THESIS

THE APPLICATION OF MODEL-BASED SYSTEMS ENGINEERING TO UNDERSTAND SECURITY OF SYSTEMS USING SAE J1939

Submitted by

Gabe Salinger

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2024

Master's Committee:

Advisor: Jeremy Daily

Daniel Herber
Bret Windom

ABSTRACT

THE APPLICATION OF MODEL-BASED SYSTEMS ENGINEERING TO UNDERSTAND SECURITY OF SYSTEMS USING SAE J1939

The Engineering community is adopting a Digital Engineering approach enabled by Model-Based Systems Engineering (MBSE) as an effective tool for designing complex systems. As technology continues to rapidly advance, security risk mitigation and requirements engineering is becoming a prominent and important factor in the cybersecurity domain. As a result, engineering methods and frameworks must constantly be improved and updated to implement the successful realization of cyber-physical systems (CPS). With the inherent connectivity, accessibility, and lack of security making CPSs attractive targets for cyber attacks, integrating security considerations into system development is crucial. With 'security by design' being a fundamental pillar of system development, MBSE plays a pivotal role in shaping secure system architectures.

In this thesis, I explore the application of MBSE to the system security domain, focusing on secure system development and the incorporation of security by design throughout the system development phase. This is accomplished by investigating the utility of MBSE in understanding the vulnerabilities of a Medium to Heavy Duty (MHD) vehicle, improving its security posture, and providing recommendations on how to improve the process.

This is achieved by first exploring the utility of simulation using model-based tools to better understand complex systems, and bridge the gap between bottom-up and top-down approaches. Next, an established method, MBSEsec, is applied to the system of interest (SOI) to develop security controls for the vehicle's transport protocol. Additionally, recommendations are provided for improving the method's effectiveness in documenting vulnerabilities, and risk. MBSEsec is a security-focused MBSE method using SysML to develop a system architecture that highlights

security design considerations. The method's structured workflow facilitates the elicitation of security requirements and controls using specific systems modeling activities.

The primary focus is on the heavy vehicle network transport protocol, J1939, serving as the SOI. The discovery and validation of new exploits that take advantage of vulnerabilities in the data-link layer of the protocol highlights the need to elicit better security requirements for cyber-physical systems (CPS). Using the J1939 network as the SOI for this work allows the models to be supported by and validated with on-vehicle testing. This work contributes a survey of modeling approaches for secure system design.

Lastly, this thesis details the development of a novel approach for system-level mission-focused security goal elicitation. EGRESS: Eliciting Goals for Requirement Engineering of Secure Systems, incorporates best practices from security requirement engineering works, and utilizes Model-Based Systems Engineering to formulate security goals for cyber-physical systems, aiming to create more comprehensive security requirements.

ACKNOWLEDGEMENTS

First, I would like to thank Dr. Jeremy Daily for welcoming me to his research team in the summer of 2022, and for supporting me and granting me the opportunity to pursue a Master's in Systems Engineering at Colorado State University. His support throughout my academic career at CSU is the primary reason for the successful completion of this degree. And most of all, thank you for leading by example as a supervisor, academic, and professional engineer. I'd also like to thank the members of my distinguished committee for their thoughtful considerations, time, and guidance.

Additionally, I'd like to thank the faculty and staff of the Department of Systems Engineering at Colorado State University for their instruction, guidance, facilities, resources, and academic support. I'd also like to also thank the United States Air Force for supporting my pursuit of higher education as I begin my career as a military officer.

Finally, I'd like to thank my current colleague and former teacher Trae Span, for your consistent support and advice along the way, both in and out of the classroom.

DEDICATION

This thesis is dedicated to my loving grandparents John and Ruth Salinger, who have prayerfully supported me throughout my entire life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivating the Problem

Modern systems throughout many industries such as automotive, aerospace, medical, industrial, and defense have become very complex. As a result of innovation, both complexity, and change will continue to escalate in products, services, and systems. As new systems continue to be developed and incrementally improved, there is a need for a well-defined process for life cycle management and development that reduces system risk. As a result, engineering methods and frameworks must constantly be improved and updated in order to implement the successful realization of these complex systems. The primary factors that make modern systems so complex are many, to name a few [4]:

1. Interactions of many parts (including autonomous parts, ie.)
2. Networked hierarchical activities (ie. Social networks, System of systems)
3. Unexpected or unpredictable emergence (ie. Accidents, system breakdowns)
4. Complicated transition laws (ie. markets, cascading failures)

The practice of systems engineering involves managing and combining the work of multidisciplinary engineering fields, with the goal of realizing the successful design of complex systems that meet stakeholder needs. This process typically follows the 'System-V" diagram [5] also known as the System Development Life Cycle (SDLC). This process involves four primary phases; Requirements Elicitation, System Design, Implementation and Testing, and Deployment and Maintenance. Due to the rise in the complexity of design, organizations are moving away from document-based engineering approaches to digital engineering and model-based activities in the system design process [6]. Model-Based Systems Engineering (MBSE), a methodology that employs digital engineering tools like SysML (Systems Modeling Language), has gained considerable traction in designing and developing complex systems. This is especially true for cyber-physical system design

where it has been adopted by the Department of Defense and various companies [7]. The International Council on Systems Engineering (INCOSE) describes MBSE as the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases [8]. Some of the advantages of model-based engineering include the decrease in development time, enhanced analysis, and increased re-uses of elements within a system [9].

Along with the rise in system complexity and the need for better design methodologies, also comes the ever-growing persistent cybersecurity threat. Cybersecurity can be defined as the practice of protecting systems, networks, and programs from digital attacks. The cybersecurity threat is one of the most challenging obstacles to cyber-physical systems in today's world. The INCOSE handbook states [5]:

> *"As the world becomes increasingly digital, the issue of cybersecurity is a factor that systems engineers need to take into account. Both hardware and software systems are increasingly at risk for disruption or damage caused by threats taking advantage of digital technologies"*

Cyber attacks have rapidly increased in the past decade, targeting nearly every system that can be digitally accessed [10]. The rise in attacks compels the need for strong cyber-defense strategies, especially with the important roles that electronic devices play in the operation of major systems. For example, today's connected vehicles depend on hundreds of embedded CPSs to perform correctly all the time regardless of operational environment or user behavior (malicious, benign or otherwise). Moreover, these CPSs are becoming increasingly popular cyber attack targets because of their inherent connectivity, ease of access, and lack of security. Reports of cyber attacks on power grids, medical devices, and even complex vehicles highlight the growing need to secure such CPSs [11] [12]. Car hacking competitions and conferences are becoming more commonplace in the cybersecurity industry [13] with new vulnerabilities constantly being discovered. In light of these persistent threats, it is important to implement strong systems engineering methods and techniques that enable the development of secure system architectures.

Nguyen et al. conducted a study of the state of the art of MBSE for cyber-physical systems (CPS) and concluded that there is a lack of engineering security solutions for CPSs, and that there are few industrial case studies of the use of MBSE for secure design. Nguyen also found that MBSE could be a key means to tackle the challenge of security threats primarily through security by design, abstraction, and the implementation of security objectives (Confidentiality, Integrity, and Availability) in the early stages of system design [14].

With 'security by design' being a fundamental pillar of system development, MBSE's role in secure system architecture is highly valued. Research shows that the engineering community views model-based systems engineering favorably based on an analysis of over 60 references discussing the approach [15]. The findings determined analysis and communication capabilities to be some of the most stated benefits of MBSE. The model-driven approach to secure architecture has also been proven to promote collaboration between system designers and security experts [16]. Madni et al. highlight that a lack of attention to system architecture and its influence on design can potentially lead to integration challenges later in the design process. They highlight that there is a tendency to rush the architecture stage, which leads to premature bounds on system design and life-cycle costs [6]. Furthermore, much of the current CPS design processes do not address security and security requirement elicitation until a preliminary architecture is established for the System of Interest. Additionally, most security-oriented methods require foundational security requirements and some level of system structure as input to begin the security analysis. There is a lot of potential for the use of MBSE to reduce both the issues of premature architecture specification, and requirement elicitation.

When it comes to system design, it is important to distinguish between two fundamental approaches. Traditionally, two primary design methodologies have been employed in the construction of complex systems: the top-down and bottom-up approaches. The top-down design approach involves starting with a high-level overview of a system, breaking it down into smaller components, and refining the details progressively. This method emphasizes understanding the system's preliminary architecture early on but might delay understanding of lower-level details. In con-

trast, the bottom-up approach begins with building individual components and subsystems and gradually integrating them into a larger system, promoting a deeper understanding of fundamental elements but potentially lacking a clear understanding of the system structure until later in the design process [17]. The systems engineering design process involves a combination of both approaches, which changes based on the specific stage of the system design. In the early stages of systems engineering, the top-down approach is commonly used. Starting by defining the high-level requirements and system architecture, then identifying the major components, interactions, and functionalities that the system needs to achieve. This helps to establish a clear vision of the overall system and its objectives. During the detailed design phase, when the focus shifts to individual components, the bottom-up approach is adopted. Ultimately, successful systems engineering involves a continuous iteration between the top-down and bottom-up approaches. The high-level architecture guides the development of the lower-level components, and insights gained from implementing these components can lead to refinements in the overall system design. With this perspective in mind, it is important to note that this work primarily focuses on the refinement and implementation of the top-down approach to system design, specifically in regard to system security.

This thesis aims to explore and enhance the practice of Model-Based Systems Engineering (MBSE) in system development and promote the integration of security by design through the system development phase. Its objective is to contribute to the existing knowledge by investigating the application of MBSE in the early design process, specifically in relation to securing vehicles and vehicle networks. This is achieved by exploring the utility of MBSE in understanding system vulnerabilities, applying established MBSE methods and frameworks from the security field to a novel system and providing recommendations, and developing a new approach for security goals elicitation for secure systems.

1.2 Research Questions

One of the largest challenges engineers and organizations face when developing a cyber-physical system is solving how to create a 'secure system'. With a majority of security methods and techniques in both industry and literature not relating to SE, this paper hopes to explore the convergence of MBSE and secure system design methodologies in the system development process. This thesis is written in hopes that it will consolidate information regarding system security, model-based systems engineering, requirements engineering, and Cybersecurity principles. Fig 1.2 illustrates the area of the system development life-cycle in which the role of MBSE is explored. The below research questions and tasks will guide this work.

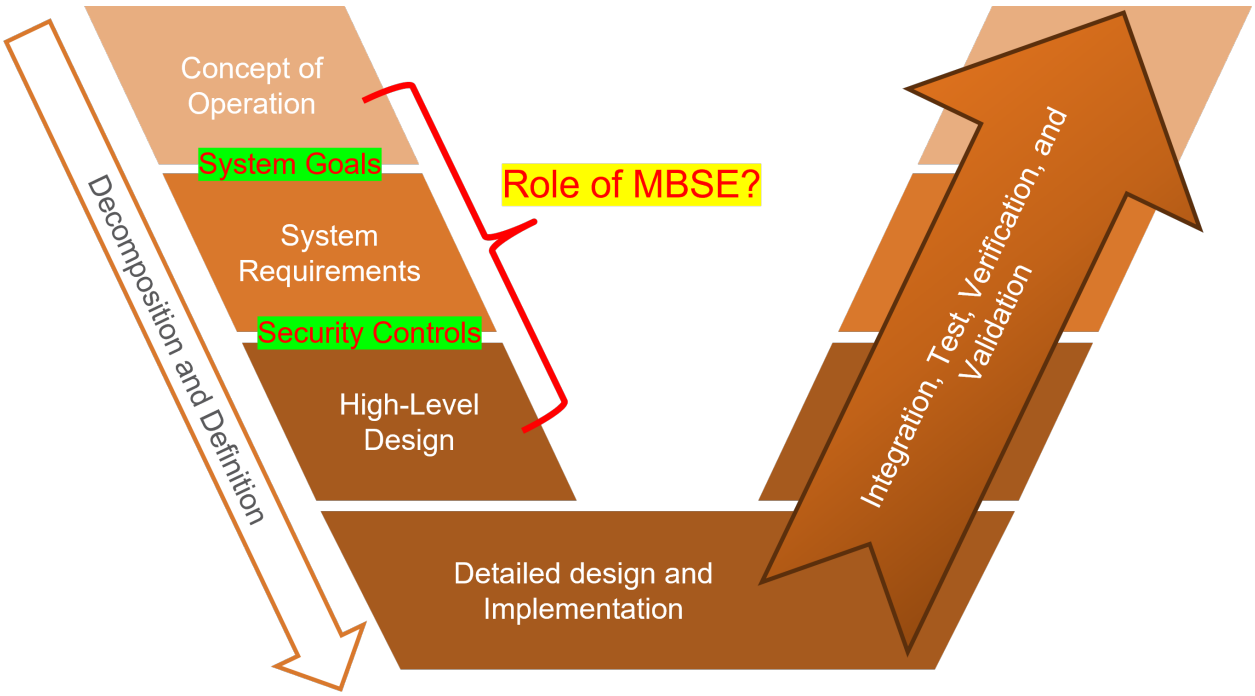

Figure 1.1: Role of MBSE in system design

Question 1: How can security be incorporated into the system development process using MBSE?

Motivation: INCOSE Vision 2035 states that "The Future of Systems Engineering Is Predom-

inantly Model-Based" (p.33), and that "Cybersecurity will be as foundational a perspective in systems design as system performance and safety are today" (p. 37) [18]

Research Tasks:

1.1 Develop an understanding of systems modeling (SysML in Cameo) and of the SOI: Truck Networks

1.2 Build a simulation of real-time network attacks (SysML) and investigate benefits

1.3 Investigate what methods exist that incorporate MBSE to develop more secure systems

1.3.1 Investigate literature, methods, applications and tools related to: SE, MBSE, Security Requirements Engineering, and primary security-related systems engineering standards

1.4 Demonstrate the utility/applicability of selected security MBSE approach to truck networks

1.5 Evaluate the approach, highlighting its benefits, and shortcomings

Output: J1939 Network attack simulation, demonstrated utility and value of an MBSE method for secure system development

Question 2: How can MBSE be leveraged to conduct security requirement elicitation in the design of secure systems?

Motivation: "The system shall be secure" is not an adequate security requirement. Specified security requirements often become security-specific architectural constraints [19] [20].

Research Tasks:

3.1 Investigate literature, frameworks, and methods related to security requirement elicitation

3.2 Support the development of a Security Focused Goal Elicitation Method using MBSE (EGRESS)

a. Develop an MBSE model that implements the proposed method

3.3 Apply the new method to a proposed system and investigate its utility.

3.4 Demonstrate the benefits of the outputs of this method to the rest of the design process.

Output: A proposed approach to elicit security goals for systems in conceptual design using MBSE.

This work will begin with investigating the use of MBSE by building an advanced understanding of a modeling tool and implementing it in vehicle truck networks. This work will then use a proposed method to secure discovered vulnerabilities in the J1939 protocol and discuss findings and shortcomings. The proposed method will be applied to a second system of interest, through which recommended improvements will be implemented into the method. Finally, this work will assist in developing a method that covers the gaps in current MBSE security methods and demonstrate its utility for system development through modeling.

1.3 Background

1.3.1 Introduction to MBSE

The overarching goal of MBSE is not to replace traditional document-based systems engineering, rather, it is to facilitate traditional SE activities resulting in enhanced communications, specification and design, design integration, and re-use of artifacts and models. In a document-based SE approach, a large amount of information about a system is gathered containing everything from stakeholder needs, trade studies, analysis reports, procedures, and design decisions. While all of this information is necessary for an effective system design, it is difficult to maintain, synchronize, update and revise. The role of MBSE is to capture and guide the system design through the development of a system model. The INCOSE SE handbook states that "MBSE formalizes the application of SE through the use of models", and that MBSE "allows systems engineers to have a single and traceable source of truth for the system requirements, specification, design, and validation and verification" [5].

The INCOSE Systems Engineering Vision 2035 highlights that the future of SE Is predominantly model-based, stating that

> *"Although a growing number of systems engineering organizations have adopted model-based techniques to capture systems engineering work products, the adoption is uneven across industry sectors and within organizations. Custom, one-off simulations*

are used for each project, and there is still limited reuse of models especially during critical early phases of systems architecting and design validation." [18]

Here, we can see that MBSE's role in engineering and system design will only become more critical as we look to the future and that there is a current need for centralized and proven MBSE methodologies. With the inherent complexity of modern systems, and the constant threat of cybersecurity, the objective of MBSE is to serve as part of the engineering baseline, and the emphasis is placed on defining and evolving the model using model-based tools [21]. Effectively, the systems model is developed and elaborated simultaneously with the system under development during the conceptual development and system development phases. In the field of SE, the challenge of communication across engineering disciplines, completeness of design, and well-traced documentation are all addressed through the use of MBSE [22].

Effective systems modeling relies on three core MBSE elements. These elements are the modeling language, modeling tool, and modeling method [21]. Fig 1.1 illustrates the role of the core elements of MBSE within the systems engineering process. Ultimately, these three core elements combine to produce a systems model that includes system specifications, design, analysis, and verification. The model consists of model elements that represent different key parameters such as requirements, design, test cases, design rationale, behaviors, blocks, and their interrelations [21].

The generation of system models requires a modeling language that defines the syntax (modeling elements) and a standardized medium for communicating and information sharing. The modeling language defines the elements, relationships, and visual aspects of the diagrams. Typically, when discussing modeling languages, the two most common languages to arise are the Unified Modeling Language (UML) and the Systems Modeling Language (SysML), and the distinction between these two is very important. UML is a language for specifying, visualizing, constructing, and documenting artifacts of software and non-software systems (eg. business models) [23]. Due to the complexity of the SE process, and the domain-specific terminology required, UML is not sufficient for systems modeling. In a joint effort, the Object Management Group, and INCOSE worked together to develop a domain-specific modeling language for systems engineering -

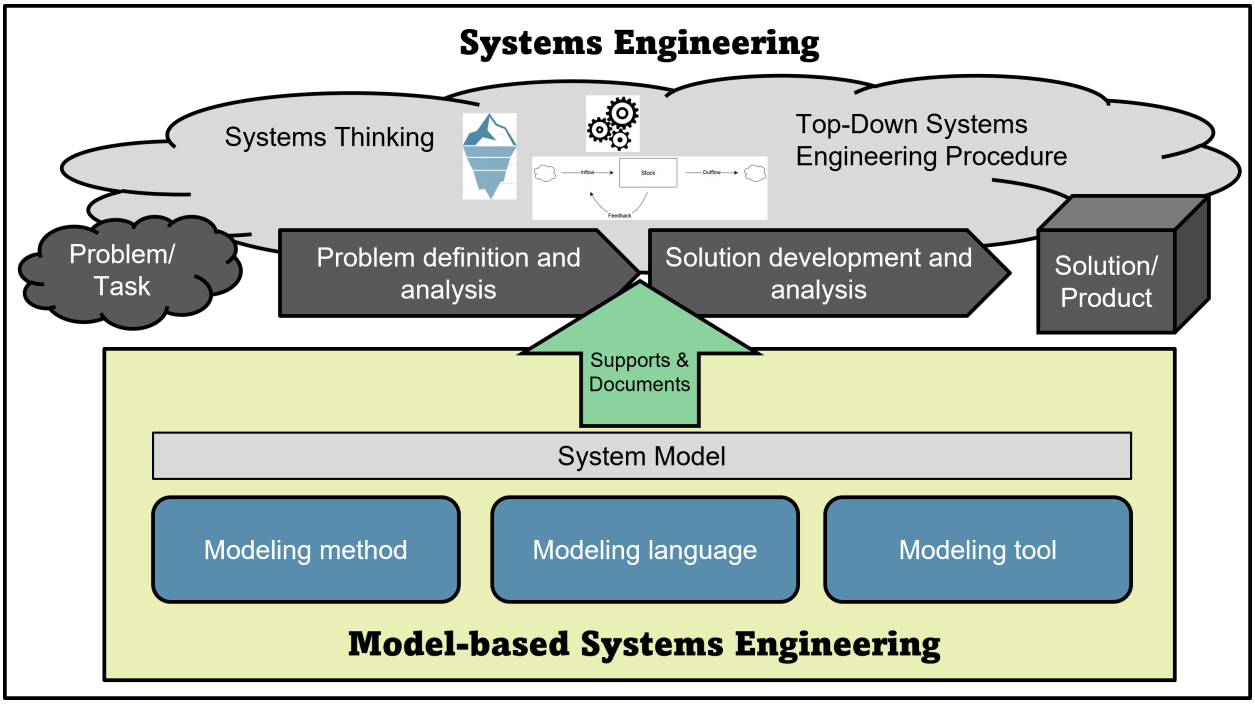

Figure 1.2: MBSE role in systems engineering

SysML. Simply put, SysML is a profile of UML - a systems engineering-domain-specific modeling software. Every effective model has three major aspects, Requirements, behavior, and structure, which are the three primary categories of the taxonomy of SysML diagrams, as can be seen in figure 1.3. Within the field of MBSE, the use of Systems Modeling Language (SysML) is the most common and established language [24]. It is important to note that modeling Languages are not tied to specific methodologies, but are a means of implementing them.

Developing a system model using SysML

As can be seen in Fig 1.2, there are 9 primary types of SysML diagrams used in MBSE. The four *behavior* diagrams are used to describe the behaviors of the system and its elements. These diagrams help answer questions such as; What do the structural components do or perform? How do things flow among or through the system? What operating states does the system have? and how do the sequence of events take place during the operation of the system [25]?

The four *structure* diagrams describe the structure and performance of the system in terms of; components (block elements) and how they are hierarchically connected, parts, detailing the

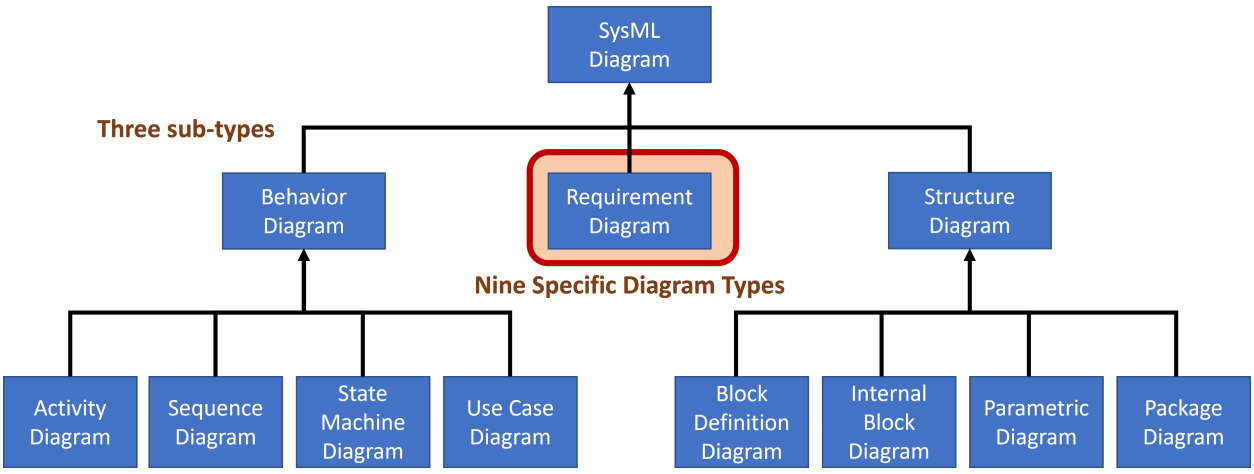

Figure 1.3: Taxonomy of SysML diagrams

internal structure of the components, performance indicators and variables that define how well the system and its components are satisfying requirements, and organization which outlines the containment of the model elements.

Requirement diagrams rest in their own category of the system model. They are used to depict and organize text-based requirements and illustrate their relationships and hierarchy. They can be used to support matrix displays that display requirements being satisfied by functions and structure.

Within these diagrams, system elements are represented as specific *types* of elements. These element *types* in the SysML language possess a variety of different characteristics and features, such as value parameters, parts lists, unit of measure, ports, etc. Each element relationship also has unique characteristics that help define and illustrate the nature of the relationship such as flow rates and directional indicators.

Using these primary sub-types of SysML diagrams, a system model can be developed to reflect any system. Modeling tools for MBSE are a special kind of authoring software that implements the rules for modeling and visualization based on modeling languages to create and manipulate the model [21]. The tool is the actual interface between the language and method, and the engineer. Properly developed tools provide the users with all of the key capabilities required to develop system models. Some of the industry-tested tools are Cameo Systems Modeler, Enterprise Architect, and Rhapsody.

A systems model is built based on different SE activities performed throughout the design process and by different engineering teams [24]. In order to develop a consistent model, different methods provide guidance and a structure that defines what needs to be modeled, at which stages in the process, and how to do it (Techniques). Modeling without a method to be followed results in an ineffective and incoherent model that becomes very difficult to implement and interpret. A complete modeling method defines a clear scope and purpose of the model, as well as clearly defined activities to follow. In [26] Sannes et. al. highlight that modeling methods can be categorized into four primary subsections; Methods compliant with SE standards, methods compliant with aeronautical standards, methods associated with a modeling tool, and methods that were not originally designed for MBSE. Sannes's work provides System Engineers with decision criteria for effective selection of the appropriate MBSE languages, tools and methods. The author also highlights that many issues remain to be explored in terms of languages, tools and methods to promote the transition of the industry away from document-based design to MBSE. A variety of different modeling methods exist in the literature, some of the most prevalent being Object-Oriented System Engineering Methodology (OOSEM), MagicGrid, and NASA JPL State Analysis.

In an effort to better understand MBSE, Campo et al. conducted a survey of previous studies concerning the effectiveness of MBSE, then conducted a study of their own. The objective of the study was to investigate how the SE community perceives the value of MBSE. The primary research question was; What attributes and impacts of MBSE are perceived positively or negatively in literature? Based on an analysis of over 60 references discussing the approach, the systems engineering community view model-based systems engineering favorably. The findings determined Analysis and communication capabilities to be some of the most stated benefits of MBSE. The extensive review of SE literature found a 4:1 positive to the negative-attribute ratio for MBSE. [15]

Originally, MBSE was dedicated to managing the complex system creation in terms of key SE activities such as system requirements, design, analysis, verification, and validation activities leaving security aspects aside. This work focuses on how to leverage MBSE for secure system design while implementing security principles into the MBSE process.

1.3.2 Key Systems Engineering and System Security standards

In order to address the question of how can 'security by design' be improved in early system development, it is important to have a foundational understanding of the guiding principles laid out in the literature. These guiding standards help shape the current state-of-the-art for security and secure system design. Understanding these standards is vital to ensure effective system development and proper security control implementation. The key guidance documents influencing the system development efforts in this work are the ISO/IEC/IEEE 15288 (Systems Engineering lifecycle process) [27], ISO/IEC/IEEE 42010 (Architecture Description - MBSE models) [28], and the INCOSE Handbook for Systems Engineering [5]. The key Guidance documents influencing the implementation of security and MBSE efforts in this work are NIST 800-160v1 specification (Systems Engineering lifecycle process) [27], NIST 800-53 v2 (Security and Privacy Controls for Federal Information Systems and Organizations) [29], and SAE ISO 21434 (Need for Cybersecurity) [30].

Systems Engineering Standards

The Key systems engineering standards that influence this work are as follows:

ISO/IEC/IEEE 15288: Systems and Software Engineering - System Life Cycle Processes: This standard is one of the foundational documents in systems engineering and is heavily referenced in the INCOSE SE handbook. This document provides a framework for systems and software engineering processes, including the system development life cycle. While it doesn't specifically focus on cyber security, it offers valuable insights into how Model-Based Systems Engineering (MBSE) can be integrated into the system development process. This document emphasizes the systematic and structured approach to systems engineering, which highlights the need for MBSE to enable the development of a detailed and traceable model. This standard also highlights the need for early requirement elicitation and analysis, which is one of the primary benefits of using MBSE. Lastly, the standard promotes an incremental and iterative design process that has a robust architecture. MBSE facilitates the creation of detailed system architecture models and is very effective at promoting traceability and iteration.

ISO/IEC/IEEE 42010: Systems and Software Engineering - Architecture Description: This standard lays out the guidelines for architecture descriptions, encompassing MBSE models. It provides valuable guidance on how to structure, organize, and ensure the quality of architecture descriptions, leading to improved communication and collaboration among stakeholders. Within Model-Based Systems Engineering (MBSE), this standard holds significant importance as it furnishes essential principles for articulating the system's structure, behavior, and requirements, with special consideration for security-related aspects. This standard maintains an architecture-centric view of the system, and highlights that MBSE facilitates security architecture by design, leading for early capture of security related elements. Lastly, the standard highlights the need or stakeholder involvement in architecture definition, which promotes better traceability throughout the design process to ensure the system is meeting stakeholder concerns.

INCOSE Systems Engineering Handbook: Although not an ISO/IEC standard, INCOSE published the Systems Engineering Handbook, which is a valuable reference for all matters related to systems engineering. It covers various aspects of systems engineering, including modeling techniques and tools. INCOSE outlines 14 Systems Engineering Technical Processes from ISO/IEC/IEEE 15288 [27] in the Systems Engineering Handbook [5]. These processes enable systems engineers to identify system needs and shape the design across engineering disciplines to meet stakeholder requirements. Starting from conceptual design and spanning the entire lifecycle, these processes facilitate coordination between specialists, engineering disciplines, stakeholders, and operators, resulting in a system solution that fulfills performance, environmental, interface, and design constraints.

INCOSE also discusses the importance of security engineering within the design process. The objective of security engineering is to ensure that the system can function under disruptive conditions caused by misuse and/or malicious behavior. INCOSE details the important sources for potential disruptive conditions and the importance of using SE principles to mitigate the risks. The handbook highlights that in order to minimize the security risk potential of a system during its life cycle, design engineers must be provided with a complete set of security-related requirements.

These requirements must be derived from stakeholder security interests during the stakeholder needs and requirements definition process and should be captured in clear system requirements. This document also emphasizes that established security requirements must be allocated to functional system elements during the architecture design process.

The INCOSE handbook also discusses the benefits of MBSE, and the primary methods that can be followed when building a model. The benefits of MBSE are described as improved communications, increased ability to manage system complexity, improved product quality, enhanced knowledge capture, and improved ability to teach and learn SE fundamentals [5]. The book states that leveraging MBSE throughout the SE design process significantly improves system requirements, architecture, and design quality.

System Security Standards

NIST 800-160v1: National Institute of Standards and Technology (NIST) Special Publication 800-160, is a comprehensive document that provides guidance on systems engineering principles and practices for building secure and resilient information systems. The publication emphasizes the integration of security and resilience principles into the entire SDLC, from planning to decommissioning. It provides an in-depth exploration of systems engineering fundamentals, risk management, and the essential aspects of security and resilience integration. The guidelines underscore the importance of tailoring the system development lifecycle to meet specific system requirements while focusing on security engineering practices such as threat modeling, secure coding, and access control, as well as resilient engineering considerations like redundancy and continuity planning. By addressing these critical aspects, this document supports organizations in creating robust and dependable systems capable of withstanding adverse events and safeguarding sensitive information [31].

Appendix D of the standard discusses the approach and considerations for applying concepts of a secure system. The objective of this approach is to deliver system capabilities at an acceptable performance level while minimizing loss occurrence and extent. The approach incorporates both preemptive and reactive aspects to achieve intended behaviors and outcomes. Preemptive mea-

sures involve addressing potential loss scenarios before they occur, while reactive measures focus on informed decision-making and actions after a loss event. The design approach emphasizes identifying intended behaviors and outcomes, recognizing potential loss conditions, altering the system design to prevent or limit loss, and iteratively addressing potential failure scenarios. The standard highlights that early recognition of states in which loss may occur leads to preemptive solutions.

NIST SP 800-53 Revision 5: NIST Special Publication 800-53 Revision 5 serves as a valuable framework for security and privacy controls in federal information systems and organizations. This standard provides vital guidance for securing systems through a comprehensive set of security controls and a risk-based approach. These standards play a crucial role in safeguarding sensitive data and protecting against cyber threats in government agencies and beyond.

Its utility in the context of Model-Based Systems Engineering (MBSE) lies in its control selection and tailoring capabilities, allowing organizations to choose and adapt security controls that align with the specific requirements of their system models. SP 800-53 encourages the integration of security controls into the system development life cycle, which can be applied to MBSE by embedding these controls directly into system models and designs. The risk-based approach advocated by the standard aligns well with MBSE principles, facilitating the early identification and mitigation of security risks during the modeling process. Furthermore, SP 800-53 promotes traceability, enabling clear linkage between security controls, system requirements, and system architecture in MBSE.

SAE ISO 21434: Need for cybersecurity: ISO 21434 serves as a comprehensive guide for cybersecurity development in vehicles. It encompasses vocabulary, objectives, requirements, and guidelines for cybersecurity engineering. The standard outlines a step-by-step process for cybersecurity activities, starting from concept definition and extending through product development, verification, validation, production, and operations and maintenance. One crucial aspect of ISO 21434 is the Threat Analysis and Risk Assessment (TARA) section. TARA involves several steps, including asset identification, threat scenario identification, impact rating, attack path analysis, attack feasibility rating, risk value determination, and risk treatment decision. Incorporating this

process during the conceptual development of the system can help in generating security requirements, and integrating appropriate security controls.

1.3.3 Survey of existing MBSE methods for security

After exploring the different languages, tools, and methods that have been introduced for MBSE, Saqui-Sannes et al. [26] concluded that there has been a lot of work and publishing on modeling languages and tools, but there is a lack of MBSE methods that facilitate these languages and tools. D. Mazeika et al. [32] explored the current modeling approaches for security analysis and identified the most prominent methods for using MBSE for secure design. Based on the conclusions in the Mazeika work, these four methods were explored, as well as any other well-documented methods that utilize MBSE as a core pillar of the process. As noted previously, a Systems Modelling language is not sufficient to effectively run a project or design a system. A language must be combined with an appropriate methodology designed to facilitate effective 'security by design'. It is important to note that the focus of this review of methodologies is to present methods that incorporate both MBSE and security aspects of system design. While methods exist that just focus on security (ex. TARA), and others that just use MBSE (ex. OOSEM), this section focuses on those that incorporate both.

SysML-Sec: A Model Driven Approach for Designing Safe and Secure Systems [16]

This work introduces a SysML model-driven approach designed with the primary goal of promoting collaboration between system designers and security experts. The SysML-Sec method presents an extension of SysML diagrams for security as well as a methodology to be used for securing embedded systems. The three main phases of the methodology include a system analysis phase (based on the Y-Chart approach system analysis), a system design phase (Based on the standard V model), and a system validation phase. The methodology recommends beginning with an analysis of both requirements and vulnerabilities while identifying the main functions of the system. Roudier demonstrates the method with an example of securing vehicle network communication.

The methodology presents good techniques to document and analyze the system for vulnerabilities and recommends beginning the approach by identifying security objectives, and system assets. However, it is not clear how the security requirement definition and attack scenario definition steps are structured. The relationship between the attack analysis and the mapping of functions over execution nodes is very difficult to follow from a model's perspective. The SysML-Sec methodology recommends beginning the security design process with an architectural analysis which implies that a preliminary architecture must exist. A more effective approach to secure system design appears to be beginning with the development of system goals, context, and use cases, followed by preliminary security requirements.

Enhancing CHASSIS: A Method for Combining Safety and Security [33]

The Combined Harm Assessment of Safety and Security for Information Systems method defines a process for security and safety evaluation, emphasizing the importance of addressing both of these aspects of a system during the development phase. CHASSIS is used in the requirements engineering process during the development lifecycle, with a particular focus on eliciting functional, safety, and security requirements. Raspotnig et al. highlight that one of the key advantages of CHASSIS is the use of visualization capabilities to ease communication between stakeholders of various backgrounds in combined safety and security assessments is very beneficial. They discuss that there are no popular model-based development techniques that incorporate both safety and security, that also center both assessments around common artifacts (diagrams), which highlights the need for MBSE in these phases of development. This method extends the UML use case diagram with additional elements of misuse and misuser to allow defining attackers and their threats to the system of interest within the diagrams. While this method highlights good techniques, it is outdated and focuses on eliciting requirements, not effective model development.

UMLSec: Extending UML for Secure Systems Development [34]:

The UMLSec approach is a light extension of the Unified Modeling Language (UML) that offers a means to specify security requirements for a system under analysis. Instead of introducing new diagrams, UML Sec employs specialized stereotypes written in the Object Constraint Lan-

guage (OCL). These security-related stereotypes facilitate the expression of security requirements and the depiction of potential attack and failure scenarios, utilizing standard UML diagrams like use case, activity, and sequence diagrams. It is designed to help analysts and developers to model the security aspects of a system, such as security policies, access control mechanisms, and security protocols. While in theory, this method would be very effective, it assumes a known system architecture at some level and is based on known threats. It, like many other methods, starts with an assumption that a set of security requirements already exist. The method also focuses on the development of stereotypes, instead of a thorough model development process, and is based on UML, an older language, not tailored for domain-specific systems engineering terminology.

MBSEsec: Model-Based Systems Engineering Method for Creating Secure Systems [2]

MBSEsec incorporates security analysis into the systems engineering process at an early stage of system design. The MBSEsec method outlines the activities and steps for designing secure systems with the SysML security profile. The method involves four phases that provide the appropriate techniques and diagrams to be used. The first phase involves identifying and capturing security requirements in a SysML requirements diagram or table. The second phase involves defining the structure of the system by capturing and allocating assets using the block definition diagrams and allocation relationships. The third phase outlines the process for modeling behavioral and structural threats and risks through misuse case, activity, and threat definition diagrams. Lastly, the fourth phase defines security control objectives and controls using activity diagrams and a control definition diagram. The MBSEsec method states that its primary objective is to "develop a feasible and efficient MBSE method for creating secure systems while respecting ISO/IEC 27001 [Information Security Management] requirements". The objective of MBSE is to facilitate system design that further aligns with security methods, as highlighted by this method. While the objective of MBSEsec is to derive security controls, it can also be used to elicit better security requirements. Adjusting the sequence in which the phases are accomplished, with a focus on deriving functional security requirements can be very beneficial to the design process. This method loses effectiveness if there is no preexisting or preliminary architecture, and if a threat and risk assessment has not

been conducted. This method requires a detailed understanding of a system and would be harder to use if the design began with stakeholder needs.

Integrating Security Requirements Engineering into MBSE: Profile and Guidelines [32]

This work focuses on the application of security requirements engineering to MBSE. It highlights that MBSE does not directly address the security aspects of a system, rather it is a tool that can be used to generate better requirements. The paper presents a security profile that can be used in SysML to characterize and allocate the security aspects of a previously modeled (Legacy) system. The profile defines new security-related stereotypes that are used throughout the model to characterize different elements and is primarily comprised of assets, threats, and vulnerabilities. This work also presents a security domain model that outlines assurance concepts, items to be protected, and risk-related concepts. However, this work does not clarify the inputs that must be generated before using this profile. It is important to have a clear CONOPS, as well as established security goals before using this profile. It also does not present an effective means of verifying that all security requirements are met by the system design.

Towards an Integrated Model for Safety and Security Requirements of Cyber-Physical Systems [35]

Brunner et al. introduce a domain-agnostic approach to jointly model security and safety requirements. The work combines best practices from requirements engineering, risk management, and systems and software engineering to produce a generic model that can be aligned with pre-existing modeling techniques by not introducing specialized concepts. A key distinction of this method is that it begins with defining clear security goals, which act as a foundation for the security requirements that follow. In order to address the risk scenarios, the model assumes that all safety and security requirements are non-functional. This limits the scope and effectiveness of these requirements, whereas other methods encourage functional safety and security requirements as the primary means of secure system development. Constraining safety and security requirements as nonfunctional increases the risk of missing a security requirement designed to prevent a certain risk scenario.

Identifying Security Issues with MBSE while Rebuilding Legacy Software Systems [36]

Mazeika et al. introduce the use of MBSE to tackle security issues when recreating legacy software systems. The paper provides an approach to applying various techniques in the MBSE environment in order to secure such systems. The authors highlight that MBSE is a valuable tool because of its effectiveness in enhancing communication between stakeholders, modeling, comparing, building current and future systems, conducting change analysis, and in rapid prototyping. Mazeika et al. detail the security techniques that can be used in MBSE: Security requirements engineering, Misuse cases, attack scenarios, and security controls. The authors then present their approach to securing legacy systems which involves capturing security requirements and goals, capturing the system assets, modeling threats and risks, and deciding objectives and controls. These steps are all effective activities to accomplish, however, this method fails to detail the appropriate inputs to this approach, as well as the intended outputs that this method provides. Once the security controls are developed, there needs to be an indication of where the design process should go next. This method must have clear inputs and an iterative process that then leads the design team smoothly into the next phase of the system development. This work is a precursor to the introduction of the MBSEsec methodology, therefore, it follows a similar pattern of steps and techniques.

A Conceptual Model-Based Systems Engineering Method for Creating Secure Cyber-Physical Systems [37]

Larsen et. al. provide a review of relevant state-of-the-art MBSE methods including MBSEsec, CHASSIS, SysML-Sec, and SEED. The paper then provides a comparison of terminology between the methods, and highlights the advantages and disadvantages of each. The authors then produce a conceptual security-orientated solution very similar to MBSEsec and apply it to an air traffic control system. This paper is helpful in summarizing the well-established MBSE security methodologies but fails to introduce significant new ideas or contributions.

A taxonomy of MBSE Approaches by Languages, Tools and Methods [26]

While not particularly focused on security aspects of system design, this paper details, as well as compares and contrasts the different languages, tools, and methods involved with MBSE. By doing so, it provides System Engineers with decision criteria for the effective selection of the appropriate MBSE languages, tools and methods for a given system. The paper provides an overview of semi-formal modeling languages, like SDL (Specification and Description Language), UML, AADL (Architecture and Analysis Description Language), MATLAB Simulink, and a detailed description of SysML. De Sannes et. al. highlight that modeling methods can be categorized into four primary subsections; Methods compliant with SE standards, methods compliant with aeronautical standards, methods associated with a modeling tool, and methods that were not originally designed for MBSE. This paper was used as a tool in selecting the appropriate languages, tools, and methods for this work.

Systems Thinking and Model-Based Systems Engineering's Utility to Solve Complex Organizational Problems - Cyber-Physical System Design Team [38] Span et. al. apply MBSE to better design cyber-physical design teams. This work provides a great example of a well-written systems engineering paper, focused on the use of MBSE, and utilizes SysML. In this paper, MBSE and systems engineering principles are applied to enhance the security posture of cyber-physical design teams. The paper addresses the current state of CPS design teams using SysML artifacts, highlighting their vulnerabilities, and provides a recommended solution. The paper is an excellent demonstration of how a systems engineering paper should be structured in regard to the merging of MBSE and cybersecurity.

1.3.4 Relevant Security Requirements Engineering frameworks

In the later sections of this work, the emphasis shifts towards integrating Model-Based Systems Engineering (MBSE) into the early stages of system development, particularly in the context of security requirement elicitation. The subsequent literature review highlights various notable approaches employed for security requirement engineering (SRE), along with their respective limitations. Included is also a review of the STRIDE method for security threat analysis modeling,

which is used in Chapter 3. This comprehensive review aims to provide insights into the most effective strategies that can be incorporated into MBSE for the requirement elicitation phase.

Security Requirements Engineering Process (SREP) [39]

The Security Requirements Engineering Process (SREP) is a systematic methodology designed to address the need for the early integration of security considerations into system development. This method is designed to ensure that security requirements are effectively identified, documented, and managed throughout the system development life cycle. SREP involves structured activities like security risk assessments, threat modeling, and the selection of security controls to mitigate identified risks. It is a foundational methodology in the field of Security Requirements Engineering (SRE), and emphasizes the proactive inclusion of security requirements in the development process to foster a security-centric system design. One of the key contributions of SREP is the implementation of a Security Resource Repository, which is based on the idea of reusing security requirements proposed. The approach aims to build and manage security knowledge, assure the quality of security work, and focus on security early in the requirements stage of development through reuse, generic threats, and requirements from a shared repository. Along with the repository, SREP outlines the following activities to be followed during the development process:

1. *Agree on Definition:* Define stakeholders, security policies, and the organization's vision, forming the foundation for overall security requirements.
2. *Identify Vulnerable and/or Critical Assets:* Identify vulnerable and critical assets (Typically done by requirements engineer)
3. *Identify Security Objectives and Dependencies:* Involves identifying security objectives and dependencies using the predefined repository. Documented through security policies and a Security Objective Document.
4. *Identify Threats and Develop Artifacts:* Focuses on finding threats that could target assets and developing artifacts like misuse cases, attack tree diagrams, and UMLSec use cases

5. *Risk Assessment:* Estimates security risk based on relevant threats, their probability of occurrence, and negative impacts, with risk impact and likelihood both categorized on a scale of 1-5 in terms of severity.
6. *Elicit Security Requirements:* Corresponding requirements or clusters of requirements are identified for each threat, and additional security-related requirements are discovered.
7. *Categorize and Prioritize Requirements:* Categorizes and prioritizes security requirements based on the impact and likelihood analysis of threats.
8. *Requirements Inspection:* Validates the quality of teamwork and deliverables, assessing the overall security requirements engineering process
9. *Repository Improvement:* Updates to the repository are made with new and modified elements.

It is important to note that SREP's effectiveness is limited by the expertise and quality of threat modeling conducted, making it potentially challenging for inexperienced practitioners to identify and prioritize security risks adequately.

Security Requirements Engineering: A Framework for Cyber-Physical Systems [40]

This paper introduces a security requirements engineering framework for cyber-physical systems called Cyber-Physical Systems - Security Resource Repository (CPS-SRR). This method is an extension of SREP and is designed to create a secure CPS development process. The purpose of the paper is to demonstrate that SREP has many advantages, and is effective when extended to Cyber-Physical Systems. The authors introduce the framework as a centralized approach to security requirements engineering that allows for the storage of key information about the system and its domain and can be easily modified throughout the process of iteration. The framework involves the progression through 11 activities very similar to that of SREP, and it aimed at developing security requirements, specifically for cyber-physical systems:

1. Agree on definitions and gather security goals
2. Identify vulnerable and/or critical assets
3. Identify security objectives and dependencies

4. Identify threats and develop artifacts
5. Domain analysis and vulnerability identification
6. Risk Assessment
7. Develop Patch Management
8. Elicit Security Requirements
9. Categorize and prioritize requirements
10. Requirements inspection
11. Repository improvement

As the steps in this approach progress towards eliciting security requirements in step 8, the activities in the initial part of this framework are security goal definition, asset identification, security objective development, and threat and vulnerability identification. This top-down approach is facilitated effectively by beginning with defining security goals in step 1, whereas other approaches do not begin with them at all. Once the security goals and critical assets are defined, the methodology progresses to identifying threats and vulnerabilities and uses misuse cases and attack trees to develop artifacts. However, modeling is only added during steps 4 and 5 instead of the onset of goal definition. Establishing a model from the genesis of system development will substantially improve the effectiveness of the model by incorporating traceability into the security design processes.

A comparison of security requirements engineering methods [41]

This work provides a detailed summary of the primary methodologies for security requirement engineering. Fabian et al. highlight that the security requirement engineering process must support engineers in identifying the security goals of the stakeholders. The paper emphasizes that the process of converting stakeholder needs to requirements is extremely crucial to system success:

> *"This process must establish a coherent set of security requirements for the entire system, which is complete and consistent within itself and with the other kinds of requirements that are relevant for the system."*

The authors also discuss that security requirements are consequences of the identified threats to the system. And that all security requirements must be done before the design of the system, because of the influence that these requirements may have on the final design. Fabian et al. discuss that in the security requirement engineering community, the distinction between security goals (abstract) and security requirements (more detailed) is often not completely precise. The paper helps distinguish the difference between security goals and requirements in the following manner:

Security goals are defined as very general statements about the security of an asset. They are primarily established based on a stakeholder's expressed concerns about an asset. They are traditionally classified into *integrity*, *confidentiality*, and *availability* goals.

Security requirements capture security goals in more detail. They refer to a particular piece of information that refines one or more security goals.

The paper then summarizes different security requirement engineering frameworks, the relevant methods are summarized below:

Security quality requirements engineering methodology (SQUARE)

This is a comprehensive methodology for security requirement engineering. It is primarily used for software development and provides an organizational framework for carrying out security requirement engineering activities. It is comprised of 9 steps: 1. Agree on Definitions 2. Identify security goals 3. Develop artifacts 4. Perform risk assessment 5. Select elicitation technique 6. Elicit security requirements 7. Categorize requirements 8. Prioritize Requirements 9. Requirements inspection. This framework offers a very effective means of developing security requirements and follows the foundational elements defined by SREP, with validation built in as exit criteria for each step. However, there is an identified lack of resources or approaches in regard to eliciting security goals from stakeholders in an efficient and effective manner (Steps 1 and 2).

Misuse cases: UML-based approach

As opposed to the traditional use case approach in systems engineering, UML-based Misuse cases identify behavior not wanted in the system to be developed. Simply put, misuse cases are caused by misusers. This approach requires the modeling of an abbreviated use case diagram which is

composed of use cases and actors, as well as misuse cases and misusers. This approach helps brainstorm and conceptualize other security requirements and goals that must be addressed. The five steps of this iterative method are as follows:

1. Identify critical assets in the system
2. Define security goals for each asset
3. Identify threats for each security goal
4. Identify and analyze risks for the threats
5. Define security requirements

While the use of misuse case diagrams is very effective in this stage of system development, the approach does not cover requirement elicitation based on the misuse cases, ensuring all requirements are included, validation, verification, conflicting requirements, or the interplay between security and other non-functional requirements.

Engineering Security Requirements [19]

Firesmith's primary findings state that "Most Requirements engineers are poorly trained to elicit, analyze, and specify security requirements, often confusing them with the architectural security mechanisms that are traditionally used to fulfill them." They end up specifying design constraints rather than true security requirements. The common problem with specifying security requirements is that they tend to be accidentally replaced with security-specific architectural constraints that may unnecessarily constrain the most appropriate security mechanisms for the system. To combat this problem, Firesmith introduces 12 types of security requirements that are useful for guiding the requirements engineers and the process of eliciting precise security requirements. These 12 include:

- Identification Requirements
- Authentication Requirements
- Authorization Requirements
- Immunity Requirements

- Integrity Requirements
- Intrusion Detection Requirements
- Nonrepudiation Requirements
- Privacy Requirements
- Security Auditing Requirements
- Survivability Requirements
- Physical Protection Requirements
- System Maintenance Security Requirements

The author highlights that often security control mechanisms are selected as requirements and may unnecessarily over-constrain the system implementation. Therefore, the requirements must have clear guidelines and specifications, in order to not lead to any preemptive design decisions.

Security Requirements Engineering: A Framework for Representation and Analysis [42] Haley et al. claim that Security Requirements must satisfy three criteria: *definition*, *assumptions* and *satisfaction*. While the work is focused on software security requirements, the need for a systems approach is clearly highlighted. The paper discusses the importance of context in a requirement, i.e. confidentiality could be solved by protecting the actual data physically (locked door). They propose to generate security goals by first enumerating all system assets, then postulating actions that would violate security concerns for the assets, i.e. brainstorm threats and threat vectors. Then, a set of security goals can be generated by listing the actions that need to be taken to prevent (or avoid) the threats to the assets.

STRIDE - Threat Modeling Designing for Security [43] The STRIDE method was proposed by Microsoft, and represents a mnemonic for six different types of security threats that a system may encounter. STRIDE helps identify and categorize threats by focusing on six main dimensions: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. This structured approach helps security professionals analyze and mitigate potential security risks in software and system design by considering these threat categories. Each threat

category relates to a specific security theme that is violated. Table 1.1 outlines the relationship between the six threat dimensions and their associated security theme.

Table 1.1: STRIDE Threat Classification

Security Threat	**Security Theme**
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-Repudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

In simple terms, the STRIDE security threats are described in the following manner [44]:

Spoofing: Pretending to be someone or something you're not in the system.

Tampering: Changing or messing with real information.

Repudiation: Denying you did something in the system when you actually did.

Information Disclosure: Unauthorized access to secret data or a data breach.

Denial of Service (DoS): Stopping legitimate users from using a service.

Elevation of privilege: Gaining more access rights in the system when you shouldn't have them.

This framework plays a vital role in systems security engineering and modeling by providing a structured and systematic approach to identifying and categorizing security threats. This framework is particularly valuable for several reasons. Firstly, it helps security professionals and system engineers to comprehensively assess and classify potential risks and vulnerabilities associated with a system. Secondly, STRIDE establishes a common language for discussing and documenting these threats, which is essential for effective communication and collaboration among various stakeholders involved in system development. Furthermore, it assists in risk assessment, enabling organizations to prioritize and address the most critical security concerns. Once threats are iden-

tified using STRIDE, it facilitates the development of tailored security controls and mitigation strategies to address those threats effectively.

1.3.5 Key Automotive related literature

The automotive industry significantly influences security engineering by integrating advanced technologies into vehicles, leading to interconnected and data-driven ecosystems. With the increasing adoption of connected cars and autonomous features, security engineering and secure system design have become paramount to safeguard against cyber threats targeting vehicle systems. The industry's focus on developing standards such as the ISO/SAE 21434 (Need for cybersecurity) demonstrates its effectiveness and commitment to establishing robust cybersecurity practices, emphasizing the critical intersection of automotive innovation and security engineering. Therefore, key automotive-related literature that discusses secure system design will be reviewed below.

Case Study for Defining Security Goals and Requirements for Automotive Security Parts Using Threat Modeling [45]

This paper focuses on the importance of threat modeling in enhancing the security of automotive systems, especially in the context of connected cars and increasing cyber threats. It emphasizes the systematic identification of potential threats and the subsequent derivation of security goals and requirements to mitigate these threats. The paper utilizes the Microsoft threat modeling process, beginning with asset identification and use case functional definition, followed by threat identification using STRIDE to classify and identify potential threats. Once the threats were identified, the paper utilized the HEAVENS method to classify security risk levels. The following steps of the Microsoft modeling process were followed in the case study: Identify Assets, Create an Architecture overview, Decompose the Application, Identify the Threats, Document the Threats, and Rate the Threats. HEAVENS was introduced as a means of risk assessment, in which threat levels are assessed, considering factors like Expertise, Window of Opportunity, Knowledge, and Equipment. Simultaneously, the impact levels, including Safety, Financial, Operational, and Privacy, are evaluated. These combined threat and impact assessments led to the determination of security levels for

each threat, subsequently leading to the development of security goals and requirements. While the paper utilized a combination of best practices to effectively derive security requirements and goals, it proved the necessity of a centralized approach and model to relate each phase to the next.

Cyber Security Threat Analysis and Modeling of an Unmanned Aerial Vehicle System [46] This paper focuses on the security threats and challenges associated with Unmanned Aerial Vehicles (UAVs) and their communication networks. Using a goal-oriented approach, the authors conduct a high-level systematic risk assessment of a generic UAV. The approach begins with architectural definitions of the UAV and its communication module, followed by a threat and risk analysis of the system using the CIA classification methodology. The authors then develop a hierarchical model of the discovered threats to the UAV, and evaluate each risk using likelihood and impact criteria. Using a risk evaluation grid, each risk was evaluated using a prescribed numerical scale. This paper demonstrated the utility of combining architectural definition with risk analysis phase, and the combination of models and text-based approaches. The paper demonstrates the effectiveness of likelihood and impact as effective means of risk classification, however, the incorporation of security controls and requirements is never discussed. It is important to ensure that the outputs of a risk assessment provide streamlined feedback capabilities to security requirements and control elicitation processes.

Information Security Risk Management of Vehicles [47]

This paper applies the ISO 27005-defined Cybersecurity risk management process to assess the cybersecurity risks associated with modern vehicles. It defines "assets" within the realm of road vehicles as components significantly impacting road safety and personal information security. The authors categorize four vulnerability classes in vehicle protection, encompassing direct and indirect (USB) physical access, as well as close-range (Bluetooth) and long-range wireless access. Risk is defined as a product of threat probability and consequence and key risks are detailed for the power unit, chassis, electronic body systems, and comfort systems. The results of the risk management approach demonstrated that all but the comfort systems pose substantial cybersecurity risks. This

research offers a robust methodology for risk definition and classification, serving as a valuable foundation for vehicle risk assessment.

J1939

In order to apply, test, and analyze different systems engineering security methods and tools, a system must be selected on which to perform these tasks. The system of interest for the initial part of this work was the network protocols on Medium to Heavy Duty (MHD) vehicle networks, specifically the J1939 transport protocol. The following section discusses the system of interest as well as relevant literature that was instrumental in gaining an understanding of the network, its intricacies, and its shortcomings.

MHD vehicle mechanical operations, as well as data recording and display, are primarily controlled by Electronic Control Units (ECUs). These ECUs form networks within the vehicle to communicate information and commands to other entities via a bus topology. The ECUs within these vehicles utilize J1939 which is an application layer communication protocol built on top of the controller area network (CAN). The MHD community is invested in increasing the security of J1939 network communications.

Protocol overview:

The J1939 protocol is a set of Society of Automotive Engineers (SAE) standards that designate how various ECUs communicate using the CAN bus for MHD vehicles. This standard, consisting of multiple documents, corresponds to five out of the seven Open Systems Interconnection (OSI) layers. J1939-21 covers the data link and transport layer [1], while J1939-71 covers the application layer [48]. These two layers set the rules for constructing messages and accessing the bus, as well as what data is contained in each message sent onto the network.

The messages communicate data such as engine speed, wheel-based vehicle speed, and accelerator pedal position; a complete list can be found in J1939-71. This standardization of messages allows for communication between ECUs from different manufacturers.

Typical communication on the network falls into one of two categories: Destination-specific, or Broadcast messages. Destination-specific communication between two ECUs follows a specific

progression of messages. The first message by the sending party is a Request to Send (RTS) message that contains information about the amount of data (packets) and the sequence in which they will be sent. The receiving party then sends a Clear to Send (CTS) message that contains details on the amount of data to be sent in the following messages, according to the RTS information. The sending party can then send the information in data transfer (DT) messages until the transfer is complete. Lastly, the receiving party sends an acknowledgment message to confirm the reception of the data.

Broadcast messages are messages sent to all parties on the network (destination address 0xFF) containing information that can be used by many ECUs. A broadcast message is initiated by sending a broadcast announcement message (BAM) containing information about the number of packets and bytes to be sent. It is then followed by a single or multiple messages containing the data packets. The broadcast messaging format does not require CTS or acknowledgment messages from the receiving party. Communication can also be initiated from the receiving party by sending a destination-specific request message that details the information needed. The communication then follows the same structure.

Exploiting Transport Protocol Vulnerabilities in SAE J1939 Networks [49]

This paper presents five different cases in which shortcomings of the J1939 standard are exploited. The paper first introduces the SAE J1939 standard as well as the Controller Area Network (CAN) that is used on Medium to heavy vehicles. The paper then details the structure of J1939 messages, such as parameter groups, Parameter Group Numbers (PGNs), Protocol Data Units (PDUs), Source addresses, and destination addresses. The paper then explains how typical communication between ECUs on the network takes place. The paper then details previous work that discovered exploits in the J1949 protocol and explains that his paper will expound on these ideas, as well as validate some of the already established attacks. Chatterjee then details the experimental test setup that was used to test and validate the 5 exploits. The test bed consisted of 4 separate ECM and EBC configurations, with a laptop connected to the ECM via a can bus. The test-bed also consisted of the actual Kenworth truck that we have here at CSU. The following five attacks were conducted on

the test-bed and the results were reported: 1. Request Overload Attack (Validation) 2. Connection Exhaustion Attack (Validation) 3. BAM Block Attack 4. Malicious Clear to Send (CTS) Attack 5. Memory Leak Attack. Of the five attacks, the first two were successful on all test setups. The last three were only successful on one ECM configuration.

1.4 Objectives

Given everything discussed above, the objective of this work is to contribute to systems security research in hopes that I will progress the state-of-the-art and improve elements of systems engineering and Model-Based Systems Engineering that pertain to secure system design in the early stages of development using SAE J1939 and vehicle networks. Additionally, this thesis is written in hopes that it will consolidate information regarding Model-Based Security engineering, security requirements and controls, and Model-based security methods for outside reference. These objectives will be addressed through accomplishing the research tasks associated with each research question discussed above.

1.5 Overview

This thesis is divided into six chapters:

- Chapter 1 provided an introduction to the work, including the motivation for the work, an outline of the research questions, background on MBSE, and a literature review of related standards and research.

- Chapter 2 focuses on the utilization of Model-Based Systems Engineering (MBSE) through SysML simulation to better understand and enhance the security of complex systems, specifically in the context of the J1939 protocol used in heavy-duty vehicles. It introduces the J1939 protocol, explores known vulnerabilities, and presents two attack models. The chapter highlights the benefits of MBSE simulation as a powerful tool for collaboration, brainstorming solutions, and bridging the gap between stakeholders and subject-matter experts.

- Chapter 3 focuses on the application of the Model-Based Systems Engineering method; MBSEsec to secure the J1939 network protocol on the Lab's research truck. The chapter outlines how MBSEsec is used to develop security requirements, allocate assets, model threats and risks, and create security controls for the J1939 protocol. It explores the benefits and shortcomings of the method, emphasizing the iterative nature of the method, the value of using MBSE for security analysis, and the recommended inclusion of an "attacker" element to enhance security measures.

- Chapter 4 introduces significant updates and enhancements to the MBSEsec methodology through the application of the MBSEsec methodology to a new System of Interest. The changes focus on a more robust risk classification approach, incorporating qualitative and quantitative elements, as well as traceability, simulation, and the iterative refinement of risk criteria and control effectiveness, offering a dynamic framework to make informed design decisions and improve overall system security posture.

- Chapter 5 introduces the EGRESS method as a novel approach to generating security goals during the conceptual design phase of Cyber-Physical System (CPS) design. This method combines elements of STPA, MBSEsec, and other approaches, leveraging MBSE as a foundation to streamline security goal elicitation and identification.

- Chapter 6 concludes the thesis with a restatement of the abstract, contributions, and lists some future works for project improvement.

1.6 Contributions

In light of the objectives and having followed a structured approach, the principal contributions of this thesis are:

- Developed SysML models that reflect J1939 protocol network attacks.

- Presented at Western States Regional Conference 2023 (WSRC) on: *A Demonstration of MBSEsec Applied to Securing J1939 Protocols on Heavy Vehicle Networks* (Thesis Chapters 2 and 3)
- Paper accepted and in publishing through IEEE Aerospace Conference 2024: *A Demonstration of MBSEsec Applied to Securing Cyber-Physical System Communications* (Thesis chapter 3)
- Co-Authoring a Journal Paper: *(EGRESS): Eliciting Goals for Requirement Engineering of Secure Systems* (Thesis chapter 5)
- Developed SysML models that support both MBSEsec and EGRESS papers, as well as secondary models applied to the MRZR system.
- Developed a EGRESS template overlaid on top of the MagicGrid framework in SysML

Chapter 2

Modeling and simulation of J1939 Transport Protocol Attacks

2.1 Introduction

Often in the field of systems engineering, modeling is performed without simulation. Typically, modeling acts as a validation mechanism to bring stakeholders, end users, engineers, and business holders into agreement. These models serve as static architectural blueprints that act as a source of truth and reference throughout the system life cycle [50]. Simulation typically finds its utility in validating the effectiveness of a model, however, little research has been conducted in the evaluation of the benefits and effectiveness of MBSE for this step in system design. Zeigler et al. [51] highlight that the absence of proper modeling and simulation infrastructure creates gaps in our understanding of complex engineering systems. These gaps result in unexpected behaviors and the discovery of unexpected vulnerabilities. To address this issue, efforts in literature are in progress to better integrate Model-Based Systems Engineering (MBSE) with advancements in model-based simulation.

While the question of the utility of model simulation to validate models remains to be addressed in the literature, this work will focus on the utility of simulation through model-based tools to better understand complex systems. As mentioned before, early system design typically follows a top-down approach, which starts with a high-level overview of a system, breaking it down into smaller components, and refining the details progressively. However, when dealing with securing systems that have already been fielded and have vulnerabilities that have been discovered and exploited, a bottom-up approach is warranted in order to develop a detailed understanding of the exploited components. These types of systems typically require the implementation of a security control, which is defined as a safeguard or countermeasure prescribed for an information system designed

to protect the confidentiality, integrity and availability of the asset's information and to meet a set of defined security requirements [52].

The implementation of effective security controls in a cyber-physical system typically involves consulting subject matter experts as part of an Integrated Product Team, who can brainstorm effective solutions at the component level. Effective systems engineers must be able to translate these solutions into well-documented security requirements and controls.

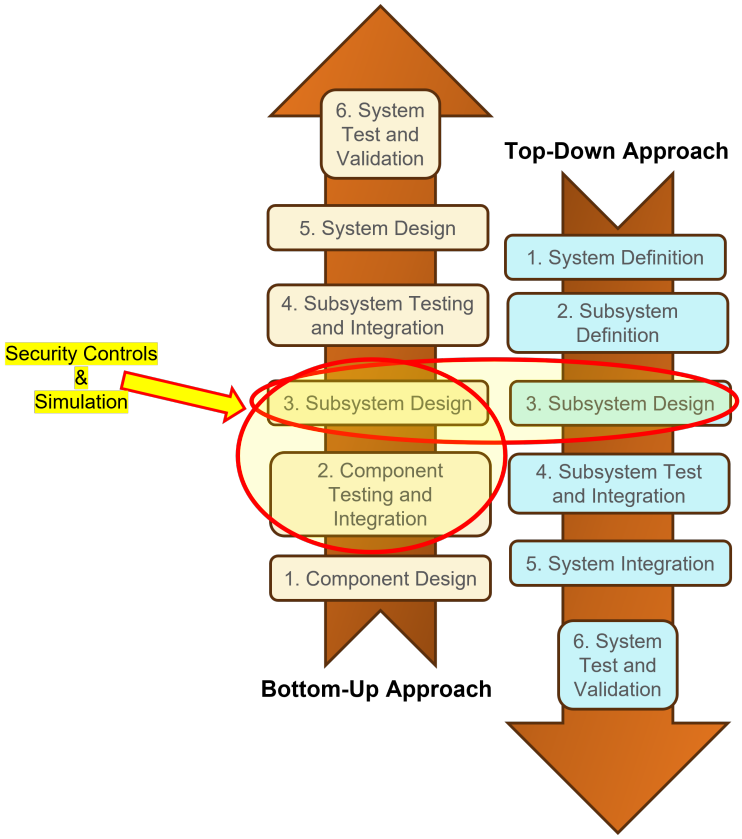

Figure 2.1: Chapter 2 within Bottom Up and Top Down approaches

The top-down approach to system security design allows for the functions to dictate the form, where the overall security-related functions of the system will dictate the design and implementation of secure components. This approach first investigates *what* the system intends to do and the *how* follows during the subsystem and component design. In contrast, the bottom-up approach focuses on securing the components and subsystems, which then iteratively are brought together to

make the system functional. This is typically how security is conducted in system design because securing system functionality before the *form* is difficult. However, it is difficult to obtain holistic security implementations when beginning with component design. Systems engineers typically lack a bottom-up understanding of complex systems and therefore rely on the input of subject matter experts to provide the details. This chapter explores the use of MBSE as a tool for model simulation to better understand complex systems with the goal of enhancing collaboration between system engineers and experts in the field of security and merging the gap between these two approaches. Fig 2.1 highlights where this work fits within the context of the top-down and bottom-up approaches to system design. As can be seen, using simulation to better understand a system of interest converges at the subsystem and component level, a point where systems engineers typically lack a detailed understanding, and where a lot of design decisions are made.

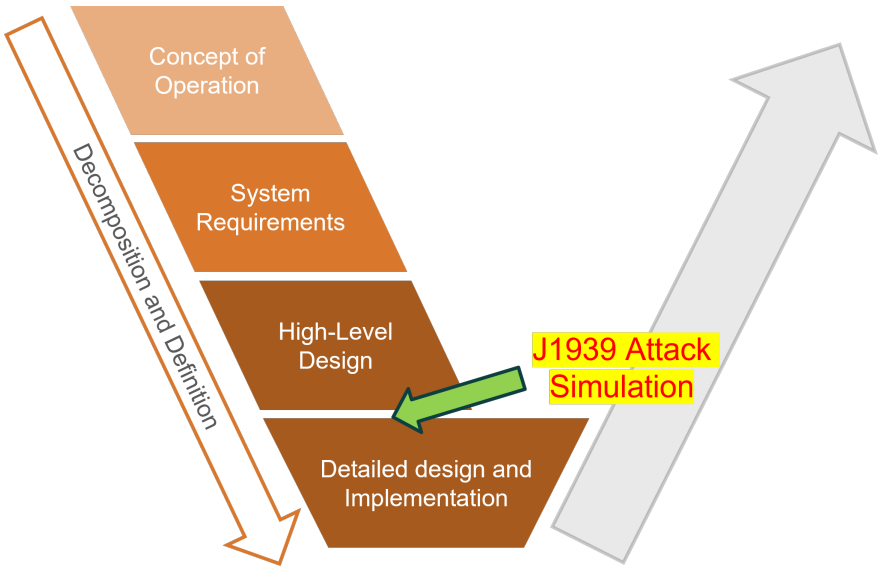

Figure 2.2: Location of work within SE V-diagram

The utility of simulation to enhance the understanding of systems engineers will be investigated by modeling attacks on the J1939 network protocol that have been validated on the Lab's research truck. Fig 2.2 elaborates where this chapter fits in the systems engineering V-diagram. As can be seen, the target of this model is the design and implementation of a network protocol within the

truck network. The goal is to use MBSE simulation to better understand the details of a system whose implementation has flaws, leading to exploited vulnerabilities.

2.2 Understanding the System of Interest

2.2.1 Building context for the SOI

In order to better understand the role of J1939 within the truck as a whole, and capture the context in which the protocol is used, a block definition diagram was developed in SysML to identify what is external to the system that may either directly or indirectly interact with the system. The block definition diagram for the Truck Domain in Figure 2.3 defines the Vehicle and the external systems, users, and other entities with which the vehicle may interact. This diagram helps frame the big-picture context as well as the top-level blocks (elements for the model) in which the simulation takes place. As can be seen, the truck domain is generally comprised of vehicle occupants, the physical environment, the truck itself, and external actors involved with the maintenance of the vehicle.

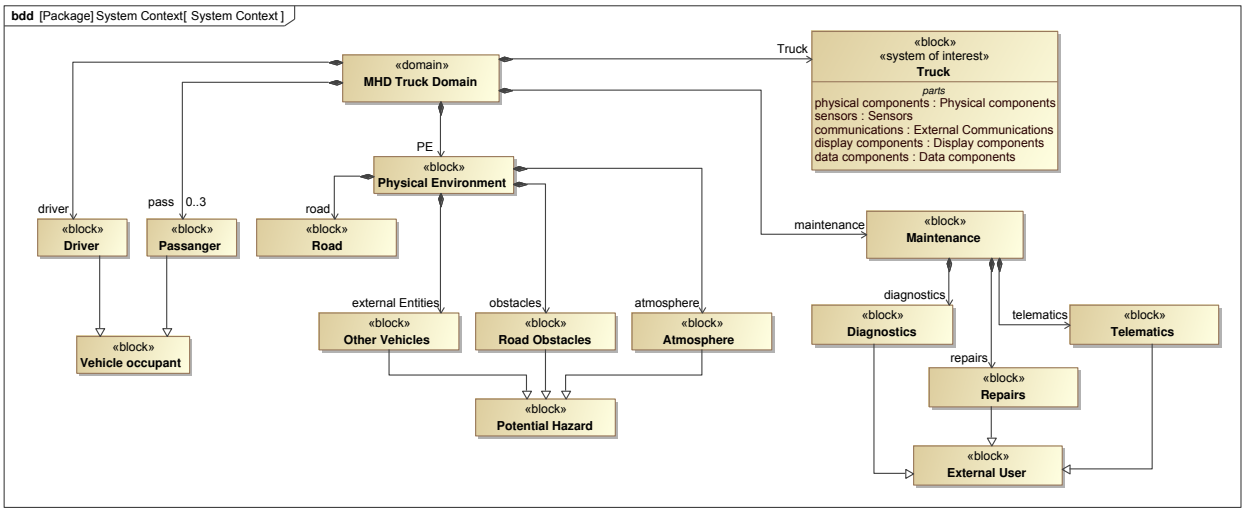

Figure 2.3: Block Definition Diagram of Truck Domain

The components of the domain are generalized into three categories; *vehicle occupants*, *potential hazard*, and *external user* which provides a hierarchy to the system domain. Developing

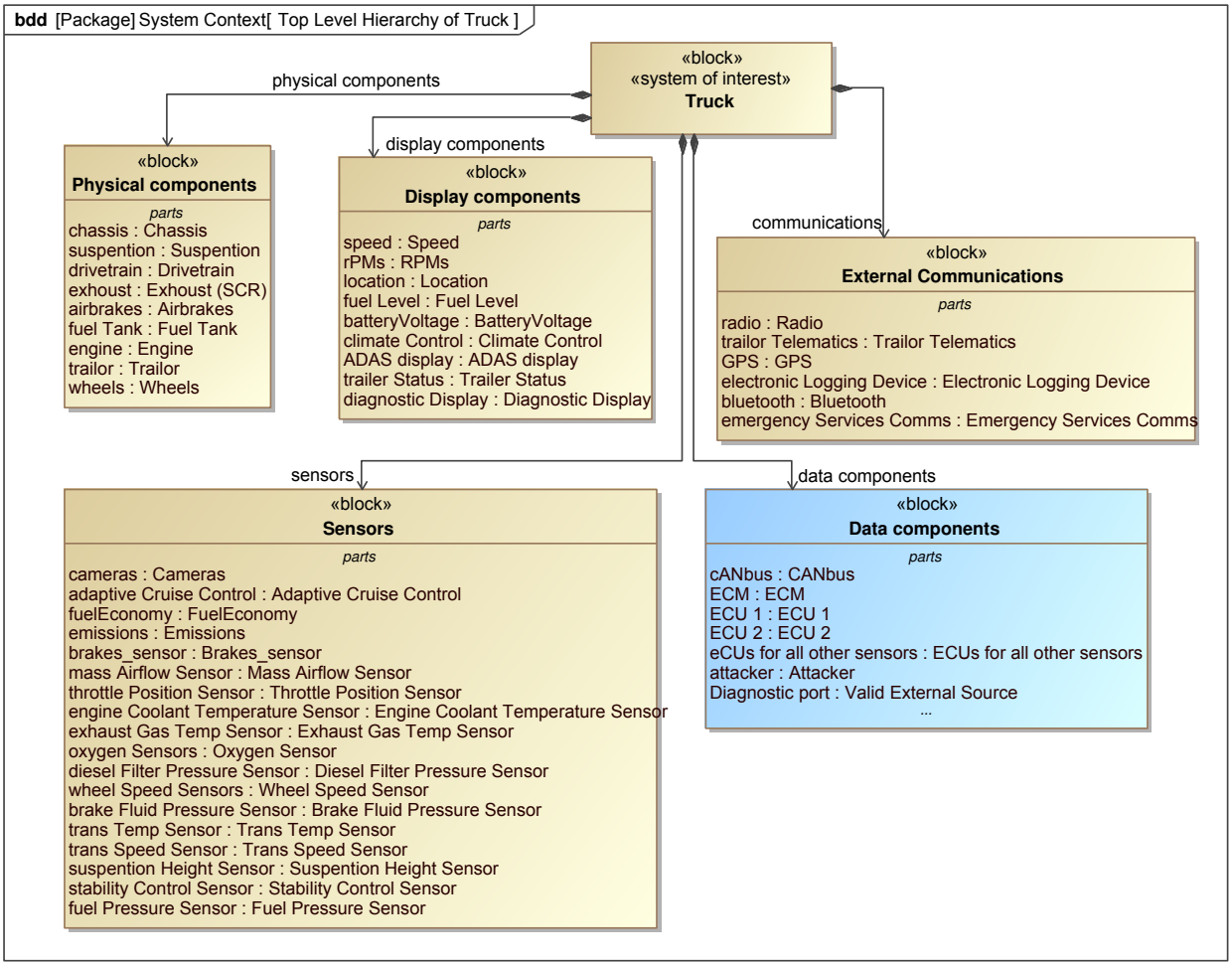

Figure 2.4: Top Level Hierarchy of Truck

a context architecture enables engineers and stakeholders to consider actors of interest that may either directly or indirectly interact with the system, as well the environment in which the system is designed to operate. This step can be considered 'complete' when all relevant elements external to the system have been identified.

A block definition diagram was then developed to capture the top-level hierarchy of the truck domain (Fig 2.4. This diagram includes a top-level block called *Truck* which provides the context for the other blocks in the diagram. The *Truck* block is decomposed into functional components that divide the system into the following subsystems: *Physical components, Display components, External communications, sensors,* and *Data components*. This diagram specifies the blocks and their interrelationships, which depicts multiple levels of the system hierarchy from the top-level

domain or context block down to the blocks representing the vehicle components [53]. As can be seen in Figure 2.4 the system of interest is highlighted in blue. This block is comprised of all the components of the truck that deal with data transmission using the J1939 protocol. It is important to note that the system of interest was simplified for the sake of the simulation. This model elaborates on the communication between two ECUs, although a typical MHD vehicle may have up to 50 ECUs on its network. Since these simulations are designed to replicate the effect of an attacker on a truck network, the element *attacker* was added to demonstrate the presence of an attacker within the truck network. Practically, in order to reproduce the actions that an attack may perform on the network, a block with this title was appropriate, therefore the attacker can be seen as one of the *parts* within the *data components* block. In order to construct a simulation that replicates the attacks conducted on the J1939 protocol of the tuck, it is important to first understand the system of interest at a level that enables a detailed simulation.

2.2.2 J1939 Protocol

The J1939 protocol is a set of Society of Automotive Engineers (SAE) standards that designate how various electronic control units (ECUs) communicate across a Network. This protocol is widely used in heavy-duty vehicles and industrial applications, facilitating standardized communication between electronic control units over the controller area network (CAN). This protocol defines a comprehensive set of parameter groups and messages for various vehicle functions such as engine control, transmission, diagnostics, and more. J1939 enhances interoperability and diagnostics, contributing to efficient and reliable operation in complex vehicle and machinery systems

This standard consists of multiple documents, corresponding to many of the Open Systems Interconnection (OSI) layers. The Open Systems Interconnection (OSI) is a conceptual framework that standardizes how different networking protocols interact and communicate. It is divided into the following seven layers:

1. Physical Layer: The lowest layer of the model and deals with the physical transmission of data over the network.

2. Data Link Layer: Manages the framing, addressing, and error control of data packets. It ensures reliable point-to-point or broadcast communication.

3. Network Layer: Responsible for addressing, routing, and forwarding data between different parts of the network.

4. Transport Layer: Ensures reliable end-to-end communication, including error detection and correction as well as sequencing of data.

5. Session Layer: Manages the establishment, maintenance, and termination of communication sessions between two entities.

6. Presentation Layer: Responsible for data translation, encryption, and formatting. This layer ensures that the data exchanged can be understood by both parties

7. Application Layer: The top layer of the framework, is responsible for the interaction between software applications and the transmitted data.

J1939-21 covers the data link and transport layer [1], while J1939-71 covers the application layer [48]. These three layers set the rules for constructing messages and accessing the CAN bus, as well as what data is contained in each message sent onto the network. The messages communicate data such as engine speed, wheel-based vehicle speed, and accelerator pedal position; a complete list can be found in J1939-71. This standardization of messages allows for communication between ECUs from different manufacturers. Figure 2.5 illustrates the relation of the relevant J1939 documents to the overarching OSI model.

According to these documents, the network communication process can be categorized into two distinct types: Destination-specific communication and Broadcast communication. In the context of destination-specific communication between two Electronic Control Units (ECUs), a structured message progression is followed to ensure effective data exchange. Initiating this sequence is the transmitting ECU, which dispatches a Request to Send (RTS) message containing crucial metadata such as data packet count and sequencing specifics. This RTS message serves as the foundation for the subsequent data transfer. Upon reception of the RTS, the receiving ECU responds with a Clear to Send (CTS) message. The CTS message functions as an acknowledgment of the RTS receipt and

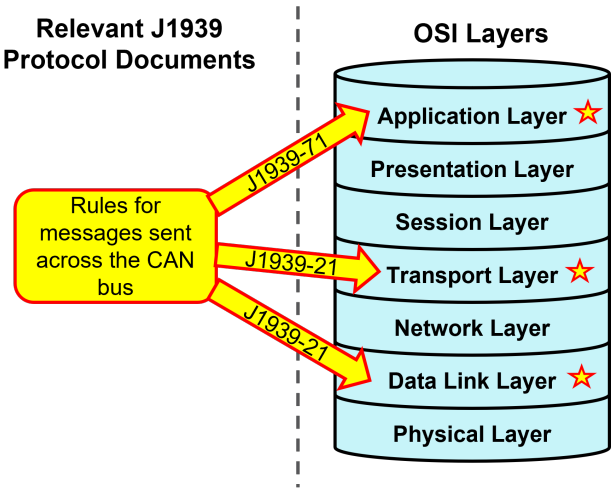

Figure 2.5: J1939 Layers in relation to OSI model

delineates the specific data quantity that the receiving ECU is prepared to accept. Subsequently, the transmitting ECU commences the transmission of data through a series of data transfer (DT) messages.

These DT messages encapsulate the actual data payload and are transmitted in accordance with the specifications outlined in the CTS message. The receiving ECU systematically reconstructs the incoming data packets to reconstitute the original information. Significantly, if the receiving ECU anticipates the need for additional data, it can continue the CTS-initiated interaction by sending further CTS messages. These successive CTS messages convey the required data quantity, allowing the transmitting ECU to progressively transfer more data through additional DT messages. Upon the successful completion of data transmission, the receiving ECU concludes the communication sequence by transmitting an acknowledgment message. This acknowledgment serves as a formal confirmation of the accurate reception of the transmitted data, aligning with the guidelines specified in the J1939-21 document. Figure 2.6 illustrates the communication in the form of a sudo-sequence diagram taken from the J1939 specifications.

Broadcast messages encompass communications disseminated to all network participants (address 0xFF), providing information relevant to multiple Electronic Control Units (ECUs). The broadcast messaging process entails a systematic sequence to facilitate effective data distribution. The broadcast sequence commences with the transmission of a broadcast announcement message

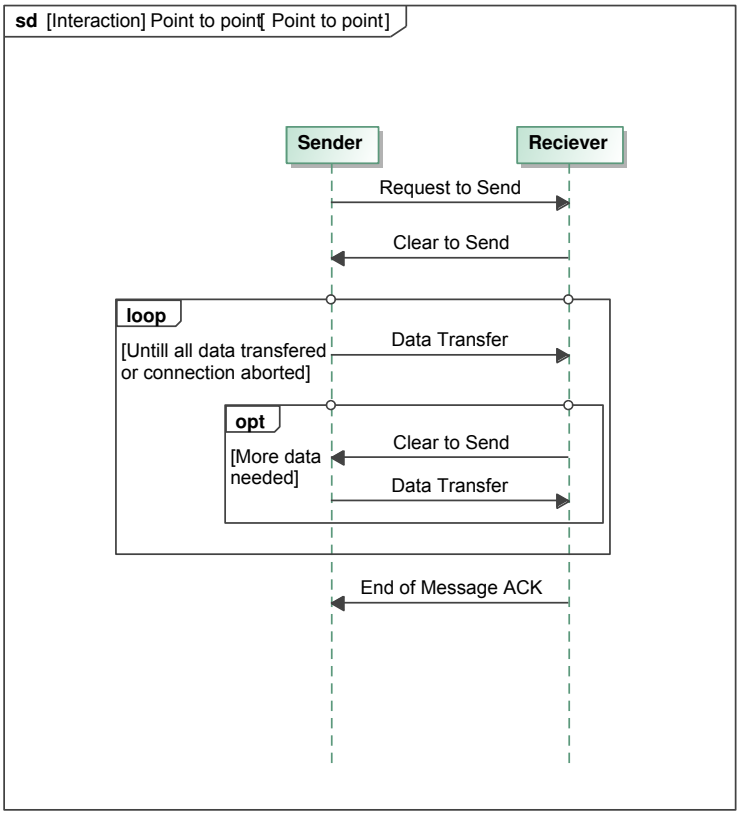

Figure 2.6: Error free point to point communication [1]

(BAM) by the initiating ECU. This BAM incorporates vital data, including the count of data packets and the total bytes slated for transmission. Subsequently, a series of one or more messages, carrying the actual data packets, follow the BAM. Unlike the destination-specific communication process, the broadcast messaging protocol deviates in that it does not necessitate the involvement of Clear to Send (CTS) or acknowledgment messages from the recipients. This divergence simplifies the broadcast communication flow, contributing to the efficient dissemination of shared information. It's important to note that communication initiation can also stem from the receiving party. In such cases, a destination-specific request message is transmitted, outlining the specific information required. Following this request, the communication unfolds along the same structured lines as previously detailed. Figure 2.7 illustrates typical broadcast communication in the form of a sudo-sequence diagram taken from the J1939 specifications.

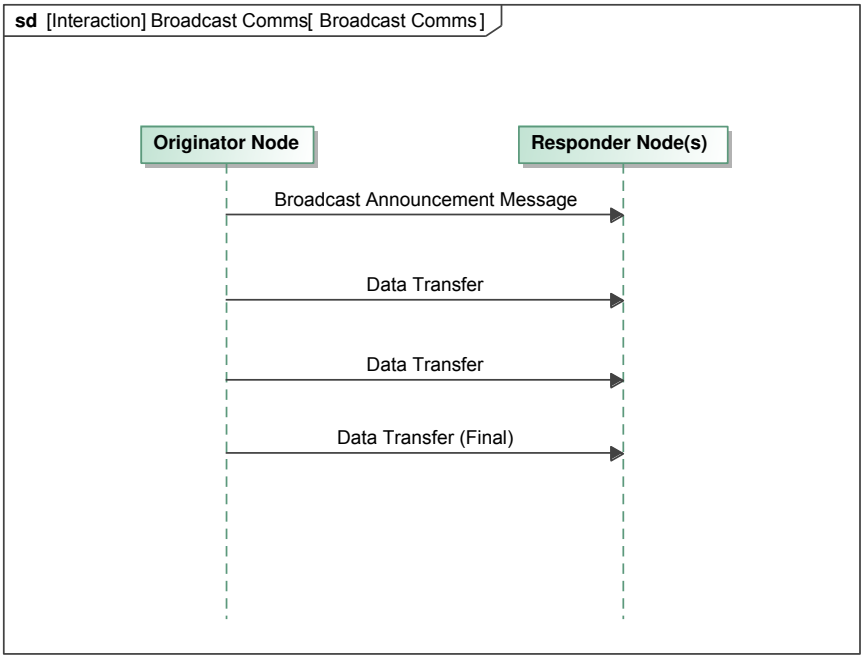

Figure 2.7: Error free Broadcast communication [1]

2.2.3 J1939 Exploits

Currently, the industry standard for connecting to in-vehicle networks of MHD vehicles is through the J1939 Diagnostics port which allows for direct access to the vehicle bus. The consequence of this design is that limited access control can be applied to regulate the types of entities that connect to the network. Moreover, the CAN bus is a broadcast medium without the ability to identify valid senders [54]. There is no inherent security on a typical CAN bus. Once accessed, either via a physical connection to the J1939 port or through wireless connections like telematics, an attacker can read J1939 data, send messages, and send commands to the ECUs. When obtaining access to the network, the attacker can claim any approved source address, which gives it the ability to impersonate important entities such as the engine control module (ECM). It is also possible for the attacker to send any J1939-compliant message across the network with total control over all parameters of the message [55].

Attacks have been discovered at multiple layers of network communication, primarily the application layer, network management layer, and data-link layer. Chatterjee et al. [49] present five

different cases in which shortcomings of the J1939 data-link layer can be exploited. Using a robust test bed, the five exploits were validated. The five attacks are detailed below:

1. *Request Overload Attack* - The J1939 transport protocol specifies that direct request messages to an ECU must all be processed. This attack involves overloading a target ECU with request messages all containing the same parameter group number. Since the ECU is attempting to process all of the incoming messages, it fails to perform its normal operations.

2. *Connection Exhaustion Attack* - The J1939 standard states that only one connection for multi-packet data transfer can be established at any time for an ECU. The attack takes advantage of this specification and denies legitimate entities access to the ECU by maintaining a spoofed open connection with an ECU and sending messages periodically.

3. *BAM Block Attack* - The attack takes advantage of the fact that the standard specifies that ECUs must respond to destination-specific requests. It is executed by sending request messages for data that would otherwise be a periodic broadcast message to the network. By doing this, other parties are denied the data.

4. *Malicious CTS Attack* - This attack is initiated by sending a CTS message that specifies a different amount of data to be sent than was indicated in the RTS message, meaning the sending ECU receives a different value of data packets than anticipated. Due to the information not aligning, the ECU enters an unknown state.

5. *Memory Leak Attack* - This attack involves using the CTS message to request more data packets that an RTS message specifies, hoping for the sending party to 'leak' more data than intended.

The primary effects of these attacks lead to one of two outcomes; a denial of service (DoS) to the ECU, or an ECU processing (memory) malfunction. Both of these effects lead to a degraded functionality of the ECUs and therefore are effective at damaging vehicle functionality. The Request overload attack and connection exhoustion will be used as vulnerabilities that will be simulated in SysML to further understand the system of interest. It is also important to note that

other vulnerabilities exist on other layers of the protocol, but this work focuses on securing the J1939 Transport protocol, and on these verified and effective attacks.

2.3 Model Based Simulation development

2.3.1 Model Development

The system context diagrams for both the truck domain and the top level hierarchy of the components were used as the foundational models for the network simulations. The *data components* block within the block definition diagram was elaborated through the use of an internal block diagram, which describes the structure of the *data components* block in terms of how its parts are interconnected [53]. Using internal block diagrams, interface relationships between the components are established. Data *flow* is represented by the connections established between the ports on each element.

Connection Exhaustion Attack

Once an architecture was established, and the appropriate components were identified, the model was elaborated to describe the behavior of a system during a connection exhaustion attack. Figure 2.8 illustrates an elaborated internal block diagram. The top of the diagram consists of three system elements *ECU 1*, *ECU 2*, and *Attacker*, which are all connected through *ports* and *connectors*. The ports are illustrated as small squares on the boundary of the parts to represent an access point through which the elements interface. The behavior of these three blocks are described through state machine and activity diagrams illustrated below each block. It is important to note that in order to simulate an attack, it was only necessary to have three primary elements; two ECUs and an attacker.

The attack modeling began with building state machine and activity diagrams that describe normal ECU traffic on the network. Nominal traffic consists of typical back-and-forth information sharing among the ECUs that allow the truck to function normally.

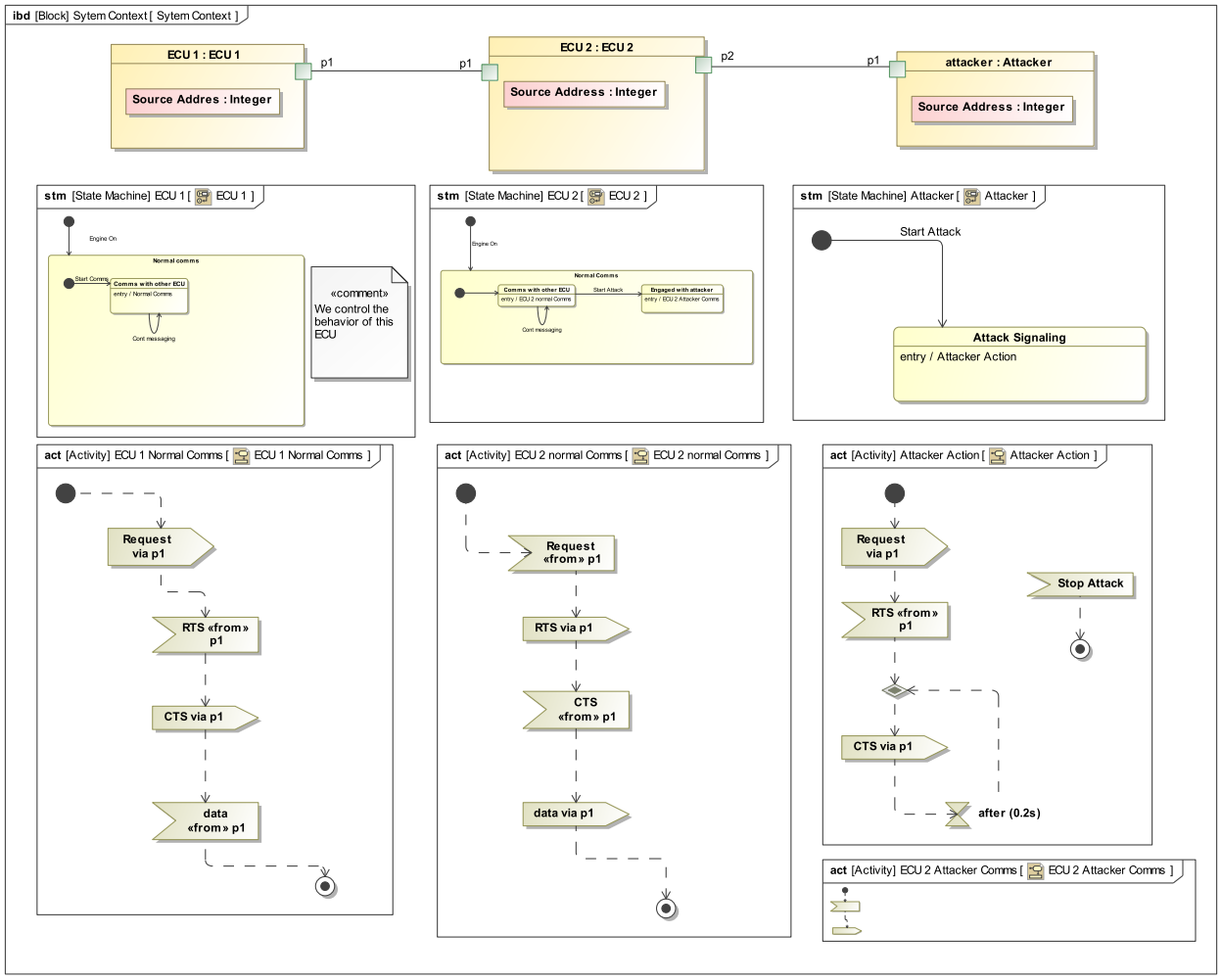

Figure 2.8: Elaborated Internal Block Diagram

State machine diagrams were first established to outline the different states in which the two ECUs could be. Normal communication consisted of both ECUs remaining in a *comms with other ECUs* state, in which typical network traffic took place. These states were characterized by activity diagrams that specify the sequence of messages that would be sent during normal communication. As can be seen in Figure 2.8, the state machines entered an initial state once the simulation began, these initial states would trigger the message sequence detailed by the appropriate activity diagram.

The implementation of the *Attacker* element was done through the establishment of a flow relationship (through a connector) between the *Attacker* and *ECU 2*. In the case of this model, the attacker was strictly bound to communicate only with the second ECU. Similar to the two ECUs,

the attack behavior was characterized by a state machine and activity diagram. The activity diagram outlined the sequence of actions that the attacker completes during a connection exhaustion attack.

The systems model was designed to auto-generate a sequence diagram that documents the messages sent across the network throughout the simulation. This was done in order to produce an artifact that accurately represents the model's behavior. As can be seen in Figure 2.9, The sequence diagram begins with normal network traffic between the two ECUs. Once the attack is initiated, the *Attacker* element establishes a connection with *ECU 2* and exhausts the connection through repeated CTS messages.

The benefits of this sequence diagram are twofold, first, it verifies that what was modeled actually takes place and that the sequence of messages and behavior is in order. Second, it provides an effective means of documenting the attack, which can be used as a tool to educate stakeholders and team members who are not SMEs in the field of truck networks.

Request Overload Attack

Using the same foundational model as the connection exhaustion attack, a second internal block diagram was developed for the Request overload attack by elaborating the *data components* block. In the case of this model, the internal structure of the network was simplified to three components; *ECU 1*, *Attacker*, and *Bus*. The objective of this attack is to overload a target ECU with request messages all containing the same parameter group number, which would overwhelm the receiving ECU, and cause it to malfunction. Therefore, the model was built to first demonstrate simple network traffic leaving a single ECU, then an attacker action was implemented. Figure 2.10 illustrates the internal block diagram as well as the state machines and activity diagrams that characterize behavior. This figure also illustrates how the model visually progresses through the simulation which can be seen through the green red and yellow status of different elements.

It is important to note that the simulation was kept simple and was designed to demonstrate the attack sequence through the development of a sequence diagram (Figure 2.11), therefore only the elements required to generate an appropriate sequence diagram were used.

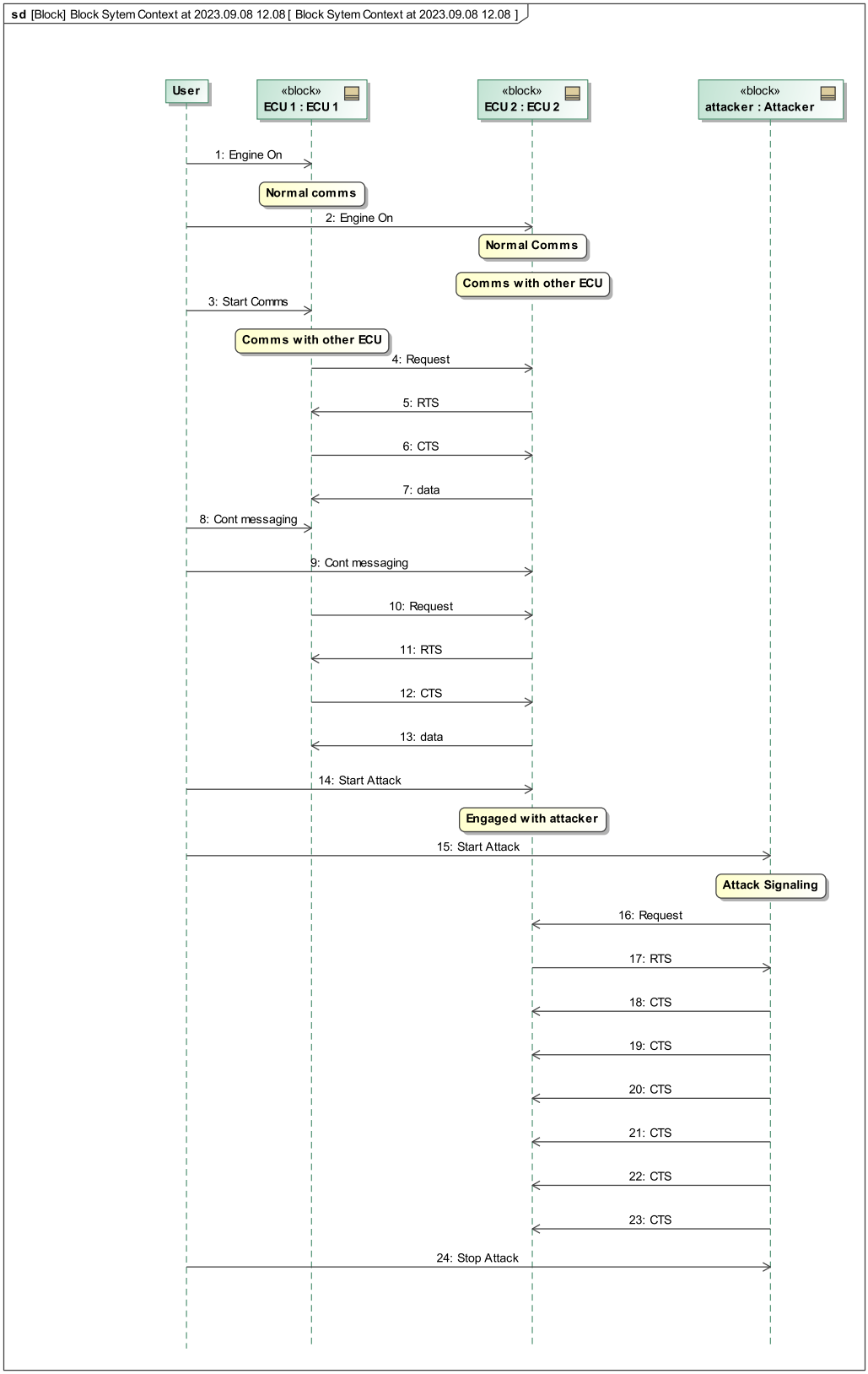

Figure 2.9: Auto Generated Sequence diagram of Connection Exhaustion Attack

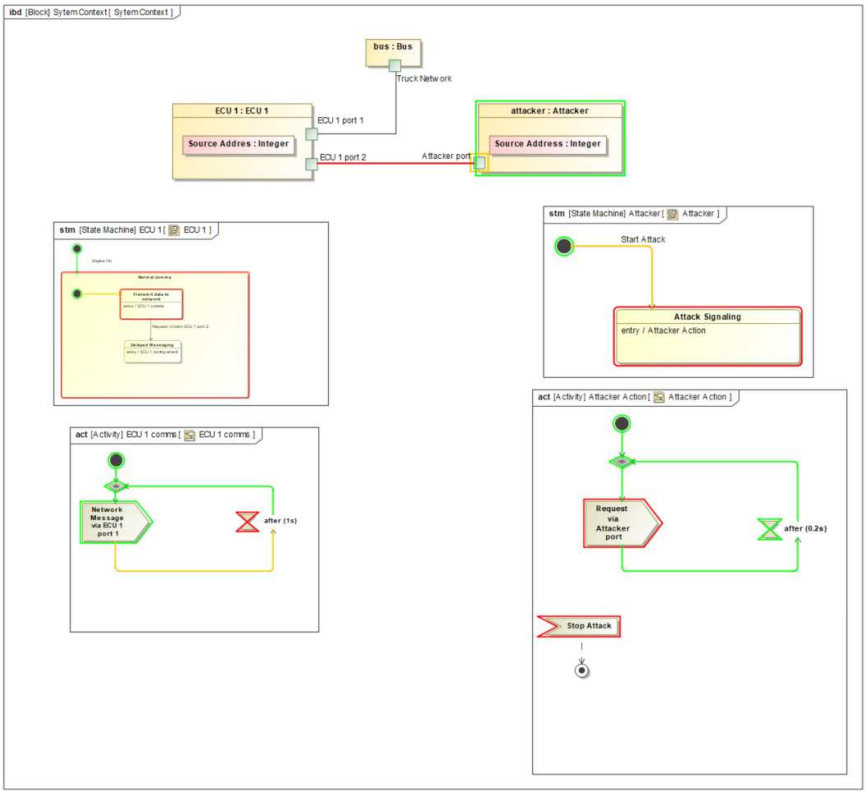

Figure 2.10: Screen capture of Internal Block Diagram of Request overload attack mid-simulation

2.4 Anticipated benefits and conclusions

This chapter explored the use of MBSE simulation as a tool for enhancing collaboration between systems engineers, stakeholders, and experts, as well as its role in developing secure systems. The convergence of top-down and bottom-up approaches during the component and subsystem design highlighted the need for tools to enhance collaboration between system engineers (Top-down), and subject-matter experts design engineers (Bottom-up). MBSE simulation was explored as a viable tool for meeting this need. In this chapter, two context diagrams were built in order to better understand the role of J1939 within the truck system of systems. Once the context was defined, and the appropriate components that rely on J1939 were identified, two models were built which replicate two attacks that exploit network vulnerabilities.

Understanding that the future of systems engineering is model-based, this SysML simulation tool served as an effective means of demonstrating a problem within a system. Especially in the field of network protocols, which deal with message sequences and delicate time intervals,

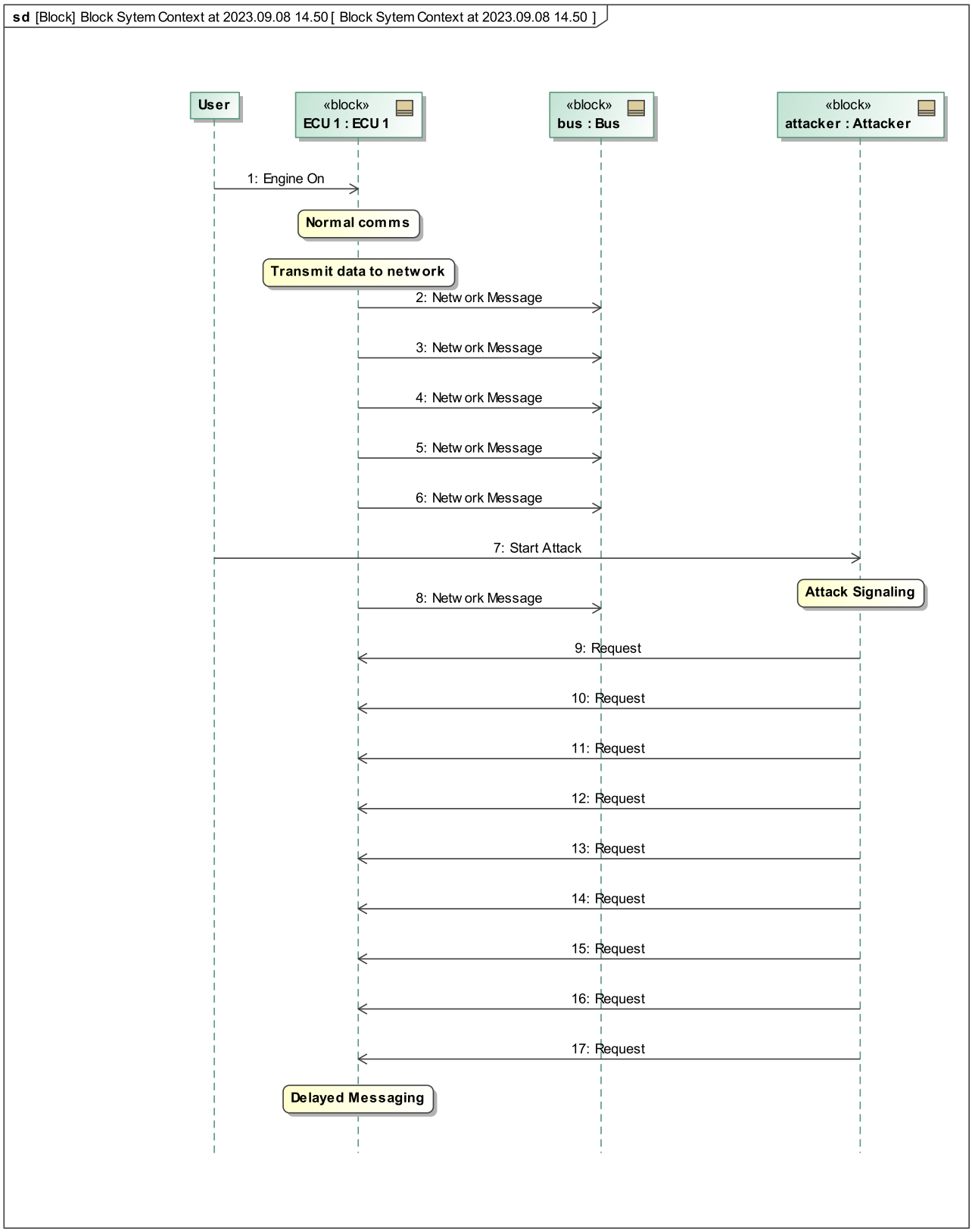

Figure 2.11: Auto-generated sequence diagram of Request overload attack

these models served as a method of visually highlighting the issues with the protocol, with activity diagrams to detail why certain network behaviors lead to such attacks. In the case of the connection exhaustion attack, the simulation demonstrated how an attacker can establish a connection with an ECU and maintain the connection for extended periods of time.

These models serve as dynamic and interactive visualizations, enabling stakeholders to gain deeper insights into potential threats and their ramifications. Importantly, they provide decision support capabilities, enabling teams to make informed regarding system security. Specifically, our simulations excel at illustrating attack scenarios with a high degree of realism. The sequence diagrams accurately mimic network behaviors and vulnerabilities, offering clear and interactive engagement with all parties involved. Additionally, activity diagrams were used to describe why certain network behaviors lead to such attacks, making difficult network concepts and attack mechanisms accessible to both technical and non-technical stakeholders. For instance, in the case of the connection exhaustion attack, the simulation demonstrates how an attacker can establish and maintain a connection with an ECU for extended periods, shedding light on the critical network behaviors that lead to such threats.

Another benefit to SysML simulation is its use as another tool to enhance collaboration and brainstorming for solutions to the demonstrated issues. These models serve as dynamic platforms where cross-functional teams of system engineers, subject-matter experts, and stakeholders can come together. These simulations, with their interactive and visual nature, encourage stakeholders to actively engage in discussions. They provide a shared context where team members can collectively explore potential solutions, experiment with various strategies, and analyze their effects on system behavior. The iterative nature of MBSE allows for a hands-on, solution-oriented collaborative environment that contributes to the generation of innovative and effective countermeasures. By leveraging SysML simulation in this manner, organizations harness the collective intelligence of their teams, ultimately leading to more robust and secure system designs.

Lastly, a notable perceived benefit of MBSE simulation is its ability to bridge the gap between stakeholders and subject-matter experts (SMEs) at the critical juncture where top-down

and bottom-up approaches converge. In complex system development, stakeholders often bring high-level requirements and strategic goals (top-down), while SMEs possess detailed technical knowledge and insights (bottom-up). This convergence point can be a point of potential miscommunication and misalignment. MBSE simulation, with its capacity to create visual and interactive models, provides a common language and platform where these two groups can collaborate effectively. Stakeholders gain a deeper understanding of technical intricacies, and SMEs can grasp the broader strategic context. The simulations serve as a bridge, translating high-level objectives into actionable technical requirements and, conversely, illustrating how detailed technical decisions impact overarching system goals. Having used MBSE to explore the vulnerabilities through simulation, it will now be used to generate solutions through structured modeling following a methodology.

Chapter 3

Investigation of MBSEsec method for secure system development

3.1 Introduction

As stated in the introduction to this work, the objective of this thesis is to investigate the application of MBSE in the early design process, specifically in relation to securing vehicles and vehicle networks. In this chapter, this will be addressed through the application of an MBSE method for secure design. The second research question in this work is "What methods exist that incorporate MBSE to develop more secure systems?", the first part of this question was addressed through a literature review in section 1.3.3 of existing MBSE methods for security. In this chapter, the selected method is applied to two separate systems, the first is detailed in the previous chapter, the J1939 transport protocol. The second system to which this method was applied was a Defense Advanced Research Projects Agency (DARPA) remote vehicle.

In the context of the system development lifecycle, Figure 3.1 illustrates where this work falls on a System-V diagram. At the point where system requirements are being refined and established, and a preliminary architecture is being drafted for the system.

A significant shift has occurred in the last 50 years, with functionality being largely delivered previously through mechanical actuation to a high reliance now on software and electrically implemented functionality [56]. This functionality change has resulted in a majority of complex systems now being Cyber-Physical Systems (CPS), where the mechanical physical functionality is largely delivered or controlled via an electronic, cyber-based approach. The push for the interconnection of devices increases functionality and convenience features, but these also increase the vulnerability profile [57]. Aerospace and vehicle-based systems are key examples of cyber-physical systems that need a systems approach to master the complexity of interactions and secure functionality

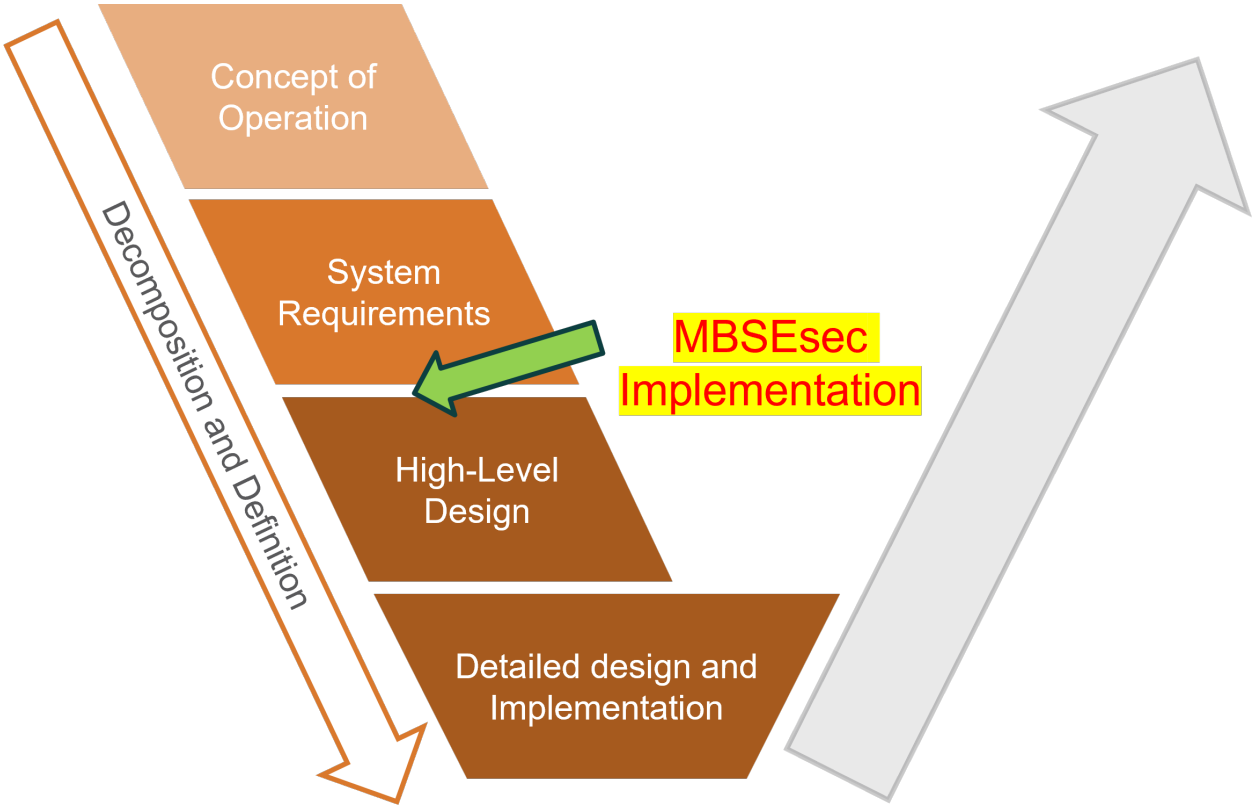

Figure 3.1: Where MBSEsec fits into the system life cycle

in the current digital environment. This chapter will explore the use of the model-based systems engineering method MBSEsec to secure such cyber-physical systems.

3.2 Application of MBSEsec to J1939

The J1939 protocol is implemented as an enabling 'subsystem' on the truck network and drives many of the network requirements and ECU message handling behavior. Therefore, the use of a method to secure this protocol falls between system requirements and high-level system design, as illustrated in Figure 3.1.

As stated previously, the motor vehicle industry is increasingly interested in the cybersecurity of vehicles, both of automobiles and heavy trucks. As with most complex CPSs, such as airplanes and aerospace vehicles, road vehicles communicate over an internal physical network infrastructure. In the field of Medium and Heavy duty (MHD) road vehicles, the primary means of software

communication is through the Controller Area Network (CAN), as highlighted in the previous chapter.

The success of CAN technology in the automotive industry has led the aerospace sector to embrace CAN as a viable solution for secure, reliable, and cost-efficient communication [58]. In a parallel manner to the SAE J1939 standard governing heavy vehicle CAN communication, described in detail in section 2.2.2, the ARINC-825-4 standard defines communication standards for avionics systems on aircraft using CAN [59]. The ARINC-825-4 is an emerging standard for aviation network communication protocols and is currently used in aircraft for systems such as environmental control, doors, galleys, smoke detection, potable water, and de-icing. Its reliability and efficient data exchange, as well as its compatibility among components, has led to large-scale integration efforts within aerospace systems [58]. This integration is evident in the Airbus A350 aircraft, where design directives request the use of ARINC-825 [60]. Due to the extensive research already conducted on CAN security in the automotive field, these same security considerations have been thoughtfully extended to the ARINC-825-4 standard [61]. This convergence highlights the applicability and relevance of exploring the application of MBSE for security to both ground vehicle and aerospace network communication. The utility of MBSE is also evidenced in the Department of Defense Digital Engineering strategy, which highlights the need for cyber-physical system design that employs digital engineering tools including model-based systems engineering [7]. This model-driven approach to secure architecture has also been proven to promote collaboration between system designers and security experts [16].

With the rapid technological advancements of cyber-physical systems and improvements in vehicle autonomy and connectivity, network security continues to rise as a critical field of research. Vehicle transport protocols operate as an application layer protocol on top of the Controller Area Network and are responsible for ensuring that messages are transmitted and processed reliably and efficiently between senders and receivers. However, poorly written requirements and inadequate security controls can result in vulnerabilities in these protocols, as seen in section 2.2.3. These vulnerabilities can be exploited by malicious actors to compromise the safety and security of the

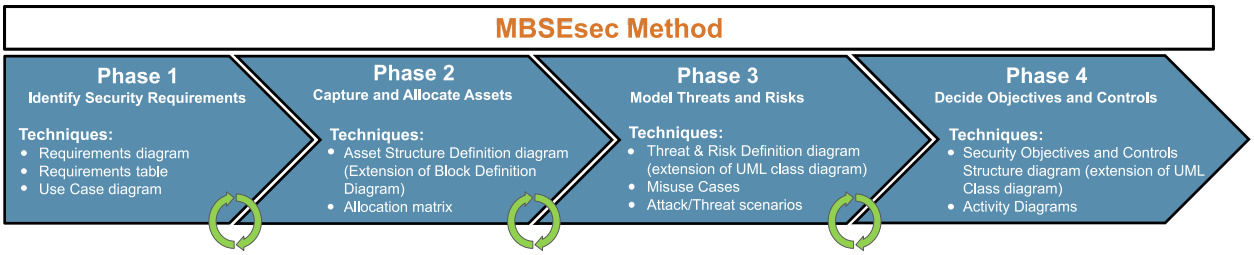

Figure 3.2: MBSEsec method workflow adapted from [2]

vehicle. To mitigate this risk, MBSE can be used to develop better requirements and security controls for vehicle network transport protocols. By leveraging MBSE, engineers can follow a structured approach to the design and implementation of these protocols, reducing the likelihood of vulnerabilities being introduced.

This chapter applies the MBSEsec method detailed in section 1.3.3 to secure the transport layer protocol from vulnerabilities discussed in section 2.2.3. Figure 3.2 illustrates the MBSEsec method adapted from [2]. The iterative method follows four main phases that include: identifying security requirements, capturing and allocating assets, modeling threats and risks, and deciding objectives and controls. The outputs of this method are security requirements and security controls that address previously overlooked vulnerabilities.

3.3 MBSEsec Applied to J1939

The MBSEsec method was selected to be applied to the system of interest because of its clear and distinguished phases of design, and the perceived effectiveness of iteration throughout the method. While some of the other methods focused heavily on requirements or partitioning, MBSEsec presented the most holistic, balanced, and systems engineering-focused approach to developing a secure system. The other methods that were concidered were introduced before 2016, and are specialized in one field of system security, while MBSEsec is the most recent, and was developed with a best practices approach.

Before beginning to develop a security-oriented model of the system of interest (SOI), the MBSEsec security profile was implemented using a UML profile diagram. Profile diagrams are a type

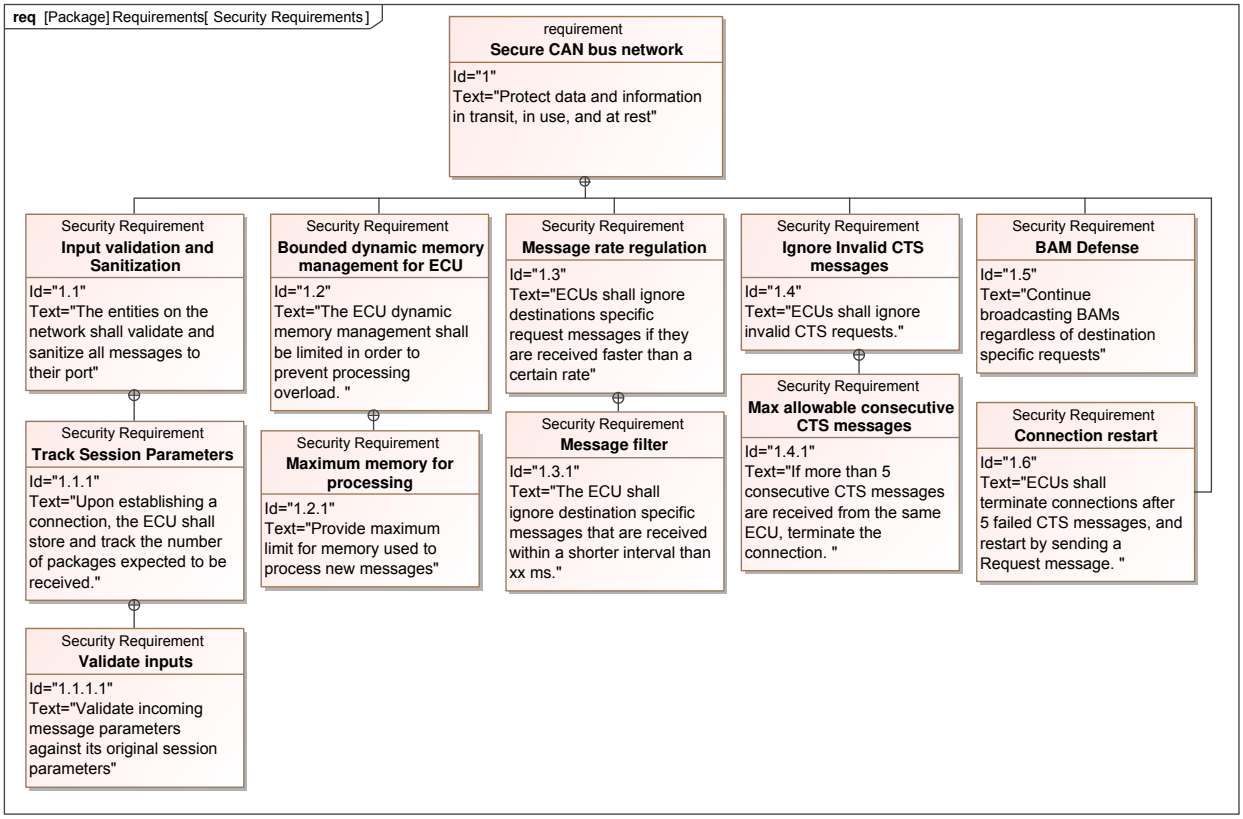

Figure 3.3: ECU Security Requirements

of diagram used for customizing and extending the SysML language to meet specific modeling requirements for a particular system or organization. Profile diagrams are an important aspect of model-based systems engineering (MBSE) and are used to define stereotypes, tagged values, and constraints that can be applied to SysML elements. This profile serves as a means to translate security concepts into modeling language stereotypes that can be implemented throughout the model [32]. The security profile encompasses several key elements, including *security requirement*, *asset*, *risk*, *risk treatment*, *vulnerability*, *threat*, and *security control*. These stereotypes created within the security profile can then be readily applied to various elements within the system, enabling the comprehensive integration of security concepts into the model.

3.3.1 Identify Security Requirements

With a fundamental understanding of the SOI, and the current vulnerabilities established, security requirements that reduce the attack risk of these threats were drafted. Since the primary structural elements on the truck network are the ECUs, it is clear that these entities will be the ones targeted by attacks. The discovered vulnerabilities highlight the fact that the ECUs are the system assets that must be secured. Because ECU software is confidential information and not publicized by the manufacturers, it must be treated as a black box. With that being said, the security requirements focused on ECU behavior and message handling. A system-level requirement was defined as "Secure CAN Bus Network" which serves as the *parent* for the security requirements. The subsequent requirements that were developed focused on securing the ECUs from the current vulnerabilities within the transport protocol, and established functions and behaviors that prevent and mitigate these attacks. Figure 3.3 illustrates the security requirements diagram that was generated for the ECUs on the truck network. The top-level security requirements for the system are described below:

1) *Input validation and Sanitization* - This requirement establishes that once a network connection is made with a sending or receiving party, the ECU will store the initial value of packets requested or established to be sent and will validate all incoming messages against the initial parameters.

2) *Bounded dynamic memory management* - While the ECUs are treated as black boxes, it was deduced that the primary reason for ECU malfunction is due to memory overload. When a high volume of messages floods the ECU, or a message is received that contains packets out of bounds, the ECU memory is responsible for the processing. Setting an upper bound on the amount of memory allocated for processing new messages would prevent the ECU from memory overload and malfunction.

3) *Connection Restart* - This requirement ensures that if data transmission between two ECUs fails, the receiving party will terminate the connection after five failed attempts to receive

data and restart the connection with a request message. This ensures the reduction in Denial of Service attacks.

4) *Message rate regulation* - Since two of the validated exploits involve high-frequency messages being sent to a single ECU, a requirement was established to ignore messages from a sending party if they are sent within a given interval. This requirement aims at reducing DoS attacks, as well as preventing processing malfunctions.

5) *Ignore invalid CTS messages* - This requirement was established to ensure functions were implemented to screen for invalid CTS messages. This will defend against Malicious CTS message attacks and Memory leak attacks.

6) *BAM Defense* - This requirement ensures that no destination-specific messages will alter the necessary broadcast messages to other ECUs.

The security requirements were iteratively updated as the attacks were analyzed and the security controls were refined in the following phases.

3.3.2 Capture and Allocate Assets

This phase of the MBSEsec methodology involves representing the structure of the system and allocating assets as stereotypes defined in the MBSEsec profile to the elements within the system. In order to capture the appropriate system structure, standard network traffic from the CAN Bus was monitored by connecting to the onboard diagnostics port of our truck. The truck that was used to collect this information was a 2014 Kenworth T270, a medium-duty truck designed for commercial applications. The primary ECUs on the network were determined to be the Engine Control Module (ECM), Electronic Brake Control Module (ECM), Transmission Control Module (TCM), and Body Controller (CECU). In order to complete the structure of this system, the Bus, Diagnostic Application, and Attacker were added. It is important to note that due to the system's compromised state, the attacker was added to ensure a system was designed that assumes an at-

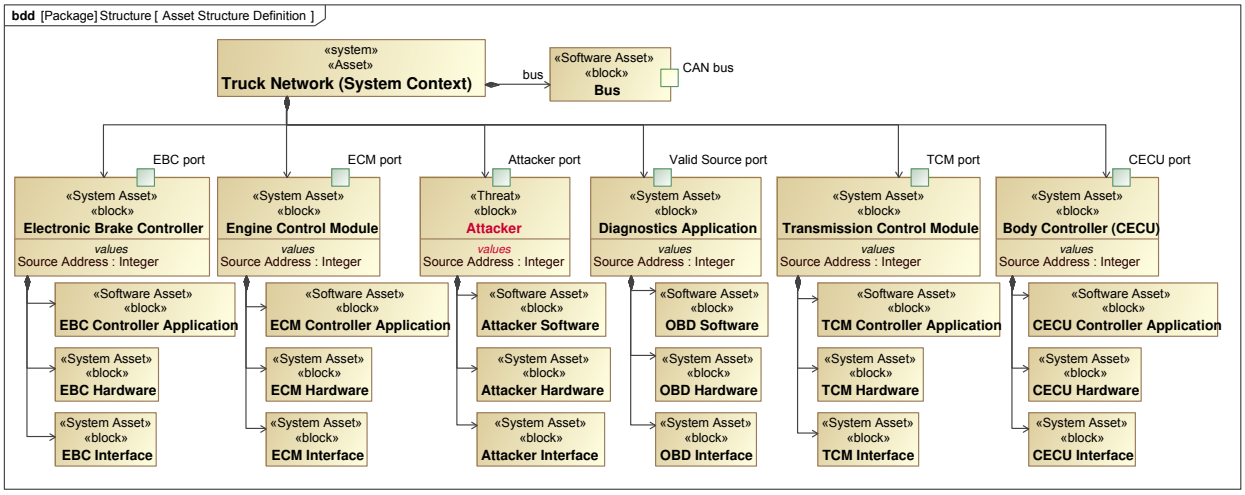

Figure 3.4: Asset Structure Definition

tacker had infiltrated the network. While not explicitly part of the MBSEsec profile, this addition follows the principle of zero trust, which is a strategic objective of secure system design.

Using Blocks to depict the entities, and parts to illustrate the important components of the blocks, an Asset Structure Definition diagram was created. Each structural element comprised key components such as hardware, controller application (software), and interface. Next, the asset classes that were created as stereotypes in the MBSEsec profile were allocated to the system blocks. Fig. 3.4 illustrates the complete structure and asset definition for the truck network. After the stereotypes were applied, the security requirements were allocated to the appropriate system assets. The process of assigning assets and security requirements to the structural elements of the system established a traceable relationship among these elements, facilitating the creation of a comprehensive model (Appendix A, Fig. A.2). This traceability was used later in the process to ensure all requirements were satisfied, and all system assets were properly allocated.

3.3.3 Model Threats and Risks

This phase consists of modeling both behavioral and structural security specifications. For behavioral risk and threat definition, a Misuse Case diagram (Appendix A, Fig. A.1) was developed to describe the goals of a system from the perspective of the users of the system, as well as the goals of the misuser to exploit the system. In this diagram, the goals are described in terms of

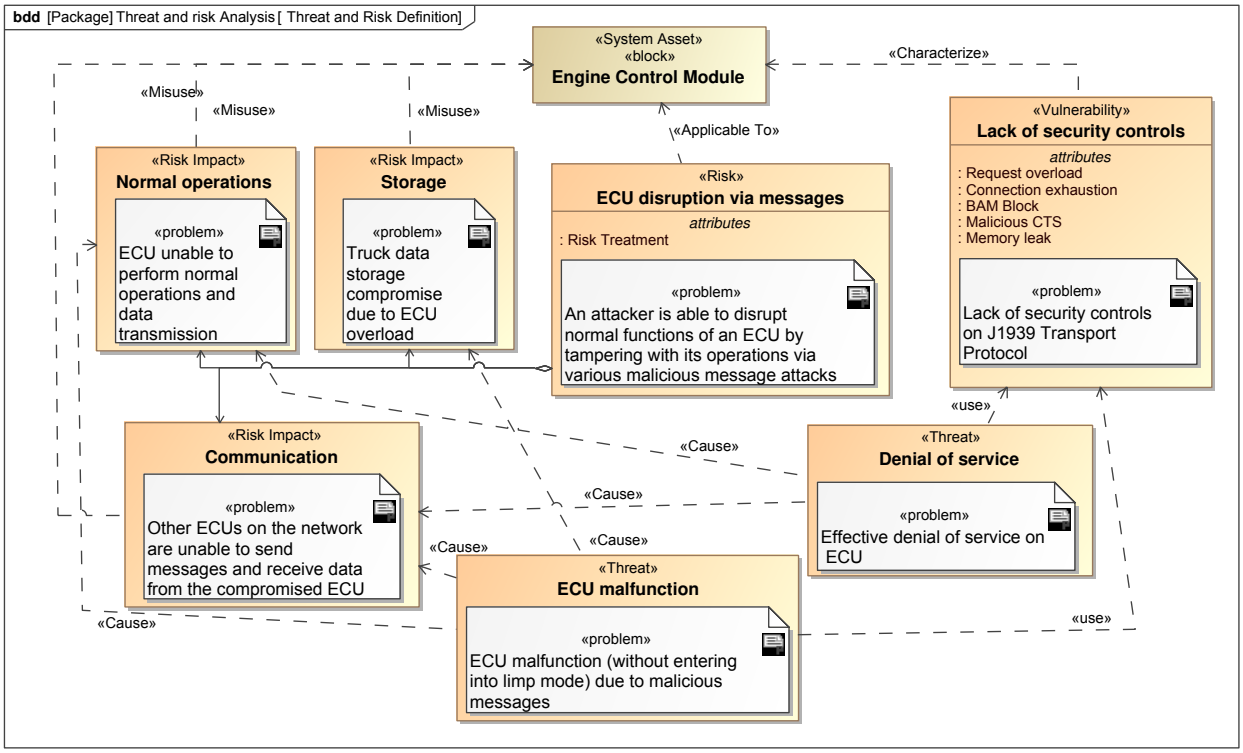

Figure 3.5: Threat and Risk Definition Diagram

functionality that the system must support, and the exploited functionality that the attacks support. The diagram acts as the overall vulnerability specification and describes the interaction between an attacker and the truck network. The misuse case diagram was further elaborated with activity diagrams that describe the attack scenarios stemming from the previously discussed vulnerabilities. The activity diagrams are comprised of swim lanes that represent the behavior of each actor within the scenario. Fig. 3.6 illustrates the attack scenario for the connection exhaustion attack.

Structural risk and threat definition was completed by using the MBSEsec Risk Definition Diagram. The following elements were used in the diagram: *Vulnerability, Threat, Risk, Risk Impact, and Risk Treatment*. These classes were specified as part of the MBSEsec profile and are used to structurally elaborate the systems' risks. Fig. 3.5 displays the Threat and Risk Definition diagram that was created for the system. The risk associated with this system can be defined as "An attacker is able to disrupt normal functions of an ECU by tampering with its operations via various malicious message attacks", which is applicable to all system assets on the network. The

vulnerability of the system can broadly be described as a "Lack of security controls on the J1939 Transport Protocol" which is comprised of the attacks previously characterized. The threats that stem from the vulnerabilities are either "Denial of Service" threats or "ECU malfunction" threats. These two threats cause three primary Risk Impacts: Communication Disruption, a hindrance to normal operations, and loss of available memory. The Threat and Risk diagram is the primary artifact used to generate the appropriate security controls for the system. The Threat and Risk definition process was iterative, with further elaboration of the model completed as the attack scenarios were expounded.

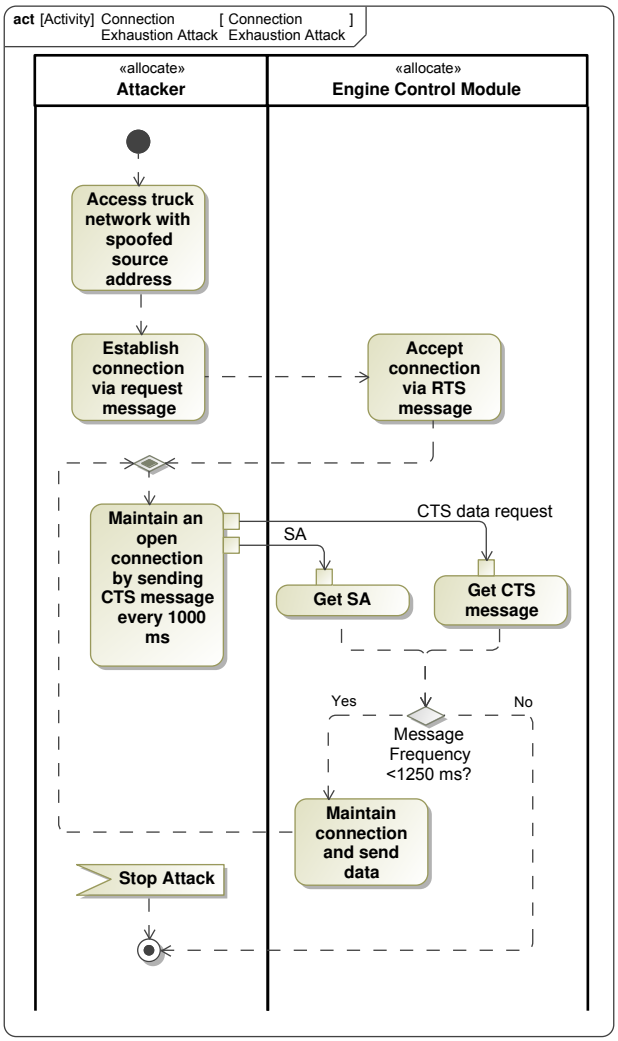

Figure 3.6: Attack scenario depicting the connection exhaustion attack

3.3.4 Decide objectives and controls

The final step of the MBSEsec method involves developing a risk mitigation approach by creating security controls. In the case of the MHD vehicle Network J1939 Protocol, the security objective is to defend against transport protocol attacks. Security controls in the form of activity diagrams were developed to meet the security requirements and objectives by reducing the risk of both DoS attack impacts and ECU malfunctions. Two security control diagrams were developed. It is important to note that a secondary objective of the security controls is to minimize false detection of malicious messages because this would further disrupt network communication. The security controls are listed below:

1) *Validate and Sanitize incoming messages* - This security control outlines an algorithm that dictates whether an ECU maintains a connection with another party. The ECU must first check the capacity it has to process a new message, and will only proceed with processing if there is enough memory. The ECU must then determine the message format. If it is a broadcast message it will simply process and store the data, and if it is a destination-specific message it will continue. The algorithm then specifies for the ECU to check certain characteristics about the incoming message to ensure that it is from a valid party. These characteristics include CTS message frequency and CTS message volume. If data is to be transmitted, the ECU tracks the packets expected to be received and verifies the parameters with the incoming messages. If an inappropriate number of packets is sent or received, the connection is terminated. The *Validate and Sanitize incoming messages* security control is illustrated in Fig. 3.7.

2) *BAM Block Defense* - In order to prevent BAM Block attacks, a security control was implemented through an activity diagram that ensures ECUs will continue to broadcast BAMs regardless of destination-specific data requests. This will prevent destination-specific requests from denying data to the rest of the network.

Activity diagrams were then created that encapsulate the described security control behavior of the ECUs. These security controls were then allocated to the appropriate system assets through an allocation matrix. In order to ensure that the security requirements for the system were met, a requirements satisfaction matrix was developed and crosschecked with the established control behavior (Appendix A, Fig. A.3). Throughout the "Decide Objectives and Controls" phase, the model was iteratively updated to ensure that the security requirements were refined and that behaviors were allocated to the correct system assets.

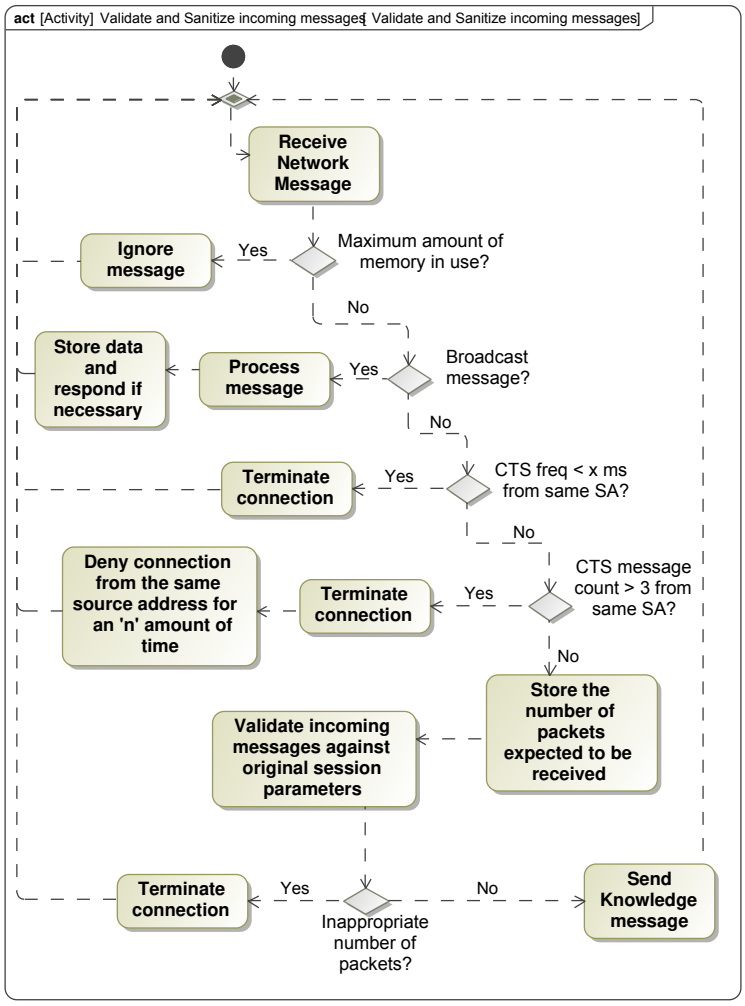

Figure 3.7: Message validation and sanitization security control

Lastly, a security objectives and control diagram was created to structurally define the *control objective*, *risk treatment*, and the *security controls* (Appendix A, Fig. A.4). In this diagram, the relationship between *risk*, and *risk treatment* is defined, and a "generalization" relationship is made between the *security controls* and the *risk treatment*. The security control activity diagrams are also attributed to the security control classes within the diagram to ensure traceability. This diagram ensures that the security controls, in the form of activity diagrams, present a preventive behavior structure and specific actions that would allow for fulfilling the security *control objective*.

3.4 Conclusions from Application of MBSEsec

This chapter demonstrated the use of MBSEsec to elicit security requirements, identify system assets, model threats and risks, and generate security controls for a cyber physical system's network transport protocol with previously validated exploits. Beginning with a detailed understanding of J1939 and its existing vulnerabilities, an initial iteration of security requirements and asset definition was effectively achieved. With the Authors' extensive background knowledge of J1939 and its security threats and vulnerabilities, eliciting security requirements was possible because the risks and threats to the system were well understood. A critique of MBSEsec is beginning with security requirements can be challenging if there is limited system knowledge. This is the case for new systems in design and a possible challenge for practitioners with limited expertise on the system or similar systems. If a full system architecture is present, more security requirements are likely to be available or generated, but systems in earlier phases of system design will have more unknowns in implementation and more challenges to compiling a set of security requirements to begin the MBSEsec method. Our assessment of MBSEsec is its utility is highest for uncovering new security controls for systems already fielded or with a detailed architecture.

Throughout the phase of modeling threats and risks, iterative updates were made to the security requirements, ensuring the mitigation of the elaborated attack scenarios on the system. Modeling the threats and risks forced the team to better understand the vulnerabilities, and therefore enhance and update the security requirements as the model progressed. The MBSEsec method's

structure, coupled with the dynamic nature of MBSE, facilitated continuous updates to the model. The presence of embedded traceability within the model greatly simplified the identification of gaps in security controls and requirements. As the design process progressed, requirements underwent multiple updates, and security control objectives were modified to address the evolving understanding of system threats and risks. Consequently, a comprehensive model was achieved, encompassing refined security controls and requirements.

The application of the MBSEsec profile (stereotypes) to the model elements helped focus the modeling process on the security aspect of the system. Characterizing different elements within the model with their security-related stereotypes helped facilitate the modeling techniques, diagram development, and traceability. This was especially helpful when developing the Threat and Risk Definition, Asset Structure, and Security objective and control diagrams.

To enhance the applicability of the MBSEsec method to fielded cyber-physical systems, we created an additional structural element, *attacker*, to be modeled in the system architecture. In our case, the J1939 network protocol was already known to have active vulnerabilities that were previously exploited, but we recommend for all systems there is value in including an *attacker* element as a part of this security analysis. This is based on the zero trust architecture principles specified in [62] where we assume a network will be compromised at some point. Utilizing the zero trust principle enables further elaboration and enhancement of the security controls, which can be seen in the implementation of the *Message validation and sanitization* security control that tests each incoming message against a set of parameters to ensure that the message is not malicious.

Chapter 4

Recommended improvements to the MBSEsec Method

4.1 Application of MBSEsec MRZR

The first application of the MBSEsec methodology was to a cyber-physical system that was understood thoroughly by our team and was primarily used to elicit requirements and controls for the J1939 protocol given all the industry's best practices for securing this system. Applying MBSEsec to the J1939 protocol provided a detailed understanding of both the system of interest and the modeling methodology, and enabled effective brainstorming and documentation of the outputs; requirements, and controls. However, to further examine the utility of MBSEsec, and the utility of MBSE to secure system design, the approach was applied to a second vehicular system comprised of similar network configurations and properties. In this application of the methodology, the objective was to investigate how MBSEsec can help find additional security considerations with a potentially lower security Subject-Matter Expert (SME) level, and prove its value in being a repeatable and iterative approach, enabling better requirements and security control development.

The system to which the MBSEsec is applied is a remote control system applied on top of an off-road vehicle, the MRZR platform developed by Polaris. The MRZR is a versatile and lightweight all-terrain vehicle designed for military use. It offers high levels of mobility and can be configured for various mission requirements, making it well-suited for rapid deployment and maneuverability in challenging terrains.

The design team for this remote control system had already conducted a cybersecurity table-top, in which they performed a risk analysis, and detailed the primary vulnerabilities that can be expected for the system. With the appropriate documentation from the system's design team, a model was developed using the MBSEsec methodology to accurately reflect the system design.

Figure 4.1: Polaris MRZR [3]

The objective of the model development was to investigate how well the model could document the system, as well as its effectiveness as a tool to make better design decisions at earlier phases of system design when SME input is less frequent.

In the development of the model, a few shortcomings of the approach were discovered in phase 3 of the methodology: Model Threats and Risks. In the following sections, these shortcomings are discussed, and a recommended improvement to the MBSEsec methodology is detailed. It is important to note that the goal of this section is to improve the MBSEsec method by adding rigor while limiting the complexity of the model development process, with the goal of improving its posture as an effective tool for engineers.

4.1.1 MBSEsec shortcomings

During the model development, the first two steps of the MBSEsec process were accomplished in which initial security requirements were established, the system architecture was defined, and assets were allocated to system elements. These two phases were completed in accordance with the documentation provided by our partners, as well as effective systems engineering practice. In order to effectively accomplish the third phase of modeling threats and risks, a working group with subject-matter experts from the lab was created in order to implement the third phase into the

model using the provided cybersecurity tabletop. In doing so, we found that the threat and risk definition phase lacked a sufficient level of qualitative detail when addressing risk.

Currently, this phase of the method involves structural and behavioral definition of the threats and risks that pertain to the system. The behavioral definition involves the use of activity diagrams and misuse cases, and the structural definition involves developing a threat and risk definition diagram. When detailing the MBSEsec method, the author alludes to the implementation of risk criteria into the model through the implementation of a block stereotype *Risk Assessment Configuration* with an attribute *Criteria for Accepting Risks* as seen in Figure 4.2. However, other than implementing a *Risk* stereotype with a risk level enumerated from one to ten, no quantifiable relationship is established to ensure that risks are mitigated and that they do not exceed an established threshold. The Author of MBSEsec does not discuss or expound on how to incorporate the *Criteria for Accepting Risk* defined in the risk assessment configuration stereotype, or how to classify levels of risk that pertain to each vulnerability. Lastly, there is no defined relationship between the security controls and their 'reduction' of the risk caused by vulnerabilities.

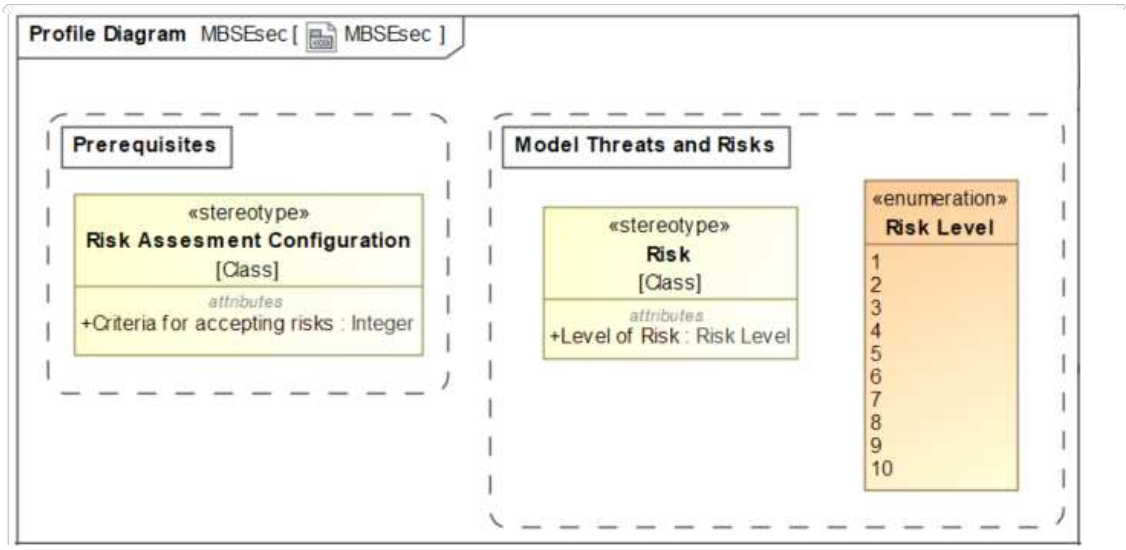

Figure 4.2: MBSEsec stereotypes pertaining to risk

While the profile does address the value of risk by providing a way to numerically define it, the extent to which it would help progress the user of the method to produce better security controls

is very limited. It is important to note that MBSE and methods such as MBSEsec are not designed to replace the conventional methods of risk analysis, rather they are used to document and capture the most relevant information for system design, such as the outputs of a risk assessment, and implement them into the model for improved results and design decisions. This chapter will focus on improving MBSEsec's utility as a tool to perform simple and effective risk classification.

4.1.2 Recommended Changes

In order to address these changes, new techniques and diagrams were added to the third and fourth phases of MBSEsec (Figure 4.3). The objective of these additions is to add a more robust definition of threats and risk within the model, enabling better documentation, which leads to more secure design decisions. The added techniques and diagrams were developed with the user in mind and were applied using SysML's capabilities.

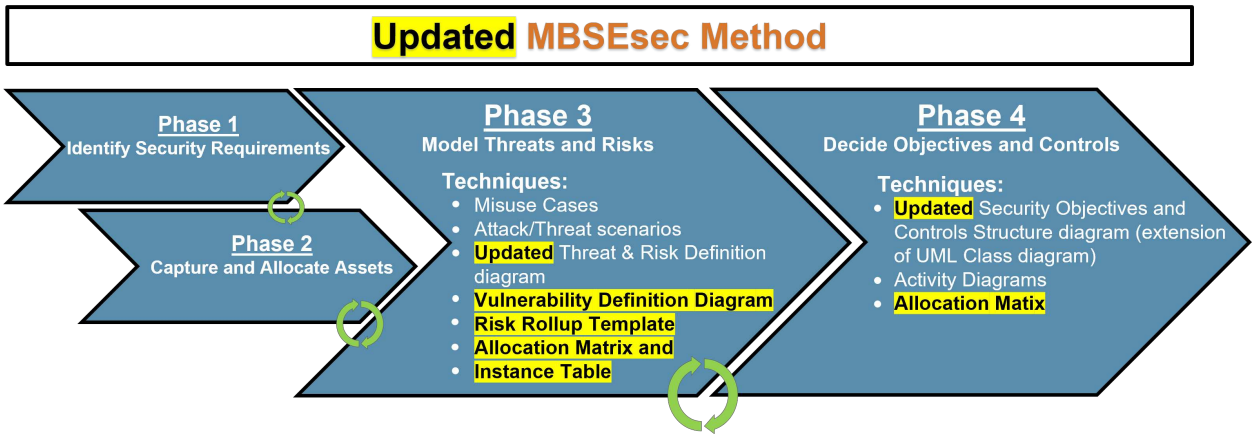

Figure 4.3: Recommended updates to MBSEsec

In its simplest form, the outputs of a risk assessment are the discovered or suspected vulnerabilities that threaten the system, and their associated likelihood and impact, as well as recommended mitigations and controls. Likelihood can be defined as the probability or chance that a specific threat will occur, quantifying the potential for its realization. Impact can be defined as the severity and consequences of a threat, measuring the extent of harm, loss, or adverse effects it could cause [63]. These quantities are often defined in a 'heat map' where the risks of vulnerabilities are

graphically depicted using colors to indicate the severity or impact of various risks. Both Likelihood and impact are quantified by receiving a rating between 1 and 5, with five being the highest severity or frequency, and one being the lowest [63] [39].

While there are many ways to define and classify risk, STRIDE (1.1) stands out as an effective framework for defining risks, provides structured categorization, establishes a common language, and aids in risk prioritization. STRIDE allows for the systematic identification of security threats, ensuring that no critical risks are overlooked, and helps organizations communicate and collaborate effectively among various stakeholders.

Using these two basic principles of both qualitative risk definition in the form of STRIDE, and quantitative risk definition in the form of threat likelihood and impact, a more robust threat and risk definition phase was implemented. As can be seen in figure 4.3, The Threat and Risk Definition diagram was updated, a vulnerability Definition Diagram was created, a Risk Rollup pattern was introduced, and Allocation Matrices and Instance tables were added. In order to demonstrate the changes to the method, the MRZR system and its associated documentation is used as the system of interest.

4.1.3 Vulnerability Definition Diagram

Currently, the MBSEsec method defines risk using a threat and risk definition diagram, as seen in Figure 3.5 in chapter 3. While this diagram provides value, it does not provide a means of documenting the vulnerabilities as well as their associated quantifiable risk characteristics. Therefore, a vulnerability definition diagram was implemented to meet this need. A block definition diagram that follows the structure of an attack tree was implemented into the model. Attack trees are a visual tool used in risk analysis to model and analyze potential threats or attacks on a system [64]. They are effective because they provide a structured way to break down and visualize various attack scenarios, starting from a central objective or goal of an attacker and branching into sub-goals and specific attack techniques. By visually representing attack paths and dependencies, engineers can improve their overall security posture, making them a valuable asset in the field of

cybersecurity and risk management. This diagram is used as a structural definition of all the relevant vulnerabilities that threaten the system and follows the same pattern as an attack tree. The 'parent' block in this diagram is titled "System Security Risk".

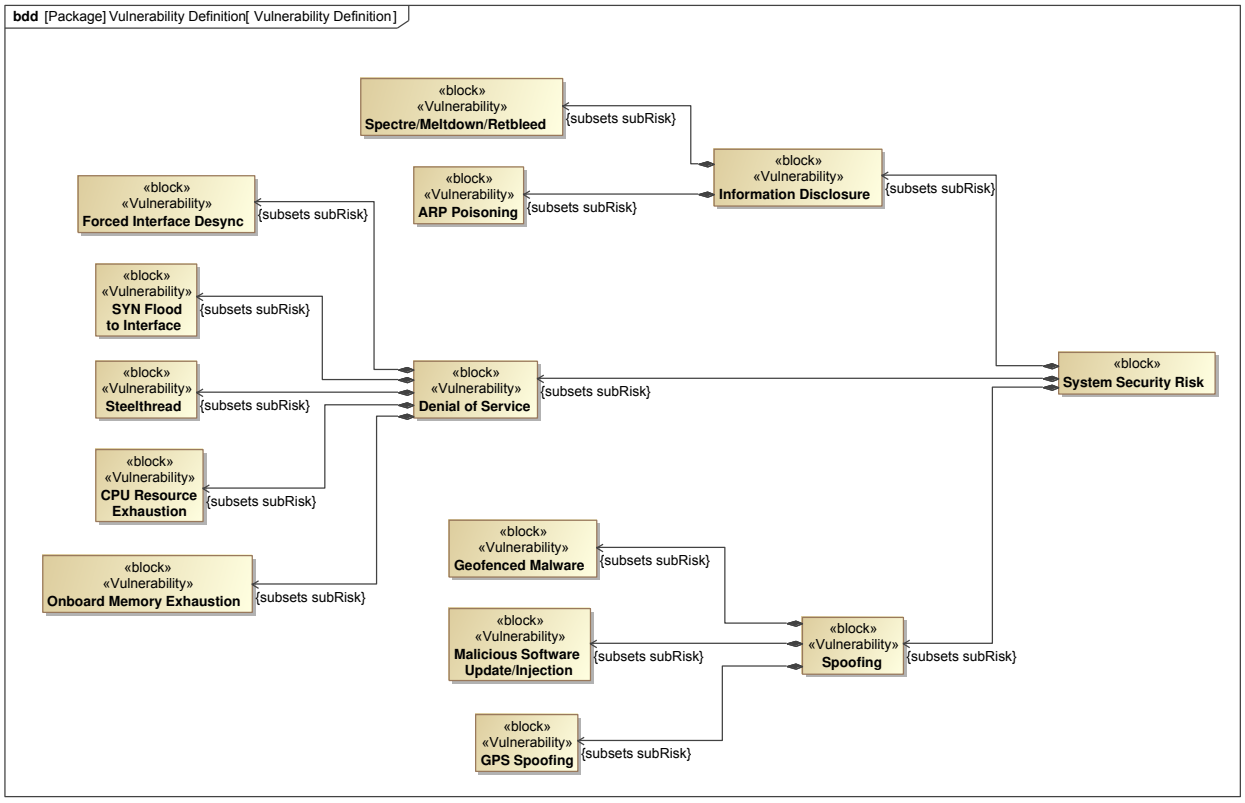

Figure 4.4: Vulnerability Definition Diagram

In order to capture the different attack paths that the attacker may attempt to achieve, the system risk block is composed of five elements, each representing a different component of the STRIDE methodology for classifying threats. In this diagram, the STRIDE framework is used to categorize and analyze potential threats to the system, each block in the diagram will represent the different threats: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege [44]. In the case of the MRZR system in Figure 4.4, the vulnerabilities that were discovered all fell into three of the five classifications; Spoofing, Information Disclosure, and Denial of Service. As can be seen in the diagram, each vulnerability is classified with the

MBSEsec stereotype *Vulnerability*, and each element of STRIDE is composed of its associated vulnerabilities.

4.1.4 Characterizing Risk

Once the attack tree is built, and the primary vulnerabilities are established, a roll-up pattern is implemented into the diagram. A roll-up pattern is a modeling technique that represents the hierarchical aggregation of components within a system. It involves creating aggregation relationships, assigning property values, and rolling up information from lower-level components to higher-level blocks using parametrics. Using this approach enables a simple yet effective qualitative evaluation of the systems' risk. In the vulnerability definition diagram, each block is elaborated with its respective *Impact* and *Likelihood* using value properties.

As mentioned above, the output of a risk analysis is an outline of all the vulnerabilities that threaten the system, along with their associated impact and likelihood. Adding these value types enables each vulnerability to be structurally defined with its associated risk characteristics. It is important to note that MBSEsec is not intended to be a standalone risk assessment, but rather a tool to capture the appropriate elements of the analysis. As seen in Figure 4.5, each vulnerability discovered in the cybersecurity tabletop for this system is assigned a value property for its likelihood and impact. A scale of 1-5 was used for classifying each vulnerability.

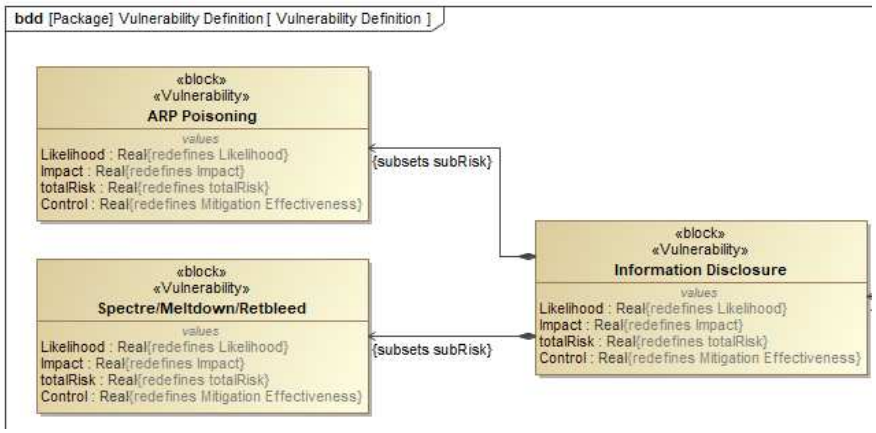

Figure 4.5: Value properties assigned to each block

Using a roll-up parametric pattern in SysML, the total risk of the system can be calculated through the sum of all the vulnerabilities. This is done by multiplying the likelihood by the impact and summing up all the risks into one value assigned to the *System Security Risk* block. A simple equation was used within each block to derive the appropriate risk value, which can be seen in Equation 4.1. J. Woodruff defines risk as the chance that someone or something that is valued will be adversely affected in a stipulated way by the hazard [65], and defines risk as the multiplication of the severity of the harm (Impact) by the Likelihood of the occurrence. In the same manner, Equation 4.1 was used in the model to calculate the total risk for the system, where the individual risks were summed as the parametric "rolled up" the block hierarchy.

Overall risk for each vulnerability:

$$\text{Total} = (\text{Likelihood} \times \text{Impact}) + \sum \text{Total Risk} \tag{4.1}$$

4.1.5 Risk Roll up Diagram

With an overall risk defined for the system, a Risk roll-up diagram is then created to define a framework to analyze the system's risk posture. As can be seen in Figure 4.7, a *Security Risk Rollup* block with the appropriate MBSEsec 'Risk Assessment configuration' stereotype was implemented. Through the use of a generalization relationship, the *Security Risk Rollup* block is attributed two constraints; *Risk Criteria* and *Normalize Risk*. These two constraints are the means by which the system risk posture is evaluated. As mentioned previously, the MBSEsec method provides a 'criteria for accepting risk' but does not provide a means of testing whether this criterion is met. It also provides a 'scale' from 1-10 for ranking a certain risk, but does not provide any context to the scale or a means to use it for evaluation purposes. These two constraint blocks help solve these issues in the current methodology. The *Normalize Risk* constraint block is designed to normalize the system risk value in order to ensure that systems with a different number of vulnerabilities can be compared and analyzed on the same scale. This is accomplished with equation 4.1.

Normalized system Risk Equation:

$$\text{Relative Risk} = \frac{\text{Total Risk}}{\text{Number of Vulnerabilities} \times 25} \tag{4.2}$$

The relative risk defined in the constraint block outputs a value between 0 and 1, which specifies the severity of the risk posture, with a 0 inferring no risk from the vulnerabilities, and a 1 inferring every vulnerability is likely to occur and have high impacts.

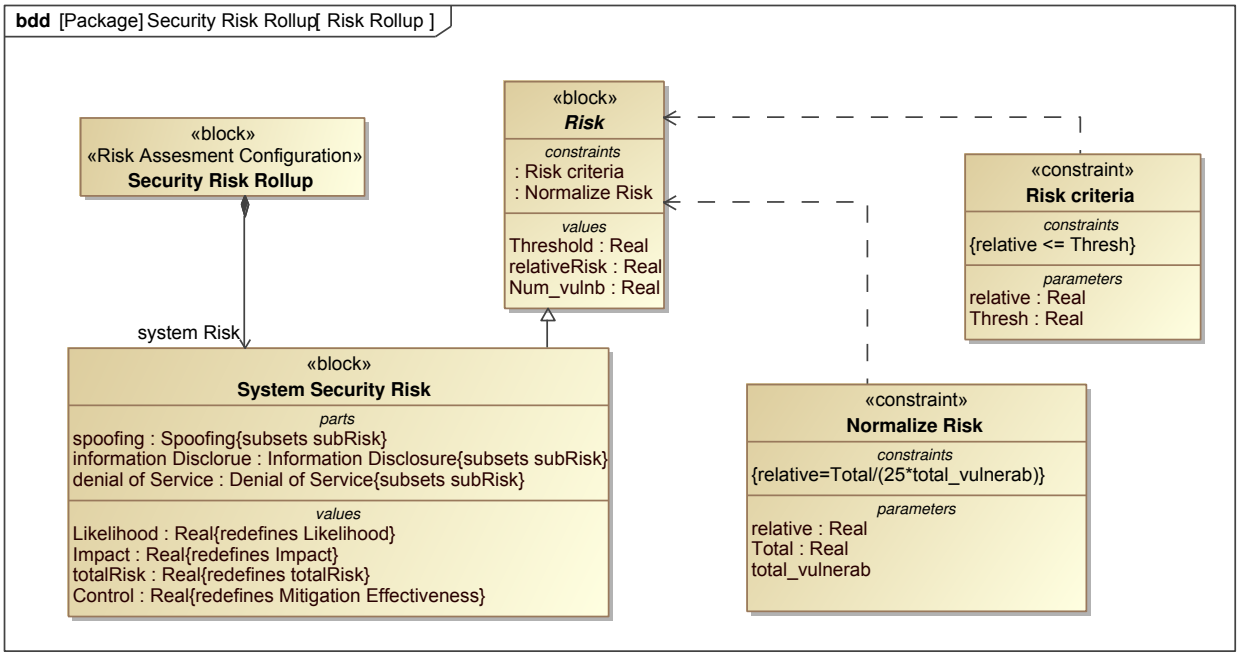

Figure 4.6: Risk Roll up diagram

The *Risk Criteria* constraint simply compares the normalized system risk to the risk threshold defined by the user (Equation 4.3). This enables the SysML tool to notify the user whether the system risk is below the threshold.

Risk Criteria Equation:

$$\text{Relative Risk} \leq \text{Threshold} \tag{4.3}$$

Lastly, the *Security Risk Rollup* block is elaborated with a parametric diagram which is implemented in order to execute the defined constraints (Figure 4.7). Using the user inputs of a risk threshold and the number of discovered vulnerabilities, as well as the previously established 'total risk', the parametric diagram evaluates the system's risk posture.

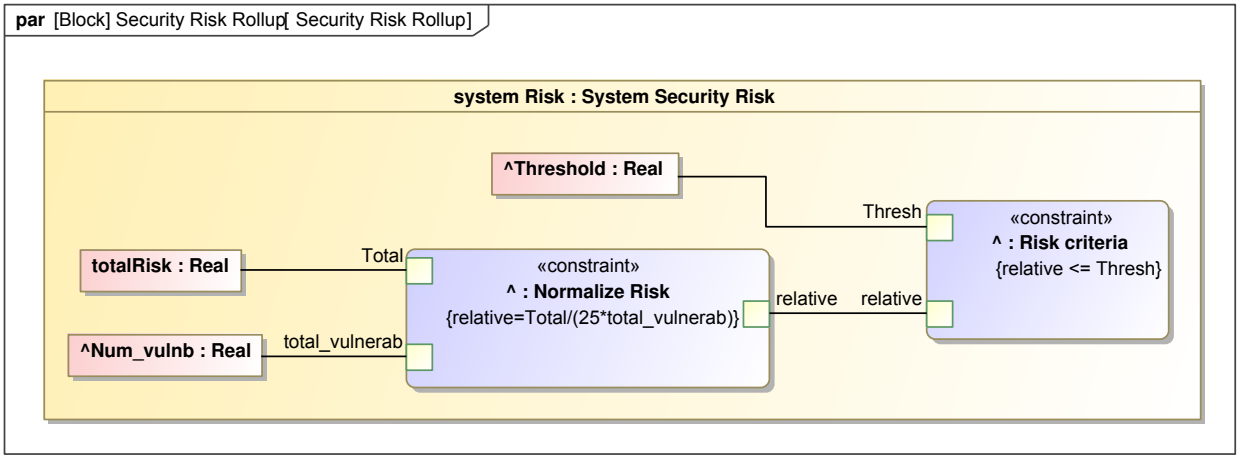

Figure 4.7: Risk Roll up diagram

4.1.6 Simulating the risk Analysis

The additions to the MBSEsec model are designed to be run as a simulation, in which the values of likelihood and impact are defined for each vulnerability, and the simulation will output the values of overall risk, normalized risk, and whether the system meets the defined risk threshold. If the threshold is met, the values are displayed in green, and if it is not, the values are displayed in red.

4.1.7 Updated Threat and Risk Definition Diagram

Lastly, the threat and risk definition diagram proposed by MBSEsec was revised to reflect the proposed changes. First, The elaborated system security risk block was added to the diagram to further characterize the overall system risk. As can be seen in Figure 4.8, the Security Risk block indicates that the system's total risk is at 168 based on the risk roll-up and that the system sta-

tus is "pass" in regard to meeting the threshold requirements. Implementing this block enhances traceability, and provides better documentation. Second, the vulnerabilities classified in the vulnerability definition diagram (attack tree) were added to the diagram. To simplify the diagram, the threat blocks were removed, due to a lack of contribution to the model. All the 'trace' relationships and block stereotypes were preserved from the previous MBSEsec iteration. As the model is simulated, the displayed values would change, therefore enabling a much better qualitative risk definition.

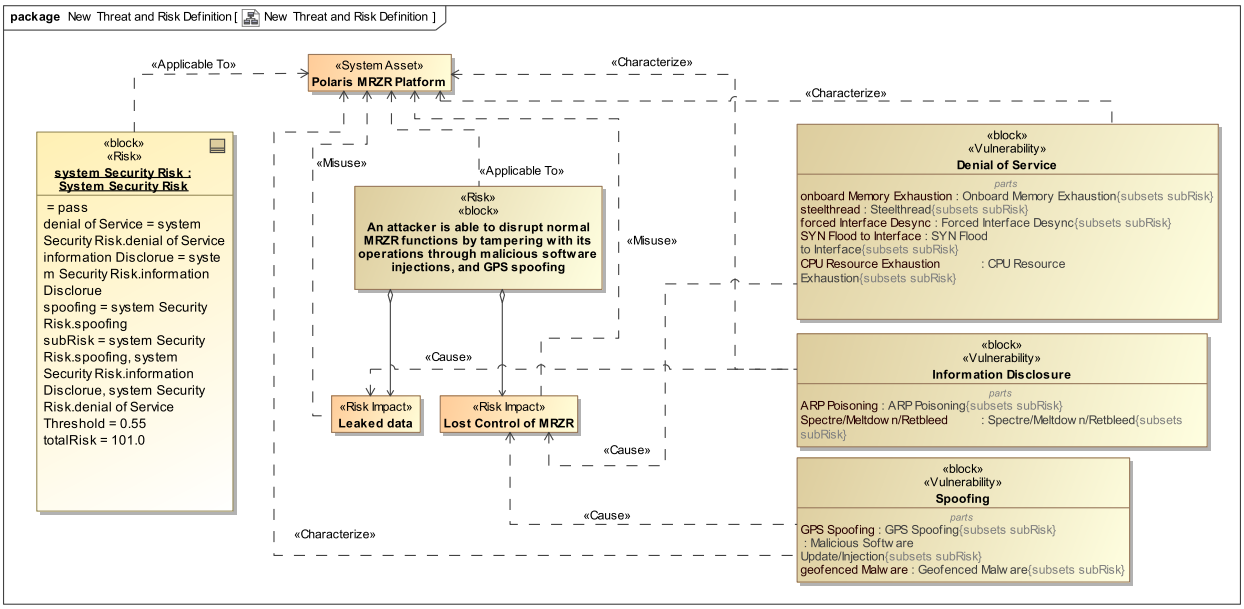

Figure 4.8: Revised Threat and Risk Definition

4.1.8 Integrating Security controls

Phase 4 of the MBSEsec method defines the process for defining security controls. However, as mentioned previously, there is no qualitative relationship between the effectiveness of the selected security controls and the overall system risk. Therefore, to better integrate the risk definition and security control phases, a Security Control Diagram was developed (Figure B.1, Appendix B), in which each security control is specified, as well as its mitigation effectiveness as a percentile (between 0-1). In order to implement the control effectiveness into the model simulation, a 'control'

value property was added to the vulnerabilities in the attack tree to be used as part of the Risk Roll-up process (See Figure 4.5). Using the Mitigation effectiveness value assigned to each security control, a new overall risk is calculated for each vulnerability, implementing the added mitigation effectiveness value. As can be seen in Equation 4.4, the mitigation effectiveness was added to the risk calculation.

New overall risk for each vulnerability, with added mitigation effectiveness:

$$\text{Total} = (\text{Likelihood} \times \text{Impact}) \times (1 - \text{Mitigation Effectivness}) + \sum \text{Total Risk} \tag{4.4}$$

Figure 4.9 demonstrates the parametric diagram that was implemented into the model in order to perform the risk roll-up for the system. As the controls are articulated, a mitigation effectiveness value between 0 and 1 is assigned to each vulnerability to characterize how effective the mitigation will be at reducing the vulnerability impact and likelihood.

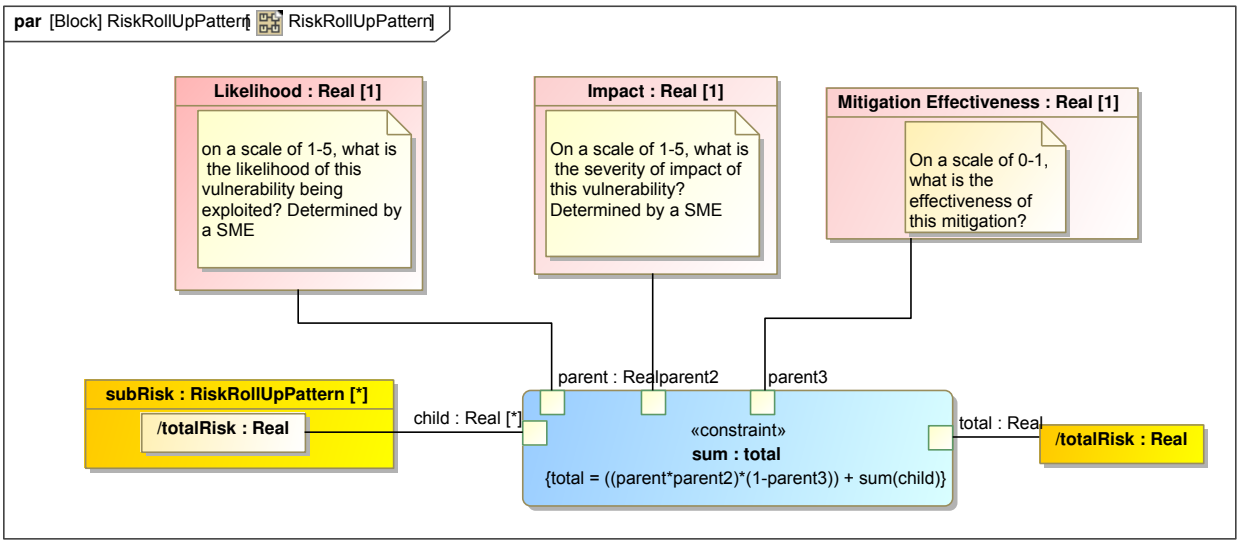

Figure 4.9: Overall risk roll-up parametric

By incorporating the security controls into the model simulation and implementing their associated mitigation effectiveness, a comprehensive model simulation can be executed to systematically assess the security posture of the system. This process allows for a dynamic view of how the sys-

tem's risk level evolves with different security measures, enabling users to evaluate the effectiveness of these controls in risk reduction and make informed decisions about their implementation. Leveraging the simulation capabilities inherent in SysML, stakeholders can iteratively refine the system's risk criteria, adjust control effectiveness, and consequently engineer a heightened level of security for the system.

4.1.9 Overview of Revised MBSEsec

The following section outlines the new steps to follow when using the MBSEsec Method. Underlined descriptions highlight the recommended changes. Table 4.1 details the added diagrams to the method, the SysML elements used, and the rationale behind the additions.

Phase 1 - Identify Security Requirements

a) Identify security requirements as an additional part of functional and non-functional requirements. Capture requirements in a SysML Requirement diagram or table using the "Security Requirement" Stereotype. Utilize an overall system Use Case Diagram to better refine Phase 1.

Phase 2 - Capture and Allocate Assets

a) Define the system structure using a block definition diagram (bdd).

b) Define the system asset types that the design team must secure. Capture them in an Asset Structure Definition diagram (extension of a bdd).

c) Using an allocation matrix, link system structural blocks and assets with the SysML 'allocation' relationship.

Phase 3 - Model Threats and Risks

a) *Behavioral security specification*: Using the Use Case diagram developed in phase one, use the extended Use Case diagram for identifying Misuse Cases.

Table 4.1: Recommended changes to MBSEsec

Added MBSEsec Diagram	**Diagram type(s)**	**SysML Elements Used**	**Utility**
Vulnerability definition diagram	Block definition diagram	- Block - Vulnerability stereotype - Risk rollup pattern	a) Follows attack tree pattern for specifying threats b) Provides simple documentation of all vulnerabilities c) Enables the application of the roll-up parametric
Risk Roll-up Diagram	Block Definition Diagram and Parametric Diagram	- Block - Constraint Block - Risk Assessment config Stereotype - Risk Rollup Parametrics - Value Properties	a) Provides a more effective 'Risk Assessment Configuration', by qualitatively evaluating the system risk b) Provides the capability of specifying the likelihood and impact of each vulnerability c) Enables the use of simulation to iteratively refine the system's risk criteria and mitigation effectiveness
Updated Threat and Risk Definition	UML Package Diagram	- Block - Class -Vulnerability Stereotype - Risk Stereotype - Associations	a) Refines and tailors the previous Threat and Risk Definition diagram to document the appropriate changes to Phase 3 b) Provides improved traceability
Security Controls Diagram	UML Package Diagram	- Class - Security Control stereotype - Effectiveness Enumeration	a) Captures the selected security controls b) Provides the ability to enumerate each control with its mitigation effectiveness c) Enables traceability between vulnerabilities and controls
Controls to Vulnerabilities Allocation	Allocation Matrix	- Block - Class - Allocate Relationship	a) Ensures all vulnerabilities have been appropriately 'controlled' b) Improves model traceability

b) *Behavioral security specification*: Elaborate the Misuse Cases with activity diagrams to model the attack scenarios.

c) *Structural security specification*: Using the knowledge gained from the behavioral security definition, develop a vulnerability definition diagram (attack tree), and apply the 'RiskRollupPattern' to the *System Security Risk* element.

d) *Structural security specification*: Implement the Security Risk Rollup diagram into the model. Simulate the Security Risk Rollup block, specify the number of vulnerabilities, the risk threshold, and each vulnerability's likelihood and impact.

e) *Structural security specification*: Develop the New Threat and Risk Definition diagram as a summary of Phase 3. The following elements should be created and linked: System Asset; Risk; Risk Impact; System security Risk; and Vulnerability.

Phase 4 - Decide Objectives and Controls

a) Capture all the selected security controls in the Security Controls Diagram (UML Class Diagram).

b) Simulate the Security Risk Rollup block with the appropriate mitigation effectiveness values applied to each vulnerability. Ensure that the overall system risk is below the threshold.

c) Summarize results in a Security Objectives and Controls Structure diagram (an extension of the UML Class diagram) for defining elements of security objectives and controls.

d) Develop Activity Diagrams that identify workflow or algorithm for the selected security control.

4.2 Discussion of results

This chapter summarizes the conclusions and results obtained through the investigation of the MBSEsec methodology, particularly its utility in securing intricate cyber-physical systems. In this

chapter, the MRZR remote control system was used as the system of interest for the application of the MBSEsec method. This work uncovered shortcomings within the methodology, primarily in the phase of modeling threats and risks. While the methodology introduced stereotypes that addressed risk such as the Risk Assessment Configuration, the method lacked a quantifiable component to risk, and did not relate risks to their mitigations.

Consequently, a series of enhancements were proposed designed to bolster MBSEsec's effectiveness. These changes encompassed a more robust risk classification approach, emphasizing effective documentation, and simulation capability. New techniques and diagrams, such as the Vulnerability Definition Diagram and Risk Roll-up Diagram, were introduced. These refinements enabled a more detailed understanding of threats and risks to the system. The integration of a qualitative and quantitative approach to risk assessment, using STRIDE for threat categorization and likelihood multiplied by impact for risk assessment, improved the method's capability for risk evaluation. The Risk Roll-up Diagram provided a comprehensive view of the system's risk posture, provided parametric to normalized risk values, and compared them to user-defined thresholds for a clearer risk assessment.

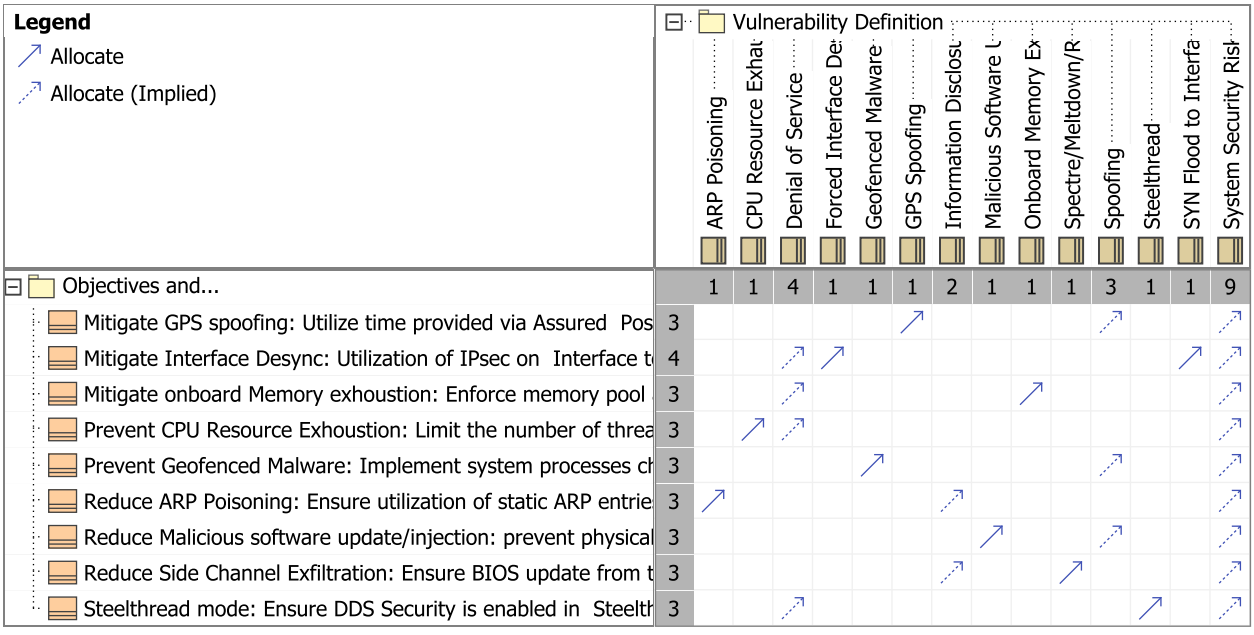

Figure 4.10: Allocation of security controls to vulnerabilities

NIST SP 800-53r5 [29] states that "The combination of a catalog of security and privacy controls and a *risk-based control selection process* can help organizations comply with stated security and privacy requirements, obtain adequate security for their information systems, and protect the privacy of individuals." The provided additions and changes to the MBSEsec method support the risk-based control selection methodology detailed in this standard, therefore helping the modeling process better align with the standards. These changes offer an actionable framework for engineers to iteratively refine their system's risk criteria, evaluate the effectiveness of security controls, and enhance the overall security posture of their systems. The two primary benefits of these recommended changes are the enhanced traceability between vulnerabilities and security controls and the ability to simulate the risk evaluation and make iterative changes to the model mid-simulation. Figure 4.10 illustrates the result of allocating the system's vulnerabilities to their associated security controls discovered in the stakeholder cybersecurity tabletop. As can be seen in the figure, each vulnerability has at least one security control designed to mitigate its effect. With the vulnerabilities following the structure of an 'attack tree' it can be seen that all the security controls are allocated by means of implication to the overall system security risk. These security controls were all attributed to an associated 'mitigation effectiveness' which enables the effective use of simulation.

#	Name	Likelihood : Real	Impact : Real	totalRisk : Real	Mitigation Effectiveness : Real
1	Instances				
2	system Security Risk	0	0	101	0
3	system Security Risk.spoofing	3	4	20	0
4	system Security Risk.spoofing.gps spoofing	2	4	2	6
5	system Security Risk.spoofing.malicious Software Update/Injection	1	4	2	2
6	system Security Risk.spoofing.geofenced Malware	4	3	4	8
7	system Security Risk.information Disclorue	3	3	30	0
8	system Security Risk.information Disclorue.arp poisoning	4	5	11	9
9	system Security Risk.information Disclorue.spectre/Meltdown/Retbleed	3	5	10	5
10	system Security Risk.denial of Service	5	4	51	0
11	system Security Risk.denial of Service.onboard Memory Exhaustion	3	5	6	9
12	system Security Risk.denial of Service.steelthread	3	4	4	8
13	system Security Risk.denial of Service.forced Interface Desync	3	5	5	10
14	system Security Risk.denial of Service.syn flood to wmi	3	2	6	0
15	system Security Risk.denial of Service.cpu resource exhaustion	4	5	10	10

Figure 4.11: Instance table of risk values assigned to the vulnerabilities of the Polaris MRZR

One of the key benefits of simulation is the streamlining of decision-making for the user through parameter adjustments. This is best accomplished through the use of an instance table, which is a tabular representation of system elements and their states in a simulation. Using an instance table provides a concise overview of the system vulnerabilities, which helps facilitate the efficient management of simulation scenarios. As can be seen in Figure 4.11, the vulnerabilities discovered for the Polaris MRZR were assigned the appropriate likelihood and impact values, along with the calculated mitigation effectiveness. Each vulnerability was then given a risk value, for example, the ARP poisoning vulnerability was given likelihood and impact values of 4 and 5 respectively. The associated mitigation effectiveness was 9, bringing the total risk to 11 (20-9=11). As the simulation is run, these values can be updated, and the table reflects the new outputs.

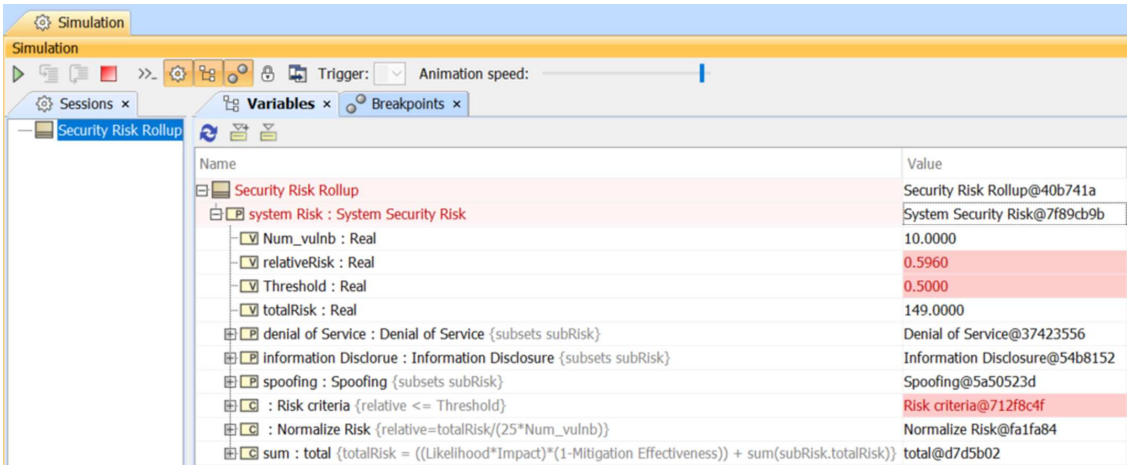

Figure 4.12: Simulation of updated MBSEsec without controls implemented

Figures 4.12 and 4.13 illustrate the utility of using the simulation capability of the model to generate better security controls. Figure 4.12 demonstrates the model simulation with the ten implemented vulnerabilities to the system, with the relative risk being 0.5996 which is above the threshold of 0.5. Once the mitigation effectiveness values were added to each vulnerability based on the selected security controls, the overall system risk decreased to an allowable level of 0.4164. These figures demonstrate the utility of using MBSE as a tool for relating system risks captured in a risk analysis to the resort of the system design (requirements, structure, and security controls).

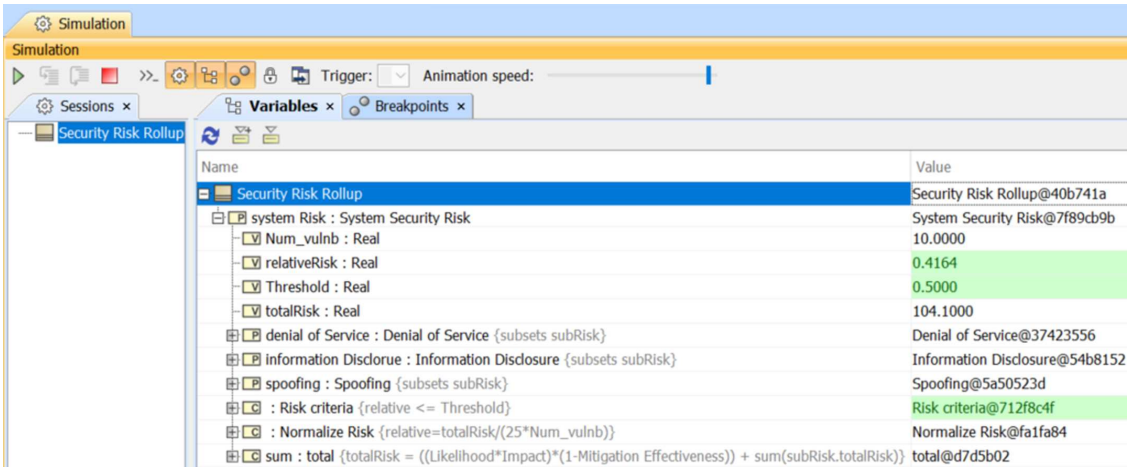

Figure 4.13: Simulation of updated MBSEsec with controls implemented

This chapter presents MBSE and MBSEsec as valuable tools in the domain of cyber-physical system security design. As the digital landscape evolves, this methodology continues to adapt, improve, and fortify our understanding of system security. The recommended changes to the method provide a dynamic view of how the system's risk level evolves with different security measures, enabling a simple evaluation of the effectiveness of each control and making informed decisions about its implementation. Secondly, the recommended changes help in predicting and optimizing the system's responses to potential security threats and vulnerabilities, enabling users to proactively identify and address security weaknesses. Lastly, they add a layer of documentation, traceability, and robustness to the model, enabling systems engineers to utilize an MBSE model as an effective 'source of truth' for the design process, as cybersecurity continues to progress toward 'security by design'.

Chapter 5

The application of using the enhanced MBSEsec method to eliciting security goals

5.1 Introduction

The INCOSE Vision 2035 states that one of the key objectives is that "Cybersecurity will be as foundational a perspective in systems design as system performance and safety are today" [18]. To achieve this vision statement and address security design from a system's inception, system goals for security must be elicited and translated into functionality-focused security requirements for CPS design. Based on the previous chapter's work on exploring MBSE's use for security, and the lessons learned from the analysis of MBSEsec, in this chapter, I discuss the contributions made by MBSE to a new security goals elicitation method.

The systems engineering community continues to focus on and improve system requirements elicitation as requirements provide the design to specifications that are verifiable and testable to ensure the system meets the user's needs. However, as can be seen in the literature review conducted in section 1.3.4, most security requirement elicitation methods use threat and risk assessments that require a preliminary or detailed system architecture to elicit security requirements. While beneficial, this approach does not address security in conceptual system design before a system architecture is identified. Thus, the design trade space to implement a secure system architecture is limited and relies heavily on the knowledge of available subject-matter experts.

The third research question proposed in this work is "How can MBSE be leveraged to support security requirement elicitation in the design of a secure system?" To answer this question, this chapter focuses on the contribution of MBSE to a new approach intended to generate security goals in the conceptual design phase of system design for Cyber-Physical Systems (EGRESS). The goal of this method is to expand the options for system design to create an inherently secure

system. This is done by setting security goals that guide initial design choices and serve as the foundation for defining additional security requirements throughout the design process. Using MBSE as the foundation, a security goals elicitation method was developed that streamlines and combines STPA, MBSEsec, and other best practices in modeling and requirement engineering. A balance was achieved by using preexisting SysML diagrams and adding custom profiles in order to develop the most effective method. As opposed to the previous two chapters, this work falls at the highest level of abstraction within the System-V model, between the concept of operation and system requirements (Figure 5.1).

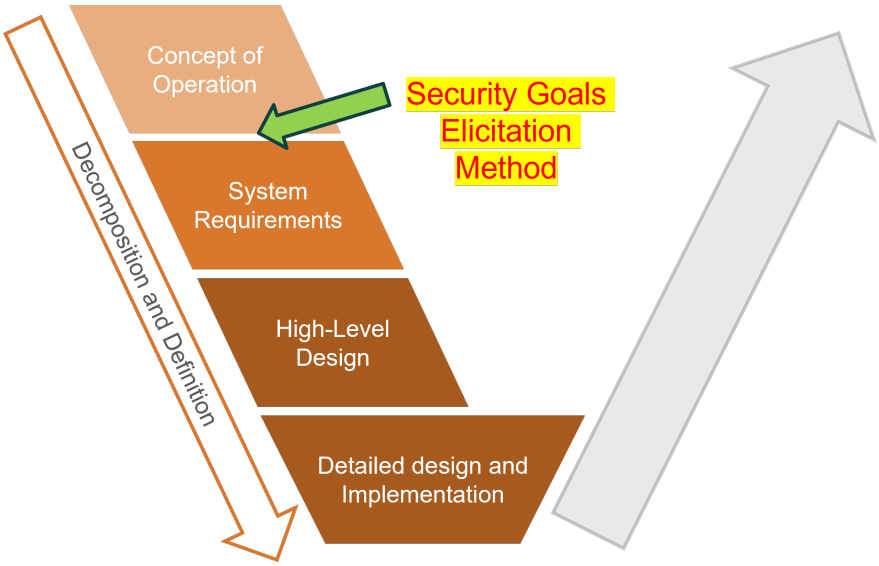

Figure 5.1: Security goals elicitation in system lifecycle

5.2 The EGRESS Method

The EGRESS method is designed for application in the conceptual design phase of complex cyber-physical system design before a preliminary architecture is established for the System of Interest (SoI). As highlighted previously, there is a gap in this field in regard to eliciting early security requirements when architectural features are unavailable. Figure 5.2 outlines the overall flow of the EGRESS method, with the recommended modeling techniques to be followed. The method

begins with stakeholder input and employs an iterative approach combining structural and behavioral analysis to derive security-focused goals for guiding subsequent design activities. The output of this method is a set of security-focused system goals, laying the groundwork for the elicitation of SMART (Specific, Measurable, Achievable, Relevant, Time-bound) security requirements.

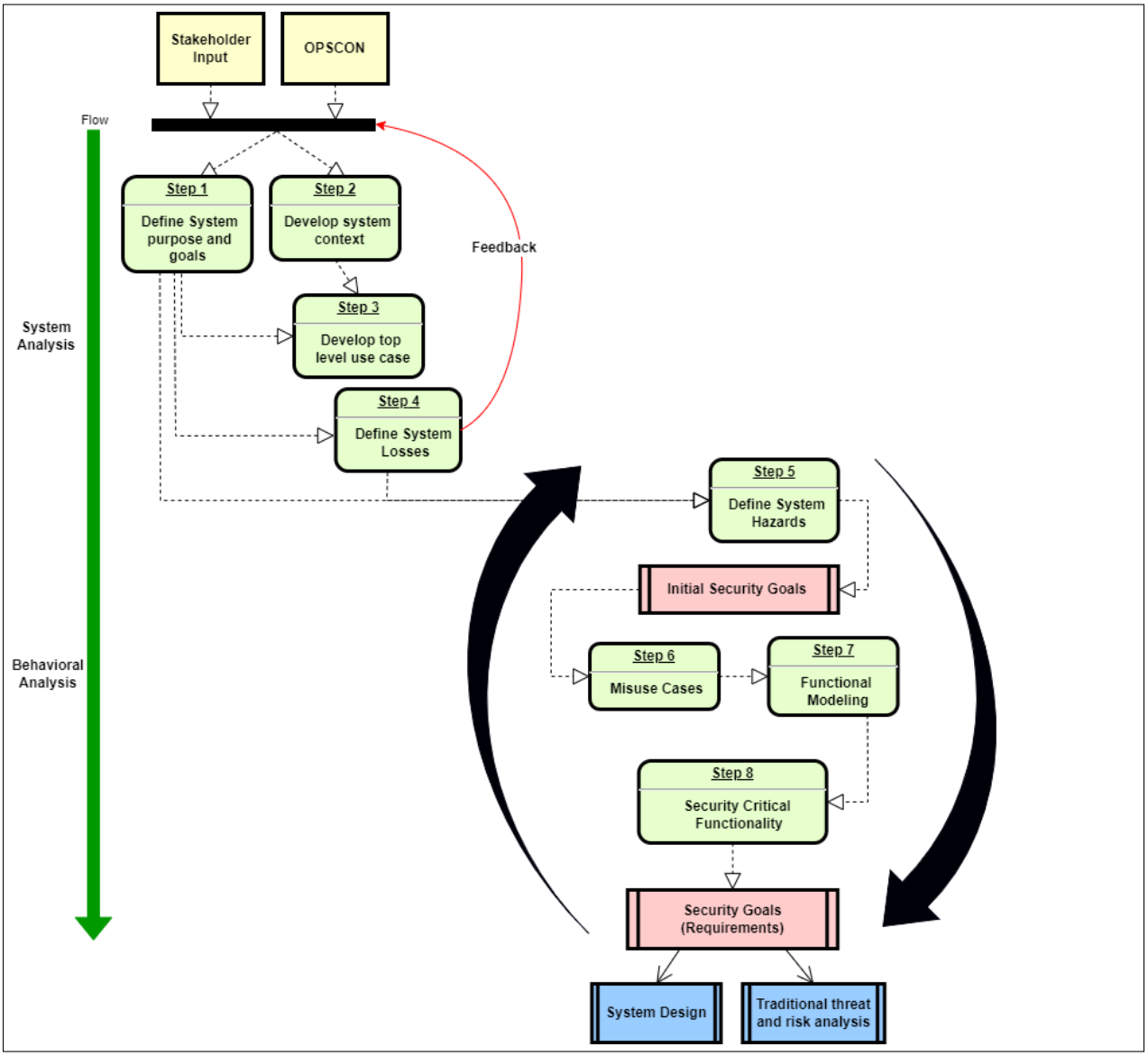

Figure 5.2: EGRESS Method

The following sections outline the EGRESS methodology step by step, and detail how MBSE was used to elaborate the methodology.

5.2.1 Step 1: Define System Purpose and Goals

The first step of the EGRESS method is to develop clear system purpose and system goal statements. These statements are designed to guide the rest of the modeling process in a conceptual manner and should be taken directly from stakeholders and any system documentation available, ideally a CONOPS or OPSCON. The purpose and goals statements are at the highest level of abstraction for the system and ensure stakeholder agreement on the mission and vision of the system's intended use.

It is prescribed to document the system purpose and goals through a custom SysML Block Definition Diagram that characterizes the elements and defines their relationship to one another. Custom block stereotypes *Goal*, *Purpose*, and *Method* are created to allow the user to clearly label each block. Custom relationships of *In order to*, and *By means of* are created to define the relationship between the elements. These relational elements are implemented to help demonstrate how the elements relate to one another. For example, the *Purpose* of the system must be achieved *In order to* accomplish the system *Goal*. And the system *Purpose* is completed *by means of* the *Method*. This diagram can be used as a tool for stakeholder alignment, to ensure that all parties agree on the definitions.

5.2.2 Step 2: Develop System Context

The purpose of this step is to capture the context in which the system will operate. In this step, system engineers and stakeholders must consider the domain in which the system operates, including external actors of interest that may either directly or indirectly interact with the system. This step can be considered 'complete' when all relevant elements external to the system have been identified. The objective of this step is to provide a clear outline of the system domain, to provide all parties involved an understanding of what is 'in bounds' for the system design and what is a relevant environmental concern, but not a part of the system design. It is important to note that this phase does not focus on any internal parts or behavior of the system, rather it is a structural representation of all external elements of the system around the SOI.

This step is to be captured through a block definition diagram that defines all the objects and elements through system blocks. The block definition diagram visually represents system context, defines system boundaries, and identifies external actors, fostering clear communication. The diagram facilitates traceability, ensuring consistent integration of security considerations throughout the system engineering process. Additionally, it helps bridge the gap between technical and non-technical stakeholders, creating a shared understanding of the system's context.

5.2.3 Step 3: Elaborate Use Cases for SOI

Step 3 of EGRESS involves developing a top-level use case diagram for the system under consideration. The primary purpose of this diagram is to depict the system's objectives from the perspective of its users [53]. These objectives are defined in terms of the functions the system needs to support and how external entities or actors interact with the system. This provides a broader context for understanding the system's functional aspects and outlines high-level behavioral goals that the system must fulfill.

Developing this diagram serves several important purposes. Firstly, it identifies key actors involved in the system and ensures that the primary use cases encompass all critical behavioral elements of the system. Additionally, this diagram acts as a valuable tool for brainstorming and facilitates reaching stakeholder agreement. Furthermore, it aids in establishing and refining the system's context and boundaries as the development process progresses. This step is captured through a use-case diagram which not only create key use cases for the system but also offers a dynamic visual representation that aids in stakeholder communication, agreement, and ongoing refinement of the system's functional objectives.

5.2.4 Step 4: Define System Unacceptable Losses and Hazardous states

This step involves the systematic definition of system unacceptable losses and the identification of hazardous states that may lead to these losses. Unacceptable losses are specific, undesirable outcomes as defined by the key stakeholders. System losses should identify what is of highest value to the stakeholders and differentiate from what is nice to have/desired. Unacceptable losses

can be mission, personnel, or equipment losses with common unacceptable losses including loss of life or mission essential equipment [66]. Hazardous system states are a set of system conditions, that when paired with a worst-case set of environmental conditions, can lead to an unacceptable loss. Identifying hazards can also serve to refine and clarify the list of unacceptable losses, as each hazard should be mapped to one or more unacceptable losses (otherwise it is not a hazard or the list of unacceptable losses is incomplete).

This step is captured by a simple block definition diagram used to document the hazard and loss elements, as well as trace the relationship between the two. Two new stereotypes, *Loss* and *Hazard* are introduced to this diagram, which enables the clear distinction between the types of elements in the diagram. These stereotypes also enable simple and easy traceability later in the model. As shown in Figure 5.2, this step provides a good opportunity to iterate on the first few steps of EGRESS and update the System OPSCON as needed. The stakeholder alignment and system understanding gained through the activities of defining the system purpose, elaborating its context, top-level behavioral functionality, and unacceptable losses are non-negligible and provide a solid foundation for the system design efforts.

5.2.5 Step 5: Draft Initial Security Goals

Initial security goals are now drafted for the system. These system-level security goals are based on ensuring that the system does not enter the hazardous states identified in Step 4. These goals must act as constraints to prevent the system from entering these states. The security goals are defined using a standard SysML requirements table and are allocated to corresponding system hazards with a custom *prevents* relationship, which ensures that each system Hazard has been 'prevented' by security goals.

It is important to emphasize that while integrating custom elements into SysML for EGRESS is highly beneficial, heightened customization concurrently introduces increased complexity for users. Therefore, instead of opting for a potentially complex custom table, we chose to use a requirements table, a familiar tool for SysML users, to define security goals. This decision prior-

itizes user understanding and ease of use amidst the potential challenges associated with elevated customization in SysML.

5.2.6 Step 6: Elaborate Misuse Case Diagram

This step of the EGRESS method involves further elaboration of the previously established use case diagram. A Misuse case diagram is an extension of the use case diagram that focuses on detailing the high-level attack scenarios and the involved malicious actors. This step is accomplished by adding an *attacker* actor to the use case diagram and brainstorming possible vulnerabilities and attack vectors a malicious actor may attempt to exploit. The misuse case diagram is used as a brainstorming tool to consider high-level potential attack vectors independent of a candidate system architecture. To simplify and structure this step, we propose reviewing each use case for its susceptibility to Confidentiality, Integrity, and Availability (CIA) exploits. The CIA Triad is well-defined and commonly used to structure many security analyses. The CIA definitions are defined and detailed throughout the ISO 27000 series of standards [67], [68]:

- Confidentiality: Ensure only authorized parties can access sensitive data and processes.
- Integrity: Ensure information is not altered, corrupted, or destroyed.
- Availability: Ensure authorized users can obtain information and use processes.

CIA provides a structured classification of potential threats and organization by the threat type to the system. Custom SysML stereotypes *Availability Attack*, *Confidentiality Attack*, and *Integrity Attack*, are added and used to label misuse cases with which element of CIA is under attack. A new dependency, *Exploits*, is added that documents which misuse cases exploit which primary use cases of the system. Additionally, a *attacker* stereotype was used to the actor blocks to enhance traceability.

The goals of this step are threefold:

1. This step brainstorms how we enter hazardous states from a security perspective, and can be used as a tool for identifying any hazardous states not previously considered prior to this step.

2. Developing a more detailed understanding of the potential attack vectors and potential vulnerabilities to the system. As well as classifying these possible threats with CIA.

3. A validation of the initial security goals is performed by reviewing if the goals in Figure 5.8 address preventing misuse cases identified in this step.

5.2.7 Functional Modeling and Security Critical Functions

In this step of the modeling process, the top-level use cases identified in 5.2.3 are elaborated through activity diagrams that describe the key behaviors of the system. By doing this, important activities can be identified as security-critical functions. It is important to note that the behavior diagrams are not intended to introduce any design constraints, but rather outline the expected high-level behavior from the system. As a new contribution, we propose identifying important activities in each activity diagram as security-critical functions as a custom stereotype *security critical function*. The behaviors identified as a *security critical function* are then used to ensure that security goals have been produced to protect these actions. This enables goal verification and traceability. In addition to classifying security-critical functionality, custom stereotypes are used to identify if the critical activity has information confidentiality integrity or availability concerns. This helps guide the identification process and incorporates security principles into the identification process.

The objective of this step, through the use of Misuse cases, is to uncover any additional security goals that may have been overlooked during structural analysis. One of the advantages of MBSE modeling is its capability to represent both the functional and behavioral facets of a system. This comprehensive approach aids in identifying diverse security aspects of the system that might not be explicitly enumerated through other methods.

5.2.8 Goal Verification

Once the behavioral analysis is complete, the initial security goals can be evaluated for their completeness and accuracy. This is done by developing allocation matrices, and crosschecking all security-critical functions to ensure that each one is allocated to at least one specific security goal. If gaps are identified, then new security goals can be added and related to hazards and functions for traceability.

This step in EGRESS is accomplished by creating an allocation matrix comparing security-critical functionality to the current security goals. If each security critical function is not addressed by a security goal, then we need to add a new security goal to address the security of the system functionality. The allocation matrix enables traceability within the system elements and is automatically updated as changes are made, therefore making it a tool for ensuring design completeness.

5.2.9 Output - Initial System Goals for Security

The resulting output of EGRESS is a table consisting of a list of security goals developed and verified through both structural and behavioral modeling. These goals are system-level goals elicited from a security lens. They are NOT system requirements, instead, they form a foundation in which specific measurable system requirements can be elicited as a part of the system design process using existing available best practice methods (such as MBSEsec) available in the community of practice.

5.2.10 EGRESS Stereotypes

In order to support the incorporation of the EGRESS method into a systems modeling language, custom stereotypes were defined for various steps within the method. Table 5.1 summarizes the stereotypes that were implemented into five of the diagrams recommended for the system model.

The implementation of these stereotypes facilitates the integration of the EGRESS method into the standard SysML modeling language. They are crucial for capturing the security-specific nuances and concepts of EGRESS, which ensures that the resulting model is highly representative

of the methods' objective; eliciting early system security goals. Furthermore, the use of custom stereotypes enables a more streamlined and coherent modeling process, enhancing the effectiveness of brainstorming and stakeholder engagement, and ultimately contributing to the method's effectiveness in addressing the unique characteristics of the targeted system of interest.

Table 5.1: Stereotypes & Elements used in proposed EGRESS Diagrams

EGRESS Diagram	**Custom Stereotype(s)**	**SysML Element**
System Purpose and Goal Diagram	- Goal - Purpose - Method - *By means of* - *In order to*	- Class
Loses and Hazards	- Hazard - Loss	- Block - *Allocate*
Requirement table (SysML Standard)	- Hazard - *Prevent*	- Requirement
Misuse Case Diagram	- Confidentiality Attack - Integrity Attack - Availability Attack - Attacker - *Exploits*	- Use Case - Association - Include - Extend - Generalization
Activity Diagrams (SysML Standard)	- Security Critical Function	- Initial Node - Activity Final - Action - Control Flow - Decision and Merge - Fork and Join - Swimlanes

5.2.11 Tracability

A key benefit of the application of MBSE to the EGRESS approach is the embedded traceability within the model. Significant traceable relationships include:

- Security Goals to stakeholder defined system objectives (unacceptable losses)

- Hazardous states to unacceptable losses
- Security critical functions to Security Goals

Traceability empowers decision makers with clear information on how a security goal is required to ensure required system functionality in adverse operating conditions. This presents strong evidence to combat the mindset of security as an inconvenience or an unnecessary cost. This becomes even more effective as future specific security requirements are elicited. They can then be mapped back to the system security goal which is mapped to hazards and losses thus empowering the design engineers with clear rationale for answering the question of *why* the security feature or control is required.

5.3 EGRESS Method Applied to Off-Road Navigation System

To demonstrate the utility of this method, we decided to apply the EGRESS method to a new system concept. This system can be described as an 'Off-Road Navigation system', and its proposed purpose is to provide reliable and user-friendly navigation guidance through outdoor natural terrains that do not have pre-existing paths. The system aims to provide users with reliable directions, real-time updates, and essential information to ensure safe and efficient navigation through these untamed landscapes. To accomplish this, the system will offer intuitive and personalized navigation instructions leveraging onboard sensors, GPS communication, and mapping algorithms. It will also consider the distinctive features of the terrain, encompassing factors such as topography, vegetation, and natural landmarks.

This system is inspired by collaborative work with government partners utilizing the MRZR 4.1 and for the sake of confidentiality, will not be elaborated further. However, the EGRESS process began with top-level stakeholder documentation from our partners.

5.3.1 Define System Purpose and Goals

In the case of the Off-road Navigation system, a purpose and goal diagram was generated that captures the highest level of abstraction of the system. These statements should bring the

stakeholders and engineers to an agreement and can be documented both in the model and the system OPSCON or CONOPS to guide the security goals elicitation process. Figure 5.3 illustrates the use of the purpose and goal diagram. In practice, this diagram helps facilitate discussion and agreement on these definitions, as well as relate each conceptual element through the relational phrases.

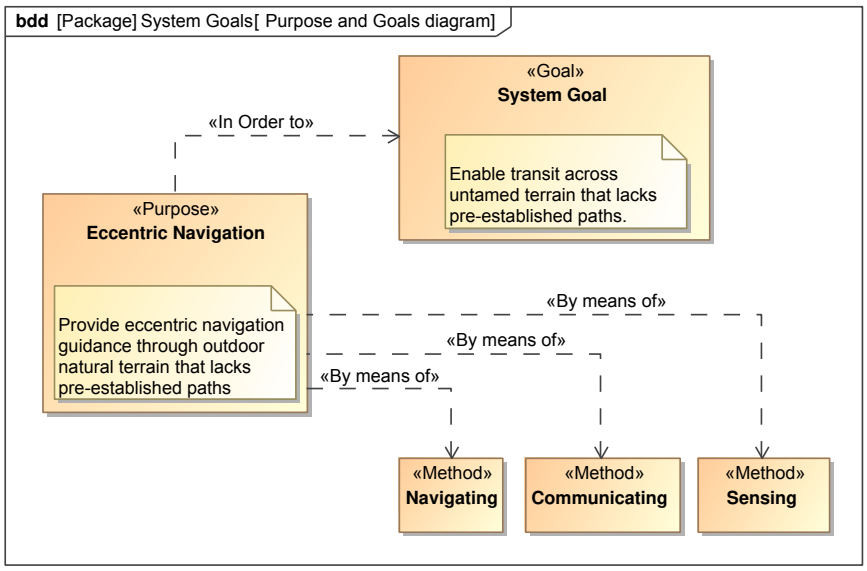

Figure 5.3: Off-Road Navigation system Purpose and goals diagram

As can be seen in figure 5.3, the three stereotypes of *purpose*, *goal*, and *method* are elaborated and the relationship between each one is clear.

5.3.2 Develop System Context

In this second step of the EGRESS method, we identify all the elements external to the system that must be considered. In the case of an Off-road Navigation System, the majority of the elements pertain to the environment in which it operates. This diagram frames the 'big-picture' context of the system of interest through top-level blocks that represent key elements within the environment.

The context diagram (Figure 5.4) illustrates the elements that the system of interest will interact with. As can be seen in the diagram, the navigation system is modeled as a 'component' of the

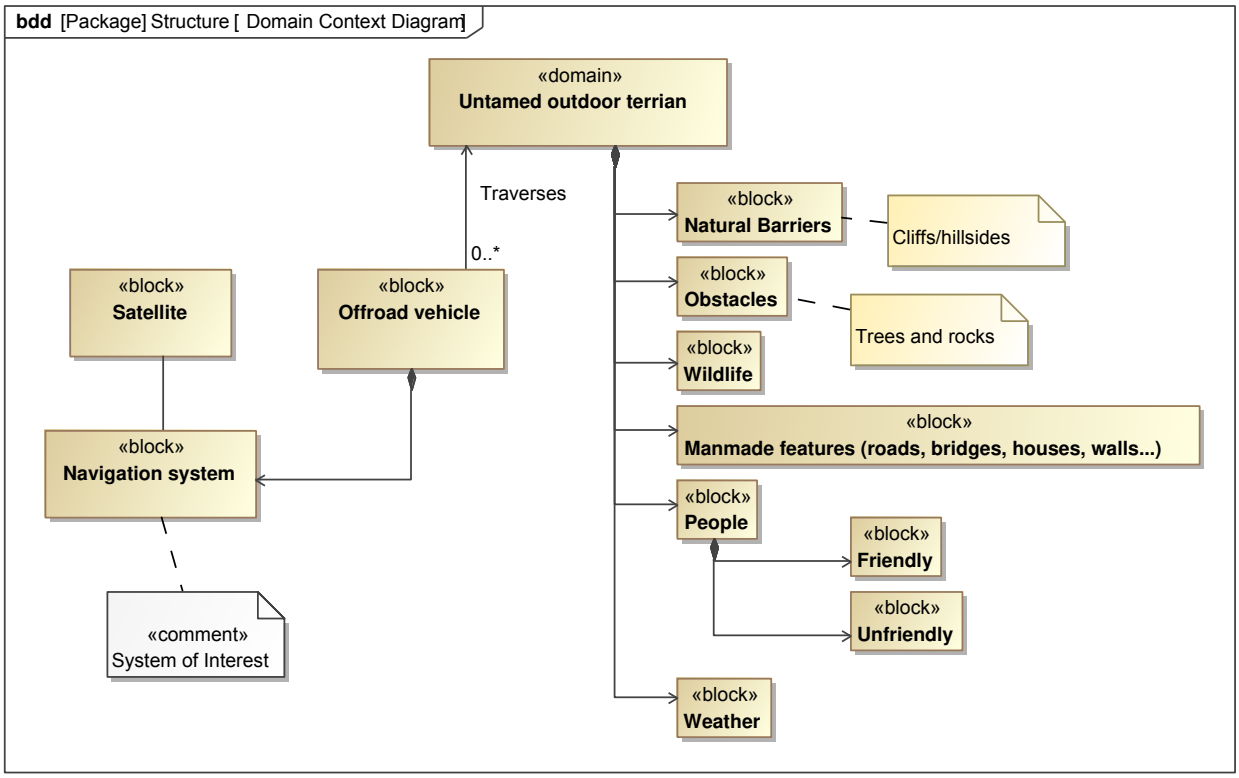

Figure 5.4: Off-Road Navigation system Context diagram

Off-road vehicle and is highlighted as the system of interest. The elements within the environment such as *natural barriers* and *Weather* represent objects that the system of interest will physically interact with, whereas the *satellite* represents an external actor that the navigation system will interact with for data and communication purposes.

5.3.3 Elaborate Use Cases for SOI

Having established the system's purpose and goals, and built a context diagram for the system, the use case diagram serves as a means of establishing behavioral goals for the system. It is important to note that this step should be completed with no preliminary architecture in mind, and should not introduce any design constraints. The Off-road navigation system use case diagram (Figure 5.5) reflects the key use cases for this system. In the diagram, the user is represented on the left, and the external actors are represented on the right.

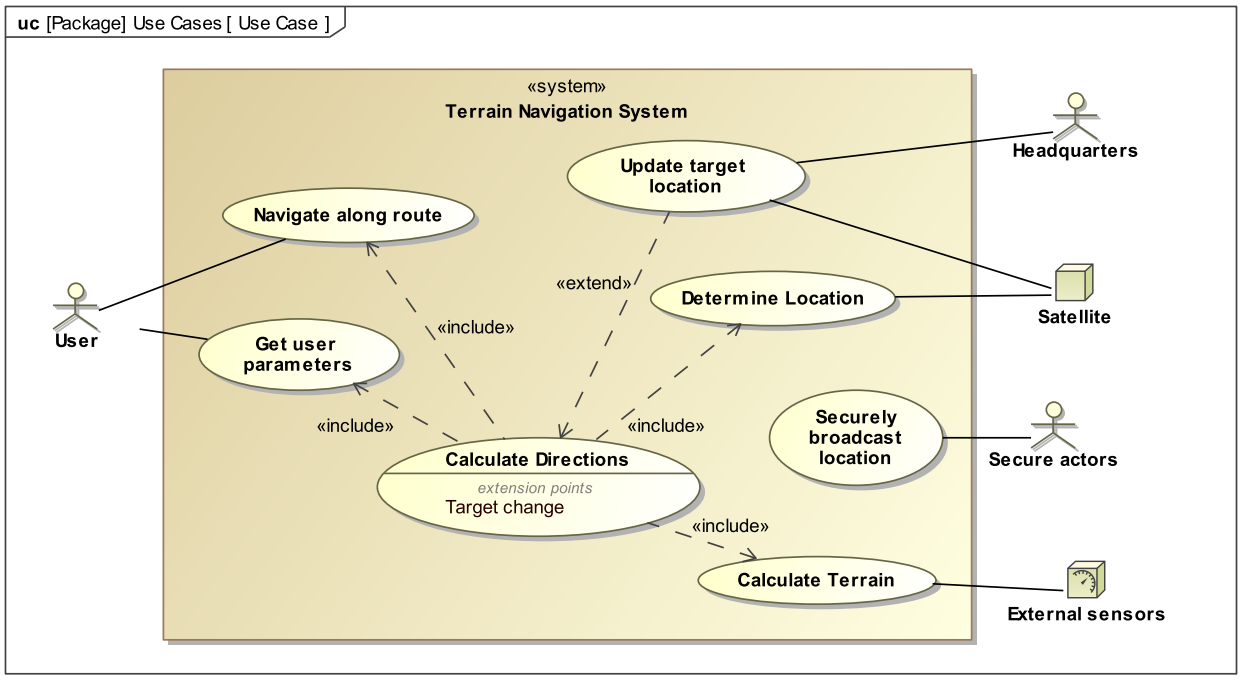

Figure 5.5: Off-Road Navigation system Use Case Diagram

As illustrated in the diagram, the two primary use cases for the user of the system are *navigate along route*, and *get user parameters*, which describe two primary functions that the system must accomplish from the end user perspective.

5.3.4 Define System Unacceptable Losses and Hazardous states

This step involves the systematic definition of system unacceptable losses and the identification of hazardous states that may lead to these losses. This step is supported by a simple block definition diagram used to document the hazard and loss elements, as well as trace the relationship between the two.

Two new block stereotypes, *Loss* and *Hazard* were introduced into this diagram, which enables the clear distinction between the types of elements in the diagram. These stereotypes also enable simple and easy traceability later in the model. As can be seen in Figure 5.6, the unacceptable losses include property damage, injury/death, loss of vehicle, and mission failure.

As the hazardous system states were elaborated and refined, the unacceptable Losses were clarified and updated to ensure that the losses encompassed all key undesirable outcomes of the

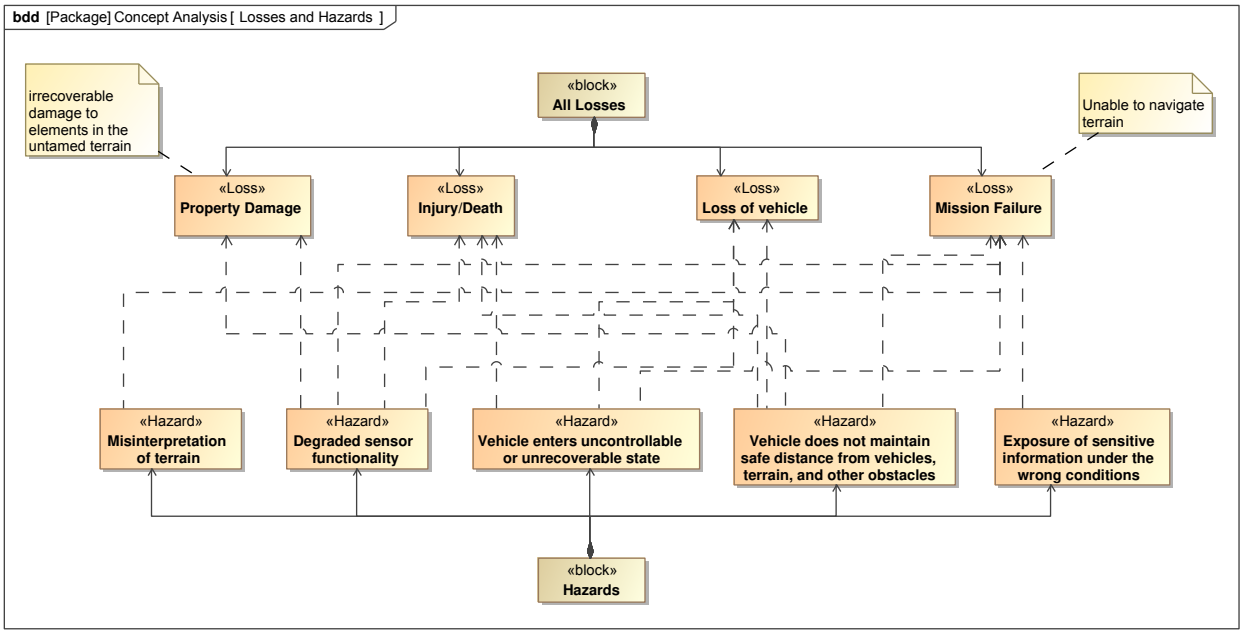

Figure 5.6: EGRESS Losses and Hazards diagram

stakeholders. Each hazard was then mapped to one or more unacceptable Losses, building the traceable relationships to ensure that each Loss and Hazard was relatable and required for the system.

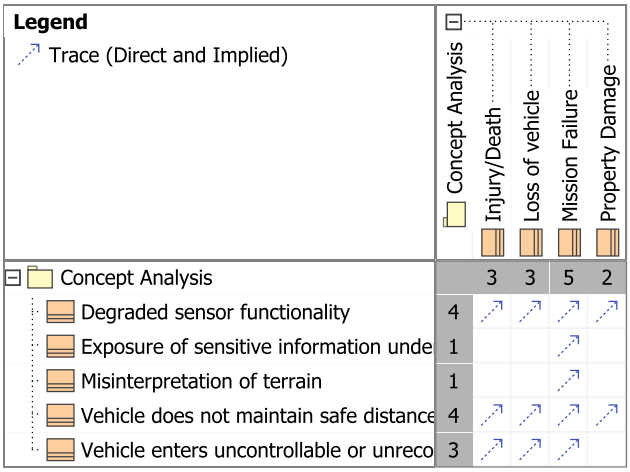

Figure 5.7: Hazards to Losses traceability

5.3.5 Initial Security Goals

Here we generate initial security goals for the Off-road navigation system based on the objective of preventing the system from entering the hazardous states defined in 5.3.4. As can be seen in the table, column one defines the security goal, column two describes the security goal, and column three allocates the goal to the appropriate Hazard(s) that it is designed to prevent. Figure 5.8 illustrates the initial security goals defined and allocated to the appropriate system hazard(s). For example, Security goal 3.1 titled *Critical Sensor ID*, is attributed to the system hazard *Degraded sensor functionality*

#	△ Name	Text	Id	Hazard
1	R 1 Sensor input validation	The system should validate sensor inputs against available terrain maps	1	Misinterpretation of terrain Vehicle enters uncontrollable or unrecoverable state
2	R 2 Secure comms	The system should not transmit mission critical information over unsecure methods	2	Exposure of sensitive information under the wrong conditions
3	⊟ R 3 Redundant Sensors	The system should have redundant safety critical sensors	3	Degraded sensor functionality Vehicle enters uncontrollable or unrecoverable state
4	R 3.1 Critical Sensor ID	Critical sensors for navigation and terrain mapping must be identified	3.1	Degraded sensor functionality
5	⊟ R 4 Safe separation	The system should maintain minimum user defined separation limits	4	Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles Vehicle enters uncontrollable or unrecoverable state
6	R 4.1 Min Separation Distances	The system should allow users to input minimum separation distances	4.1	Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles
7	R 5 Selectable position sharing	The system should have selectable position sharing capabilities	5	Exposure of sensitive information under the wrong conditions
8	⊟ R 6 Display and warning	The system should display the potential hazards and obstacles along route	6	Vehicle enters uncontrollable or unrecoverable state Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles
9	R 6.1 Audible and visible warnings	The system should have both audible and visual warning indicators for hazards along route	6.1	Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles Vehicle enters uncontrollable or unrecoverable state
10	R 7 Changes to destination	The user should be notified and accept any change to target destination	7	Vehicle enters uncontrollable or unrecoverable state

Figure 5.8: Initial System Security Goals

5.3.6 Elaborate Misuse Case Diagram

In this step of the process, the previously defined use case diagram is elaborated (Figure 5.9). An attacker element is added, and the use cases are reviewed for their susceptibility to CIA attacks. Once a susceptibility is defined, a misuse case is added to the diagram, and an activity diagram detailing the attack sequence is built to describe the attack. For the Off-road navigation system, the *availability attack* of blocking the systems' location services is a potential threat.

As demonstrated in Figure 5.9, a misuse case can exploit more than one system use case, which can be seen on the 'exploits' relationship highlighted. For clarity and emphasis, the misuse case

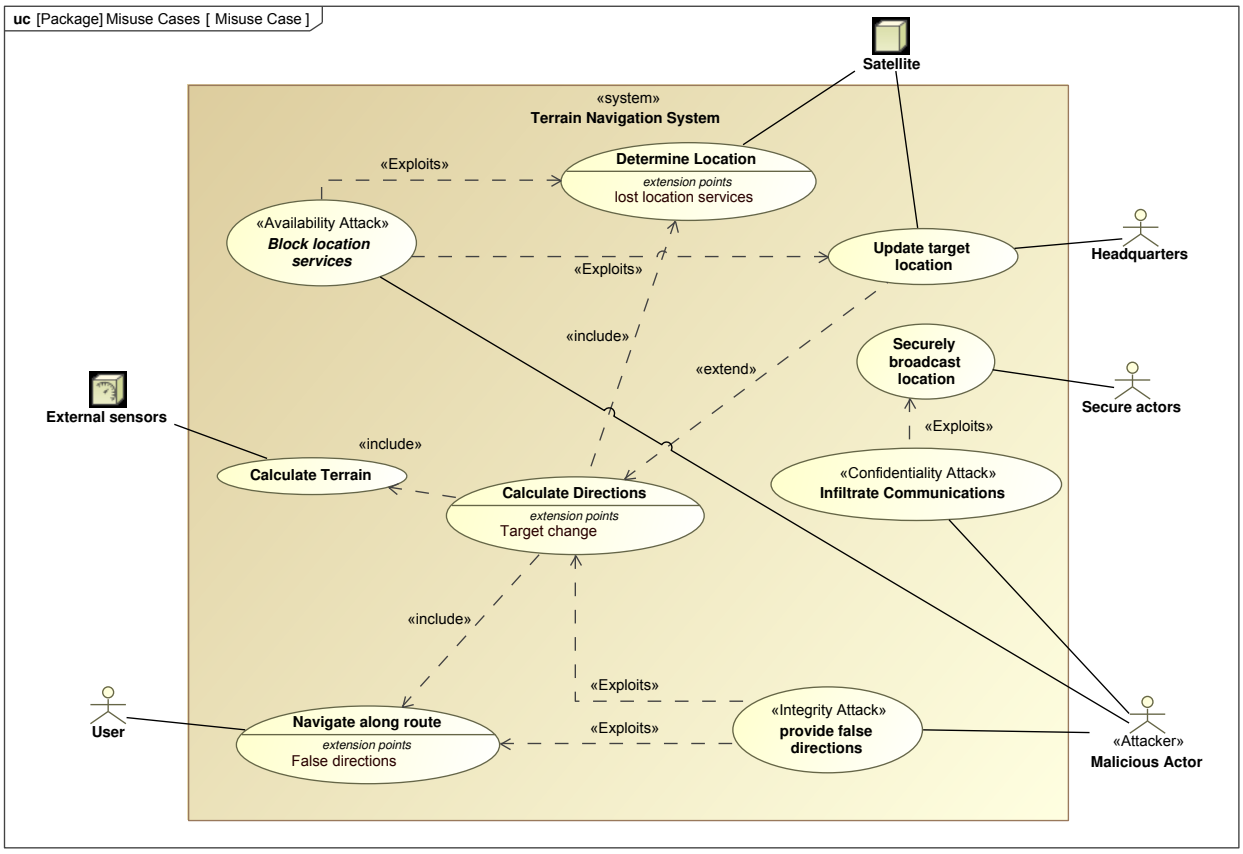

Figure 5.9: Off-Road Navigation system Misuse Case Diagram

elements and relationships are highlighted in red to distinguish malicious activity from operator goals.

5.3.7 Activity Diagramming and Security Critical Functions

In this step, activity diagrams are developed for each use case. As seen in Fig. 5.10 the activity diagrams detail the high-level activities that the system is expected to perform. In the case of this diagram, the Off-road navigation system is tasked with updating its coordinates, acquiring data from internal sensors, and satellites, and calculating the local terrain and elevation maps. Since receiving data from the satellite requires external communication, this activity was identified as a *security critical function*, and classified as an *Information Integrity* threat. Similarly, validating the authenticity of an external connection during secure communication is a *security critical function*, and is classified as an *Information Integrity* threat.

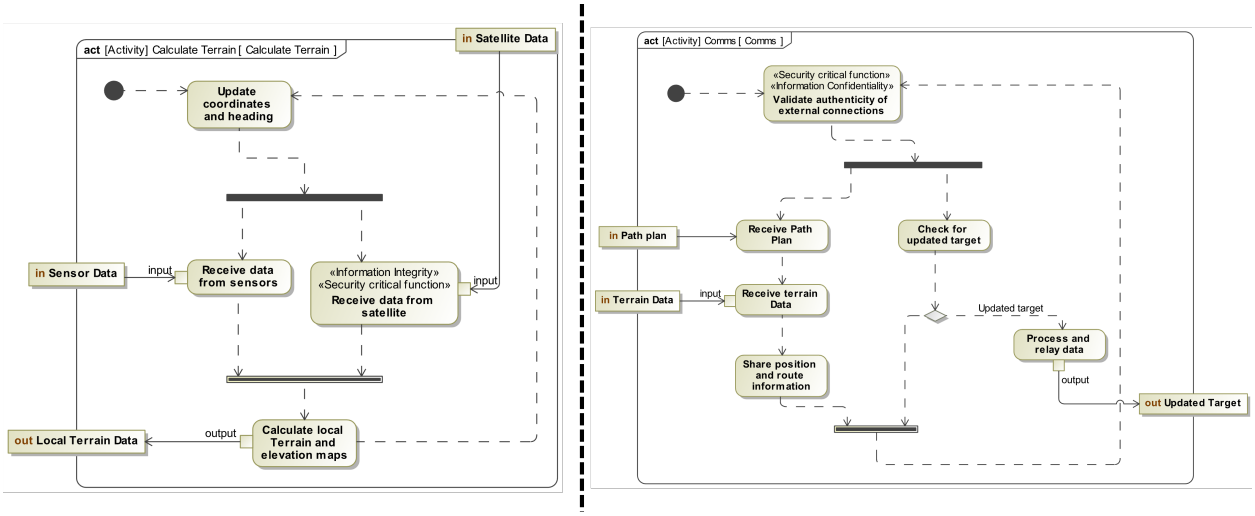

Figure 5.10: Security critical ID for 'Calculate Terrain' and 'Comms' use cases

The objective of this step is to validate that all security-critical functions are 'covered' by the defined security goals. These security-critical functions were then allocated to the appropriate security goals.

5.3.8 Goal Verification and final output

As described above, the goal of this step is to verify existing security goals and add any that are missing through a behavioral analysis of the system. In the case of the Off-road navigation system, an activity titled *authenticate user* was identified, as did not have a corresponding security goal. Therefore, a new security goal, *User Authentication and Authorization* was developed to accommodate for this gap.

As can be seen in Figure 5.11, the activity diagram highlights the key activities that the system is expected to accomplish as it obtains the user parameters before providing navigation. The activity *Authenticate user* was identified, and Security Goal number 12 was added (Fig 5.12 to ensure that all users are authenticated and authorized before use.

Fig 5.12 outlines the validated set of security goals derived from both structural and behavioral modeling of the system of interest.

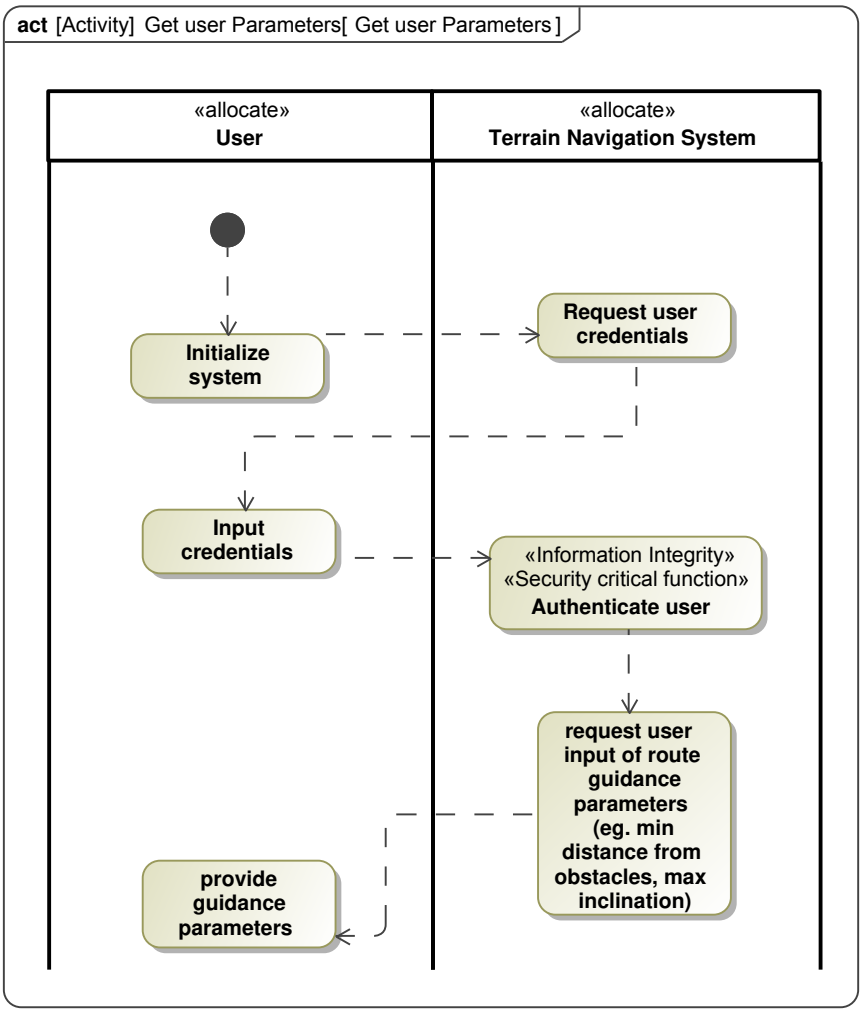

Figure 5.11: Security critical ID for 'Get User Parameters'

5.4 Discussion and Conclusions

This chapter introduces a novel method, EGRESS (Eliciting Goals for Requirement Engineering of Secure Systems), for eliciting security goals at the system level with a mission-focused approach. Utilizing best practices from loss-driven engineering analysis and employing Model-Based Systems Engineering (MBSE), EGRESS facilitates goal generation with enhanced traceability and a primary emphasis on mission-centric considerations. Addressing a critical gap in the conceptual design phase of cyber-physical system development, the work aligns with the INCOSE Vision 2035, emphasizing the need for security to be as foundational a requirement in system design as

#	△ Name	Text	Id	Hazard
1	R 1 Sensor input validation	The system should validate sensor inputs against available terrain maps	1	Misinterpretation of terrain Vehicle enters uncontrollable or unrecoverable state
2	R 2 Secure comms	The system should not transmit mission critical information over unsecure methods	2	Exposure of sensitive information under the wrong conditions
3	⊟ R 3 Redundant Sensors	The system should have redundant safety critical sensors	3	Degraded sensor functionality Vehicle enters uncontrollable or unrecoverable state
4	R 3.1 Critical Sensor ID	Critical sensors for navigation and terrain mapping must be identified	3.1	Degraded sensor functionality
5	⊟ R 4 Safe separation	The system should maintain minimum user defined separation limits	4	Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles Vehicle enters uncontrollable or unrecoverable state
6	R 4.1 Min Separation Distances	The system should allow users to input minimum separation distances	4.1	Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles
7	R 5 Selectable position sharing	The system should have selectable position sharing capabilities	5	Exposure of sensitive information under the wrong conditions
8	⊟ R 6 Display and warning	The system should display the potential hazards and obstacles along route	6	Vehicle enters uncontrollable or unrecoverable state Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles
9	R 6.1 Audible and visible warnings	The system should have both audible and visual warning indicators for hazards along route	6.1	Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles Vehicle enters uncontrollable or unrecoverable state
10	R 7 Changes to destination	The user should be notified and accept any change to target destination	7	Vehicle enters uncontrollable or unrecoverable state
11	R 8 External Connection Authentication and Authorization	The system should validate the authenticity of external connections (Satellites, other vehicles, C2) and limit external inputs to authorized entities only (white listing, role based access control) - Derrived from reviewing security critical functionality in activity diagrams	8	Exposure of sensitive information under the wrong conditions Misinterpretation of terrain
12	R 9 User Authentication and Authorization	The system should implement role based access control to ensure only valid users can input system parameters.	9	Vehicle enters uncontrollable or unrecoverable state Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles Exposure of sensitive information under the wrong conditions

Figure 5.12: System goals table with added constraints

functionality and safety. It contributes a best-practice-based approach for eliciting security goals during the early stages of complex CPS design using structural and behavioral system analysis.

Furthermore, the utility of implementing this method through MBSE provides a structured and systematic approach, enhancing the overall efficiency and effectiveness of the security goal elicitation process within the complex system design context. To further represent the utility of EGRESS, and its role within system design, the EGRESS method can be implemented within the MagicGrid Framework [69]. The MagicGrid framework is a structured approach to systems engineering, developed to address increasing complexity in the field. It organizes the system into layers of abstraction and pillars, emphasizing requirements, behavior, structure, and parameters across problem, solution, and implementation domains. Compatible with ISO 15288, the framework guides processes from business analysis to detailed design, integrating model-based design practices and facilitating interdisciplinary collaboration.

Figure 5.13 illustrates the implementation of the EGRESS steps using the MagicGrid MBSE framework in SysML. It is important to note that EGRESS is specifically designed to facilitate security requirement elicitation using conceptual design, whereas MagicGrid defines the modeling process in its entirety. As can be seen in the figure, EGRESS falls heavily on the conceptual

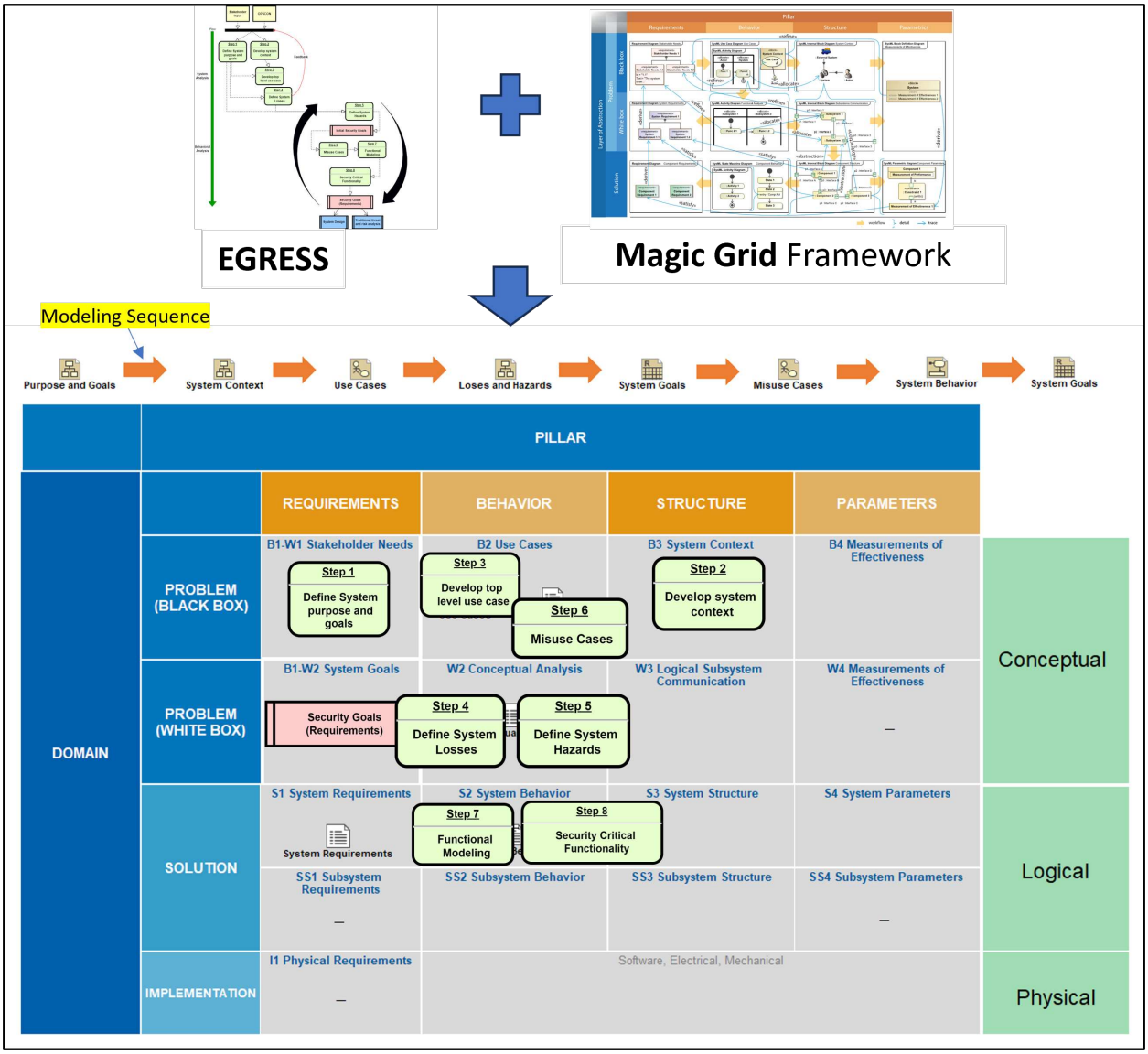

Figure 5.13: Implementation in SysML of an augmented magic grid using EGRESS showing a sequence of modeling activities

and problem domain region of the framework. This domain is dedicated to addressing the early stages of systems engineering, focusing on defining and understanding the problem that needs to be solved. The conceptual layer of abstraction involves analyzing stakeholder needs, conducting high-level conceptualization, and defining system requirements. Through its recommended diagrams, EGRESS can provide clear and coherent descriptions of the problem domain pertaining to security, laying the foundation for subsequent phases in the systems security engineering process. In order to accommodate for the use of hazard and loss analysis in EGRESS, the *Functional*

Analysis block under behavior was substituted for *Conceptual Analysis*. Future work involves developing a complete EGRESS profile which would add structure to the metamodel, and ensure that the creation and interpretation of model elements is accurate and consistent. This would also facilitate interoperability between different tools and modeling environments, promoting a standardized understanding of the system being modeled.

Chapter 6

Conclusion

This thesis presents an extensive exploration of Model-Based Systems Engineering (MBSE) methodologies applied to the field of systems engineering, with a particular focus on security, enhancing collaboration, eliciting security requirements, modeling threats and risks, and eliciting security goals. Below are the major conclusions from each chapter:

MBSE Simulation for Collaborative Systems Engineering:

This first chapter laid the foundation by showcasing the potential of MBSE simulation in enhancing collaboration among systems engineers, stakeholders, and experts. The system of interest (J1939) was detailed and structurally modeled through SysML modeling tools. SysML simulation tools were effectively employed to expose vulnerabilities in network protocols, providing dynamic visualizations that aid in decision-making and fostering cross-functional collaboration.

Investigation of the MBSEsec method

The second chapter introduced MBSEsec as a robust methodology for eliciting security requirements and modeling threats in cyber-physical systems. The methodology was applied to the J1939 protocol to develop security controls that would mitigate the previously modeled vulnerabilities. The structure of the MBSEsec method, along with its dynamic characteristics, allowed for iterative modifications to the model. The incorporation of inherent traceability within the model significantly simplified the detection of deficiencies in security controls and requirements. Notable enhancements, including the introduction of the "attacker" element and adherence to zero trust principles, strengthen the methodology's applicability to fielded systems, ensuring a proactive approach to cybersecurity.

Enhancements to MBSEsec for Cyber-Physical Systems:

The third chapter critically evaluated the MBSEsec methodology through its application to a new system autonomous off-road system (MRZR) and proposed enhancements, addressing shortcomings in the method's threat modeling and risk assessment. Elements such as roll-up patterns, parametrics, new stereotypes, and simulation were used to produce a more comprehensive model that enabled a more detailed understanding of the threats and risks to the system. The integration of qualitative and quantitative risk assessment approaches, along with new diagrams, enhances the method's effectiveness in evaluating and mitigating system risks.

EGRESS for Mission-Focused Security Goal Elicitation:

The fourth chapter introduced EGRESS, a novel method focusing on mission-centric security goal elicitation in the early stages of complex cyber-physical system design. The use of MBSE to produce necessary artifacts through a model greatly improved the utility of the approach. Aligning with the INCOSE Vision 2035, EGRESS provides a structured and systematic approach for security goal elicitation, contributing to the foundational role of security in system design.

6.1 Future Work

Working on this thesis uncovered additional research questions that did not fall within the original scope of the work. Future work in this field may involve further exploration of MBSE simulation with parameters and logic that more accurately reflect system behavior. Quantitative evaluation of the benefits of model simulation to improve communication and collaboration between SMEs and systems engineers would be highly beneficial.

Further refinement of MBSEsec to component-level requirements would help an MBSE model embrace the role of 'source of truth' for a system under design and would help bridge the gap between software developers and systems engineers. It is also recommended to conduct a detailed comparison of the results of using MBSEsec and other security threats and risk analysis methods. This work provides a foundation for demonstrating the application of MBSEsec to a cyber-physical

system and the promising merits of utilizing MBSE for secure system design, but I do not provide an exhaustive comparison against the results of other security analysis approaches.

Lastly, future work could be done on discussing if a new item 'System Goal' should be a formally represented item in an MBSE requirements table. Currently, SysML does formally differentiate between a system requirement vs. goal. As such we used the stereotype 'requirement' but the addition of formal goals within SysML could provide utility clarity beyond just security applications.

Bibliography

[1] Society of Automotive Engineers (SAE), "SAE J1939 Standards Collection." Available online: https://www.sae.org/standardsdev/groundvehicle/j1939a.html.

[2] D. Mažeika and R. Butleris, "MBSEsec: Model-Based Systems Engineering Method for Creating Secure Systems," *Applied Sciences*, vol. 10, p. 2574, Apr. 2020.

[3] "Polaris Details New MRZR Alpha Light Tactical Vehicle for SOCOM." Available online: https://military.polaris.com/en-us/mrzr-alpha/.

[4] M. San Miguel, J. H. Johnson, J. Kertesz, K. Kaski, A. Díaz-Guilera, R. S. MacKay, V. Loreto, P. Érdi, and D. Helbing, "Challenges in complex systems science," *The European Physical Journal Special Topics*, vol. 214, pp. 245–271, Nov. 2012.

[5] D. D. Walden, G. J. Roedler, K. J. Forsberg, R. D. Hamelin, and T. M. Shortell, *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. John Wiley & Sons, Inc, 4 ed., July 2015.

[6] A. M. Madni and M. Sievers, "Model-based systems engineering: Motivation, current status, and research opportunities," *Systems Engineering*, vol. 21, no. 3, pp. 172–190, 2018. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sys.21438.

[7] Office of the Deputy Assistant Secretary of Defense for Systems Engineering, "Digital engineering strategy," tech. rep., Department of Defense, Washington, DC, 2018. Available online: https://man.fas.org/eprint/digeng-2018.pdf.

[8] International Council on Systems Engineering (INCOSE), "Model-Based Systems Engineering (MBSE) Initiative," tech. rep., International Council on Systems Engineering (INCOSE), 2022. Available online: https://www.incose.org/communities/working-groups-initiatives/mbse-initiative.

[9] J. Holt, S. Perry, M. Brownsword, D. Cancila, S. Hallerstede, and F. O. Hansen, "Model-based requirements engineering for system of systems," in *2012 7th International Conference on System of Systems Engineering (SoSE)*, pp. 561–566, July 2012.

[10] W. Duo, M. Zhou, and A. Abusorrah, "A Survey of Cyber Attacks on Cyber Physical Systems: Recent Advances and Challenges," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, pp. 784–800, May 2022. Conference Name: IEEE/CAA Journal of Automatica Sinica.

[11] S. Boulevard, "Cyber attacks on the power grid," *https://securityboulevard.com/2022/05/cyber-attacks-on-the-power-grid/*, 2022.

[12] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," p. 6, 2011.

[13] H. Kang, B. I. Kwak, Y. H. Lee, H. Lee, H. Lee, and H. K. Kim, "Car Hacking and Defense Competition on In-Vehicle Network," in *Proceedings Third International Workshop on Automotive and Autonomous Vehicle Security*, (Virtual), Internet Society, 2021.

[14] P. H. Nguyen, S. Ali, and T. Yue, "Model-based security engineering for cyber-physical systems: A systematic mapping study," *Information and Software Technology*, vol. 83, pp. 116–135, Mar. 2017.

[15] K. X. Campo, T. Teper, C. E. Eaton, A. M. Shipman, G. Bhatia, and B. Mesmer, "Model-based systems engineering: Evaluating perceived value, metrics, and evidence through literature," *Systems Engineering*, vol. 26, no. 1, pp. 104–129, 2023. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sys.21644.

[16] Y. Roudier and L. Apvrille, "SysML-Sec: A model driven approach for designing safe and secure systems," in *2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pp. 655–664, Feb. 2015.

[17] V. Crespi, A. Galstyan, and K. Lerman, "Top-down vs bottom-up methodologies in multi-agent system design," *Autonomous Robots*, vol. 24, pp. 303–313, Apr. 2008.

[18] International Council on Systems Engineering (INCOSE), "Systems engineering vision 2035," tech. rep., International Council on Systems Engineering (INCOSE), 2022. Available online: https://www.incose.org/publications/se-vision-2035.

[19] D. Firesmith, "Engineering Security Requirements.," *The Journal of Object Technology*, vol. 2, no. 1, p. 53, 2003.

[20] P. Salini and S. Kanmani, "Survey and analysis on Security Requirements Engineering," *Computers & Electrical Engineering*, vol. 38, pp. 1785–1797, Nov. 2012.

[21] L. Delligatti, *SysML Distilled: A Brief Guide to the Systems Modeling Language*. Addison-Wesley Professional, 2013.

[22] C. Delp, D. Lam, E. Fosse, and C.-Y. Lee, "Model based document and report generation for systems engineering," in *2013 IEEE Aerospace Conference*, pp. 1–11, Mar. 2013. ISSN: 1095-323X.

[23] "OMG Unified Modeling Language (UML) Specification." Object Management Group (OMG), 2017. Version 2.5.1 *https://www.omg.org/spec/UML/2.5.1/About-UML*,.

[24] M. C. Berschik, T. Schumacher, F. N. Laukotka, D. Krause, and D. Inkermann, "MBSE WITHIN THE ENGINEERING DESIGN COMMUNITY – AN EXPLORATORY STUDY," *Proceedings of the Design Society*, vol. 3, pp. 2595–2604, July 2023.

[25] B. S. Blanchard and W. J. Fabrycky, *Systems engineering and analysis*. Prentice-Hall international series in industrial and systems engineering, Englewood Cliffs, N.J.: Prentice-Hall.

[26] P. De Saqui-Sannes, R. A. Vingerhoeds, C. Garion, and X. Thirioux, "A Taxonomy of MBSE Approaches by Languages, Tools and Methods," *IEEE Access*, vol. 10, pp. 120936–120950, 2022. Conference Name: IEEE Access.

[27] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers (IEEE), "ISO/IEC/IEEE 15288:2023 Systems and software engineering — System life cycle processes," Standard ISO/IEC/IEEE 15288, International Organization for Standardization (ISO), 2023. Available online: https://www.iso.org/standard/81702.html.

[28] E. S. Palma, E. Y. Nakagawa, D. M. B. Paiva, and M. I. Cagnin, "Evolving reference architecture description: Guidelines based on iso/iec/ieee 42010," 2022.

[29] Joint Task Force Interagency Working Group, "Security and Privacy Controls for Information Systems and Organizations," tech. rep., National Institute of Standards and Technology, Sept. 2020. Edition: Revision 5.

[30] International Organization for Standardization (ISO) and Society of Automotive Engineers (SAE), "ISO/SAE 21434: Road Vehicles - Cybersecurity Engineering," Standard ISO/SAE 21434, SAE International, 2021. Available online: https://www.iso.org/standard/70918.html.

[31] R. Ross, M. Winstead, and M. McEvilley, "Engineering Trustworthy Secure Systems," Tech. Rep. NIST Special Publication (SP) 800-160 Vol. 1 Rev. 1, National Institute of Standards and Technology, Nov. 2022.

[32] D. Mažeika and R. Butleris, "Integrating Security Requirements Engineering into MBSE: Profile and Guidelines," *Security and Communication Networks*, vol. 2020, pp. 1–12, Mar. 2020.

[33] C. Raspotnig, V. Katta, P. Karpati, and A. L. Opdahl, "Enhancing CHASSIS: A Method for Combining Safety and Security," in *2013 International Conference on Availability, Reliability and Security*, pp. 766–773, Sept. 2013.

[34] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development," in *UML 2002 — The Unified Modeling Language* (G. Goos, J. Hartmanis, J. van Leeuwen, J.-M. Jézéquel,

H. Hussmann, and S. Cook, eds.), vol. 2460, pp. 412–425, Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. Series Title: Lecture Notes in Computer Science.

[35] M. Brunner, M. Huber, C. Sauerwein, and R. Breu, "Towards an Integrated Model for Safety and Security Requirements of Cyber-Physical Systems," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, (Prague, Czech Republic), pp. 334–340, IEEE, July 2017.

[36] D. Mazeika and R. Butleris, "Identifying Security Issues with MBSE while Rebuilding Legacy Software Systems," in *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*, pp. 83–86, June 2020.

[37] M. H. Larsen, G. Muller, and S. Kokkula, "A Conceptual Model-Based Systems Engineering Method for Creating Secure Cyber-Physical Systems," *INCOSE International Symposium*, vol. 32, no. S2, pp. 202–213, 2022. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/iis2.12909.

[38] M. T. Span, L. O. Mailloux, M. R. Grimaila, and W. B. Young, "A Systems Security Approach for Requirements Analysis of Complex Cyber-Physical Systems," in *2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1–8, June 2018.

[39] M. Deshmukh, "Security requirements engineering process," in *Seminar in Information System, Security Engineering*, Citeseer, 2009.

[40] S. U. Rehman, C. Allgaier, and V. Gruhn, "Security Requirements Engineering: A Framework for Cyber-Physical Systems," in *2018 International Conference on Frontiers of Information Technology (FIT)*, pp. 315–320, Dec. 2018. ISSN: 2334-3141.

[41] B. Fabian, S. Gürses, M. Heisel, T. Santen, and H. Schmidt, "A comparison of security requirements engineering methods," *Requirements Engineering*, vol. 15, pp. 7–40, Mar. 2010.

[42] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh, "Security Requirements Engineering: A Framework for Representation and Analysis," *IEEE Transactions on Software Engineering*, vol. 34, pp. 133–153, Jan. 2008.

[43] A. Shostack, *Threat Modeling: Designing for Security*. Hoboken, New Jersey: John Wiley & Sons, 2014.

[44] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, "STRIDE-based threat modeling for cyber-physical systems," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, (Torino), pp. 1–6, IEEE, Sept. 2017.

[45] J. S. Park, D. Kim, S. Hong, H. Lee, and E. Myeong, "Case Study for Defining Security Goals and Requirements for Automotive Security Parts Using Threat Modeling," pp. 2018–01–0014, Apr. 2018.

[46] A. Y. Javaid, W. Sun, V. K. Devabhaktuni, and M. Alam, "Cyber security threat analysis and modeling of an unmanned aerial vehicle system," in *2012 IEEE Conference on Technologies for Homeland Security (HST)*, (Waltham, MA, USA), pp. 585–590, IEEE, Nov. 2012.

[47] D. Klets, I. V. Gritsuk, A. Makovetskyi, N. Bulgakov, M. Podrigalo, I. Kyrychenko, O. Volska, and N. Kyzminec, "Information Security Risk Management of Vehicles," pp. 2018–01–0015, Apr. 2018.

[48] Society of Automotive Engineers (SAE), "SAE J1939-71: Vehicle Application Layer," Standard J1939-71, SAE International, Aug. 2022. Available online: https://www.sae.org/standards/content/j1939/71_202208/.

[49] R. Chatterjee, S. Mukherjee, and J. Daily, "Exploiting Transport Protocol Vulnerabilities in SAE J1939 Networks," in *Proceedings Inaugural International Symposium on Vehicle Security & Privacy*, (San Diego, CA, USA), Internet Society, 2023.

[50] S. Mittal, U. Durak, and T. Ören, eds., *Guide to Simulation-Based Disciplines: Advancing Our Computational Future*. Simulation Foundations, Methods and Applications, Cham: Springer International Publishing, 2017.

[51] B. Zeigler, S. Mittal, and M. Traore, "MBSE with/out Simulation: State of the Art and Way Forward," *Systems*, vol. 6, p. 40, Nov. 2018.

[52] D. Mažeika, *Model-based systems engineering method for creating secure systems*. PhD thesis, Kauno technologijos universitetas, 2021.

[53] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Waltham, MA: Morgan Kaufmann, 2nd ed ed., 2012. OCLC: ocn754518532.

[54] S. Stachowski, R. Bielawski, and A. Weimerskirch, "Cybersecurity research considerations for heavy vehicles," tech. rep., University of Michigan, Ann Arbor, Transportation Research Institute, 2019.

[55] P.-S. Murvay and B. Groza, "Security Shortcomings and Countermeasures for the SAE J1939 Commercial Vehicle Bus Protocol," *IEEE Transactions on Vehicular Technology*, vol. 67, pp. 4325–4339, May 2018. Conference Name: IEEE Transactions on Vehicular Technology.

[56] D. Wagner, "Building More Resilient Cybersecurity Solutions for Infrastructure Systems," in *Systems Engineering in the Fourth Industrial Revolution*, pp. 415–443, John Wiley & Sons, Ltd, 2019.

[57] C. Team82, "Claroty Biannual ICS Risk and Vulnerability Reprt 1H 2021." https://claroty.com/resources/reports/2h-2021, 2021.

[58] J. Haase and G. Rogers, "Testing of can signals," *Aerospace Testing International*, vol. 40, pp. 43–44, 2015.

[59] Aeronautical Radio, Incorporated (ARINC), "ARINC 825: Aircraft Data Network, Part 4: Deterministic Ethernet Networks," Standard ARINC 825, Aeronautical Radio, Incorporated (ARINC), 2018. *https://www.sae.org/standards/content/arinc825-4/*.

[60] A. Brehmer and R. Lotoczky, "CAN based protocols in avionics," in *2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC)*, pp. 1–33, Oct. 2014. ISSN: 2155-7209.

[61] D. Dubois, *Evaluating the Security of ARINC-825 and Controller Area Networks, the Impact of Bus Security in Aerospace*. M.E., McGill University (Canada), Canada – Quebec, CA, 2018. ISBN: 9798582579786.

[62] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," tech. rep., National Institute of Standards and Technology, Aug. 2020.

[63] P. Curtis, M. Carey, C. of Sponsoring Organizations of the Treadway Commission, *et al.*, "Risk assessment in practice," 2012.

[64] S. Mauw and M. Oostdijk, "Foundations of Attack Trees," in *Information Security and Cryptology - ICISC 2005* (D. H. Won and S. Kim, eds.), vol. 3935, pp. 186–198, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. Series Title: Lecture Notes in Computer Science.

[65] J. M. Woodruff, "Consequence and likelihood in risk estimation: A matter of balance in UK health and safety risk assessment practice," *Safety Science*, vol. 43, pp. 345–353, June 2005.

[66] M. Span, L. O. Mailloux, and M. R. Grimaila, "Cybersecurity architectural analysis for complex cyber-physical systems," *The Cyber Defense Review*, vol. 3, no. 2, pp. 115–134, 2018.

[67] B. Lundgren and N. Möller, "Defining Information Security," *Science and Engineering Ethics*, vol. 25, pp. 419–441, Apr. 2019.

[68] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), "ISO/IEC 27000:2018 Information technology – Security techniques – In-

formation security management systems," Standard ISO/IEC 27000:2018, (ISO), 2018. *https://www.iso.org/standard/73906.html*.

[69] A. Morkevicius, A. Aleksandraviciene, D. Mazeika, L. Bisikirskiene, and Z. Strolia, "MBSE Grid: A Simplified SysML-Based Approach for Modeling Complex Systems," *INCOSE International Symposium*, vol. 27, no. 1, pp. 136–150, 2017. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.2334-5837.2017.00350.x.

Appendix A

Chapter 3 Diagrams

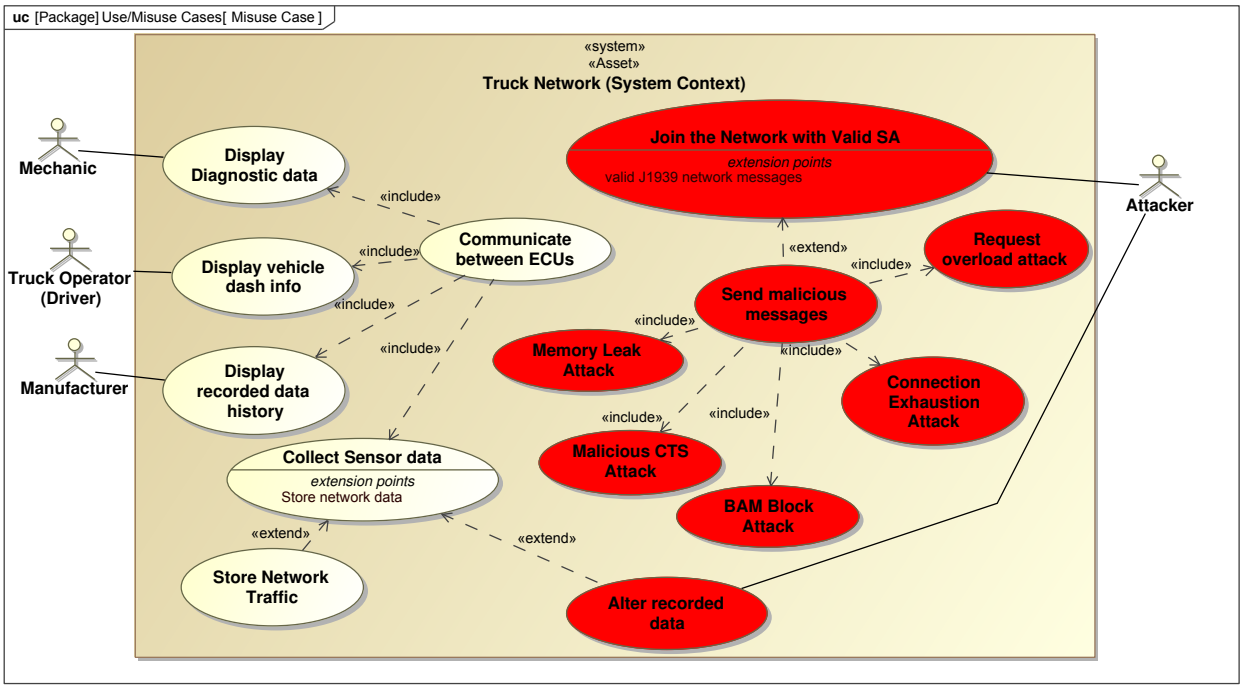

Figure A.1: Misuse Case Diagram

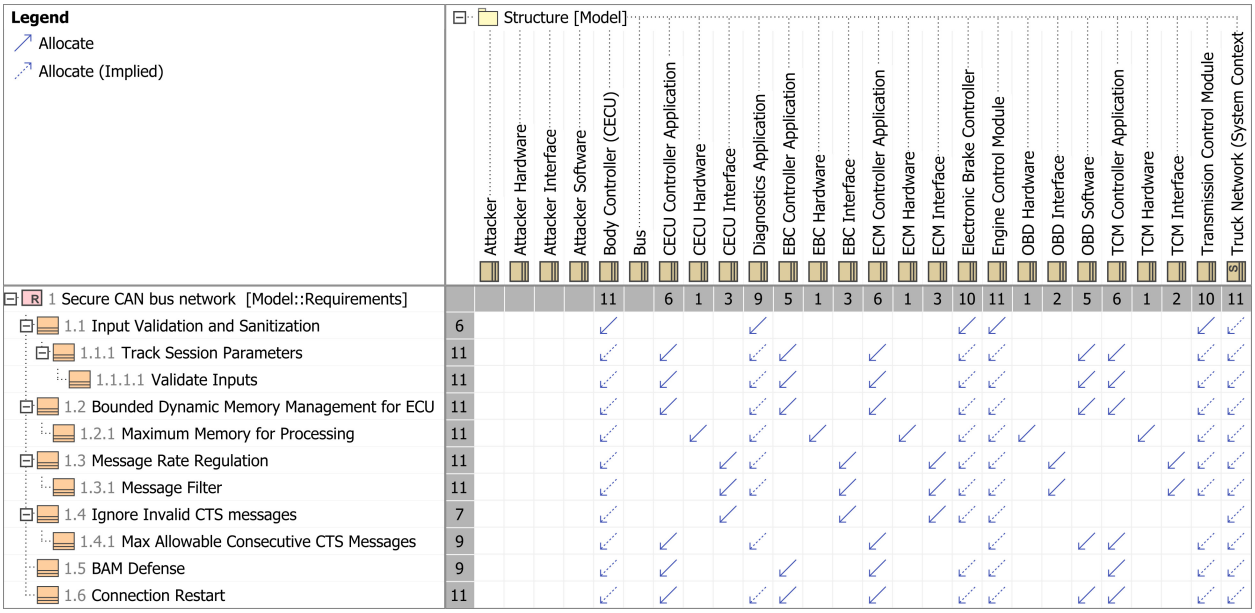

Figure A.2: Security requirements allocated to system assets

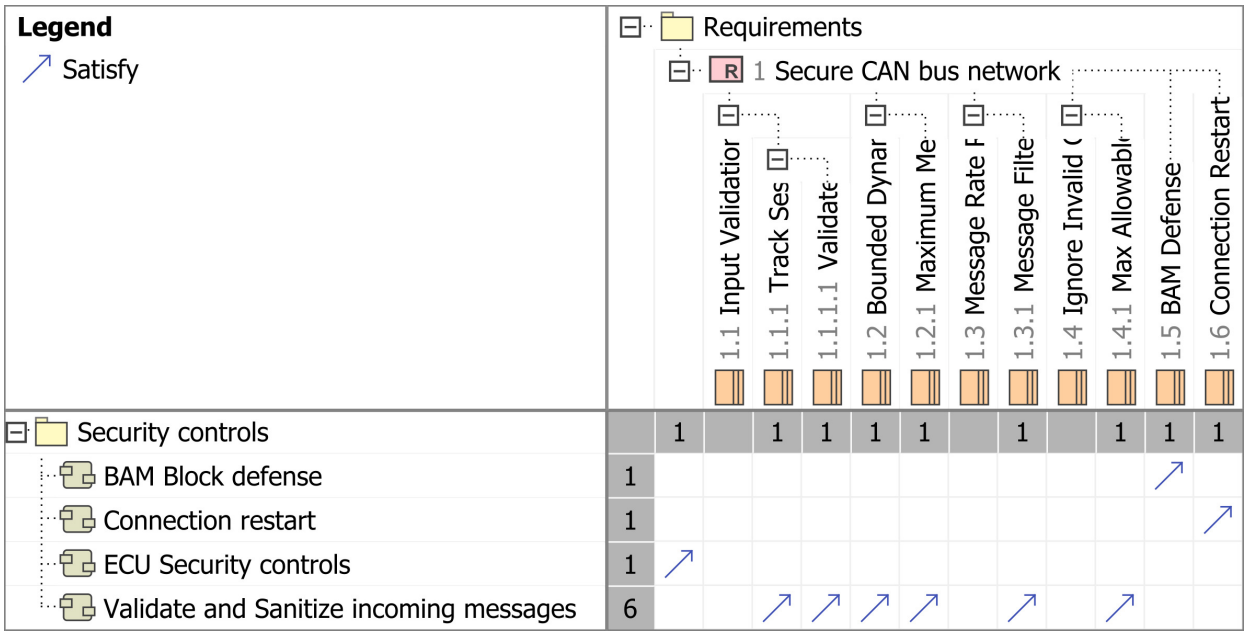

Figure A.3: Requirements satisfaction

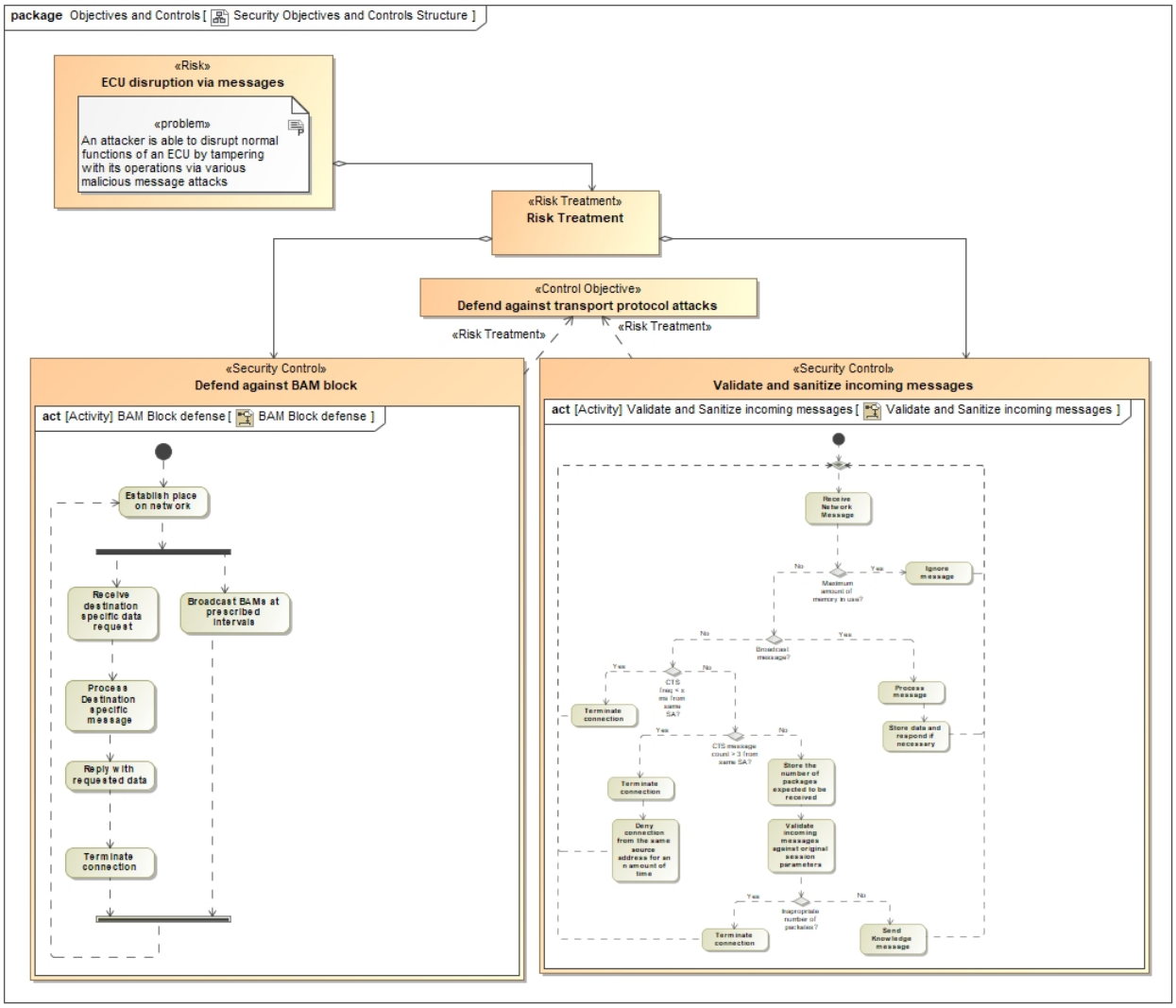

Figure A.4: Security Objectives and Control Structure

Appendix B

Chapter 4 Diagrams

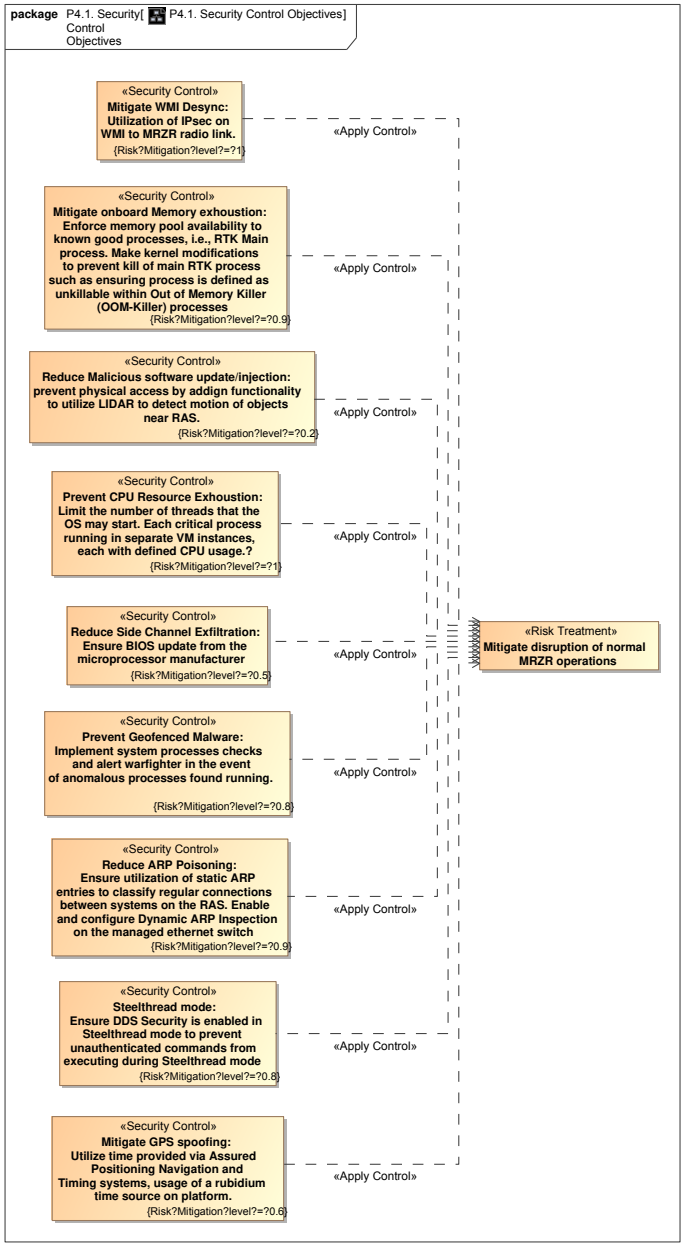

Figure B.1: New Security Controls Diagram