisfactory, and excellent). The probabilities are shown in

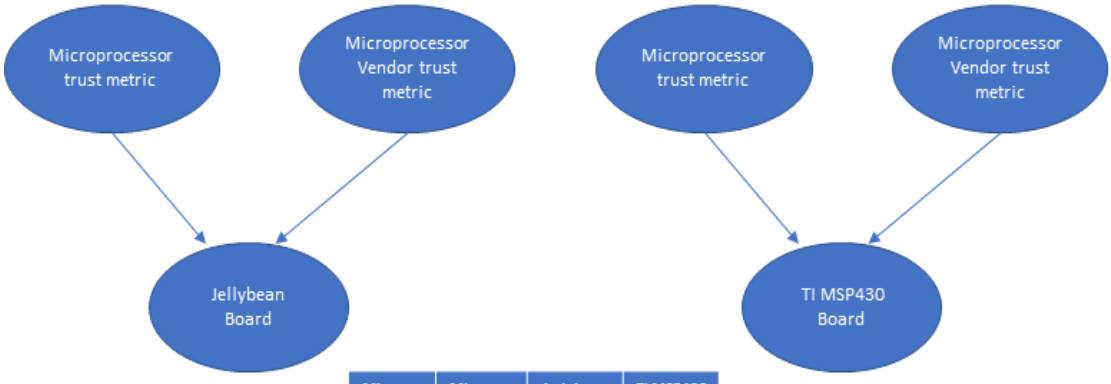
the next two columns corresponding to the different processors. By intuition, the TI MSP430 provides a higher trust level than the Arduino-like chip, since the manufacturer is US-based, the supplier evaluation ratings and its hardware design integrity are higher as well. These were compared to the Arduino-like processor where the supplier did not have a rating and did not have complete design artifacts (back to the hardware trust metric for Logical Equivalence, Signal Activity Rate, Structural Architecture, Functional Correctness, and Power Consumption). By setting evidence on a node, one can reason about the outcome in a downward path, so setting the supply chain node to unsatisfactory will cause the board node to change by observing its state. In the case of the evidence being set on the bottom node, the two top nodes will change accordingly. This effect of change when evidence is set is known as inference and BNs provide this capability.

What does it do well: Bayesian Networks are able to perform casual inference and reason with uncertainty. When reasoning with uncertainty, the BN model can update the posterior probabilities of other states of the system when evidence is set. This technique is well suited for autonomous robotics system because of their nature in uncertainty and having the capability to reason about new evidence gained from sensors about tasks or environment.

Shortcomings: There is no standard method for constructing a BN. However, there are techniques for determining if the model has been constructed correctly.

Each of these techniques has both pros and cons associated with them as enumerated above. There have been three prongs to the development of these techniques. The first prong being the ongoing work to extend trees and PN by increasing their capabilities after they were first introduced in 1960's. The second prong, being a

combined approach, where pre-preprocessing is done in (trees and PN) and post-processing is completed in BNs. The third prong being an only BN usage model for security analysis.



Micropro cessor Trust Metric	Micropro cessor Vendor Trust Metric	Arduino (like built overseas)	TI MSP430
None	U,M,S,E	.5	.5
Low	U,M,S,E	.9	.1
Low to Med	U,M,S,E	.7	.3
Med to High	U,M,S,E	.2	.8
High	U,M,S,E	.1	.9

Figure 4-22: A BN example of two microprocessors

In addition to the approaches described above, we also observed that attack graphs combined with CVSS base metrics were common methods for security assessment. Others have used PNs to cover a small portion of the architecture, while others are using AI techniques to perform prediction on security outcomes. A common problem with these techniques is that they do not support uncertainties that may arise from autonomous robotic systems.

On the other hand, BNs provide a number of benefits over these other approaches. These include the casual inference where reasoning can be performed from observations and the compactness of information can be encoded into joint probability

distributions. In fact, the efficiency of inference algorithms can be seen as a main reason for the success of BNs, since querying general graphs is an NP-hard problem [160], [161] [162]. Another strength of BNs is their ability to update the model, i.e., compute a posterior distribution when new information is available [162]. Using a BN with the associate metrics is one potential assessment solution for autonomous robotic systems.

4.9 A Solution Outline

We outline a solution that takes a number of the autonomous robotic system layers and brings them together to form a holistic trust model. This trust model is different from other approaches because we are proposing a complete solution from a security perspective, eliminating the gaps left by other techniques in the evaluation of system, hardware, and software layers. Our model also includes AI robustness attributes and supply chain characteristics. The overall complexity of the space and the security problems are bad enough in a controlled environment, now add high-value targets in an unconstrained environment where they get much worse. The unconstrained environment makes the problem computationally intractable using earlier approaches. Thus, there is a need to come up with a way to make assessment more computationally feasible.

A probabilistic approach to analysis using BNs provides a natural way to reason about uncertainty. BN-based models allow for efficient factorization of the set of system states, without the need for an explicit representation of the whole joint distribution; moreover, they have the additional advantage of inference algorithms available for the analysis of any posteriori situation of interest (i.e. evidence can be gathered by a monitoring system) [163]. Bayesian Inference simplifies the way to

reason about a complex domain problem like security for autonomous robotic systems. Our survey findings did reveal some usage of BNs, but these researchers did not define address complete systems, nor did they include cost values (collateral damage and perceived target value) in their work. Our goal is to expand this work and apply it to more complex autonomous systems.

In order to represent an autonomous robotic system architecture and assess the security of it, we have discussed the different layers (system, hardware, software, Cognitive/AI, and supplier chain) as independent trust metrics. These individual trust metrics account for the different levels depending on the security features supported by the autonomous robotic system and may be expanded to cover other elements of importance.

With each individual trust metric and its associated level, we also need to account for the collateral damage that may result from an attack on that part of the system, as well as the perceived target value of that element. In other words, we need to account for the adversary's actions and by combining these values with the trust metric we get the general probability equation:

$$TM = LV*AER* AED*ATA,$$

where TM = trust metric, LV = level value, AER = probability of adversary exploit reward and AED = probability of adversary exploit damage, ATA = likelihood of an adversary taking action to exploit.

The combination of these values provides a set of metrics that can be assigned to each corresponding component in the system. The BN will provide the casual inference by linking these components and the values for the conditional probability tables. A full joint distribution is defined as the product of the conditional distribution

of each node. This is shown in the equation below where the left-hand side is the joint distribution, the center is the conditional probability (using the chain rule), and the right-hand side is the conditional probability given the parents.

$$P(x_1 \dots x_n) = \prod_{i-1}^{n} P(x_i | x_1 \dots x_{(i-1)}) = \prod_{i-1}^{n} P(x_i | \text{parents}(x_i))$$

In general, a trust metric for a given level is created as a function, f_i, where f_i = Factor 1 * Factor 2 * … * Factor N. The number of metric values is a function of the granularity of the analysis, and generally ranges from 3 to 7 with 5 being a reasonable tradeoff between resolution and complexity. From these factors, a trust metric, T, is derived by normalizing the f_i over the range, so $T = 0 \leq |f_i| \leq .\ 1$.

For example, at the system level trust metric, we start with the related CC's EAL 1 to 5 levels and combine them with cost/reward/likelihood values, this provides five levels for the system metric and each having a cost/reward/likelihood value, then that is normalized.

STn = EALn * *AERn* * *AEDn* **ATAn*, where n is the level

ST = system trust metrics ranging in a discretized continuous set from [0, 1] where 1 is the highest and zero the lowest trust set. Where five levels will be defined for the range. The other layers HW, SW, AI and Supply chain follow the same pattern.

Next, we take the hardware design integrity values from the hardware evaluation section and do the same for the cost/reward/likelihood values.

HWT = hardware trust metrics will have a range [0, 1].

Next, we take the software base metric in the software evaluation section and do the same with the cost/reward/likelihood values.

SWT = software trust metrics will have a range [0,1].

Since cognitive/AI is a new area and using the distance metric seems to fall into the same thought process as the others, we come up with five levels with a range between [0,1] in which the closer to the certified area signifies greater protection/risk than moving further away where perturbation is easier to detect.

AR = AI robustness trust metrics will consist of five levels of distance error + cost values ranging in a discretized continues set from [0, 1] where ¬ [0,1], so that 0 is the lowest and 1 becomes the highest trust metric level.

Next, we take the supplier chain metric in the supply chain evaluation section and do the same with the cost/reward/likelihood values.

VT = Supply chain trust metrics will have a range [0,1].

Finally, we take these individual trust metrics that represent the system levels and sum the different parts into a whole system trust metric. The following equation represents the joint distribution over all variables, where $P(X_i \mid P_a(X_i)$ is the conditional probability distribution (CPD) for each variable in the network.

$$P(ST, HWT, SWT, AR, VT) = \prod_i p(X_i | P_a(X_i)$$

By using the above equation, we can reason about outcomes of the network while making observations. This allows the intractable problem to be computationally feasible using Bayesian Inference. BNs fulfill the local Markov property, so that each variable is conditionally independent of its non-descendants given its parent variables. This property reduces the joint probability to a compact form by using the chain rule.

4.10 Conclusion

We surveyed the trust metric space to determine if a set of security metrics were well defined and covered a complete robotic system (the results were in Section 2). A mind map of a robot system broke down the different layers into system, hardware, software, cognitive layer, and supplier chain to provide a holistic security view. Our findings showed that system level trust metrics are difficult and complex, and several research papers scaled the problem to a small set of components or just a specific area of a system. Table 4-4 is a summary of the findings that cover a holistic system trust model.

TABLE 4-4: SUMMARY OF TRUST METRICS

Trust Level	Trust Metric Recommendation	Values	Comment
System	EAL 1 to 5	[0,1]	Security and Safety
Hardware	Hardware Component	[0,1]	
Software	Base Impact	[0,1]	
AI Robustness	Distance Metric	[0,1]	
Supply Chain	NG Scorecard	[0,1]	

We believe that using the Bayes Inference is the correct choice to build on, since it provides several benefits to overcome uncertainties for a complex system like an autonomous robotic system. By using Bayesian Inference, we also remove the intractable problem to a computational feasible one. We provide a brief set of definitions for each of the CPD nodes that will represent the joint distribution in a BN.

As autonomous robotic systems become more popular, the standards that govern them should consider the usage of trust metrics for security. Several standards like the IEEE global initiative on Ethics and the EU's regulatory work on ethics are targeting the bias/transparence for AI systems, but little is being said about the security of these

systems. It is our belief that this survey will help establish a set of security metrics that the ethics groups can utilize.

Another consideration for security trust metrics is the autonomous vehicle space. The industry is seeing a movement in the autonomous vehicle space where the Society of Automotive Engineers (SAE) has defined a scale for automation. Level 0 is fully human controlled, and level 5 is fully automated. We are witnessing the gradual paradigm shift to more technology being incorporated into the vehicle as the goal of level 5 is being realized. Even with these advancements there have been several accidents where life was lost/injured. As these systems shift to more AI capabilities, this will increase the attack vector for nefarious actors.

In the next chapter a BN is shown where the trust metrics are defined and used to model an autonomous robotic system with security features. The next chapter also discusses how this model can be used for an internal assurance model.

CHAPTER 5 

5 A SOLUTION

Internal Cognitive Assurance Model

5.1 Introduction

In order to construct a Bayesian Network that enables reasoning about system security, one requires domain knowledge and credible metric values.

Bayesian Networks use the Bayes rule in that P(a|b) is the posterior, or degree of belief in a given b. Likewise, P(b|a) is the likelihood that the event will happen given that a has happened. In our context, P(a) is the prior or initial evidence accumulated about event a, and P(b) is the marginal probability of observing the evidence. This marginal probability, P(b), acts like a normalization constant. This can be restated as Posterior = Likelihood * Prior / Evidence and is the familiar Bayes Rule:

$$\mathrm{P}(a|b) = \frac{P(b|a)P(a)}{P(b)} \qquad (5\text{-}1)$$

In our previous research work in Chapter 4, we discussed our survey into trust metrics that allowed taking a holistic security view which included system, hardware, software, cognitive trust, robustness, and supplier layers in an autonomous robotic system design.

We highlight some of the results that were discovered during our research in Chapter 4. At the system level some of the attractive features to incorporate into our

model, are the need to categorize attack paths, determine behavioral states of an autonomous system and integrate the concept of trust and resilience. The system model should also incorporate attributes for physical protection and safety, since these elements will be needed for autonomous robotic systems that are exposed to and interact with humans. Additionally, a robot has many hardware components from processing data to sensing the environment, and these must also be accounted for in our model.

All hardware components must be integrated together, but security threats like hardware trojans, reverse engineering and counterfeiting, can compromise a system design at different points in the hardware lifecycle. Software in a robotic system extends from firmware to kernel/Operating System (OS) and above to applications. Software security vulnerabilities can come in many forms, but most exposure can be reduced in the development, testing and deployment phases. Having proper design methodologies, practices and controls provides multiple security checkpoints. An autonomous robotic system may have one or more AI algorithms in its cognitive layers to allow operating in environments having many uncertainties. AI opens a number of exploits that are not commonplace in other systems. Several adversarial attacks have been discussed in the literature related to poison (tainting the training data), white/black box (using physical and alternative training models), and evasion (misclassification by spoofing). Additionally, the supply chain through which components are procured touches nearly every component of an autonomous robotic system. Therefore, to ensure the trustworthiness of a system, each of the components or systems, must come from an entity that is reputable, provides reliable product, support services and follows best security practices. Often ignored, the supply chain is also prone to many vulnerabilities,

such as injecting malware into a component's development cycle, allowing counterfeit parts to be introduced into the production cycle and security keys being compromised.

Our conclusion was that several papers presented stove pipe solutions but did not cover the holistic view that we are trying to achieve. Other takeaways were related to techniques in the models or metrics that captured some elements that could be reused but did not accountant for costs (reward and loss) and uncertainty that in the wild, an autonomous robotic system might reasonably experience.

Although researchers have demonstrated the utility of BNs for assessing various aspects of the overall security of a system, we extend this approach in several ways. We define an internal assurance model by focusing on different layers called system, hardware, software, AI robustness and Supply chain vendor(s). In combination, these layers make up a holistic model of the security architecture that is incorporated into a Bayesian Network. We believe that this approach is superior given the alternative of simply relying on the OS to determine the security posture of the system. The OS is a large attack surface and is prone to a number of vulnerabilities. Most OSs do not support the concept of resiliency where under attack concern features shutdown to not allow the system to still function in some capacity. In the Bayesian Model we have separated each of the layers to have their own individual scores, but they are coupled together to provide a system assurance level that is dependent of the security features.

The rest of the chapter is divided as follows. First, we provide some background on Bayesian Networks in section 2. In Section 3 defines the methods for calculating each of the trust metrics used in our model. Section 4 presents the Bayesian Network models that is broken down into each layer of the system and Section 5 shows how this

methodology extends into a Dynamic Bayesian Network model where time-based events can be handled. We conclude in Section 6.

5.2 Background on Bayesian Networks

We focus on the use of Bayesian Networks for several reasons. First BNs are a natural representation for causal inference; second, they support incorporating uncertainties using probabilistic distributions; third they are able to accommodate missing data; fourth they are more accurate than Fault Trees for assessments since they explicitly represent the dependencies between events and updating probabilities. Finally, BNs can convert otherwise intractable problems into tractable ones by efficiently encoding the joint distributions for large number of variables.

Bayesian Networks are Probabilistic Graphical Models (PGMs) that represent the qualitative and quantitative relationships between a set of variables or nodes in a model structure. In this PGM model, arrows represent relationship dependencies between nodes, and each node contains probability distributions, or conditional probability tables (CPTs), that are used to represent the qualitative strength of those dependencies. Both qualitative and quantitative information can be used to define the probability distributions captured in the CPTs.

A BN is structured from a set of nodes (which represent random variables) and nodes are defined as a Parent or a Child. For example, if two nodes A and C are connected to node B, the random events from A and C are said to be conditionally independent, but conditionally dependent when given an event at B. This means that knowledge of event A occurring cannot influence C's outcome and similarly event C

occurring cannot influence A. In other words, P(A,C|B) = P(A|B), P(C|B). This relationship can be stated as A ⫫ C|B. Figure 5-1 shows three forms of conditional independence (Cascade or Casual Chain, Common Parent or Common Cause and V-Structure or Common Effect). These node configurations represent the qualitative relationships within a BN.

A Causal Chain implies that the probability of C, given B, is the same as the probability of C, given both B and A (i.e. knowing that A has occurred does not make any difference to our beliefs about C if we already know that B has occurred)) [164]. Common Cause captures the notion that both A and C have a common cause, B. Common Effect implies that both A and C are marginally independent (A and C are independent while ignoring B) but become conditionally dependent given information about B, the effect.

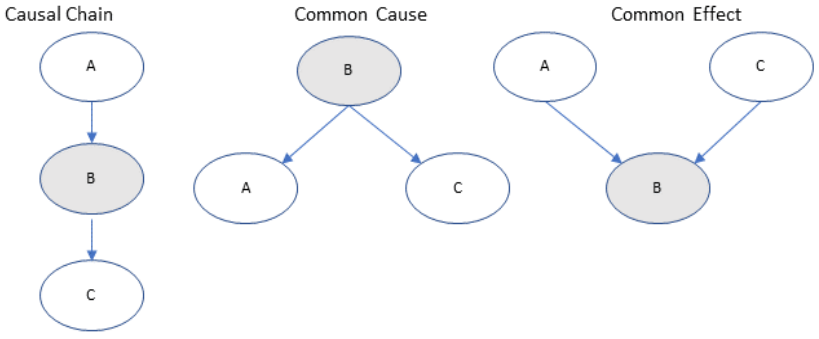


Figure 5-1: Types of Conditional Independence

We clarify some terms that may be confusing like unobserved, observation, evidence, variable/node, information/influence flow and active trail for this section. In a BN, the term "unobserved" refers to missing or hidden data. While much literature uses the term "unobserved," some authors use the terms "latent" or "hidden" synonymously, since data is not observed. This can be caused by data being unable to

be observed directly or it is completely missing. Similar to the unobserved, the observation term refers to a set of findings, but some authors use the terms finding or evidence interchangeably [165]. Different information sources are not always perfect; therefore, the observation can be uncertain and inaccurate. The term evidence is defined as hard or virtual, where a finding on a variable refers to an instantiation of the variable. A usual way to enter an observation in a BN is shown in Figure 5-2, which illustrates the propagation of evidence in belief updating.

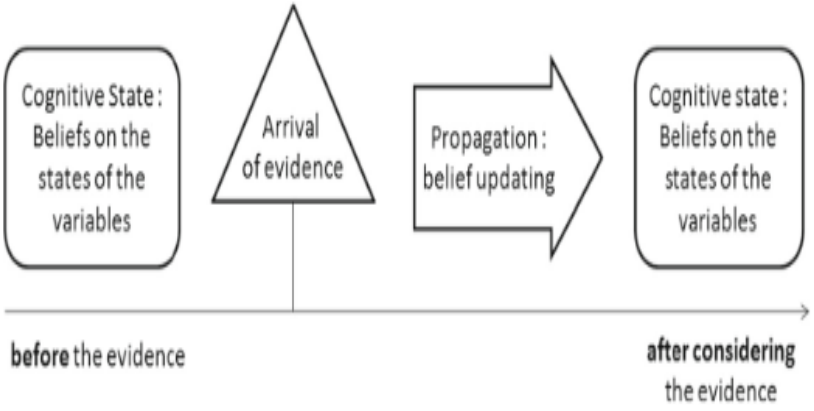


Figure 5-2: Evidence update flow

In BN, a node is a representation of a variable and two variables are connected via an edge. Once connected they are able to influence each other through information or influence flow. Moreover, the influence flow of more than two variables is called an active trial when not blocked or observed. Examples of active trails are shown in Figure 5-3 [166] where an undirected path is active if a Bayes ball travelling along it never encounters the "stop" symbol. $\longrightarrow|$ The top row shows unobserved nodes, and the bottom has shaded nodes as observed. From left to right are the influence flows starting first with (a) causal chain, next (b) the common cause, then (c) the common effect and finally, (d) the parent to child.

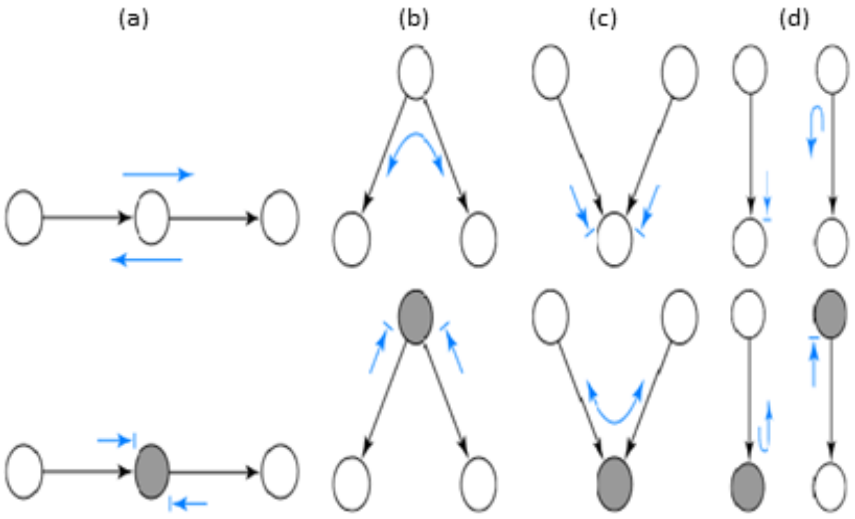


Figure 5-3: The ten rules of Bayes Ball

Direction dependent separation, also called d-separation, is important to understand network behavior and independence properties. D-separation is a criterion for deciding if a set A, of variables is independent of another set C, given a third set B. If the variables are connected, it implies dependency and if not, independence is implied. If there is no information flow or active trail, between any nodes of A and C given B, they are considered d-separated. Similar to Figure 5-3, Figure 5-4 shows four different scenarios for d-separation from left to right. The Causal Chain (a) where the active trail from A to C is blocked if B is observed and B acts like a flow control value. Similar to (a), but (b) is the Evidential effect where information flow is from C to A via B, only if B is unobserved otherwise B observed blocks. In (c) the Common cause is where A can influence C if B is not observed, otherwise B observed blocks (Observing blocks the influence flow between effects and makes it inactive). In (d) the Common effect is where A cannot influence C if B is not observed, otherwise B does not block (This is backwards from the other cases where observing an effect activates influence flow between possible causes this makes it active) [167].

In a Bayesian network, a variable, X, is conditionally independent of all other variables if it is Markov blank (given a BN, let X and Y be two variables and Z be a set of variables that does not contain X or Y. If Z d-separates X and Y, then X ⫫ Y|Z). The Markov Blanket of a node is the set containing the node's parents, children, and co-parents [168] and is the only knowledge needed to predict the behavior of that node and its children [169]. Figure 5-5 illustrates the Markov Blanket of node X_6 which includes nodes X_3, X_4, X_5, X_8, X_9, and X_{10}.

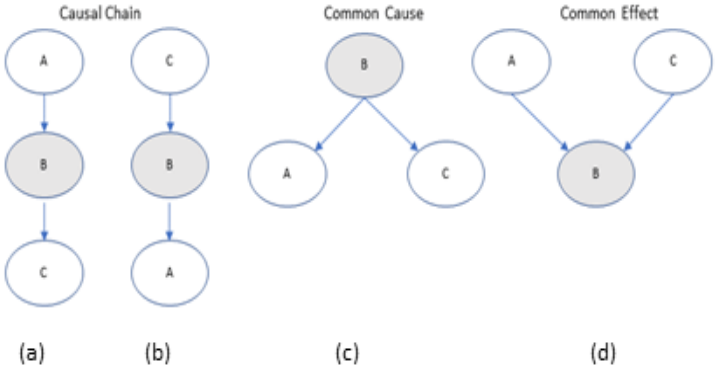


Figure 5-4: Four scenarios for d-separation

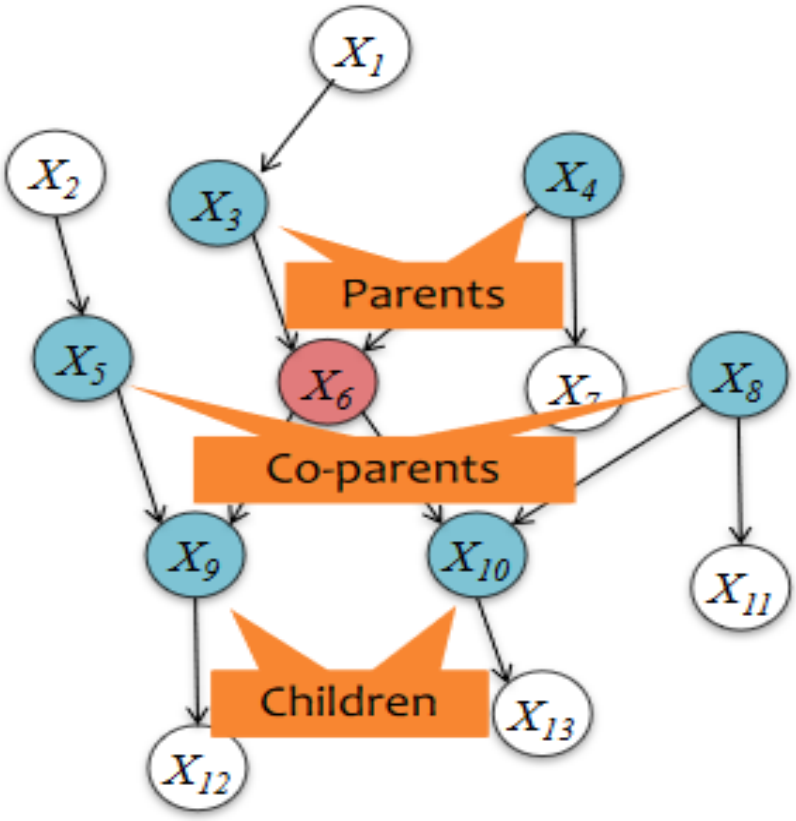


Figure 5-5: An example of a Markov Blanket

A conditional probability is the probability of event A occurring, given on a condition that event B occurred. The joint probability is the probability of events A and B happening simultaneously. A CPT is the decomposed representation of the joint probabilities and is used to display the conditional probabilities. There are different methods for developing the quantitative values for each node's random values and these are represented in the CPT. A node's random value can be either discrete or continuous, for example a discrete value can be T or F, which may represent a probability of 0.5 of either value occurring. The sum of the probability values of the possible outcomes must equal 1, whereas a continuous value is a range of values between [0,1], where $0 \leq$ value ≤ 1. These values can be acquired from domain experts, elicitation from domain users (interviews, case studies, and observations), and/or data driven (machine learning).

BNs fulfills the local Markov property so that each variable is conditionally independent of its non-descendants given its parent variables [170]. This property reduces the joint probability to a compact form by using the chain rule in equation 5-2. In Bayesian Networks (a member of joint distribution can be calculated from conditional probabilities using the chain rule).

$$P(x_1 \dots x_n) = \prod_{i-1}^{n} P(x_i | x_1 \dots x_{(i-1)}) = \prod_{i-1}^{n} P(x_i | \text{parents}(x_i)) \quad (5\text{-}2)$$

In the following example the nodes "Designer," "Manufacturer," Sensor," "Infected" and "Intermittently Operating" are shown in Figure 5-6 and are represented in the joint distribution as:

P (M, D, S, I, IO) =
P (M), P (M|D), P (S| M, D), P (I| S, M, D), P (IO| S, M, D)

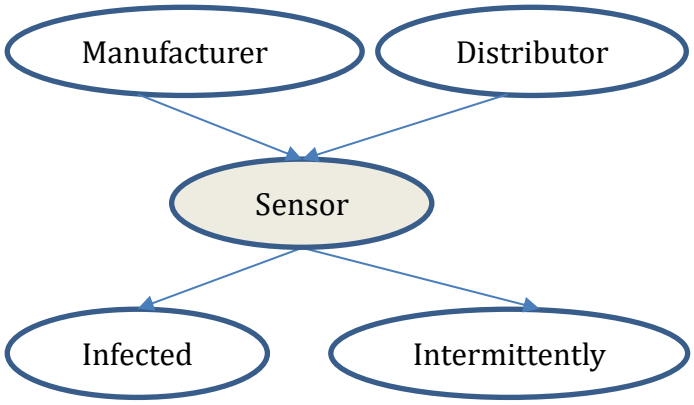


Figure 5-6: Local Markov example of a sensor set of nodes

From the above Figure 5-6, we have two casual cases, where the manufacturer, distributor and sensor nodes create a common effect and sensor, infected and intermittently operating create a common cause. These two cases were covered above where influence flow can be active or inactive depending on which nodes are observed. Applying the chain rule, the joint distribution reduces to the following:

P (M, D, S, I, IO) =
P (M), P (D), P (S| M, D), P (I| S), P (IO| S)

The local Markov property means that each variable is conditionally independent of its non-descendants given its parent variables. In the case of D and M they are D ⫫ M since D is a non-descendant of M and the reverse is also true. In the case of I being independent since S is the parent and D and M are non-descendants, so I ⫫ D, M| S is true. In the case of IO being independent since S is the parent and D and M are non-descendants, so IO ⫫ D, M| S is true.

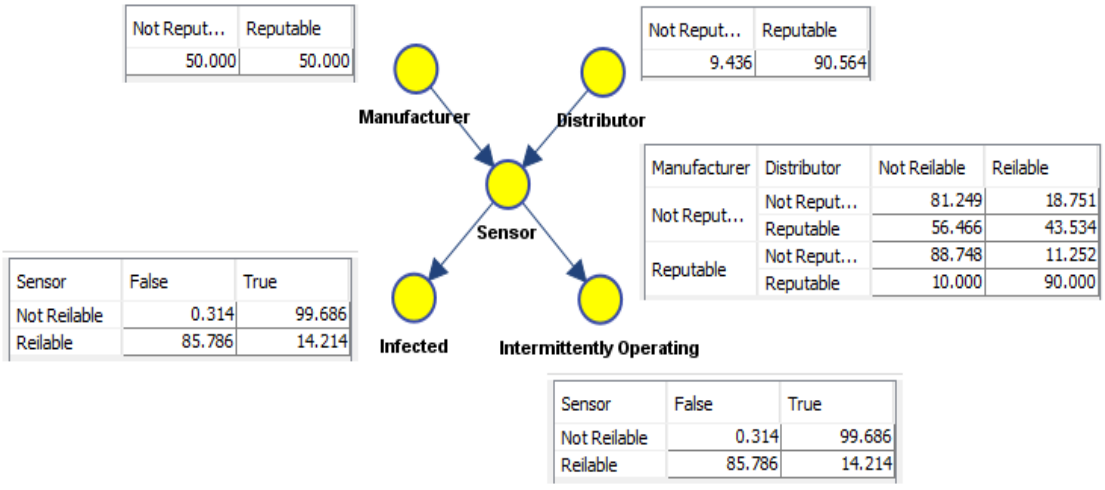


Figure 5-7: Sensor BN example

In large BNs the joint probability distribution of the model is equal to the probability of X given its parents (refer to equation 5-2), this reduces the computation since most nodes have fewer parents relative to the overall network. In other words, the full joint distribution needs K^n parameters, where K is the number of values for the variable and n is the number of variables in the BN. This means that the network grows linearly, for n variables, the order of magnitude is expressed as $O(m\ K^n)$ vs $O(K^n)$ for a BN having m(parents) < n (variables). This compact representation makes the computation problem tractable.

Back to referencing Figure 5-6, we expand on this network to formulate a BN with initial CPT values as an example of bringing the pieces together as shown in Figure 5-7. We combine the types of reasoning discussion with this illustration below. The

values in each CPT were randomly chosen for the purpose of an example, which covers diagnostic, predication, intercausal, and combined reasoning. On the bottom of Figure 5-7, variables Infected and Intermittently Operating have both the same set of values and on top Manufacturer and Distributor are slightly different with a bias toward reputable for Distributor. The Sensor node is the conditional dependency of Manufacturer and Distributor values for the sensor in a reliable or not reliable set of states. In actuality it is important to choose these values from experts or good sources, since the bias of reasoning is built on this foundation.

Once the joint probability distribution is known, queries can be made on subsets of variables. How reasoning can be performed is determined by the technique that is

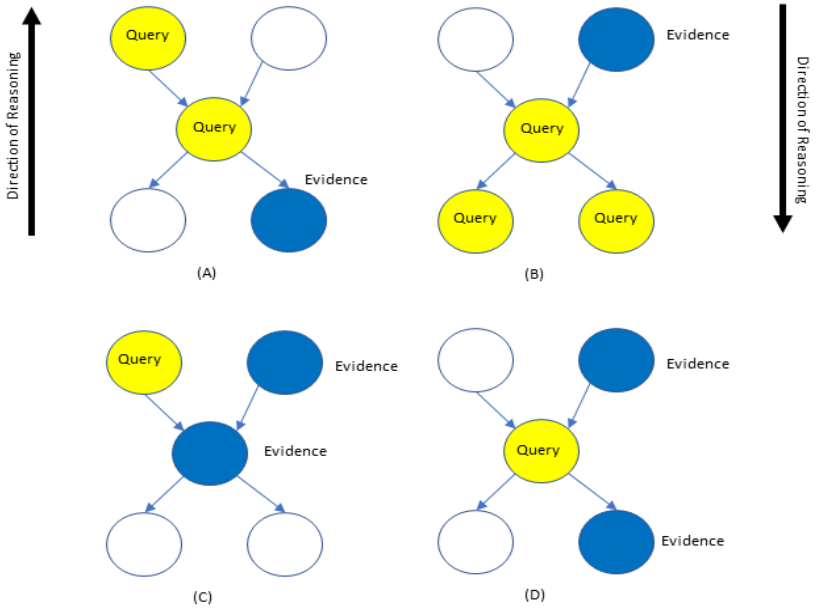


Figure 5-8: Types of reasoning

used. Figure 5-8 [164] shows the different types of reasoning that can be performed when PGMs are used. Even though BNs are directed graphs, data flow can occur in both directions. To illustrate this concept both 8A and 8B are opposite flow directions from each other.

In the diagnostic reasoning case, reasoning occurs in the opposite direction from symptom (evidence) to cause (query) shown by the blue circle and two yellow circles in Figure 5-8 (a). For example, in Figure 5-7, if observing intermittently operating, one's belief can be updated that the sensor is not reliable, and the manufacturer is not reputable. We set the evidence on intermittently operating and the manufacturer node changes from .5/.5 (reputable and non-reputable) to 68.89 % being non reputable.

In order to illustrate the calculation performed by the BayesiaLab tool, which generated the results for these reasoning examples. A closer look at how setting evidence at IO will affect the M node in a reverse information directional flow from bottom to top. The question being asked is what is the probability of M given IO? A representation of the question is shown in the below equation using Bayes equation 5-1.

$$P(M|IO) = \frac{P(IO|M)P(M)}{P(IO)}$$

We first need to solve for a couple of the terms: P(IO|M), P(S) and P(IO) in the equation.

Finding P(IO|M), we need to first find P(S|M) using the following equation, P(M) ==1 and the sensor CPT values for non-reliable.

P(S|M) = P(M)P(¬D) *.88748 + (P(M)P(D)*.1
=1*0.09436×0.88748 +1* 0.90564×0.1
= 0.1743

Using the result of the above, this gets fed into the equation below and using IO CPT table value for sensor being not reliable.

P(IO|M) = P(S|M) *P(¬IO) +P(¬S|M) *P(IO)
= 0.174× 0.00314 + (1 – 0.174) × 0.99686
=0.8236

Using the CPT values for sensor, where S will be the corresponding value in the table for each associate pair.

P(S) = P(M)P(D)*S +P(¬M)P(D)*S +P(M)P(¬D)*S +P(¬M)P(¬D)*S

=0.5*0.90564*0.1 + 0.5 *0.90564*0.55466+ 0.5*(1-0.90564) *0.88748+ 0.5*(1-0.90564) *0.81249
0.3766

Using the result from P(S) above we can now find P(IO) value.

P(IO) = P(S)*P(¬IO) +P(¬S) *P(IO)
=0.3766× 0.00314 + (1 – 0.3766) × 0.99686
=0.6225

Using Bayes equation and the supporting terms, the result is the following where the calculated is approximately 1 % close to the BayesiaLab tool produced result.

P(M|IO) = (0.8236×0.5)/0.6225 = 0.6614822

In the case of prediction reasoning, reasoning from new information about causes can lead to new beliefs about effects. This is shown in Figure 5-8 (b) where evidence is set on the blue node and queries can take place on yellow nodes. For example, a user may tell a technical support person that they are having issues with a specific sensor from a manufacturer. Even before any symptoms have been assessed, the technical support person already predicts that specific sensor has an issue and that he/she will increase the chances of the user having sensor problems and the chances that the sensor will exhibit other symptoms, such as the sensor being infected, or sensor is working intermittently are increased. To illustrate prediction reasoning in Figure 5-7 when the initial setting for the distributor, sensor, infected and intermittently operating are (.9056, reputable, .6188 reliable, .5321 false, and .5321 false) respectively. When evidence is set on distributor being non reputable, the model is updated with (1, non-reputable, .85 non reliable, .8686 true, and .8686 true) respectively.

First set the initial values by using the below equation, P(D) ==1 and the sensor CPT values for reliable.

$$P(S|D) = P(D)P(\neg M) *S + P(D)P(M)*S$$
$$=1*.5\times0.43534 +1* .5\times0.9$$
$$= 0.66767$$

Since both I and IO have the same value in their CPT table a single equation will be used where P(I|D) or P(IO|D) are the same.

$$P(I|D) = P(S|D)\ P(I) + P(\neg S|D)\ P\ (\neg I)$$
$$= 0.6676\ 3 \times 0.14214 + (1 - 0.6676) \times 0.85786$$
$$= 0.5698$$
$$P(IO|D) = P(S|D)\ P(I) + P(\neg S|D)\ P\ (\neg I)$$
$$= 0.6676\ 3 \times 0.14214 + (1 - 0.6676) \times 0.85786$$
$$= 0.5698$$

Now in the predictive reasoning evidence is set on P(D) == non-reliable and nodes S, I and IO change to different values. P(D) ==1 and the sensor CPT values for non-reliable.

$$P(S|D) = P(D)P(\neg M) *S + P(D)P(M)*S$$
$$= 1*.5 \times 0.8874 + 1*.5* 0.8124$$
$$= 0.8499$$

$$P(I|D) = P(S|D)\ P(I) + P(\neg S|D)\ P\ (\neg I)$$
$$= 0.8499 \times 0.99686 + (1 - 0.8499) \times 0.14214$$
$$= 0.8686$$

$$P(IO|D) = P(S|D)\ P(I) + P(\neg S|D)\ P\ (\neg I)$$
$$= 0.8499 \times 0.99686 + (1 - 0.8499) \times 0.14214$$
$$= 0.8686$$

In the intercausal reasoning (Explaining away) cases, reasoning is having exactly two possible causes of a particular effect. In Figure 5-8 (c) evidence is set on the blue nodes while querying the yellow node. As an example of the intercausal reasoning, the manufacturer (building the sensors) is independent of the distributor (selling the sensor). For example, a user that has purchased the sensor from a distributor does not change the probability of the sensor being built by the manufacturer. However, if we know that the user purchased a sensor then the chances that the sensor came from a specific manufacturer and distributor are increased. In the case of intercausal

reasoning using Figure 5-7, we set evidence on both distributor and sensor to be 1 on non-reputable and not reliable, this changed the manufacturer from .5/.5 for non-reputable and reputable to .4779 and .5221, having very little change.

In calculating the equation for finding the probability of M given both S and D and setting both sensor and distributor to one. The sensor CPT table is used for the two values where distributor is not reputable. Since M and D are independent, the presence of one may make the other more or less likely, this is shown with the very little difference in M. This yields the following:

$$P(M|S\wedge D) = S*D*\neg CPT\_value + S*D*CDT\_value$$
$$= 1*.09436*.88748 + 1*.09436* .81249$$
$$= 0.1604$$

In our final type of reasoning, this brings diagnostic, predictive and intercausal together called combined. Since, in BNs any node can be queried and nodes may have evidence set, this allows any combination to be achieved. In Figure 5-8 (d), evidence is set on the two blue nodes and the yellow node is queried. To illustrate this combined case, the evidence is set on both distributor and intermittently operating to be 1 non-reputable and true. When the sensor is queried, this changed to being .9755 not reliable. The use of BNs and applying these reasoning strategies allows a more insightful analysis of the domain space of interest.

At this point we have a notion of nodes and arcs and what they mean, we have a notion of probabilities, and we have a notion of reasoning. The next step is a methodology for constructing the network. The following is an algorithm for constructing a Bayesian Network from Pearl, 1988[171]:

1) Choose a set of relevant variables $\{X_d\}$ that describe the problem.

2) Choose an order for the variables [X_1, X_2, ..., X_D]
3) For each variables X_d from d=1 to D:
    a) Create a Node for X_d.
  b) determine the minimal set of previous nodes from 1 to d-1 on which X_d depends.
  These are the Parents of X_d : Parents(X_d).
  $P(X_d \mid X_{d1},..., X_{dm}) = P(X_d \mid \text{Parents}(X_d))$
  Such that $\{X_{d1},..., X_{dm}\} \subseteq \{X_1, ..., X_{d-1}\}$
  c) Define the Conditional Probability Table (CPT) for X_d

We leverage this algorithm below when we discuss the models in section IV.

The following examples provide evidence that using a Bayesian Network (BN) based technique for assessment, risk management, evaluation and analysis of threats is a reasonable approach. Each example presents a different twist on how data is being generated for the conditional probability tables needed to construct a BN. All utilize BNs for quantifying uncertainty and providing evidence-based reasoning in a variety of different application areas. We briefly describe what these examples are by the following topics, why BNs, how were the values created, an overview of how it works and what is the crux of the example.

Prabhakaran, et al., present an example of a BN for sensor safety assessment used in Unmanned Aerial Vehicles (UAV).

What is it – Prabhakaran, et al. presents a safety assessment for Unmanned Aerial Vehicles (UAV) where the sensors are analyzed for faults or failures, since critical sensors affect the total functionality of the system and failures can make the system unsafe [172]. Typically, safety assessment for aerial vehicles uses Fault-tree analysis or Markov models, however in this case the authors used BayesiaLab's software to construct a Bayesian Network model that reasons about the analysis and sensor performance (output data to ensure operational functionality).

Why use BN- The authors believe that using a BN in a safety assessment has its advantages, for example using both priori and posterior information, deals with incomplete data sets, easy to find the causal relationships between the data and is highly efficient in a reasoning algorithm that is mature.

How were the values created- Sensor data was generated using MATLAB/Simulink for modeling the ground speed Vg from input values including yaw angle, airspeed, windspeed, and wind direction. A set of constraints were added to distinguish between safe and unsafe values. The output from the MATLAB model created a database with 160 samples, from which some data is used for learning and the other for test.

Overview – The BayesiaLab tool provides two paths to generating a BN, the first being a Machine Learning (ML) approach where data is fed into the tool to create the model and the other being a manual process where experts create the model. The author's tried both approaches, where the ML approach left data nodes unattached in the model because the data was uncorrelated to the defined target node. The other method is the knowledge-based approach where an expert defines the nodes and the BN structure. A more detailed sensor BN was created manually with expert knowledge where the target nodes called safe, unsafe, and partially safe were linked to six sensors, gyro, ground speed, angle of attack, differential pressure, static pressure, and heading angle. The BN model was used to reason about the safety of UAV for ground speed.

What is the crux – The authors have implemented a BN for System Safety Analysis (SSA), where the research goal was to monitor the safety critical sensor outputs and ensure successful UAV sensor performance. The BN simulation model provided successful results, which the authors suggest can be expanded into a larger

model for all the sensors in the UAV for real-time safety checks (using the ML approach with defined target nodes). As the authors have stated, using BN lends itself to modeling both hardware and software. We agree with that observation, since we are considering using it for the holistic security architecture that includes these layers.

Ramakrishnan presents a BN example for security risk management looking at insider and external cyber-attacks.

What is it – Ramakrishnan, applies Bayesian Networks to further expose the complexities of risk factor interdependencies, providing a quantification of risks and decision support for risk management. He examines some cases related to the risk of data loss from the perspectives of insider and external cyber-attacks.

Why use BN- Ramakrishnan states that BNs can capture the complex interdependencies among risk factors and can effectively combine data with expert judgment. BNs can provide rigorous risk quantification and true decision support for risk management.

Overview- The major groupings of the BN's nodes were system security controls, phishing controls, and network controls for the data leakage BN. Three different BNs were generated for best, moderate, and worst-case scenarios for data leakage related to control effectiveness. Another three set of scenarios (best, moderate and worst) were presented that added nodes, vulnerability, controls, web server compromise, web server access, logging infrastructure and attackers' action. Table 5-1 shows the data used in the BN for the different scenarios. When the security operation center (SOC) team's skills and responsiveness were set to high, the loss amount went down from $33 M to 7M as the best case, in the moderate case (11.5M) and in the worst case a maximum loss of 33M was concluded. The BN provided the casual

interdependencies and relationships that helped define the risk factors being created from different perspectives [173].

TABLE 5-1: DATA FOR SECURITY RISK ASSESSMENT

Scenario	Control Effectiveness	Probability of Data Leakage (approximate	Risk Rating	SOC Team Response Effectiveness	Loss Amount (in Millions of US Dollars)
Best	• Very effective controls with a few extremely effective controls • Data loss prevention (DLP) in block mode • Employee awareness high	85 percent unlikely. 13 percent extremely unlikely. Less than 1.5 percent likely	Low	High	7.5
Moderate	• Moderately effective controls with a few not effective controls • DLP in monitor mode • Employee awareness medium	91 percent likely. 7.9 percent very likely. Less than 1.4 percent unlikely	Medium	Medium	11.5
Worst	• Not effective controls with a few controls slightly effective • No DLP • Employee awareness low	68 percent very likely. 30 percent extremely likely. Less than 1.5 percent likely	High	Low	33.5

What is the crux – The BN approach helps to identify, understand, and quantify complex interrelationships and can help make sense of how risk factors emerge and are connected, and how to represent control and mitigate them. This work parallels our

thinking that as the defenses increase the risk/damage should also decrease. This all depends on the skills/resources that are available, for example on an autonomous robotic system.

Atoum and Otoom presents a BN example for cyber security risk assessment and identifying risk reduction for financial investments.

What is it- Atoum and Otoom create a Bayesian Network called the Holistic Cyber Security Implementation Framework (HCS-IF) where their proposed model (13 nodes) analyzes and quantifies information security risks caused by several threats. The term holistic is used in an abstract manner to articulate that security encompasses at the national level. This cyber security framework helps security managers identify potential risks as well as where investment dollars should be allocated to reduce risks. In the HCS-IF, the supportive evidence values toward cyber security objectives are mainly the controls, the strategic moves, the requirements, identified goals and the Cyber Security Strategy (CSS).

Why use BN- BN was used to formally validate the ability of the Holistic Cyber Security Implementation Framework of [3] (H-CS-IF) to achieve the required security level utilizing a set of controls that have an effect on each other. In the HCS-IF, the supportive evidence values toward cyber security objectives are mainly: The Controls, the Strategic Moves, the Requirements, Identified Goals, and the CSS. Unfortunately, to our knowledge, there is no direct way to calculate the probability of each component. Therefore, we depend on domain knowledge and expert expectation [174].

Overview- A thirteen node BN had the following labeled nodes Cyber Security Strategy, Audit, Business, Requirements, Controls, Framework Controls, Goals, Strategic Moves, Governance, Prioritization, Change Control Board, Strategic Controls

and at center was the Strategic Objective. By taking the current CSS as the prior, one could manipulate a set of transform nodes (all remaining nodes above) to achieve an outcome result of a security level. The evidence was set on several nodes to show that the target node identified as the security objective was able to be influenced by the other linked nodes. In the first scenario, the assigned metrics on network nodes were set by experts, for example the CSS node had values set to holistic at .91 and non-holistic set to .09 where the security objective node had a .88 security level [174]. In the second scenario, the evidence was set on Business, Framework, Governance, Audit, and Strategic Controls not being achieved, this set the security objective to .28 level. The BN showed that the transform nodes can influence the target node and as a result provide an outcome of a new security level. The proposed BN is able to give direction to the security managers in the early stage of the cyber security implementation, at design time

What is the crux – The results showed that the proposed model worked and could be used by security managers for guidance. The BN model was used as an assessment tool to help guide the budget and resources to where the threat level was greatest. As knowledge is obtained about the security effects on autonomous robotic systems, the design phase budget and resource can be shifted to the appropriate layers of the architecture.

De Wilde presents a BN example for health care data breach security risk assessment.

What is it- De Wilde presents her research into creating a Bayesian network that predicts the probability of a data breach caused by a group of insiders in a health care organization given certain prior indicators and preventive measures and test its usefulness in practice [175]. The indicators will be related to malicious and accidental

insider threats and focus on the motivation, capability, and opportunity of a group of insiders. A malicious insider attack can be characterized by the motivation and capability of the attacker and the opportunity to perform the attack. In general insiders do not have a reason to make mistakes and therefore the accidental insider threat can be characterized by the (lack of) capability and the opportunity to perform the attack. These elements can be observed before a data breach occurs and therefore are called "prior indicators" of a data breach. Each element can be divided into specific prior indicators related to the insider threat. This model can also be used to determine which measures should be taken to minimize the probability of a data breach.

Why use BN- In the context of security and privacy however, there is limited information available on how BNs can be created and used in practice. This research contributes to this by developing a model that combines observed prior indicators of a data breach and measures taken by an organization to predict the probability of a data breach in a health care organization as a kind of risk assessment. The model combines both malicious and accidental insider threats posed by a group of insiders. When changing the observations, the probabilities for different scenarios can be determined. In this way the best combination of measures to minimize the probability of a data breach given certain prior indicators can be identified.

How were the values created- De Wilde created her own values based on literature that covered insider threats.

Overview- Four different types of BN were looked at to support the data breach model structure, these were BN, Multi-Entity Bayesian Network (this combines First-Order Logic with BNs for representing and reasoning about uncertainty with domain rich content), Dynamic Bayesian Network and a BN based on attack graphs. The

decision was made that the BN was the most useful one, because it provided the ability to make data breach predictions with limited CPT data. It was also decided that discrete nodes would be used, since it is not clear how continuous nodes can be used effectively in BNs and the tool AgenaRisk is mostly designed for discrete values. The basic model is built by having two parent nodes called measures that point to prior indicator nodes and a child node called data breach, which had a child node called posterior indicators. Three iterations of a Bayesian network model are described called alpha (mobile device case), beta, and gamma where the main idea is to detect insider data breaches. The nodes in the BN are simple values of true, false, a coarse ranking (low, medium, or high), negative, neutral, positive and employee or employer owned device (none, personal, personal + staff) [175].

In the Alpha mobile device case, the model has two use cases. The first being to determine the probability of a data breach caused by a group of insiders who lose employee or employer-owned mobile devices or misuse the mobile devices. The second use case is to help health care organizations determine which additional measures they should take to protect themselves against data breaches caused by insiders. Using the basic model, the data breach node has two parent nodes, Mobile device misuse and Mobile device loss since these are the threats in the mobile device case and these will have Low, Medium, and High states. In the Alpha model additional nodes were added to the base model nodes measures (Policy protection level, Accident protection level) and prior indicators (motivation, job type, attacker opportunity) to name a few.

In the Beta case model, the Alpha model is updated and validated where Misuse is further defined as not returning the devices when needed and/or copying personal data to private mobile devices. The Beta model can, similar to the Alpha model, can be

used to determine the data breach probability for one specific organization. To validate the model a focus group was surveyed as well and an interview with a security officer was conducted. From the data collected the Alpha model was modified by removing the competitive advantage node, renaming six nodes, and changing CPT values. The Alpha model was renamed to Beta to reflect the changes.

In the Gamma model, the Beta model was further validated for its usefulness and effectiveness by using it in practice. Three different data protection employees in different Dutch hospitals were interviewed and the data breach assessment was performed with them. As a result of the interviews, changes were made to the model. Specifically, the Mobile device misuse and Mobile device loss changed to Boolean values (other changes included name change and nodes changes in measures).

A general case model is shown in Figure 5-9 that can be extended to other use cases.

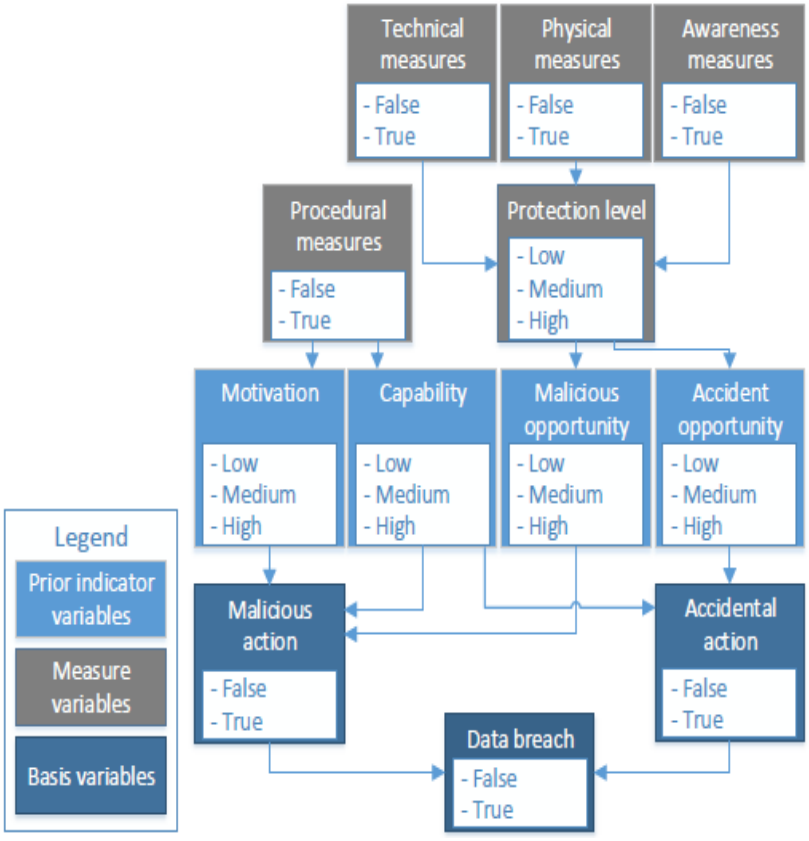


Figure 5-9: General model

What is the crux – Bayesian networks can be used to predict the probability of a data breach, but also to detect a data breach. The research focuses on prediction, but also demonstrates how the BN can be extended to include the possibility for detection. De Wilde has stated that the values in the CPT should be obtained from experts to make the models more meaningful. We agree with this statement that obtaining CPT values from experts provides a more meaningful and realistic BN. We have sourced our values from credible source standards and/or industry known.

Henshel, et al., presents a BN example for cyber security risk assessment that includes the human factor as an attacker/defender.

What is it- Henshel, et al., create a MulRAP framework that is a universal approach for parameterizing complex systems, facilitating the detailed characterization of the network structure and explications of the relationships between the nodes (components or assets) and edges (processes) [102]. The parameterization process can be applied to any high-level concept (e.g., risk, vulnerability, resilience) quantitatively after defining, quantifying, and validating the relationships and inherent network properties. The goal of the parameterization process, as should be of any modeling endeavor, is to identify the minimum number of necessary and sufficient information-rich variables in order to accurately describe the emergent properties of a complex system.

Why use BN- Bayesian Networks provide a solution for studying classical cause-and-effect relationships. Since it combines graphical analysis with Bayesian analysis, it provides a more intuitive way to represent causal relationships. Bayesian networks are widely used to construct risk models to solve complex risk assessment problems.

How were the values created- The CPTs were populated by experts in risk assessment and cybersecurity. The experts were made up of 30 Cyber Security Collaborative Research Alliance (CSecCRA) researchers that participated over a year in consultation.

Overview- A BN is created incorporating variables representing critical risk-inducing and risk-mitigating human and cultural factors into a proof of concept. The introduction of human factor parameters is incorporated into the model by two sets of skills, attacker and defender and integrates a country-based modifying threat factor. The use of empirical data from the studied SQL injection attacks was incorporated into the

model. In comparison to using theoretical assumptions, the empirical evidence provided a view into the effects of the network on the overall risk. The model shows that by including human factors this contributed to altering risk in cyber networks, detailing the potential impacts and effects of human actors on risk posture, strategy, and response. The model captures the SQL injection attack on a highly sensitive database server and an evaluation is performed on the risks parametrization and validation of the empirical data obtained in the experiment on a virtual network testbed.

What is the crux- An advantage of modeling cybersecurity risk using Bayesian networks instead of statistical techniques is that they allow identifying the factors which contribute the most towards detecting high risk situations. The authors used empirical data vs theorical, which provided a comprehensive knowledgeable view of how network changes affected the overall risk analysis. Like this example of using BNs, we are also using empirical data acquired from standards and other known sources for our model.

Herland, et al., presents a BN example for smartphone personal data security assessment.

What is it – Herland, et al., present a Bayesian Network that models security risks, consequences, impacts of Finland's smartphone users' personal data being compromised.

Why use BN- Herland, et al, looked at different techniques such as Markov chains and Petri nets but concluded that they are not suitable for causal relationship analysis, as they become large, complex and confusing [176]. BNs were found to be an effective method for documenting and analyzing causal knowledge of domain

experts. The model lends itself well to different types of sensitivity analysis, which would be especially useful when analyzing potential controls and mitigations for risks.

How were the values created- Their BN model uses a knowledge-based approach due to a lack of available data. First, information is reviewed from the literature in order to determine the known assets and risks related to smartphone use. Then a two-stage expert interview process was completed where the first stage gathered information to build a qualitative model and in the second stage, this model was validated for dependencies, impacts and quantitative values and correctness. Government, network, application, organizational security, telecom operator, and smart phone development experts were interviewed to populate the CPTs.

Overview- In the BN model each risk node is defined as (True or False) and each consequence as negligible, low, medium, or high. The model included the following nodes (Network spoofing attack, malware, phishing, sniffing on legitimate network, shoulder or eavesdropping, loss or theft of device, unauthorized physical device access, vendor backdoor, surveillance on network level, and unintentional data loss). These nodes, all focused on the consequence of leakage of personal or confidential data. The model results showed that malware, phishing, loss or theft of device and unintentional data disclosure were among the most common threats for leaked data. Using a BN was essential for domain experts to reason about information security risks for smartphones because it allowed different scenarios to be analyzed, and the model could be easily extended.

What is the crux- The outcome of the study was a BN model that demonstrated that the most important risks in Finland's smartphone personal data security included traditional security risks such as malware and phishing, general risks such as loss or

theft of device, and new risks such as unintentional data disclosure through legitimate applications. BNs were suitable for information security risk assessment because they were flexible in model construction that could be used for various kinds of analysis. The resulting model lends itself well to different types of sensitivity analysis, which is useful when examining risks potential controls and mitigations. We concur that using BNs provides a flexible method for asking questions about the security posture of the systems once the model has been setup with realistic data.

From these different examples, there is strong evidence that suggests BN's are effective in causal reasoning about risk assessments, analysis, and evaluation of security threats. In a number of examples, it was expressed that obtaining CPT values was difficult, since no data was available, while in others cases a simple boolean approach was taken. A number of examples also expressed that expert knowledge was obtained to define the values or construct the BN. Although these researchers have demonstrated the utility of BNs for assessing various aspects of the overall security of a system, we extend on the base approach of using BN for security by one that models the internal state of an autonomous robotic system with respect to its configuration. Another difference is that we utilize specific metrics from specification and standards that incorporate more meaningful information about the values that are associated with variables. Lastly, our approach includes vulnerabilities associated with system, hardware and software levels, supply chain vendors, AI robustness and a resilience set of elements. In our opinion, this model supports the construction of a holistic security model that can react to threats.

5.3 Methods to calculate the metrics

To construct this holistic security model that incorporates the elements discussed above, we utilize a BN that uses empirical data for its CPT values. In order to construct a BN, we need the values for the CPT of each node. To acquire these values, we use our previous research results where we identified sources for assessing trust at the system, hardware, software, AI robustness, and supply chain levels. We take those values and formulate our own metrics, that include the impact, cost of damage and perceived target cost for each layer of the system. These metrics will become the basis for the CPT of each node in the BN.

System trust metric calculation:

From our previous research work in Chapter 4 we defined a generic equation for calculating the trust metric for each layer, that was defined by the following equation 5-3:

$$\forall x\ \forall y\ \forall z\ \forall \alpha \text{ at } (x, y, z, \alpha\ TM_n) \Rightarrow L(x_n) \wedge R(y_n) \wedge D(z_n) \wedge A(\alpha_n) \qquad (5\text{-}3)$$

$$\begin{cases} 0 \le x \le 1 \\ 0 \le y \le 1 \\ 0 \le z \le 1 \\ 0 \le \alpha \le 1 \end{cases}$$

trust metric = *level value* $_n$ * *probability of adversary exploit reward* $_n$ * *probability of adversary exploit damage* $_n$ * *likelihood of adversary taking action* $_n$

Where the level value can be constructed from a number of parameters specific to that layer and n defines the levels associated with each of the parameters. For example, a level value can be high, adversary exploit reward can be high, the adversary

exploit damage can be high, but the adversary taking action to exploit maybe low. The trust metric parameters are independent of each other, but for simplicity in the tables below the parameters will be aligned with the associated levels. This means that when the level value is high, each of the remaining parameters will be high. The likelihood for an adversary taking action to exploit will not be shown in the calculations below, since this is being kept at a probability of 1 for the rest of the discussion. The probability of 1 represents that the adversary will also try to exploit.

We first define the equation for our system metric as shown in equation 5-4. Where E_n is the common criteria evaluation assurance level (EAL), C_n is the cost for development to achieve that level, T_n is the time it takes to achieve that level, CDP is the Collateral Damage Potential that an adversary can cause from an exploit, PVT is the Perceived Target Value is the reward that an adversary can gain from the exploit and L is the likelihood that an adversary will exploit. The cost and time variables are controlled by the assurance level being targeted, the deliverables to meet the requirements, the gates for verification/validation, independent 3rd party lab for validation and by the certifying entity.

$$SysScore(n) = \quad (E_n * C_n * T_n * CDP_n * PTV_n * L_n) \qquad (5\text{-}4)$$

At the system level of trust metrics, we will use the first five levels of common criteria specification since it provides a basis for assigning values that is widely known and used to evaluate system security trust. CC also correspond to FIPS and safety specifications as shown in Table 5-2.

TABLE 5-2: MAPPING SAFETY AND SECURITY SPECIFICATIONS

Domain	Domain Specific Assurance Levels						
Automotive (ISO 26262)	QM	ASIL-A	ASIL-B	ASIL-C	ASIL-D	ASIL-+	
General (IEC-61508)	-	SIL-1	SIL-2	SIL-3	SIL-4		
Aviation (DO-178/254)	DAL-E	DAL-D	DAL-C	DAL-B	DAL-A		
Railway (CENELEC 50126/128/129)	-	SIL1	SIL2	SIL3	SIL4		
FIPS 140-3	L1	L2	L3	→	L4	→	→
CC (ISO 15408)	EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7

In calculating the system trust metric, the cost to certify and time to certify will be multiplied into each of the levels, this will differentiate each level. Dale, presented a briefing in which the cost and time to assess different EALs was collected from >250 evaluations completed or in the evaluation process among 14 different laboratories [104]. As estimates for different EALs, Dale determined these parameters.

EAL 1 = 0
EAL 2 = cost $100 to $170k and 4 to 6 months
EAL 3= $130 to $225K and 6 to 9 months
EAL 4 = $175 to $750k and 7 to 24 months
EAL 5 = $750 to $2M and 24 to 48 months

The values for the Collateral Damage Potential are obtained from both the Common Configuration Scoring System and the Common Misuse Scoring Specification [120] [121] as shown in Table 5-3.

TABLE 5-3: COLLATERAL DAMAGE POTENTIAL VALUES

Metric	Value	Description
None	1	There is no potential for loss of life, physical assets, productivity, or revenue.
Low	1.25	Successful exploitation of this vulnerability may result in slight physical or property damage or loss. Or there may be a slight loss of revenue or productivity.

Low-Med	1.5	Successful exploitation of this vulnerability may result in moderate physical or property damage or loss. Or there may be a moderate loss of revenue or productivity.
Med-High	1.75	Successful exploitation of this vulnerability may result in significant physical or property damage or loss. Or there may be a significant loss of revenue or productivity.
High	2	Successful exploitation of this vulnerability may result in catastrophic physical or property damage or loss. Or there may be a catastrophic loss of revenue or productivity.

The values for the Perceived Target Value are obtained from both the Common Configuration Scoring System and the Common Misuse Scoring Specification [120] [121] as shown in **Table 5-4**.

TABLE 5-4: PERCEIVED TARGET VALUE

Metric	Value	Description
Low	.8	The targets in this environment are perceived as low value by attackers. Attackers have low motivation to attack the target system relative to other systems with the same vulnerability.
Med	1	The targets in this environment are perceived as medium value by attackers. Attackers are equally motivated to attack the target system and other systems with the same vulnerability.
High	1.2	The targets in this environment are perceived as high value by attackers. Attackers are highly motivated to attack the target system relative to other systems with the same vulnerability.

We adjust Table 5-4 to have five values instead of three by continuing the sequence as shown in Table 5-5. This is to discretize each element that goes into the metric into the same number of levels to have a 1-to-1 relationship with the other values.

TABLE 5-5: ADJUSTED PERCEIVED TARGET VALUES

Metric	Value	Description
None	.4	The targets in this environment are perceived as no value by attackers. Attackers are not motivated to attack the target system.
Low	.6	The targets in this environment are perceived as low value by attackers. Attackers have low motivation to attack the target system relative to other systems with the same vulnerability.
Low-Med	.8	The targets in this environment are perceived as low to medium value by attackers. Attackers have low to medium motivation to attack the target system relative to other systems with the same vulnerability.
Med-High	1	The targets in this environment are perceived as medium to high value by attackers. Attackers are equally motivated to attack the target system and other systems with the same vulnerability.

High	1.2	The targets in this environment are perceived as high value by attackers. Attackers are highly motivated to attack the target system relative to other systems with the same vulnerability.

Table 5-6 is the likelihood of the adversary taking action to exploit. The likelihood is a function that a threat exists and that the threat can successfully exploit the component or system. The likelihood values are aligned with **Table 5-3** and **Table 5-4**.

TABLE 5-6: THE LIKELIHOOD OF AN ADVERSARY TAKING ACTION

Metric	Value	Description
None	.2	
Low	.4	The likelihood of the adversary taking action to exploit is low
Low-Med	.6	
Med-High	.8	
High	1	The likelihood of the adversary taking action to exploit is high

In order for our data to fall within a probability range of [0,1] a couple of techniques can be used to manipulate the data to ensure it is normalized into this range. These common techniques are the min-max or z score for data normalization.

The equations for min-max, where v' is the new value of each entry in the data, v is the old value of each entry in the data, *newmax* and *newmin* are the boundary values within the data and *max* and *min* are the values of *a* respectively. This is used to scale the values in an array, for example from [0,1] range.

$$v' = \frac{v - min_a}{max_a - min_a}(\text{newmax}_\text{a} - \text{newmin}_\text{a}) + \text{newmin}_\text{a} \quad (5\text{-}5)$$

The method to normalize values is called z-score where x is the random value with a mean of μ and the standard deviation σ. If the z-score equals zero, it is on the

mean and if it equals 1, it is 1 standard deviation above the mean. This is used during the execution of the MATLAB code below to normalize the values.

$$z = \frac{(x - \mu)}{\sigma} \qquad (5\text{-}6)$$

By simply plugging the values from the tables into eq. 3 will generally not result in a range of values between 0 and 1. Therefore, to obtain a system score it is necessary to normalize and interpret the results of eq. 3 to obtain a system score

Calculating these values EAL *Cost (Upper and Lower range) * Time (Upper and Lower range) * Collateral Damage Potential * Perceived Target Value into a range that has been normalized interpolated using MATLAB code shown below:

```
% How to calculate the system score using cc*cost*tm*C*P values
cc = [.2, .2, .2, .2, .2]; %set values
costh = [2e6, 750e3, 225e3, 170e3, 0];
costl = [750e3, 175e3, 130e3, 100e3, 0];
timeh = [48, 24, 9, 6, 0];
timel = [24, 7, 6, 4, 0];
c = [2, 1.75, 1.5, 1.25, 1];
p = [1.2, 1, .8, .6, .4];
 syt = cc.*cost[h,l].*time[h,l].*c.*p;
display(syt)
nsyt = normalize(syt,'range');
display(nsyt)
xq = 0:.20:1;
vq = interp1(nsyt,xq,'pchip');
vq1 = normalize(vq,'range');
display (vq1)
```

To illustrate the need to perform the normalization and interpretation as shown in Figure 5-10, the output from the code above is shown for nsyt, in blue as first normalization of the calculation. The line vq is the interpreted operation and vq1 is the normalized as shown in yellow that falls between [0,1].

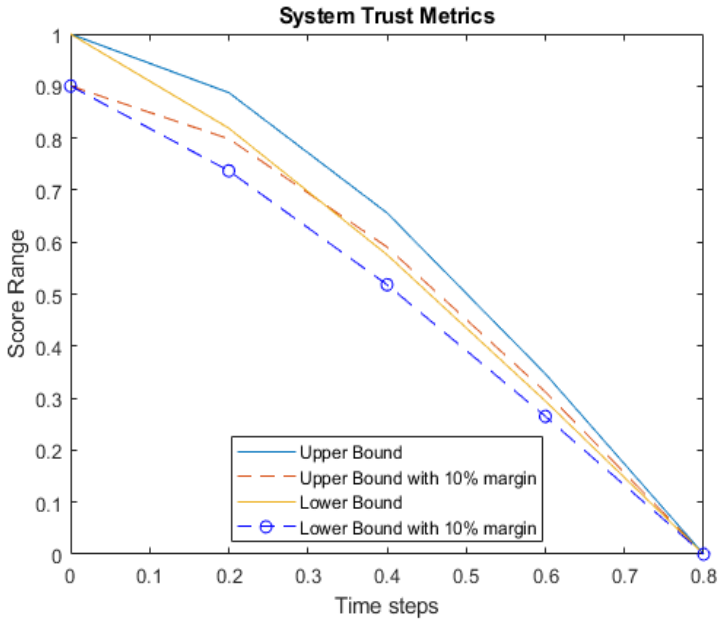


Figure 5-10: Outputs from calculations

To continue, the high and low values from the tables above are shown as the upper and lower bound (blue and yellow solid lines) in Figure 5-11. A 10% delta is taken from each respectively and those are the dash lines. The ten percent is to accommodate for the uncertainty.

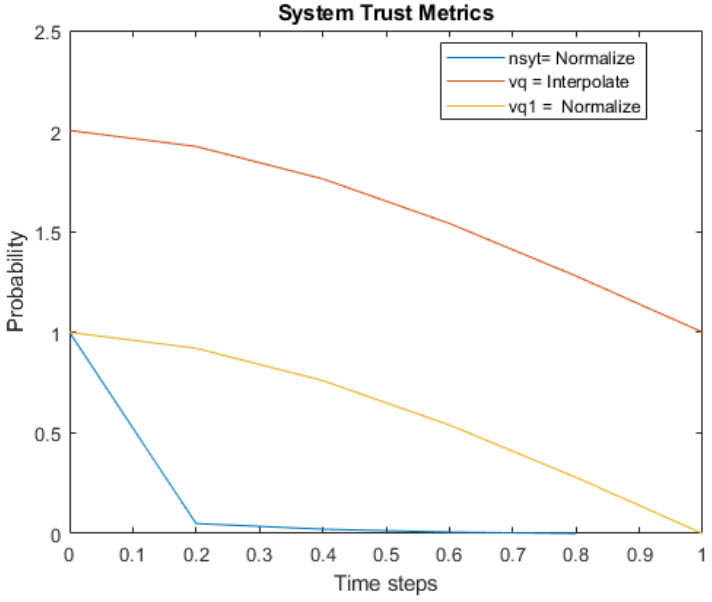


Figure 5-11: Upper and Lower Bounds with tolerances

The calculation methodology discussed above is applied to the system, hardware, software, and supply chain metrics.

The resultant definition for the system level metric is shown in **Table 5-7**.

TABLE 5-7: NEW SYSTEM METRICS

Metric	Value	Description
Sys1	0 to .40	This is the lowest trust level, damage, reward to exploit, with no cost and time
Sys2	.41 to .60	
Sys3	.61 to .76	
Sys4	.77 to .89	
Sys5 +	.9 to .1	This is the highest trust level, damage, reward, with highest cost and duration to achieve.

Hardware trust metric calculation:

We first define the equation for our hardware metric as shown in equation 5-7. Where TM_n is the hardware design trust metric, CDP is the Collateral Damage Potential that an adversary can cause from an exploit and PTV is the Perceived Target Value is the reward that an adversary can gain from the exploit.

$$HWScore(n) = (TM_n * CDP_n * PTV_n * L_n) \qquad (5\text{-}7)$$

Kimura presents the development of Trust Metrics for quantifying design integrity. The Design Integrity (DI) trust metric accounts for the signal activity rate, logical equivalence, power consumption, functional correctness, and structural analysis. TM = DI * R, where DI is the Design Integrity and R is the Reference Design Quality (what artifacts are available from the design) [118] as shown in **Table 5-8**.

TABLE 5-8: HARDWARE DESIGN TRUST METRICS

Metric	Value	Description
None	0 to .15	Where this is the lowest level of trust
Low	.16 to .33	
Low-Med	.34 to .51	
Med	.52 to .70	
Med-High	.71 to .89	
High	.9 to 1	Where this is the highest level of trust

We extend both Collateral Damage Potential and Perceived Target Value to one more placement in the sequence in order to align with Table 5-8 values. Using equation 5-7 and the procedure outline above, we can create a hardware score by combining the

hardware variables TM (Upper and Lower range) * Collateral Damage Potential * Perceived Target Value. The normalized result is shown in Table 5-9.

TABLE 5-9: HARDWARE METRICS

Metric	Value	Description
None	0 to .15	Where this is the lowest trust level, damage, and reward to exploit
Low	.16 to .33	
Low-Med	.34 to .51	
Med	.52 to .70	
Med-High	.71 to .89	
High	.9 to 1	Where this is the highest trust level, damage, and reward to exploit

Software trust metric calculation:

We first define the equation for our software metric as shown in equation 5-8. Where TI_n is the Technical Impact from a potential software vulnerability, CDP is the Collateral Damage Potential that an adversary can cause from an exploit and PTV, the Perceived Target Value is the reward that an adversary can gain from the exploit.

$$SWScore(n) = (TI_n * CDP_n * PTV_n * L_n) \qquad (5\text{-}8)$$

We selected the technical impact score from the Common Weakness Scoring System (CWSS) specification, since this is more detailed than confidential, integrity and availability as in the other specifications [122]. The technical impact covers the potential result of an exposure, whereas in the other specifications a vulnerability must be known, this is shown in Table 5-10.

TABLE 5-10: CWSS TECHNICAL IMPACT METRICS

Metric	Value	Description
None	0	Where low sensitive data is exposed
Low	.3	Where low sensitive data is exposed
Med	.6	Where medium sensitive data is exposed
High	.9	Where highly sensitive data is exposed
Critical	1	Where tasks cannot be performed

Using equation 5-8 and the procedure outline above, we can create a software score by combining the software variables TI * Collateral Damage Potential * Perceived Target Value. The normalized result is shown in **Table 5-11**.

TABLE 5-11: SOFTWARE METRICS

Metric	Value	Description
None	0 to .35	Where this will indicate no impact, no damage, and no reward for exploit.
Low	.36 to .53	Where this will indicate low value to impact, damage, and reward for exploit
Low-Med	.54 to .71	Where this will indicate low value to impact, damage, and reward for exploit
Med-High	.72 to .89	Where this will indicate high medium value to impact, damage, and reward for exploit
High	.9 to 1	Where this will indicate high value to impact, damage, and reward for exploit.

Supplier trust metric calculation:

We first define the equation for our supplier metric as shown in equation 5-9. Where Sup_n is the supplier trust metric, CDP is the Collateral Damage Potential that an adversary can cause from an exploit and PTV is the Perceived Target Value is the reward that an adversary can gain from the exploit.

$$SupScore(n) = (Sup_n * CDP_n * PTV_n * L_n) \qquad (5\text{-}9)$$

Northrop Grumman (NG), presents its supplier score card [154] with defined metrics related to a framework called Supplier Assessment Management System

(SAMS). Within the same document a Systems Applications and Products (SAP) supplier score card is found. For the supplier trust metrics, we will use the SAP range and the NG evaluation categories, as shown in Table 5-12.

TABLE 5-12: SUPPLIER TRUST METRICS

Metric	Value	Description
Unsatisfactory	0 to 50	
Marginal	51 to 74	
Satisfactory	75 to 90	
Excellent	91 to 100	Where this is the highest level of trust

Using equation 5-9 and the procedure outline above, we can create a supplier score by combining the supplier variables Sup * Collateral Damage Potential * Perceived Target Value. The normalized result is shown in **Table 5-13**.

TABLE 5-13: SUPPLIER METRICS

Metric	Value	Description
Unsatisfactory	0 to .51	Where this will indicate unsatisfactory rating level of trust, damage, and reward to exploit.
Marginal	.52 to .70	Where this will indicate the minimal level of trust, damage, and reward to exploit.
Satisfactory	.71 to .89	Where this will indicate the next highest level of trust, damage, and reward to exploit.
Excellent	.9 to 1	Where this will indicate the highest level of trust, damage, and reward to exploit.

AI robustness trust metric calculation:

As presented in our previous survey research work in Chapter 4, AI introduces different types of attack vectors. The focus is on AI adversarial attacks, the classification of data being poisoned, evasion attacks and black box attacks to name a few. For every attack, a remedy may arise to counter, but this takes time and there needs to be a method to identify these types of attacks. In the case of autonomous mobile robots, an AI/learning layer makes a system susceptible to new types of attack strategies

which more conventional attacks may not consider. This leads to a different approach that was taken with the AI robustness calculations since the approaches for evaluating AI robustness are very different than those used at the system, hardware, software, and supply chain metrics.

An adversarial example is when x is recognized and classified as the original as target t = arg-max F(x) and a new desired target where t′ not equal to t, this is called x′ a targeted adversarial example if arg-max F(x′) = t′ and x′ is close to x given a distance metric [134]. The minimum distance of a misclassified nearby adversarial example to x is the minimum adversarial distortion required to alter the target model's prediction, which is referred to as the lower bound. A certified boundary guarantees the region around x that the classifier decision cannot be influenced from all types of perturbations in that region. In other words, the robustness is being able to detect perturbation as close to *x* as possible and, in some cases, this is an approximation or an exact guarantee to determining the lower boundary point. In order to evaluate the distance, sometimes called distortion or error, between x' and x, the generalized Minkowski's formula is used to calculate the distance metric within p-norm space. The generalized Minkowski's formula is shown in equation 5-10.

We select Minkowski's formula to calculate the distance metric for p-norm when p=1, is a Manhattan distance, when p=2 it is a Euclidean distance, and when p=∞ it is a Chebyshev distance. In equation 5-10, the distance formula represents a generalized approach for distance measurements.

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}} \qquad (5\text{-}10)$$

By using these lower bound techniques (CLEVER and CNN-Cert), a minimum distortion level is established and from this point we can define ranges for rating AI implementations against these known values. To better illustrate this concept the following Figure 5-12 shows a center region equal to the certified region and each subsequent ring are correlated to the rating or strength of the AI implementation using a distance function. Let x be the certified region and y be the AI implementation, we can use the p-norm distance equation to determine the differences for adversarial perturbation detection.

As one moves further away from the certified region the rating should decrease with 1 being the lowest.

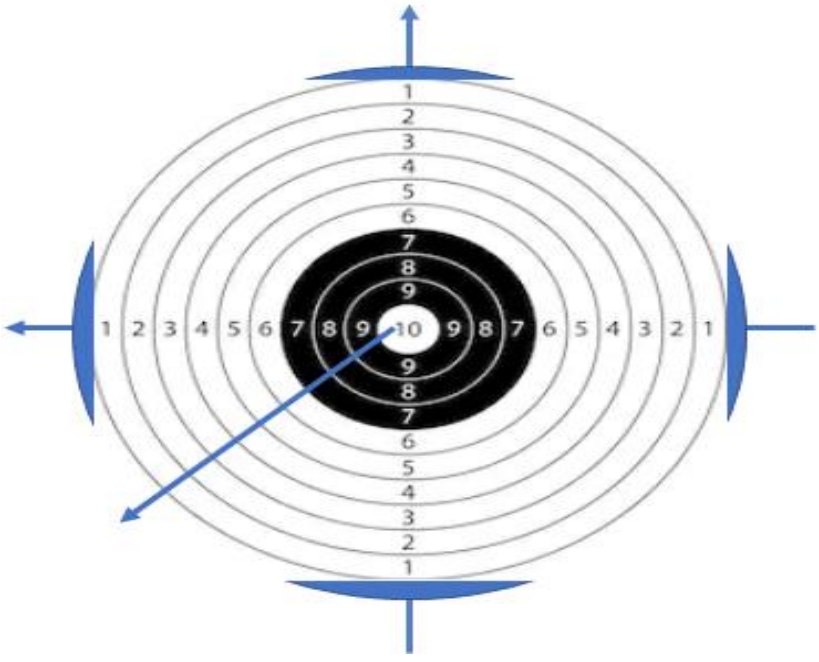


Figure 5-12: Robustness distance metric

We first define the equation for our AI Robustness metric as shown in equation 5-11. Where D_n is the distance from the lower bound (certified area) of detecting an adversarial attack. As the distance is closer to the certified area it becomes more

difficult to detect, therefore, resulting in higher risk. The supplier must provide the testing results where these distance values can be obtained from or provide the testing logic, so that others can validate these values. CDP is the Collateral Damage Potential that an adversary can cause from an exploit and PTV, the Perceived Target Value is the reward that an adversary can gain from the exploit.

$$airScore(n) = (D_n * CDP_n * PTV_n * L_n) \qquad (5\text{-}11)$$

Using equation 5-11 and the MATLAB code below to simulate the distance function, we can create an AI robustness score by combining the AI variables D* Collateral Damage Potential * Perceived Target Value. The normalized result is shown in **Table 5-14**. The procedure below loops 5 times for each of the variable levels where distance is the err function of 1 – e. The variable e represents 3 digits over the range of 0 to 1 in .01 steps. The result of this procedure produces peaks for the 5 variables levels and those are used for the values in **Table 5-14**, this is shown in Figure 5-13.

```
E = .1e-3 :0.01:1;
err = 1-e;
c = [2, 1.75, 1.5, 1.25, 1];
p = [1.2, 1, .8, .6, .4];
cnt=0;
air = zeros(1,500);
for j=1:5
   for i= 1:100
      cnt = cnt+1;
      air(cnt) = err(i).*c(j).*p(j);
   end
end
```

```
nait = normalize(air,'range');
```

From plotting nait we obtain the nominal values for our metrics. The five peaks in the graph represent the distance from the lower bound but, in order to align with the rest of the metrics, we subtracted from 1 making the most robust closer to one. The five nominal values also align with CDP and PTV values in equation 5-11. The nominal

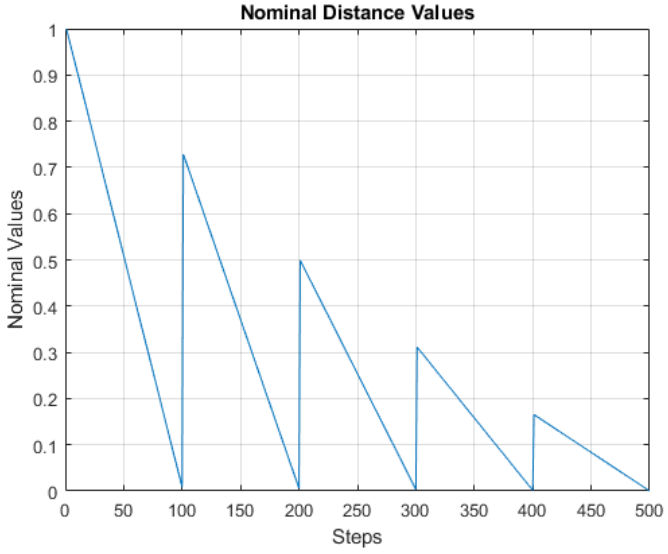


Figure 5-13: AI Robustness results

values are then multiplied by 10% (+/-) to give an upper and lower boundary at each level, so the new AI robustness metrics are shown in **Table 5-14**.

TABLE 5-14: AI ROBUSTNESS METRICS

Metric	Value	Description
None	0 to .30	Where this will indicate farthest away, no damage and no reward for exploit.
Low	.31 to .49	Where this will indicate low value for robustness, damage, and reward for exploit
Low-Med	.50 to .71	Where this will indicate low medium value for robustness, damage, and reward for exploit
Med-High	.72 to .89	Where this will indicate high medium value for robustness, damage, and reward for exploit
High	.9 to 1	Where this will indicate high value for robustness, damage, and reward for exploit

By using distance formulas one can determine a trust metric. Each defense technique has a distance/error from the certified area (boundary) where perturbations

can be detected, we can consider these values as trust metrics in a continuous set of ranges between [0,1]. Unlike some of the other measures we have seen so far that this is still an area that is less mature but apply the same technique to derive metrics.

5.4 Trust Models using Bayesian Networks

The need to use a Bayesian Network in the context of trust and causal inference, is part of reasoning [177]. By using causality, several questions can now be asked about the security posture of an autonomous robot system using BNs, but most importantly the robotic system can act on the knowledge it has from an internal point of view. Some questions to postulate against are: does having vendors that are more reliable than others decrease risk; do manufacturers that follow a security-aware development process reduce risk vs ones that do not; and if the platform supports a specific security configuration, can it be trusted to process an increased level of sensitive information? In the algorithm to create a BN, we have completed the quantitative portion by defining the metrics for our CPTs in the previous section and now we start to define the nodes of the BN is this section. We use a research platform for creating and analyzing the causality of the Bayesian Network.

The models and simulations are created in a product called BayesiaLab [178], where a supported workflow is shown in Figure 5-14. BayesiaLab is a graphical desktop application that can run on Apple, Windows and Linux platforms and it provides functions like supervised machine learning, unsupervised machine learning, knowledge modeling, observational inference, casual inference, diagnosis, analysis, simulation,

optimization, and visualization in 2D/3D formats. Formulas can be utilized as well as different probability distributions.

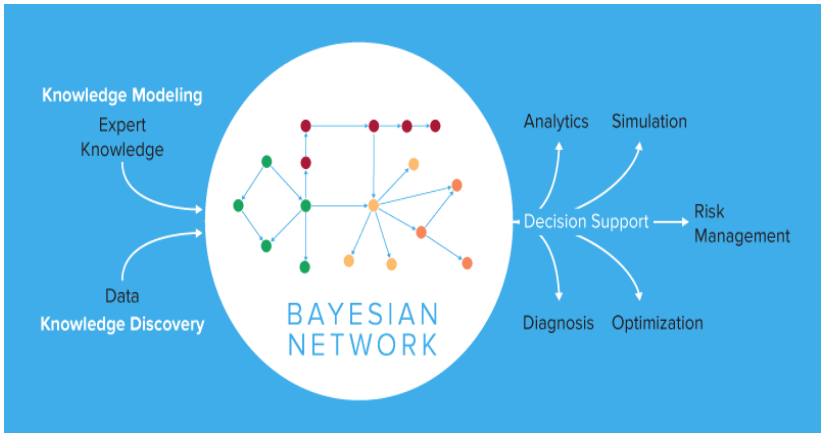


Figure 5-14 : BayesiaLab workflow process for Bayesian Network.

The methodology for assessing assurance involves applying an assurance score and a reputation score at each level of the system, so that the child layer has knowledge about its parent layer. The assurance score utilizes the trust metrics defined earlier, which includes the reward and damage values for each of the components that make up the system. The reputation score reflects a value to account for errors that may arise during the bootup of the system or during operational state execution. The collective assurance score is used by the system to assess external requests in order to properly fulfill it in a secure manner. This also reflects on the security posture of the system, meaning can the system support the request from a security point of view if the system is configured correctly to support the request and what are the potential risks. Utilizing the assurance and reputation scores goes deeper into the security model then the authentication/authorization controls that are used in conventional systems today.

The following BN example shown in Figure 5-15, Figure 5-16 and Figure 5-17 has two different microprocessors being used, where Figure 5-15 illustrates the initial

conditional distributions for the three nodes. In Figure 5-16 and Figure 5-17, the nodes have discrete values and are represented by solid circles and continuous values are drawn with a broken circle. The first microprocessor is a RISC-V (overseas manufacturer and open sourced), the other is an Intel x86 with Trusted eXecution Technology (TXT) (USA based manufacturer, reputable suppliers, etc.). The microprocessor trust metric and supplier trust metric nodes have a common effect causal relationship with the microprocessor node. The conditional probability table is constructed by the first node being the hardware trust metric using values from **Table 5-9**, and the second node is the supplier trust metric with values from **Table 5-13**. The probabilities are shown in the boxes to the right of the diagram. By intuition, the Intel x86 with TXT provides a higher trust level than "other" (the RISC-V) chip, since the manufacture is USA based, followed the supplier check list ratings and its hardware design integrity was higher as well. These were compared with "other" (RISC-V) processor where the supplier was marginal, and the design integrity was low-med rating and did not have complete design set of artifacts.

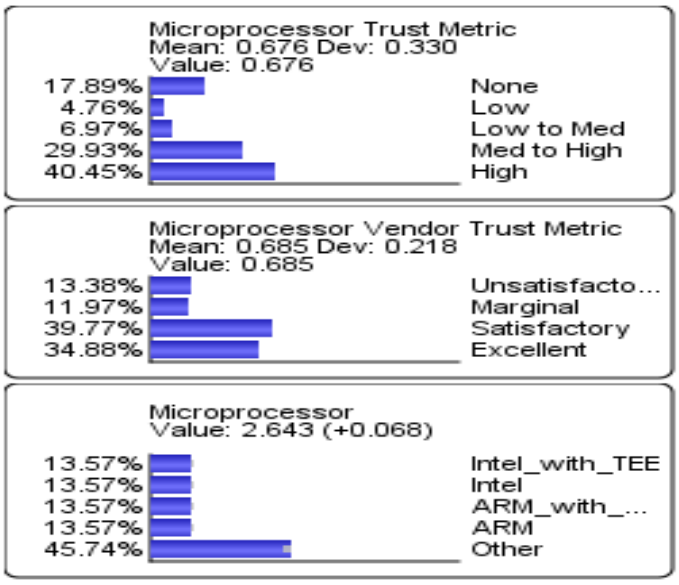


Figure 5-15: Initial conditional distributions

In Figure 5-16, we first set the observation for both the hardware trust metric and supplier metric as shown in the green bars on the right-hand side, the microprocessor effect is shown in blue.

By setting both values for hardware and supplier to having low to medium design quality and marginal rating for the supplier, we infer that the microprocessor node changes to the "other" type in the model. We can reason about the microprocessor assurance where the conditional probability is dependent given both hardware design and the supplier qualities. In Figure 5-17, we set the observation at the microprocessor

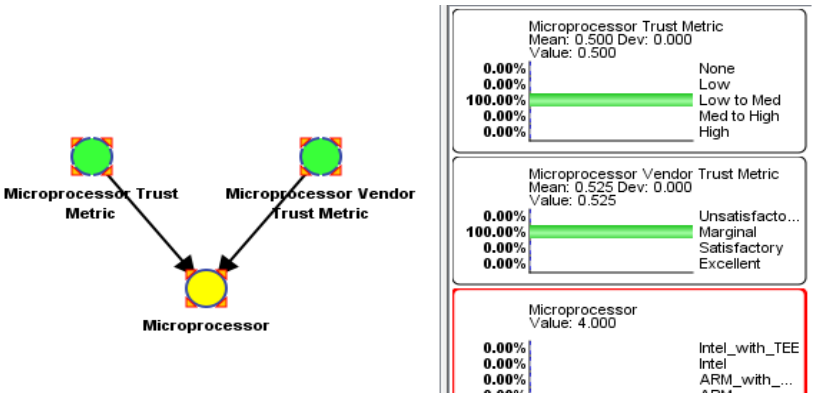


Figure 5-16: RSIC-V open source processor

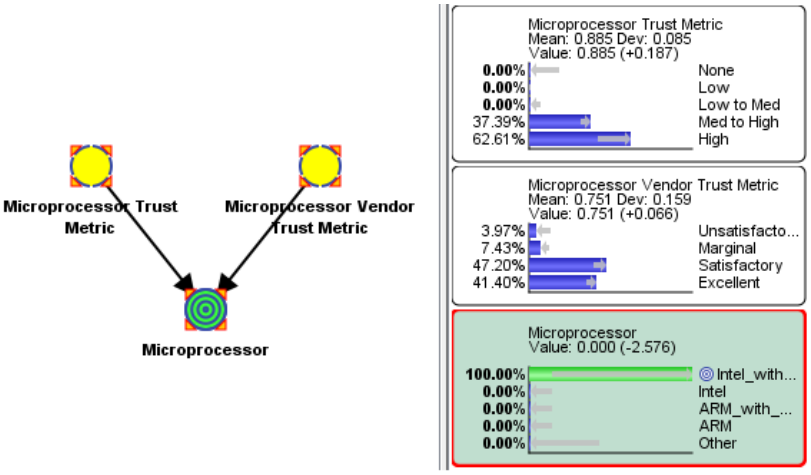


Figure 5-17: Intel TXT processor

node (selecting the Intel TXT processor) first as shown by the green bar and both hardware (hardware design quality is set at 62.61%) and vendor trust metrics (vendor quality is set at 47.2%) as shown in blue bars. We can deduce that having an Intel TXT processor from both design quality and supplier is a lower risk than having a lower design quality and a marginally rated supplier. This is the posterior on both hardware and vendor metrics given observation on the microprocessor. The tool allows a unidirectional path for causal inference.

By understanding the autonomous robotic system configuration, programming logic can determine the best optimization for how an external request can be handled by selecting various parameters within the BN. Utilizing the tool, we can look to find the best solution for achieving the Intel TXT as the target using the target optimization function, the results are shown in Figure 5-19. The best solution was having both microprocessor trust metric and microprocessor vendor trust metric set to high and excellent in green. The initial values were set in yellow at (.6297, .6853) and the best solution at (.95, .9), respectively. A list of solutions is given where the best solution had the posterior probability at 25 percent, marginal likelihood (joint probability) at ~12

percent, likelihood (evidence) at ~15 percent and Bayes factor (quantifies the impact of observing the Target State) at ~ 1.2. This is followed by three solutions ranked by a score.

By using these types of capabilities, we can further explore the effects of how both the design quality and supplier affects the security posture of the systems. This provides insight into potential risks that may not have been visible before. We utilize these concepts for an autonomous robotic system example.

Initial State	
Prior Probability	Joint Probability
20.0554%	100.0000%

Search Method: Hard Evidence

Synthesis		
Nodes	Microprocessor Trust Metric	Microprocessor Vendor Trust Metric
Initial Value	0.6297	0.6853
Best Solution	0.9500 (0.3203)	0.9000 (0.2147)
Min	0.2500 (-0.3797)	0.2255 (-0.4598)
Max	0.9500 (0.3203)	0.9000 (0.2147)

Best Solutions							
Microprocessor Trust Metric	Microprocessor Vendor Trust Metric	Score	Posterior Probability P(s|E)	Marginal Likelihood P(E)	Likelihood P(E|s)	Bayes Factor BF(s,E)	Generalized BF GBF(s,E)
High(5/5)	Excellent(4/4)	31.7620	25.0000%	12.5937%	15.6986%	1.2465	1.2925
Low to Med(3/5)	Satisfactory(3/4)	34.4577	20.0000%	14.5105%	14.4704%	0.9972	0.9968
High(5/5)	Satisfactory(3/4)	34.8175	20.0000%	14.3606%	14.3209%	0.9972	0.9968
Low to Med(3/5)	Excellent(4/4)	39.2923	20.0000%	12.7252%	12.6900%	0.9972	0.9968

Figure 5-19: The result from running a target optimization function

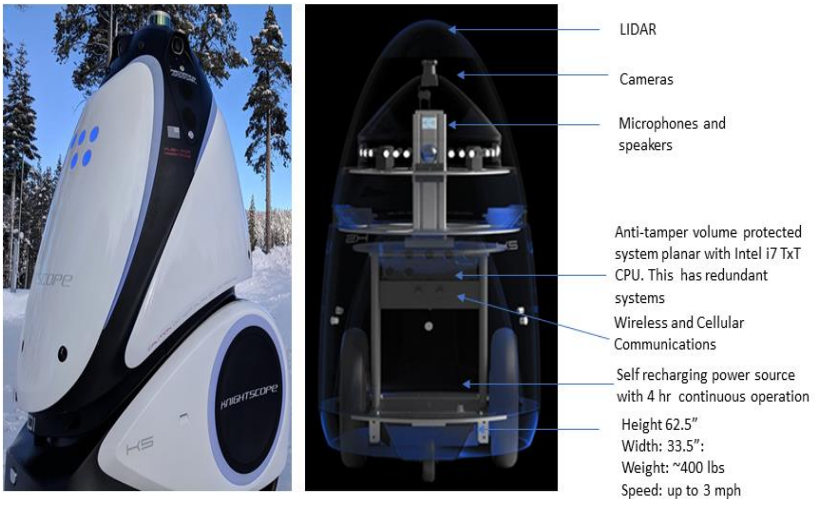


Figure 5-18: Autonomous Robotic System Model

We first define an autonomous robotic system with security features and then describe the model using the trust metrics defined above. An autonomous robotic system is shown in Figure 5-18 that is a Knightscope K5 product that is modified to support security features [179] [180].

These security features including the following:

- Anti-tamper volume protection for the system planar, that houses an Intel i7 Trusted Execution Environment (TEE) microprocessor and associated components like memory, cryptographic modules, and security manager logic. The tamper protection supports countermeasures against physical probing, rapid environmental temperature changes, voltage, and timing induced attacks. A battery source is supported to protect against offline attacks.
- A Linux OS variant that is stripped-down with security features like SELinux Mandatory Access Control (MAC) policy, signed audit subsystems, vetted device drivers and removal of a direct root account.
- ROS 2 and RTI DDS are supported. DDS security is supported for Data in Motion protection.
- Fault tolerance is supported, so that continuous processing is available for operational resiliency.
- Cryptographic keys are supported in a Battery Backed up or Physical Unclonable Function (PUF) version.
- Trusted Platform Module 2.0 is supported.
- Secure Boot for Firmware and OS is supported where both confidentially and integrity or only integrity are implemented.
- Trusted Boot is supported utilizing Integrity Measurement Architecture (IMA) and Extended Verification Module (EVM) for offline file integrity protection.
- Hardware Cryptographic Module (HSM) is supported with Commercial National Security Algorithm Suite (CNSA Suite). These are the new quantum resistant algorithms.
- Data at Rest is supported utilizing the HSM and on-board 1T NAND flash storage.
- Camera sensors are redundant to protect against AI attacks.
- AI robustness metric have been validated on algorithms used for navigation and object recognition.
- Supervisor logic is supported to monitor the anti-tamper sensors and monitor system resources for anomaly behavior.
- Working with the supervisor logic, the Cognitive logic supports the offense/defense and resiliency logic. The offense/defense logic analyzes the data produced in the supervisor logic module to take an offense/defense position when resources are being exhausted in an abnormal behavior by shutting down the offending process or limiting resources for the process. This same logic

interrogates external requests by understanding the resources it has and applying a best approach to fulfilling it. On the defensive side it takes input from the National Vulnerability Database (NVD) feed and determines what vulnerabilities are pertinent to the system configuration and how to handle those threats.

- Secure software updates are supported with confidentiality and integrate or only integrity protection.

Other features:

- Each of the supply chain vendors are highly regarded and meet the criteria for an excellent trust metric score.
- The system, hardware, software, and AI also have developed their components to meet the highest trust metric scores in each individual category.

We also define a set of operational states that a robot experiences during its lifecycle, but these can be modified or extended to different systems. Referring to Figure 5-20, the first state is called initialization where the system is setup by loading the platform firmware, OS, services/application software and platform keys. If the loading of data is highly sensitive, this will need to be done in a secure environment and with the anti-tamper components armed. In the personalization state, the system components are configured, and any additional required keys can be brought on the platform via the platform keys in a secure manner. The benefit of having keys brought on the platform vs generating on the platform is that they can be recovered. Configuration can consist of an IP address for home servers where data can be uploaded, and software updates can be downloaded. User IDs and policies can also be configured in this state. In the maintenance state, the robot can undergo a number of actions related to repairs of broken components, updates for both hardware/software components, recharging its power source and calibration. Calibration maybe self-induced with a scheduled time for it to occur or can happen by a service person after a repair. Calibration is the process to algin a component within a tolerance range using

equipment or process that provides repeatability results. After a maintenance action is performed, the robot can transition to operational or end-of-life state. The operational state is defined as the normal operating condition of the overall system and from this state the robot can transition to maintenance state to end-of-life. The end-of-life state enables an entity to achieve data and/or clear all sensitive data.

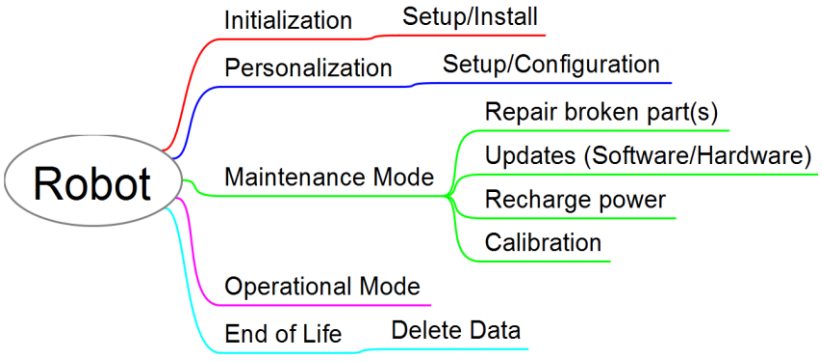


Figure 5-20: A Robot's lifecycle states

We showed a small example of a Bayesian Network, described the autonomous robotic system security features, and would like to expand the view to present the internal layers of the autonomous robotic system. There are a number of steps or guidelines that need to be taken in order to construct a BN for a security assessment. First create the corresponding nodes for the hardware components of a robotic system, this should include the hardware design and vendor metrics for each of the components. The next step is to add the software nodes to the BN model, this also includes the software and vendor metrics. If security features are supported, they must also be included as part of hardware and software components. If Cognitive/AI components are supported, these nodes should be layered in the model above the OS layer. An AI vendor should provide robustness parameters, but in the event they do not, they should provide testing models/dataset so that one can obtain them from running validators like

CLEVER or CNN-Cert. The accuracy, supervisor, and maintenance layers will need to be defined by the underlying hardware/software/security features. Once the model is constructed and the nodes are defined with their corresponding CPT values, one can start to set the appropriate evidence for the nodes. By defining an assurance level of a target node, the BN model can be used to simulate the outcomes of the evidence set on the nodes or using the optimization technique as discussed above, they could set the desired values to obtain the results. Each of the layers should have a corresponding assurance level, so that by setting the security features as evidence on several nodes, this in turn will change the other nodes. By running the simulation, the result on the target node will change to the level that was selected. A target node is considered a dependent variable in traditional modeling approaches.

Figure 5-21 provides an example BN that applies the trust metrics outlined in section 5.3 with the security features/functions of the autonomous robotic system described above. Starting from the right, in blue, is the firmware layer, yellow/orange

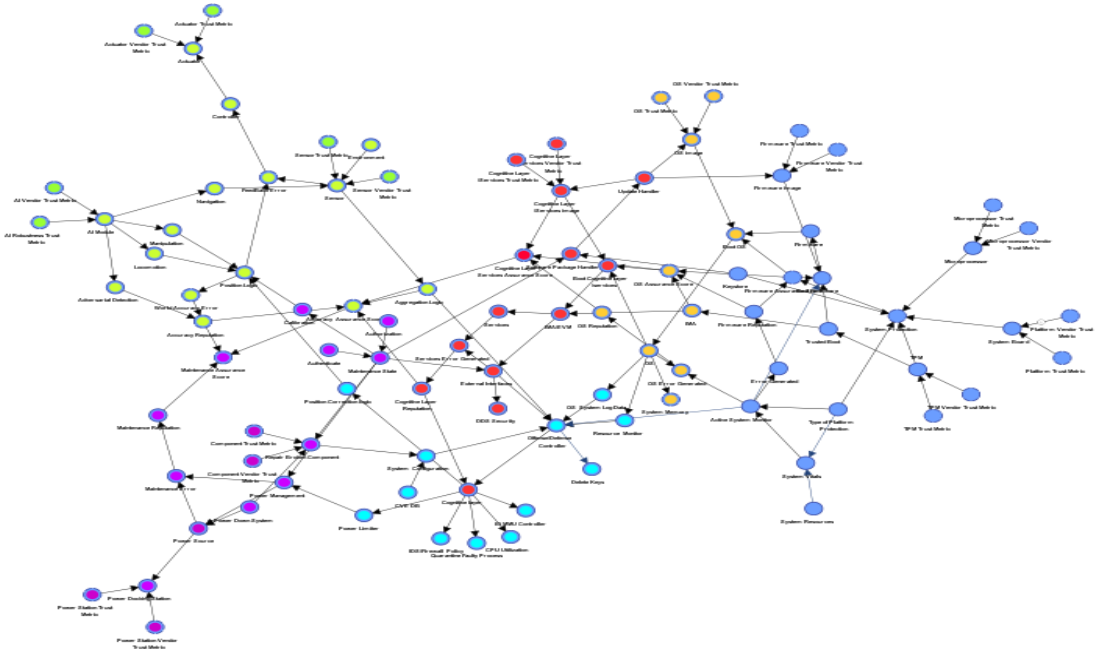


Figure 5-21: A Bayesian network of an autonomous robotic system's internal assurance model

is the OS, red is the cognitive/services layer, green is the accuracy (robot control system and AI algorithms) followed by purple as the maintenance logic and teal is the offense/defense or supervisor control logic. Each of the layers have independent scoring related to the configuration and supported security features of the system.

The assurance score and a reputation score are propagated into each level so that the child layer has knowledge about its parent layer. These scores can be used to control the overall assurance level for the system. For example, a robot that needs to support a high assurance level, high loss of damage and high reward to exploit would select each of the layers to a high assurance score. Of course, the assurance scores are based on the physical hardware and what security features are enabled. To add additional assurance at the platform and accuracy layers a set of fault tolerant features can be enabled. Once the security posture is assumed to be a specific level, the offense and defense controller can take actions on a process that it has determined is abnormal. The CVE database can be used to determine if any knowledge can be obtained for this abnormal behavior or take precautionary steps to prevent an event. The cognitive layer interacts with the offense/defense controller to ensure that the system can function within certain limits. The internal state of the system must be sound and known if external interaction is to take place, the external request can be scrutinized and therefore, extend the trust model with the entity. Having the supply chain vendors also being part of the system model helps establish the pedigree of the components. The ISO9000 is a standard that follows the documentation of component provenance and quality assurance.

The methodology for assessment is largely applied on the hardware and software (including AI) design for the components, the integrated system, and the supplier (both at component and system levels). We describe each layer of the Bayesian

Network model, but only focus on a single layer in detail since the same methodology can be applied to the other layers.

Firmware

The firmware layer is the very first layer of the system stack where hardware and software interact when power is applied to the system. In this example, we assume that the system vendor has loaded initialization and personalization values into the system at a secure manufacturing site to achieve high assurance levels. The firmware

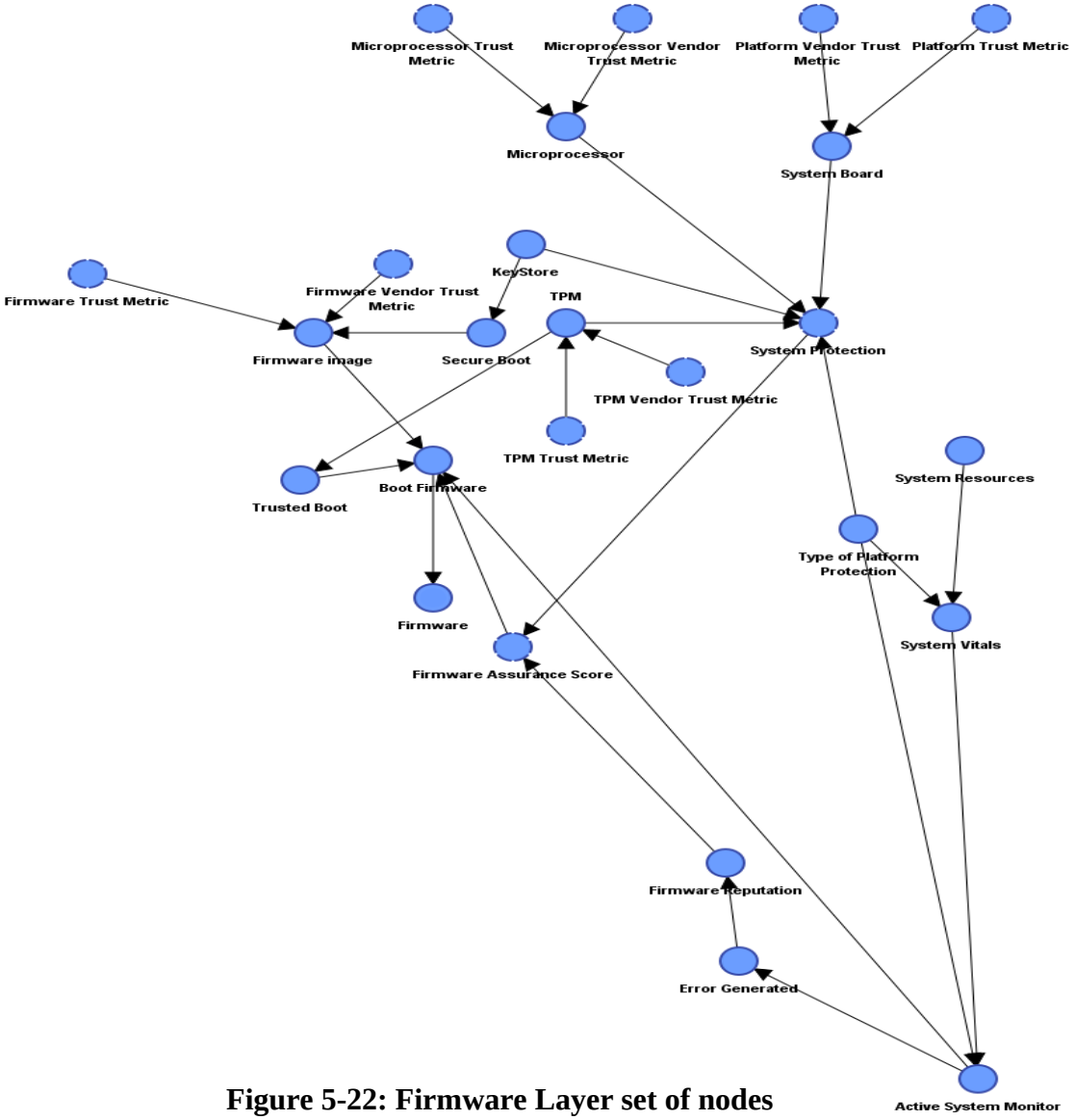


Figure 5-22: Firmware Layer set of nodes

layer nodes represent system, hardware, software, and supplier entities as shown in Figure 5-22.

From the top we have a supplier vendor and the hardware design metrics for the microprocessor and the system board. These are shown in Figure 5-23 with their respective probabilities. The set of nodes for the microprocessor and system board are both common effects. The microprocessor was introduced in the example above, but in the context of the larger BN the ability to differentiate the types of processor is needed. Here we defined the Intel TXT and AMD Trusted Zone as having a higher security level than regular Intel and AMD processor types. We have an "other" field for all other

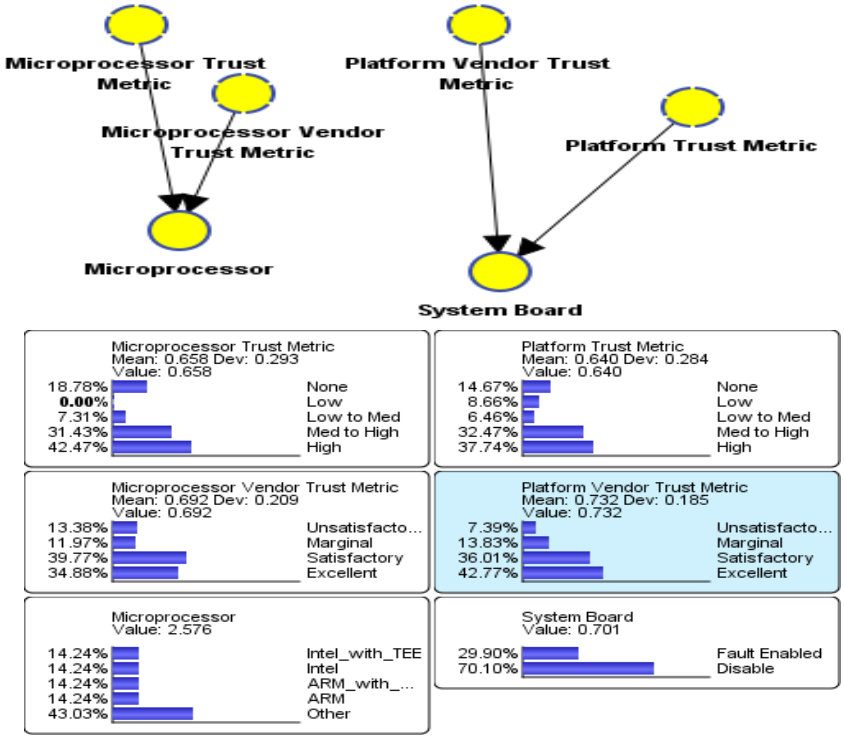
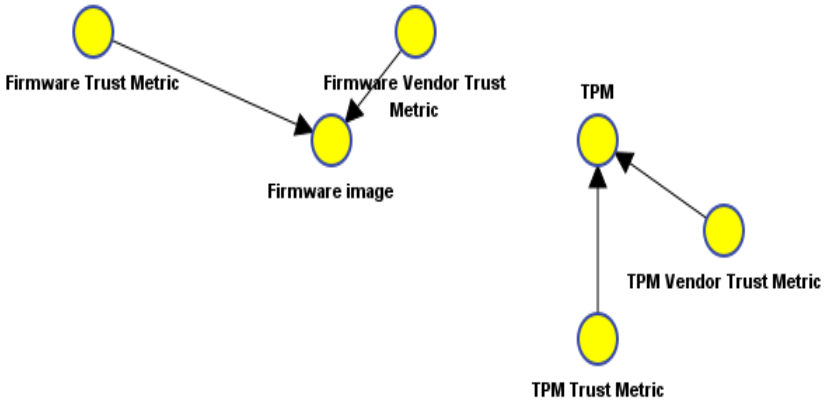


Figure 5-23: Microprocessor and System board related nodes

types of processors. The system board represents the supported feature of fault tolerance as being enabled or not.

A set of parent-nodes for the Trusted Platform Module are the TPM vendor and the hardware design trust metric. The firmware has a similar set of parent nodes that are related to the software trust metric. These are shown in Figure 5-24 with their respective probabilities. The set of nodes for TPM and firmware are both common effects. Here we defined the TPM as being present/enabled or not available for trusted boot and other functions. The firmware can have a configuration of secure boot with confidentially or secure boot with integrity or no protection at all. For high assurance levels having the firmware encrypted and/or signed is a requirement. Having a Read Only Memory (ROM) helps with kicking off everything with a mini bootloader that is immutable. If this is enabled on the system planar that helps with the firmware decryption and validation before it is loaded, and the secure boot chain can start from the ROM up the software stack. By having the Trusted Platform Module being present, a trusted boot process can also be activated so that both secure and trusted boot functions can be performed at each software layer.



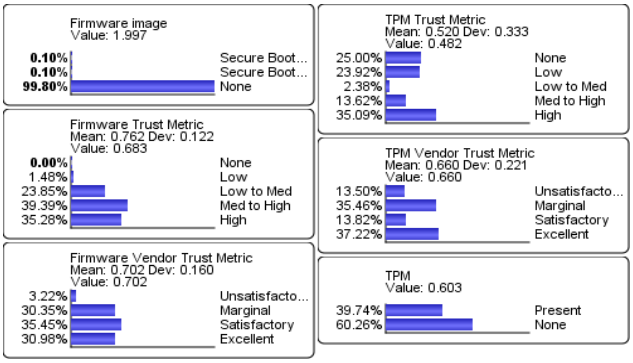


Figure 5-24: Firmware and TPM related nodes

In order to achieve higher assurance levels a set of features must be supported by the platform and these include hardware protection against physical and side channel attacks. A set of nodes that captures the protection configuration are shown in Figure 5-25 with their respective probabilities, where common effect is being used. The system protection node uses the system trust metrics that captures the different levels of assurance protection and collects input from the microprocessor, system board, type of protection, TPM, and key store nodes. The type of protection node covers the tamper levels including tamper protection, tamper evidence or none are defined. These are high level definitions, but the BN can be expanded to include more details. The different types of keys stores are defined as (Battery Backed RAM (BRAM), Physical Unclonable Function (PUF), fuse based and none. It is important to understand the differences between the key stores, since key management and maintenance may become an issue if the policy were to have the keys wiped due to a tamper, resulting in consequences.

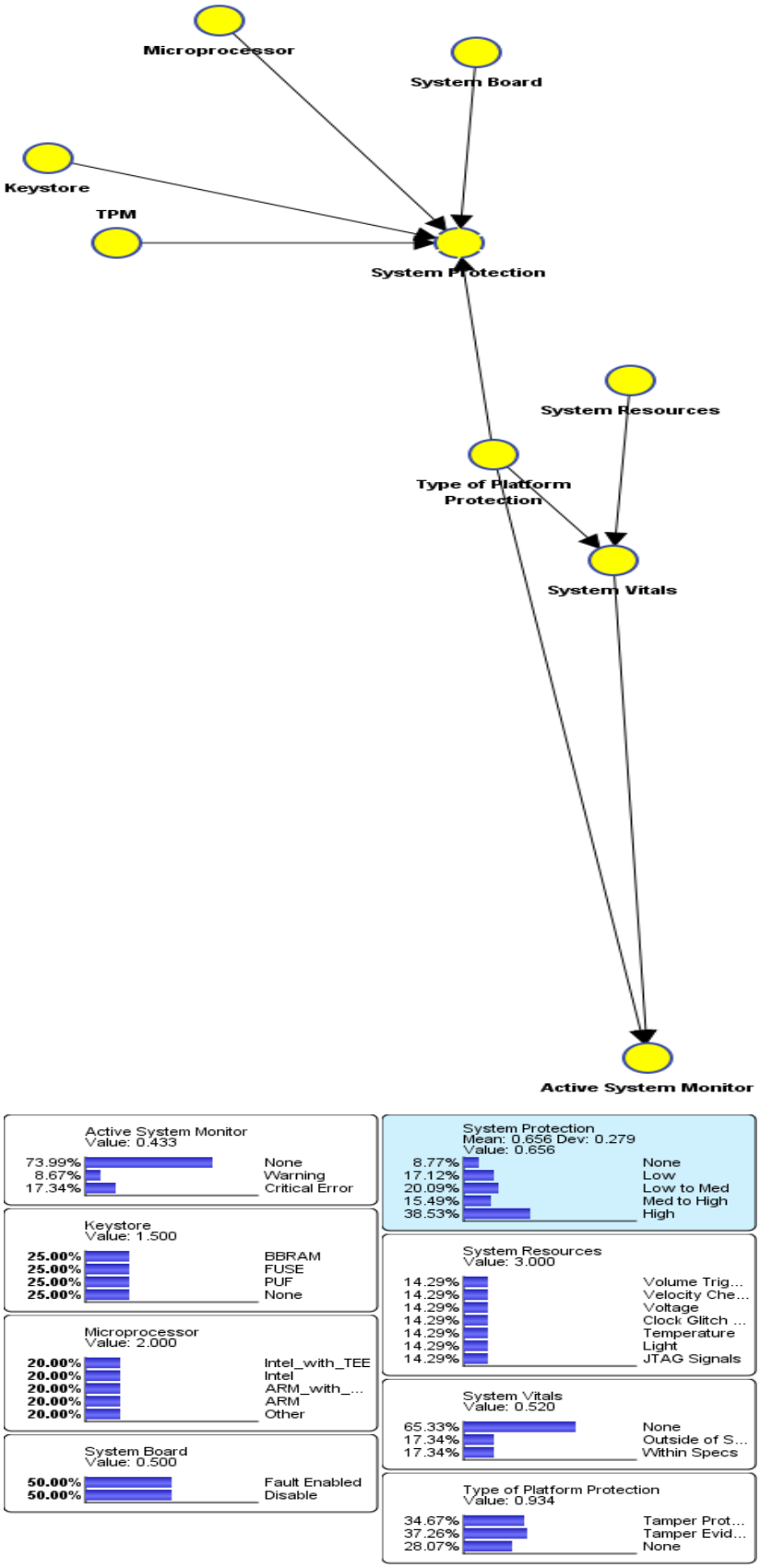


Figure 5-25: Secure protection nodes

As part of the system protection node there is an active monitor node, where inputs from different sensors are monitored for tolerance levels since these levels will put the system into a countermeasure state if any of the trip levels are reached. The active monitoring functionality is directly related to the type of protection that is configured. Tamper protection inputs from system resources and system vitals are fed into the active monitor. System resource are capturing the volume trigger, velocity check (number of Power on Reset cycles), voltage, clock glitch check, temperature, light (X-Ray or other types), and Joint Test Action Group (probing). This is an example list, but again the node can be modified to include several sensor types. The system vitals are defined as none, outside or with specification.

The remaining set of nodes are primarily related to firmware assurance score and the booting of the firmware. The firmware assurance score represents the elements of the secure configuration to meet the desired assurance level, but takes into account a reputation node where if an error has occurred this is tracked as being a negative rating vs a positive one with no error. The set of nodes in Figure 5-26 shows a causal chain where the data flow determines if the system should boot depending on the assurance score. To obtain a firmware assurance score the configuration evidence parameters must be set to the appropriate level and if no errors are indicated the boot image can be executed.

Figure 5-26 depicts the error generated node defined as (True or False) and the reputation node defined as (Positive or Negative). The firmware assurance node is defined by the system trust metrics since this supports both hardware and software components as a system. The boot node is simply a (true or false) choice, and the

firmware node is reflective of the post boot behavioral states (normal, slow, not responding or not executed).

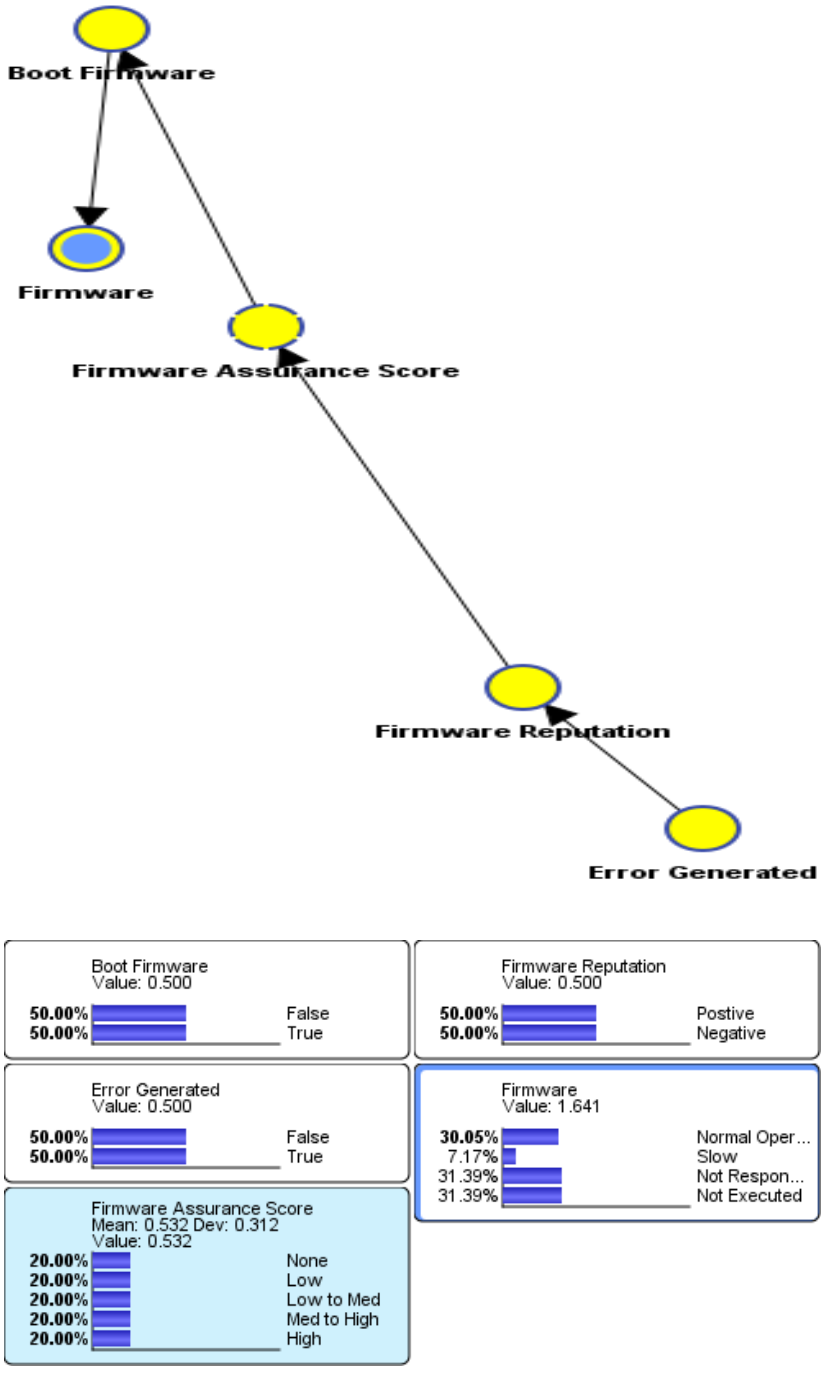


Figure 5-26: Firmware assurance score

We have described all the nodes for the firmware layer and would like to simulate the BN by setting the firmware assurance as the target with value of high. We now need to set the observation on the following nodes shown in Figure 5-27 and Figure 5-28 in green. Figure 5-27 represents casual inference changes on the remaining nodes:

- Microprocessor = Intel with TEE
- Keystore = BBRAM
- TPM = Present
- System Board = Fault Enabled
- Type of Platform Protect = Tamper protection
- Firmware image = Secure Boot Encrypted
- Active System Monitor = None

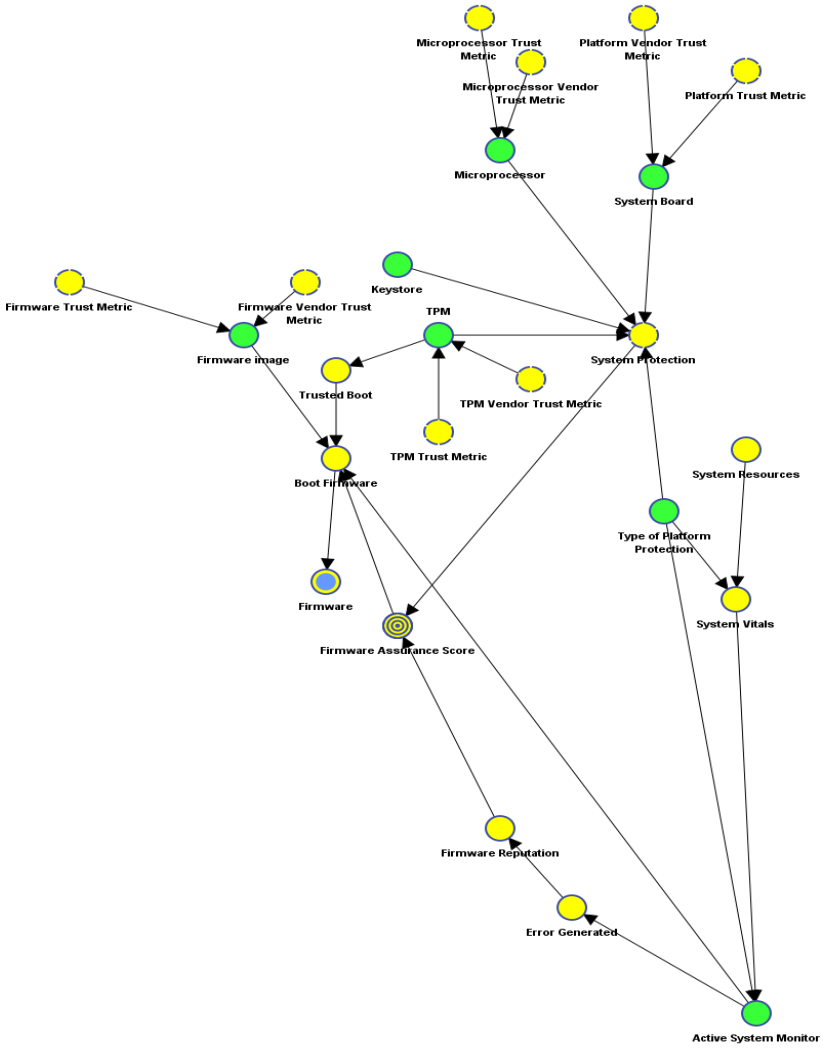


Figure 5-27: The result of the nodes when the observed nodes in green are set.

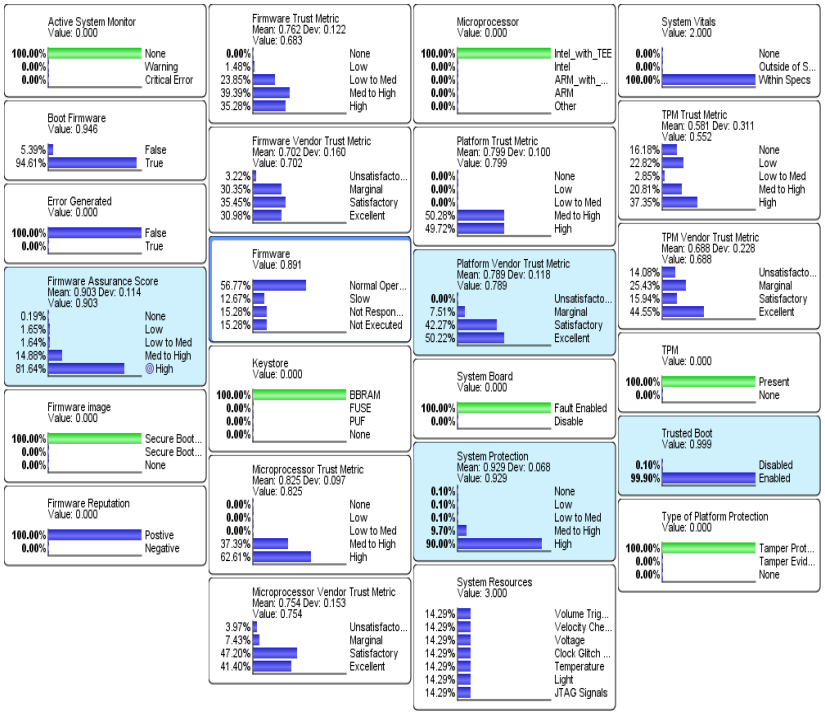


Figure 5-28: Firmware Assurance target set to high, that results in a posterior distribution.

In Figure 5-29 is the result of the target's posterior direct effect on the target node shown by a tornado diagram. The 16 nodes are represented on the left-hand side minus the 7 nodes that had their evidence set. The x-axis is the probabilities for each of the levels for the firmware assurance score where the yellow indicates the highest at 81.64% probability.

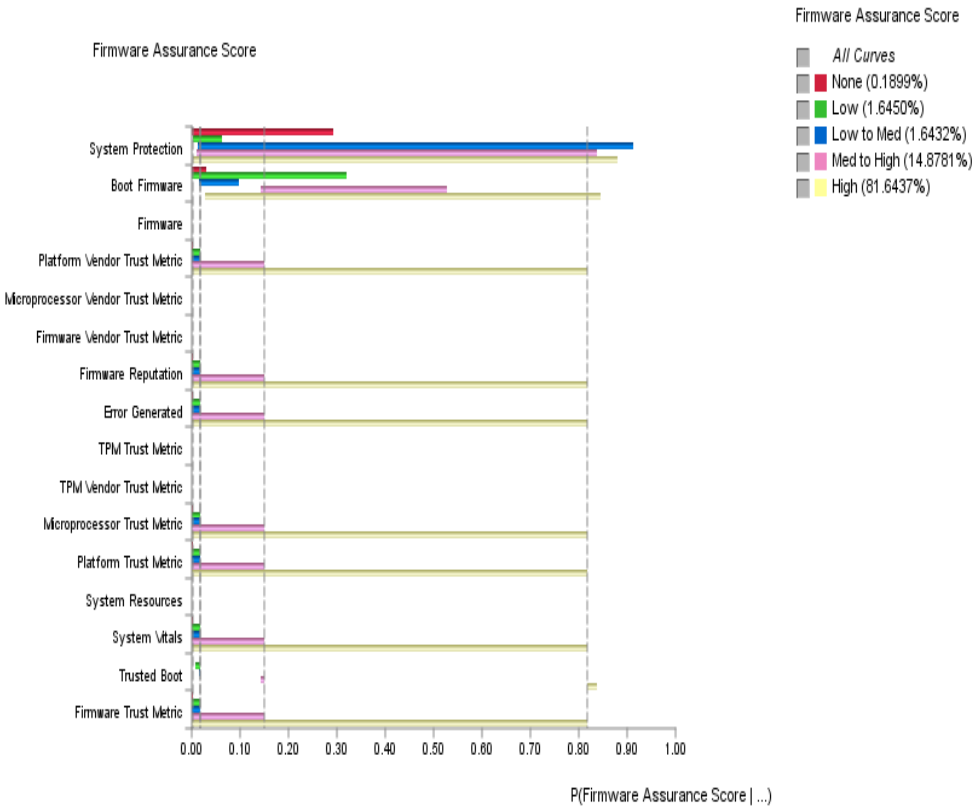


Figure 5-29: Posterior probability analysis of firmware assurance score

In Figure 5-30 is the result of the target's posterior direct effect on the target node shown by a parameter sensitivity analysis as a distribution function. This decomposes the marginal distribution over the number of scenarios where the joint distribution was defined. Using the same evidence set above, the probability of firmware assurance = high ranges from 0.001% to 92.96%, with a mean value of 83.04%. By tweaking the inputs, the assurance level can be adjusted to reflect the security features supported by the platform.

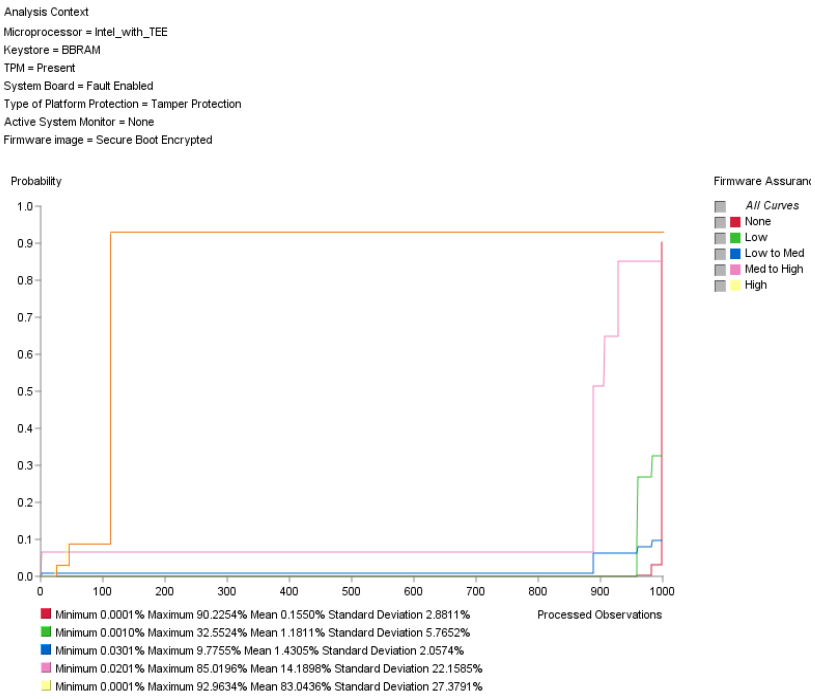


Figure 5-30: Firmware Assurance set to high with observations set on green nodes

The firmware is the first layer of the stack that enables the underlining protection configuration parameters. The firmware assurance score is an input into the OS layer as the next layer to execute. The score provides evidence to the OS layer that the system can support a set of security features and the boot process executed with a reputation rating. In modern systems the firmware boots with very little information being relayed on the next layers.

Operating System

The OS image node has parents that are both the metrics for the supplier and the software itself. The OS image can also support its protection as being encrypted, signed or none. The Boot OS has the values of true or false, depending on the input from

firmware assurance score, and how the firmware is operating (performance characteristics). If the Boot OS is enabled the system memory is monitored for performance characteristics.

If the TPM is present and a Linux OS is used, the Integrity Measurement Architecture (IMA)/Extended Verification Module (EVM) features can provide protection to files above the OS layer from being manipulated. This also helps with offline attacks on the files since hash values and digital signatures are used and stored in the TPM.

Like the firmware layer, the OS has an error/reputation node so that past performance can be analyzed before booting the next layer. Figure 5-31 shows the OS layer nodes and Figure 5-32 is a representation of the CPT values. Depending on the OS boot process system, memory characteristic should be known for each of the

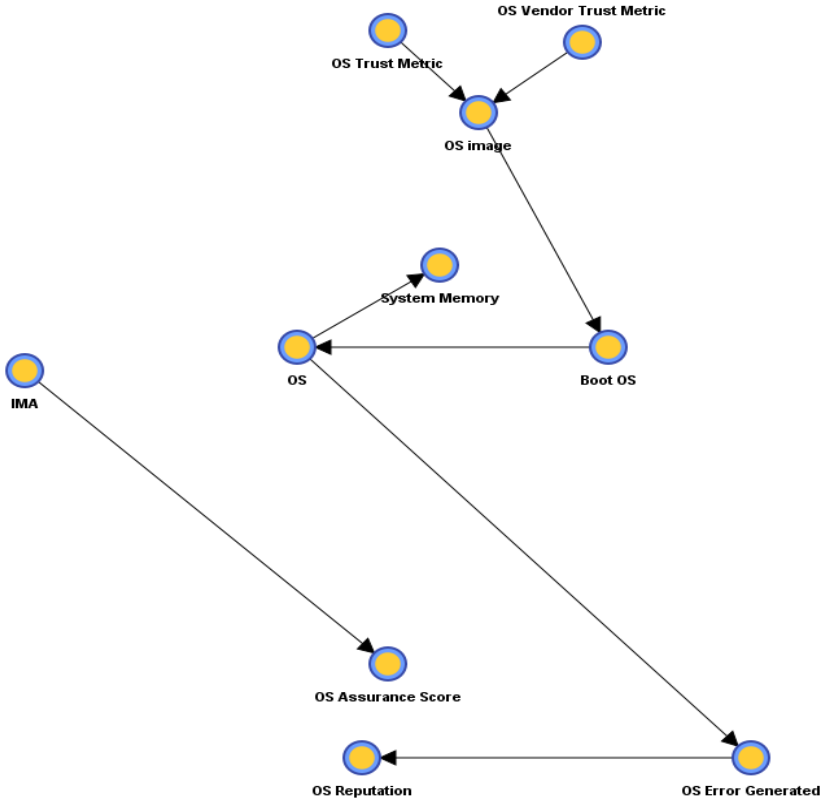


Figure 5-31: The OS layer set of nodes

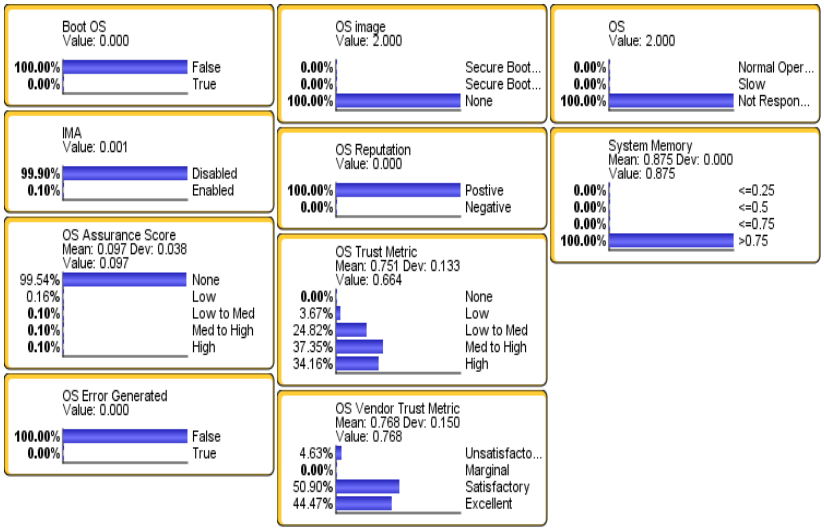


Figure 5-32: OS Layer monitor nodes

configurations and if the utilization is outside the known values, the system will be able to determine the process ID of the offender. We do not consider the authorization mechanism and if other policies like Security-Enhanced Linux (SELinux) are used to lock down the system. The model can be easily expanded to include these features. We do assume that mandatory access control (MAC) is supported on most modern OSs. The OS assurance score builds on the firmware assurance score by adding the additional security features such as the OS image being protected, the support for IMA/EVM, system memory being monitored and the firmware's reputation. The OS assurance score is sent onto the cognitive layer as the next layer to be executed.

Cognitive Layer

The cognitive layer is a set of nodes for determining the health of the system and can adjust resources accordingly. The cognitive layer is also part of the resiliency logic in teal, but the different colors make it easier to see the difference between them. Some of the functions that the cognitive layer performs are related to observing the

behavior of the system, like making sure that correct package protection is being used for the known vendor updates, or for external communications that the correct protection mechanisms are being utilized.

The other key responsibility of the cognitive layer is to interact with the resiliency layer to determine the course of action when evidence is presented to act. Where the cognitive and resiliency layers live and run is an implementation question, but having them being separated from the firmware and OS layers isolates the logic from being attacked. The cognitive layer receives an assurance score from the OS layer to set its policy for that level.

The Cognitive layer/service image node is a common effect where both the AI robust trust metric and the vendor metric represent the quality and effectiveness of the image. Once the OS is finished executing it starts booting the required services. As part of the cognitive services boot process an interaction will take place for checking the services under the IMA/EVM protection and ensure that they were brought up correctly. For example, if DDS security for external secure communications is utilized, the different protection kinds will be analyzed to ensure that they correspond to the assurance score. This technique was discussed in the mitigation for ROS 2's security vulnerability [181]. Other secure protocol mechanisms can be used and added to the model. If any issues do arise from the boot process the error and reputation nodes will be updated, this will be reflected in the assurance score. Figure 5-34 shows the Cognitive layer nodes and Figure 5-33 is a representation of the CPT values.

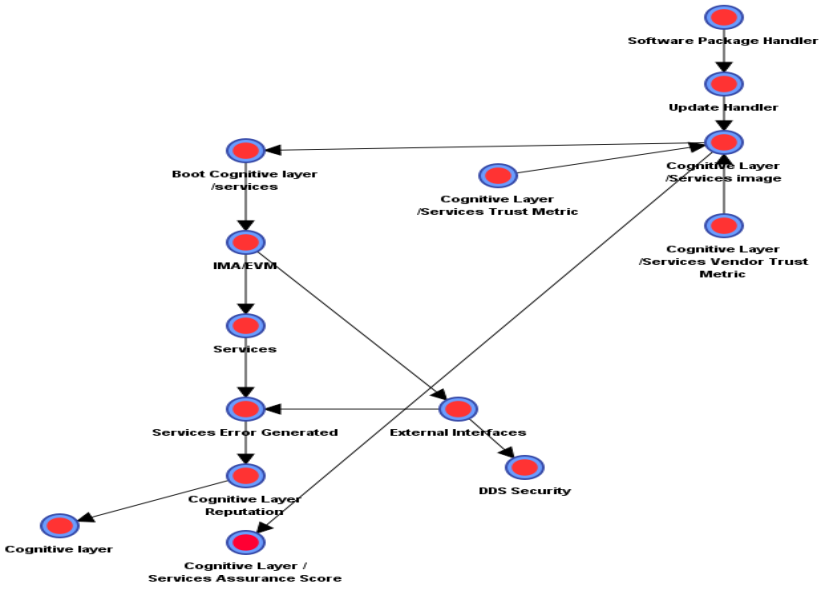


Figure 5-33: The Cognitive layer nodes

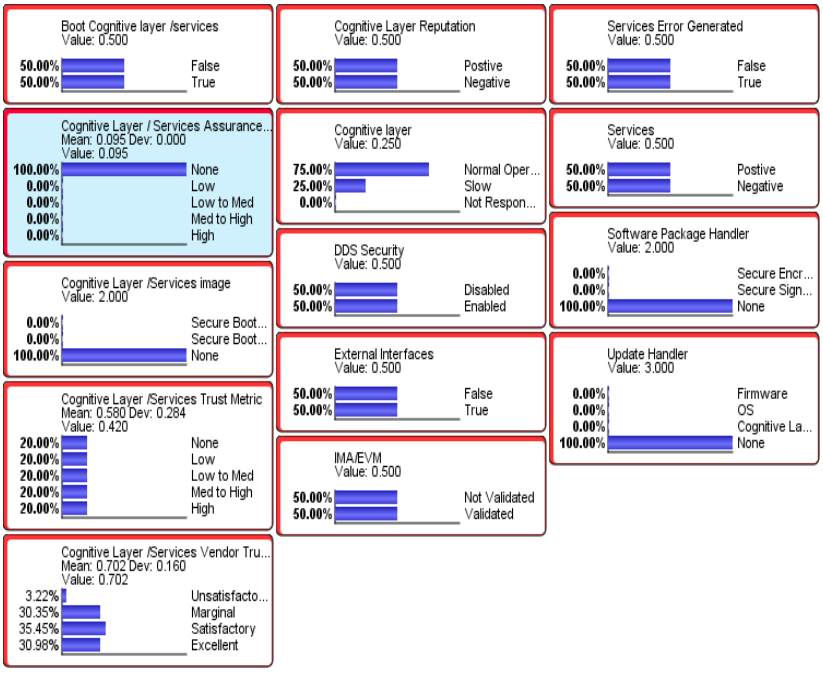


Figure 5-34: The Cognitive layer CPT values

Offense/Defense or Supervisor Logic Layer

The Offense/Defense, Resiliency, or sometimes called the Supervisor layer provides logic to act on one or more nodes that might be constructed to counter a threat. The Cognitive node is presented within this set of nodes, as it is the node that makes the decision to determine best course of action with the evidence that it has been presented from the Supervisor layer.

The Supervisor has a couple of input streams that are related to a resource monitor that determines if power, memory, and CPU utilization are being consumed correctly by a process's characteristics. The log data can be analyzed for anomalies and alerts from external vulnerability databases like the Common Vulnerability and Exposure (CVE), which can be incorporated into the supervisor's knowledge base. The entries will need to be matched up against the system configuration to ensure that the correct data is being correlated to the platform's policy.

Active Monitor is another input, when tripped from a physical attack the mitigation could delete platform keys. This would then degrade the system to a limited level of capabilities and should be considered a dire step. All other types of threats if detected, should have a countermeasure such as shutting down resource, blocking processes or isolating them. If supported, the IPS/Firewall will be controlled to open/close ports based on security policies and the application profiles. Application profiles have been used in different industries, which describes the resources required to run the application or dependent security features/protocols.

Depending on the application/service error and reputation status, if it is flagged for having a negative reputation it could get quarantined using the Input/Output

Memory Management Unit (IOMMU), or other isolations mechanisms. In the extreme case the process could get killed for alleviating resources and memory space.

Unique to autonomous systems is the capability to know where it is in the world coordinate system. The conversion between world and robot coordinates may fall into several accuracy faults. The notion is to have the cognitive layer be the governing body that assists in the overall mission of navigation by looking at the bigger picture of the environment and goals. This is analogous to a captain and crew, where the crew for autonomous systems are the sensors. Navigation logic, drive system and cognitive layer are the captain function. Figure 5-35 shows the Supervisor layer set of nodes and Figure 5-36 are the corresponding CPT values.

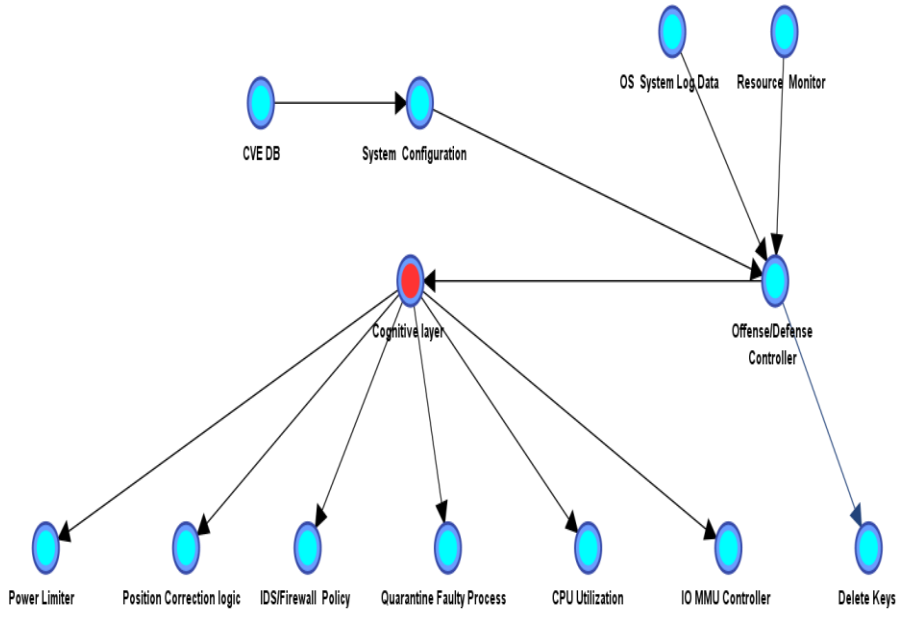


Figure 5-35: Resiliency Layer set of nodes

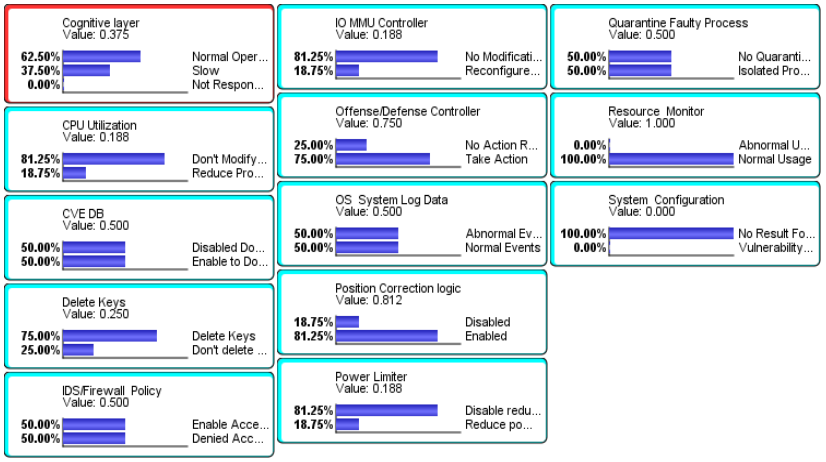


Figure 5-36: Supervisor Layer CPT values

Accuracy Layer

A robot must provide the assurance that its accuracy in world and local coordinates must be trustworthy within an environment. Different types of sensors provide real-time data about the environment and the robot must plan and act on this data. The data is represented in both world and robot/local coordinates and accuracy related to locomotion, manipulation and/or navigation must be precise. Figure 5-37 shows an accuracy layer in green nodes. We first assume that firmware, OS, and cognitive layers were started with an assurance score level that is determined by the security features in hardware and software. This base provides a root of trust for upper layer dependency, but the security posture must be viewed as a holistic system and not at individual layers.

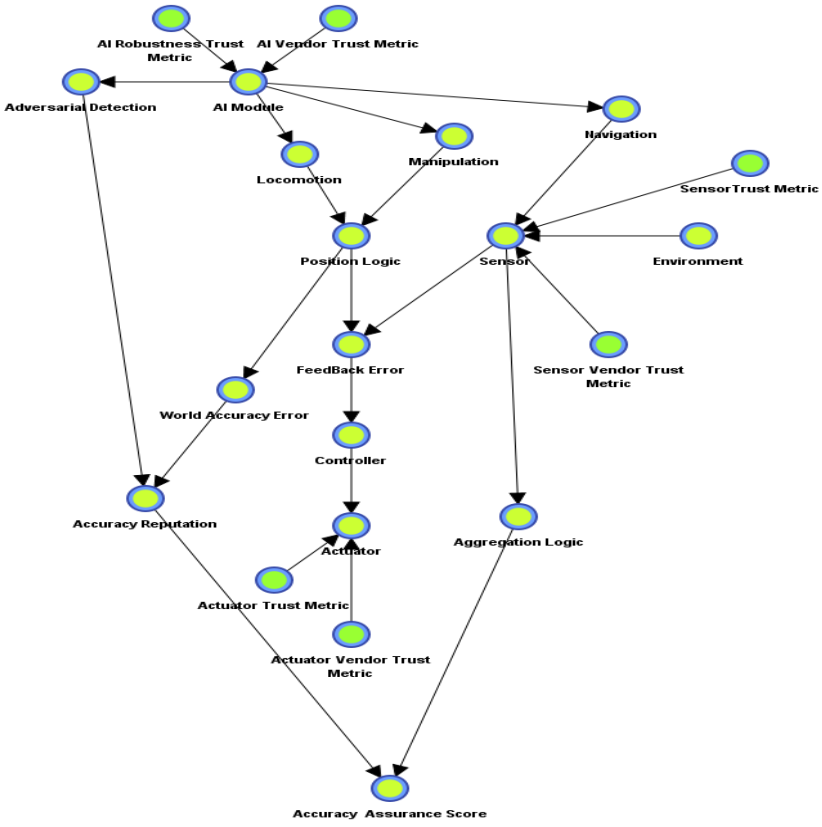


Figure 5-37: Accuracy Layer set of nodes

A basic autonomous robotic system is constructed by these sets of nodes. A robot must sense its environment for it to plan and act within it. The environment node provides a type of class for where the robot is located (land, air, and water), but this can be extended into other domains. The sensor is generic to represent a plethora of types. Associated with it, is the trust metric and vendor trust metric for the quality of the sensor and vendor rating. To increase the accuracy data a fault tolerance scheme can be designed to collect from multiple sensor sources. This is where the aggregation logic node is relevant since this collects data from each of the same sensors and performs the summing of data. Fault tolerant is an important feature to sensor data since it is hard to attack multiple targets at the same time. The aggregation node is defined as continuous since its values are in three range measurements. A controller logic is associated with

an actuator to perform locomotion of the robot platform or manipulation of the robot end effector. The position logic, feedback error, controller and actuator nodes represent a closed loop control system, mostly common in robots. The feedback error is a continuous node with values ranging in four groups that define the error margins. The controller has values that describe its feedback loop mechanism, and the actuator has values that describe its behavior when operating. The world accuracy error is relative to the world coordinate system and transform functions.

As part of this layer the AI algorithms perform the functions of navigation, locomotion, and manipulation. The AI robustness metric scores the certified boundary that the algorithms detect adversarial attacks. The distance closer to the certified area is the most robust and furthest away is least. Attacks on these AI algorithms are becoming a norm and having a metric to determine its effectiveness is needed in autonomous systems. The reputation node supports the rating from the accuracy errors, so that retained values can be analyzed for decisions. Figure 5-38 shows the CPT values for the Accuracy Layer set of nodes

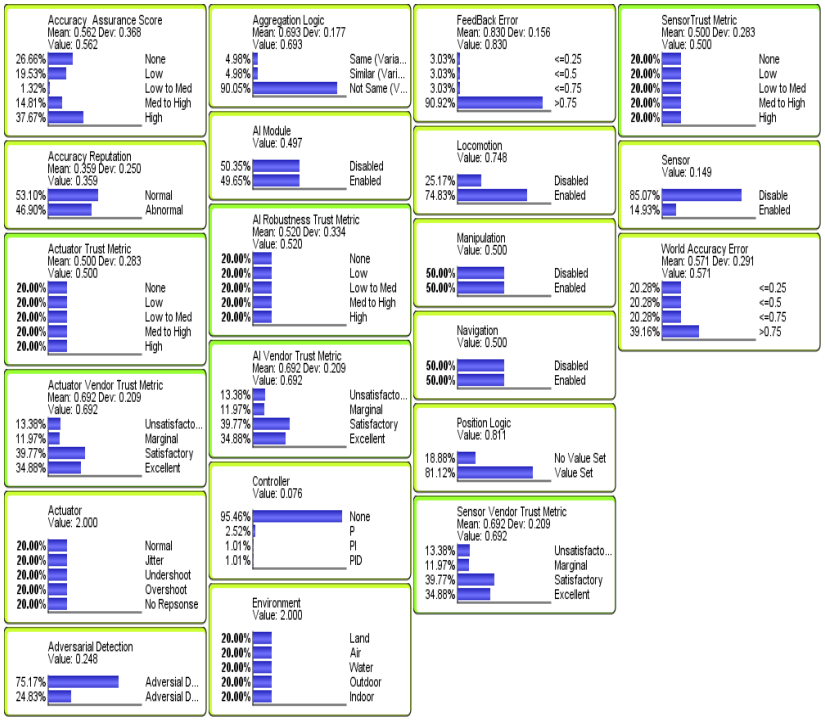


Figure 5-38: Accuracy Layer CPT values

Maintenance Layer

During the lifecycle state an autonomous robotic system must perform operations to replenish its batteries, update its software, perform calibration, or repair a component. We capture these operations in our model to determine if any abnormal activity is commencing or being detected. Figure 5-40 shows the maintenance layer set of nodes and Figure 5-39 shows the CPT values.

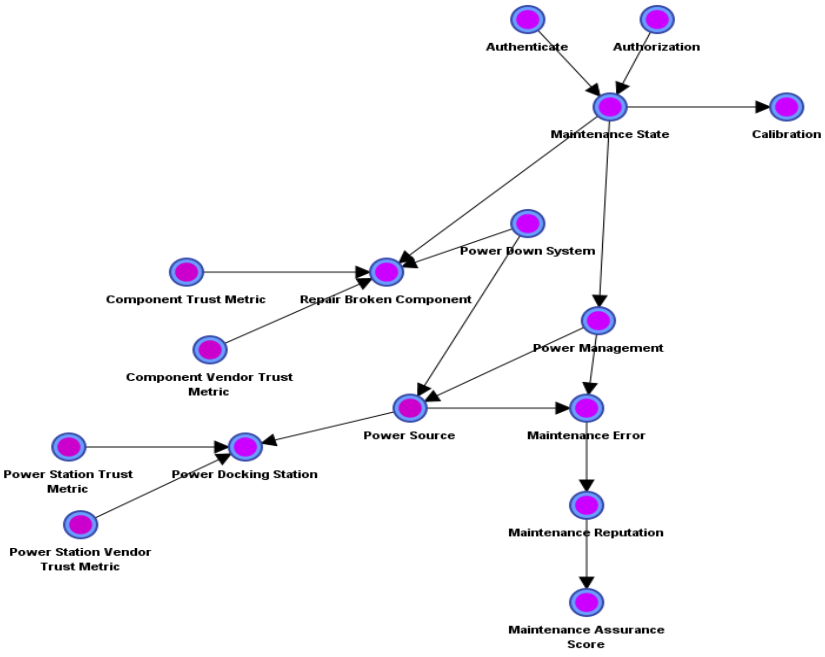


Figure 5-39: Maintenance layer set of nodes

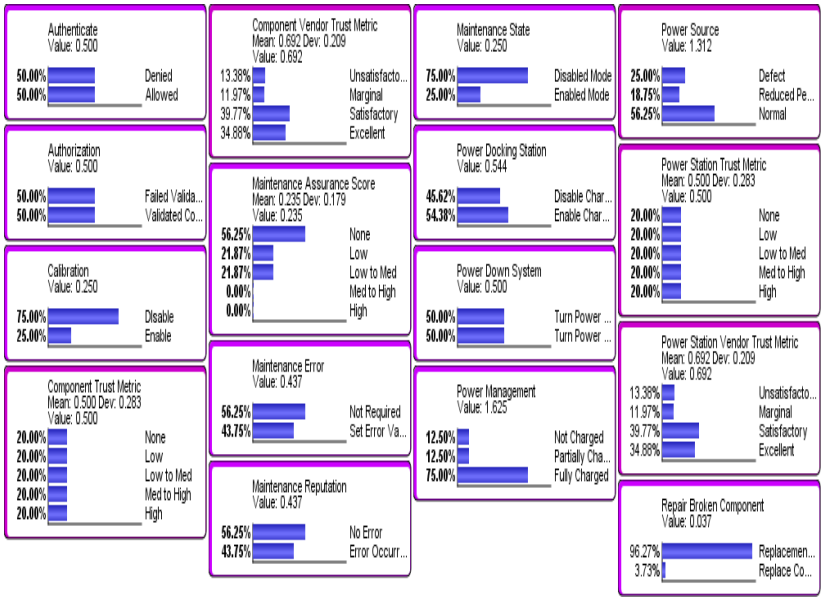


Figure 5-40: Maintenance layer CPT values

In order to perform any of the maintenance functions the autonomous system must complete its current task if one has been initiated or try to suspend it depending on the request. This will place the system under direct control of the maintenance state. For power replenishment the robotic system should have a map were to go, as an

authenticated and authorized supply station will be desired. The model provides the metrics for both the power station supplier vendor and for the power station equipment, hardware or system metrics can be used for this node. By having known sources, this eliminates potential security risk, like skimming ATM machines, where personal data is stolen.

Depending on policies, software updates may not be available in a dynamic manner, where anytime an update is released it automatically gets propagated down to devices. The question of how these systems will be managed is still a research topic. The model does provide the capability to support updates in three flavors of packaging (secure encrypted, secure signed and none). A package handler will be used to distribute the payload to the appropriate system level, being firmware, OS, or cognitive layers.

For component replacement, the system might be required to be shut down. This will take an authorized request to perform this action. Both nodes Authentication and Authorization will check that validation has been performed. A manifest will need to be updated with a record of the service (what was done, by whom and timestamped) that was performed. This will keep a rolling log of maintenance events; this could be placed under IMA/EVM control.

Calibration is another function that is performed while in maintenance state, this can happen at a predefined date/time or after component replacement. Calibration can also happen after an update to system parameters. It is important to ensure that the system is intact from an assurance point of view. Different scenarios can be envisioned for how these systems will be maintained (in the field or local service shop).

We now reflect on the holistic model that was shown in Figure 5-21 where setting evidence on the following nodes and a target node with Accuracy Assurance Score being High:

- Type of Platform = Tamper Protection
- System Board = Fault Enabled
- Microprocessor = Intel_with_TEE
- Key Store = BBRAM
- Firmware image = Secure Boot Encrypted
- Active System Monitor = None
- OS Image = Secure Boot Encrypted
- Cognitive Layer/Services image = Secure Boot Encrypted
- TPM = Present

The following is the result of the model changing its node values to reason about the target node being set to high. Figure 5-41 show the green bars as being observations and the shaded box on the upper left corner is the target node. Figure 5-42 is the target posterior probability as a tornado diagram for the total effects on the Accuracy Assurance Score node. The bottom line is the probabilities in the x-axis and y-axis is the value range for each node. We learned that by setting the nine node values for the supported security features, a desired target node level was achieved. These nodes represented the influence that they had on the remaining network, since the information had to flow from the parent nodes to the target node. We can assess the assurance level at the individual layer or at the system level by assigning a different target node value and seeing how the data flow effects the target node. By doing so, we can assess what security features have more impact on the desired assurance level. The tornado diagram in Figure 5-42 will change accordingly to provide a visual representation of the effects of the settings. Depending on the environment the autonomous robotic system is operating in and what data is being processed, this model can be adjusted accordingly to provide an assurance level. If the desired level is not met, security features will need

to be incorporated to achieve higher assurance levels. This model also provides a reach back into the supplier/vendor organization that contributes to the assurance level of the component or system.

We have defined an internal assurance model for autonomous robotic systems where security features provide an assessment for the level of security. This knowledge helps to reason about the different threats and countermeasures. Some of the concepts are new and have been adapted to old methodologies that have been proven in the security world.

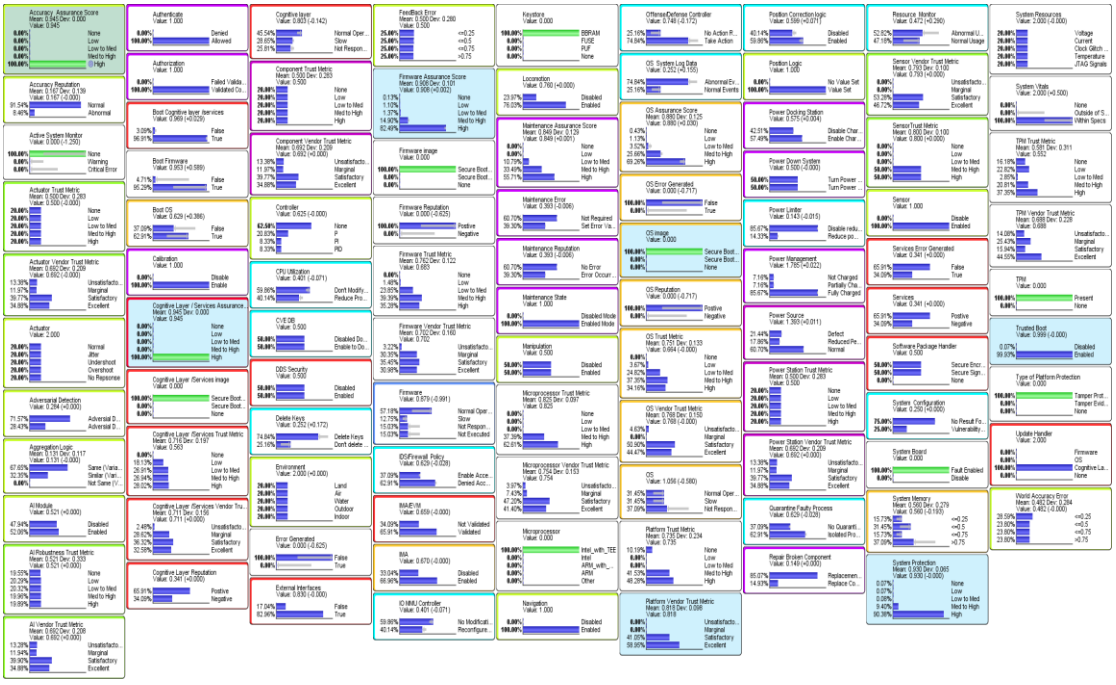


Figure 5-41: BN node values when target and observation nodes are set

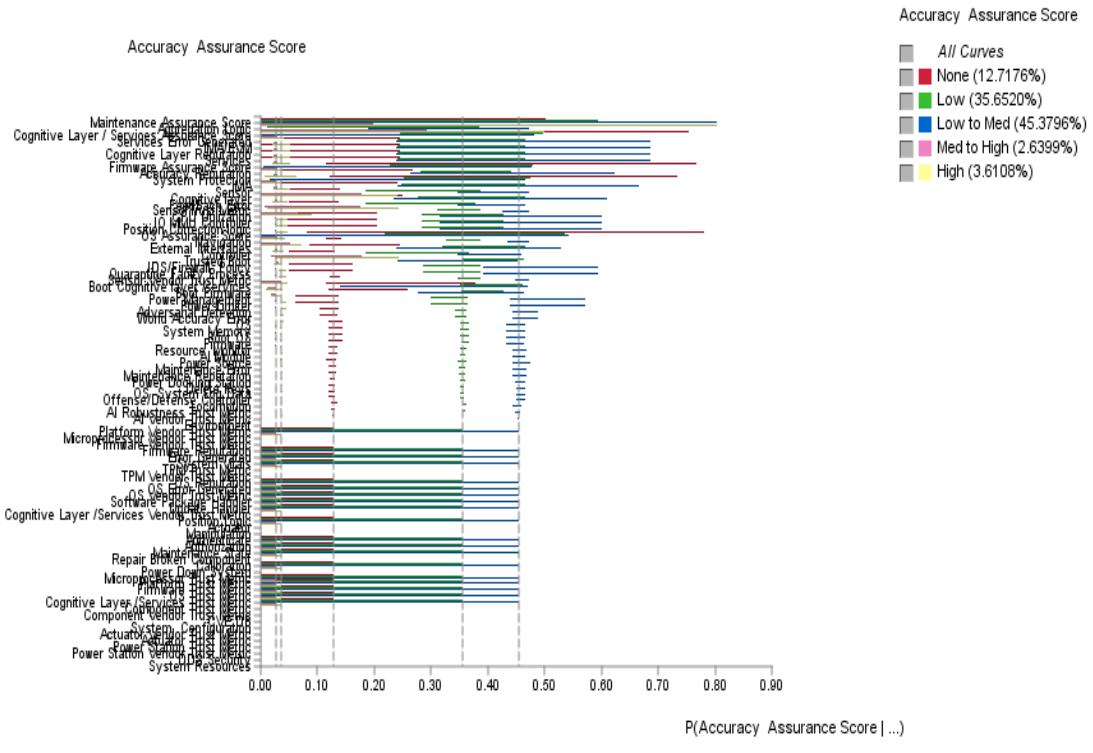


Figure 5-42: Posterior Probability Analysis for the Accuracy Assurance score being set to high

5.5 Conclusion

We have defined several metrics that have incorporated not only integrity, but also the monetary value of loss from the damage incurred and the reward of an adversary attack. A Bayesian Network has been created to represent the assurance scores using these new trust metrics at each layer covering system, hardware, software, Al robustness, and supply chain security features. This holistic architecture provides a base for an internal assurance trust model in autonomous robotic systems. Al algorithms are becoming relevant within autonomous systems and as part of the system security measures must be in place to deter threats from adversarial attacks. AI robustness is a new concept, but research is pointing toward distance measures to provided/certify assurance from threats. By breaking the security model into two parts, internal and external, this provides the capability to reason about the interactions and analyze the resource interactions as casual inference. This approach can be leveraged for how processes may affect the overall system at a task-by-task level. An offense/defense controller is added as another layer to address abnormal behaviors and can be reasoned from a holistic point of view. This resiliency enables an autonomous system to potentially still function while under attack or defend itself from attacks.

A full BN model was presented, but the firmware portion of the model was described in detail. The results provided show the differences between the assurance scores. By setting evidence on the nine nodes to high, a high assurance score was achieved by the target node. The sensitivity diagram gave a representation of the distributions. A full BN model tornado diagram was shown when each layer assurance score was set to high. By creating a full BN, the questions posed earlier can now be

addressed. By creating an assurance score at each layer, the robotic system can determine its security posture before interacting with an external entity. The trust metrics defined in section 3 consider levels of damage and exploit reward, so if the supported security features meet the highest levels, the attackers will incur a higher cost to achieve an exploit with regard to both time and money. The offense/defense controller can take the appropriate actions if the correct controls and data are provided to plan and act accordingly. Other questions may arise from this paper's approach compared to traditional computer security and the reliability of a system.

A traditional computer system is mostly protected in a physical structure such as buildings and under system management whereas, an autonomous robot is in an environment which must sense, plan and act accordingly. As discussed in the above section, the accuracy layer is germane to a robotic system and this is quite different from a traditional computer system. This paper presented the concept of security layers, integrity metrics, offense/defense controller, AI Robustness and assurance scores related to system, hardware, software, cognitive and supply chain vendors. This is different to a reliability study where components are verified by mean time failures. Another element that is being discussed in the industry are ethics and how well behaved or provable AI systems are. In order to have the ethics discussion, autonomous systems must have integrity and assurance at the security level before behavioral policies can be followed. This BN model can be leveraged to address the ethical questions.

In the next chapter, we propose that the same scoring system can determine the security posture when external dynamics are influencing the internal state of the system. This concept is discussed where a Dynamic Bayesian Network model is presented related to a defenders' attack chain.

CHAPTER 6 

6 FUTURE WORK

Dynamic Bayesian Network

We have presented the static model above and would like to address the dynamic model in our future work. This chapter addresses the dynamic model using a Dynamic Bayesian Network (DBN). By enabling a dynamic model that can react in real-time, several questions can now be asked about the security posture of an autonomous robot system using a DBN, but most importantly, the robotic system can act on the knowledge it has from an internal point of view. Being cognizant of this knowledge, the robot can reason about the actions it can take to provide the highest level of assurance.

A Dynamic Bayesian Network (DBN) is an extension to a BN structure that is repetitive, time sliced, and used for modeling dynamic systems in state space. A DBN is defined to be a pair of Bayesian Networks (B0 ,B→), where B0 represents the initial distribution P(Z0), and B→ is a two-time-slice Bayesian Network, which defines the transition distribution P(Zt+1|Zt) [182]. The set Zt is commonly divided into two sets: Xt (hidden states) and Yt (observed). The result of unrolling the 2TBN is a DBN joint distribution shown in equation (6-1), where $P(x_t|x_{t-1})$ defines the time dependent between states, $P(y_t|x_t)$ defines the observed node dependency related to the time slice t, and $P(x_0)$ is the initial probability distribution in the time series.

$$P(x_1 \dots x_t, y_1 \dots y_t) = \prod_{t=1}^{t-1} P(x_t | x_{t-1}) \prod_{t=0}^{t-1} P(y_t | x_t)\ P(x_0) \quad (6\text{-}1)$$

An example of a DBN is shown in Figure 6-1 where three variables are represented in a time series [183].

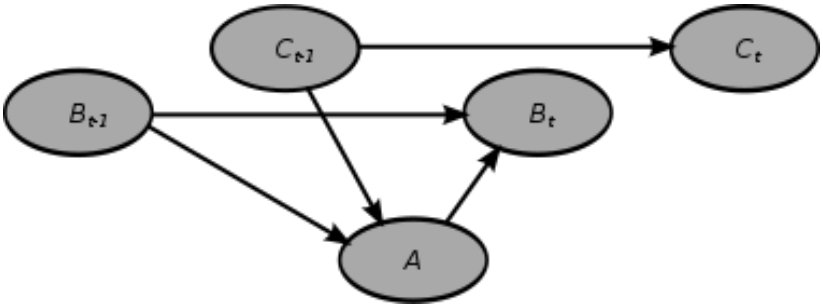


Figure 6-1: An example of a three variable DBN

To address why DBN was chosen over other state space techniques we compared it to the Hidden Markov Model (HMM) and the Kalman Filter Model (KFM). A DBN provides a more generalized graph structure, which allows hidden states in terms of random variables where joint probabilities can be obtained at each time step by unrolling the DBN. An HMM is different from a DBN, because a single random variable defines the state space and in the case of KFM, all CPD must be gaussian distributions. This is only addressing the non-extension version of KF. Both HMM and KF use only chain structures.

By extending the BN model that was described above, we consider the temporal aspect of the problem where offense/defense action time events are related. We first define the notation of kill chains that an adversary will take to exploit the system from both cyber and physical perspectives. The top path is the cyber kill chain with following steps and the bottom describes the physical view as shown in Figure 6-2 [184]. Both

cyber and physical kill chains start with reconnaissance. This first step is taken to understand how the system or component may work.

During reconnaissance, knowledge can be acquired through obtaining schematics, part lists, or monitoring inputs and outputs. Based on the reconnaissance step an adversary will construct one or more weapons to expose the vulnerabilities. These weapons may come in a myriad of forms such as malware, sniffing attacks, and brute force attacks to name a few. Once executed an observation, or C2 step, is used to determine if the attack was successful and at what cost, or did it fail to achieve its goal.

In the physical kill chain, the steps are similar. Instead of a purely software-based attack, signals on buses at the execution and observation steps may be probed, perturbed, or injected to perform the exploit. The results of the combined steps are instantaneous; therefore, an adversary can determine what works or does not in this repeated cycle. Since this type of attack requires physically interacting with the system being attacked, the hardware may be damaged. Therefore, having a number of units helps, which will also ensure same behavior across units. As autonomous robots become commodity items, it is not unreasonable to expect nefarious actors to stage this type of attack.

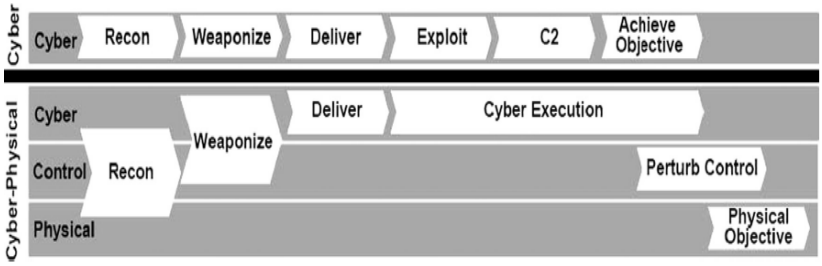


Figure 6-2: Cyber and physical kill chains

To counter an attacker's threats there are countermeasures that a defender creates. The countermeasures can be acquired by white hat attackers (ethical hackers), vulnerability analysis, or by existing exploits. A defender's lifecycle is shown in Figure 6-3 [185]. Prevention indicates basic tools (like firewalls/ Intrusion Detection Systems (IDS) and Endpoint Segmentation). Detection uses automated methods like machine learning to build knowledge. Prediction uses machine learning to understand the anomalies. Lastly, containment takes the decision made from prediction and adds controls. The defender lifecycle is a continuous flow as more knowledge is acquired about the dynamic network traffic.

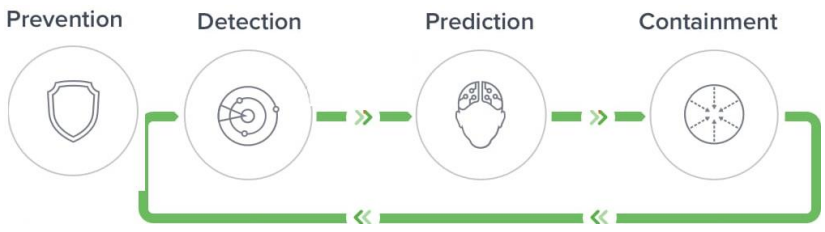
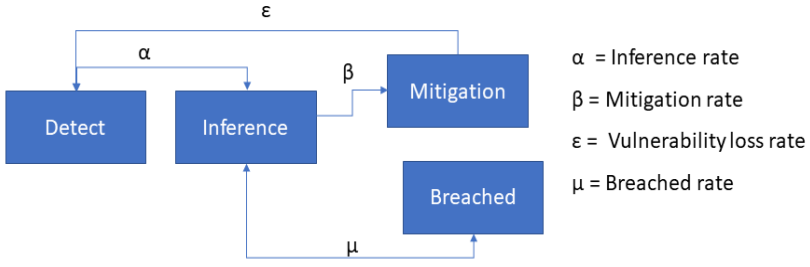


Figure 6-3: A Defender's Lifecycle

From the above, we define a defender's kill chain where detect, inference, mitigation, and breach are steps for countering a threat in a continuous flow. Tying this back to the BN in Figure 5-21, the nodes/layers defined a number of security features for detecting threats. However, these features were dependent on system support. The inference is connected to the cognitive layer, which includes the offense/defense set of features and how mitigation can take part in the protection of the system. The internal security assurance model can change accordingly to an external request; this will help with the mitigation strategy if and when a threat is detected. Again, depending on the security features of a system, breaches may occur over time. The defender's simple block diagram and transition matrix are shown in Figure 6-4 [185] [186].

To simplify the diagrams alpha, beta, epsilon, and mu define the inference rate, mitigation rate, vulnerability loss rate, and breach rate respectively. The transition from detecting a threat to understanding what the threat is/means is the inference rate in time slices. The transition from inference (knowing what the threat is) to migration (handling it with countermeasures) is the migration rate in time slices. The transition from inference (knowing what the threat is) to determining that no migration strategy is available is the breach rate. This assumes that security features are limited. The vulnerability loss rate is the ratio of vulnerabilities over the total number of threats detected/mitigated. Using the generic equation for differential equations provides a mathematical understanding of the change for each of the transitions where alpha, beta, epsilon, and mu are the functions related to equation $f(x) = dy/dx$. To simplify the transition table the values are numeric.



Transition Matrix

	D	I	M	B
D	1-α	α	0	0
I	0	1-β-μ	β	μ
M	ε	0	1-ε	0
B	0	0	0	1

Figure 6-4: Defender's chain and transitions

A simple model helps to explain the need for DBNs related to Autonomous Robotic Systems (ARSs). The transition matrix defines the relationship between the different blocks and interactions. A DBN and associate set of nodes are shown in Figure

6-5 and Figure 6-6. In Figure 6-5 ARS (t) and (t1) are the time steps that get propagated over time events.

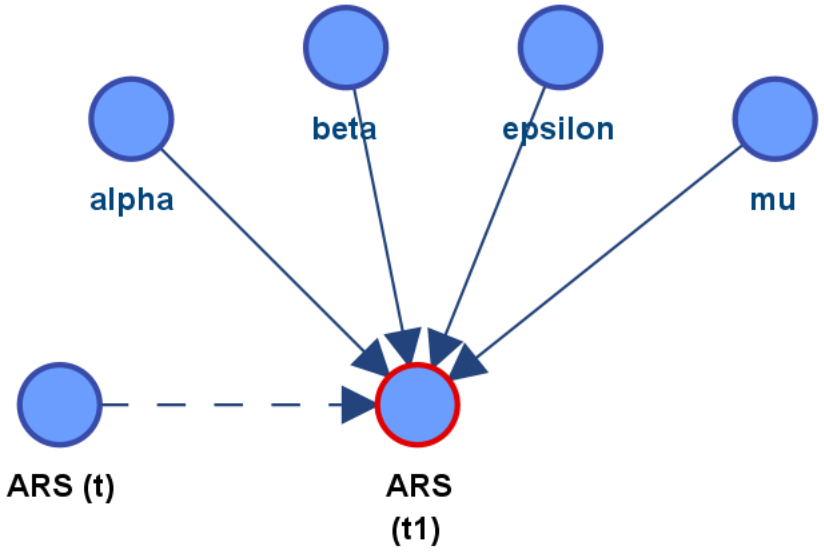


Figure 6-5: A Dynamic Bayesian Network for ARS

Figure 6-6 shows the DBN nodes and their initial values.

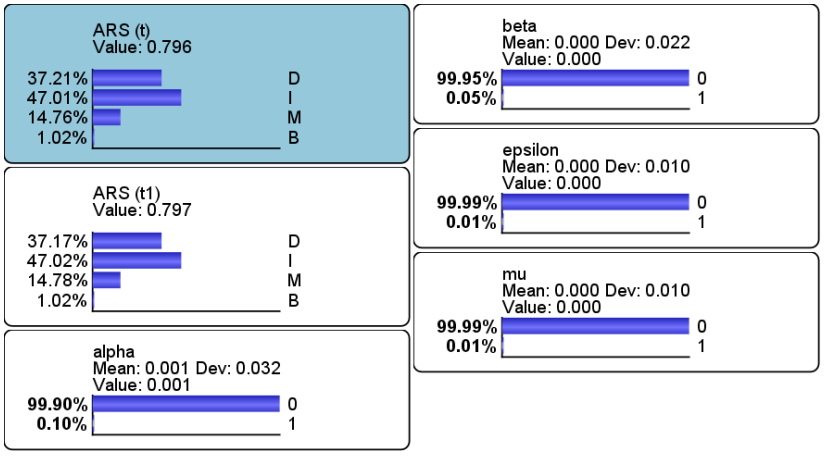


Figure 6-6: A set of nodes that define the different rates

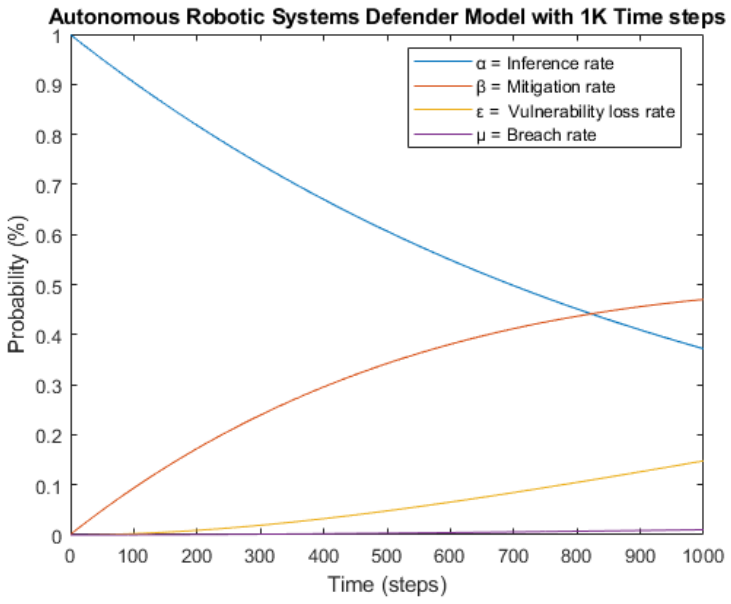


Figure 6-7: DBN run for 1K time steps

When the DBN is run over one thousand-time steps as shown in Figure 6-7, the number of vulnerabilities mitigated starts to increase as shown by the red curve.

As the number of time steps increases as shown in Figure 6-8, the number of breaches increases. This is intuitive, but still makes the point that over time when confronted by an adversary that has the means (resources and money), the systems will eventually be broken.

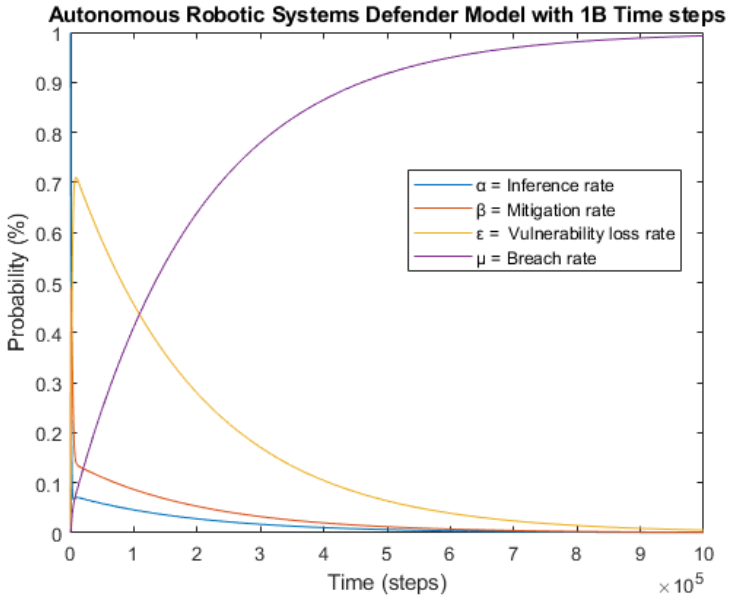


Figure 6-8: DBN run at longer time steps

An expanded view of a DBN is shown in Figure 6-9 where four different time events are shown to illustrate that taking Figure 6-5 model above can be expanded into a more meaningful representation of a DBN. The DBN considers the temporal events such as detecting a threat, determining what actions should take place, what mitigation

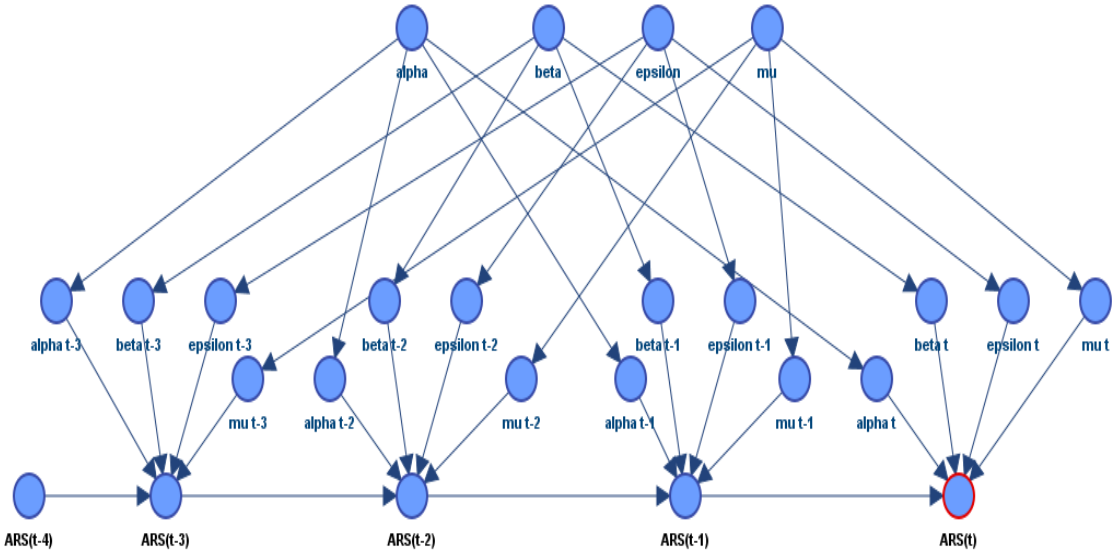


Figure 6-9: DBN with 4-time events

mechanism is best for the threat, documenting threat records, and determining if a breach has taken place.

CHAPTER 7 

7 CONCLUSION

The goal of this research was to create an internal assurance model for autonomous robotic systems using a Bayesian Network that would assess threats and provide resiliency depending on the platform supported security features. The internal assurance model provides a holistic security view into the system, hardware, software, Cognitive/AI, and supplier layers along with their associated trust metrics that were derived from standards and well-known sources. The trust metrics also take into account the adversary's reward and damage costs for potential threats.

To accomplish the research goal, a number of questions had to be addressed and challenged. We began by understanding the autonomous robotic system and the current offerings related to security. Since our research determined that ROS was a leading middleware/framework used by a number of industries, we focused on ROS 2 for its support of DDS.

The question raised regarding ROS 2 supporting DDS security: What system elements were exposed or vulnerable in a robotic system? Chapter 2 discussed the system security review and explained the different types of threats that are exposed. The vulnerability analysis used a taxonomy for both the hardware and software components. An advanced set of threats were described related to hardware and the cognitive layer using ML. It was determined that the DDS Security Standard does not

cover these advanced types of threats and leaves a number of areas exposed to threats. DDS covers the data in motion (DIM) protection by encrypting messages between nodes, which is only a part of the overall security of a robotic system.

The next set of questions related to ROS 2 were related to performance vs security, the real-time constraints, and the difference between using DDS security features or different cryptographic algorithms. The performance versus security model showed that adding security features to protected data resulted in a performance degradation. This performance degradation was evident when configurations ranged from no security to full security configuration. Table 2-4 showed a comparison of the measurements performed when security is fully enabled versus not enabled at all. In revealing the considerable difference between the security levels effect on performance, it was concluded that further analysis is required. Algorithm and key size made little difference compared to data protection features.

TABLE 7-1: SECURITY MEASUREMENT COMPARISON

Security Enabled	Latency (average µsec)	Throughput (average packets/sec)	Speed (average Mbps)
Plain	260	70772	35669
Full	1363	14382	72485

Can security be applied to everything or nothing in a robotic system. The ROS 2 security model is flexible in segmenting domains and participants to topics, so inherently, ROS 2 allows the ability to be selective about what security techniques are applied to various portions of the robotic system. The question at hand can be addressed using two-level enforcement for access control, such as the Governance and

Permissions policies. Some challenges were experienced in answering the performance vs security question. Since the protocol is using RTPS as discussed in Chapter 2, we needed a way to configure security and block size in a consistent manner for repeatability. We first looked at Linux tools like Perf and Tops, but these did not support the security functions or address the publisher/subscribe protocol. One tool that was available is called Message Queuing Telemetry Transport (MQTT) which is a lightweight broker-based software stack. However, it did not cover our needs, nor did it match the RTPS protocol. At the time of the research, we found that RTI developed a RTI Perftest for performance measurements on DDS messages with enabled security, but this was in early stages of development. We leveraged the software to the best of our ability and were able to successfully compile and run the software against our version of DDS. Not all the security features were initially supported, but shortly the RTI developer was able to support our needs and respond to our support questions in a timely fashion.

Questions arise related to DDS security and if it provided adequate security for the system. At the software stack: what components provided the least protection, but allowed an adversary complete control of the system? In acquiring knowledge of ROS 2/DDS, we were able to develop several exploits that were covered in Chapter 3. Two attack vectors related to ROS 2/DDS security were presented and we have identified four different use case scenarios that are plausible. In each use case, the adversary was able to obtain control of data or direct manipulation of the platform using the modified OpenSSL library and/or the configuration file used for credential masquerading. Several mitigation techniques were covered, but even when files have been downloaded and checked with a hash, that one-time check does not provide the safeguards against

ongoing threats. We concluded that using a TPM with IMA/EVM at the OS layer can help mitigate against these threats, since runtime and offline security set of features protect the files. Enabling a TPM within a robotic platform and performing attestation are new concepts that need to be extended beyond the traditional mechanism of trusted boot in traditional computer platforms. Several challenges were met during this research that were related to understanding the inner workings of OpenSSL and how the DDS security plug-in implementation was using the library calls. The DDS security plug-ins were a black box; in order to understand how the plug-ins were being used, we had to investigate the DDS Security Specification. This led to understanding the cryptographic algorithms being used at the appropriate time. We modified the OpenSSL library and peppered the different layers with hex dump routines until we understood the specific layers that were being used. We took the hex dump routine outputs and verified them against known good vectors to validate the correct sequence of logic. This drove the use of Wireshark with RTPS support, which allowed us to inspect the network traffic. It was difficult to understand how the puzzle was put together, but this was overcome once patterns became clear. The unraveling exposed sensitive data or the golden nuggets that were sought. Another set of challenges were related to validating this concept on a clean system with the change in Linux between 16 and 18. This was overcome by a large amount of testing and understanding the difference between the two versions related to the crypto libraries.

Since DDS security is only protecting data in motion, and we provided evidence of exploits against it, does a solution exists that would cover the rest of the system? What trust metrics are needed to define a holistic security architecture? In order to answer these questions, we surveyed the trust metric space to determine if a set of

security metrics were well defined and covered a complete robotic system. The results were covered in Chapter 4. A mind map of a robot system broke down the different layers into system, hardware, software, cognitive layer, and supplier chain to provide a holistic security view. Our findings showed that at the system level trust metrics are difficult and complex. Several research papers scaled the problem to a small set of components or just a specific area of a system. Table 4-4 is a summary of the findings that cover a holistic system trust model. The largest challenge of our investigation was sifting through the large amount of information, drilling down on the details for the specific focus areas, and structuring the content into a document.

TABLE 7-2: SUMMARY OF TRUST METRICS

Trust Level	Trust Metric Recommendation	Values	Comment
System	EAL 1 to 5	[0,1]	Security and Safety
Hardware	Hardware Component	[0,1]	
Software	Base Impact	[0,1]	
AI Robustness	Distance Metric	[0,1]	
Supply Chain	NG Scorecard	[0,1]	

Can a BN be used to support all the trust metrics and represent an internal security model. What are the benefits from creating an internal/external security model? Our findings confirmed that using the Bayesian Inference is the correct choice to build on, since it provides several benefits for overcoming uncertainties for a complex system like an autonomous robotic system.

To address these questions, in Chapter 5 a Bayesian Network was created to represent the assurance scores using the new trust metrics at each layer covering system, hardware, software, Al robustness, and supply chain security features. We have defined several metrics that have incorporated not only integrity, but also the monetary value of loss from the damage incurred and the reward of an adversary attack. This holistic architecture provides a base for an internal assurance trust model in autonomous robotic

systems. Al algorithms are becoming relevant within autonomous systems and as part of the system, security measures must be put in place to deter threats from adversarial attacks. AI robustness is a new concept, but research is pointing toward identifying certified boundaries where perturbations can be detected. These certified regions provide a means to assign trust metrics for measuring the strength of AI models. The training data must also be protected from poison attacks, since this needs to be distributed for validation purposes.

By breaking the security model into two parts, internal and external, this provides the capability to reason about the interactions and analyze the resource interactions as casual inference. This approach can be leveraged for how processes may affect the overall system at a task-by-task level. An offense/defense controller is added as another layer for how abnormal behaviors can be reasoned from a holistic view. This resiliency enables an autonomous system to potentially still function while under attack or defend itself from attacks.

A full BN model was presented, but only the firmware portion of the model was described in detail to avoid redundancy. The results provided showed the differences between the assurance scores and what evidence is needed to achieve the highest score. The sensitivity diagram gave a representation of the distributions. A full BN model tornado diagram was shown when each layer assurance score was set to high. By creating a full BN, the questions posed earlier can now be addressed. Further, creating an assurance score at each layer, the robotic system can determine its security posture before interacting with an external entity. The trust metrics defined in Chapter 5 section 3 consider levels of damage and exploit reward, so if the supported security features meet the highest levels the attackers will incur cost (both time and money). The

offense/defense controller can take the appropriate actions if the correct controls and data are provided to plan and act accordingly. Other questions may arise from this chapter's approach compared to traditional computer security. The difference is that a traditional computer system is mostly protected in a physical structure like buildings and under system management, whereas an autonomous robot is exposed to many elements in the environment. The autonomous robotic system must sense, plan, and act accordingly within this environment to perform the tasks or jobs given.

There were a number of challenges with this research goal, the first was to define the trust metrics and the appropriate values. Defining the trust metrics and appropriate values took several iterations of calculations, during which MATLAB helped with the visualization of the results. The second challenge was finding a tool that was rich in features. Three such tools we reviewed were GeNIe, SamIam, and BayesiaLab. The BayeiaLab tool was feature rich over the others and had a number of positive reviews from other researchers. The third challenge was building the full BN; we begin by creating the individual standalone models to understand how things worked. This led to tying the individual layers together so that we can see how a score can be propagated into the next layer. The rule in constructing BN is to keep the parent nodes down to a reasonable number as the CPT tables get large quickly, which this took several iterations to reduce the size of the tables. The fourth challenge was understanding the results related to causal inference when the model was stitched together. In a simple set of five nodes, one can understand the different cases of inference as explained in the chapter, but when the number of nodes increases it becomes more complex. To help alleviate the complexity issue, the tool allowed

defining a target node and setting the evidence on parent nodes to see the different observations occurring at each step.

The last question is "can a static model be extended into a dynamic model"? To address this question, Chapter 6 discussed anticipated future research into using a DBN for this purpose.

Since both attacker and defender actions are time dependent, using a DBN addressed the temporal relationships. We do see a path forward on this front, which will enable an autonomous robotic system to have a real-time internal trust assurance model for interacting with external entities. This will allow a finer security control model of the system and support resiliency to threats; this all depends on the platforms security features.

From the evidence supporting this research we can concluded the following:

1) Autonomous robotic system security is still a new field and conventional security is not the same.
2) ROS2/DDS security is a step forward but doesn't address the holistic approach as covered in this research.
3) Researchers have created stovepipe solutions, assessment methods, and techniques which only address specific parts of the overall system. These leave gaps in securing the overall system.
4) The contribution of this research has created a new set of metrics that provide a harmonized assessment method for a holistic security system model.
5) BN as an assessment technique was the right choice and this was proven by existing research. We simulated a test case using BN that showed good

results in the static model scenario. It was shown that starting from a static model and moving to a dynamic one could be done.

6) Our paper titled “Robot Operating System 2: The need for a holistic security approach to robotic architectures” has received considerable activity in this new autonomous robotic system security field. The paper titled “Credential Masquerading and OpenSSL Spy: Exploring ROS 2 using DDS security” has also been noticed.

As a closing remark, I am eager to move forward in advancing the security standards of autonomous robotic systems. It is my hope that the standards created as a result of this research be incorporated into the security assessment methodologies.

BIBLIOGRAPHY

[1] T. Denning, C. Matuszek, K. Koscher, J. R. Smith, and T. Kohno, “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, [Online]. Available: http://dl.acm.org/citation.cfm?id=1620564.

[2] V. DiLuoffo, W. R. Michalson, and B. Sunar, “Robot Operating System 2: The need for a holistic security approach to robotic architectures,” *International Journal of Advanced Robotic Systems*, vol. 15, no. 3, p. 1729881418770011, May 2018, doi: 10.1177/1729881418770011.

[3] V. DiLuoffo, W. R. Michalson, and B. Sunar, “Credential Masquerading and OpenSSL Spy: Exploring ROS 2 using DDS security,” *arXiv:1904.09179 [cs]*, Apr. 2019, Accessed: Jul. 07, 2019. [Online]. Available: http://arxiv.org/abs/1904.09179.

[4] “Autonomous Robot Market 2020 Growth Trends by Manufacturers, Regions, Type and Application, Forecast 2026 – Owned.” https://primefeed.in/news/42233/autonomous-robot-market-2020-growth-trends-by-manufacturers-regions-type-and-application-forecast-2026/ (accessed Oct. 15, 2020).

[5] “Acutronic Robotics fails to find funding for H-ROS for robot hardware,” *The Robot Report*, Jul. 23, 2019. https://www.therobotreport.com/acutronic-robotics-h-ros-robot-hardware-fails/ (accessed Oct. 15, 2020).

[6] “Robot operating system - global ROS-based robot market 2024,” *Statista*. https://www.statista.com/statistics/1084823/global-ros-based-robot-market-volume/ (accessed Oct. 15, 2020).

[7] N. DeMarinis, S. Tellex, V. Kemerlis, G. Konidaris, and R. Fonseca, “Scanning the Internet for ROS: A View of Security in Robotics Research,” *arXiv:1808.03322 [cs]*, Jul. 2018, Accessed: Mar. 16, 2019. [Online]. Available: http://arxiv.org/abs/1808.03322.

[8] Pardo-Castellote, Gerardo, “Distributed Data Management,” *http://www.omg.org/news/meetings/workshops/Real-time_WS_Final_Presentations_2008/Tutorials/00-T1_Pardo-Castellote.pdf*. http://www.omg.org/news/meetings/workshops/Real-time_WS_Final_Presentations_2008/Tutorials/00-T1_Pardo-Castellote.pdf.

[9] M. Arguedas, “ROS 2 raodmap,” *GitHub*. https://github.com/ros2/ros2.

[10] Davide Quarta, Marcello Pogliani, Mario Polino, Andrea M. Zanchettin, and Stefano Zanero, and Federico Maggi, “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.

[11] A. Gholami and E. Laure, “Security and Privacy of Sensitive Data in Cloud Computing : A Survey of Recent Developments,” Dec. 2015, pp. 131–150, doi: 10.5121/csit.2015.51611.

[12] R. P. Jover, “LTE security, protocol exploits and location tracking experimentation with low-cost software radio,” *arXiv:1607.05171 [cs]*, Jul. 2016, Accessed: Feb. 15, 2018. [Online]. Available: http://arxiv.org/abs/1607.05171.

[13] D. Thomas, “ROS 2 middleware interface.” http://design.ros2.org/articles/ros_middleware_interface.html.
[14] H. Delfs and H. Knebl, *Introduction to Cryptography: Principles and Applications*. Springer Science & Business Media, 2007.
[15] M. Dworkin, “Recommendation for block cipher modes of operation. methods and techniques - nistspecialpublication800-38a.pdf,” 2001. Accessed: Apr. 23, 2017. [Online]. Available: http://http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf.
[16] M. Dworkin, “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.
[17] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Pearson Education, 2016.
[18] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.
[19] N. Ferguson, B. Schneier, and T. Kohno, *Cryptography Engineering: Design Principles and Practical Applications*. John Wiley & Sons, 2011.
[20] D. Cooper, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.” https://tools.ietf.org/html/rfc5280.
[21] “About the DDS Security Specification Version 1.0.” https://www.omg.org/spec/DDS-SECURITY/1.0 (accessed Oct. 06, 2020).
[22] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[23] Information Technology Laboratory, “Digital Signature Standard (DSS),” National Institute of Standards and Technology, Jul. 2013. doi: 10.6028/NIST.FIPS.186-4.
[24] E. Rescorla, “Diffie-Hellman Key Agreement Method.” https://tools.ietf.org/html/rfc2631.
[25] M. Lepinski and S. Kent, “Additional Diffie-Hellman Groups for Use with IETF Standards.” https://tools.ietf.org/html/rfc5114.
[26] O. M. G. Specification, “The Real-time Publish-Subscribe Protocol (RTPS) DDS Interoperability Wire Protocol Specification,” *Object Management Group pct/07-08-04*, 2007.
[27] Daniel J. Bernstein and Tanja Lange, “SafeCurves: choosing safe curves for elliptic-curve cryptography.” http://safecurves.cr.yp.to/.
[28] N. Ferguson, “Authentication weaknesses in GCM,” *Comments submitted to NIST Modes of Operation Process*, 2005, Accessed: Apr. 18, 2017. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf.

[29] Hanno Böck, Aaron Zauner, Sean, Devlin, , Juraj, Somorovsky, and , Philipp Jovanovic, “Nonce Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS.” https://www.usenix.org/sites/default/files/conference/protected-files/woot16_slides_bock.pdf.

[30] J. Mattsson and M. Westerlund, “Authentication Key Recovery on Galois/Counter Mode (GCM),” in *International Conference on Cryptology in Africa*, 2016, pp. 127–143, Accessed: Apr. 18, 2017. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-31517-1_7.

[31] S. Gueron, A. Langley, and Y. Lindell, “AES-GCM-SIV: Specification and Analysis,” 2017, Accessed: Apr. 18, 2017. [Online]. Available: http://eprint.iacr.org/2017/168.pdf.

[32] Yu Sasaki and Kazumaro Aoki, “Finding Preimages in Full MD5 Faster than Exhaustive Search.pdf.” https://www.iacr.org/archive/eurocrypt2009/54790136/54790136.pdf.

[33] S. M. Bellovin and F. Gont, “Defending Against Sequence Number Attacks.” https://tools.ietf.org/html/rfc6528.

[34] G. Irazoqui, T. Eisenbarth, and B. Sunar, “Cross processor cache attacks,” in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, 2016, pp. 353–364, Accessed: Apr. 25, 2017. [Online]. Available: http://dl.acm.org/citation.cfm?id=2897867.

[35] B. Biggio *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, Accessed: Apr. 18, 2017. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-40994-3_25.

[36] B. Biggio, B. Nelson, and P. Laskov, “Poisoning Attacks against Support Vector Machines,” *arXiv:1206.6389 [cs, stat]*, Jun. 2012, Accessed: Apr. 18, 2017. [Online]. Available: http://arxiv.org/abs/1206.6389.

[37] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical Black-Box Attacks against Machine Learning,” 2017, pp. 506–519, doi: 10.1145/3052973.3053009.

[38] K. D. Akdemir, D. Karakoyunlu, T. Padir, and B. Sunar, “An emerging threat: eve meets a robot,” in *International Conference on Trusted Systems*, 2010, pp. 271–289, Accessed: Apr. 21, 2017. [Online]. Available: http://link.springer.com/10.1007%2F978-3-642-25283-9_18.

[39] R. Jiménez-Naharro, F. Gómez-Bravo, J. Medina-García, M. Sánchez-Raya, and J. Gómez-Galán, “A Smart Sensor for Defending against Clock Glitching Attacks on the I2C Protocol in Robotic Applications,” *Sensors*, vol. 17, no. 4, p. 677, Mar. 2017, doi: 10.3390/s17040677.

[40] Y. Shoukry and P. Tabuada, “Event-triggered projected Luenberger observer for linear systems under sparse sensor attacks,” in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, 2014, pp. 3548–3553, Accessed: Apr. 21, 2017. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7039940/.

[41] J. Petit and S. E. Shladover, “Potential Cyberattacks on Automated Vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, May 2018.

[42] S. Sun, Z. Yan, and J. Zambreno, “Experiments in attacking FPGA-based embedded systems using differential power analysis,” in *Electro/Information Technology, 2008. EIT 2008. IEEE International Conference on*, 2008, pp. 7–12, Accessed: Apr. 25, 2017. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/4554259/.

[43] E. De Mulder *et al.*, “Electromagnetic analysis attack on an FPGA implementation of an Elliptic curve cryptosystem,” in *Computer as a Tool, 2005. EUROCON 2005. The International Conference on*, 2005, vol. 2, pp. 1879–1882, Accessed: Apr. 25, 2017. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/1630348/.

[44] “DDSI-RTPS.” http://www.omg.org/spec/DDSI-RTPS/.

[45] *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.

[46] Object Management Group Specification, “Data Distribution Service (DDS),” 2007.

[47] T. Bonaci, J. Herron, T. Yusuf, J. Yan, T. Kohno, and H. J. Chizeck, “To make a robot secure: An experimental analysis of cyber security threats against teleoperated surgical robots,” *arXiv preprint arXiv:1504.04339*, 2015, Accessed: Apr. 08, 2017. [Online]. Available: https://arxiv.org/abs/1504.04339.

[48] S. Gil, S. Kumar, M. Mazumder, D. Katabi, and D. Rus, “Guaranteeing spoof-resilient multi-robot networks,” *Autonomous Robots*, Feb. 2017, doi: 10.1007/s10514-017-9621-5.

[49] A. WEIMERSKIRCH and D. DOMINIC, “Identifying and Analyzing Cybersecurity Threats to Automated Vehicles,” p. 10.

[50] J. Petit and S. E. Shladover, “Potential Cyberattacks on Automated Vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11.

[51] V. DiLuoffo, W. R. Michalson, and B. Sunar, “Robot Operating System 2: The need for a holistic security approach to robotic architectures,” *International Journal of Advanced Robotic Systems*, vol. 15, no. 3, p. 1729881418770011, May 2018, doi: 10.1177/1729881418770011.

[52] RoboticsandAutonomousSystems100(2018)95–107, Contents lists available at ScienceDirect, RoboticsandAutonomousSystems, and journal homepage: www.eFrancisco Martín, Enrique Soriano, JoséM.Cañas, “Quantitative analysis of security in distributed robotic frameworks.”

[53] D. Hu, “SECURITY SERVICES FOR DDS SECURITY PLUGINS Based on Arm TrustZone.” ARM Limited.

[54] D. Goodin, “‘Most serious’ Linux privilege-escalation bug ever is under active exploit (updated),” *Ars Technica*, Oct. 20, 2016. https://arstechnica.com/information-technology/2016/10/most-serious-linux-privilege-escalation-bug-ever-is-under-active-exploit/ (accessed May 04, 2018).

[55] M. C. Long II, “Attack and Defend: Linux Privilege Escalation Techniques of 2016,” p. 20, 2017.

[56] “New Privilege Escalation Flaw Affects Most Linux Distributions,” *The Hacker News*. https://thehackernews.com/2018/10/privilege-escalation-linux.html (accessed Oct. 30, 2018).
[57] “Privacy-enhanced Electronic Mail,” *Wikipedia*. Apr. 23, 2018, Accessed: May 06, 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Privacy-enhanced_Electronic_Mail&oldid=837833336.
[58] Object Management Group Specification, “Data Distribution Service (DDS),” 2007.
[59] O. M. G. Specification, “The Real-time Publish-Subscribe Protocol (RTPS) DDS Interoperability Wire Protocol Specification,” *Object Management Group pct/07-08-04*, 2007.
[60] “DDS Security Specification Version 1.1.” https://www.omg.org/spec/DDS-SECURITY/About-DDS-SECURITY/ (accessed Sep. 29, 2018).
[61] “OpenSSL.” https://www.openssl.org/ (accessed Apr. 29, 2018).
[62] J. Jonsson and B. Kaliski, “Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1.” https://tools.ietf.org/html/rfc3447 (accessed Nov. 12, 2018).
[63] “Elliptic Curve Digital Signature Algorithm,” *Wikipedia*. Mar. 03, 2018, Accessed: May 06, 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Elliptic_Curve_Digital_Signature_Algorithm&oldid=828648744.
[64] “Elliptic-curve Diffie–Hellman,” *Wikipedia*. Apr. 12, 2018, Accessed: May 06, 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Elliptic-curve_Diffie%E2%80%93Hellman&oldid=836070673.
[65] D. A. McGrew, S. Jose, and J. Viega, “The Galois/Counter Mode of Operation (GCM),” p. 43.
[66] “Secure Hash Algorithms,” *Wikipedia*. Apr. 21, 2018, Accessed: May 10, 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Secure_Hash_Algorithms&oldid=837517049.
[67] “HMAC,” *Wikipedia*. Mar. 27, 2018, Accessed: May 10, 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=HMAC&oldid=832688082.
[68] S. Shopova, “Programming with OpenSSL and libcrypto in examples,” p. 40.
[69] *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.
[70] “RunKit + npm: node-aes-gcm.” https://npm.runkit.com/node-aes-gcm (accessed Jun. 02, 2018).
[71] “Root of Trust Definitions and Requirements v1.0.1,” p. 74.
[72] S. Schrecker *et al.*, “Industrial Internet of Things Security Framework,” p. 173.

[73] “Trusted Execution Technology,” *Wikipedia*. Mar. 07, 2018, Accessed: Jun. 15, 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Trusted_Execution_Technology&oldid=829214751.
[74] “Platform Configuration Registers | SpringerLink.” https://link.springer.com/chapter/10.1007/978-1-4302-6584-9_12 (accessed Jun. 15, 2018).
[75] D. Hu, “DDS Security Plugins Internal API specification.” ARM Limited.
[76] Z. Ning, F. Zhang, W. Shi, and W. Shi, “Position Paper: Challenges Towards Securing Hardware-assisted Execution Environments,” 2017, pp. 1–8, doi: 10.1145/3092627.3092633.
[77] N. Zhang, K. Sun, D. Shands, W. Lou, and Y. T. Hou, “TruSpy: Cache Side-Channel Information Leakage from the Secure World on ARM Devices,” 980, 2016. Accessed: Jun. 18, 2018. [Online]. Available: http://eprint.iacr.org/2016/980.
[78] A. Tang, S. Sethumadhavan, and S. Stolfo, “CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management,” p. 19.
[79] M. Lipp, “eu-16-Lipp-ARMageddon-How-Your-Smartphone-CPU-Breaks-Software-Level-Security-And-Privacy-wp.pdf,” *Cache Attacks on ARM Master thesis*. https://www.blackhat.com/docs/eu-16/materials/eu-16-Lipp-ARMageddon-How-Your-Smartphone-CPU-Breaks-Software-Level-Security-And-Privacy-wp.pdf (accessed Jun. 18, 2018).
[80] J. V. Bulck *et al.*, “FORESHADOW: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution,” p. 18.
[81] “New Cold Boot Attack Unlocks Disk Encryption On Nearly All Modern PCs,” *The Hacker News*. https://thehackernews.com/2018/09/cold-boot-attack-encryption.html (accessed Oct. 01, 2018).
[82] “Ubuntu – Details of package ima-evm-utils in xenial.” https://packages.ubuntu.com/xenial/utils/ima-evm-utils (accessed Jun. 18, 2018).
[83] “Integrity Measurement Architecture (IMA) / Wiki / Home.” https://sourceforge.net/p/linux-ima/wiki/Home/ (accessed Jun. 18, 2018).
[84] “IMA Template Management Mechanism.” https://www.kernel.org/doc/Documentation/security/IMA-templates.txt (accessed Jun. 18, 2018).
[85] Y. L. Sun, Z. Han, W. Yu, and K. J. R. Liu, “A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks,” in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, Barcelona, Apr. 2006, pp. 1–13, doi: 10.1109/INFOCOM.2006.154.
[86] F. Azzedin and M. Maheswaran, “Evolving and managing trust in grid computing systems,” in *IEEE CCECE2002. Canadian Conference on Electrical and Computer Engineering. Conference Proceedings (Cat. No.02CH37373)*, May 2002, vol. 3, pp. 1424–1429 vol.3, doi: 10.1109/CCECE.2002.1012962.
[87] C. Castelfranchi and R. Falcone, “Socio-Cognitive Theory of Trust,” p. 32.

[88] S. Kate Devitt, “Trustworthiness of Autonomous Systems,” in *Foundations of Trusted Autonomy*, H. A. Abbass, J. Scholz, and D. J. Reid, Eds. Cham: Springer International Publishing, 2018, pp. 161–184.
[89] D. Henshel, M. G. Cains, B. Hoffman, and T. Kelley, “Trust as a Human Factor in Holistic Cyber Security Risk Assessment,” *Procedia Manufacturing*, vol. 3, pp. 1117–1124, 2015, doi: 10.1016/j.promfg.2015.07.186.
[90] Y. Wang, C. Hang, and M. P. Singh, “A Probabilistic Approach for Maintaining Trust Based on Evidence,” *Journal of Artificial Intelligence Research*, vol. 40, pp. 221–267, Jan. 2011, doi: 10.1613/jair.3108.
[91] J. Patel, “A Trust and Reputation Model for Agent-Based Virtual Organisations,” p. 184.
[92] E. Hernandez and D. Wunsch, “Graphical Trust Models for Agent-Based Systems,” *IEEE Potentials*, vol. 37, no. 5, pp. 25–33, Sep. 2018, doi: 10.1109/MPOT.2016.2578966.
[93] W. Liang, G. Xiongtao, and G. Yajun, “Construct a novel reputation-based trust model for P2P E-commerce,” in *The 27th Chinese Control and Decision Conference (2015 CCDC)*, May 2015, pp. 6339–6344, doi: 10.1109/CCDC.2015.7161958.
[94] R. Mittu, D. Sofge, A. Wagner, and W. F. Lawness, Eds., *Robust intelligence and trust in autonomous systems*. New York Heidelberg Dordrecht London: Springer, 2016.
[95] Y. Wang and G. Dai, “PC platform trustworthiness evaluation based on Bayesian Network,” in *The 2nd International Conference on Information Science and Engineering*, Dec. 2010, pp. 6760–6763, doi: 10.1109/ICISE.2010.5691847.
[96] R. Paul, J. Dong, I.-L. Yen, and F. Bastani, “Trustworthiness Assessment Framework for Net-Centric Systems,” in *High Assurance Services Computing*, L.-J. Zhang, R. Paul, and J. Dong, Eds. Boston, MA: Springer US, 2009, pp. 19–44.
[97] R. Ross, M. McEvilley, and J. Carrier Oren, “Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems,” National Institute of Standards and Technology, NIST SP 800-160, Nov. 2016. doi: 10.6028/NIST.SP.800-160.
[98] R. Ross, V. Pillitteri, R. Graubart, D. Bodeau, and R. McQuaid, “Developing Cyber Resilient Systems: A Systems Security Engineering Approach,” National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-160v2, Nov. 2019. doi: 10.6028/NIST.SP.800-160v2.
[99] E. R. Griffor, C. Greer, D. A. Wollman, and M. J. Burns, “Framework for cyber-physical systems: volume 1, overview,” National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 1500-201, Jun. 2017. doi: 10.6028/NIST.SP.1500-201.
[100] E. R. Griffor, C. Greer, D. A. Wollman, and M. J. Burns, “Framework for cyber-physical systems: volume 2, working group reports,” National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 1500-202, Jun. 2017. doi: 10.6028/NIST.SP.1500-202.

[101] S. Shetty, “Cyber Risk Scoring and Mitigation for Resilient Cyber Infrastructure.” https://ciri.illinois.edu/sites/default/files/CIRI_Webinar_Shetty_v2_slides.pdf (accessed Dec. 02, 2019).
[102] D. Henshel, A. Alexeev, M. Cains, V. Agarwal, and B. Hoffman, “Integrating Attackers and Defender into Cyber Security Risk Models,” p. 19, 2018.
[103] J.-H. Cho, P. M. Hurley, and S. Xu, “Metrics and measurement of trustworthy systems,” in *MILCOM 2016 - 2016 IEEE Military Communications Conference*, Baltimore, MD, USA, Nov. 2016, pp. 1237–1242, doi: 10.1109/MILCOM.2016.7795500.
[104] A. M. Dale, “National Information Assurance Partnership (NIAP) Common Criteria Evaluation and Validation Scheme (CCEVS),” Feb. 06, 2019. https://csrc.nist.gov/csrc/media/events/ispab-march-2006-meeting/documents/a_dale-march2006-ispab.pdf (accessed Feb. 05, 2019).
[105] “Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments,” in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, Cambridge, United Kingdom, Jul. 2015, pp. 1361–1362, doi: 10.1109/INDIN.2015.7281933.
[106] C. Schmittner and Z. Ma, “Towards a Framework for Alignment Between Automotive Safety and Security Standards,” in *Computer Safety, Reliability, and Security*, vol. 9338, F. Koornneef and C. van Gulijk, Eds. Cham: Springer International Publishing, 2015, pp. 133–143.
[107] “Automotive Safety Integrity Level,” *Wikipedia*. Jul. 13, 2018, Accessed: Jan. 01, 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Automotive_Safety_Integrity_Level&oldid=850139392.
[108] J.-P. Blanquart, P. Bieber, G. Descargues, E. Hazane, M. Julien, and L. Léonardon, “Similarities and dissimilarities between safety levels and security levels,” p. 9.
[109] “sparkfun-9dof-razor-imu-v30-schematic.” https://www.mouser.com/datasheet/2/813/sparkfun-9dof-razor-imu-v30-schematic-1074856.pdf (accessed Dec. 17, 2019).
[110] “Motion Processing Unit (block diagram).” https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf (accessed Dec. 17, 2019).
[111] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, “Hardware security: Threat models and metrics,” in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, Nov. 2013, pp. 819–823, doi: 10.1109/ICCAD.2013.6691207.
[112] S. M. Wong H. S. Philip Wong and Simon, “Stopping Hardware Trojans in Their Tracks,” *IEEE Spectrum: Technology, Engineering, and Science News*, Jan. 20, 2015. https://spectrum.ieee.org/semiconductors/design/stopping-hardware-trojans-in-their-tracks (accessed Dec. 29, 2018).
[113] B. A. Mozzaquatro, C. Agostinho, D. Goncalves, J. Martins, and R. Jardim-Goncalves, “An Ontology-Based Cybersecurity Framework for the Internet of Things,” *Sensors*, vol. 18, no. 9, p. 3053, Sep. 2018, doi: 10.3390/s18093053.

[114] Thamburu, T. R and Vinitha, A.V, "A Survey on Trust Management Models in Internet of Things Systems," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 7, no. 1, pp. 15–21, Jan. 2017, doi: 10.23956/ijarcsse/V7I1/0115.
[115] B. Liu and S. Cheng, "State Space Model based Trust Evaluation over Wireless Sensor Networks: An Iterative Particle Filter Approach," *The Journal of Engineering*, vol. 1, Apr. 2017, doi: 10.1049/joe.2016.0373.
[116] N. Boudriga, P. N. Marimuthu, and S. J. Habib, "Measurement and security trust in WSNs: a proximity deviation based approach," *Annals of Telecommunications*, vol. 74, no. 5–6, pp. 257–272, Jun. 2019, doi: 10.1007/s12243-018-0675-y.
[117] Adam G Kimura, Steven B Bibyk, Brian P Dupaix, Matthew J Casto, Gregory L Creech, "Metrics for Analyzing Quantifiable Differentiation of Designs with Varying Integrity for Hardware Assurance," *Wright Patterson Air Force Research Labs*, Mar. 2017, [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/1041893.pdf.
[118] A. G. Kimura, "Development of Trust Metrics for Quantifying Design Integrity and Error Implementation Cost." https://etd.ohiolink.edu/!etd.send_file?accession=osu1492607691591962&disposition=inline (accessed Apr. 19, 2019).
[119] "CVSS v3.1 Specification Document," *FIRST — Forum of Incident Response and Security Teams*. https://www.first.org/cvss/v3.1/specification-document.
[120] K. Scarfone and P. Mell, "The Common Configuration Scoring System (CCSS): Metrics for Software Security Configuration Vulnerabilities," p. 42.
[121] E. LeMay, K. Scarfone, and P. Mell, "The Common Misuse Scoring System (CMSS): Metrics for Software Feature Misuse Vulnerabilities," National Institute of Standards and Technology, Gaithersburg, MD, Jul. 2012. doi: 10.6028/NIST.IR.7864.
[122] "CWE - Common Weakness Scoring System (CWSS)," Dec. 27, 2018. https://cwe.mitre.org/cwss/cwss_v1.0.1.html.
[123] P. Cheng, L. Wang, S. Jajodia, and A. Singhal, "Refining CVSS-Based Network Security Metrics by Examining the Base Scores," in *Network Security Metrics*, L. Wang, S. Jajodia, and A. Singhal, Eds. Cham: Springer International Publishing, 2017, pp. 25–52.
[124] M. Frigault, L. Wang, S. Jajodia, and A. Singhal, "Measuring the Overall Network Security by Combining CVSS Scores Based on Attack Graphs and Bayesian Networks," in *Network Security Metrics*, L. Wang, S. Jajodia, and A. Singhal, Eds. Cham: Springer International Publishing, 2017, pp. 1–23.
[125] Peng Xie, J. H. Li, Xinming Ou, Peng Liu, and R. Levy, "Using Bayesian networks for cyber security analysis," in *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, Chicago, IL, USA, Jun. 2010, pp. 211–220, doi: 10.1109/DSN.2010.5544924.
[126] Y. Cheng, J. Deng, J. Li, S. A. DeLoach, A. Singhal, and X. Ou, "Metrics of Security," in *Cyber Defense and Situational Awareness*, vol. 62, A. Kott, C. Wang, and R. F. Erbacher, Eds. Cham: Springer International Publishing, 2014, pp. 263–295.

[127] “The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems Announces Two New Standards Projects.” https://standards.ieee.org/news/2018/p7011_p7012.html (accessed Dec. 20, 2018).
[128] “High-Level Expert Group on Artificial Intelligence,” *Digital Single Market*. https://ec.europa.eu/digital-single-market/en/high-level-expert-group-artificial-intelligence (accessed Dec. 20, 2018).
[129] D. Kang, Y. Sun, D. Hendrycks, T. Brown, and J. Steinhardt, “Testing Robustness Against Unforeseen Adversaries,” *arXiv:1908.08016 [cs, stat]*, Aug. 2019, Accessed: Dec. 14, 2019. [Online]. Available: http://arxiv.org/abs/1908.08016.
[130] “Welcome to the Adversarial Robustness 360 Toolbox — Adversarial Robustness Toolbox 1.0.1 documentation.” https://adversarial-robustness-toolbox.readthedocs.io/en/latest/ (accessed Dec. 14, 2019).
[131] D. H. Siegelmann, “Guaranteeing AI Robustness against Deception (GARD),” p. 39.
[132] D. Heaven, “Why deep-learning AIs are so easy to fool,” *Nature*, vol. 574, no. 7777, Art. no. 7777, Oct. 2019, doi: 10.1038/d41586-019-03013-5.
[133] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, “Adversarial Policies: Attacking Deep Reinforcement Learning,” *arXiv:1905.10615 [cs, stat]*, Feb. 2020, Accessed: Aug. 17, 2020. [Online]. Available: http://arxiv.org/abs/1905.10615.
[134] N. Carlini, G. Katz, C. Barrett, and D. L. Dill, “Provably Minimally-Distorted Adversarial Examples,” *arXiv:1709.10207 [cs]*, Feb. 2018, Accessed: Jun. 22, 2020. [Online]. Available: http://arxiv.org/abs/1709.10207.
[135] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014, Accessed: Apr. 18, 2017. [Online]. Available: https://arxiv.org/abs/1412.6572.
[136] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, “Robustness May Be at Odds with Accuracy,” *arXiv:1805.12152 [cs, stat]*, May 2018, Accessed: Jun. 23, 2019. [Online]. Available: http://arxiv.org/abs/1805.12152.
[137] P. Nakkiran, “Adversarial Robustness May Be at Odds With Simplicity,” *arXiv:1901.00532 [cs, stat]*, Jan. 2019, Accessed: Jun. 23, 2019. [Online]. Available: http://arxiv.org/abs/1901.00532.
[138] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, and Y. Gao, “Is Robustness the Cost of Accuracy? -- A Comprehensive Study on the Robustness of 18 Deep Image Classification Models,” *arXiv:1808.01688 [cs]*, Aug. 2018, Accessed: Jun. 23, 2019. [Online]. Available: http://arxiv.org/abs/1808.01688.
[139] T.-W. Weng *et al.*, “Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach,” *arXiv:1801.10578 [cs, stat]*, Jan. 2018, Accessed: Jun. 23, 2019. [Online]. Available: http://arxiv.org/abs/1801.10578.
[140] M. Hein and M. Andriushchenko, “Formal Guarantees on the Robustness of a Classifier against Adversarial Manipulation,” *arXiv:1705.08475 [cs, stat]*, May 2017, Accessed: Jun. 23, 2019. [Online]. Available: http://arxiv.org/abs/1705.08475.

[141] A. Boopathy, T.-W. Weng, P.-Y. Chen, S. Liu, and L. Daniel, “CNN-Cert: An Efficient Framework for Certifying Robustness of Convolutional Neural Networks,” *arXiv:1811.12395 [cs, stat]*, Nov. 2018, Accessed: Aug. 22, 2020. [Online]. Available: http://arxiv.org/abs/1811.12395.
[142] *IBM/CLEVER-Robustness-Score*. International Business Machines, 2020.
[143] 14:00-17:00, “ISO 28001:2007,” *ISO*. http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/04/56/45654.html (accessed Apr. 28, 2019).
[144] “DJI Drone Security and US Cyber Threats,” *Vericlave*, Mar. 09, 2020. https://www.vericlave.com/dji-drone-security-and-us-cyber-threats/ (accessed Aug. 25, 2020).
[145] I. H. Abbassi *et al.*, “TrojanZero: Switching Activity-Aware Design of Undetectable Hardware Trojans with Zero Power and Area Footprint,” *arXiv:1812.02770 [cs]*, Nov. 2018, Accessed: Aug. 29, 2020. [Online]. Available: http://arxiv.org/abs/1812.02770.
[146] S. H. Russ and J. Gatlin, “Three Ways to Hack a Printed Circuit Board - IEEE Spectrum,” *IEEE Spectrum: Technology, Engineering, and Science News*. https://spectrum.ieee.org/computing/hardware/three-ways-to-hack-a-printed-circuit-board (accessed Aug. 29, 2020).
[147] Dr. T. Herr, W. Loomis, S. Scott, and J. Lee, “Breaking trust: Shades of crisis across an insecure software supply chain,” *Atlantic Council*, Jul. 27, 2020. https://www.atlanticcouncil.org/in-depth-research-reports/report/breaking-trust-shades-of-crisis-across-an-insecure-software-supply-chain/ (accessed Aug. 29, 2020).
[148] J. Eklow, “Managed Security Services: Metrics for Assessing Vendor Risk | onShore,” *onShore Security*, Feb. 28, 2018. https://www.onshore.com/managed-security-services/guest-blog-experts-share-important-metrics-assessing-vendor-risk-third-party-trusts-jeffrey-spetter/ (accessed May 24, 2019).
[149] PECB, “ISO 28000 Supply Chain Security Management Systems.” https://pecb.com/whitepaper/iso-28000-supply-chain-security-management-systems (accessed Aug. 29, 2020).
[150] “IEEE Spectrum: This Car Run...,” p. 6.
[151] A. M. Wyglinski, X. Huang, T. Padir, L. Lai, T. R. Eisenbarth, and K. Venkatasubramanian, “Security of Autonomous Systems Employing Embedded Computing and Sensors,” *IEEE Micro*, vol. 33, no. 1, pp. 80–86, Jan. 2013, doi: 10.1109/MM.2013.18.
[152] “The 5 most used open source software to develop self-driving platforms for ADAS (Advanced Driver Assistance System) | Twitter Tech News,” Aug. 17, 2018. https://twittertechnews.com/software-reviews/the-5-most-used-open-source-software-to-develop-self-driving-platforms-for-adas-autoware-apollo-eb-robinos-eb-robinos-predictor-nvidia-driveworks-and-openpilot/ (accessed Aug. 29, 2020).
[153] “Apex.OS 1.0 now available, brings ROS 2 development to self-driving cars,” *The Robot Report*, Feb. 02, 2020. https://www.therobotreport.com/apex-os-1-0-available-brings-ros-2-development-self-driving-cars/ (accessed Aug. 29, 2020).

[154] "NG Supplier_Scorecard_Guidelines," Dec. 29, 2018. https://www.northropgrumman.com/suppliers/OasisDocuments/Supplier_Scorecard_Guidelines.pdf (accessed Dec. 28, 2018).
[155] "Attack tree," *Wikipedia*. Mar. 07, 2020, Accessed: Mar. 30, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Attack_tree&oldid=944367692.
[156] H. S. Lallie, K. Debattista, and J. Bal, "A review of attack graph and attack tree visual syntax in cyber security," *Computer Science Review*, vol. 35, p. 100219, Feb. 2020, doi: 10.1016/j.cosrev.2019.100219.
[157] "Fault tree analysis," *Wikipedia*. Feb. 25, 2020, Accessed: Mar. 30, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Fault_tree_analysis&oldid=942559256.
[158] M. D. Walker, "being a Thesis submitted for the Degree of Doctor of Philosophy in the University of Hull," p. 299.
[159] "Petri net," *Wikipedia*. Mar. 21, 2020, Accessed: Mar. 30, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Petri_net&oldid=946658258.
[160] S. Arnborg, "Efficient algorithms for combinatorial problems on graphs with bounded decomposability — A survey," *BIT*, vol. 25, no. 1, pp. 1–23, Mar. 1985, doi: 10.1007/BF01934985.
[161] H. L. Bodlaender, "A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth," *SIAM J. Comput.*, vol. 25, no. 6, pp. 1305–1317, Dec. 1996, doi: 10.1137/S0097539793251219.
[162] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer, "DAG-based attack and defense modeling: Don't miss the forest for the attack trees," *Computer Science Review*, vol. 13–14, pp. 1–38, Nov. 2014, doi: 10.1016/j.cosrev.2014.07.001.
[163] P. Weber and C. Simon, *Benefits of Bayesian Network Models*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2016.
[164] K. B. Korb and A. E. Nicholson, *Bayesian artificial intelligence*. Boca Raton: Chapman & Hall/CRC, 2004.
[165] A. B. Mrad, V. Delcroix, S. Piechowiak, P. Leicester, and M. Abid, "An explication of uncertain evidence in Bayesian networks: likelihood evidence and probabilistic evidence: Uncertain evidence in Bayesian networks," *Applied Intelligence*, vol. 43, no. 4, pp. 802–824, Dec. 2015, doi: 10.1007/s10489-015-0678-6.
[166] M. Paskin, "2. Structured Representations," p. 35.
[167] D. X. Huang, "D-separation." https://cgi.csc.liv.ac.uk/~xiaowei/ai_materials/26-PGM-D-separation.pdf (accessed Feb. 09, 2020).
[168] "Markov blanket," *Wikipedia*. Jan. 20, 2020, Accessed: Feb. 09, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Markov_blanket&oldid=936640055.

[169] M. Gormley, “Bayesian Networks.” https://www.cs.cmu.edu/~mgormley/courses/10601-s17/slides/lecture23-bayesnet2.pdf (accessed Feb. 09, 2020).
[170] D. Koller, N. Friedman, L. Getoor, and B. Taskar, “2 Graphical Models in a Nutshell,” p. 43.
[171] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.
[172] R. Prabhakaran, R. Krishnaprasad, M. Nanda, and J. Jayanthi, “System Safety Analysis for Critical System Applications Using Bayesian Networks,” *Procedia Computer Science*, vol. 93, pp. 782–790, 2016, doi: 10.1016/j.procs.2016.07.294.
[173] C. Venkatasubramanian Ramakrishnan, “Cyberrisk Assessment Using Bayesian Networks,” 106AD, Accessed: Jan. 18, 2020. [Online]. Available: https://www.isaca.org/Journal/archives/2016/volume-6/Pages/cyberrisk-assessment-using-bayesian-networks.aspx?utm_referrer=.
[174] I. Atoum and A. Otoom, “Effective Belief Network for Cyber Security Frameworks,” *International Journal of Security and Its Applications*, vol. 10, no. 4, pp. 221–228, Apr. 2016, doi: 10.14257/ijsia.2016.10.4.21.
[175] L. De Wilde, “A Bayesian Network Model for Predicting Data Breaches,” p. 174.
[176] M. Rausand, “Risk Assessment: Theory, Methods, and Applications | Wiley,” *Wiley.com*. https://www.wiley.com/en-us/Risk+Assessment%3A+Theory%2C+Methods%2C+and+Applications-p-9780470637647 (accessed May 23, 2020).
[177] K. H. Quanta, “How a Pioneer of Machine Learning Became One of Its Sharpest Critics,” *The Atlantic*, May 19, 2018. https://www.quantamagazine.org/to-build-truly-intelligent-machines-teach-them-cause-and-effect-20180515/ (accessed Jan. 12, 2019).
[178] B. USA, “Bayesia S.A.S. Corporate Homepage.” http://www.bayesia.com (accessed Feb. 10, 2019).
[179] “Gallery,” *Knightscope*. https://www.knightscope.com/gallery (accessed Sep. 27, 2020).
[180] “‘Minority Report’ Movie Vindicated As KnightScope K5 Brings Precrime Technology To The Forefront • Now The End Begins,” *Now The End Begins*, Apr. 19, 2014. https://www.nowtheendbegins.com/minority-report-movie-vindicated-knightscope-k5-brings-precrime-technology-forefront/ (accessed Sep. 27, 2020).
[181] Vincenzo DiLuoffo, William R.Michalson, Berk Sunar, “Credential Masquerading and OpenSSL Spy: Exploring ROS 2 using DDS security.”
[182] S. Sanghai, P. Domingos, and D. Weld, “Relational Dynamic Bayesian Networks,” *Journal of Artificial Intelligence Research*, vol. 24, pp. 759–797, Dec. 2005, doi: 10.1613/jair.1625.
[183] “Dynamic Bayesian network,” *Wikipedia*. Dec. 02, 2019, Accessed: May 12, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Dynamic_Bayesian_network&oldid=928922993.

[184] A. Hahn, R. K. Thomas, I. Lozano, and A. Cardenas, “A multi-layered and kill-chain based security analysis framework for cyber-physical systems,” *International Journal of Critical Infrastructure Protection*, vol. 11, pp. 39–50, Dec. 2015, doi: 10.1016/j.ijcip.2015.08.003.
[185] “Threat Mitigation| Incident Response | Defender Lifecycle | Gigamon.” https://www.gigamon.com/campaigns/network-security-threat-defense.html (accessed May 03, 2020).
[186] B. USA, “Bayesia S.A.S. Corporate Homepage,” Feb. 10, 2019. http://www.bayesia.com (accessed Feb. 10, 2019).