Threat Modeling for Train Control and Management Systems based on the Ethernet Train Backbone

Johan Holmberg

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 10.10.2016

Thesis supervisor:

Prof. Ville Kyrki

Thesis advisor:

M.Sc. Altti Helläkoski

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Johan Holmberg

Title: Threat Modeling for Train Control and Management Systems based on the Ethernet Train Backbone

Date: 10.10.2016 Language: English Number of pages: 8+78

Department of Electrical Engineering and Automation

Professorship: Automation Technology

Supervisor: Prof. Ville Kyrki

Advisor: M.Sc. Altti Helläkoski

Ethernet technologies are being increasingly adopted for use as the communication network of Train Control and Management Systems (TCMSs). With the introduction of the Ethernet Train Backbone (ETB) standard, Ethernet is set-up to become the default choice.
The use of technologies and solutions from the Information Technology (IT) sector may potentially introduce new security threats against TCMSs. There is little knowlege within the railway industry of how to tackle these new risks.
Threat modeling has been a central part of security analysis and threat mitigation within the IT sector. The purpose of this thesis is to study best practices from IT, select the most promising methodologies and adapt them as well as test their suitability for the railway industry.
In this thesis the most fitting threat modeling methodologies were merged into a threat modeling framework. The framework was tested in a case study by analysing a TCMS design based on a real world system. The expectations for the framework set prior to the case study were met and threat modeling was deemed beneficial for analysing the security of TCMSs. Further practical improvements to be studied were identified and suggested.

Keywords: Threat Modeling, Train Control and Management System, Security, Ethernet Train Backbone

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Johan Holmberg

Työn nimi: Ethernet Train Backbone junaverkkoon perustuvien junahallintajärjestelmien uhkamallintaminen

Päivämäärä: 10.10.2016 Kieli: Englanti Sivumäärä: 8+78

Sähkötekniikan ja automaation laitos

Professuuri: Automaatiotekniikka

Työn valvoja: Prof. Ville Kyrki

Työn ohjaaja: DI Altti Helläkoski

Junateollisuudessa Ethernet verkot ovat saamassa vahvan jalansijan junahallintajärjestelmissä. Tätä edesauttaa Ethernet Train Backbone (ETB) standardi.
IT-maailmasta tuttujen tekniikoiden tuonti juniin saattaa kuitenkin tuoda mukanaan uusia tietoturvauhkia ja riskejä. Junateollisuudesta ei löydy juuri kokemusta tietoturvauhkien hallintaan liittyen.
Uhkamallinnuksesta on IT-maailmassa tullut yksi tärkeä osa tietoturvauhkien torjuntaa. Tämän opinnäytetyön tarkoitus on tutkia IT-alalla käytettyjä uhkamallinnustekniikoita, valita ja soveltaa parhaimmat junanhallintajärjestelmien uhkamallinnukseen sekä kokeellisesti osoittaa niiden toimivuus.
Tässä työssä parhaista uhkamallinnustekniikoista muodostettiin viitekehys junahallintajärjestelmien uhkamallinnusta varten. Viitekehystä testattiin uhkamallintamalla oikeaan projektiin perustuvan junahallintajärjestelmän. Ennen tapaustutkimusta asetettiin tavoitteet, jotka tapaustutkmuksessa pääosin täyttyivät. Uhkamallinnus todettiin sopivan junahallintajärjestelmien tietoturvan analysointiin. Tapaustutkimuksessa esiin tulleita mahdollisia viitekehyksen parannuksia identifioitiin ja ehdotettiin tulevia tutkimuksia varten.

Avainsanat: Uhkamallinnus, Junahallintajärjestelmä, Tietoturva, Ethernet Train Backbone

Preface

Writing this thesis has been a roller coaster. Euphoric moments at its best, contrasted with long but slow train rides through deep motivational valleys. However, an intuition of an extremely important subject is the thought that kept me pushing on.

I would like to thank supervisor Professor Ville Kyrki for his support and patience with a student, who during the past 2 years presented a number of thesis-topics not countable on one hand. Keep up your enthusiasm, it helped me a long way! I would also like to thank my advisor M.Sc. Altti Helläkoski for the interesting discussions we had as well as the advice and guidance you gave me.

I would also like to express my appreciation for everything my parents have done for me through the years. Likewise, I am also thankful to my sisters, who have always been my role models and helped me in finding my path.

Last but most importantly, thank you my dear Jessica for your never ending support. You picked up the pieces every time I fell, you believed in me when I did not. I finally pulled through and I am eternally grateful!

Espoo, 10.10.2016

Karl J. P. Holmberg

Contents

Abbreviations and Definitions

Abbreviations

AA	Attacker Action
API	Application Programming Interface
ARP	Address Resolution Protocol
BCU	Brake Control Unit
BDMP	Boolean logic Driven Markov Process
CSTINFO	Consist info telegram sent as part of ETB inauguration
DAG	Directed Acyclic Graph
DCU	Door Control Unit
DDU	Driver Display Unit
DFD	Data Flow Diagram
DHCP	Dynamic Host Configuration Protocol
DoS	Denial of Service
ECN	Ethernet Consist Network
ED	End Device
ETB	Ethernet Train Backbone
ETBN	Ethernet Train Backbone Node
ERU	Ethernet Router Unit
ESU	Ethernet Switch Unit
Gbps	Gigabits per second
GDU	Guard Display Unit
GPS	Global Positioning System
HVAC	Heating, Ventilation and Air Conditioning
ICMP	Internet Control Message Protocol
IEC	International Electrotechnical Commission
IGMP	Internet Group Management Protocol
ISE	Instant Security Event
IT	Information Technology
LAN	Local Area Network
MAC	Message Authentication Code
Mbps	Megabits per second
MVB	Multifunction Vehicle Bus
NTP	Network Time Protocol
PD	Process Data
PES	Passenger Entertainment System
PIS	Passenger Information System
PISC	Passenger Information System Client
PISM	Passenger Information System Manager
PLC	Programmable Logic Controller
PMU	Power Management Unit
TCMS	Train Control and Management System (also known as Train Management System)

TCN	Train Communication Network
TCP	Transmission Control Protocol
TCU	Traction Control Unit
TPM	Time and Position Manager
TRDP	Train Real-Time Data Protocol
TSE	Timed Security Event
TTDB	Train Topology Database
TTW	Train To Wayside
UDP	User Datagram Protocol
UIC	Union Internationale des Chemins de Fer
UML	Unified Modeling Language
VCU	Vehicle Control Unit
WTB	Wire Train Bus

Definitions

Consist	Part of train which forms an inseparable unit
Inauguration	The initial configuration phase of a Train Communication Network
Operator	A company or organisation that provides freight or passenger rail services
Rolling stock	Train coaches, locomotives and other vehicles with wheels moving on tracks
Stepping stone	An intermediate target that has to be achieved in order to reach the main goal
Traction	The system that provides mobility to a train, i.e. enables the train to move
Train composition	The current set of rolling stock vehicles that form a train
Train guard	The train conductor, i.e. the crew member that is responsible for all other on-board duties except driving

1 Introduction

Security of vital infrastructure has in recent years become an important topic across the world, especially with revelations of cyber-attacks on industrial control systems using highly sophisticated malware such as Stuxnet. Stuxnet revealed in 2010 that industrial control systems are susceptible to cyber-attacks which effectively can cause physical damage to infrastructure [1], [2]. Trains are an important part of public transport infrastructure in a lot of countries and concerns related to security of on-board electronic equipment have increased. Train Control and Management Systems (TCMSs) are a central part of a train and enable comprehensive train control and monitoring capabilities, through which train operation, availability and safety are improved and assured [3]. TCMSs are in many ways similar to industrial control systems in that a TCMS controls and monitors a real-time system where synchronization of up to thousands of sub-systems is required for safe operation. Hence, there is an increasing need for reviewing and improving TCMS security.

Despite security concerns, TCMSs are becoming more automated and complex. Adding a lot of new sensors and controllers, video surveillance as well as on-board broadband internet access for passengers creates a high demand for increased data transfer bandwidth within a TCMS. In response to these needs, Ethernet has been introduced as the new main data carrier technology to be used on trains. The Ethernet Train Backbone (ETB) (IEC 61375-2-5) [4] standard specifies a new communication network which is recommended to be used for data transfer in TCMSs. Ethernet provides several benefits, such as improved network management capabilities and bandwidth speeds of up to even 10 Gbps. Ethernet equipment and know-how is also widely available due to it being ubiquitous in the modern IT sector.

However, the IT sector has faced serious security problems with Ethernet networks since the rise of the modern Internet. A recent survey [5] listed several different threats to Local Area Networks (LANs), such as Denial of Service (DoS) attacks, Media Access Control (MAC) flooding, Address Resolution Protocol (ARP) and Dynamic Host Configuration Protocol (DHCP) poisoning. These attacks combined with application level problems can be used to gain illegal access to networks, affect traffic confidentiality, traffic integrity and system security or may simply be used to stop the system from functioning through a DoS attack [5]. The question remains, will security issues frequent in the IT sector and visible on today's Internet have a serious impact on the railway industry? Correct operation of a TCMS is crucial for train safety, as hazards may lead to loss of life. Hence, the effect that security has on safety is an important topic when deploying Ethernet on rolling stock in the railway industry.

The concept of security is still very new to the railway industry when regarding rolling stock and especially TCMSs. The new ETB standard does not specify how security should be approached on board trains. The situation is similar with industrial control systems in that methods, practices and themes common in the IT sector are still uncommon in relation to industrial control systems [6, p. 2]. The fact remains that due to the ever increasing complexity of TCMSs, with the introduction of Ethernet as the main network technology and with the requirements for better

and more comprehensive remote access and remote control features, the security aspects of TCMSs can not be ignored. A safe system can be made unsafe through the exploitation of security vulnerabilities of the system [7]. Therefore, a pure hazards analysis is no longer enough to provide an acceptable level of safety. As already mentioned, even if a system is not made unsafe, an attacker can cause serious equipment availability issues for instance with the use of a DoS attack. In an area where trains are important, halting an entire fleet can have serious economic or political consequences by preventing the movement of a large number of people. The security risks affecting TCMS safety and availability have to become part of the risk analysis process when designing such a system. The interdependencies between safety and security have indeed been increasingly studied in recent years [8], [9].

Threat modeling may provide the capabilities to answer these mentioned concerns. One definition for threat modeling is that it is the act of systematically identifying threats to a software system [10, p. 1609]. In this work the definition is extended to mean any system and also to include the rating and impact analysis of threats based on criteria such as potential damage, frequency of occurrence, etc. During the last decade, numerous different threat modeling methodologies have been developed [11], [12], [10]. The results of threat modeling practices can be used to evaluate risks and possible countermeasures to protect systems. So far, threat modeling has mainly been used within the IT sector.

The purpose of this thesis is to study threat modeling methodologies used in IT systems today and evaluate how they would suite to be used for analysing security threats against modern TCMSs based on the ETB standard. If necessary, modifications will be done to the existing methodologies. The chosen methodologies are tested by threat modeling a generic ETB based TCMS. Due to the complexity of a complete TCMS, only a single sub-system of the TCMS will be threat modeled. This sub-system will be chosen to best replicate generic traits of a TCMS. The TCMS system itself is specifically designed and specified for this work with the help of the case study company. A new TCMS has to be designed as the case study company currently has no pure ETB based TCMS in use. However, the TCMS design is based on an existing Ethernet TCMS and experiences from other real world projects. The intention is to identify and analyse threats to the chosen sub-system of the TCMS in order to determine how well threat modeling fits into the design process of a TCMS. The goal is to come up with recommendations for a customized TCMS threat modeling process to be used on a regular basis.

The security concerns mentioned before arise mainly from the introduction of the ETB standard and the use of Ethernet as the main communication network technology in the train. Hence, only the ETB network and TCMS components directly attached to it are considered in this work. External interfaces, actors and attack vectors into the ETB itself will not be analysed in detail. The threat modeling will be done based on the assumption, that a malicious adversary has already gained access to the TCMS. The desire is to strengthen the internal structure of the TCMS by tightly integrating threat modeling activities in the existing TCMS development processes. Additionally, many threat modeling methodologies contain both qualitative and quantitative aspects. A qualitative approach fits better as an initial survey into the

topic with the intention of increasing understanding about threat modeling. Problems related to quantitative aspects e.g. evaluating numeric values for various traits, such as for instance mean time to successful attack or cost of attack, are not considered within scope of this work. Finally, this work does not strive to a combined safety and security analysis. Topics regarding safety are only indirectly of importance when arrived at through security flaws and vulnerabilities.

The text is structured as follows. Background information about the railway industry and threat modeling concepts are presented in chapter 2. The following chapter 3 provides a more in depth review of various threat modeling approaches and methodologies. Some of these methodologies are chosen and tested as part of a case study, which is described in chapter 4. The results of the case study are presented in chapter 5 while conclusions are provided in chapter 6.

2 Background

This chapter presents background information about TCMSs, communication networks used in trains and threat modeling. Section 2.1 presents the TCMS and describes general properties of a typical TCMS. Section 2.2 provides background information related to the original communication network used on trains, the Train Communication Network (TCN) (IEC 61375-1) standard [13]. The TCN is important for understanding basic concepts and requirements regarding trains and TCMSs. Section 2.3 covers the characteristics of the ETB, including changes and improvements over the original TCN. Security concerns related to introducing the ETB in trains are reviewed in Section 2.4. Lastly, the concept of threat modeling is presented in section 2.5. Threat modeling can be interpreted in many ways, which necessitates defining terminology. Section 2.5 functions as a basis for chapter 3 where actual threat modeling methodologies are presented.

2.1 Train Control and Management Systems

Train Control and Management Systems (TCMSs) have always existed on trains in some form, the first systems being based on train whistles or air pipes as well as interaction between crew using whistles and flags. A modern TCMS is in essence a complex data processing system [3, pp. 6-7], which consists of all devices and equipment enabling the control and management of all, or some of the functions of a train. Examples of these high level functions include traction, braking, doors, lighting and air-conditioning. The purpose of the TCMS is to provide control and monitoring capabilities of the functions and over the equipment installed and integrated into the train. An example of part of a TCMS is depicted in Figure 1.

Every TCMS needs some sort of means for sending and receiving data between systems. This capability is provided to the TCMS by a communication network, which can be a manufacturer specific custom solution or something based either

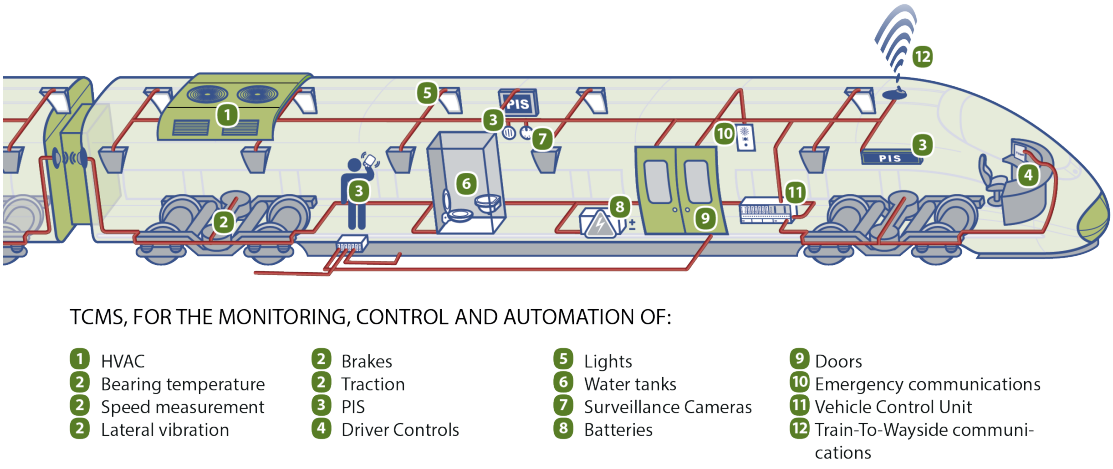

Figure 1: Overview of a TCMS.

partly or completely on the TCN standard presented in section 2.2. In Figure 1, the numbered items are the TCMS sub-systems, which are connected to each other by the communication network depicted as the red lines.

A complete train usually comprises several rolling stock vehicle types, i.e. locomotives, power cars, coaches, etc. Each vehicle contains a varying set of sub-systems and devices that are needed for controlling and monitoring the status of that particular vehicle or vehicle type. In the generic case the train length can vary. This means that either vehicles or some set of vehicles can be attached or detached to and from the train. An obvious example is when an operator needs more passenger capacity during rush hours and therefore adds passenger coaches to the trains. After these peak usage periods there is no need to pull the extra weight introduced by the additional coaches, so they will be removed. The current structure or configuration of the train is called the train composition. Static trains are trains where the compositions is fixed, while dynamic trains allow the composition to change during normal operation.

The structure of a train is illustrated in Figure 2. The train structure is divided into sections as follows. Individual rolling stock vehicles are the base unit of the train. One or several rolling stock vehicles may form so called consists. Consists are inseparable sections of the train, i.e. consists have to be attached or detached to or from a train as a complete unit. One or several consists then form the train. Thus, depending on the design of the train, one or many rolling stock vehicles span a consist while one or many consists span a train and thereby also the entire TCMS.

A typical TCMS has a Vehicle Control Unit (VCU) in each vehicle. The VCU often has the main responsibility for controlling the systems in the vehicle. It often functions as a gateway between the vehicle and the rest of the train. Other typical devices in a vehicle are: Door Control Units (DCUs), Brake Control Units (BCUs), Heating, Ventilation and Air Conditioning (HVAC), Passenger Information System (PIS), and Traction Control Units (TCUs). The systems are highly interconnected, in that they often require a lot of different information to function. For instance, in hot countries air conditioning is usually halted in order to save power when the

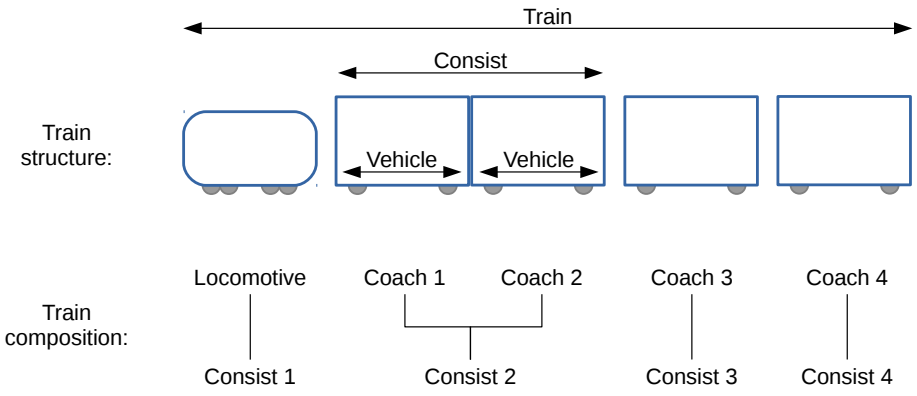

Figure 2: The structure of a train.

trains external doors are opened at stations. Another example is that the external doors are usually prohibited from opening if the train is travelling faster than some predefined threshold. Thus, the air conditioning unit needs information about the doors while the doors in turn need speed information from the train. The VCU typically relays needed information to and from the vehicles sub-systems and raises alarms for the driver if anomalies are detected.

2.2 Train Communication Network

Communication networks on trains enable the equipment of the TCMS to communicate with each other. According to Neil [3, pp. 1-2], the first electronic control systems used function or subsystem dedicated equipment and wiring. This was not a very scalable approach as cabling costs rapidly increased along with the number of systems. Communication networks based on serial communication techniques used in other industries provided a solution to the rising costs. Utilizing these in trains allowed removing redundant cabling and reducing costs by allowing multiple systems to transmit data over a single communication link.

Train manufacturers and subsystem suppliers adapted newly developed fieldbus technologies for trains and created proprietary technologies and protocols for communication. TCMS suppliers quickly realized that device interconnections were still too cumbersome to manage, especially when TCMS complexity increased. The need for a common standard was obvious [14, p. 798]. As a result the International Electrotechnical Commission (IEC) in collaboration with Union Internationale des Chemins de Fer (UIC) released the Train Communication Network (TCN) (IEC 61375-1) [13] and the Information transmission in the train (UIC 556 leaflet) [15] standards.

The TCN standard lays down hardware specifications, medium and link protocols defining inter-link communication as well as higher level real-time protocols which define methods for node-to-node communication. The purpose of the TCN is to provide the TCMS with the means for communicating control and diagnostics data between equipment and systems. The TCN standardization process also meant specifying some general requirements for the TCN. These were:

1. The TCN has to be suitable for use on rolling stock, referring for instance to mechanical robustness and electro-magnetic interference [16, p. 4].

2. Automatic reconfiguration upon changed train composition [16, p. 4], [17, p. 1117]. Unlike in automotive, aviation or ship industries, train compositions may change during normal operation. Trains or individual vehicles can be coupled and detached, the direction of travel may change. The TCN has to reconfigure itself in these events.

3. The TCN should allow for varying data scope and performance metrics, i.e. there are different needs such as latency, throughput, etc. for control signals versus system diagnostics or monitoring data [16, p. 4].

4. The TCN should provide high levels of reliability, availability and maintainability [16, p. 4], [17, p. 1117]. Even minor subsystem failures can cause injury or loss of life. Additionally, a running train depends on many interacting subsystems enabling safe operation. Compromising safety is considered unacceptable in the industry and a train out of operation usually quickly generates considerable losses for the operator.

A communication network based on the TCN encompasses one or more buses. In the general case where the train composition is dynamic and may change, a layered approach is required with at least two buses; One train bus and one vehicle bus. Such an example is shown in Figure 3. There is a clear separation between the train bus and the vehicle buses. The purpose of the train bus is to interconnect all the rolling stock vehicles to each other and enable communication between devices in different consists. Therefore the train bus must be able to automatically reconfigure itself as needed and handle related issues such as addressing and train direction determination. The vehicle buses simply interconnect equipment within a vehicle or consist, depending on design. A gateway node in each vehicle or consist relays data as needed between the train bus and the local vehicle bus. This network structure and the properties of the train bus allow vehicles or consists to be attached or detached from the train. Trains with static and fixed compositions may discard the concept of a train bus and utilize only a single vehicle bus. This is due to the fact that the network topology can be precisely predefined and never changes. The entire train is regarded as a single consist. A layered communication network design might still be adopted for other reasons, but it is not strictly required.

The TCN standard defines these different use cases as application domains [13, pp. 44-45]:

– Open train configuration

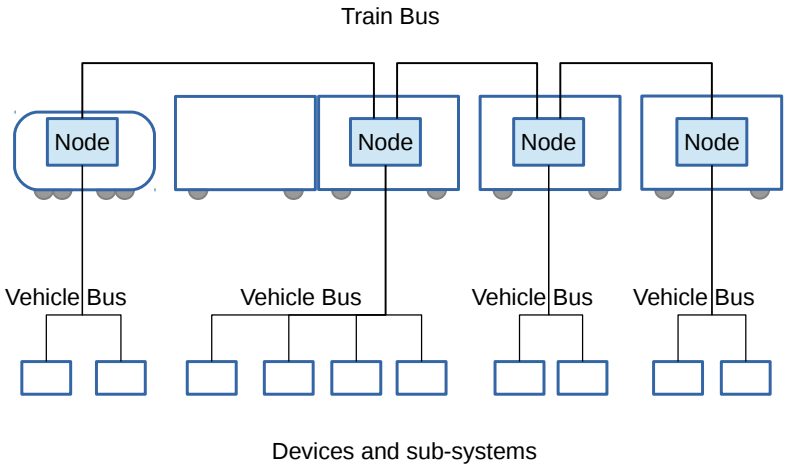

Figure 3: Generic TCN structure [13, p. 44].

- Multiple unit train configuration
- Closed train configuration

Open trains correspond to dynamic composition trains and closed trains to static composition trains while the multiple unit train configuration is specified as multiple connected closed trains. These domains are perfectly applicable to the structure of a TCMS in general, which is no surprise since the structure of a TCMS is to a large extent defined by the network.

The TCN standard introduced two new buses, the Wire Train Bus (WTB) and the Multifunction Vehicle Bus (MVB). Having been specifically designed for the railway industry, the WTB fulfils the properties of a train bus while the MVB was intended for use as a vehicle bus. The WTB has a bandwidth of 1Mbps while the MVB can do up to 1.5Mbps. Both buses are managed by a bus master, which guarantees deterministic data transfer times [18, pp. 84-85]. The bus master of the WTB is decided as a result of the so called inauguration phase. The inauguration is the process where the train topology is determined. Once the train topology is determined, every node on the bus, i.e. vehicle, knows the following [18, p. 83]:

- The bus address, position relative to the master, orientation (same as train or reversed)
- The address of other vehicles on the bus
- The type and version of other vehicles
- Dynamic properties of the train, e.g. is there a driver present somewhere

The TCN real-time protocols provide two mechanisms for communication: periodic process data and sporadic message data [18, p. 85]. Process data is cyclically broadcast data which can be used by devices to report their most important status data. Each node on both the WTB and MVB is reserved a timeslot for sending its predefined process data. However, not all data sent on the MVB may be sent over the WTB due to bandwidth limitations. The message data transfer facility should be used for larger, event like data transfers which can be split up into several parts and sent subsequently. Transfer time for message data is not deterministic by design as the amount of data being sent as well as the frequency at which it is being sent is not strictly predefined.

2.3 Ethernet Train Backbone

The original TCN standard has seen wide adoption around the globe since its introduction, even if many TCMS solutions only implement part of it. However, during the last decade, the trend has been to add more data bandwidth intensive functionalities, e.g. comprehensive video surveillance, on-board broadband Internet access for passengers, interactive passenger entertainment systems and remote monitoring and software update capabilities. These new systems have rendered the original TCN

standard partly obsolete, as the bandwidth capabilities of the buses presented in the standard are insufficient for current demands.

The Ethernet Train Backbone (ETB) (IEC 61375-2-5) [4] introduced in 2014 as an addition to the original TCN, brings the TCN to the 21st century. It improves bandwidth capabilities as well as network dynamics. IEC 61375-2-5 contains descriptions of the actual backbone network, while higher level protocols and application interfaces are specified in the IEC 61375-2-3 [19]. Like the original TCN, the ETB uses the same high level real-time protocol specification called Train Real-Time Data Protocol (TRDP). This means that while the underlying medium and link layer has changed, the application interface remains mostly the same compared to the TCN based on the WTB and MVB buses. However, because Ethernets are inherently built out of point-to-point connections using packet switching for relaying data, some new features have been introduced to TRDP. The ETB can enable bandwidth speeds of up to even 10 Gbps and moreover, Ethernet equipment and know-how is widely available due to it being ubiquitous in the modern IT sector. This makes Ethernet an attractive technology for the next TCN solution.

The structure of the ETB is similar to the original TCN and is illustrated in Figure 4. There is at least one backbone network, the ETB, running along the length of the entire train. Communication within individual vehicles or consists is handled by one or more Ethernet Consist Networks (ECN). Ethernet Train Backbone Nodes (ETBN) function as gateways and route traffic between the ECN and ETB, enabling communication between so called End Devices (ED).

The TRDP services of the ETB are implemented on top of the common protocol stack used in IT. More specifically data is sent using the Internet Protocol (IP) for network transfer and either the User Datagram Protocol (UDP) or the Transmission Control Protocol (TCP) for transport. Additionally the Internet Control Message Protocol (ICMP) and Internet Group Management Protocol (IGMP) need to be

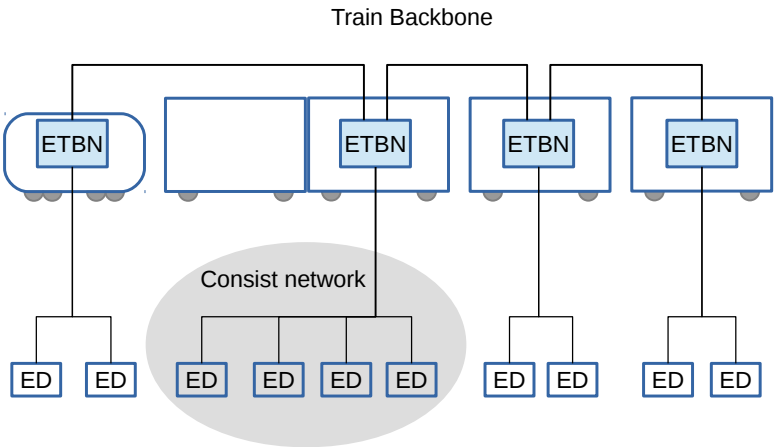

Figure 4: Structure of the ETB [4, p. 38]

supported for network management. The ETB protocol stack is shown in Figure 5. IP addresses are handed out to equipment on the network during the inauguration procedure which in turn is controlled by the Train Topology Discovery Protocol. In practice, each ETBN is constantly checking for new nodes connected to its ETB interfaces. If a new node is discovered or if one that previously existed has disappeared a new train inauguration takes place. During inauguration there is one device in each ECN responsible for sending out a so called CSTINFO telegram which contains information about all devices connected to that specific ECN. The information in the combined CSTINFO telegrams are used to construct the Train Topology Database (TTDB), which contains the train's complete network information, such as global and local IP addresses for each ED and domain names. There is a fully functioning Domain Name System (DNS) that uses the TTDB for name resolution. It allows End Devices (ED) on the network to be addressed by domain names such as for instance "doorController4.consist2.ltrain" instead of cryptic numeric IP addresses.

Similarly to the TCN, each ED connected to an ECN has two basic methods of communication: Process data and message data. A big difference to the original TCN is that the ETB supports multiple process data packets per ED. Each ED is allowed to transmit several different process data packets corresponding to the different functions that the node controls or monitors. While the WTB or MVB bus-master cyclically gives each node on the bus a time slot to transfer its one set of process data, the ETB does not centrally control the transmission at all. The process data is simply sent by the ED and then routed by the network using the IGMP and UDP protocols. An ED, for instance a control computer or a brake controller, may send one or more process data packets to specific predefined IP multicast addresses for each packet. A specific data content is identified with the combination of the source IP of the sending device, the destination multicast IP and by a so called communication identifier value. The EDs on the receiving side have to register

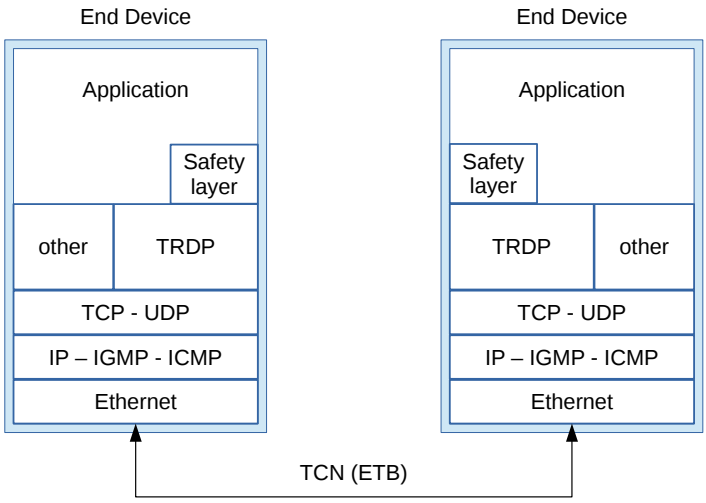

Figure 5: Protocol stack of the ETB [4, p. 114].

their interest in receiving process data sent to specific IP multicast addresses. This means that each Ethernet Train Backbone Node (ETBN) knows which ED in its ECN are interested in specific process data packets and reroutes the data packets accordingly. This will save some bandwidth because only necessary data is routed from the backbone to the consists.

To sum up, the ETB standard provides higher bandwidth for trains in response to increasing data transfer needs. Additionally, new features are added to the TRDP which make the ETB a more flexible network and enhance the services provided by the original TCN standard.

2.4 Security concerns with the Ethernet Train Backbone

The security aspects of the original TCN have not seriously been evaluated since its introduction in 1999. Even so, the WTB and MVB buses have some good security properties. Communication is controlled by an active bus manager which controls deterministic data transfer. The WTB and MVB are also less known in general and while security through obscurity is not something to rely on, it does mean that the script-kiddie type of attacker does not pose a threat to TCN based trains.

While the ETB improves performance and makes the TCN more dynamic and configurable, it does introduce new security related questions. First of all, the process data mechanism seems extremely vulnerable to some device or equipment pretending to be another ED. Because process data utilises UDP, there is no sense of a verified connection between the two communicating entities. Data monitoring could also be easy as anyone can register to the needed multicast address and receive the process data they are interested in.

Secondly, the fact that hard addressing is not used can also cause problems. If IP addresses are resolved from the TTDB, then a successful attack on the TTDB could enable the attacker to do a man in the middle attack and take control of the communication between two EDs. Similar attacks can of course also be targeted at the lower levels of the Ethernet/IP stack.

Moreover, the network is not controlled by a bus master like the WTB in the original TCN. It would seem intuitive that Denial of Service attacks by flooding the network with garbage data could be done. The ETB does support the concept of quality of service, but will it be enough to absolutely assure data bandwidth for critical applications?

Another notion is that since the ETB supports dynamic train compositions, i.e. the concept of open trains, it can not be known at the initial design exactly what devices and sub-systems will be connected to the network in later years. This is a strong reason for why the ETB should not be treated as a closed network, where all defences are concentrated on fending off threats at the borders to the network. The internal boundaries within the ETB have to be examined as well.

Finally, the ETB relies on the common TCP/IP protocol stack used on the Internet. There is wide-spread know-how of how to attack systems using these protocols. At the same time the railway industry is not equally familiar with these technologies, creating an undesired gap in knowledge between potential attackers and

defenders. It is imperative to close this gap, further motivating why threat modeling practices for railway rolling stock applications should be evaluated.

2.5 Threat Modeling

Threat modeling is difficult to define and there have been many definitions as a result. Examples include *"the process of identifying threats to systems"* [10] and *"involves understanding the complexity of the system and identifying all possible threats to the system"* [20]. These kind of definitions are narrow in scope and rely heavily on the concept of a threat without defining what it means. For instance, what is the definition of a threat, i.e. can a threat be a person, an object or perhaps an event? Further, does threat modeling only encompass identifying threats but not analysing their severity or mitigation techniques? Shostack [12, p. xxii, p. 30] provides an overview of different meanings to the word threat modeling. According to Shostack, threat modeling can mean:

- A verb: Asking someone, are you threat modeling?
- A noun: Concluding that our threat model is an ex employee with inside information.
- An objective: Listing potential targets of an attacker.

The examples above approach threat modeling from different perspectives. A truly generic definition to threat modeling would have to encompass all the cases above and possibly more. Shostack concludes that threat modeling is simply a technical term for the practical process of thinking about what and how things can go wrong with a system. *"Threat modeling is the use of abstractions to aid in thinking about risks"* [12, p. xxiii]. This definition has benefits such as avoiding the ambiguity of defining what a threat is as well as covering a wide array of different approaches. Additionally, both people with technical and economic background can relate to the word risk, as it can be associated to both technical risk and business risk. Discussing risks of various sort seems logical in relation to threat modeling. The biggest problem with Shostack's proposal is that threat modeling could be mistaken for simple risk management. While not far from the truth as security affects safety, threat modeling does not focus on malfunction or accidents as the source of problems. For the purpose of this thesis, threat modeling is the use of abstractions to aid in thinking about risks related to security.

There are a lot of different threat modeling methods. Some are very specific in scope and procedures, something that characterizes scientific methods. Others function more as general guidelines of where to start and how to continue analysing system security. The wide array of approaches to threat modeling may be categorized in an effort to better understand how they fit together. There are at least three structured approaches to threat modeling; Asset centric, attacker centric and software centric. An asset centric approach focuses on finding the most important assets of for example a company. It is more a method of identifying what should be protected and

thereby concentrating resources where most needed. The goal with attacker centric methods is to find out what motivates the adversary and if possible what his resources and capabilities are. A deep understanding of the attacker enables the mitigation of the attacker's perceived malicious actions. Finally, software centric approaches concentrate more on threat mitigation through system correctness. Additional attention is paid to security aspects during system development in order to prevent vulnerabilities from ever being created.

Shostack notes that both the asset centric and attacker centric perspectives have severe drawbacks [12, pp. 36-43]. Focusing on assets has the drawback that it is difficult to define what the system assets are, i.e. what the defender tries to protect, what the attacker wants and what are the stepping stones. Stepping stones are unwanted but necessary intermediate goals that the attacker has to achieve to reach his actual goals. Also, concentrating on the assets does not directly help in identifying threats against the assets. Furthermore, every asset might not be relevant from a security perspective. Similarly, focusing on attackers means making assumptions of the attacker, such as skills, knowledge of the system and disposable financial assets. An incorrect assumption here can render a threat model useless, as it is based on false premises. Shostack concludes [12, p. 43] that a software centric approach focuses on what developers of the system already know, i.e. their own software.

There are contradicting thoughts to those of Shostack. The most prominent one is that no system is ever completely secure [21, p. 4]. Therefore it is only a matter of attacker skill, resources and time in conjunction with the actual system security which defines if a system is secure or not. Hence, without understanding who the attackers are, there is no way to know when a good enough level of security has been reached. Conversely, time and resources might be wasted on mitigating threats that the attacker couldn't afford or accomplish. While these ideas are perfectly valid, they are based on assumptions that are not ideal for the railway industry. A security expert is required to properly understand attacker capabilities and motivations. It takes time to acquire security know-how and some companies might not have resources to dedicate to a full-time security expert. On the contrary, expertise about the own system is readily available and system correctness is by default a desired property of system development. Therefore, it makes a lot of sense to start evolving the threat modeling process with the software as a basis.

3 Research material and methods

This chapter presents research related to well known threat modeling methods. The next section 3.1 will cover a complete framework for threat modeling. This framework is based on the software centric approach to threat modeling that was discussed in section 2.5. The subsequent sections, 3.2 and 3.3, describe more specific threat modeling methods in detail. These methods can be used as building blocks within the framework described in section 3.1. Section 3.4 discusses how system modeling relates to threat modeling practices and lastly, conclusions of the methods chosen for the case study are discussed in section 3.5.

3.1 The Four-Step Framework

Shostack [12] presents a holistic software centric four-step framework for approaching threat modeling. The framework is heavily based on Shostack's personal experiences and is thus derived more by informal empirical means. This makes the framework more easily approachable for newcomers to the field of threat modeling and enables it to function as a guide to threat modeling activities. The framework divides threat modeling into four parts or sub-goals, which are described best as the activities related to answering the following questions [12, pp. xxviii-xxix]:

1. *"What are you building?"*
2. *"What can go wrong with it once it is built?"*
3. *"What should you do about those things that can go wrong?"*
4. *"Did you do a decent job of analysis?"*

In a software centric approach to threat modeling it is important to start with the question what are we building? It should be obvious that without a proper and complete understanding of the system being built, all efforts to secure it are destined to fail. Once it is known what the system is supposed to do and how it works, the analysis of what can go wrong may begin. In practice, this means identifying risks and threats and is the most central topic in threat modeling. An important notion is that a software centric approach is focused on what can go wrong instead of how things go wrong. By asking what instead of how, focus is concretely placed on what the system is designed to do and what might happen if its operation diverges from the designed behaviour. A natural subsequent step is then to think about what can actively be done to prohibit threats from being realized. Finally, a review of the work done is intended to reveal any missteps during the process.

In order to answer the question related to step 1, various types of diagrams can be used to help describe the system or software that is being developed. Such diagrams are e.g. Data Flow Diagrams (DFD), Unified Modeling Language (UML), swim lane diagrams and state diagrams. These are all good alternatives and are often already utilized as part of standard software development processes within the industry. These same diagrams are also useful for security analysis and threat

modeling purposes. Different system modeling techniques are discussed in section 3.4. Steps two and three of the framework represent the more traditional view of what directly constitutes threat modeling. This non-exhaustively includes identifying the threats to the system, how attackers might try to realize these threats and how the attackers may be stopped. There are a great variety of specific threat modeling methods that cover this area. These methods, categorized as formal and informal methods, will be discussed in the following sections 3.2 and 3.3 respectively.

This four-step framework also reflects the fact that threat modeling is a process which has a place in all phases of system development. Already when negotiating system specifications, step one is highly relevant and steps two and three modestly relevant. At the start of actual development, steps two and three become more prevalent. Lastly, review phases are already present in engineering processes in general so adding a threat modeling review fits in naturally. It is also good to note that threat modeling is specific to a system and even more precisely, to a certain design of a system. If a system is modified or its operating environment changes the analysis and evaluation process has to be redone or at least revisited.

The four sub-goals discussed above form the complete framework, but this thesis will mainly focus on steps two and three, as these parts are the unknowns to the railway industry. However, especially the first step impacts steps two and three and can not therefore be entirely disregarded. Hence, the threat modeling methods presented in the following sections (3.2 and 3.3) will be matched to the framework presented in this chapter.

3.2 Formal Threat Modeling Methods

Formal methods are techniques based on mathematical principles and provide properties such as provability and repeatability. Formal methods are identified by clearly defined language syntax, logic and semantics and may be used for requirements engineering, specification, development and verification of both software and hardware systems.

There are many formal threat modeling methods, such as attack trees [22] and its derivatives, multiple Directed Acyclic Graphs (DAG) based methods like Boolean logic Driven Markov Processes (BDMP) [23]. A lot of these are either derived or have borrowed a lot of concepts and ideas from the risk modeling scene while themselves varying mainly in aspects such as quantification abilities and being of static or dynamic nature [11].

This section will introduce the most relevant formal threat modeling methods. Attack trees, which is one of the oldest and a quite simple threat modeling method, is described first in section 3.2.1. Subsequently more evolved and dynamic methods such as Petri nets in section 3.2.2 and BDMP in section 3.2.3 are introduced.

3.2.1 Attack Trees

Attack trees are a threat modeling method which started to gain a foothold at the end of the the 1990s. The attack tree method is based on threat trees, which in turn

were developed from fault trees and some would even consider the former two to be the same thing. The method has been presented as part of a full attacker centric framework to system security modeling, covering the entire life-cycle of systems or products [24]. Attack tree is a formal and methodical process for approaching system security [22]. Graphs created by the method are Directed Acyclic Graphs (DAG), which due to the lack of cycles generally tends to make the graphs very clear and readable. According to Kordy, attack trees is one of the most widely used threat modeling methods [11, pp. 31-2]. Most other methods refer to attack trees and often describe how they differ from them.

The principle of attack trees is simple and is explained initially by Salter et. al. [24], but in even more detail by Schneier in 1999 [22]; You depict ways to attack a system in a tree structure. The ultimate goal of a particular attack scenario and thereby of a particular attack tree is the root node. The various ways of reaching that goal are the leaf nodes. The path from the leaf nodes to the root describe the intermediate steps required to reach the attack goal. A threat analysis of a system would typically contain several separate attack trees, thus reflecting that there are many different ways an attacker could affect the system in undesired ways. Due to the inherent hierarchical structure of attack trees, they make great building blocks for a complete system analysis and fit well into the four-step framework already presented in chapter 3.1 [12, p. 87]. Each individual attack tree stores information in a re-usable form and if clearly defined can be used as a building block of a bigger attack tree [22]. If the attack tree describes an attack against some sub-system which is used in many different projects, this analysis can directly be shared between projects. This re-usability is a clear advantage of attack trees over many informal methods (section 3.3).

Figure 6 depicts an imaginary attack tree of a safe as presented by Schneier [22]. The root node, i.e. the goal, is to open the safe. The immediate children of the root node depict the direct ways to open the safe. One could pick the lock, learn the code combination that opens the lock, cut open the safe with tools and force or have the safe installed improperly in order to open it later on. The children of each independent node describe the ways to reach that particular node. Nodes can be AND or OR nodes; OR nodes are alternatives while AND nodes are requirements. In Figure 6, all nodes are OR nodes except for the one explicitly marked as an AND node. This means there are four ways to open the safe, while the requirements for successfully eavesdropping on a person is to both be able to listen to a conversation and getting the target to state the needed information. Additional useful information can be added to an attack tree such as for instance the cost of each objective or the feasibility of an objective. This information has also been added to Figure 6. The information can then be propagated up the hierarchy of the tree, e.g. each parent node gets the cost of the cheapest child. In Figure 6 the cheapest way to open the safe, i.e. cutting it open, is highlighted in red. Further, the only way to open the safe without breaking it is noted with the dashed path.

Figure 6 also illustrates how attack trees store information in a re-usable format. If the ultimate goal is protecting e.g. a public person's privacy, opening this person's safe could be only one way of many to access sensitive information or information

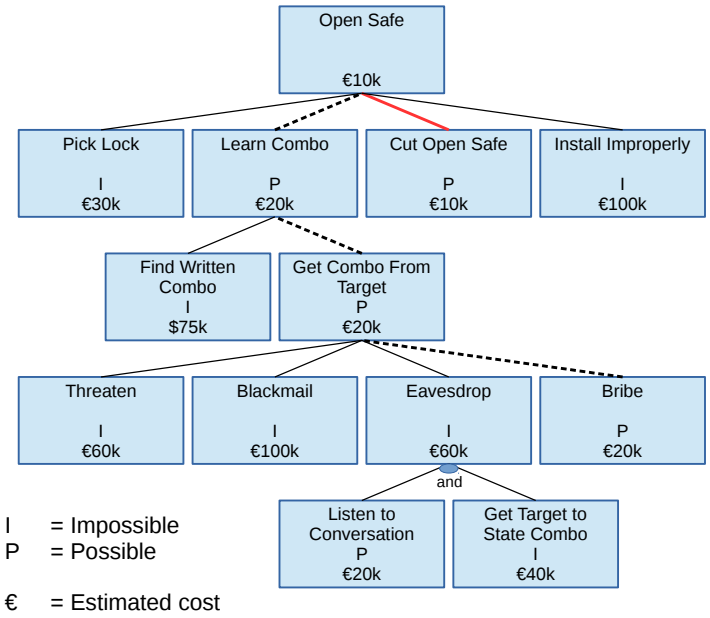

Figure 6: An example of an attack tree [22].

that might enable further attacks on the person. In this case the attack tree of the safe may be placed into other attack trees.

The best aspects of attack trees are that they are simple to construct and the created figures are usually very clear. The difficulties are related to their simplicity, i.e. it might be difficult to break down complicated attacks into simple steps. For instance, attack trees have no way of depicting interdependencies between different sections of a graph. If a certain set of steps are required for many attack paths, these steps will have to be listed multiple times in the graph.

3.2.2 Petri Nets

Petri nets are an old but dynamic modeling tool that offer a very powerful formalism, which is applicable to many fields and purposes [25]. Petri nets have seen many various adaptations and have been used for security modeling already since 1994 [26]. In more recent years, they have been further refined for use in security modeling [27], [28], [29].

Petri nets are directed, weighted, bipartite graphs that consist of two types of nodes, named places and transitions, which in turn are connected by arcs [25, p. 542]. Places can be interpreted as conditions while transitions are events. The bipartite property stems from the fact that every arc is directed from either a place to a transition or a transition to a place. Hence, Petri nets can be interpreted as pre-conditions leading to events, which result in post-conditions. Arcs may be

marked with a weight, where a k-weighted arc corresponds to n parallel arcs. Places may be marked with tokens, which indicate the state of the condition. According to Murata [25], token markings can represent that the condition is true or that a certain amount of resources is available. In short, a Petri net is defined as a 5-tuple $PN = (P, T, F, W, M_0)$ [25, p. 543], where:

- P is a finite set of places
- T is a finite set of transitions
- F is a finite set of arcs
- W is a finite set of weights
- M_0 is the finite set of initial token markings

Figure 7 shows a simple Petri net that contains seven places and six transitions. There are 15 arcs connecting places with transitions. All arcs have a weight of one except for two arcs that have a weight of two. The initial tokens are marked as black dots in Figure 7. Place P1 may transition into place P2 through transition T1. However, place P2 might transition into either place P3 or P4 depending on the transition T2. Lastly P1 may only be marked with a token if both places P3 and P7 enable the transition T6. Murata [25] describes the dynamics of transitions more accurately:

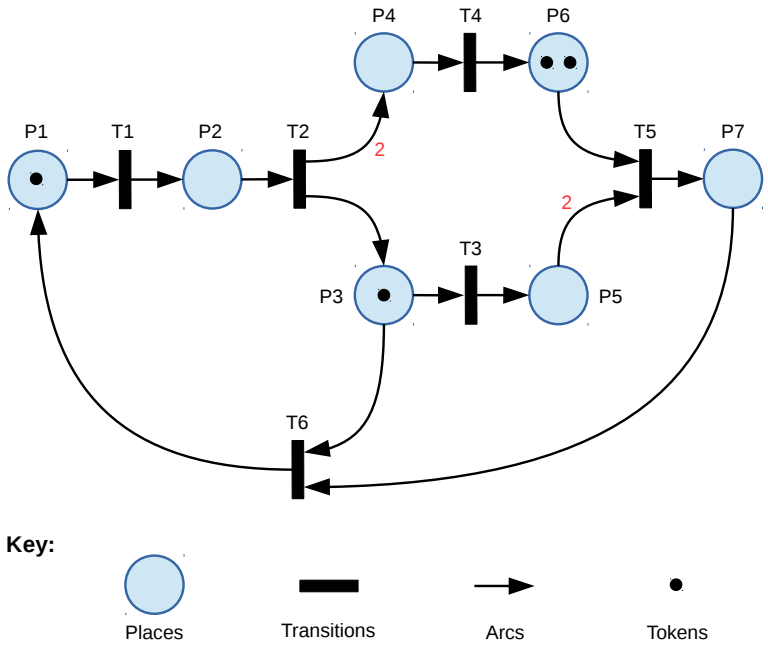

Figure 7: An example of a Petri net [28].

1. If each pre-condition for a transition t contains $w(p,t)$ tokens, where $w(p,t)$ is the weight of the arc from p to t, the transition is regarded as enabled.

2. An enabled transition is not forced to cause a state change. It depends on the corresponding event occurring.

3. An occurring event causes the transition to remove $w(p,t)$ tokens from each pre-condition p of t. It then adds $w(t,p)$ tokens to each post-condition p of t, where $w(t,p)$ is the weight of the arc from t to p.

Petri nets enable the construction of very complicated graphs which means that they can also be used to accurately threat model large and complex systems. Also, the way states transition through the model in the form of token markings allows for modeling concurrency. This is visible in Figure 7 as it is not possible to know which of the following three transitions will occur next: P1 transitions into P2, P3 transitions into P1 or P3 transitions into P5. The events might occur simultaneously, one after the other or not at all. These properties of Petri nets provide great modeling power.

On the other hand, Petri net graphs easily become visually cluttered. Many other modeling techniques simply use an arrow to depict a state transition but Petri nets require two arcs and a transition node. Moreover, because Petri net graphs are so versatile and can be used to describe detailed behaviour they quickly become complex. Lastly, the versatility causes a problem in an explosion in combinatorial calculations when quantifying a large and complex model.

3.2.3 Boolean logic Driven Markov Processes

There has been significant progress in threat modeling since the introduction of attack trees. One of the latest is the Boolean logic Driven Markov Processes (BDMP) method.

BDMP has originally been derived from DAG based safety modeling methods, mainly fault trees from which also attack trees have been derived. Fault trees were initially improved by giving new meaning to the original representation of edges and nodes in the trees as well as introducing some entirely new links called triggers. Bouissou et al. describes the generic BDMP formalism applied to safety and fault analysis in [30]. The new BDMP methodology from the safety field has later been applied to the security field with slight modifications [23].

The BDMP approach has improved both qualitative and quantitative aspects in relation to Attack Trees. In a BDMP graph adapted for security modeling, leaves in a tree structure are interpreted as the basic attack steps, much like in an attack tree. However these basic attack steps are assumed to be triggered Markov Processes, i.e. stochastic processes that satisfy the Markov property [23]. The Markov property states that the probability distribution of future states only depend on the present state, not the history leading up to the present state. The process can be said to be memoryless. They are called triggered Markov Processes, since they may be activated by the so called trigger links. Triggers react to the source signal in their source

node and function as an activating function for sub-structures that they point to in BDMP graphs. Hence, triggers allow the representation of sequences and sub-tree or sub-structure dependencies. A simple BDMP graph containing all the basic BDMP elements is depicted in Figure 8.

In the BDMP in Figure 8 the main node r is the goal of the BDMP. Nodes G1 and G2 represent intermediate goals for an attacker. The basic attack steps, i.e. the Markov processes of the graph are nodes f1, f2, f3 and f4. The only trigger is from node G1 to node G2. Thus, when G1 becomes true, attack steps related to node G2 are said to become active. The true power of the BDMP formalism lies in defining different Markov processes that represent the basic attack steps. In the original formalisation of BDMP, four different kinds of predefined Markov processes were used [30]. However, for the security field Pietre-Cambacedes and Bouissou [23] originally listed only two types of Markov processes. This shows that the authors meant that the used Markov processes should be adapted for the intended use. A third needed Markov process was identified when modeling the infamous Stuxnet worm using BDMP [31].

The three Markov processes used for the Stuxnet analysis are the Attacker Action (AA), Instantaneous Security Event (ISE) and Timed Security Event (TSE) Markov processes. These are depicted in Figure 9, along with a dedicated graphical representation for each process. The AA process in Figure 9 models basic intermediate attack steps, through which an attacker intends to achieve the main goal of the BDMP. The Active mode means that attacks are currently undertaken and the time needed for the attacker to succeed is exponentially distributed with parameter λ. While the attacker has not yet succeeded, the leaf is in a the "On-going" (O) state. When the attacker succeeds in realizing the attack step, the leaf enters the "Success" (S) state. The Idle mode means that no attacks are yet carried out and this mode

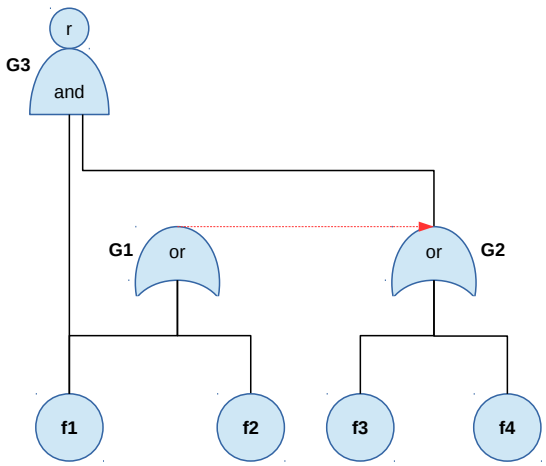

Figure 8: Example of a BDMP graph [23].

only has the "Potential" (P) state. The mode of the attack step shifts from idle to active through an activated trigger link.

The ISE process depicted in Figure 9 models events that happen instantaneously such as for instance detections. Similarly to the AA process, the instantaneous security event leaf is in the "Potential" (P) state when in Idle mode. The Idle mode depicts that the event may not happen at all. Once a trigger activates this process, it will instantaneously enter either the "Realized" (R) state with probability $Pr\{R\} = \gamma$ or the "Not-Realized" state with probability $Pr\{N\} = 1 - \gamma$.

Finally, the TSE process is very similar to the AA process mentioned first. The TSE models an event that the attacker does not control. The occurrence of the event is something the attacker needs but can not affect. The time needed for its realization is exponentially distributed with parameter λ. The "Potential" (P) state describes that the event can not take place. Once the process is activated it enters the "Normal" (N) state, eventually shifting to the "Realized" (R) state. This process differs from the AA process in that it can return to Idle mode and the potential state. However, if the threat occurred in Active mode, the process remains in the realized state.

Using the previously described Markov processes with the new trigger links, very complex attack scenarios can be accurately modeled and depicted in graphical form. This is especially evident when reviewing the results of the attempt to threat model the Stuxnet worm with BDMP by Kriaa, Bouissou and Piètre-Cambacédès [31]. In addition to the qualitative aspect, additional benefits of the methodology can be gained through the excellent quantification properties of BDMP.

In adopting BDMP for the security field, the goal was to preserve the readability of attack trees but improve the static nature of attack trees. By introducing the new links and new semantics, attack trees are augmented with the ability to model time

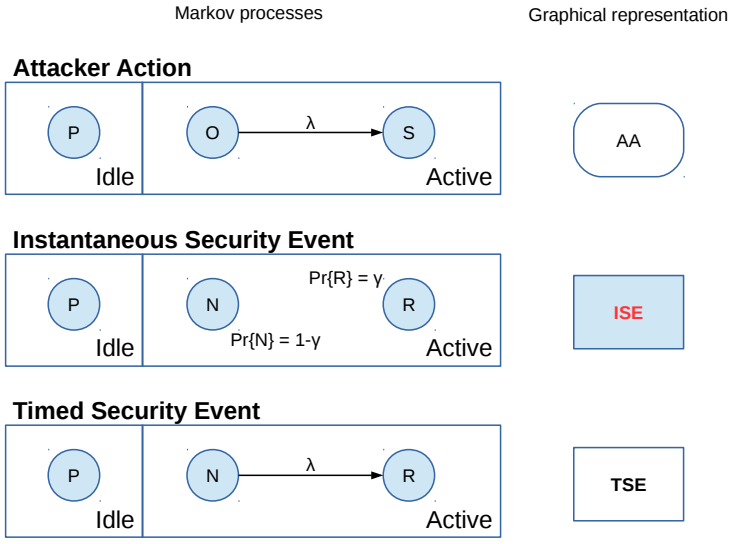

Figure 9: Markov processes for threat modeling using BDMP [23], [31].

dimensions and causalities [23]. In practice, this means that it is acknowledged that certain paths of an attack tree are dependant on other paths. The advantages are seen both in qualitative and quantitative aspects. The BDMP graphs can potentially provide more information visually and provide better accuracy and scale reduction for quantitative calculations.

The biggest drawback of BDMP is that it does not allow as accurate modeling as for instance Petri nets. This is an intentional trade-off by the original authors [23].

3.3 Informal Threat Modeling Methods

While formal methods form the most researched topics in academia, other threat modeling methods have emerged from the commercial sector. While the formal methods can be accurately described as theoretical, the informal methods focus more on practical processes and practices in organizations. They strive to offer concrete step-by-step instructions on how to assist an organization in strengthening security in their systems. In contrast, the formal methods presented in section 3.2 concentrate far more on providing clearly defined and measurable results, i.e. quantification.

Are informal methods needed? Why not simply provide concrete step-by-step instructions on how to use formal methods to achieve the best of both worlds? While a formal method can guarantee a clearly defined formalism, they mostly focus on how an attack is carried out. They do not empower the user to initially find and identify threats. If a system contains dozens of subsystems, all interconnected and communicating with each other, where should one start the security analysis? These sorts of fallacies are exactly what the more modern informal methods are intended to mitigate. The most important parts of threat modeling are finding and identifying threats followed by mitigating them [12, pp. 36-43].

In the following chapters the two most widely used informal threat modeling method types are described. Section 3.3.1 covers the STRIDE approach while the following section 3.3.2 describes different attack libraries. These will at the very least assist a threat modeler in finding threats in the first place, filling a natural step in the four-step framework presented in chapter 3.1.

3.3.1 STRIDE

STRIDE is an acronym that stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege. It is a list of typical ways in which software can be attacked [12, pp. 62-63]:

- Spoofing is pretending to be someone or something that you are not (Authentication)
- Tampering is modifying e.g. data in a file system, in memory or in transit on a network (Integrity)
- Repudiation is saying that you did not do something (Non-repudiation)

- Information disclosure is revealing information to parties it was not intended for (Confidentiality)
- Denial of service is using up for instance network bandwidth needed for a service (Availability)
- Elevation of privilege is providing access to something that was not authorized (Authorisation)

The STRIDE approach is designed to help developers find threats to software. It is a widely adopted threat modeling technique in the software industry that was originally introduced at Microsoft in 1999 [12, p. 61]. STRIDE has been a central part of Microsoft's Software Development Lifecycle framework, which in turn has emerged as a result of observed best practices used to improve software security [32]. It is clear that STRIDE firmly has its roots in empirical approaches, this in contrast to formal methods mentioned in section 3.2. The background of STRIDE is the source of both praise as well as criticism. Proponents mean STRIDE is based on real world experiences and is in touch with the real issues faced every day. Opponents on the other hand stress that methods based purely on empirical evidence are more like quick fix bandages and can not possibly provide complete and proven solutions.

The STRIDE mnemonic functions as a simple guideline for what to look for when trying to identify threats against a system. STRIDE mentions a set of properties, mentioned in the previous list in parenthesis, that should not be violated. The way to use STRIDE is to systematically screen the system being investigated by trying to apply the STRIDE threat types to parts of the system. This can be done as simply as writing down threats as bullet points in a list or by creating some sort of basic form, into which found threats and their details are listed. The most important thing initially is to list everything that comes to mind. The list can be pruned later on. Listing obvious threats, or threats that either are not possible or can be trivially mitigated is not a waste of time. It is vital that even such threats are documented for later review [12, p. 64]. As an example, even trivial mitigations should have requirements that the mitigations are implemented and tested against. Table 1 shows an example of a form that can be used for implementing STRIDE in practice. The table has been filled out with some imaginary threats and their descriptions.

Table 1: Example of a basic STRIDE form [12, pp. 65-74].

Threat	**The attacker steps**	**Notes**
Spoofing a file, such as a known executable on a system	Create a file in the local directory or change the path so a user or script inadvertently runs the malicious binary	Small utility programs are often not "installed" so an Operating System will not notice this attack

Table 1: Example of a basic STRIDE form *(continued)*

Threat	**The attacker steps**	**Notes**
Tampering with a network, for instance reconfiguring the network to reroute data through the attacker's device	This can be used for logging network activity enabling other forms of attacks	
Repudiation of an action that the adversary states did not happen	The adversary claims that it is possible that somebody else did it	If logs are unprotected they might have been directly attacked, i.e. deleted
Information disclosure of information stored in a database, such as for instance the users table in an SQL database	The attacker might simply try SQL injection on every form on web-pages	An often overseen method for information disclosure is social engineering, i.e. utilizing behavioural patterns in humans
Denial of Service attack by filling up storage on a hard drive	An attacker might upload files to the system if no restrictions are imposed	The attack may also be effective even if the hard drive is not completely filled
Elevation of Privilege by abusing data input to a program	The attacker provides malicious input which is not handled properly by the receiving process	There are multiple ways of performing for instance buffer overflow attacks which allow attackers to take control of entire computers

One specific criticism of STRIDE has been that the threat types mentioned in the mnemonic are in many cases redundant. For instance, if an attacker is able to modify a supposedly protected configuration file of some software, is the threat type tampering of the file or elevation of privilege regarding the system where the configuration file resides? Shostack counters this by pointing out that STRIDE is not meant to be used for classification or categorization of threats [12, p. 64]. In other words, it is irrelevant if a threat is categorized as e.g. tampering or elevation of privilege. Instead, the only thing that matters is that the threat has been found in the first place. According to Shostack, precise categorization serves no purpose in itself and will only distract attention away from the task of finding threats, i.e. thinking about what can go wrong with the system.

A positive aspect of STRIDE is that it does not specify every detail of how the method is supposed to be used. It allows some freedom to adapt the method according to for example existing processes. At the same time, the mnemonic constantly reminds its users about the most prominent threat types to look for. The combination of guidance and adaptability should make the method easier to adopt in an environment lacking previous threat modeling experience.

3.3.2 Attack Libraries

Literature review, i.e. relying on available information and expertise of earlier threats and exploits, is another way of trying to find out what can go wrong with systems. There are different ways of compiling a library of attacks or threats. One extreme is more like the STRIDE approach, i.e. generic guidelines to follow, while the other extreme is direct recipes or checklists of what to look after or what to do. Shostack [12, pp. 101-103] mentions that three major decisions have to be made when constructing an attack library:

- Target audience
- Level of detail of library
- Scope of content

There are many different kind of attack libraries, even available online on the internet free of charge. For instance the Open Web Application Security Project (OWASP) [33] has every third year gathered a list of the ten most prominent threat types for web applications. This is a good example of a list with low level of detail and the web community as the specific target audience. The OWASP top-ten list is easy and light-weight to use, but without further work it can only function as a set of guidelines that one should remember to check. MITRE's Common Attack Pattern Enumeration and Classification (CAPEC) is another attack library [34] which has chosen very different trade-offs from that of the OWASP top-ten. The CAPEC library provides a very high level of detail and targets a broad audience. The CAPEC library is hierarchically sorted and can be browsed either based on mechanisms or domains of attack. Mechanisms include attack vectors such as injection, execute code and probabilistic techniques while domains are more abstract attack vector classes such as for example social engineering, communications, software and hardware. Higher level concepts have children which specify details of how higher level goals can be reached. The attack patterns contained in CAPEC are well described and in addition to a simple threat summary contain among other information about:

- Attack execution flow
- Attack prerequisites
- Severity
- Likelihood of exploit
- Methods of attack
- Examples
- Required attacker skills
- Solutions and mitigations

The size of CAPEC can make it slow to use. However the library's impressive size and scope does make it a good go-to reference.

Attack libraries have benefits in that they help threat modeling beginners get started. They are also extremely beneficial as references. Simply being able to check for instance if a threat is feasible or if it has other methods of attack provides concrete assistance for threat modeling. Another example is reviewing attack libraries when trying to decide how to allocate resources most effectively in order to mitigate as many threats as possible. On the other hand they do not in themselves function as a method for threat modeling. How does a person know when enough literature has been reviewed? This could perhaps be possible in the case of CAPEC with a clearly defined problem or restricted system, but should the entire library be parsed every time a more complex system is being analysed?

3.4 System Modeling

As already mentioned in section 3.1, the system being threat modeled has to be properly understood before threats to it can be identified and understood. System models help developers create a common understanding and aid in discussing system traits. Several types of system modeling techniques exist: Data Flow Diagrams (DFD), Control Flow Diagrams, Unified Modeling Language (UML), Swim Lanes and State Diagrams to name a few.

Shostack contends that the system diagram type that best communicates how a system works and therefore allows developers to focus on the intended topics are the ones that should be used [12, pp. 43-44]. One can not state objectively which type of system model or diagram will serve this purpose best, as existing company practices, existing know-how within the company, the type of system being developed, etc. affect the choice of diagram to use.

Even if almost any diagram type may be used, Data Flow Diagrams (DFD) are the most commonly used diagram type for threat modeling, especially for large, network dependent systems [12, p. 44]. As mentioned in section 2.1, one of the main purposes of TCMSs is processing large quantities of data sent over many different networks and buses. For these reasons DFDs appear to be the ideal choice for threat modeling a TCMS. Additionally DFDs also seem to provide a good balance between simplicity and conveyed information. Finally, the choice of system modeling technique is not the core topic of this thesis. Hence, a thorough study of other system modeling diagram types will not be conducted. DFDs is chosen as the system modeling technique to use for the purposes of this work and is presented in this section.

Data Flow Diagrams

A Data Flow Diagram (DFD) describes a system by graphically showing how data flows between components of a system. A DFD graph consists of elements such as data stores and processes, data flows and external parties which are outside of control from the modelers perspective. The usually numbered data stores and processes are linked through data flows and interact with the external entities [12, p. 44].

An example of a simple DFD model is depicted in Figure 10. Originally processes were depicted by circles, but modern DFDs tend to use rounded rectangles. Data flows are represented by arrows connecting elements of the graph. Data stores are shown as two parallel horizontal lines while external entities are depicted as regular rectangles. Modern DFD graphs, particularly when used for threat modeling purposes, often also include the notion of trust boundaries. These are boundaries which group processes or entities that are very closely tied together. Essentially the goal is to show who controls what.

Figure 10 already gives a good picture of how the example system works. The front end opens up access to the database cluster. Data requests can me made, or conversely data can be supplied through these interfaces. The DB process functions as the central element that controls all data stored. Three different data stores are illustrated in the example. Administrators may also access the system through the DB admin interface. Logs may be accessed through the log analysis interface. Note that not all data flows are bi-directional; Logs can only be written by the database process and read by the log analysis process.

Shostack mentions that DFDs have proven to be very useful because problems tend to follow the data, not control flows [12, p. 44]. Especially threats that cross threat boundaries are naturally of great interest. Furthermore, if information flows in both directions between two components, DFDs make it easy to distinguish the separate data flows. Threats are often not symmetric, i.e. one of the two communicating components might be concerned about different security properties than the other.

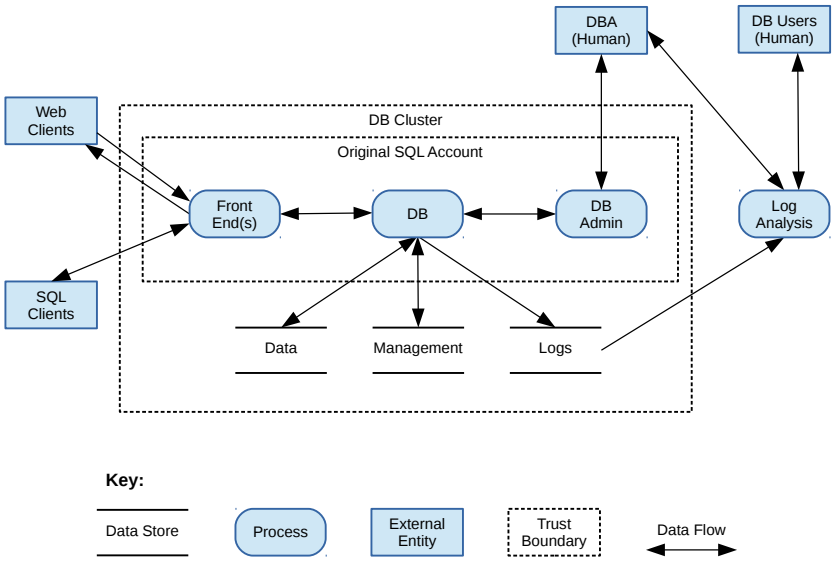

Figure 10: Example of a Data Flow Diagram [12, p. 46].

In Figure 10 this has been emphasized with two separate data flows between the web clients and the front end. For instance, users might disclose their login credentials while transmitting data to the front end but receive incorrect information from the database if somebody is able to tamper with the data returned by the database. DFD graphs can usually be made very clear and easily understandable. Their biggest lack is in modeling detail, with only a handful of main elements available. This is obviously a trade-off made between readability and level of detail.

3.5 Discussion

Different approaches to threat modeling have been presented in chapter 3. Section 3.1 introduced the four-step framework that may be used as a basis for threat modeling practices. Subsequently various formal as well as informal threat modeling methods were described in sections 3.2 and 3.3. Lastly, the topic of system modeling and how it affects threat modeling was discussed in section 3.4. This section will draw conclusions about which tools to use in practice.

The choices presented here have been made to fit the railway industry, particularly related to traits of electronic systems on rolling stock as presented in sections 2.1, 2.2 and 2.3. The following points were the biggest contributors to the choices of practices and methods made here:

1. There is little existing security know-how within the railway industry regarding TCMS

2. TCMS subsystems are heavily interconnected, making it difficult to segment the TCMS and design clear boundaries and Application Programming Interfaces (API) between subsystems

3. Requirements in the ETB standard limit the freedom for designing a network layout with security as first priority

4. A generic ETB based TCMS can not rely on a static system design, as train compositions may change. The system will always face unknown factors to various degrees

All the four points listed above indicate that a software centric approach appears to be best suited for analysing the security properties of a TCMS. Hence, the four-step framework presented in section 3.1 can justifiably be used as a basis for threat modeling TCMSs. The four-step framework focuses on the system being built which helps counter the four issues mentioned before. Especially due to point 1, it makes little sense to begin threat modeling for the first time by either trying to understand what attackers desire of the system or by figuring out what the key assets of the system are security wise. It is easier for developers without any security background to understand the original purpose of the system, i.e. the functionality, and simply figure out how it could break in different ways. This approach functions more as an extension to risk and safety management which is an already familiar topic to many developers.

However, the four-step framework will be adapted with a new step called *"how do things go wrong?"*. This modification is also done mainly due to point one above. This new step will make the threat modeling process more clear for non security experts. The steps of the new five-step framework then are:

1. What is being built?
2. What can go wrong with it once it is built?
3. How do things go wrong?
4. What can be done about the things that can go wrong?
5. Was the analysis properly executed?

This new five-step framework makes a clear separation between finding threats and understanding the threats. These tasks require very different sets of skills. The types of threats that are relevant might be heavily reliant on for instance operator specific customs or usage habits. This is the domain of practically everybody involved in a project, such as project managers, project engineers, system designers or even in extreme cases sales people. Understanding the technical details of threats on the other hand means digging to the depths of the mechanics of particular attacks. In the case of highly sophisticated technical attacks, only experts in their field are likely to possess the skills to evaluate the potential for such attacks. Additionally, it may be difficult to comprehend the breadth of possible attack vectors, something which can be seen in that even very secure systems have been broken into by simple means of social engineering. Generally speaking step two more clearly requires brainstorming like activities where as much ground as possible is covered and where exchange of thoughts is extremely important. Step three requires more attention to detail and specific know-how of security threats. It makes sense to split the original step two of the four-step framework into two more clearly defined steps as outlined here.

System modeling will be done using Data Flow Diagrams (DFD) as shown in section 3.4. The restrictions imposed by the ETB standard as mentioned in point 3 above, means that it is crucial to understand how data flows through the system. The ETB allows almost any device to connect to any other device and provides services for sending so called process data as broadcasts in the system. It is absolutely vital to clearly depict which components are expected to communicate with each other. DFDs should provide a clear picture of what is being built.

Step 2 of the five-step framework will be carried out using STRIDE. As mentioned in 3.3.1, STRIDE is extremely well founded as a tool when the participants lack previous threat modeling experience. Moreover, it is the only tool that provides a structured way for finding and identifying threats in the first place. TCMS systems are complex, even more so when used in dynamic configurations as indicated by point four previously. This means that threat identification in relation to complete system coverage will be of importance and STRIDE will help resolve these issues.

Finally, assuming the STRIDE analysis reveals complex attack vectors on the system, these will be analysed in detail using Boolean logic Driven Markov Process

(BDMP) diagrams. This will empower the analysis process with the needed formal methods of analysing complex threats in an on-demand manner. BDMP is preferred over attack trees and Petri nets because of well suited trade-offs between modeling power and simplicity in its design, as mentioned in section 3.2.3. Although the quantification properties of the BDMP formalism have been proven useful, they depend heavily on accurately determining the Markov Processes parameters. However, parameter estimation is not examined as quantification of possible output variables of the BDMP formalism is out of scope of the thesis. The primary goal of the created BDMP graphs will be to provide a clear picture of attacks and a deep understanding of different attack vectors.

4 Case Study

The chosen framework and threat modeling methods presented in section 3.5 were tested in practice by threat modeling a TCMS system of a TCMS producing company. The company will be referred to as Company A, in short ComA. However, there are no existing ETB based TCMS systems to be used for reference since the ETB standard is so young. Therefore, an existing Ethernet but non-ETB based TCMS design was used for threat modeling. The TCN part of this existing system was replaced in the design of the TCMS to utilize the ETB structure and services.

While further discussing the matter with ComA, it was deemed unnecessary to perform an analysis of a complete TCMS system. The main reason this decision was made is that a TCMS is very large to analyse as one single system and would include a lot of repetitive work. DFD diagrams of the system would be huge and threat lists extremely long. Due to the size of TCMSs, splitting a TCMS analysis into parts by threat modeling sub-systems or functionalities separately is inevitable. Therefore, the threat modeling evaluation could be done for only one sub-system, which was specifically chosen to best represent properties of a TCMS. These properties were partly discussed in sections 2.1, 2.2 and 2.3. Threat modeling one sub-system was determined to be enough to arrive at conclusions about the appropriateness of the threat modeling methods in relation to TCMSs.

Once a sub-system was selected, a workshop was held at ComA where threat identification related to the chosen sub-system was done according to STRIDE. Results from the STRIDE workshop were reviewed and a few selected threats were modeled with BDMP.

In the next section (Section 4.1), the goals and targets of the case study exercise will be determined. The rest of this chapter is structured according to the five-step framework presented in section 3.5. First, the adapted TCMS design that will be threat modeled will be presented in section 4.2, along with needed overview illustrations of the system. Next, the sub-system to be threat modeled is chosen and DFD diagrams as well as data specification lists related to the operation of this particular sub-system are presented. This will describe what is being built and corresponds to step one in the five-step threat modeling framework. Following the system specification, the STRIDE analysis is applied on the chosen subsystem. Threats are identified and listed in section 4.3. Lastly, a more thorough threat analysis will be done on chosen threats that are identified during the STRIDE phase as needing further review. This deep analysis is presented in section 4.4 and will be performed by constructing BDMP graphs of the chosen attacks.

The two last steps of the five-step framework, threat mitigation and review are in general left undone. Mitigation is briefly noted in the STRIDE analysis as the ranking of threats includes identifying trivial mitigations as well as threats in need of further review. Reviewing the threat modeling process, i.e. the last step in the five-step framework, is also touched upon when analysing the results of this work in general in section 5. However, how to perform a systematic analysis process of the threat modeling practices used is not covered.

4.1 Goals of the Case Study

Before any threat modeling practices may commence, it is necessary to define what is desired of this case study. Reviewing the results is a challenge since no previous threat modeling practices exist in ComA. There is also no existing threat modeling expertise in the company. This means that there is nothing to compare results against and no existing expectations about what should be achieved. However, some natural expectations do in fact arise exactly from the premise of no existing experience in threat modeling:

1. The case study exercise should improve understanding about TCMS security.
2. For the threat modeling process to be functional, it must have the following traits:
   – Systematic
   – Sufficient coverage
   – Finite scope, i.e. a clearly defined end
3. It is expected that new threats to TCMSs should be found.
4. It is expected that new attacks on TCMSs should be found.

These expectations are in fact quite obvious. Since there is no existing process in place and no threat modeling has previously been done, new threats and attacks should be found. A lack of new threat findings would reveal either that threat modeling is not suited for TCMSs or that the methods used are not appropriate for the purpose. Further, the whole point of defining threat modeling practices is to provide a repeatable and systematic process. Simple brainstorming is also an approach to threat modeling, but how would you know when you are finished? The STRIDE and BDMP methods should provide a clear frame for the activities and give an indication of when the work is done.

How well these goals are met will be determined by reviewing the results of the STRIDE workshop and ultimately by the resulting BDMP graphs. Additionally, participants of the workshop will answer to a questionnaire so that their view of the workshop's usefulness can be accounted for.

4.2 Train Control and Management System

The TCMS design used to test threat modeling practices was specified with assistance from ComA. This section will present the structure of the TCMS and briefly explain the purpose of the components. Threat modeling decisions, such as the sub-system to be threat modeled and its related DFD graphs, are presented following the TCMS presentation.

4.2.1 System Overview

A general overview of the TCMS system to be threat modeled is depicted in Appendix A in Figure A1. Each end of the train is occupied by a motorized power head car

which control the train. In between the power heads are the trailer cars that accept passengers. The trailer cars only contain functionality related directly to servicing passengers. The rear end of the power head also accepts a small amount of passengers. This is why all the systems present in the trailer cars are also present in the power heads.

Several of the systems in the TCMS are redundant systems, which means that there are two units performing the exact same task. According to experiences at ComA, TCMSs with redundancy capabilities have been a clear increasing trend in the industry. The redundant systems with the same responsibility monitor each other and synchronize their state with each other. The idea is that one assumes an active role while the other one remains in a passive or standby mode. If the passive unit notices that the active unit has failed, it assumes the active role while forcing the previous active unit into standby mode.

A lot of the data shared between the sub-systems is shared as process data instead of over direct point-to-point connections. This is an important factor, since process data uses the connectionless User Datagram Protocol (UDP) instead of the Transmission Control Protocol (TCP). The reason for using process data is that the communication network based on ETB provides the process data service (See section 2.3) as an easy mechanism for sub-systems to deliver cyclic status information updates. In other words, process data is the intended delivery channel through which real-time data needed regularly by various sub-systems can be delivered. All sub-systems source at least one process data packet containing the sub-system's status information.

4.2.2 System Component Descriptions

This section will in more detail explain the purpose, functionalities and responsibilities of the various subsystems of the TCMS depicted in Appendix A.

Brake Control Unit - BCU

Each vehicle has its own Brake Control Units (BCUs). As the name suggests, they control and monitor the local brakes on the vehicle. BCUs in the train are communicating with each other and synchronize their controls in order to avoid skidding or sliding.

Driver and Guard Display Units - DDU and GDU

The Driver Display Unit (DDU) and Guard Display Units (GDU) provide the crew with the main human machine interface for controlling and monitoring the operation of the train. The displays receive real-time status information about the train from a PLC application. The information shown on the displays such as speed, door status, brake pipe pressures, distance to next station and more general sub-system information enables the crew to control the train safely. The driver can use the DDU to issue commands such as opening and closing doors, switching on lighting,

air-conditioning etc. The most vital train controls such as traction and brakes are not controlled via the DDU but have dedicated physical inputs on the dashboard.

Door Control Unit - DCU

There are several Door Control Units (DCU) in the train, each controlling a set of local doors. The DCU relays door commands sent over the ETB network by PLC applications to the actual door actuators.

Ethernet Routing Unit - ERU

Each vehicle in the train needs at least one ETBN. The Ethernet Routing Unit (ERU) reroutes packages between the ETB and the ECN networks as well as functions as the TRDP service provider for the local vehicle. Upon train power up, the ERU takes part in the ETB inauguration procedure in order to set up the ETB network for communication. Once the inauguration is done, sub-systems may communicate with each other using the ETBs process or message data services. An illustration of high level steps involved in successfully sending process data from one system to another is available in Figure A2. This procedure involves the sending ED resolving the DNS name of the multicast address before the process data can be sent. The receiver ED on the other hand has to register to this same multicast address so that the local ETBN knows to forward the process data to the ED.

Ethernet Switching Unit - ESU

The Ethernet Switching Unit (ESU) is a simple switch that relays Ethernet packets sent and received on its ports.

Global Positioning System - GPS

The Global Positioning System (GPS) receivers track GPS satellites and provides a position lock and time information. This information is broadcast as part of the GPS modules process data for use by other sub-systems.

Heating, Ventilation and Air Conditioning - HVAC

The Heating, Ventilation and Air Conditioning unit is the system responsible for climate control in all cars. It controls all related systems in the vehicle and assures that desired temperatures are kept in the vehicles. It also checks that for instance when outside temperatures are high, air conditioning shuts down temporarily when doors are opened in order to preserve energy.

Input Output module - IO

Each vehicle will have a need to both read and set various digital and analog signals. The Input Output modules (IO) provide an interface to the ETB for doing just that.

Passenger Information System - PIS

The Passenger Information System is split into two sub-systems: The managers and the clients. The Passenger Information System Manager (PISM) is responsible for tracking the train's movement along a designated route. In practice this means it handles the train's timetables and route records which may be remotely updated using the TTW. To check that the train is moving according to schedule, PISM also listens to location and time data sent by the TPM. Up to date information about the train and its progress is then delivered to the The Passenger Information System Clients (PISC) which convey the information to the passengers. The PISCs encompass all equipment in the passenger areas of the coaches that provide train and route information to the passengers. These are the audio announcement systems, the internal passenger screens as well as the external displays. The crew Wi-Fi links are also considered part of the PISC systems because their primary purpose is to provide the train guard with train status information.

Passenger Entertainment System - PES

Passengers on the train have a possibility to connect to an on-board Wi-Fi network. This enables passengers to view train information like for instance speed and upcoming stations on their own mobile devices. The Passenger Entertainment System (PES) system functions as a content delivery server which services requests sent by the client devices of the passengers.

Power Management Unit - PMU

The power head cars contain the pantographs, electric converters and spare batteries that supply electric power to all the other systems on the train. These components are controlled by the Power Management Unit (PMU), which ensures correct specified output voltages and currents on all existing lines. For instance, if line voltage drops from the overhead wires through the pantographs, battery power is enabled. When line voltage is available the charging of the batteries is monitored by the PMU.

Programmable Logic Controller - PLC

There are several Programmable Logic Controllers (PLC) on the train. Depending on their position in the train these have different tasks and responsibilities. The PLC modules in all the trailer cars and one of the power head PLC modules are used to provide a more standard interface for digital and analog input and output signals. Digital and analog signals are usually part of simple functionalities such as lighting systems and various other less important train status sensors. The PLC provides an easy way to control these functions in the train.

The PLC modules in the motorized power heads that are directly connected to the DDU and GDU have more responsibilities. These act as the main status information repository for the entire train. This means that these particular PLC applications listen to the process data packets of all sub-systems in the train. Therefore, the PLC

is able to do things like log events or data for maintenance purposes and show alerts to the driver by commanding the DDU. These two particular PLC applications perform tasks that require aggregate information from several subsystems across the entire train. For instance, opening the passenger side doors requires the verification that the train is at a station, the train has stopped and that the platform is situated on the correct side. Even if the driver commands the doors to be opened from the DDU, the PLC will verify that this is indeed a safe operation and will ultimately send the commands to the correct DCUs. The two PLCs in the power heads can be considered to run the most central logic to the train's operation.

Station Identification Beacon - SIB

The Station Identification Beacon is a system which scans for track-side station identifier tags. Through this mechanism, the train gets a clear indicator that it is approaching a specific station. Tags are usually located in both ends of a platform on a station, hence the system is also be able to tell when the train leaves the station.

Traction Control Unit - TCU

The electric engines that provide traction to the train are controlled by the Traction Control Unit (TCU). The TCU controls the tractive effort produced by the engines and monitors the operation of the engines.

Time and Position Manager - TPM

Time and position information are naturally linked in the case study TCMS since both are most accurately provided by the GPS receivers. The Time and Position Manager (TPM) is the software providing a coherent time and position source for the rest of the trains systems. The GPS provides atomic time and very precise location information. When GPS signals are not available, time is temporarily tracked using internal computer clocks while position information relies on odometer information received by the Traction Controller Unit (TCU). There are two GPS receivers, one in either end of the train. Both TPM systems can listen to either GPS receiver unit. The Time and Position Manager uses the Network Time Protocol (NTP) to synchronize clocks in the train.

Train To Wayside - TTW

Modern trains often utilize mobile networks to provide a 24/7 connection from the train to some control and command monitoring central. This data link is called the Train To Wayside (TTW) data link and functions as a gateway and firewall between the trains internal ETB and the outside world. The TTW link only allows predefined data to pass in either direction.

By using the TTW, new databases containing route or timetable information can be uploaded remotely to the train, mainly for the PISM to use. This allows for easy updating of an entire fleet when as an example switching to a new set of timetables

across the entire fleet. Additionally train diagnostics data can be downloaded from the train through the TTW to be used for e.g. estimating upcoming maintenance needs.

4.2.3 Case Study System Conclusions

The case study system presented in section 4.2 shows how complex and interconnected a modern TCMS is. Especially the large number of sub-system interconnections makes describing the system using simple DFDs hard. The TCMS is too big to include all components and all data traffic in one diagram. However, dividing the system into many smaller sections based on for example physical location, so that the aggregate of the sections combined with an overview graph form the entire TCMS system is not trivial either. There are simply too many data flows coming into and flowing out of the smaller diagrams to accurately depict the operation of the system.

To counter the issues mentioned above, the description of the TCMS using DFD will be split into many graphs in another way. To best describe the functionalities of the system, each DFD will focus on the data flow regarding one single component at a time. These system individual DFDs will assure that the operation of one component at a time is understood completely. The negative side of this approach that it introduces redundancy, the same data flows may be described in many different DFDs and may exhibit different types of threats related to different components. This is no problem for using the STRIDE mneumonic, as the purpose of STRIDE is to help identify new threats and not to necessarily categorize them [12, p. 12]. Applying the STRIDE method to each DFD separately will ensure that all possible threats can at least in theory be discovered and identified.

As mentioned, this threat modeling case study will focus on a single sub-system in the TCMS presented above. The purpose is to examine how well threat modeling practices fit into the railway industry and in a general sense to analysing the security of TCMS. That goal can be reached by performing the analysis on some typical TCMS sub-system.

By reviewing the sub-system descriptions from section 4.2.2, it is clear that the most heavily interconnected systems in this particular TCMS are the PLC applications and PISM. Out of these two candidates, the PISM is better suited for an initial threat modeling analysis. The PLC in most TCMSs function more like a general purpose, do it all system, that handles all tasks not assigned to a dedicated sub-system. Such a system is firstly hard to define and therefore hard to threat model. Secondly, a general purpose logic unit is subject to be used for widely different purposes in different systems. This means time would unnecessarily be wasted trying to describe and explain project specific details that have little importance for a generic TCMS. The PISM on the other hand has a clear, although somewhat broad purpose. The PISM represents typical aspects of a TCMS sub-system by having many data interconnections, performing complex tasks as well as also controlling systems which are vital for safety. Due to these considerations, the PISM was chosen as the sub-system to threat model for the case study.

The PISM is described in detail in Appendices B and C. First, the PISM's DFDs

which precisely depict the sub-systems with which the PISM is communicating with are depicted in Figures B1 and B2. Also in Appendix B, the DFDs are followed by a more thorough description of the PISM system operation to give a proper idea of the system's tasks.

There are two different DFDs. The first one contains only data flows directly related to PISM, i.e. either the PISM reads or writes the data present in the DFD. The second DFD illustrates the bigger picture with additional data flows that take place between other sub-systems, that are themselves also communicating with the PISM. The idea is to discover potential loops in information flow, i.e. situations where the PISM might supply data to one sub-system, which in turn uses that data to provide some other information to another sub-system, which then again based on the received data provides the PISM with some information. These situations might for instance cause the PISM to trigger undesired behaviour in itself. Such design flaws should also be revealed when threat modeling the system. The goal of the first DFD is to provide a clean graph with only the most relevant information, while the second graph provides full information which can be used for reference if needed. The data flows in the DFDs are tagged for easier identification. Data flows read by the PISM begin with the "I-" tag while data flows sourced by the PISM are marked with the "O-" beginning. Data flows between other sub-systems are tagged with the "E-" beginning for "External"-data flows.

Further describing the details of the PISM, the tables in Appendix C list the details of the tagged data flows from the DFDs in Appendix B. Note that several tagged data flows might be different instances of the same data packet type. As an example, there are several sub-systems that read the process data of the PISM, i.e. they read the same data type. However, due to latencies in the network, they might not read the same data content at the same time and in any case they might not be interested in the same data parts of the process data. This is an important distinction that might reveal additional threat, for instance if strict synchronization is required between sub-systems for some functionalities of the train. An obvious example would be the train applying brakes in the front part of the train while applying full throttle at the back.

In the following section, 4.3, the actual threat modeling workshop that is based on the information in Appendices A, B and C is described.

4.3 Threat Identification

The threat identification step was performed using the STRIDE method on the DFD models presented in section 4.2.1 and related Appendix B. There were 7 participants in the brainstorming session; 3 software developers which represented the technical expertise; 2 project engineers with higher level system understanding; 2 managers with technical background. No participant had any prior experience in threat modeling.

The entire session comprised of two phases. First, a brainstorming session was held where each data flow in the Figure B1 were discussed separately. These initial results were then further analysed in the second part, with the goal of doing some

initial severity analysis of the findings as well as potentially revealing new related threats.

Both phases were structured according to the STRIDE model. Brainstorming was done with the following restrictions; Discussion was limited to one source of problems, i.e. data flow related to PISM, at a time. All the different attack modes of the STRIDE mnemonic were addressed in order. All data flows directly affecting the PISM needed to be processed. If someone deviated from this rule, the threat was quickly written down to be handled later and discussion was immediately returned to the original topic. In the second phase, all found threats from the first phase were revisited and discussed. Core causes were identified, severity and effects on related sub-systems were discussed and obvious mitigations if evident were noted.

The template for the STRIDE analysis is presented in Appendix D. For each threat the data flow causing the threat was marked and a unique ID was given to ease discussion and documentation. The four left-most columns contain the resulting information from the first phase of the threat identification step. In the second phase, the threat description and Affected Component columns were briefly revisited to verify their accuracy. Further, the three right-most columns: Severity, Mitigation and Mitigation Notes were then filled in to give a more complete picture of the overall threat to the system. The results of the STRIDE analysis are listed in the tables of Appendix E and are reviewed in Section 5.2.

4.4 Threat Analysis

Threats that can not be mitigated with some simple method require further analysis. In this section, a threat from the identification phase is examined more carefully and potential attack scenarios or vectors are modeled using formal methods.

The original plan was to construct a BDMP on a per STRIDE item basis. This is in fact non-sensical, since each STRIDE item is a threat that functions more as a source of issues. Hence, the threats identified as part of the STRIDE analysis will be situated on the very bottom of a BDMP graph. The BDMP formalism is based on the notion of how an attacker will reach a certain goal or objective, i.e. the analysis has to begin with setting the goal node. However, the goal node can be derived from the STRIDE threat-items, i.e. choose a threat and evaluate what the end-goals of the attacker might be. This approach still fulfils the original goal of providing deeper understanding of threats found using STRIDE. Furthermore, as can be seen from the STRIDE results (Appendix E), this is a natural part of the STRIDE threat identification. Several threats already contain proposed attack goals in the threat description.

STRIDE and BDMP complement each other when BDMP is used as previously described. STRIDE will start with a bottom-up approach while the BDMP will begin with a top-down approach. It is noteworthy that BDMP is still very much focusing on the details of a threat, i.e. most of the effort is spent understanding the low level details of each identified attack vector.

The Elevation of Privilege threat 6002 in Table E6, originating from data flow I-2, raised some interest during the STRIDE workshop. The threat goal of simply

showing incorrect information to for instance passengers was initially thought of as a non-issue. Some first reactions to the threat was questioning what the attacker would gain from such an attack. However, such an attack could be intended to simply cause confusion or it could function as a visible verification step that the attacker has breached the system. This discussion meant that the threat was chosen for further threat modeling using the BDMP formalism. The goal of the attacker was set to show incorrect data on the internal passenger information displays inside the train.

The two resulting BDMP graphs are depicted in Appendix F. The first, Figure F1, is the main graph containing the main goal while Figure F2 illustrates the attack vectors for reaching stepping-stone objectives that function as intermediate targets towards reaching the main goal of Figure F1.

The majority of the triggered Markov processes that form the basic attack steps are attacker action nodes, denoted AA in the figures. There are two Instantaneous Security Event (ISE) nodes which in turn are triggered by the only Timed Security Event (TSE) node. The TSE node in Figure F1 represents ETB inauguration, i.e. the initial network configuration phase. This TSE node is active all the time, which means inauguration may occur. The node then switches from not-realized to the realized state as the ETB actually inaugurates. The TSE node in turn triggers the two Instant Security Events (ISE) that are the stepping-stones for a successful man-in-the-middle attack. This construction states that each time the ETB inaugurates, there is a certain probability that the man-in-the-middle attack step instantly succeeds. The two ISE attack steps can only succeed during inauguration because the operational Train Topology Database (TTDB) is locked by all ETBN nodes after ETB inauguration. Data packets dropped onto the network is assumed not to be routed if the traffic originates from unknown devices, i.e. devices not listed in the TTDB.

It is noteworthy that a lot of the attacks are triggered by the pre-requisite of the attacker gaining access to the ETB network. Steps to achieve this goal, i.e. how to find an entry point into the ETB network and the TCMS is out of scope of this work and is therefore not studied.

5 Results

This chapter will review the results achieved in the case study in chapter 4. The system modeling, threat identification and threat analysis phases of the used five-step framework will each be covered in their corresponding sections: 5.1, 5.2 and 5.3. The results obtained are then presented in section 5.4 as a threat modeling process for TCMSs.

5.1 System Modeling Results

The creation of a proper system model and associated system descriptions proved to be a difficult task. This is mostly due to the sheer amount of data flows that need to be represented. Another related issue that caused problems is the real-time, process control nature of the system. The PISM sub-system uses several constantly updated data inputs from different sources to produce control outputs for other sub-systems in real-time. There is little notion of sessions or other similar concepts frequent in IT. In IT systems it is easier to follow cause and effect. When looking at a TCMS, it is unclear if the system will react differently if some particular process data is invalid for a minute instead of a single second. These differences are not obvious from simple DFDs but both of these issues are very typical traits of TCMSs and make the creation of DFDs demanding.

As a result of the aforementioned reasons, the system model presented in Figure B2 was found to be insufficient. The PISM SM block in the PISM DFD is essentially a big black box which merges all input and output relations. Cause and effect relations are not visible, but need to be traced from the data flow description tables listed in Appendix C. The PISM SM box functionality should be clarified in more detail in a separate diagram. Alternatively the input-output relations could potentially be more visible if listed in a matrix format.

The problems faced made it clear that the DFD designs should be revisited after an initial STRIDE session has taken place. Different sub-systems and components are likely to be best represented by different types of DFDs. For instance, an alternative approach for the PISM in the case study could have been to draw a separate DFD for each functionality of the PISM. One figure would depict how data flows in order to produce output to passenger screens, while another DFD would show how data flows for producing status and route information for the driver. The drawback of this approach is that understanding of the complete picture is obfuscated while details of the separate functions are clarified.

In spite of the issues mentioned, DFDs were deemed an appropriate approach to describing the system functionality from a security perspective. The DFD models provided a good platform for discussion, something that Shostack finds of utmost importance [12, p. 44]. In particular, cross relations between subsystems were revealed by following the data flows and looking up details in the data flow description tables. Comments during the STRIDE sessions that similar diagrams should be used more often, even outside threat modeling practices serves as a testimony of their usefulness.

5.2 Threat Identification Results

The STRIDE workshop formed a good framework for discussions around security and subsequently a significant number of threats were identified. The approach to concentrate discussion directly on data flows had both pros and cons. It provided desired focus to the discussion which lead to discoveries of concrete threats that could be described and listed. The drawback was that it also focused conversation to the application layer, i.e. usually the PISM functionality was the target of the threat. Hence, few threats and vulnerabilities related to the ETB protocol stack and the PISM's underlying system were found. A separate STRIDE analysis focused on the ETB network and its services as well as the operating system the TCMS applications run on might be appropriate.

At least two clear patterns are visible from the threats listed in the STRIDE results in Appendix E. Firstly, it became obvious that TCMSs are currently treated as closed systems where it is assumed that adversaries have no access to the system. Internal data flows are not protected against intentional malicious behaviour in almost any way. This is most evident with respect to process data, which is sent using the connectionless UDP. The result is that process data appears highly susceptible to spoofing attacks and accordingly a lot of related threats were identified. These threats could be mitigated using sender authentication and data integrity verification techniques. The other generic threat is that the ETB network is vulnerable to DoS attacks. An attacker that is able to push high amounts of network traffic onto the backbone can cause serious problems. The ETB does provide means for assuring data transfer for critical data, but there are a lot of data flows that need high priority assurances which in turn increases the attack surface for the attacker to successfully conduct a DoS attack.

The identified threats directly related to the PISM application functionality did result in new insights of how to for instance circumvent data unavailability and how to detect false or erroneous data inputs. This is naturally another good indication, although these threats are of less interest for this analysis due to the TCMS design not perfectly representing a real existing system. However, there was one interesting finding directly related to the PISM application; The crew members, i.e. the driver or the guard, might either willingly or by mistake abuse the system. While this insight is of importance already by itself, it also affects the repudiation property mentioned by STRIDE. ComA might as TCMS supplier be caught in a dispute where an operator blames the TCMS for malfunctioning while the fault in reality originated from user error. The TCMS design might have to be adapted to ensure the ability to prove compliance to specifications. This flaw in the TCMS design used for the case study was correctly noticed. The flaw is clearly noticeable in Figure B1 as the absence of any data flows related to logging functionality as well as data flows for retrieving logs from the train.

All seven STRIDE workshop participants filled in a questionnaire and answered questions related to the goals set for the case study. The results are listed in Appendix G. The numbers in the columns in Table G1 indicate how many participants gave a particular answer. The average values are calculated at the end of each row. The

results are generally positive with all of the participants finding the workshop very useful and most stating that the session had broadened their understanding of TCMS security. Two participants explicitly commented that the workshop had been very beneficial and that these sort of processes should be standard practice.

Even though comments from the participants were positive, some generic problems were noted with the analysis. One issue was that as the STRIDE workshop progressed from one data flow to the next, patterns arose in the type of threats that were immediately recognized. It appeared as if the participants became blinded by the obvious threats. These so to say blinding threats were mostly related to spoofing of data due to unauthenticated and unencrypted connections. Ironically these threat findings were one of the most clear and beneficial results of the workshop. Related to this lack of creativity, a threat that was not mentioned was infections of ComA's own sub-system devices through infecting PCs used to either configure or reinstall firmware or applications. For instance in the case of the infamous Stuxnet worm, this was precisely the initial attack vector [31]. This threat could of course be regarded as an entry point to the ETB network and therefore out of scope. Additionally, the lack of knowledge about security and threat modeling could have resulted in no-one questioning the absence of a method, i.e. data flow, for software updates.

In general, results from the STRIDE workshop are encouraging and a lot of both obvious threats as well as new ones were identified. This was achieved in spite of a lack of security expertise at ComA. There is no doubt that increased experience in the security field would benefit the analysis. Benefits would also be achieved from refined and improved DFDs that function as the basis for the STRIDE analysis. These are issues that would improve with time as more experience is gained. Therefore, a good idea would be to perform the STRIDE analysis in incremental steps. After the initial STRIDE session there is most likely a need to revisit the TCMS design and redraw DFDs according to STRIDE results. After this a second STRIDE session would be held. This process of iteration should be repeated until no threats related to the design of the TCMS are identified.

5.3 Threat Analysis Results

The first thing to notice from the BDMP graphs shown in Appendix F is that all the Markov process attack step types depicted in Figure 9 have been used. This is a good indicator that the BDMP formalism has useful qualitative aspects related to modeling TCMS related threats.

There were also several revelations related to actual threats while constructing the BDMP graphs:

– There is a strong tendency in TCMSs for categories of attacks. All process data flows are exposed to similar kinds of attacks. Process data is most easily attacked due to traits in the ETB TRDP services. The main BDMP graph in Figure F1 lists several process data packets, which when compromised would allow the attacker to reach the specified goal. A sub-graph, such as shown in Figure F2, should be drawn to represent attacks against these data flows, regardless of data content in the data flow.

- The creation of the BDMP graphs revealed other threats related to the original threat that was used as the basis for the graph. This is in hindsight obvious. On the other hand the attack steps depicted by the BDMP sub-graph in Figure F2, had not been mentioned before the BDMP graph was drawn. A direct attack on the passenger screen device through its remote log-on interface had not been thought of. It could be argued that this is out of scope, since the screen is the not ComA's, but the fact remains that an attack vector would have been missed.

- More importantly related to the previous point, the BDMP graph revealed new threats. The insight of the direct attack on the third party passenger screen brought to light yet another omission in the PISM DFD; There is no data flow depicting the Secure Sockets Shell (SSH) server, running on the operating system upon which the PISM is running. This is a data flow that should be included in the DFDs and it is very likely that threats could be identified regarding this interface to the PISM.

Concluding, reviewing threats with formal methods proved useful. Additionally the BDMP formalism appears to strike a good balance between modeling power, graph complexity and graph readability. It was noted that DFDs might have to be revised even after drawing the BDMP graphs. This in turn would no doubt require another STRIDE iteration before BDMPs could be updated.

5.4 The Threat Modeling Process Result

The STRIDE workshop and the threat analysis done by using BDMP formalism produced concrete results. It is therefore possible to recommend a TCMS threat modeling process based on the used methodologies. The resulting five-step process is depicted in Figure 11. Each step in the process graph corresponds to the five-step threat modeling framework described in section 3.5. The main activity of each step is mentioned at the top of the figure. Based on the experiences from the case study, resulting documentation for each step is listed at the bottom. Arrows towards the bottom of the figure indicate documentation causality. The documents produced in the different steps are:

- **Step 1 - System Description:** A description of the system being analysed that explains at least the most central purposes and functionalities of the system

- **Step 1 - DFD Graphs:** The graphs that depict all data flows from and to the system being analysed

- **Step 1 - Data Flow Specifications:** Detailed information of the data written and read by processes related to the system under analysis

- **Step 2 - STRIDE Analysis Results:** Documentation of threats identified during STRIDE sessions

- **Step 2 - Threat Index:** A properly sorted threat index list that can be used as a lookup table to review threat identification results
- **Step 3 - BDMP graphs:** BDMP graphs created with the intention of providing more detailed information about chosen attack vectors
- **Step 3 - Threat Requirement Specification:** Requirements that contain all needed information for implementing mitigations against the threats listed in the Threat Index
- **Step 4 - Threat Mitigation Design Description:** Document containing design decisions of how threats mentioned in the Threat Requirements Specification will be mitigated
- **Step 5 - Threat Modeling Review:** Documentation of how threat mitigations have been evaluated and reviewed

The suggested documentation related to steps four and five is subject to change and is only added here for a sense of completeness. Since the steps were not evaluated as part of the case study described in section 4, the practicalities related to these steps still need to be verified.

The arrows at the very top of Figure 11 are a direct result of the observations from the threat modeling results. There is a high probability that preceding steps

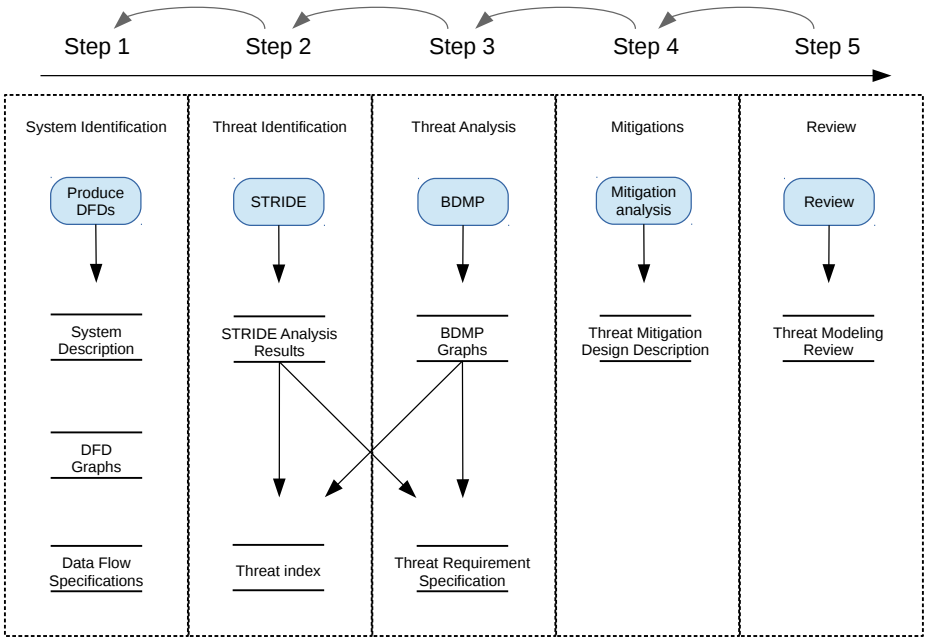

Figure 11: Proposed TCMS Threat Modeling Process

need to be refined due to new revelations of later steps. The threat modeling process should be interpreted as iterative, where indeed previous steps are revisited and documentation updated as needed. Although steps four and five were not tested in practice in this work, it is easy to imagine that this assumption holds true even for these steps. For instance mitigations actually implemented during software development might change the way the system works. As an example, introducing encryption will likely counter data tampering, but as a result, key protection becomes vital for system security which in turn affects information disclosure threats.

An important benefit of the process depicted in Figure 11 is that it resembles existing software development processes. Steps one through three can be regarded as phases of system specification resulting in the threat requirement specification. Step four corresponds to the software development phase, which is initiated by designing implementations of threat mitigations. The threat modeling review done as step five should easily fit in among other software development review processes.

Finally, how well did the new threat modeling process perform with regard to the goals set in section 4.1?

1. Feedback results in Table G1 and the BDMP graphs in Appendix F, clearly indicate that the process increased understanding about TCMS security

2. The new process is very systematic and provides a clear scope for the analysis. Nevertheless, the requirement of sufficient coverage was not properly met, as viable threats types were left undiscovered. The limited scope of the case study certainly affected the outcome to some degree

3. New threats to TCMSs were discovered, so the related requirement was met

4. New attacks and attack vectors against TCMSs were discovered, so the related requirement was met

5.5 Future Work

The resulting threat modeling process of this thesis provides a good basis for how to threat model a TCMS. However, some aspects of threat modeling were left out of the scope of the thesis.

To begin with, the choice of system modeling method was not based on a thorough study of different methods. Instead the decision to use DFDs was based on statements in literature that they are most widely used in threat modeling. Because TCMSs have unique traits that make them hard to model, this assumption should however be tested and suitable system modeling techniques should be evaluated.

Furthermore, additional benefit of the BDMP formalism would be achieved when fully utilizing the quantification possibilities of the method. This thesis did not properly investigate quantification aspects of the different threat modeling methodologies. For instance how accurate are the results of different methodologies and how hard is parameter estimation? Quantification of complex threat graphs would require using dedicated software tools. The usability of these tools is another

subject for investigation. However, the benefit of automated quantification of the BDMP formalism was demonstrated in the analysis of the Stuxnet worm [31]. This makes the topic of quantification attractive for further studies.

Threat mitigation and process review were also out of scope of the study. The goal of threat modeling is to be able to mitigate threats. Hence, without evaluating the two last steps of the process the results remain incomplete. Further effort should be dedicated to threat modeling a system according to the complete threat modeling process.

The case study confirmed assumptions that attacks can be grouped. For instance process data can be attacked in the same way almost regardless of content and an operating system login prompt can be attacked using the same techniques regardless of which applications run on the system. Further study should be conducted to reveal what parts of the TCMS can be threat modeled in a generic fashion. The goal would be to threat model a certain functionality once and then be able to utilize the analysis automatically in every system using that said functionality.

Lastly, the premise of this thesis was that there is no existing know-how about threat modeling in the organization. This premise was part of the decision for adopting the software centric approach to threat modeling. On the other hand, it was surprisingly most clearly noted during the STRIDE workshop that the process would benefit from more threat modeling experience. Once gained by an organization, such experience could be used to review a TCMS from a more holistic point of view with the purpose of focusing efforts where they are most needed. Instead of always threat modeling the entire TCMS, security could be improved by more thoroughly analysing the most vulnerable parts. Therefore, how to identify the most vulnerable parts of a system should be studied further.

6 Conclusions

The goal of this thesis was to determine if commonly used threat modeling practices in IT are suitable for analysing the security of TCMSs based on the ETB. More specifically, the intention was to identify which methodologies are the best candidates for this purpose and how they could be used.

Threat modeling covers a wide array of concepts, processes and practices. In this thesis a software centric approach to threat modeling was chosen. Based on literature review, a holistic five-step framework for threat modeling TCMSs was proposed as a foundation for threat modeling practices. Other methods, such as STRIDE and BDMP, provided the practical implementations of the steps of the framework. STRIDE was used for threat identification purposes while BDMP provided a formal method for deeper analysis needs.

The proposed threat modeling process was tested in a case study of a TCMS producing company. The case study TCMS was based on a real-world Ethernet based TCMS design, which was modified to utilize the ETB network structure. Goals for the case study were set based on the premise that there was no prior threat modeling experience at the case study company.

The proposed TCMS threat modeling process did mostly fulfil the goals set for the case study. The five-step framework proved useful for threat modeling. The STRIDE method enabled discoveries of new threats and provided structure to threat finding. The BDMP formalism was successfully used to more thoroughly analyse threats found using STRIDE. Although results were positive, room for improvement was noticed. The biggest problems arose from insufficient system models and system descriptions. Further study in the area of modeling TCMSs should be conducted to improve understanding of the system being threat modeled. Moreover, even if the used process was designed to counter lack of security expertise at the case study company, the absence of such expertise was noticeable in the results.

Nonetheless, the new proposed threat modeling process does advance efforts to secure TCMSs. For instance a concrete result of the case study was that data transfer within TCMSs need to be protected against spoofing and tampering. Thus, the proposed five-step framework functions as a first step in achieving an adequate level of security on-board trains.

The important take-away is that security is not an absolute. There is a saying in the IT-security sector: "Convenience is the enemy of security". The point is that striving for easy to use solutions for consumers is often at odds with secure solutions. This does not directly translate to the railway industry. The end users, i.e. the passengers, do not have to care about usability and technical solutions of trains. There is however another similar dilemma regarding rolling stock. Trains can be made extremely safe because they operate on the ground, on tracks and therefore in a partly controlled environment. In contrast, car traffic is completely dynamic and airplanes can not simply shutdown and stop mid-air. Trains on the other hand can always eventually be stopped. However, the goal is to provide enough safety and security while still assuring availability and herein lies the dilemma: Will the most secure train ever leave the platform?

References

[1] R. Langner, "Stuxnet: Dissecting a Cyberwarfare Weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[2] S. Karnouskos, "Stuxnet Worm Impact on Industrial Cyber-Physical System Security," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2011, pp. 4490–4494.

[3] G. Neil, "On Board Train Control and Monitoring Systems," *Professional Development Course on Electric Traction Systems, IET 13th*, pp. 1–27, 2014.

[4] International Electrotechnical Commission (IEC), "IEC 61375-2-5, Electronic Railway Equipment - Train Communication Network - Part 2-5: Ethernet Train Backbone," Geneva, 2014.

[5] T. Kiravuo, M. Sarela, and J. Manner, "A survey of ethernet LAN security," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, pp. 1477–1491, 2013.

[6] R. Piggin, "Development of industrial cyber security standards: IEC 62443 for scada and industrial control system security," in *IET Conference on Control and Automation: Uniting Problems and Solutions*, Birmingham, 2013, pp. 1–6.

[7] F. Reichenbach, J. Endresen, M. M. R. Chowdhury, and J. Rossebø, "A pragmatic Approach on Combined Safety and Security Risk Analysis," *IEEE 23rd International Symposium on Software Reliability Engineering Workshops*, pp. 239–244, 2012.

[8] S. Kriaa, M. Bouissou, F. Colin, Y. Halgand, and L. Pietre-Cambacedes, "Safety and Security Interactions Modeling Using the BDMP Formalism: Case Study of a Pipeline," *International Conference on Computer Safety, Reliability and Security*, pp. 326–341, 2014.

[9] M. Steiner and P. Liggesmeyer, "Combination of Safety and Security Analysis - Finding Security Problems That Threaten The Safety of a System," *SAFECOMP 2013-Workshop DECS (ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*, 2013.

[10] S. Hussain, A. Kamal, S. Ahmad, G. Rasool, and S. Iqbal, "Threat Modelling Methodologies : a Survey," *Science International (Lahore)*, vol. 26, no. 4, pp. 1607–1609, 2014.

[11] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer, "DAG-based attack and defense modeling: Don't miss the forest for the attack trees," *Computer Science Review*, vol. 13, pp. 1–38, 2014.

[12] A. Shostack, *Threat Modeling: Designing for Security*. Wiley, 2014.

[13] International Electrotechnical Commission (IEC), "IEC 61375-1:2007, Electric Railway Equipment - Train Bus - Part 1: Train Communication Network," Geneva, 1999.

[14] J. Jiménez, J. L. Martín, A. Zuloaga, U. Bidarte, and J. Arias, "Comparison of two designs for the multifunction vehicle bus," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 5, pp. 797–805, 2006.

[15] Union Internationale Des Chemis de Fer (UIC), "UIC CODE 556, Information transmission in the train," 1998.

[16] G. Fadin, "ICT Standardisation on board of trains - An overview of current and upcoming standards," *CENELEC-ACRI Workshop*, no. October, 2011. [Online]. Available: http://www.integrail.info/ICT2011/1-2Fadin-Overviewofcurrentandupcomingstandards.pdf

[17] A. Zuloaga, A. Astarloa, J. Jimenez, J. Lazaro, and J. A. Araujo, "Cost-effective redundancy for Ethernet train communications using HSR," in *Proceedings IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, 2014, pp. 1117–1122.

[18] H. Kirrmann and P. A. Zuber, "The IEC/IEEE Train Communication Network," *IEEE Micro*, vol. 21, no. 2, pp. 81–92, 2001.

[19] International Electrotechnical Commission (IEC), "IEC 61375-2-3, Electronic Railway Equipment - Train Communication Network - Part 2-3: TCN Communication Profile," 2015.

[20] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat Modeling as a Basis for Security Requirements," *Symposium on requirements engineering for information security (SREIS)*, pp. 1–8, 2005.

[21] A. Main and P. C. V. Oorschot, "Software Protection and Application Security: Understanding the Battleground," *International Course on State of the Art and Evolution of Computer Security and Industrial Cryptograghy*, no. June, pp. 1–19, 2003.

[22] B. Schneier, "Attack Trees," *Dr. Dobb's Journal of Software Tools*, vol. 24, no. 12, pp. 21–29, 1999. [Online]. Available: http://www.schneier.com/paper-attacktrees-ddj-ft.html

[23] L. Pietre-Cambacedes and M. Bouissou, "Beyond attack trees: Dynamic security modeling with Boolean logic Driven Markov Processes (BDMP)," *European Dependable Computing Conference (EDCC)*, pp. 199–208, 2010.

[24] C. Salter, S. O. Saydjari, B. Schneier, and J. Wallner, "Toward A Secure System Engineering Methodology," *Proceedings of the workshop on New Security Paradigms*, pp. 2–10, 1998.

[25] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[26] S. Kumar and E. H. Spafford, "A Pattern Matching Model for Misuse Intrusion Detection," in *Proceedings of the 17th National Computer Security Conference (NCSC'94)*, Baltimore, USA, 1994, pp. 11–21.

[27] J. P. McDermott, "Attack Net Penetration Testing," in *Proceedings of the 2000 workshop on New Security Paradigms (NSPW'00)*. Cork, Ireland: ACM, 2001, pp. 15–21.

[28] G. C. I. Dalton, R. F. Mills, J. M. Colombi, and R. A. Raines, "Analyzing Attack Trees using Generalized Stochastic Petri Nets," in *Proceedings of the 2006 IEEE Workshop on Information Assurance*. West Point, NY: IEEE, 2006, pp. 116–123.

[29] T. M. Chen, J. C. Sanchez-Aarnoutse, and J. Buford, "Petri Net modeling of Cyber-Physical Attacks on Smart Grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 741–749, 2011.

[30] M. Bouissou and J.-L. Bon, "A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes," *Reliability Engineering & System Safety*, vol. 82, no. 2, pp. 149–163, 2003.

[31] S. Kriaa, M. Bouissou, and L. Pièvre-Cambacédès, "Modeling the Stuxnet attack with BDMP: Towards more formal risk assessments," *7th International Conference on Risks and Security of Internet and Systems, CRiSIS 2012*, pp. 1–8, 2012.

[32] A. Shostack, "Experiences Threat Modeling at Microsoft," in *Modeling Security Workshop, in Association with MODELS '08*, 2008, pp. 1–11.

[33] OWASP, "OWASP Top Ten Project," 2016, accessed on 2.9.2016. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[34] MITRE, "CAPEC - Common Attack Pattern Enumeration and Classification," 2016, accessed on 1.9.2016. [Online]. Available: https://capec.mitre.org/

A Case Study Train Control and Management System

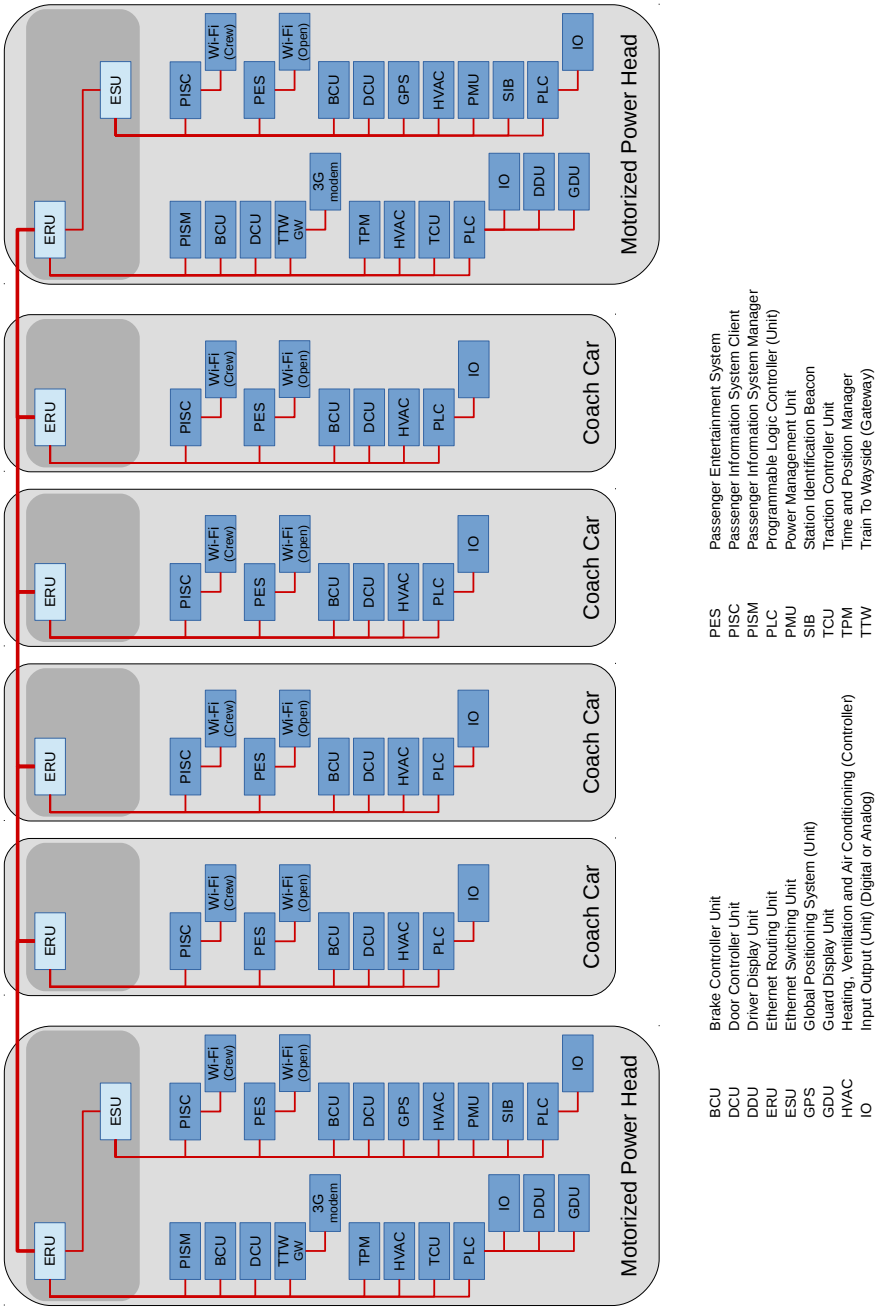

Figure A1: Structure of TCMS design of Case Study

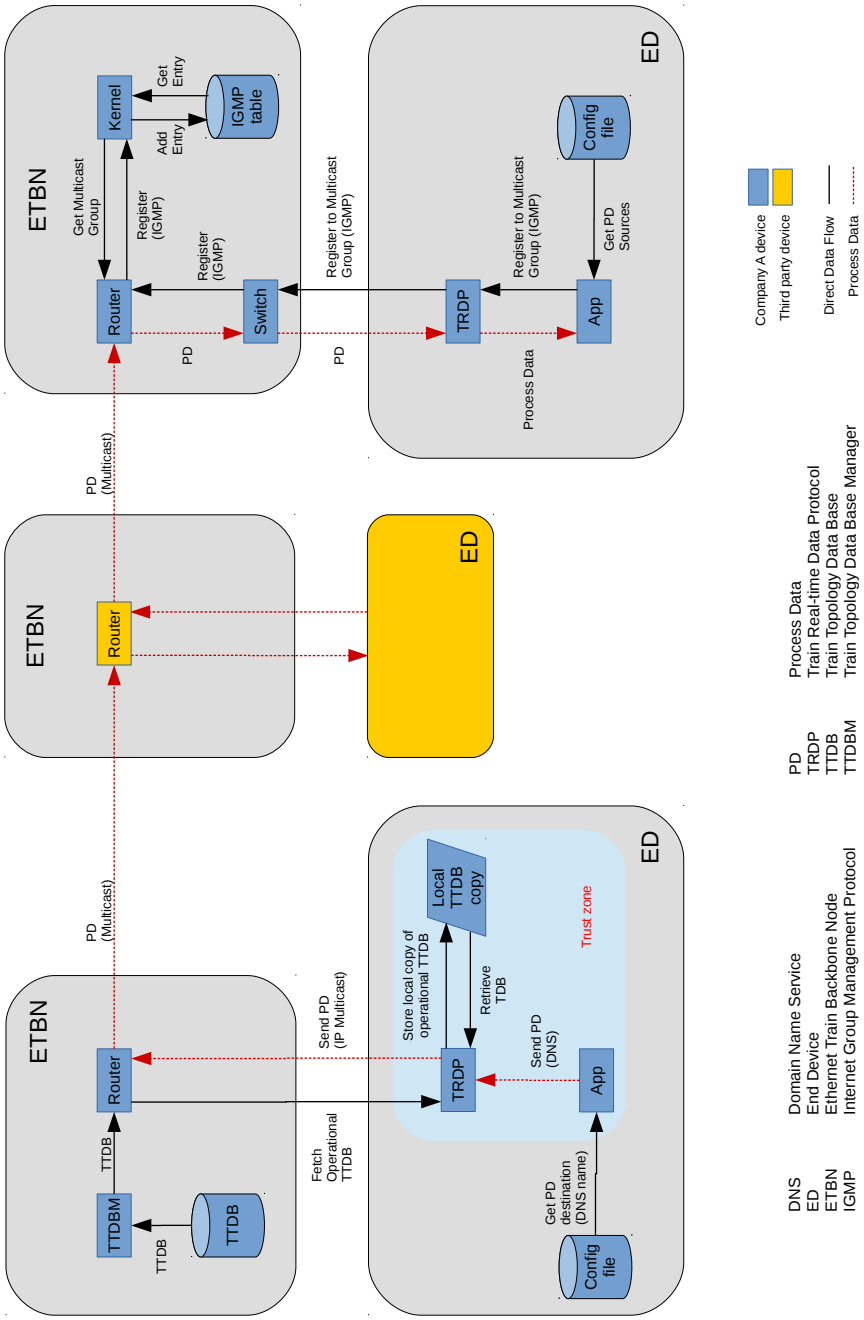

Figure A2: Procedure for sending process data over ETB

B Case Study PISM Data Flow Diagrams

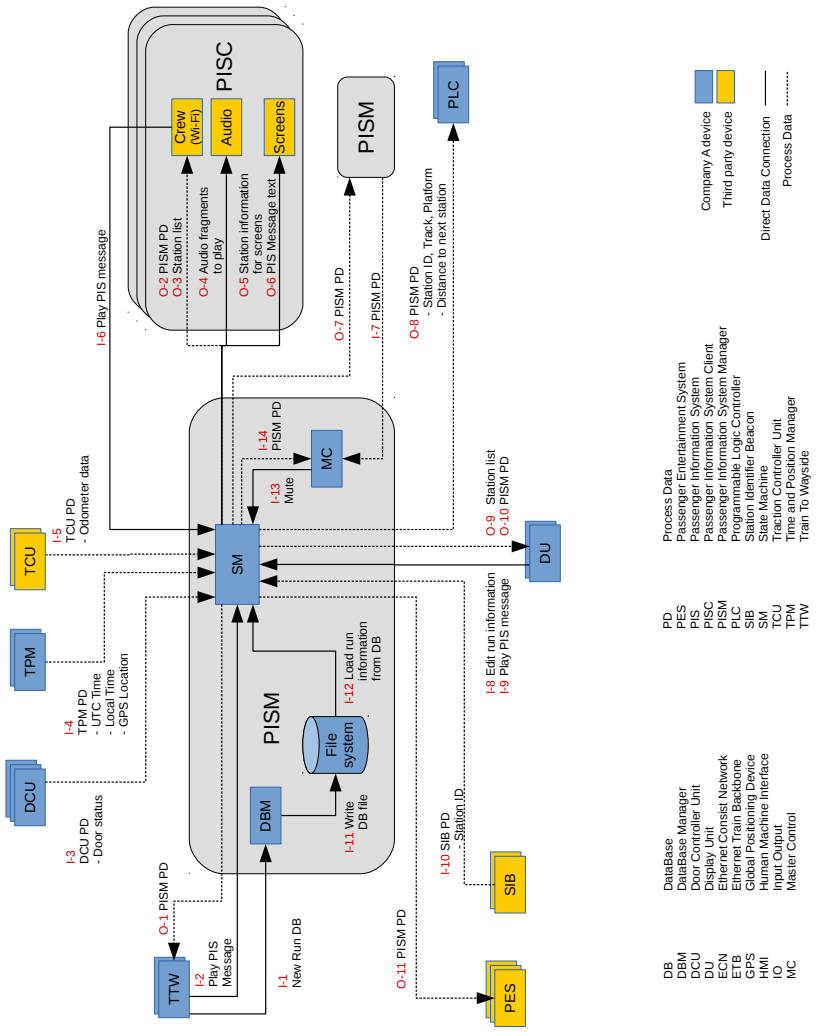

Figure B1: PISM Data Flow Diagram of direct connections

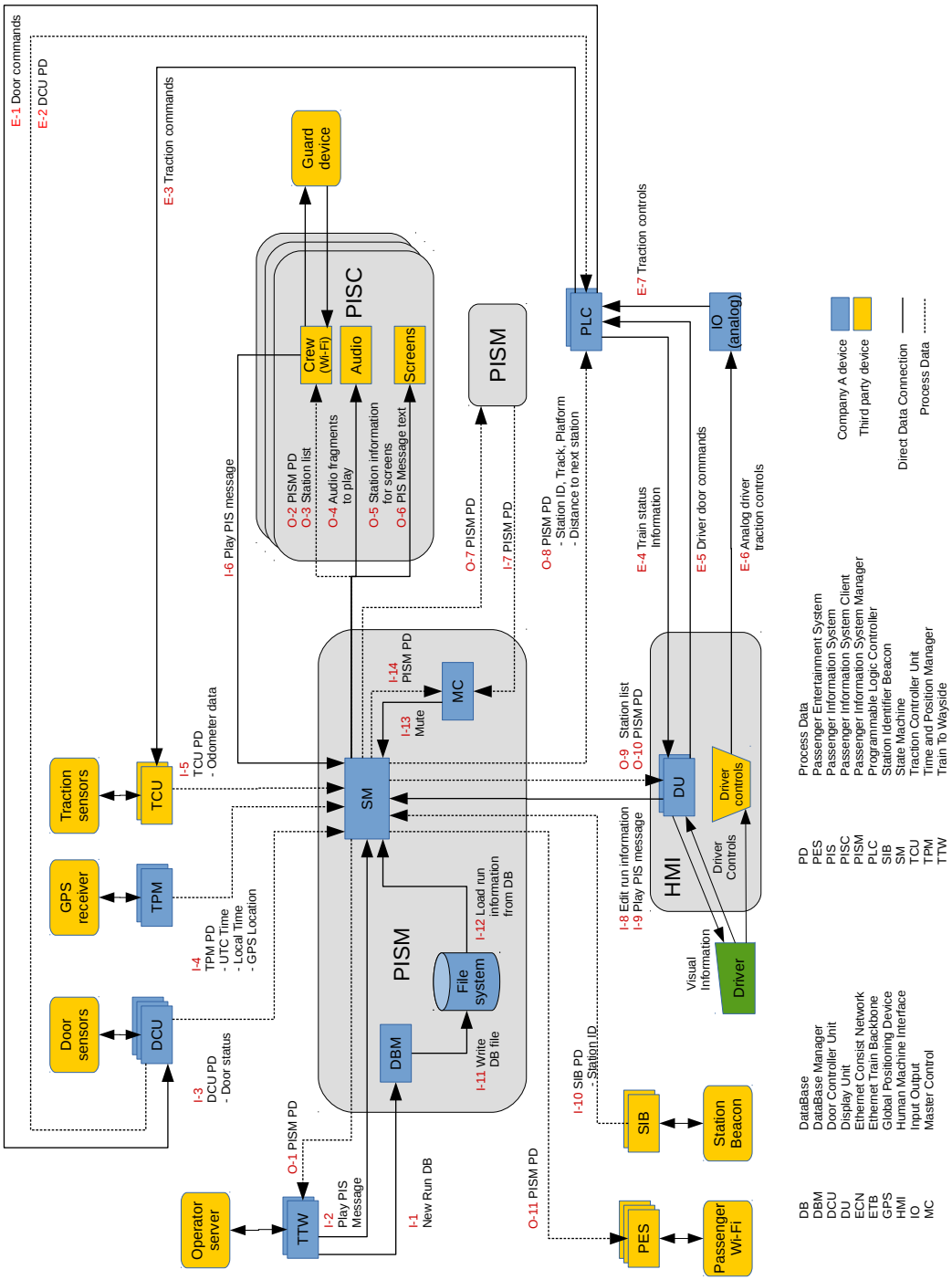

Figure B2: PISM Data Flow Diagram of operational context

PISM operation description

General purpose:

- Track train progress as it travels along a specified route
- Update passenger information systems, screens and audio announcements according to route progress
- Provide information to other systems about the train's state along the specified route
- Store route information in the route database

Processes

- The DataBase Manager (**DBM**) updates the local route database on the train. When a new route database is received from Train To Wayside (TTW), DBM parses the XML content and stores the information in a SQLite database.
- Master Control (**MC**) is the process that independently decides if the local PISM should assume active or passive mode. It makes this decision based on the local and remote PISMs' process data. If both PISMs are otherwise OK and could both assume active role, the PISM located in the lower node number will assume the active role while the other assumes passive mode.
- The State Machine (**SM**) process handles all other of the PISMs functions. The SM keeps track of route progress by listening to GPS, odometer information from traction and station information from the station identifier beacon system. PISM's generic process data is gathered and sent by the SM process. The SM may be divided into several threads, but threads belong to the same trust zone as they share their memory space with each other.

Descriptions of data flows

The current route to follow is loaded by the driver via the DDU. The driver enters the id of the route which is sent to the SM process. The SM process then fetches the route information from the route database in the local file-system and loads it into memory. It then populates information related to the currently loaded route in the PISM process data. The starting position of the loaded route has to match the current location of the train.

As the train moves along its route, PISM determines its position on the route by using GPS information (via TPM) as a primary source, odometer data (from TCU) as a secondary source and station beacon signals (from SIB) for calibration and confirmation that a station has been reached.

The SM process can have the following states: normal, approaching, at station and departing. The normal state is active when the train is travelling between

stations. The state will switch to approaching when the train approaches the next station. Once the train has stopped within a reasonable distance from a station and the doors have been opened, the at station state is entered. Finally, as the train doors have been closed and the train gains speed the SM enters the departing state.

While approaching a station, PISM will inform the PLC that it has acquired a SIB signal and that the distance to next station is less than 100 meters. From this point on the PLC takes control of traction and automatically brings the train to a halt at approximately the distance to next station equal to zero. The driver may naturally override the PLC control and stop the train manually.

Once the train reaches a station, PISM provides necessary information for DCUs. PISM knows on which side the platform is supposed to be by cross checking the SIB signals and the PISMs route database. The driver may visually confirm the platform location in case of inconsistencies in the data. The driver manually requests for door lock release on the correct side of the train. PLC verifies that PISM reports the platform side according to the drivers input and finally commands the doors to release the locks and open the doors.

Time from TPM is used to keep track of the route schedule. If TPM process data becomes available, PISM will use the local system time of the computer it is running on.

Audio announcements and text messages may be played or shown automatically to the PISC systems as PISM changes its state. This information is delivered to the sub-systems in the form of fragment IDs. The sub-systems will have their own stored fragment databases and will play or show the corresponding data fragments commanded by PISM.

C Case Study PISM Data Flow Descriptions

Table C1: PISM data flow identification table

PISM Inputs (I-)

ID	Packet type	Relevant data
I-1	DB Transfer	New XML file containing run information
I-2	PIS Message	Text message information for screens, audio fragments for audio system
I-3	DCU PD	PISM only uses the door status information from this packet
I-4	TPM PD	UTC Time, Local Time, GPS Location
I-5	TCU PD	Odometer data for calculating trains position on a run
I-6	PIS Message	See input 2
I-7	PISM PD	Heartbeat and “sources ok”-signal
I-8	Run Update	All data. A new run and updating the current run is done with the same packet
I-9	PIS Message	See input 2
I-10	SIB PD	Station ID information is used to confirm train is on a station
I-11	DB Store	New database file containing updated run information
I-12	DB Store	See input 11
I-13	Mute Master	Information about which PISM is master. Mute commanded if PISM is slave
I-14	PISM PD	Heartbeat and sources ok. Shared between SM and MC via OS IPC (Inter Process Communication)

PISM Outputs (O-)

ID	Packet type	Relevant data
O-1	PISM PD	All data. Operator can request system status information.
O-2	PISM PD	All data. Guard may view PISM data on the portable device
O-3	Stations All	All data. Guard may view all stations on the currently loaded run
O-4	Audio Fragments	All data
O-5	Stations Screen	All data

(continues on next page)

Table C1: PISM data flow identification table *(continued)*

O-6	PIS Message Text	All data
O-7	PISM PD	Heartbeat and “sources ok”-signal
O-8	PISM PD	Station ID, Track number and Platform side
O-9	Stations All	All data. Displayed to driver on DU
O-10	PISM PD	Distances displayed to driver on DU
O-11	PISM PD	Destination, next stopping station, distance to next station displayed to passengers

External Packets (E-)

ID	Packet type	Relevant data
E-1		Release door locks, Open doors
E-2		Door status information used by PLC to inhibit Traction
E-3		Full traction controls
E-4		Generic train status information displayed to Driver
E-5		Driver door commands: Open doors, release door locks. (Left/Right)
E-6		Physical driver traction controls
E-7		Traction controls measured by IO unit

Table C2: PISM data flow descriptions - Process data

DCU PD

Data packet content	Type	Unit
Heartbeat	U16	
Door status	U8[N]	0=Open, 1=Closing, 2=Opening, 3=Closed, 4=Locked
Door failure	BOOL[N]	
Emergency bypass activated	BOOL[N]	

PISM PD

Data packet content	Type	Unit
Heartbeat	U16	
Sources OK	BOOL	
Run ID	U16	
Final destination	STRING	
Next station	STRING	
Next stopping station	STRING	
Previous station	STRING	
Distance to next station	U32	m
Distance to previous station	U32	m
Run state	U8	0=Normal, 1=Approaching, 2=At station, 3=Departing
Station ID from SIB	STRING	
Track number	U8	
Platform side	U8	0=N/A, 1=Left, 2=Right

SIB PD

Data packet content	Type	Unit
Heartbeat	U16	
Current Station ID	STRING	
Current Station Name	STRING	
Track number	U8	

TCU PD

Data packet content	Type	Unit
Heartbeat	U16	
Speed	S32	m/h
Odometer total	U64	m

(continues on next page)

Table C2: PISM data flow descriptions - Process data *(continued)*

Traction interlock active	BOOL

TPM PD

Data packet content	Type	Unit
Heartbeat	U16	
GPS Latitude	U32	
GPS Longitude	U32	
UTC Time	U32	Unix time
Local Time	U32	Unix time

Table C3: PISM data flow descriptions - Point-to-point data

Audio Fragments

Data packet content	Type	Unit
Cancel current announcement	BOOL	
Audio fragments	U32 [256]	

DB Store

Data packet content	Type	Unit
Data base file	SQLite	

DB Transfer

Data packet content	Type	Unit
XML file containing run data	XML file	
Checksum	MD5	

Mute Master

Data packet content	Type	Unit
Activate slave mode	BOOL	

PIS Message

Data packet content	Type	Unit
Display to Crew	BOOL	
Display to Passengers	BOOL	
Play audio	BOOL	
Audio fragments	U32 [256]	
Special Message	BOOL	
Message fragments	U32 [256]	
Message	STRING	

PIS Message Text

Data packet content	Type	Unit
Message text	STRING	

(continues on next page)

Table C3: PISM data flow descriptions - Point-to-point data *(continued)*

Run Update

Data packet content	Type	Unit
Run ID	STRING	
Stopping stations	BOOL [256]	
Play run audio announcement	BOOL	

Stations All

Data packet content	Type	Unit
Stations list	STRING [256]	

Stations Screen

Data packet content	Type	Unit
Destination	STRING	
Next stopping station	STRING	

D STRIDE Template

Table D1: Tempalate for the STRIDE analysis

Data Flow	ID	Threat description	Simple attacker steps	Notes	Affects Component	Severity	Mitigation	Mitigation Notes
I-1	**1**	An attacker could impersonate an administrator at the front end of the system.	-Begin by executing step 1 -Then continue with step 2	Extra notes if necessary	Main component	2	Require authentication	Use Public-Key Crypto

E STRIDE Results

Table E1: Spoofing threats

Data Flow	ID	Threat description	Notes	Affects Component	Severity	Mitigation	Mitigation Notes
I-1	**1001**	Spoofing the XML file will allow any input and may abuse the XML parser		PISM DBM	3	- Use XML schema and validate XML before passing to parser - Authenticate sender - Authenticate XML using digital signature or MAC	
I-1	**1002**	Spoofing XML will allow malformed input that may crash XML parser or if accepted by XML parser the PISM SM		PISM DBM PISM SM	3	- Use XML schema and validate XML - Keep XML parser up to date in case of parser bug - Validate XML parser - Create own content validation rules	

(continues on next page)

Table E1: Spoofing threats *(continued)*

Data Flow	ID	Threat description	Notes	Affects Component	Severity	Mitigation	Mitigation Notes
I-3	**1003**	Because of being PD, another device might spoof being a non-existent DCU and stopping usage (by having the doors stuck open)		PISM SM PD	2	- Use router firewall rules to check sources of DCU PD - Authenticate DCU PD messages using digital signatures or MAC	- Signature or MAC calculations might consume significant CPU/HW resources - What is the responsibility of the originating ETBN to check the source of the data? This ETBN may be another company's device.
I-4	**1004**	TPM spoofing, PISM might listen to a fake or compromised TPM and get e.g. incorrect GPS data. This might lead to the train stopping in the middle of the track or to overshoot the station		PISM SM PLC PD	3	- Authenticate TPM using digital signatures or MAC - Use redundant TPM system to identify spoofed TPM	- Redundant systems increases system complexity and has an effect on availability - Signature or MAC calculations might consume significant CPU/HW resources
I-8	**1005**	Spoofing a valid message with incorrect RUN information will cause PISM to be unable to load the RUN, clearing the screens and confusing the Driver		PISM SM	1	- Authenticate DU messages using digital signatures or MAC - Do not clear RUN information from DU screens in case of failed load. Indicate load error to driver.	

(continues on next page)

Table E1: Spoofing threats *(continued)*

Data Flow	ID	Threat description	Notes	Affects Component	Severity	Mitigation	Mitigation Notes
I-8	**1006**	Spoofing a valid message with incorrect RUN information might cause PISM to think it is approaching a station and PLC to stop the train in the middle of the tracks		PISM SM PLC	2	- Authenticate DU messages using digital signatures or MAC - Use SIB signals to verify that the train is approaching a station	
I-10	**1007**	Spoofing an incorrect station might cause people to exit train at wrong station		PISM SM	3	- Authenticate SIB messages using digital signatures or MACs	
I-10	**1008**	Spoofing an incorrect station might cause doors to be opened on the wrong side, i.e. where there is no platform		PISM SM PLC	3	- Authenticate SIB messages using digital signatures or MACs - Use PISM RUN DB information to verify station and platform side. Require driver to verify platform side manually in case of inconsistent information	

Severity value meanings: 0=Duplicate or N/A 1=Minor 2=Needs Review 3=High

Table E2: Tampering threats

Data Flow	ID	Threat description	Notes	Affects Component	Severity	Mitigation	Mitigation Notes
I-1	**2001**	Changing platform side in RUN information of stations in the XML might cause doors to be opened on a station on the wrong side		PISM SM PLC	3	- Use digital signatures or MAC to verify integrity of the XML	- MAC is not sufficient for non-repudiation, which might be a needed property
I-3	**2002**	Modifying DCU PD data might cause PISM to think doors are open/closed when they are not		PISM SM PLC	3	- Use digital signatures or MAC to verify integrity of the PD - Use traction interlock system to stop train from departing without all doors being closed and locked	- Traction interlock system system would require expensive separate cabling to be effective.
I-4	**2003**	TPM time data to PISM has been modified	This might cause train schedules not to be followed	PISM SM	1	- Use digital signatures or MAC to verify integrity of TPM PD	
I-5	**2004**	Incorrect odometer data might confuse PISM	Odometer data is authoritative distance information when GPS data is unavailable	PISM SM	3	- Use digital signatures or MAC to verify integrity of TPM PD - Use fallback method, such as limit speed in case GPS is also unavailable. Verify position in a discrete way using SIB information.	

(continues on next page)

Table E2: Tampering threats *(continued)*

Data Flow	ID	Threat description	Notes	Affects Component	Severity	Mitigation	Mitigation Notes
I-7	**2005**	PISM might announce itself as being OK even though it is not sending any data, causing conflict between PISMs effectively a situation where no PISM is active	Only possible from node 1, which assumes master role if both PISM are OK		3	- Improve logic for determining PISM - Let both PISMs produce and send data and outsource decision of which data should be used to a third device, for instance PLC	- A more complex decision logic may introduce new problems and cause more issues than it solves
I-9	**2006**	Changing message content might show arbitrary messages to passengers, e.g. inform passengers that train are at the end station or that train is broken and will halt on the next station			1	- Authenticate message using digital signature or MAC	
I-10	**2007**	Modifying PD data might make PLC think we are at a station and stop the train too early			2	- Authenticate message using digital signature or MAC - Identify incorrect signal by comparing to other sources like GPS - Allow driver to confirm which information is correct if an inconsistency between data sources is recognized	- Should beacon or GPS be authoritative?

(continues on next page)

Table E2: Tampering threats *(continued)*

Data Flow	ID	Threat description	Notes	Affects Component	Severity	Mitigation	Mitigation Notes
I-12	**2008**	Another process might have inserted additional information to the DB, causing PISM to use incorrect RUN data			3	- Sign the DB using PKC - Restrict access using OS ACLs or similar	
SM	**2009**	Gaining root access to the SM process allows almost full control of train			3	- Verify all SW running, e.g. by comparing running software to binaries in ROM - Sign all software	

Severity value meanings: 0=Duplicate or N/A 1=Minor 2=Needs Review 3=High

Table E3: Repudiation threats

Data Flow	ID	Threat description	Notes	Affects Component	Severity	Mitigation	Mitigation Notes
I-1	**3001**	Loading an incorrect XML might cause denial of service, not logging the event stops from proving an incorrect XML was loaded		PISM SM	1	- Log XML updates	
I-5	**3002**	If GPS data is unavailable, there is no way to verify how far the train has traveled without odometer data		PISM SM	1	- Log odometer data - Odometer data should be part of data recorded to black box	
I-8	**3003**	Driver might use the system in a non-intended way, e.g. load new run in the middle of old run causing delays		PISM SM	1	- Log driver DU actions	

Severity value meanings: 0=Duplicate or N/A 1=Minor 2=Needs Review 3=High

Table E4: Information disclosure threats

Data Flow	ID	Threat description	Notes	Affects Component	Severity	Mitigation	Mitigation Notes

Severity value meanings: 0=Duplicate or N/A 1=Minor 2=Needs Review 3=High

Table E5: Denial of Service threats

Data Flow	ID	Threat description	Notes	Affects Component	Severity	Mitigation	Mitigation Notes
I-1	**5001**	Stopping transmission will stop the RUN DataBase from being updated		PISM SM	1	- Assign time validity for RUN databases so that driver will notice when a database is out of date	
I-1	**5002**	Constantly updating the DB, sending the XML, will allow consumption of CPU resource and possibly starvation of CPU		PISM DBM	3	- Do not allow processing of multiple XML files at once - Enforce a delay between XML updates - Perform early XML validation checks before engaging in CPU intensive processing - Assign the DBM process a low priority	
I-1	**5003**	Constantly sending a new DB might allow filling up the file system of the PISM		PISM DBM	1	- Limit the number of databases stored on disk	
I-1	**5004**	Constantly updating the DB might allow a denial of service attack on the network, at least the local consist network			0		PISM can't mitigate generic TCN denial of service, out of scope of PISM analysis. Threat should be moved to generic TCN analysis

(continues on next page)

Table E5: Denial of Service threats *(continued)*

Data Flow	ID	Threat description	Notes	Affects Component	Severity	Mitigation	Mitigation Notes
I-1	**5005**	Inserting a valid XML with incorrect RUN data will stop the driver from loading runs and departing for service		PISM DBM	3	- Authenticate the XML using digital signatures or MAC	
I-3	**5006**	If DCU PD data is not sent, it might stop the train from being used		PISM SM	1	- Driver bypass functions should be considered to improve availability when there is missing DCU PD	
I-4	**5007**	Time might not be sent at all, causing problems for following run timetables		PISM SM	1	- Allow using local CPU time as backup time source	
I-5	**5008**	The TCU might not send PD data at all		PISM SM	1	- Use redundant TCU if possible	
I-5	**5009**	Flooding data over ETB might cause network bandwidth starvation			0		PISM can't mitigate generic TCN denial of service, out of scope of PISM analysis. Threat should be moved to generic TCN analysis

Severity value meanings: 0=Duplicate or N/A 1=Minor 2=Needs Review 3=High

Table E6: Elevation of Privilege threats

Data Flow	ID	Threat description	Notes	Affects Component	Severity	Mitigation	Mitigation Notes
I-2	**6001**	Gaining root access might allow TTW to send messages to any device and pretend to be another device			0		- Can't be mitigated by PISM. Should be part of TCN analysis - Restrict data traffic to combinations of specific IP source addresses and destination addresses. This could be handled by ETBN
I-2	**6002**	The message format might allow sending arbitrary messages to audio, screens, driver and guard		PISM SM	2		No mitigations proposed
I-9	**6003**	Driver may send false information to passengers using the arbitrary PIS message feature		PISM SM	2	- Allowing only predefined messages will improve security at the cost of functionality	Driver is in control of PISM and can always send some messages to PIS screens and audio system, can't be completely mitigated.
I-12	**6004**	Another process might have deleted the DB causing PISM Run information to be inaccessible		PISM SM		- Make PISM SM use a local copy of the DB, so the DBM can not delete the DB which is in use	- Similar to threat 2008

Severity value meanings: 0=Duplicate or N/A 1=Minor 2=Needs Review 3=High

F BDMP Results

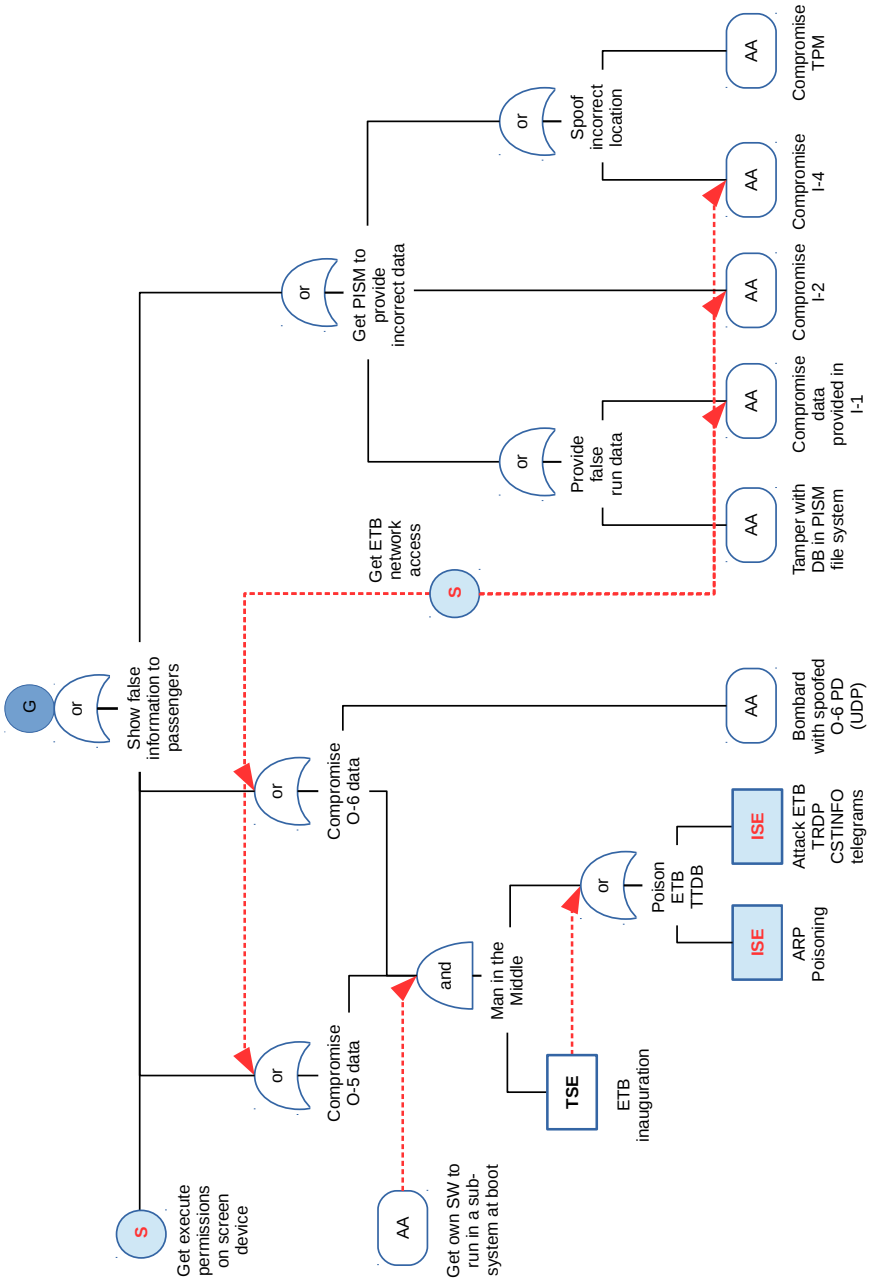

Figure F1: Main BDMP graph

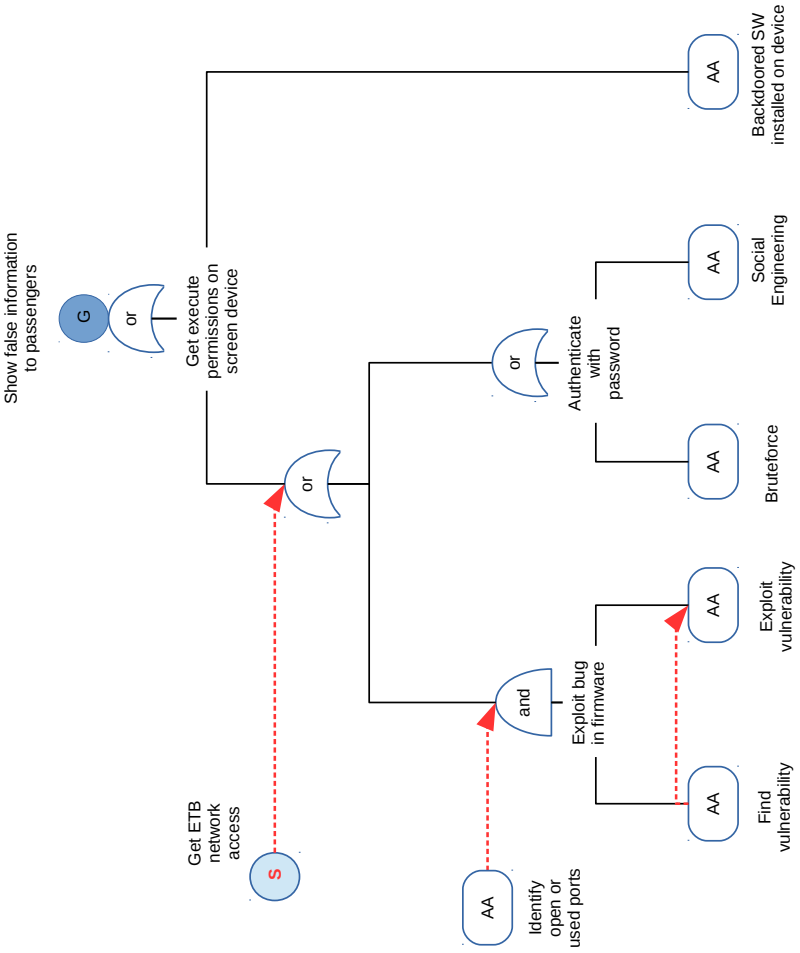

Figure F2: BDMP subgraph of direct screen attack

G Questionnaire Results

Table G1: Questionnaire results from the case study workshop

	Assigned values					
	1	2	3	4	5	
	Not at all	A little	Moderately	Significantly	Very much	Average
Do you think the workshop improved your understanding about TCMS security?	**0**	**1**	**3**	**3**	**0**	3.3
	Not at all	A little	Moderately	Significantly	Very much	Average
Do you think the used threat modeling process would be useful in the future?	**0**	**1**	**3**	**2**	**1**	3.4
	Not at all		Somewhat		Greatly	Average
Do you think threat modeling in general seems useful regarding TCMS security?	**0**		**2**		**5**	4.4
	None		A few		Many	Average
Did you learn about new threats to TCMS systems?	**2**		**4**		**1**	2.7
	None		A few		Many	Average
Did you learn about new attacks on TCMS systems?	**2**		**4**		**1**	2.7

Comments:

- Useful method to identify threats and prevent attacks. Good brainstorming. Also other things rise up concerning TCMS in general
- Should be standard practice in every project