PROTECTING CRITICAL INFRASTRUCTURE SYSTEMS USING CYBER, PHYSICAL, AND SOCIO-TECHNICAL MODELS

BY

CARMEN CHEH

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Doctoral Committee:

Professor William H. Sanders, Chair
Dr. Binbin Chen, Advanced Digital Sciences Center
Professor Carl Gunter
Professor Klara Nahrstedt
Dr. Jia Xu, Singapore Telecommunications Limited

ABSTRACT

Critical infrastructure systems are vital to all nations, and incapacitating such systems can result in devastating impact on the general public. Therefore, it is essential to protect such systems from malicious threats. Today, the increasing interconnectedness of critical infrastructure systems has greatly improved system efficiency at the cost of a larger attack surface. In recent years, we have seen cyber-attack campaigns in addition to physical attacks on various critical infrastructure systems around the world. Thus it is important to protect such systems from adversarial physical and cyber threats.

In this dissertation, we propose to protect critical infrastructure systems by (1) assessing the safety of the system and (2) detecting malicious physical threats on the system by using models that integrate the cyber, physical, and human domains. We support our dissertation statement by applying our contributions to a railway system case study.

First, we perform a security analysis to identify malicious threats and suggest potential detection mechanisms to strengthen the system defense. We define a general ontology that represents cyber-physical system components and relationships among them, and cyber and physical actions by a human actor. We model a railway station using concepts from that ontology, and feed the model into the ADVISE tool to automatically generate an attack execution graph. We analyze that attack execution graph and show that the addition of a potential defense system for physical movement is an effective mechanism for improving system security.

We then conduct a safety analysis to identify potential cyber attacks on the railway signaling system that would violate system safety. To do so, we use networks of timed automata to model the cyber-physical control feedback loop that drives system service. We develop a set of transformations on state automata that represent combinations of cyber actions of a human actor. Then, we perform model checking to identify the cyber attack scenarios that would compromise system safety. We demonstrate that while certain safety countermeasures can mitigate attacks by outsider adversaries, attacks by insider adversaries would still succeed. Reapplication of our security analysis with the addition of the cyber-attack vectors that we discovered shows that adversaries prefer to use physical and social means to gain access to the railway station and attack the system.

Thus, to strengthen the physical security of the system, we develop defense systems that detect suspicious physical movement by human actors in a railway station. We identify abnormal movement behavior by comparing sequences of movement to historic normal move-

ment models. In doing so, we first build models of normal movement behavior by using historic building access control logs. Then, in real-time, we screen physical accesses and check for deviations in users' behavior from the normal movement behavior model. If we find any, we flag those physical accesses as suspicious. We show that our detection approach is able to flag suspicious behavior with increasing likelihood as the malicious movement sequence increases.

We then develop approaches to identify tailgating in building access control logs by using physical constraints about human movement and space occupancy. This work was motivated by the observation that adversaries may thwart building access control systems by physical and social means, e.g., by "tailgating," or following closely behind, an authorized person. We use cyber and physical data sources to build models of the physical locations of people. Then, we flag tailgating instances when the physical constraints on human movement and space occupancy are violated. We show that our detection approach is able to identify certain tailgating scenarios and that the addition of other data sources, such as physical data sources, allows us to build a more complete model of physical location.

Finally, we reapply our security analysis with the addition of defense systems. The results of our analysis show that the inclusion of the defense systems incentivizes adversaries to expend more effort and time to launch a cyber-attack campaign instead of attempting to gain access to the railway station. Therefore, our defense systems help to strengthen the overall security posture of the system.

In conclusion, we identify several cyber and physical attack scenarios that would affect system safety, and we develop physical defense systems that demonstrably increase the system's security posture. Thus, in this dissertation, we present an integration of security analysis, safety analysis, and system defense that uses cyber, physical, and socio-technical models to protect critical infrastructure systems.

To Mum and Dad, I couldn't have done it without you.

ACKNOWLEDGMENTS

I would first like to thank my adviser, Professor William H. Sanders, for the financial, technical, and emotional support that he has provided to me over these years. He gave me the freedom to pursue my research interests and pushed me to be an independent researcher while always providing me with assurance and confidence that I had the ability to do great work. I would not have been able to complete my dissertation work without his advice and support.

I thank my University of Illinois at Urbana-Champaign (UIUC) committee members, Professor Carl Gunter and Professor Klara Nahrstedt, for their insightful feedback and guidance on my dissertation work. Their doors were always open to me when I needed help, and some of the technical contributions in this dissertation were based on research discussions with them, for which I'm very grateful. I thank them for believing in and supporting my work.

I would also like to thank my Singapore collaborators in the Advanced Digital Sciences Center (ADSC) and A*STAR, Dr. Binbin Chen and William G. Temple. My dissertation work started from the internship that I did at ADSC. Despite time zone differences, I had countless hours-long meetings with them that culminated in paper publications that I would not have been able to pull off by myself. Their detailed feedback and hands-on guidance pushed me to work even harder than before, and I sincerely thank them for all their time and effort. Dr. Binbin Chen and Dr. Jia Xu also graciously agreed to be my committee members, and I thank them for sacrificing their sleep to attend my presentations. I also thank my ADSC colleagues, Yan Li, Yue Wu, and Alex Tran for their help and support.

A huge thanks goes to my railway system collaborators in Singapore, without whom none of this work would have been possible. In particular, I thank the SMRT Communication and Control staff for their support of my work and for always answering my countless questions patiently. I thank Bobby Goh and Alvin Lim for providing me with immense data and knowledge about the railway system. These helpful, friendly, and approachable railway staff members have made my collaboration with them very fruitful and enjoyable.

My deepest gratitude goes to the members of the PERFORM group who helped me immensely in my work and made me feel very much at home in a foreign country. Our offices were like a second home to me, and despite my constant jokes about lack of space (among other things), I enjoyed the time spent with them talking about various random things aside from work. Also, many of the technical contributions in my work and the skills I've picked up are due to my interactions with them. In particular, I thank Ahmed Fawaz,

Uttam Thakore, Atul Bohara, Mohammad Noureddine, Benjamin Ujcich, Michael Rausch, Ken Keefe, Brett Feddersen, Varun Badrinath Krishna, Ron Wright, and David Huang. I could not have asked for better colleagues.

I extend my warmest thanks to the Coordinated Science Lab (CSL) staff, Jenny Applequist, Linda Morris, and Dawn Cheek-Wooten. My writing has greatly improved with Jenny's feedback, and I'm glad to see that the number of red marks on my papers has dropped considerably since my first year in graduate studies. I could always count on Jenny to read my papers, even when deadlines were tight, and I sincerely apologize for always having to drop things on her at a busy time. I'll sorely miss relying on her to catch my English mistakes, and I hope that in the future, I'll get to share my other nonacademic writings with her as well. I also thank Linda Morris for being a warm, friendly, motherly figure to welcome me in the CSL main office when I dropped by. I always felt comfortable and at home in her presence, and I thank her for constantly being there to support me and my work. I also thank Dawn Cheek-Wooten for all her work in handling my conference travel bookings and expenses. I could always relax knowing that she would handle matters. I thank her for being a good friend to me too.

I thank all the friends I've made at UIUC who made my stay here all the more enjoyable. In particular, I thank the folks in the DEPEND group, Arjun Athreya, Subho Banerjee, Phuong Cao, Saurabh Jha, and Yoga Varatharajah; my mentor Giang Nguyen; and my friends in the Computer Science department, Vincent Bindschaedler and Susannah Johnson. I also thank my good friend Hsiao-Ying Huang for her companionship; I thank her for all our discussions, rants, and fun outings. A big thanks also goes to Uncle Szetoh and his family for being people I could trust and depend on during my stay at UIUC.

Finally, the people I thank the most are my parents, whose love and support made me who I am today. Whatever hardships I faced during the workday, I knew I could go home, tell them about it, and feel reassured and comforted. I'm glad that I could make them proud of me. Although I hardly ever say it out loud, I love you, Mum and Dad.

This work was supported in part by the Army Research Office under Award No. W911NF-13-1-0086 and National Research Foundation (NRF), Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2014NCR-NCR001-31) and administered by the National Cybersecurity R&D Directorate, and supported in part by the research grant for the Human-Centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center from Singapore's Agency for Science, Technology and Research (A*STAR). Any opinions, findings, and conclusions or recommendations expressed in this dissertation are those of the author(s) and do not necessarily reflect the views of the Army Research Office.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Critical infrastructure systems such as power grids, water treatment facilities, and transportation systems are vital to the public welfare of a nation. Incapacitating such systems would result in devastating impacts on the safety and well-being of the general public, and critical infrastructure protection programs are therefore in place to protect such systems from both non-malicious and malicious threats [1]. While non-malicious threats result in random, accidental failures, malicious threats are targeted towards causing large system failures. Thus, in this dissertation, we focus on protecting critical infrastructure systems from malicious threats.

Today, a lot of critical infrastructure systems are becoming increasingly interconnected. By interconnecting devices in those systems, we improve the system's efficiency, but at the expense of an increased attack surface. In addition to physical attacks, malicious adversaries are now able to launch cyber-attack campaigns that result in real-world consequences, impacting citizen welfare. For example, the Stuxnet worm that infected multiple nuclear power facilities in Iran resulted in the failure of many uranium-enriching centrifuges [2]. In 2015, the cyber attack on the Ukranian power grid caused a major disruption in electrical supply to customers for hours [3]. As those examples show, it is of the utmost importance to protect the safety of such critical infrastructure systems from adversarial threats that span both the cyber and physical domains.

1.1 CYBER-PHYSICAL SYSTEMS CHARACTERISTICS

Cyber-physical systems, of which critical infrastructure systems are an important subcategory, consist of sensors and actuators that are connected together using network equipment. These systems are unlike other typical networked systems, e.g., enterprise networks, for which a significant body of research has been conducted for decades. Thus, in more recent years, a lot of effort has been put into researching security mechanisms and enforcing them in cyber-physical systems for the sake of protecting such systems.

The operation of critical infrastructure cyber-physical systems is fundamentally different from that of other networked systems, since the services provided by critical systems impact people's physical lives (e.g., by transporting commuters, powering consumers' homes through the power grid, and supplying clean water to residences). Efficiency, continuous service, and timeliness are therefore important features for cyber-physical systems to maintain. Thus, safety and security measures that are to be built in must not affect the continuing operation

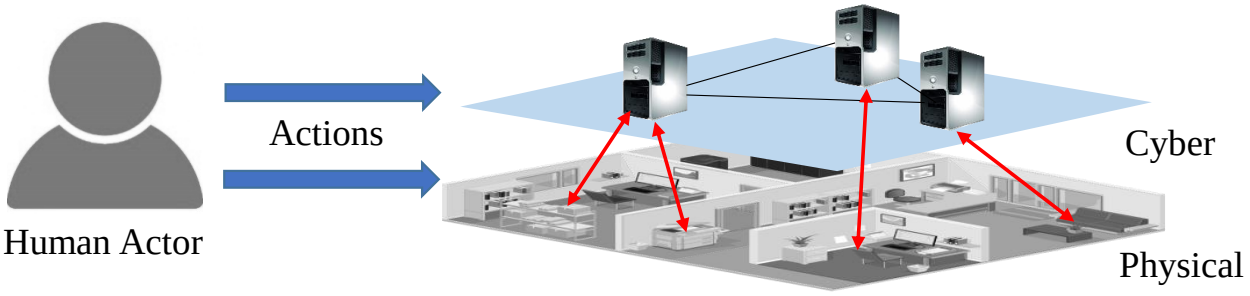


Figure 1.1: The three different domains of the system cyber-physical-human model. The human domain is shown by the User icon, which acts on the two other domains.

of a system.

We can see cyber-physical systems as encompassing three different domains: cyber, physical, and human, as shown in Figure 1.1. The arrows in that diagram show the interactions between the cyber and physical domain, as well as the human and system (cyber and physical) domain. In the following paragraphs, we describe those interactions in terms of the cyber-physical and socio-technical domains.

Cyber-Physical Domain. In cyber-physical systems, service is driven by physical processes and control feedback loops. Thus, system behavior is defined in terms of the physical operation of devices, whereas in other enterprise systems, user behavior drives system operation.

Specifically, the transmission of data and control commands over the cyber space affects the operation of the physical processes. In turn, the data transmitted in the cyber space originate in measurements of the physical processes by sensors. That complex control feedback loop ties the cyber and physical domains of the system closely together to provide system service. The interactions among heterogeneous types of devices also imply that a slight disruption in the operation of devices in the system can cause a cascade of events that affect system service. Thus, the attack surface of the system spans a wide variety of devices. The system also receives external stimuli from human actors, and thus we examine the socio-technical domain of the system next.

Socio-Technical Domain. The capabilities and roles of the human actors in cyber-physical systems are more diversified than in traditional enterprise systems. While the human actors in enterprise systems predominantly interact with the cyber domain of the system, a human actor in a cyber-physical system interacts with both the physical and cyber domains of the system.

In particular, a human actor can control network hosts, including their software, data, and network packets. A human actor can also interact with the physical equipment directly within the buildings in which it is housed and affect the physical processes. Because of the control feedback loop that links the physical and cyber domains of the system, both the physical and cyber actions have a rippling effect on those two domains, which can lead to major impacts on system operation. Thus, both the physical and cyber actions of the human actors are possible malicious threat vectors that can adversely affect the system operation and safety.

1.2 PROTECTING CYBER-PHYSICAL SYSTEMS

It is therefore important to understand how the physical state and the cyber state of the system are intertwined together, with the effects in one domain affecting the other because of the actions of human users. A cohesive approach to protecting cyber-physical systems then involves a close scrutiny of the cyber-physical and socio-technical domains. In this dissertation, we present a way to protect critical infrastructure systems by (1) assessing the safety of the system and (2) detecting malicious threats on the system.

1.2.1 System Assessment

Just as for other systems, such as enterprise and cloud systems, it is important to conduct an assessment of a critical infrastructure system to understand how it performs under malicious and non-malicious faults. However, critical infrastructure systems are unique in that system *safety* is of paramount importance. *Safety* is the "freedom from those conditions that can cause death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment" [4]. An unsafe critical infrastructure system can result in catastrophic physical effects in the real world, and therefore, it is important to analyze and protect such systems.

In particular, system administrators need to ensure that this system remains safe under malicious and non-malicious actions. To prove that the system is safe, models of the system are built and theoretically verified for compliance with the safety property.

Safety analysis has traditionally been focused on proving the correctness of physical processes and control under non-malicious faults. With the evolution of critical infrastructure systems to include cyber control and communication, safety analysis has changed to include proving of the correctness of software and application data under non-malicious faults. In

other words, safety analysis needs to take into account the cyber processes that affect the physical system.

With the larger attack surface of the cyber-physical system, malicious threats are also rapidly increasing in importance for system safety. Many researchers, domain experts, and government bodies have thus identified the need for cyber security in safety-critical systems [5, 6] and have proposed different approaches for merging safety analysis and security analysis [7–9]. It is essential to integrate security knowledge about malicious behavior into safety analysis to account for malicious faults or threats. The results of performing a safety analysis of a system under malicious threats can then inform the system administrator as to the sequence of malicious actions that will force the system into an unsafe state.

To understand what system defenses can be put into place to detect malicious actions, a security analysis is needed on top of the safety analysis. A security analysis models the adversary's decision-making process and allows the system administrator to compare the differing actions an adversary might take when different system defenses are in place. Thus, the system administrator will be able to decide what system defenses should be installed to protect the safety of the system.

Security analysis has traditionally focused on malicious attacks in the cyber domain of the system, which includes the networks and hosts. However, in cyber-physical systems, attacks on the physical domain of the system can have propagating effects on the cyber domain of the system and aid cyber-based attacks.

Therefore, it is important to consider the actions of a malicious attacker on both the cyber and physical processes and communications in conducting a safety and security analysis of a critical infrastructure system.

1.2.2 Malicious Threat Detection

In addition to conducting an analysis of the system design, it is important to implement detection mechanisms in the system to sieve out potential malicious threats that have bypassed the security and safety perimeter. Most of the existing detection mechanisms presented in research publications and implemented in tools can be grouped into three main categories: misuse-based, behavior-based, and specification-based [10, 11].

Both misuse-based and behavior-based detection mechanisms are common to other traditional networked systems. Misuse-based mechanisms model known bad system states, whereas behavior-based mechanisms model system states that are statistically deemed normal. While misuse-based detection mechanisms are unable to detect unknown bad system states, behavior-based mechanisms are able to detect bad system states that are outside

the statistically normal model. However, behavior-based mechanisms suffer from high false positive rates, since the derived model relies on a training set of normal data.

All of those detection mechanisms typically utilize data from host machines and network connections to identify threats to the system. In other words, the detection mechanisms focus on threats involving the cyber domain of the system. However, they do not take into account the physical behavior of the system and the effect of human actors' actions on the system.

The third class of detection mechanisms, those that are specification-based, is more tailored to cyber-physical systems. Since the operation of cyber-physical systems is driven by physical processes and control feedback loops, we can construct well-defined models of a system's behavior based on laws of physics. Deviations from the model would then be considered potential malicious behavior, since they could directly affect the system operation. Unlike behavior-based mechanisms that sift through data logs to model a system's normal behavior, specification-based mechanisms derive models from formal specifications of the system that represent legitimate system operation. Therefore, specification-based mechanisms complement behavior-based mechanisms by catching the effects of malicious actions that appear in the system.

1.3 CYBER, PHYSICAL, AND SOCIO-TECHNICAL MODELS FOR SYSTEM PROTECTION

In the previous section, we described the different domains of cyber-physical systems, particularly, critical infrastructure systems. We showed that the interactions among those domains drive system operation, and thus are essential to consider in system assessment and malicious threat detection.

Therefore, in this dissertation, we propose to use models of the cyber-physical relationships among system components, and socio-technical interactions between the user and the critical infrastructure system, to protect the system.

A model helps to structure expert knowledge about a given system in a way that enables algorithms to automatically explore the consequences of certain behavior. It also aids the system practitioner in providing a more intuitive view of the system state for further analysis. While most models developed for critical infrastructure systems focus on representing either cyber control and communication or physical processes, we propose the need for models that represent a combination of the cyber, physical, and human domains of the system.

As shown and discussed by various members of the research community [12], it is imperative to analyze not just one domain of the system (cyber, physical, or human) but also the

interactions among them in the system. By considering those cross-domain interactions, we gain a more comprehensive understanding of the system behavior.

More precisely, it is important to understand the interactions among the entities in the system so that we can understand how the effect of an action performed on one entity can manifest across the system's physical and cyber domains, thus affecting system operation.

Thus, we model the system as a collection of topologically interconnected components in which the connections represent physical, cyber, cyber-physical, or socio-technical relations. Specifically, we model both the cyber network and physical equipment states of the system, as well as the interactions among those states. We also model a malicious human actor that physically moves around a building and accesses the equipment within it, and tampers with network messages in the system. We model those physical and cyber actions of the human actor on top of the existing system model to analyze the impact of those actions. By building a model of the system and its human actors, we can formally reason about the cyber-physical interactions among human actors and the system.

In this dissertation, we will illustrate our modeling approach on a critical infrastructure system, namely the railway transportation system that we describe below.

1.4 CRITICAL INFRASTRUCTURE CASE STUDY

While we intend for our work presented in this dissertation to apply to general critical infrastructure systems, we focus on transportation systems so as to provide concrete results and insights. In addition, the selected system possesses properties that are similar to those of other critical infrastructure systems and, thus, are a good representative set.

Transportation systems are part of a nation's critical infrastructure systems. Ranging from railway systems to bus networks, these transportation systems aim to move people or objects across geographical space. In particular, we consider a railway transit system, which is an important component of a nation's transportation system.

Over the years, railway systems have started to integrate cyber technologies and infrastructure to improve system efficiency. Although the quality of system service has increased as a result, the dependence on cyber infrastructure also increases the system's susceptibility to attacks. For example, in August 2016, there was a five-day series of service disruptions due to intermittent signal interference in our studied railway system. When it reappeared again in November 2016, data scientists traced the issue back to the failure of signaling hardware on a single train [13–15]. The failure interrupted signaling communications on other trains that were close to that train, triggering safety features to activate emergency braking.

Although the signal interference was not the result of an attack, the incident showcased how the failure of one cyber component can affect an entire system.

The impact of an attack on a railway system can be severe, ranging from loss of service and station blackouts to derailment. For example, a Polish teenager once rewired a remote control to communicate with wireless switch junctions in a railway system, causing derailment of a train and injury of twelve people [16]. Since the track was accessible to the public, the attack was easily performed. The potential loss of revenue and human life is motivating the pursuit of greater physical and cyber security for such systems. In particular, the insider threat is of the utmost importance, as illustrated by a 2006 case in which two traffic engineers hacked into a Los Angeles signal system, causing a major traffic disruption [17].

The railway system is a distributed system with cyber controllers and sensors that automate the movement of physical trains over a railway line. System efficiency can be measured by *route capacity*, the number of trains that can be driven on the rail lines while maintaining safety conditions, which is determined by an acceptable distance between trains, or *headway*.

The signaling system involves detecting the location of trains and communicating this information back to nearby trains to inform decisions about train movement. Historically, the location of trains was detected using track circuit signaling. Track circuits, which are electrical devices located intermittently at fixed-block intervals on the rail tracks, were used to detect the presence of a train on the rail segment. That information was then relayed to the train via electromagnetic signals. Thus, the route capacity was restricted by the fixed-block configuration that could only relay discrete locations based on the position of the track circuits.

Over the years, the signaling system has evolved from track circuit signaling to *Communication-Based Train Control* (CBTC) systems, which allow for continuous monitoring of a train's location. Information is transmitted via radio signals instead of electromagnetic signals, allowing for richer and larger amounts of data to be transferred. Processing units on the trainborne system use the transmitted location information from beacons installed on the track in tandem with other sensor data to calculate the exact location of the train. So the moving block configuration of CBTC systems removes the restriction of the fixed-block configuration, thus allowing trains to be closer to each other. Therefore, route capacity has vastly improved because of the decrease in headways while maintaining safety conditions.

Many other subsystems of the railway system, such as the traction power and environmental systems, are also increasingly automated. Aided by networked communications, a centralized control center can monitor and control those mechanical devices in real-time. The data collected by the control center are also visualized and presented to the system staff members, allowing them to control the system remotely. The inherent complexity of

the system increases the difficulty of system analyses because the change in behavior of one component can cause a cascade of events, leading to consequences in the real world. For example, a malfunction of the water pumping system in the studied railway system caused the tunnels to be flooded with water, resulting in a close to 21-hour service disruption [18, 19]. Thus, targeting of subsystems that are not directly related to train movement can have an impact on system service. Therefore, in our dissertation, we build a model of the system and its users so that we can formally reason about the cyber-physical interactions among devices and the humans acting upon them.

1.5 DISSERTATION CONTRIBUTIONS

The general contribution of our dissertation is the combination of heterogeneous data to build a comprehensive understanding of system state that facilitates system design analysis and dynamic security solutions. Specifically, our dissertation statement is the following:

> *The protection of critical cyber-physical infrastructure systems against malicious human-initiated cyber and physical actions can be achieved by a novel integration of security analysis, safety analysis, and system defense that uses cyber, physical, and socio-technical models.*

Our specific contributions in this dissertation are the following:

- We introduced a novel ontology describing the cyber and physical system components of the system, and the effects of cyber and physical actions made by a human actor, for automatic generation of attack graphs.
- We developed a set of strategic transformations on state automaton models that represent the cyber actions of a human adversary on a cyber-physical system in order to verify safety properties.
- We conducted the first large-scale analysis of human movement in an indoor setting for detection of malicious physical movements.
- We performed the first real-life study of identification of tailgating through use of cyber data collected by building access control systems and physical sign-in log data collected from railway stations.

The dissertation is divided into two main components: system assessment of safety and security properties (Chapters 2 and 3), and cyber-physical threat detection (Chapters 4 and 5).

Security Analysis: In Chapter 2, we discuss a security analysis of railway stations that we conducted using an automatically generated attack execution graph, which is a variant of an attack graph. We define a general ontology that represents knowledge about the cyber and physical components in a system, the relationships among them, and the effects of cyber and physical actions made by a human actor. We developed a model of the railway station by using concepts from the ontology, and input that model into the ADVISE [20] tool to generate an attack execution graph. The results of a security analysis of the attack execution graph suggest potential defense mechanisms that can be put into place. In particular, our analysis shows that identification of malicious physical movement is an effective mechanism for securing a system.

Safety Analysis: In Chapter 3, we discuss a safety analysis of a railway system that we conducted using a state automaton model that represents both the cyber and physical states of the system. We defined a set of transformations on the state automaton model to represent the cyber actions taken by an human adversary actor. Then we built a state automaton model of the railway signaling system and used model-checking techniques on that model to verify the safety property of the system under different sets of adversary capabilities. The results of that safety analysis inform us about the potential threat vectors and the effects on system safety. We also evaluated several safety countermeasures to determine how much they alleviate the potential threats. Finally, by enhancing our ADVISE model of the railway station from Chapter 2 to include the potential threat vectors uncovered in Chapter 3, we analyze which of the threat vectors are likely to be exploited. The results of that analysis suggest that adversaries are incentivized to use physical and social means rather than cyber means to attack a system.

Physical Threat Detection: In Chapters 4 and 5, we analyze the physical movement behavior of the human actors in a system to detect potentially anomalous movement that could signify precursor behavior, such as physical reconnaissance or physical intrusions by insiders.

In Chapter 4, we develop socio-technical models that represent normal patterns of movement behavior of human actors through a building. We built those models based on historic cyber data collected from building access control systems. Then, we used those models to screen physical card accesses in real-time to detect anomalous movement patterns that might indicate malicious intent. We evaluated our approach by injecting random movement sequences into a real-life physical card access dataset and

found that our approach can detect abnormal movement with increasing accuracy as the length of the abnormal movement sequence increases.

In Chapter 5, we address the limitations of using cyber data collected from socio-technical systems, i.e., building access control systems, for analyses. Specifically, we show from cross-examinations with other physical data sources and an observation study of real-life movement behavior that the social phenomenon of tailgating (i.e., following closely behind an authorized person) is not detected by building access control systems. We developed several approaches to infer tailgating from physical card access data by using physical constraints about human movement. We modeled the physical location of human actors in the system and updated that model for every event in the physical card access data. In addition, we leveraged physical data sources collected from the railway system to obtain a more accurate model of the physical location and space occupancy. Our results show that we can identify certain scenarios of tailgating behavior by using physical constraints and physical data sources to complement cyber data.

In Chapter 5, we also analyze how our real-time approach from Chapter 4 and post-incident analyses from Chapter 5 affect the adversary's choices in attacking a system. We describe an enhancement of our ADVISE model from Chapter 3 that includes the detection probabilities of our approaches. The results of our analysis suggest that adversaries prefer to launch cyber campaigns against a system in a manner that avoids detection by the defender, despite the amount of time and effort required. In other words, our physical threat detection approaches incentivized the adversary to shift to a more costly, time-consuming attack strategy. Thus, our physical threat detection approaches are useful in increasing the level of security of a system.

Finally, we conclude the dissertation in Chapter 6 with a discussion of the generality of our approach, lessons learned in the course of our research, and avenues of future work to explore.

CHAPTER 2: CYBER-PHYSICAL SYSTEM ASSESSMENT: SECURITY RISK ANALYSIS

System administrators often conduct security risk analyses of their cyber-physical systems to determine which components of the system are at risk of being targeted by an adversary and decide what system defenses should be installed to mitigate such risks. Most security risk analyses have focused on modeling the cyber domain of such systems by enumerating the possible attacks that could be used to compromise the system and cause a cascading impact in the physical world. However, physical attacks that target devices in cyber-physical systems are just as concerning. The Metcalf power station sniper attack [21, 22], where gunmen shot the transformers in a power station, is a prime example of such a physical attack. Further, physical attacks coupled with synchronized cyber efforts are commonly overlooked by security risk analyses. In this chapter, we focus on modeling of physical attacks and their consequences, and apply our approach to a railway station case study to provide insights into the security of the system.

2.1 BACKGROUND AND MOTIVATION

In a recent survey related to industrial control systems [23], only 20% of respondents said that they had "sufficient visibility into their organization's attack surface." That is an unacceptable situation; system practitioners need a sound, scientific approach they can use to evaluate the attack surfaces of their systems. For systems that are already operational, red-teaming or penetration tests are an expensive but potentially useful option. For systems that are in the design phase, system practitioners often rely on construction of attack graphs that describe the possible ways an adversary could attack the system.

We propose to use the *ADversary VIew Security Evaluation* (ADVISE) meta modeling approach [24], which uses a variant of attack graphs to perform a formal, repeatable, and auditable analysis of system designs. Attack graphs are used to analyze a system by enumerating all the possible attack steps that an adversary could take to achieve a goal [25, 26]. However, traditional attack graphs do not evaluate useful probabilistic metrics like the most common attack vector for a given adversary, expected time to complete an attack, and expected cost to the defender. The ADVISE approach builds a richer attack graph, termed an *attack execution graph* (AEG), which models additional details about attacks in order to facilitate a more comprehensive and quantitative analysis of an adversary's attack path.

The process of building such attack graphs, however, requires a lot of manual effort by an expert practitioner. The practitioner needs to enumerate the possible attack steps for each

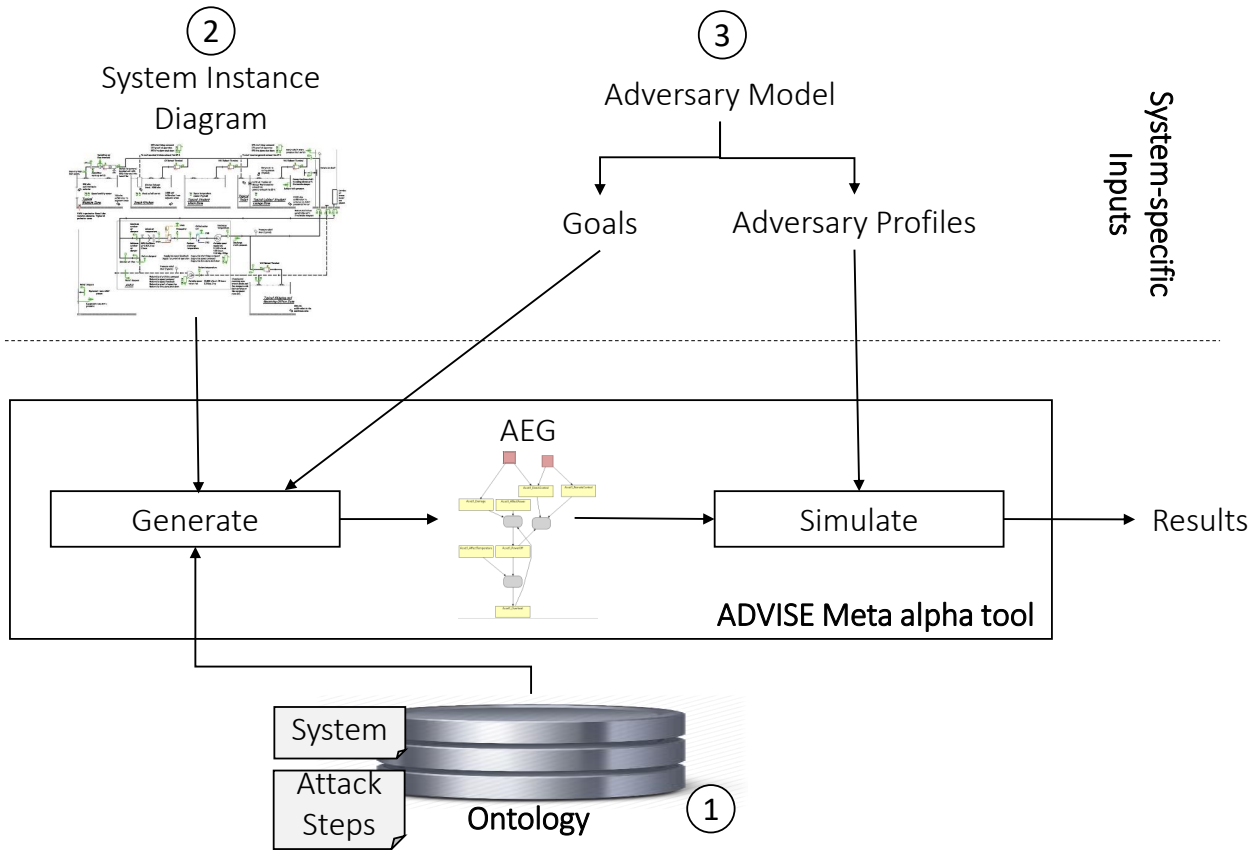


Figure 2.1: The general modeling approach. The numbered bubbles represent the order in which we describe the different inputs to the ADVISE Meta tool.

system component and connect those attack steps together. To reduce the practitioner's effort, the ADVISE meta modeling approach separates the knowledge of the system from knowledge of the attack steps, allowing the actual construction of the attack execution graph to be automated.

The ADVISE meta modeling approach relies on an ontology that represents a high-level system model that uses familiar atomic elements and semantic relationships among them. The ontology serves as a knowledge base that describes the system entities and the definitions of attacks. Unlike vulnerability databases, ontologies allow practitioners to automatically derive where and how attacks may be applicable to a given system. The ontology allows the practitioner to specify attack steps just once during the construction of the ontology; subsequently, the only changes that a practitioner needs to make are updates to the system model. The attack graph is automatically generated through reasoning on the system model instance that is based on the ontology [24].

The central component of our approach uses the ADVISE Meta tool, which takes in a

number of user-specified inputs, and outputs an AEG. Figure 2.1 illustrates that process. In this chapter, we describe how we constructed an ontology that models (1) the cyber-physical system entities and relationships, and (2) the attack steps that can be performed on those entities. This ontology is the first step towards a formal definition of a library of physical attacks that can be reused by other system case studies. Next, we discuss how we constructed a system instance diagram that describes a specific case study in terms of the types of entities and relationships that are present in the cyber-physical architecture of the studied system. Those types of entities and relationships are given by the ontology. We used the ADVISE Meta tool to reason on that system instance diagram and automatically generate an AEG [20,27] that leads to the user-specified goals. We also specified information about the threat models and quantitative metrics that we want to calculate. Given that information, the tool simulates an adversary's decision-making by using a discrete-event, game-theory-based algorithm and outputs the calculated metrics and the chosen attack path.

We have applied our modeling approach to a case study of a railway station. In particular, we have evaluated the effects of physical attacks on a railway transit system. We specified different threat models and used the ADVISE meta model tool to simulate the different adversaries' decision-making processes. Here, we analyze the attack paths that are output from the tool to obtain insights regarding the system security. From the simulation results, we also identify (1) which adversaries pose the biggest threat to the system, (2) which attacks are most likely to happen, and (3) what additional defenses should be deployed to identify and prevent certain attacks.

2.2 ONTOLOGY: SYSTEM MODEL

The ontology consists of two components: (1) a representation of cyber-physical systems, and (2) a library of attack steps that apply to that representation. We first start with a general conceptual understanding of the systems that we want to model; that will then drive the creation of classes and relations in the ontology.

Conceptually, we describe the system in terms of three non-disjoint domains: *physical*, *cyber*, and *information*. We distinguish among the three domains because they represent the levels at which human cognition perceives information [28]. The physical domain consists of devices and spaces. Devices are located in spaces. The cyber domain consists of hosts and the network topology. Correspondingly, the hosts are devices (physical domain) that contain control software that transmits and receives digital data, and the network topology consists of cables connecting hosts that allow the transmission of bits. Lastly, the information domain consists of digital data such as control data and logs. Those digital assets are represented

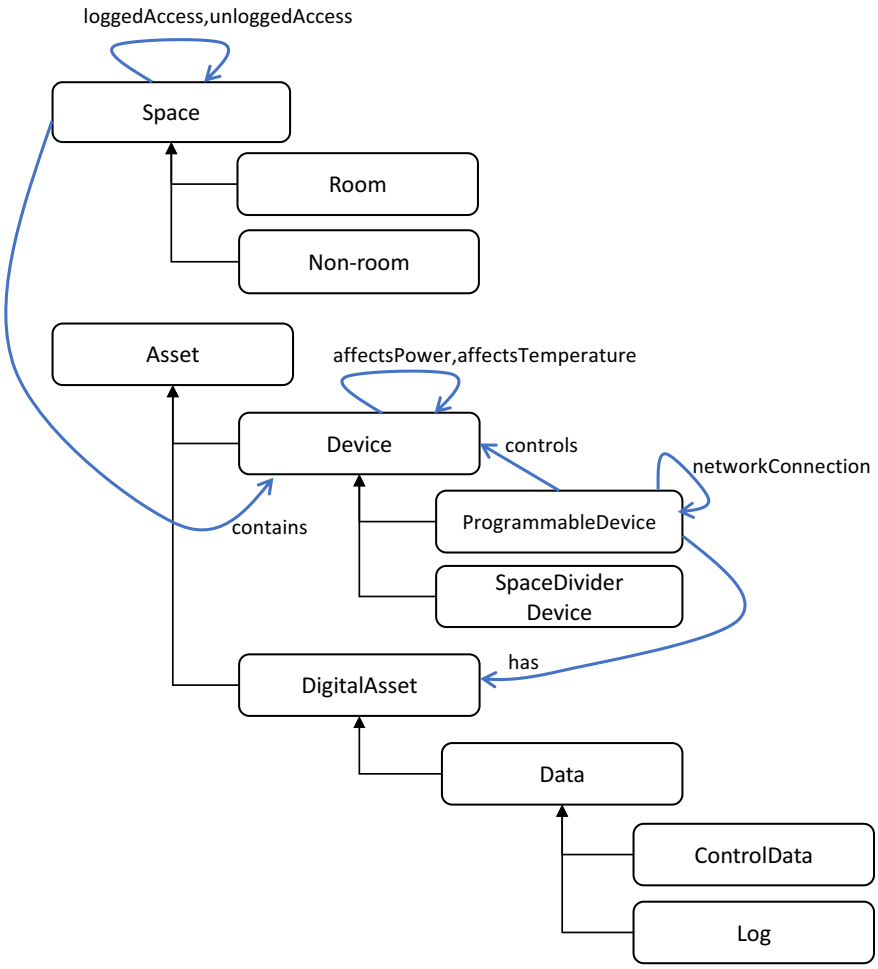


Figure 2.2: Our constructed ontology. Black boxes are component classes. Arcs with filled arrowheads represent inheritance, and labeled arcs are relationships.

as bits of information. By describing the system using the three domains, we can define actions (attack steps) that affect the system at different levels of abstraction. We can also drill down into low-level details about the effects of those actions.

We have translated our general conceptual understanding into the ontology, which is shown graphically in Figure 2.2. The ontology defines component classes and kinds of relations that are definable between instances of the classes. The component classes use inheritance to describe specialization of types. The ontology's inheritance allows us to automatically infer information regarding child classes through deductive reasoning. More specifically, child classes inherit relationships and properties (attacks) from their parent classes [29].

The ontology describes the entities and their relationships across the thee domains: physical, cyber, and information. We translated the two types of entities in the physical domain, device and space, into the parent classes **Device** and **Space**, respectively. The class **Space**

is further divided into two child classes, **Room** and **Non-room** (e.g., corridors). Compared to a **Non-room**, a **Room** houses more critical kinds of assets and is typically the destination of adversaries.

We define a class **Asset** that represents both the physical and cyber entities in the system that a practitioner seeks to defend. In particular, the class **Asset** has child classes **Device** and **DigitalAsset**.

The class **Device** also has child classes that represent the specialized capability of a device. The **ProgrammableDevice** child class contains software that can control the behavior of other assets (e.g., a host with SCADA software or a PLC); this class represents the cyber hosts. The **SpaceDividerDevice** child class represents an asset that can divide two spaces (e.g., a roller shutter or door). The class **DigitalAsset** represents the information domain. It has a child class **Data**, which is further divided into the two child classes **ControlData** and **Log**.[1]

There are two different relations between **Space** instances based on how a person can move between the spaces. If a card reader is installed on the door bordering the two spaces, a person needs to have possession of an access card in order to move from one space to the other, and the person's movement is also logged by the system. The *loggedAccess* relationship from a **Space** to a **Space** captures that situation. Otherwise, if no card reader is present, a person can move without restriction between the two spaces with no access card, and the relation is called *unloggedAccess*. If there is a **SpaceDividerDevice** separating the two **Space** instances, then a third relation, *unloggedDividerAccess*, from the two **Space** instances to the **SpaceDividerDevice** is used. The **Space** class is also related to the **Device** class. Specifically, a **Space** *contains* **Device**s.

There are a few relations between **Device**s, based on their functionality. In particular, a **ProgrammableDevice** *controls* other **Device**s programmatically. For example, a host machine with the proper control software can send commands to the *Programmable Logic Controllers* (PLCs) that it controls, and the PLCs in turn control the devices. Depending on the type of device, the results of controlling a device will affect the power, temperature, or movement of other devices. Specifically, an **Asset** *affectsPower* or *affectsTemperature* of another **Asset**. In addition to controlling other assets, a **ProgrammableDevice** also *controlsMovementToken* of a **Space** by adding or removing permissions of an access card to access that space.

The **ProgrammableDevice** also has *networkConnection*s with other **ProgrammableDevice**s in the cyber domain. In addition, a **ProgrammableDevice** *has* **DigitalAsset**s on

[1]There can be other subclasses of **DigitalAsset**, for example, to distinguish between data and software programs. However, we leave that classification for future work.

it whose values may affect the running of the system devices.

Finally, we define a *Subsystem* attribute for both **Device**s and **Space**s. This attribute identifies the subsystem to which a device or space belongs, and thus allows us to restrict access to the physical device or space. We also define state variables for **Device**s that represent the *Power*, *Temperature*, and *controlSetting* of the device. Adversary actions will affect the values of those state variables.

2.3 ONTOLOGY: ATTACK STEPS

The ontology consists of generic attack steps that can be applied to the classes. By defining a library of attack steps, we use the ontology to reason on a system instance diagram and automatically enumerate which attack steps can be applied to the system entities. In particular, we use the ontology's multiple inheritance feature to derive all the entities that are vulnerable to a specific attack.

We can illustrate the multiple inheritance feature with a small example. In our ontology in Figure 2.2, we can define an attack **A** that is applicable to any physical device. We model that by attaching **A** to the class **Device**. From the inheritance structure, that implies that **A** is applicable to all subclasses of **Device**, including **ProgrammableDevice** (or cyber hosts). That simplifies the job of creating an attack graph, since the practitioner does not need to recall that a host is also a **Device** and thus susceptible to attacks on physical devices. The ontology thus allows us to generate a complete AEG with respect to the defined attack steps, removing the chance for manual human error.

In Table 2.1, we define attack steps that can be applied at the level of abstraction corresponding to each of the three domains that we defined earlier. The preconditions and postconditions associated with each attack step are listed in Table 2.2.

Some of the attack steps (**Overheat**, **PowerOff**, **AffectsPower**, and **AffectsTemperature**) are not actual attacks performed by an adversary, but instead model the physical consequences of certain actions.

In particular, when the temperature of a device is not properly controlled and increases to a high value, the **Overheat** attack step is triggered. The result of that step is that power to the device is cut, i.e., the state variable *power* is set to a value of zero. When the power to a device is cut, the **PowerOff** attack step is triggered; that removes the *controlSetting* of the device, since it is no longer able to control other devices that may depend on it.

The effect of controlling a device and changing its *controlSetting* is then reflected in the other devices under that device's influence. Specifically, if a device d_1 **affectsPower** of another device d_2, then changing the *controlSetting* of d_1 will trigger the **AffectsPower**

attack step, which then changes the *power* state variable of the affected device d_2. Similarly, the **AffectsTemperature** attack step propagates changes to the *temperature* of the affected device.

Table 2.1: List of attack steps defined in ontology.

Domain	Attack Step	Ontology Class
Physical	Move	Space
	Damage	Device
	DirectControl	Device
	Overheat	Device
	PowerOff	Device
	AffectsPower	Device
	AffectsTemperature	Device
Cyber	Login	ProgrammableDevice
	RemoteControl	Device
	NetworkMove	ProgrammableDevice
Information	DelMovementToken	Data
	AddMovementToken	Data

Physical Domain: In the physical domain, there are three different types of attack steps. **Move** represents an adversary's movement from one **Space** to another. If the access is logged (i.e., the *loggedaccess* relation connects the two spaces), the attack step will succeed only if the adversary possesses an access card with the appropriate permission (represented by the appropriate **Space_X** access element, where **X** corresponds to the *Subsystem* attribute of the space). The access will, however, be detected. If the access is unlogged (i.e., the *unloggedaccess* relation connects the two spaces), the attack step always succeeds, because the adversary can move between the two spaces without restriction, and this attack is not detected.

An adversary can also perform an **Action** on a device. There are two main types of actions: **Damage** and **DirectControl**. Both are associated with the **Device** class. **Damage** involves physical destruction of the device. The result of the **Damage** attack step is that the power to the device is lost. **DirectControl** requires the adversary to have physical access to the cabinet housing the physical device in order to manually control the settings of the device. The result of the attack step **DirectControl** changes the *controlSetting* associated with the device. Depending on the type of device (e.g., breaker, lights), the effect of changing the *controlSetting* can range from turning the device off to controlling the device to achieve a specific setting (e.g., dimming the lights by 30%).

Table 2.2: Preconditions and postconditions associated with attack steps

Attack Step	Precondition	Postcondition
Move	Access: Space_X Space: PhysicalPresence	Space: PhysicalPresence Access: AssetAccess
Damage	Access: AssetAccess	Device: Power
DirectControl	Access: AssetAccess	Device: controlSetting
Overheat	Device: Temperature	Device: Power
PowerOff	Device: Power	Device: controlSetting
AffectsPower	Device: controlSetting	Device: Power
AffectsTemperature	Device: controlSetting	Device: Temperature
Login	Access: AssetAccess Access: Access_X	Access: RemoteToken ProgrammableDevice: software
RemoteControl	Access: RemoteToken Access: Access_X ProgrammableDevice: software	Device: controlSetting
NetworkMove	Access: Access_X ProgrammableDevice: software	ProgrammableDevice: software
DelMovementToken	Access: Access_X ProgrammableDevice: software	Access: Space_X
AddMovementToken	Access: Access_X ProgrammableDevice: software	Access: Space_X

Cyber Domain: In the cyber domain, there are two possible attack steps: **Login** and **RemoteControl**. The **Login** action is associated only with the **ProgrammableDevice** class, whereas the **RemoteControl** action is associated with the parent **Device** class. Much like **DirectControl**, **RemoteControl** changes the *controlSetting* associated with the device by using the control software on a **ProgrammableDevice** to manipulate the settings of the device. An adversary can also **Login** into a **ProgrammableDevice**. The result of the **Login** attack step is that the adversary is allowed to gain access to the control software. If the **ProgrammableDevice** has *networkConnection*s to other **ProgrammableDevice**s, the adversary can perform the **NetworkMove** attack step to gain access to the other networked hosts. Depending on the system role that is used to perform the **Login** action (which are represented by the appropriate **Asset_X** access element, where **X** corresponds to the *Subsystem* attribute of the device), the adversary can then perform the **RemoteControl** action for devices that the adversary has permission to control

Information Domain: In the information domain, we consider two more specific attack steps that are tailored to the case study. In particular, they involve either removing permissions from or adding them to the adversary's arsenal. These attack steps are associated

with the **Data** class. If an adversary gains access to the BAC server, he or she can use the **RemoveMovementToken** attack step to remove all access card permissions (all **Space_X** access elements, where **X** is any of the three subsystems) and locks the roller shutters so that people are unable to move into or out of the station. The **AddMovementToken** attack step grants an adversary all access card permissions and thus allows him or her to move through all rooms in the station.

2.4 CASE STUDY: RAILWAY STATION

In this section, we describe the case study of the railway transit system that we study in this dissertation. Physical security is an important concern for critical infrastructure systems such as industrial control facilities. In this chapter, we focus on the physical security of a railway station. The railway station is a single building with one or more railway lines running through it. The railway system we studied is located underground, and the railway lines are housed in a tunnel. We will reuse this particular railway station in Chapters 4 and 5.

The railway station contains many rooms that house the equipment necessary for running the station and its portion of the railway track. Each room serves a specific function, and there are multiple rooms that share the same function. These rooms are not accessible to the general public. Staff members possess access cards that have been granted permissions by the *Building Access Control (BAC)* server, thus allowing them to move through parts of the station according to their system role (e.g., maintenance staff, station operator).

The devices control the movement of trains (e.g., signaling, traction power) and people (e.g., escalators, roller shutters). Other devices control the environment of the station and track (e.g., fan, water chiller). The environment of a space also affects the functioning of devices and movement of people. For example, high temperatures can cause a device to overheat and trip, or cause a fire that will result in evacuation of commuters.

2.4.1 System Instance Diagram

We constructed a *system instance diagram* that represents our railway station case study. The system architecture and the relations among the devices are modeled using the constructs we defined in our ontology. We chose the following set of representative rooms:

- **PSC:** Houses cyber hosts that station operators use to monitor and control the devices in the station.

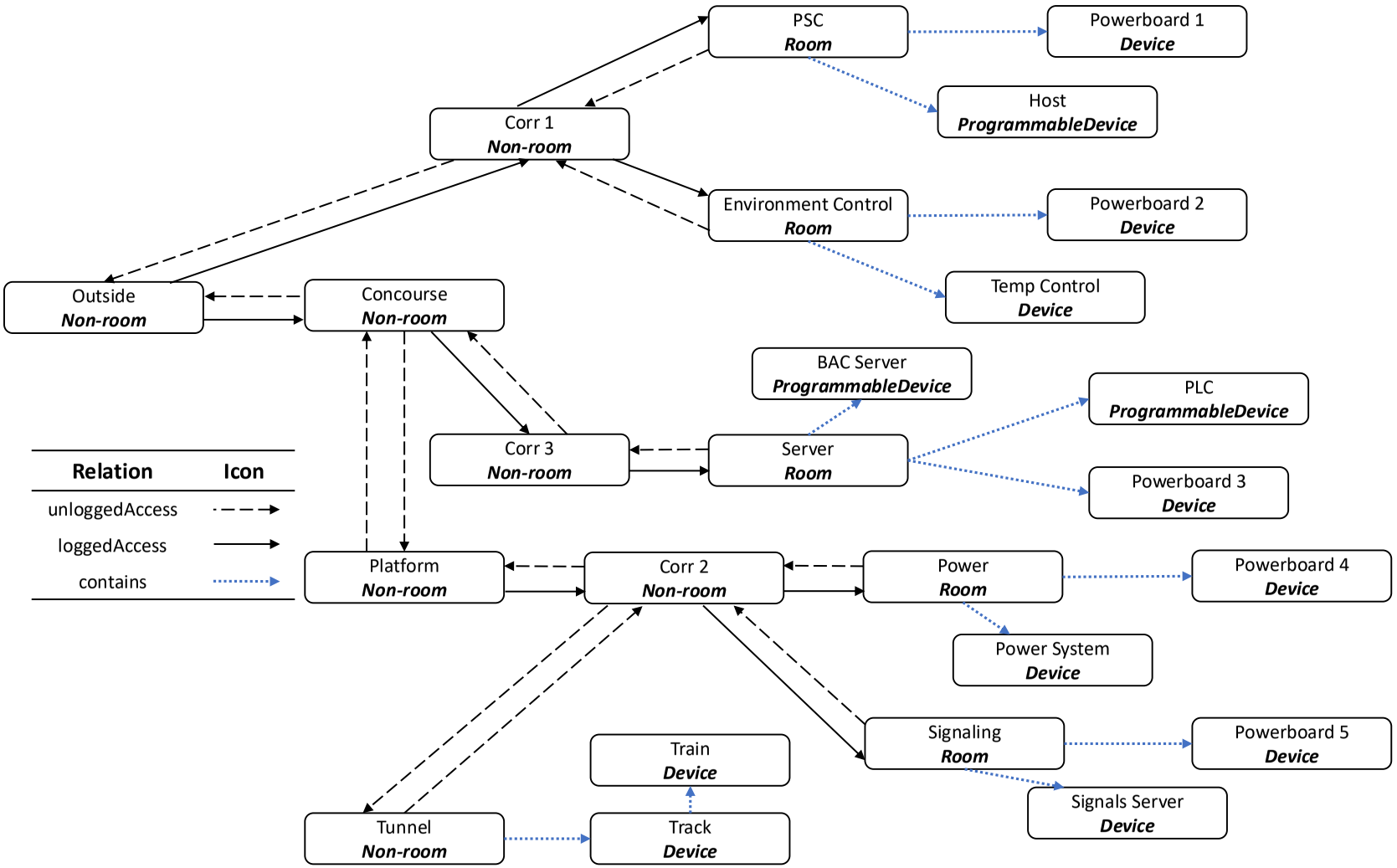


Figure 2.3: The physical station architecture view of the system instance diagram. The boxes represent the entities in the railway station. Each box has a unique name and belongs to a class in the ontology.

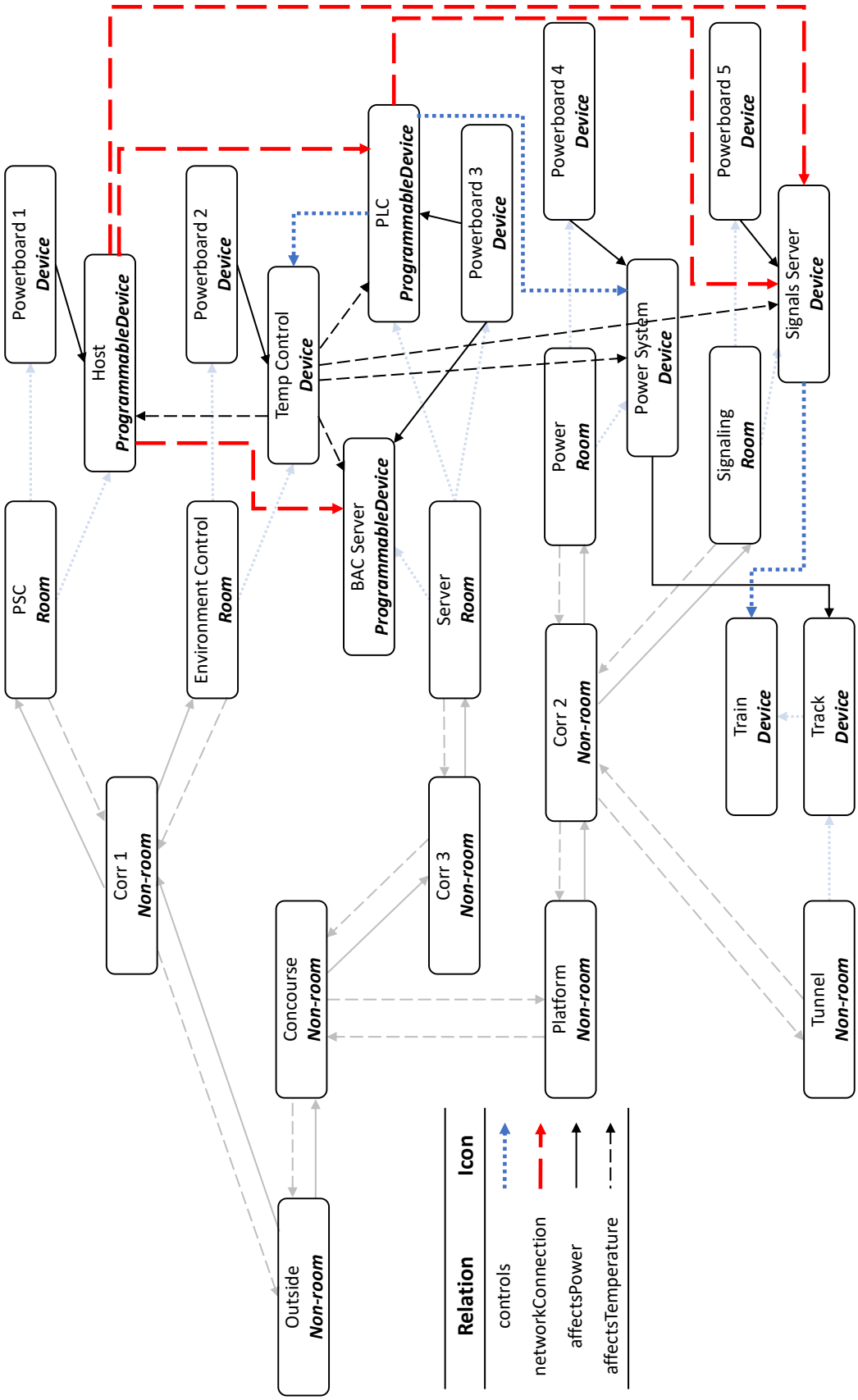


Figure 2.4: The relations among the devices in the physical station.

- **Server Room:** Houses the *Programmable Logic Controller*s (PLCs) that control the operation of other devices, and the *Building Access Control* (BAC) server that manages accesses to the doors and roller shutters.
- **Traction Power (Power Room):** Houses the equipment that controls the power to the track.
- **Signaling (Signal Room):** Houses the servers that communicate with trains on the track to control their movement as they pull into and leave the station.
- **Water Chiller (Environment Control Room):** Houses equipment that controls the temperature in the station.

The system instance diagram is depicted in Figures 2.3 and 2.4. The layout of the station and the devices within the rooms is shown in Figure 2.3, and the relationships among the devices are shown in Figure 2.4.

2.5 ADVERSARY PROFILES AND GOALS

We now define the adversary profiles and their goals. The two main assets that an adversary would target are the trains and the commuters within a station. For example, affecting the movement of trains includes delaying trains to affect service revenue, and acceleration of trains to endanger human lives. Targeting the movement of people can cause mass evacuation and chaos, which can lead to significant injuries caused by a panicking crowd. Affecting the system environment might consist of creating fire or smoke within the station or tunnel.

Within the same threat model, we can define different types of adversary profiles based on their skills, their system access, and their preferences for remaining undetected. Each attack step requires some preconditions to be satisfied before it can be attempted, e.g., the adversary's skill in an activity, or the adversary's access permissions to a system component. So depending on the adversary profile, some of the attack steps that we defined earlier can or cannot be attempted. We now define different types of adversaries who differ in system access and skill level.

The accesses that an adversary may have are (1) permissions to perform the **Move** attack step to enter a **Space**, and (2) permissions to perform the **DirectControl**, **RemoteControl**, **Login**, or **NetworkMove** attack steps to control an **Asset**. We define three types of adversary profiles: an **Outsider** adversary with skills to break door access locks, a **PrivilegedInsider** adversary who has gained access to all types of legitimate access cards, and an **Insider** adversary. Table 2.3 shows the accesses that those adversaries possess.

Table 2.3: The access elements possessed by different adversary profiles at the start of the simulation.

		Outsider	Privileged Insider	Insider						
				Cleaner	Station Op	Maintenance Staff				
						Env Ctrl	Power	Access Ctrl	Server	Signals
Physical Access	Corr 1		✓	✓	✓	✓				
	PSC		✓	✓	✓					
	Env Ctrl Rm		✓	✓	✓	✓				
	Corr 3		✓	✓	✓			✓	✓	
	Server Rm		✓	✓	✓			✓	✓	
	Corr 2		✓	✓	✓		✓			✓
	Power Rm		✓	✓	✓		✓			
	Signaling Rm		✓	✓	✓					✓
Asset Access	PB 1		✓		✓					
	Host		✓		✓					
	PB 2		✓			✓				
	Temp Control		✓		✓	✓				
	PB 3		✓						✓	
	PLC		✓		✓				✓	
	BAC Server		✓					✓		
	PB 4		✓				✓			
	Power System		✓		✓		✓			
	PB 5		✓							✓
	Signals Server		✓		✓					✓

We define four goals that the adversary wants to achieve: (1) stopping train movement (**TrainStop**), (2) locking the station exits (**LockExits**), (3) controlling the temperature in the station (**TempCtrlOff**), and (4) controlling the trains' movements (**TrainCollision**).

The first goal halts the movement of trains (`Train`) and thus disrupts system service. More specifically, it is represented by either switching the value of the *power* state variable of the `Track` **Device** from 1 (On) to 0 (Off), or switching the value of the *power* state variable of the `Signals Server` **Device** from 1 (On) to 0 (Off). The second goal affects the commuters' movements (`BAC Server`).[2] The third goal affects the station's environment (`Temp Control`) and is represented by switching the value of the *temperature* state variable of the `Temp Control` **Device** from 1 (On) to 0 (Off). Finally, the fourth goal affects system safety (`Train`) and can result in train collisions or derailment. It is represented by switching the value of the *controlSetting* state variable of the `Train` **Device** from 1 (On) to 0 (Off).

2.6 EXPERIMENT SETUP

We input our system instance diagram and adversary models into the ADVISE Meta tool, which used our defined ontology to reason on the inputs needed to generate an AEG. Figure 2.5 shows the generated AEG, and Figure 2.6 shows a portion of the AEG for an **Asset**. We describe the parameters of the model and discuss the results in the following subsections.

We used the tool to simulate the actions taken by an adversary profile and calculated metric values for that set of actions. The metrics we defined are (1) the total cost of performing attacks, (2) the number of attack steps needed to accomplish a goal, and (3) the fraction of attack steps that are likely to be detected. Those metrics describe the different trade-offs that an adversary will make when attacking the system. We analyzed the resulting metric values to obtain insights regarding (1) the possible sequence of actions taken by an adversary, (2) the assets that require further protection, (3) the user actions that require further monitoring, and (4) the set of adversaries that pose the highest risk to the system's security.

For each attack step, we defined (1) how much it costs the adversary to attempt the attack step and (2) the probability that the attack step will be detected by a defender. Those settings will impact the adversary's decision-making process when he or she is choosing among different attack paths.

The cost of an attack step is quantified in terms of the expert knowledge and physical effort

[2]Although these access elements refer to the adversary's access, we assume that removal of those elements would also remove the access of all other (benign) employees too.

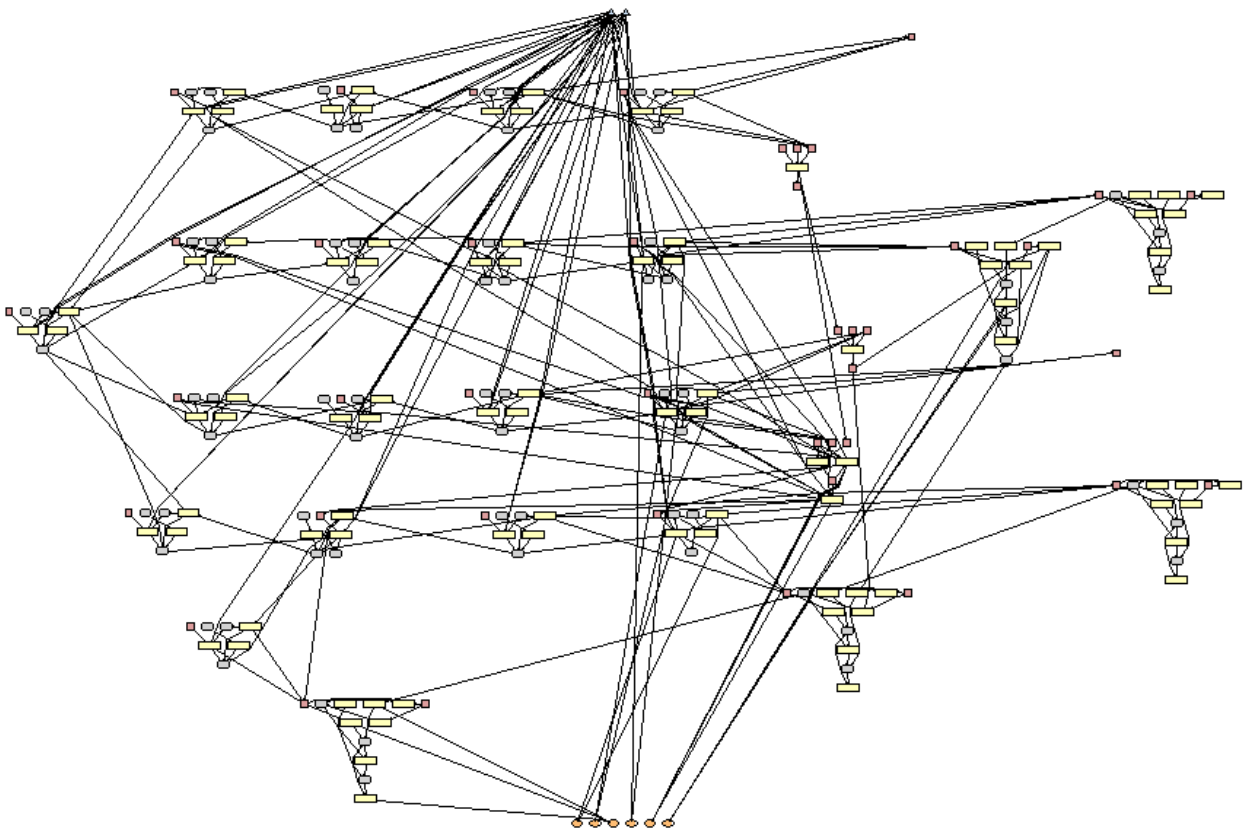


Figure 2.5: The full attack execution graph.

Table 2.4: The cost incurred by an adversary and the detection probability of each attack step. If the adversary has physical access to a device, damaging that device has a cost of 3; otherwise, damaging that device has a cost of 10.

	Move	Damage	DirectControl	RemoteControl	Login	AddMovement	DelMovement
Cost	1	3 or 10	8	7	3	5	5
Detection Prob	0	1	0	0	0	1	1

required to execute the attack. The cost is a relative score that is an integer from the range of 1 to 10, where 1 represents the least effort and 10 represents the most effort. The probability that an attack will be detected by a defender is a floating value that ranges from 0 to 1. We assume that as a preliminary defense, the defender can monitor the devices through CCTV cameras in the rooms, and can monitor the state of the commuters by direct observation. Based on our understanding of the system and possible attacks as well as discussions with

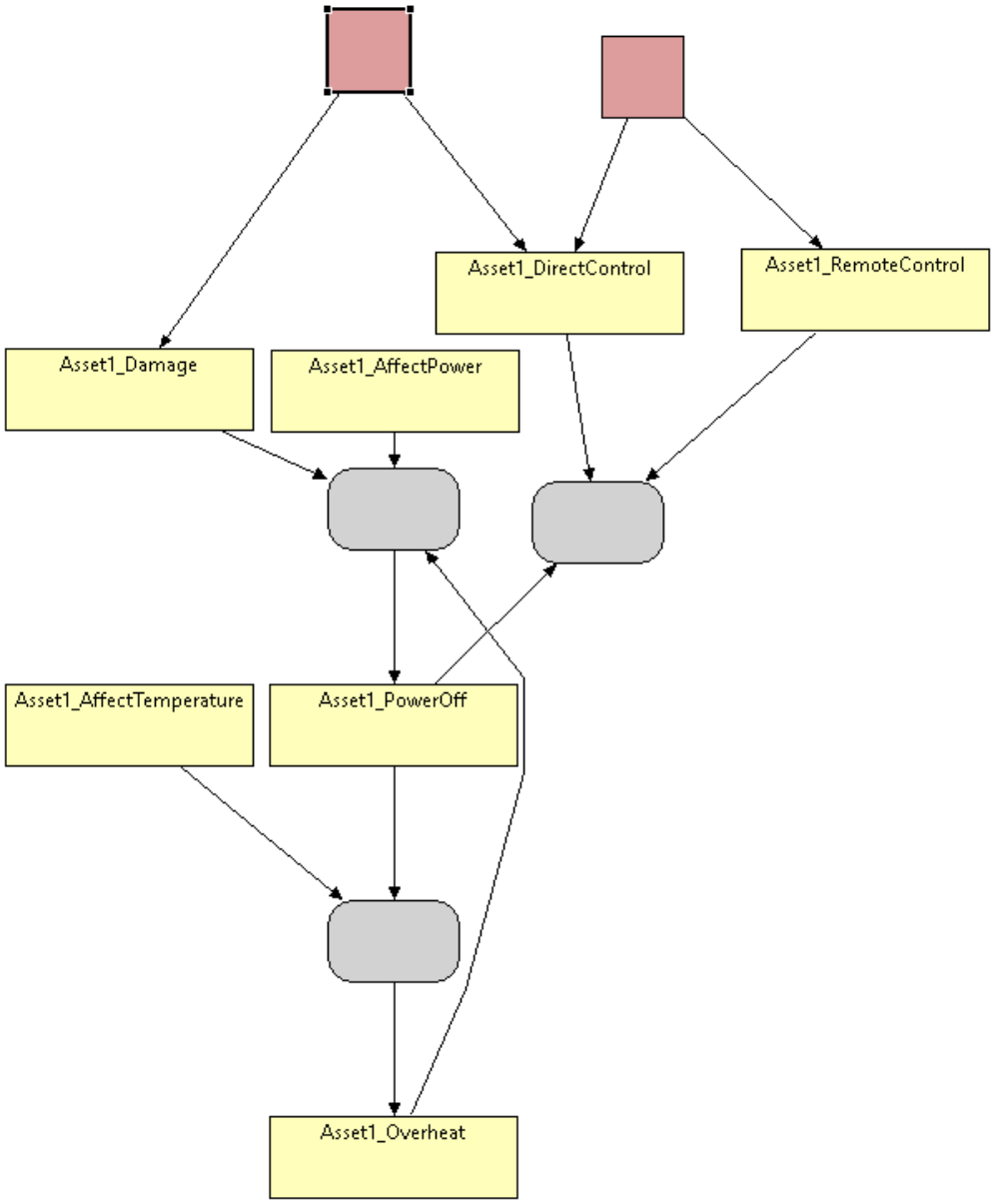


Figure 2.6: A snippet of the attack execution graph for an asset.

our railway system collaborators, we set the parameters as shown in Table 2.4.

We assigned the lowest cost to the **Move** action step, followed by **Login**. Moving through the spaces has the lowest cost, followed by logging into an asset or damaging an asset to which the adversary has physical access. After logging in, an adversary can remotely control devices. Doing so requires knowledge of how to operate the control software through the *Human Machine Interface* (HMI) (which abstracts out the details of the physical controls) of the host, so the cost of the **RemoteControl** action is higher than that of **Login**. On the other hand, directly controlling the asset (**DirectControl**) requires more knowledge

than remotely controlling the asset, since the adversary needs to know which component to control. Permissions-related attacks (**AddMovement**,**RemoveMovement**) also require intimate knowledge about the building's layout and permission rights, so their costs are moderately high. Finally, the **Damage** attack has the highest cost if the adversary does not have physical access to the asset, because of the physical effort required to penetrate the industrial cabinet and access the asset within. On the other hand, if the adversary has physical access to the asset, the **Damage** attack has a much lower cost, because the adversary can use his or her bare hands to disconnect the wires connecting the asset to the system infrastructure.

Damage and permissions-related attacks will always be detected, because the effect of the attack is obviously malicious. The **DirectControl**, **Move**, **Login**, and **RemoteControl** attacks will not be detected, because the action can be mistaken for a typical maintenance routine. Finally, the **Move**, **Login**, and **RemoteControl** actions are logged and associated with the credentials in possession of the attacker (where the credentials could be stolen, forged, or the adversary's own identity). However, since we assume that the defender does not actively scan through the collected logs for malicious activity, the probability of detecting the attack is 0. In the next subsection, we will introduce a defense mechanism that alerts the defender when suspicious movement is detected, and we then adjust the detection probability of the **Move** action accordingly.

2.7 RESULTS OF SECURITY ANALYSIS

In this section, we present the results of our simulations and discuss the insights garnered from the results, identifying areas in the system that require additional defenses. Table 2.5 shows the resulting metrics for each adversary profile (with the exception of the **PrivilegedInsider**).

Attack Paths From the simulation results, we find that all adversaries, except the access control staff, can achieve the **TrainStop** goal by damaging or controlling devices in the rooms. The **TempCtrlOff** goal is achievable by all the adversaries except the power, signals, and access control staff, whose access to the system is more specialized or limited. Similarly, the **LockExits** goal is achievable by all adversaries except the power and signals staff. Finally, the **TrainCollision** goal is achievable only by the station operator, server, and signals staff, because the attack requires specific control of the signals server.

We can see from the simulation results that although the building access control system allows staff to move only into rooms within their job roles, staff are still capable of affecting

Table 2.5: Calculated metrics for each adversary profile. The different possible attack steps (excluding actions that represent physical consequences) are listed.

	Cost	Detect	Attack Path Length	Attack Steps	Goals
Outsider	12 or 13	3 or 4	7–16	**Damage:** PB 2, 4, or 5, Temp Control, Power System, Signals Server	TrainStop TempCtrlOff LockExits
Cleaner	12 or 13	1	7–16	**Damage:** PB 2, 4, or 5, Temp Control, Power System, Signals Server	TrainStop TempCtrlOff LockExits
Station Op	12 or 13	0 or 1	7–16	**Damage:** PB 2, 4, or 5, Temp Control, Power System, Signals Server **Login:** Host **ControlSoftware:** PLC **RemoteControl:** Power System, Temp Control, Signals Server	TrainStop TempCtrlOff LockExits TrainCollision
Env Ctrl	5 or 10	0 or 1	8–15	**Damage:** PB 2, Temp Control **DirectControl:** PB 2, Temp Control	TrainStop TempCtrlOff
Server	11 or 13	0	7–9	**DirectControl:** PB 3 **Login:** PLC **RemoteControl:** Power System, Temp Control, Signals Server	TrainStop TempCtrlOff LockExits TrainCollision
Power	6 or 11	0 or 1	7–9	**Damage:** PB 4, Power System **DirectControl:** PB 4, Power System	TrainStop
Signals	6 or 11	0 or 1	7–9	**Damage:** PB 5, Signals Server **DirectControl:** PB 5, Signals Server	TrainStop TrainCollision
Access Ctrl	6 or 11	0 or 1	7–9	**Damage:** PB 3 **DirectControl:** BAC Server	LockExits

devices in different rooms to achieve their goals. In particular, the environment control staff can cut power to the track despite not having access to the power room or the server

room that controls the device logic. That attack path could easily have been missed by a human analyst who did not consider the physical consequences of changing a device's temperature. Thus, it is important not to overlook noncritical staff members or assume that a malicious action will remain contained within a room. The effects of a noncritical staff member's actions, however, take much longer to propagate through the system than those of other adversaries, so the practitioner has more time to catch the attack before the adversary's goal is achieved. Thus, it is crucial to implement detection mechanisms that monitor changes in physical processes.

Attack Steps We find from our simulation results that the **Damage** action is the most commonly used action by all the adversary profiles and often targets powerboards. Since the outsider, cleaner, and station operators have physical access to all rooms, they can perform the **Damage** action on the appropriate device to achieve any goal. Thus, the cost to damage equipment, specifically powerboards, should be increased via additional physical guards. We also note that none of the insiders were detected unless they physically damaged equipment. Having a staff member escort a cleaner through the premises reduces the chance that the cleaner will be able to perform malicious actions. However, that would not be enough to detect malicious maintenance staff or station operators, so a specialized detection mechanism is needed to distinguish between malicious actions and normal actions. In particular, we need to monitor physical movements, host logins, and actions performed using the control software.

Adversary Stealth During an adversary's decision-making process, he or she may have a choice of several attack paths. An adversary chooses one of the attack paths based on his or her preference of either minimizing the cost of the attack path or reducing the risk of being detected by a defender. We want to investigate which attack paths (which correspond to use of different access cards) an adversary will take, given differing levels of stealthiness.

In particular, we want to investigate the deterrent effect on attackers of putting in place an intrusion detector for malicious physical movement. We model the detector as a fixed detection probability (0.5) for the **Move** attack step. In other words, the attacker has a 50% chance of being detected each time he or she moves into a space that requires a card swipe (*loggedAccess*).

We represent the adversary's stealthiness with a preference weight (which is then used in the decision-making process). A weight of 0 implies that the adversary does not mind being detected; a higher weight implies that the adversary cares more about remaining undetected. We vary the preference weight from 0 to 10 and observe which access cards the

PrivilegedInsider uses to achieve the **TrainCollision** goal.[3]

The simulation results show that for weights between 0 and 8, the adversary uses the signals staff member card to access the signaling room. If the weight is above 8, the station operator staff member card is used to access the PSC.

That shows that the stealthiest adversaries, i.e., those with higher weights, will use a station operator's access card to enter the station. Since the probability that the adversary will be detected increases with the number of physical movements he or she takes, a stealthy adversary will opt to take the shorter paths through a station, at the cost of an increased amount of effort needed to conduct the attack. System practitioners should thus focus their efforts on detecting the subsequent actions of that adversary that include logging into the hosts and using the control software to perform actions.

Summary In conclusion, the adversaries that pose the biggest threat to the system in terms of shorter time (or path length) to achieve goals and lower detection probability are the server staff and station operators. When considered in combination, the outsider together with the access control staff may also pose a problem, since the access control staff member is able to grant access to the station. Our results also show that the inclusion of a detector for malicious physical movement causes an adversary to adjust strategies by choosing shorter paths through a building at the cost of increased effort in executing the attack.

2.8 RELATED WORK

Analysis of system security requires significant manual effort from the practitioner. One way to reduce the practitioner's burden is to use a common ontology or library that can be applied to multiple similar systems. Ontologies have been used to build a knowledge base of vulnerabilities and known attacks [30, 31]. The tool $P^2CySeMoL$ [32] also builds up a meta model that is based on a library of attack steps and countermeasures. Well-established tools such as the TVA-tool [25, 26] use a predefined library of exploits. However, these ontologies and libraries focus solely on the cyber domain, and thus are lacking when applied to cyber-physical systems.

Chen et al. [12] analyzed a train control system and a mobile transportation app using attack trees and a Failure Modes, Vulnerabilities and Effects (FMVEA) analysis. They showed that the key challenges in analyzing a cyber-physical system's security are those of identifying attacks from both the cyber and physical domains, and tracing the consequences

[3]The preference weight can be any integer, but we will see later that values above 11 do not affect the results.

of attacks in the physical domain. For example, Marronne et al. [33] combined two *Unified Modeling Language* (UML) models, one addressing physical protection of a system and the other addressing cyber protection, to perform vulnerability analysis in critical infrastructure systems. They applied model transformations to the extended language to obtain the corresponding Bayesian Networks and Petri Nets for the system. They used a small use case of a railway trackside shelter to evaluate their approach. The TREsPASS project [34] developed an attack navigator [35] that takes in adversary profiles and a library of attack patterns to generate an attack tree. Those attack patterns can span the cyber, human, and physical domains. Although the project has ended, the authors aim to explore the usage of ontologies.

Our approach also uses a high-level language to represent basic concepts and attack steps. However, the ontology that we built aims to address the physical consequences of attacks in far more detail. To address the effects of attacks in the physical domain, Peter et al. [36] extended attack graphs to model the details of physical processes in a system. They modeled the overheating of a transformer in a substation and show the key attack steps that need to be performed. However, their approach is very specific and difficult to apply to a diverse set of devices in a complex setting.

2.9 CONCLUSION

System practitioners often evaluate the attack surface of cyber-physical systems purely from the cyber perspective, without taking into consideration the cascade effect of physical consequences or actions that can impact the system negatively. In this chapter, we used the ADVISE meta modeling approach to analyze a railway station.

We constructed an ontology that describes the assets and spaces in the system, and physical relations between them. We also defined attack steps that model the physical effects, such as causing damage, device overheating, and powering off. That ontology can be reused on other cyber-physical system case studies. Using the ADVISE tool, we generated an AEG for a small portion of the railway station and simulated different adversary profiles' movements through the station.

The results show that adversaries can intelligently target a device within their reach, causing a cascade that leads to a bad system state. When a detection mechanism for physical movement is present in the system, the model shows how adversaries adjust their strategy in response.

That result motivated our work in Chapter 4, which involves building such an intrusion detector for malicious physical movement in a building. In Chapter 3, we will delve deeper

into the system architecture and cyber protocol to deduce the possible cyber attacks that can affect system safety (i.e., the **TrainCollision** goal). Then, we can enhance our ontology and system instance diagram to include those cyber attacks, and rerun our analyses to obtain insights into what other defense mechanisms are needed to harden the security of the system.

CHAPTER 3: CYBER-PHYSICAL SYSTEM ASSESSMENT: SAFETY ANALYSIS

As safety-critical systems become increasingly interconnected, a system's operations depend on the reliability and security of the computing components and the interconnections among them. Therefore, a growing body of research seeks to tie safety analysis to security analysis. Specifically, it is important to analyze system safety under different attacker models. In this chapter, we develop generic parameterizable state automaton templates to model the effects of an attack. Then, given an attacker model, we generate a state automaton that represents the system operation under the threat of the attacker model. We use a railway signaling system as our case study and consider threats to the communication protocol and the commands issued to physical devices. Our results show that while less skilled attackers are not able to violate system safety, more dedicated and skilled attackers can affect system safety. We also consider several countermeasures and show how well they can deter attacks.

3.1 BACKGROUND AND MOTIVATION

Cyber-physical systems involve the monitoring and control of physical processes and devices through the interactions among computing elements over a network. Many cyber-physical systems have an impact on human life, especially critical infrastructure systems such as the smart grid and transportation systems. For that reason, much effort has gone into ensuring that such cyber-physical systems are *safe*. *Safety* of a system means that the system will never enter a bad state that adversely affects people and the environment. For example, *safety* in the context of a railway system refers to the absence of collisions and derailments.

In practice, a safety analysis is performed to ensure that a system is safe despite component faults and non-malicious human or system errors. The safety analysis is typically conducted using approaches such as hazard analysis, failure mode and effect analysis, formal verification, and fault tree analysis [37]. However, as the control of physical processes is increasingly conducted remotely over communication links, the safety of a system is dependent not just on the reliability of the system components and their interactions but also on the security of the messages sent over the network. It is also necessary to consider the system security at each architectural layer because of advances in cyber attack strategies. Many researchers, domain experts, and government bodies have thus identified the need for cyber security in safety-critical systems [5, 6] and have proposed different approaches for merging safety analysis and security analysis [7–9].

Safety-critical systems can be targeted by a variety of malicious actors with differing levels of skill and system access, like nation-state attackers and hacktivists. For example, in 2010, the Stuxnet worm infected nuclear power facilities, resulting in the failure of many uranium-enriching centrifuges [2]. As seen from the incident, it is hard to guarantee a system's safety when it is under the threat of a nation-state attacker willing to dedicate time, energy, and resources to infiltrate the system. While it may not be possible to assure complete safety of a system, it is important to identify the necessary qualifications a malicious actor needs in order to carry out an attack that violates system safety. A safety case can be built using that knowledge as supporting evidence [38], and a security practitioner can focus efforts on strengthening the system against such attacks.

In this chapter, we model cyber-physical systems and human actors' capabilities using state automata. For each attack capability, we define a generic model pattern that represents the effects of an attack. This model pattern can be appended to the appropriate components in the system state automata. We can then generate a state automaton that describes both the normal workings of a system and the actions of an attacker on that system. As a case study, we analyze the safety of a railway system relative to various types of malicious actors. We consider several classes of attackers with respect to their capabilities and system access, motivated by the existing literature on the feasibility of various attacks. Based on our collaboration with railway industry partners, we constructed a hybrid automata model of a railway system by using UPPAAL [39], a tool for modeling and verifying real-time systems. Finally, using statistical model checking techniques, we assessed the safety of the system under various input configurations of the trains in the system.

In summary, the contributions in this chapter are as follows:

- We modeled cyber-physical systems and human actors' capabilities using state automata. We also constructed a framework for generating automata components in UPPAAL that represent the effects of an attack on a system. That allows us to generate a state automaton that can be used for analysis of different attacker models.

- For each set of attacks, we verify the system's safety by using statistical model checking techniques. We inspect those attacks that induce a violation of safety to draw insights into the system components that are more vulnerable to attack and the possible countermeasures.

- We describe how we applied our framework to conduct a safety analysis of a railway system subjected to a variety of malicious attacks.

- We apply our security analysis approach from Chapter 2 to determine which attacks are likely to be executed by an attacker to compromise system safety.

3.2 RELATED WORK

For many years, the well-grounded theory of formal methods has been used to prove the safety of industrial systems [40]. One of the most well-studied and successful applications of formal methods is on verifying the safety of railway signaling systems [41]. With the rapid evolution of the traditionally human-operated railway systems into a more automated communications-based system, the target of the formal verification techniques has also changed, so that they now focus on proving the correctness of software and application data [42–44]. Since the communication network also plays an integral role in the system, work has also been done to verify that service availability is not affected by network degradation or failure [45].

With the rise in cyber attacks on safety-critical systems, more researchers have investigated the safety of railway systems under communication and command errors. Chen et al. [12] conducted a security analysis of railway systems using methods from both the safety and security engineering domains, and show the need for a cyber-physical perspective to study the complicated physical consequences of cyber breaches. In [46], Ghosh et al. modeled the physical train movement and communication with the servers in the network. The safety property was specified as maintaining a safe distance between two trains. The authors found counterexamples of the system's being in an unsafe state when there were errors in the application data or in the movement authority that determines the maximum distance the train is allowed to move. In [47], the authors focused on threats to network communication. They simulated anomalies that included unknown information from track sensors, unexpected train data, and unauthorized command messages. In [48], the failure logs of a real accident were used in failure analysis to discover the source of the accident, which was traced to communication failures. Finally, Cappart et al. [49] used statistical model checking to verify that application data were accurate and that trains would not collide when there were potential errors in application data and in locking points.

However, those approaches consider only a few specific attacks (e.g., an inaccurate movement authority) and do not provide a comprehensive understanding of safety when the system is subjected to a variety of attacks. Bloomfield et al. [50] presented a risk assessment of a national railway system and considered the impact of various attacker models on the system design. The authors used a fault tree analysis to conduct their risk assessment based on expert knowledge and analysis. However, that approach may miss many potential com-

binations of attack vectors that affect system safety. Thus, a more automated and formal approach is needed to investigate the space of potential attacks.

Rocchetto and Tippenhauer [51] used the CL-Atse tool to find counterexamples (i.e., potential attacks) on a real-world water treatment testbed, given a Dolev-Yao attacker model [52] that they extended to include physical-layer interactions with the system [53]. However, their approach does not take into account network topology and assumes a single network channel. Puys et al. [54] considered both the Dolev-Yao attacker model and three other attacker models that each had a subset of the capabilities of the Dolev-Yao attacker. However, they did not consider physical-layer interactions in the attacker model. The authors also considered the network architecture and the position of the attacker within the system.

Since both of those approaches consider a generic attacker, the state space of the model is extremely large and, given a more complicated system, would result in state-space explosion. In addition, the counterexamples that are found from such a model may not give an intuitive insight into the circumstances under which attack capabilities translate into an unsafe situation. In our work, we present a more parameterizable attacker model that allows the modeler to define the situations in which an attack is executed. We also consider physical-layer interactions and the attacker's position.

3.3 STATE AUTOMATON MODEL

Our approach to verifying the safety of a system is shown in Figure 3.1. The first step is to model the normal system operation as a state automaton, and define the model templates for attacks. Then, we insert those attack templates into the system automaton to create a parametrizable state automaton. Next, we generate the full state automaton with the given parameters and use UPPAAL's model checker to verify that the state automaton satisfies the safety property. In this section, we describe the first step of modeling normal system operation and defining model templates.

We use UPPAAL to model and verify a cyber-physical system. The tool allows us to specify networks of timed automata and verify properties, which are specified using temporal logic, on the automata.

Definition 3.1. *We use $B(V)$ to denote a Boolean constraint over a set of symbols (or variables) V. A timed automaton is a tuple (V, C, L, l_0, I, A, E) where*

- *V is a set of variables,*

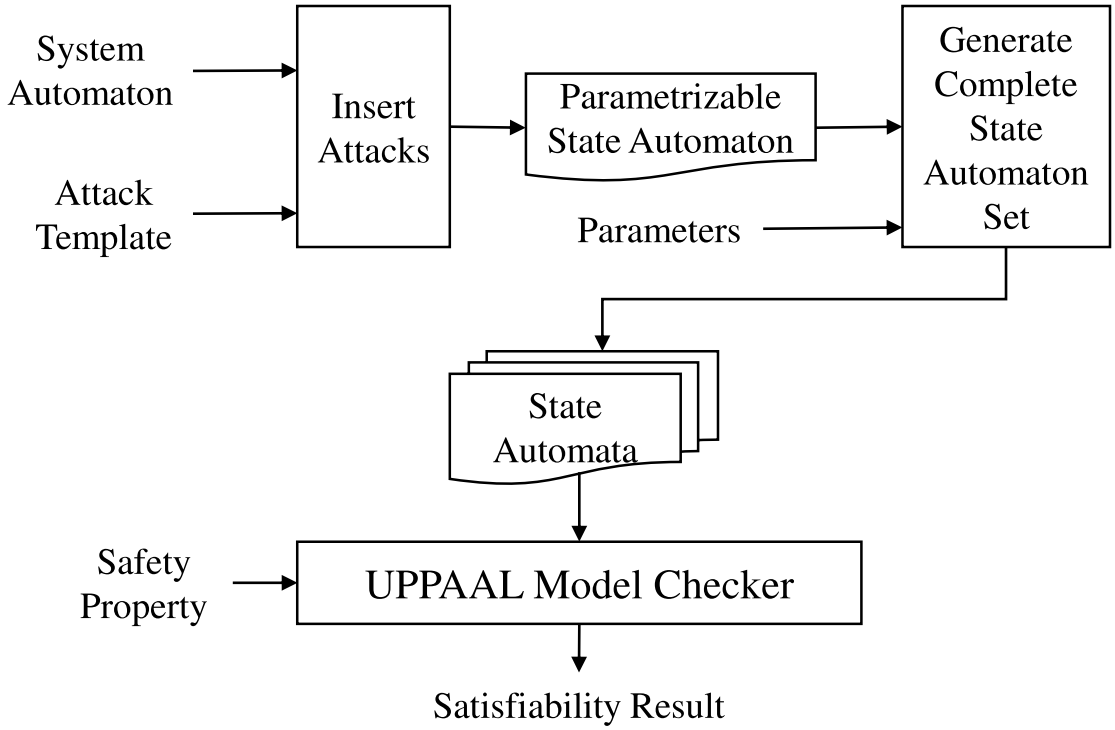


Figure 3.1: The general workflow of our approach.

- C *is a set of clocks,*
- $\mathcal{H}$ *is a set of channel names,*
- L *is a set of locations (or states),*
- $l_0 \in L$ *is the initial location,*
- $Sync = \{ch!, ch?|ch \in \mathcal{H}\}$ *is a set of synchronization actions over the channels,*
- $I : L \to B(C \cup V)$ *maps locations to invariants over clocks and variables,*
- $A : 2^V \times 2^C \to 2^V \times 2^C$ *is a set of actions that does an assignment of variables* $w = exp(2^V), w \in V$ *to an expression over the variable values, and clocks* $c = n, c \in C, n \in \mathbb{N}^+$ *to a natural number, and*
- $E \subseteq L \times A \times B(C \cup V) \times Sync \times L$ *is a set of transitions.*

The system moves from one state to another by taking transitions. A transition $(l, a, g, l') \in E$ is fired if its guard g (i.e., a condition on variables and clocks) is enabled. If the transition is taken, its action a (i.e., assignment of variables and clocks) is performed[1]. The UPPAAL

[1]UPPAAL allows custom-coded functions in the actions to perform the assignment of variables and clocks.

modeling language also extends the timed automata with concepts such as *synchronization* that allow state automata to communicate via *channels* [39]. More specifically, two or more automata will take a transition at the same time if they are synchronized on the same channel `chan`. Messages are sent on a channel through use of the syntax `chan!`. The sending of the messages are synchronized by (possibly multiple) receivers via the syntax `chan?`.

Timed automata model the progression of time, and networks of such timed automata model concurrent processes, so timed automata are very suitable for modeling the network communications and physical processes in a cyber-physical system. Figure 3.2 shows a small timed automaton for a cyber-physical system that consists of a moving vehicle and its cyber controller.

Now, we describe how we build a model of a cyber-physical system using timed automata. We use variables to model the physical attributes of a system. In our small example in Figure 3.2a, the physical attribute of the system that is important to its state is the velocity of the vehicle, which is given by the variable v. The evolution of the physical process is governed by differential equations, e.g., $\dot{v} = c$, an approach that is supported by the *Statistical Model Checking* (SMC) extension of UPPAAL [55].

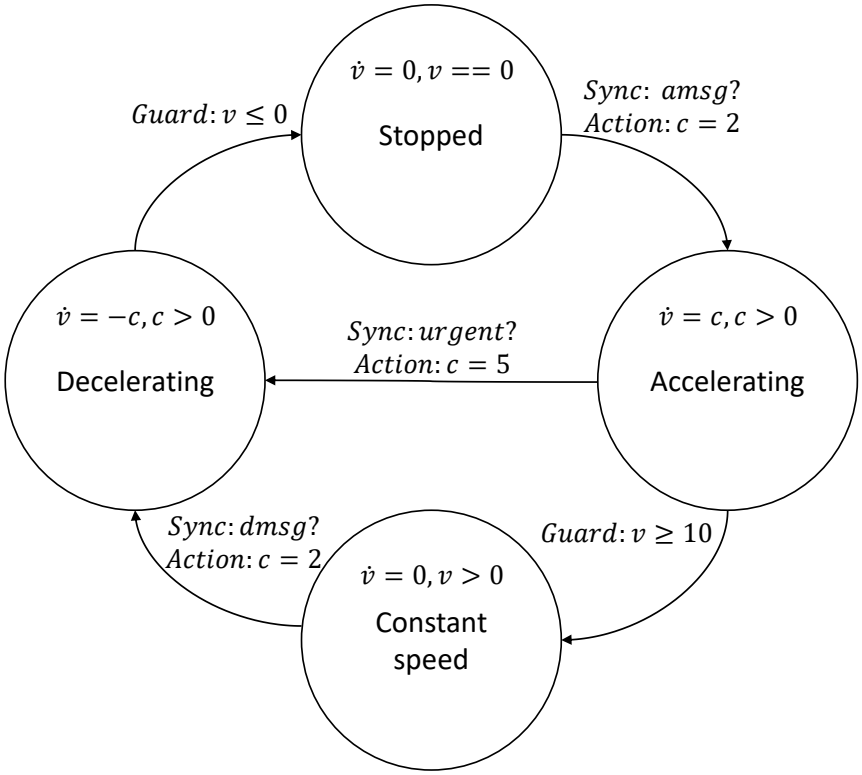
Digital data such as memory contents and network message contents are similarly modeled as variables. The communication between the cyber controller and physical component is achieved using synchronization over the channels, e.g., $urgent$, $amsg$, $dmsg$ in our small example.

Then, we can represent the safety property of the system as an invariant of the system $Inv(System)$. For example, the formula $A\Box(0 < v \leq 10)$ specifies that the vehicle's velocity must always remain below a speed limit of 10 and be non-negative in value.

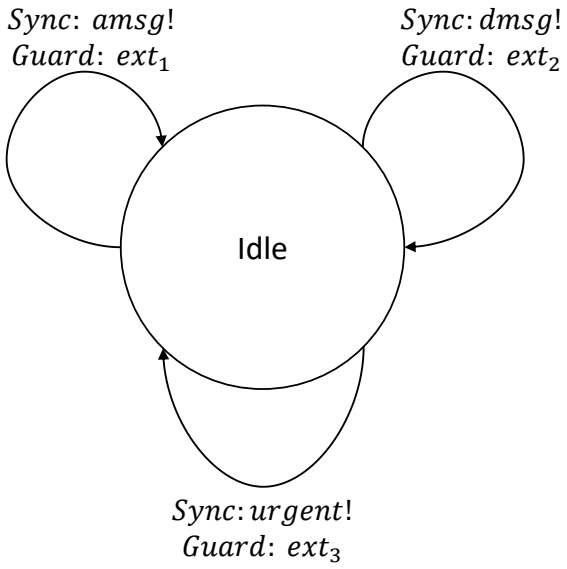
3.3.1 Attacker Model Template

We represent the attack capabilities as transformation functions Δ over the state automaton representing the cyber components. In general, the attacker capabilities we consider in this chapter are (1) removing, (2) delaying, and (3) inserting network messages and control commands (i.e., the server's sending of a signal to set points). For each class of attack, we define a generic model pattern that can be applied to a specified target.

The circumstances under which an attack is performed can be either probabilistic or deterministic. For example, the attack mentioned in [56] that spoofs messages between the trainborne system and server involves a probabilistic chance of being able to successfully craft a message. On the other hand, a jamming attack relies on the location of the jammer in the case of certain wireless communications, and as such can be represented as a guard.



(a)



(b)

Figure 3.2: Small state automaton example of (a) a train's physical movement, and (b) a controller for that train. The controller receives external input ext_1, ext_2, ext_3 from a human.

We represent that circumstance as a branch point when probabilities are involved or as a committed location when deterministic decisions are involved. The two can be combined when the circumstances involve both a deterministic choice and a probabilistic bound.

We first model the class of attacks that involve removal of a network message or command. If the attacker is trying to remove a network message, the target of the attack is a transition with a synchronization channel `message`. If the attacker is trying to remove a command, the target is a transition with the corresponding action. Targeting of a `message` can happen on either the sender's or receivers' side. If the sender's `message!` is targeted, all receivers are equally affected by the change. On the other hand, if a receiver's `message?` is targeted, only that receiver is affected. Removal of a network message involves removal of the synchronization `message?` or `message!` from the transition and removal of any other actions that refer to the assigning of the message contents. Removal of a command involves removal of the actions associated with the command. We show in Figure 3.3a an attack that probabilistically succeeds in removing a command.

Delay of a network message or command $(u, a, g, v), u, v \in L$ by t time units involves addition of a location x and a new clock $delayTimer \in C$. The invariant of x is $I(x) = delayTimer \leq t$. The transition (u, a, g, v) is replaced by (x, a, g', v) with a guard $g' = delayTimer \geq t$, which implies that the delayed time has elapsed. A transition (u, a', g, x) is added with the update $a' = delayTimer = 0$ to start the timer. We show in Figure 3.3b an attack that deterministically decides when to delay a network message.

Finally, insertion of a network message or command is more circumstantial than the first two classes of attacks. In general, we can model such an insertion by creating a new automaton and modeling the circumstances under which the insertion should take place. The insertion itself is modeled by a transition with either a synchronization `message!` in the case of network messages, or updates in the case of a command.

Thus, we can model a set of attacker capabilities as $\delta \in 2^{\Delta}$ and apply that transformation δ to the system automaton $System$. Then, we formulate the problem to be solved in this chapter as the following question: Given $\delta \in 2^{\Delta}$, is the invariant $Inv(\delta, System)$ satisfied?

3.4 CASE STUDY: RAILWAY SIGNALING SYSTEM

Over the years, railway systems have started to integrate cyber technologies and infrastructure to improve system efficiency with regard to route capacity, that is, the number of trains that can be driven on the rail lines while maintaining safety conditions. The safety condition is measured by the acceptable distance between trains, or *headway*.

Previously, the train signaling system was a fixed-block system wherein train locations

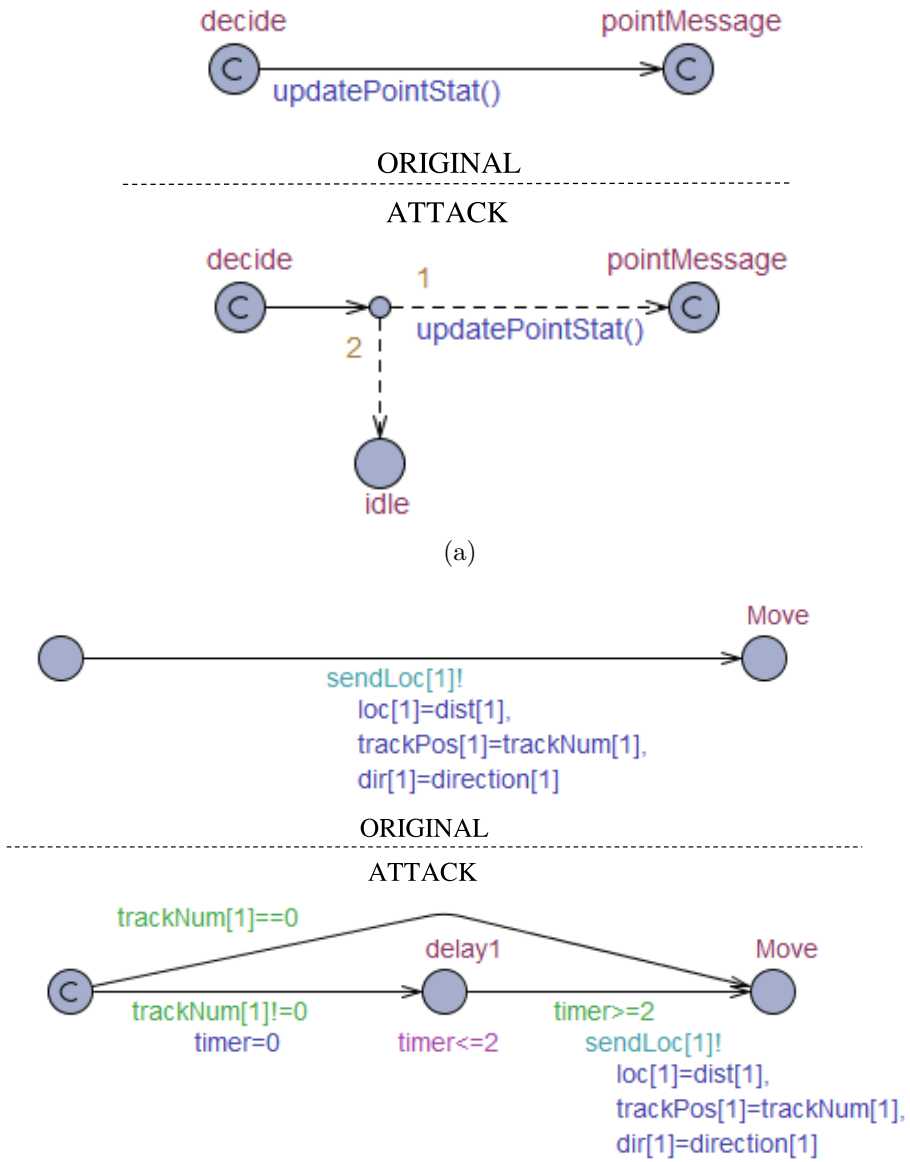


Figure 3.3: Model template that (a) removes a command with probability 0.3, and (b) delays a network message by 2 time units if the train is on a given track.

were determined by old track circuits located at fixed intervals on the rail tracks. Now, the train signaling system has evolved into a moving block system whereby trains constantly relay their locations to a controller via a networked system, i.e., a *Communication-Based Train Control* (CBTC) system. This increased precision in identifying the location of trains allows the system to enforce a shorter headway between trains.

Although the quality of system service is better with a CBTC system, the dependence on cyber infrastructure also increases the system's susceptibility to attacks. The impact of

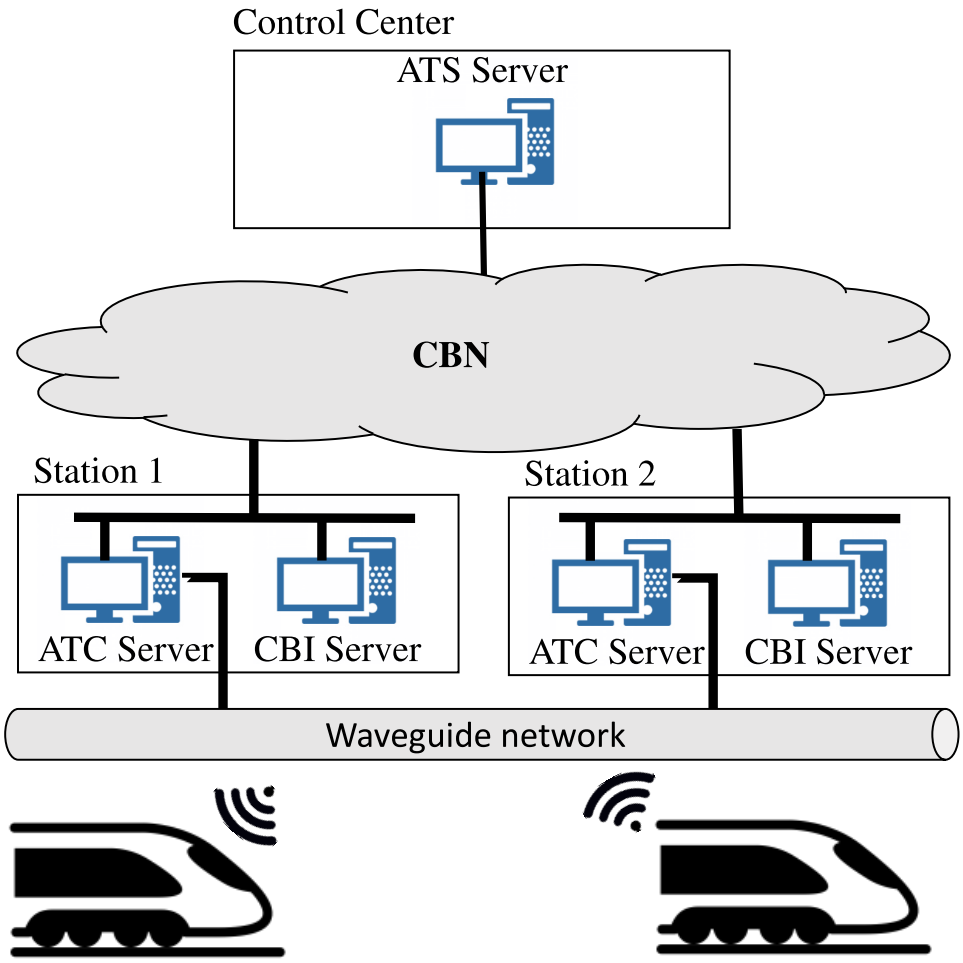


Figure 3.4: The architecture of the signaling components of the railway system.

such attacks can be very severe, ranging from service delays to derailment. For example, as mentioned in Section 1.4, a Polish teenager once rewired a remote control to communicate with a wireless switch junction, causing derailment of a train and injury of twelve people [16]. It is thus important to understand the potential extent of malicious actors' impact on railway system safety.

In this paper, we focus on the signaling components of the railway system, because they directly affect the speed of the trains and the routes that they take. While the general workings of the signaling systems are the same across all railway systems, railway companies differ in the exact communication protocols used and divisions of computation among the architectural components. We modeled our signaling system based on information we obtained via collaboration with our railway system partner. Although our results are based on that model, they are easily generalizable to other signaling systems.

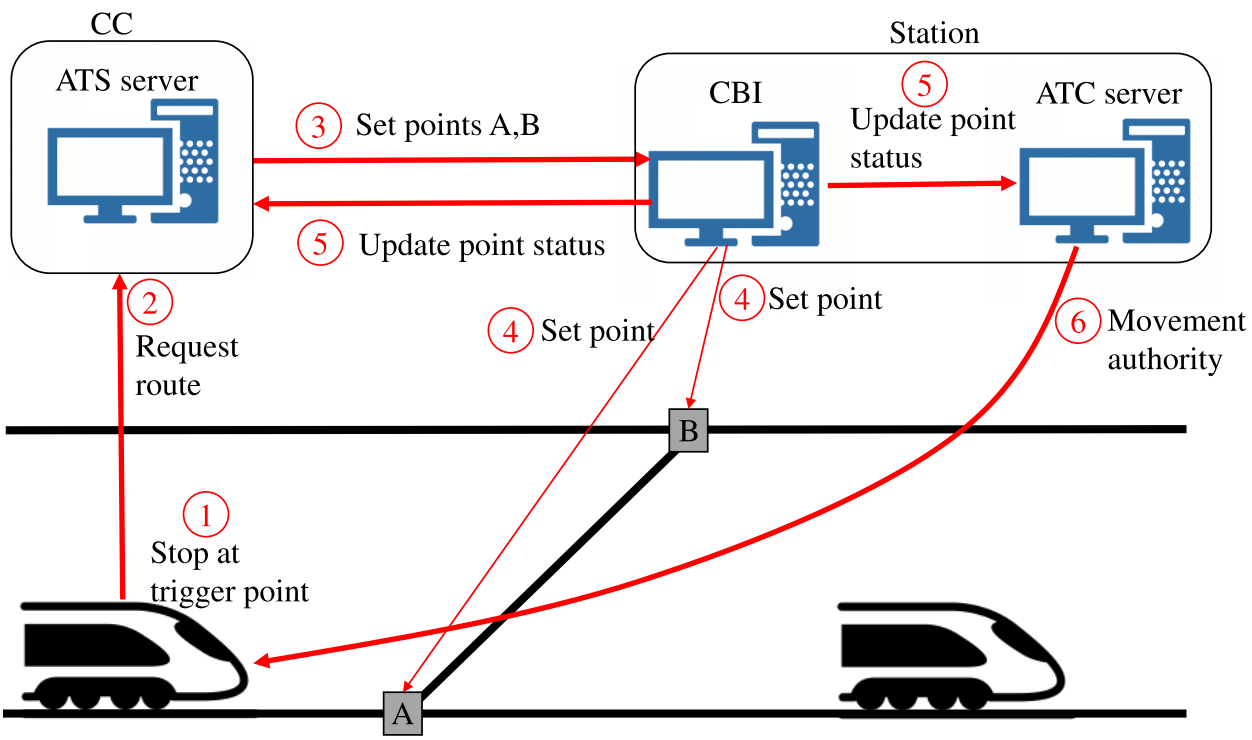


Figure 3.5: The protocol underlying the movement of trains as they approach a branch in the tracks. The numbers in circles indicate the sequence of messages sent, and the black undirected lines represent the tracks.

3.4.1 Signaling System

The signaling subsystem consists of distributed components in three different locations: the *Control Center* (CC), the station, and the trackside, as shown in Figure 3.4. The *Automatic Train Supervision* (ATS) server in the CC manages the train schedules and routes, while the *Automatic Train Control* (ATC) server controls the safe distance that trains are allowed to travel. The *Computer-based Interlocking* (CBI) server manages the point machines that operate mechanical switches (i.e., points). Points are located at track branches and guide the train from one track to another, as shown in Figure 3.6.

Permanently fixed beacons placed at regular intervals across the length of the track transmit absolute positions to the trainborne system on the train. The train regularly transmits its location in a train status update message to the ATC and ATS server. Based on the information collected about the current system state (i.e., train locations, points' statuses), the ATC server generates a *movement authority* (MA) for each train that describes the maximum distance the train can travel before encountering an obstacle. The train then adjusts its speed based on the received MA.

Communication between the trackside equipment and train and the station servers is

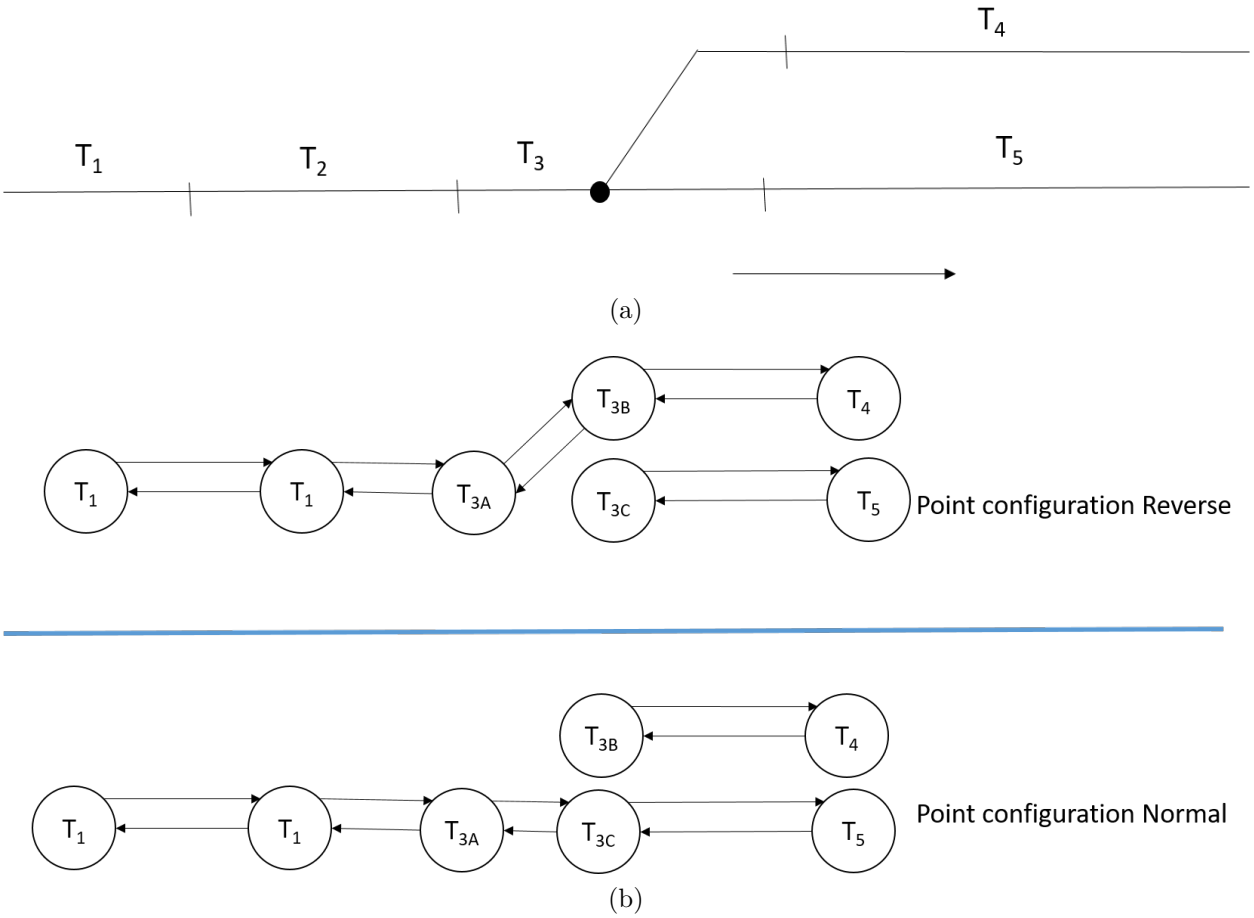


Figure 3.6: The railway track configuration at a station. (a) The labels represent track segments, and the black circle represents a point that switches the tracks so that a train can move on either track T_4 or T_5. (b) Graph representation of (a) during two different point configurations. Given a certain point configuration, parts of the graph may be disconnected because the train is unable to move to that segment of the track.

through a wireless waveguide system placed along the entire stretch of the track. Communication among the servers is through a wired backbone network that is isolated from the Internet.

When a train reaches a branch in the tracks, it halts and sends a message to the ATS server to request a route, as shown in Figure 3.5. The ATS server checks the train's schedule and, based on the embedded route tables in its software, will choose the appropriate route and inform the CBI server to lock the appropriate points. Once the CBI server has locked the points, it informs the ATC and ATS server of the change in point status. The route is then confirmed, and the ATS server reserves the set of track segments for the train. The ATC server sends the new MA to the train, allowing the train to move to the end of the

Table 3.1: The attacker models described in terms of the positions they occupy in the system and the potentially affected messages or commands. The numbers in the "protocol component" column refer to the numbered messages in Figure 3.5, where 0 represents the train status update.

Position in System	**Protocol Component Affected**	**Example of Gaining Position**
Train	0, 2, 6	Ride the train
ATC Server	0, 5, 6	Insider, malware
CBI Server	3, 4, 5	Insider, malware
Station Network	0, 2, 3, 5, 6	Insider, malware
ATS Server/CC Network	0, 2, 3	Insider, malware

reserved track segments.

3.4.2 Threat Model

We assume that the attacker does not have direct physical access to the trainborne system or the trackside devices, because those components are located in places that are hard to access. For example, accessing the trainborne system involves opening up the car exoskeleton, and accessing trackside devices involves entering an underground tunnel.

We describe different levels of capability an attacker can possess, while providing examples of how the attacker could have gained access to such a capability, as shown in Table 3.1. Since railway systems around the world vary in their implementations of security mechanisms, we take into consideration both the case in which a certain security mechanism is in place, and the case in which it is not.

The attacker with the lowest capability level is the outsider who has the least access to the railway system. However, the outsider can access the system as a passenger, and thus can interact with the communication between the trainborne system and the waveguide network. In particular, the attacker can jam communications between the trainborne system and the rest of the network, as shown in several research papers [57, 58].

Even if current modern authentication and integrity protection levels are provided for the network messages, a skilled attacker can also spoof messages to and from the train, given enough time to capture and analyze transmitted messages between the train and network [56, 59]. Depending on the networked system and authentication of endpoints, an attacker could also set up his or her own access points and perform *man-in-the-middle* (MITM) attacks.

The next attacker we consider is the insider who is a legitimate employee in the railway

system company. The insider can have physical and/or cyber access to servers and networking equipment. Although insiders may not have all the prerequisite skills for maliciously manipulating the system, they may be in league with a skilled outsider party who can tailor more sophisticated attacks to be delivered to the system with the help of the insider's access. They can then control the physical devices in the system by, for example, sending commands to the point machines by accessing the CBI server either directly or via injection of malware through USB ports. (We assume that such malware can tamper with messages without altering the logical behavior of the server's programs.)

Depending on the location of the insider within the system architecture (e.g., control center, station, or a server), an attacker can also manipulate the communication messages that pass through his or her stronghold. More precisely, the attacker can remove, insert, modify, or delay those network packets, much like a Dolev-Yao attacker. The insider has a much higher success rate in attacking the communication messages than the outsider does.

3.5 RAILWAY SYSTEM MODEL

A railway line has many junctions that are utilized for different purposes, like moving trains to and from the depot or isolating and redirecting trains during emergencies. In this paper, we model a diamond junction at an end station, as shown in Figure 3.7. We chose this junction because the four points at this junction are used the most frequently in the railway line. The points are used to redirect incoming trains to the opposite outgoing track. A large volume of passengers also passes through this junction. For that reason, it is a prime target for attackers who wish to cause collisions or derailment.

There are four paths that a train can take in a diamond junction. Trains coming into the station move from track T6 to T7 and then T8, or from T6 to T3 and then T1. Trains moving out of the station move from T8 to T5 and then T4, or from T1 to T2 to T4.

3.5.1 Trains

In our model, we allow an arbitrary number of trains to be specified, but we considered two trains in our experiments, because the system design allows for only two trains in the junction. However, the analysis can easily be generalized to three or more trains. We abstract away the length of the train and model the train as a single point on the track, because only the point of contact matters when dealing with derailment and collisions. Each train with an `id` is characterized by its horizontal distance, position on the track

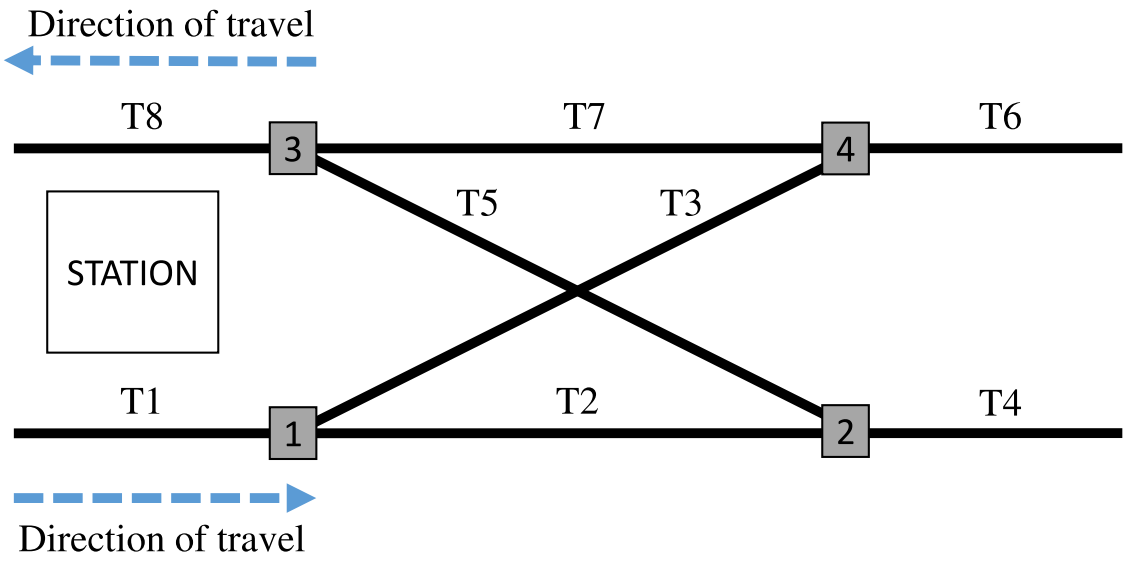


Figure 3.7: The diamond junction at an end station. Boxes on the track represent the points and are numbered. The track segments are numbered T1 to T8. Trains travel in the direction indicated by the dotted arrows.

topology, speed, acceleration, and direction of travel. Thus, each train has the variables $V_T = \{dist, trackNum, velocity, acceleration, direction\} \in V$.

We simplify the train's braking curve by specifying a constant acceleration and deceleration c. We define a Loc automaton with a single state *move* and the following invariant.

$$I(move) = dist' == direction \times velocity \wedge velocity' == acceleration \tag{3.1}$$

We define a separate automaton that periodically sends a train status update message (`sendLoc?`) to the server. We define global variables V_M representing the message contents. The transition with the `sendLoc?` synchronization has the action $V_M = V_T$, which updates the message contents.

We also define a Control automaton that describes how the train's movement is controlled, as shown in Figure 3.8. The trains are initialized in a halted state. When the transition `locFront?` is fired, an MA is received, and the action consists of calculation of the distance at which the train should start braking, `DIST_THRESH`. The train transitions to an accelerated state with action $acceleration = c$. When the guard $velocity \geq$ `MAX_VELOCITY` is enabled, the train transitions to a fixed-speed state with action $acceleration = 0$. The train transitions to a braking state with the action $acceleration = -c$ when its guard $direction \times dist \geq direction \times$ `DIST_THRESH` is enabled. When the train stops and has reached a triggering point that is a fixed distance away from the nearest point, it sends a request route message by

firing the transition `reqRoute!`. The message is resent periodically until an MA is received.

3.5.2 Track

While the Loc automaton models the physical distance, the Track automaton models the discrete position of the train in the topology, as shown in Figure 3.9. We model the topology of the track by dividing it into track segments similar to those shown in Figure 3.7. The guards for the transitions in the Track automaton are based on the distance of the train and the status of the points. When the train derails because of a particular alignment of points, it transitions to a bad state, DERAIL.

3.5.3 Servers

Each of the servers (ATS, ATC, and CBI) has a separate automaton. The servers start from an idle state and transition to a processing state when they receive a message. They then use our custom-defined functions to process the received information. Then they transition back to the idle state after sending the appropriate message. The automaton for the process threads in the ATC server is shown in Figures 3.10a and 3.10b. The automata for the two other servers are similar, so we do not show them.

The ATS server automaton receives train status update (`sendLoc?`), request route (`reqRoute?`), and point status update (`pointStatus?`) messages. The server maintains its local variables V_S, which describe the position of the trains, $V_S = V_M$. When the transition `reqRoute?` is fired, the action is a function that checks which of the four routes (mentioned in the previous section) are available; if one is available, it books the path specified by three track segments (e.g., T8, T5, T4). The transition `pointSet!` is then fired, sending a message to the CBI server, with the action that assigns the message contents to global variables. The contents include the points to be set, and an indication of whether each point should be set in the normal or reverse position. When the transition `pointStatus?` is fired, the action is setting of the route. When the train clears the last segment in its set route (e.g., T4), the server releases the route.

The ATC server automaton receives train status update messages and point status update messages. When the transition `sendLoc?` is fired together with the corresponding synchronization from the train automaton, the action consists of updating of the server's local variables and firing of a `frontLoc!` transition to send an updated MA to the trains' automata. When the transition `pointStatus?` is fired, the action is calculation of an updated MA and firing of the `frontLoc2!` transition. That transition is synchronized with

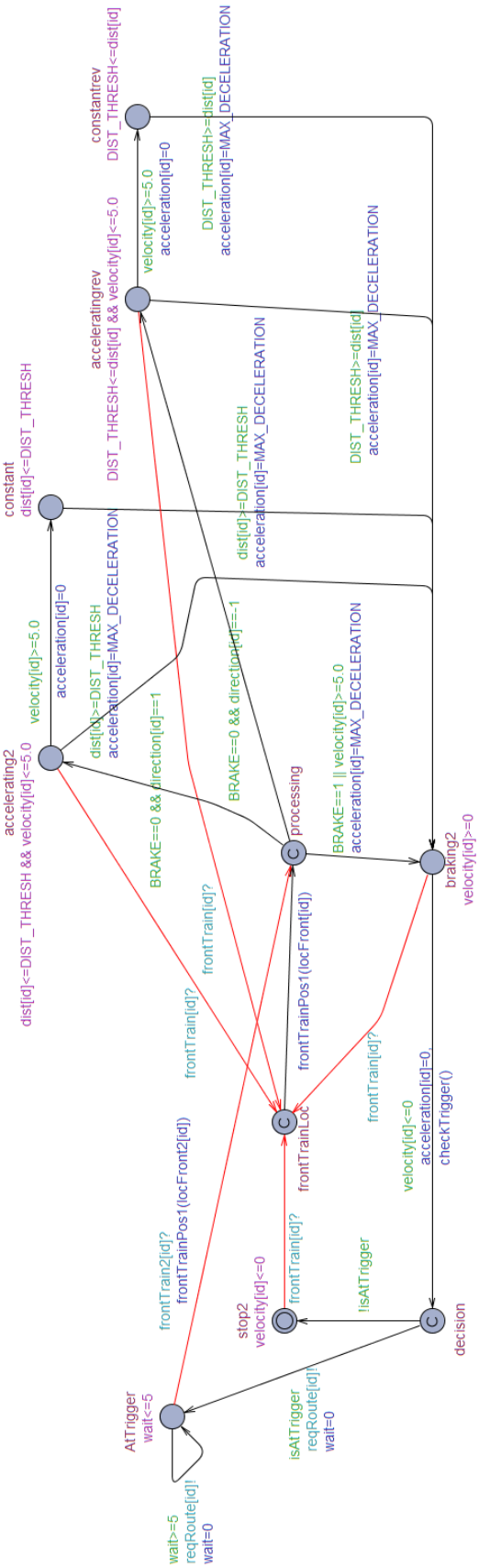


Figure 3.8: The state automaton for the trainborne system.

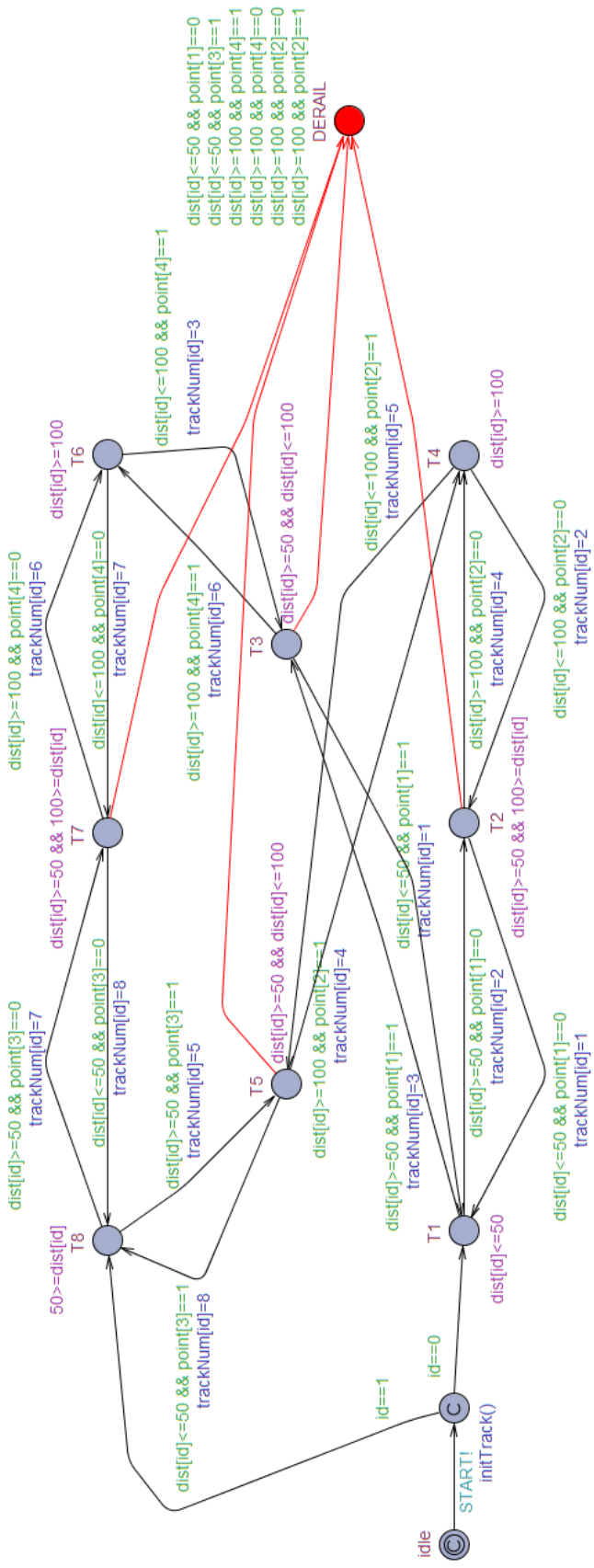
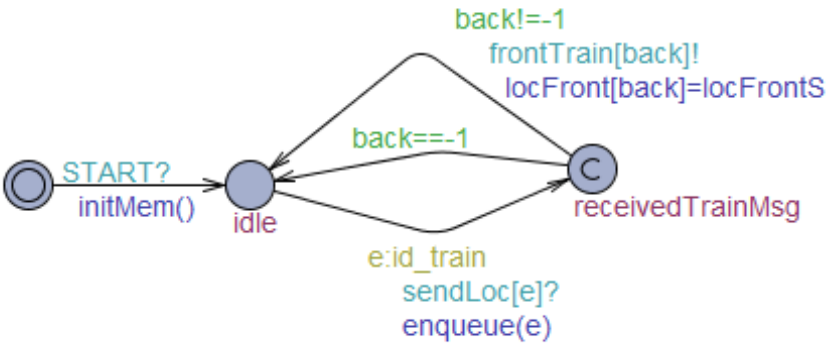
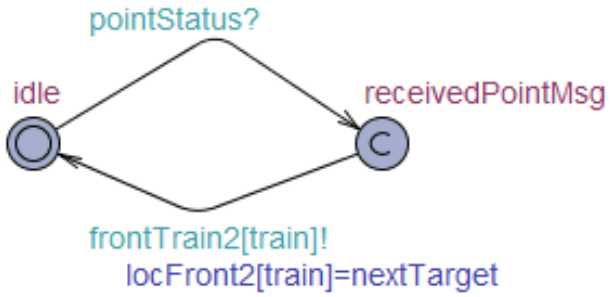


Figure 3.9: The state automaton for the track layout.



(a)



(b)

Figure 3.10: The state automata for (a) the train message thread in the ATC server, and (b) the point message thread in the ATC server.

the train held at the triggering point, allowing the train to move until it clears its set route.

Finally, when the CBI server automaton's transition `pointSet?` is fired, the action is setting of the POINTS variables' values as given by the global message variables. Then, the `pointStatus!` transition is fired.

3.5.4 Safety Property

In this chapter, we focus on verifying the safety of a railway system. Although service unavailability can be a large repercussion and affect the railway company negatively, it does not involve loss of human life. Therefore, in this work, we do not consider service availability as a property that the system must satisfy. The verification of service availability is discussed further in Section 6.2.

The safety property is specified in temporal logic and has three requirements, as follows:

$$A\Box(\neg Track(0).\texttt{DERAIL} \wedge \neg Track(1).\texttt{DERAIL}) \tag{3.2}$$

$$\begin{aligned} A\Box(&\neg(Track(0).\texttt{T3} \wedge Track(1).\texttt{T5}) \wedge \\ &\neg(Track(0).\texttt{T5} \wedge Track(1).\texttt{T3})) \end{aligned} \tag{3.3}$$

$$\begin{aligned} A\Box\ (\ &|dist[0] - dist[1]| \leq 20 \implies \nexists p \in \text{POINTS s.t.} \\ &trackNum[0], trackNum[1] \in connection(p)\ ) \end{aligned} \tag{3.4}$$

where POINTS $= \{point[i], i \in [1, 4]\}$ is the set of points and *connection* is a function that maps points to the set of tracks that it connects in the topology. Formally,

$$connection(p) = \begin{cases} \{T_i, T_j\} & \text{if } p = 0, \\ \{T_i, T_k\} & \text{otherwise} \end{cases} \tag{3.5}$$

Statement (3.2) means that a train must not transition into a derailed state, i.e., the train must not run through a point that is not aligned in the right direction. Statement (3.3) implies that two trains must not be located on tracks that cross each other, i.e., T3 and T5. Finally, statement (3.4) implies that if the two trains are on tracks that are connected, they must be separated by a safe distance of at least 20 meters.

3.6 RESULTS OF SAFETY ANALYSIS

In this section, we generate multiple state automata representing the system subjected to a set of attacks. We then use statistical model checking to verify system safety. If the safety property is not satisfied, UPPAAL will generate a counterexample showing a possible attack execution trace that results in an unsafe system state. By varying the set of attacks modeled in the state automata, we analyze the extent to which attackers will be able to compromise system safety, and we then propose countermeasures to those attacks.

Since the model we construct is complex and large, the state-space explosion makes it hard to perform a full state-space exploration. Statistical model checking, however, does not involve exploration of the full state space to determine satisfiability. Instead, it relies on generation of multiple simulations of the system and use of statistical reasoning to determine satisfiability. Although it does not cover all possible system trajectories, it gives a confidence bound on the probability of the satisfiability. Therefore, we use UPPAAL's SMC extension to model check our state automata.

We specify the safety property defined in Section 3.5.4 as the below query in UPPAAL's

syntax. We abstract away some syntax and details involved in listing the crossing tracks (i.e., T3 and T5) and the set of connected tracks. The model checker will return the probability that within 100 time units of simulation, the safety property will always be satisfied.

$$
\begin{aligned}
&Pr[\leq 100](\ [\,] \\
&\quad !Track(0).\text{DERAIL} \ \wedge \ !Track(1).\text{DERAIL} \ \wedge \\
&\quad (-20 \geq dist[0] - dist[1] \geq 20 \ \vee \ (Track(0) \neq Track(1) \\
&\quad \wedge \ \text{notCrossing}(Track(0), Track(1)) \ \wedge \\
&\quad \text{notConnected}(Track(0), Track(1)) \ )))
\end{aligned}
\tag{3.6}
$$

3.6.1 Experiment Setup

The initial parameters to the model include the trains' starting positions and the status of the points. There are three possible combinations of valid starting positions: (1) T1 and T8, (2) T1 and T6, and (3) T6 and T8. There are sixteen possible combinations of initial point positions. In total, we have 48 input configurations. We set the train status updates to be sent periodically every 300 ms. The maximum speed of the trains is 33 m/s. Finally, the length of each straight track segment (T1, T2, T4, T6, T8) is 50 meters.

The SMC extension in UPPAAL allows the modeler to configure the statistical parameters for model checking. We set the probabilistic uncertainty to 0.001 with a 95% confidence interval. We conducted the experiments on a Windows 10 Pro machine with a 3.4 GHz CPU core and 32 GB of RAM.

We consider a subset of the attacks defined in Table 3.1 that have the potential to affect the system's safety. Since we do not focus on system availability in this work, we do not consider attacks that merely reduce the quality of system services, such as by delaying the sending of the MA to the train. For each of the attacks that we consider, we define in Table 3.2 the circumstances under which the attack is executed and the extent of the attack.

3.6.2 Results

We tested for system safety under different combinations of attacker capability and input configurations. The results of our experiments are shown in Table 3.3, where we show the number of input configurations in which safety was violated. The numbers of runs that UPPAAL took to decide there was safety or to provide a counterexample ranged between 1,843 and 951,597.

Table 3.2: List of attacks with the circumstances in which they are executed and the extent of their effects.

Attacker Position	**Target**	**Attack Class**	**Circumstance**	**Parameter**	**Value**
Outside	Train Status Update (0)	Remove	Train located on track segment	Number of track segments	2 and/or 3
		Delay	N/A	Number of time units	2–4
		Insert	Every second after requesting route, probabilistically determine success of attack	Probability of success	0.01
Outside	Movement Authority (6)	Insert	Every second, probabilistically determine success of attack	Total number of messages inserted	1–10
		Remove	Every second, probabilistically determine success of attack	N/A	N/A
ATC Server	Movement Authority (6)	Insert	Send MA after train requests route	N/A	N/A
CBI Server	Set Points (4)	Remove	N/A	N/A	N/A
		Delay	N/A	Number of time units	1–10
Station Network, ATC Server, or CBI Server	Point Status Update (5)	Insert	Probabilistically determine success of attack	Probability of success	1 or 0.01

Table 3.3: Results of testing for system safety with and without safety countermeasures. For each set of attack capabilities, we list the number of input configurations that violate safety, and the average probability of the system's being safe.

Attacker Position	**Attack Capability**	**Affected Input Config**	**Probability Range**
Outsider	Remove and delay train status update	0	[0.998,1]
	Add 1 MA	0	[0.998,1]
	Add 1 MA, delay train status update	24	[0.910,0.912]
	Add 1 MA, remove train status update	44	[0.910,0.912]
ATC Server	Add MA	32	[0,0.002]
CBI Server	Remove point set command	46	[0,0.002]
	Delay point set command	36	[0,0.002]
Station Network, CBI or ATC Server	Add point status update	32	[0.910,0.912] or [0,0.002]
Safety Countermeasure: Stop When No MA Received			
Outsider	Add 1 MA, remove train status update	24	[0.910,0.912]
	Add 9 MA, remove train status update	44	[0.910,912]
	Remove MA, insert train status update	32	[0.910,0.912]
Safety Countermeasure: Secondary Track Detection System			
Outsider	Add 1 MA, delay train status update	0	[0.998,1]
	Add 1 MA, remove train status update	0	[0.998,1]
ATC Server	Add MA	0	[0.998,1]
	Add MA, remove trackside device signal	32	[0,0.002]
Station Network, ATC Server	Add point status update	0	[0.998,1]
	Add point status update, remove trackside device signal	32	[0.910,0.912] or [0,0.002]

Outsider

As mentioned in Section 3.4.2, the outsider can access the communication between the trainborne system and the base stations. The least skilled outsider can jam signals and potentially delay network communications. Although such attacks can affect service availability, the attacker will not be able to affect system safety, as shown in Table 3.3.

If the communication channel is not properly secured (i.e., there are no authentication or integrity checks), an attacker can inject messages into the communication channel with a probability of 1. Even with current security practices, a skilled outsider can utilize vulnerabilities in the communication protocol and inject messages, but with a probability of 0.01, as measured in [56].

We experimented with inserting one message in which the MA was specified as the end of the track line; that would allow the train to move unhindered as if there were no obstacles. As shown in Table 3.3, insertion of just one MA message does not affect system safety, because the ATC server will continue to send MAs that will readjust the train's movement to a safe state. However, if such insertions are combined with either delay or removal of the train status messages, or removal of the MAs, the system safety is impacted.

Delay of the train status messages affects fewer configurations, since the train will only move past the first track segment and cannot proceed past the second track segment, because the MA sent to it will ensure that it stops before the second point. It causes unsafe states when the two trains move into the crossing tracks (T3 and T5) or when the train takes the wrong path from T1 to T3.

However, removal of the train status messages affects almost all input configurations except for the case in which the trains are on T1 and T6 and all the points are in the normal state. Then, the tracks are parallel to each other, and both trains move as per normal.

ATC Server

An attacker positioned at the ATC server can manipulate the MA messages sent to the trains. Unlike the outsider, the insider has full control over the MA, and thus the execution of the attack has a success rate of 1.

We consider the case when the attacker sends an MA specifying the end of the line to the train after it has stopped at the triggering point. Since the attacker does not have control over the locking of the points and the train status update messages, the attacker cannot cause an unsafe state when the two trains are on tracks T1 and T6. Because of the locking of the routes, the two trains will move in parallel with each other. However, for the other

two track configurations (i.e., T1 and T8, and T6 and T8), the routes of the trains overlap with each other, and the trains should proceed one after the other. Thus, the attack will cause either derailment of one train or a collision at the crossing.

CBI Server

An attacker positioned at the CBI server can manipulate the commands sent to the point machines. We consider the case in which the attacker drops or delays the commands sent to the point machines while continuing operations as per normal. Since the insider has full control over the commands, the execution of the attack has a success rate of 1. For the attack to succeed when it is delaying the commands, we used the model checker to determine that the minimum delay is 900 ms. The attack affects almost all track configurations unless the points have already been set in the correct position for both trains. The attacks cause the trains to derail or move onto the wrong path.

Station Network

Finally, the last attack we consider is the addition of the point status update. The attacker may be positioned at the CBI server, at the ATC server, or within the station network. If the attacker is within the servers, the attacker has full control, and thus the attack always succeeds. Otherwise, if the attacker is within the station network, the attack has only a 0.01 probability of success, because of the security mechanisms in place. The effect is similar to that of an injection of an MA from the ATC server, so this attack affects the same number of track configurations.

3.7 SAFETY COUNTERMEASURES

Several safety countermeasures may be in place in the system. In particular, we consider the following safety measure: if a train does not receive an MA within a time limit, the train will come to an emergency stop. This safety measure affects how far the train will move even after an attacker has successfully injected a single MA message.

After adding that safety measure, injection of a single MA message and removal of the train status update do not affect as many configurations as before the addition of the safety measure. To affect as many configurations as before, the attacker needs to inject more MA messages to prevent the train from coming to an emergency stop. We used the model checker to determine that the minimum number of messages that need to be injected is nine.

Also, with the addition of the safety measure, an attacker could drop MA messages to force a train to stop while he or she is injecting (or modifying) train status updates to trick the servers into believing that the train has cleared its route. We created a separate automaton that modeled the movement of an imaginary train A that accelerates faster than the actual train. The location of the imaginary train is then used in the train status updates after the train has passed the triggering point. When the servers believe that train A has cleared its route, the other train will be given the go-ahead to move. The two trains will then collide, since train A has stopped before its expected destination, unless the tracks are parallel to each other (i.e., the two trains are on T1 and T6).

Another safety countermeasure is to use secondary track detection systems that are legacy devices such as axle counters and track circuits that use physical mechanisms to detect the presence of a train on a given part of a track. The servers can correlate data sent from such trackside devices with train status update messages to check the position of a train as it enters a junction crossing.

That safety countermeasure prevents an attacker from deceiving the servers about the train position, so attacks that involve delaying, inserting, and removing train status update messages will fail. Thus, all attacks from the outsider adversary will fail, but an insider with access to either the ATC server or the station network would be able to drop the signal from such trackside devices and successfully carry out the attacks.

In conclusion, a less skilled attacker who can only jam communication signals from the train will not be able to affect system safety. However, even under current security protections, the vulnerabilities in the network protocol between the trainborne system and the server can be exploited to violate system safety. Secondary track detection systems can stop outsiders from attacking the system. However, if an attacker has gained access to the internals of the system, such as the station network or the servers, he or she can easily force the system into an unsafe state. Therefore, it is vital that the communication channel between the trains and the station be fully secured, and that good security practices be enforced within the station network to ensure that safety-critical components are isolated and well-protected.

3.8 SECURITY ANALYSIS: IDENTIFYING THE LIKELY THREAT VECTOR TO BE EXPLOITED

The results of the analyses in Section 3.6 show that there are three ways in which an attacker can affect the safety of a system: (1) inject communication messages to the train, (2) control the ATC server, or (3) cut the network connections to the CBI server. We want

to identify which of those three threat vectors would likely be exploited by an attacker. Therefore, we expand on the security analyses presented in Chapter 2 by updating the ontology and model to include the attacks we described.

3.8.1 Updated Ontology

To model the attack on communication messages to the train, we defined an attack step **HackComms** that can be performed on a **ProgrammableDevice**. The attack step will only succeed if that **ProgrammableDevice** has a *hackable* attribute. The probability that the attack will succeed is 0.01. The result of the **HackComms** attack step changes the *controlSetting* associated with the device. The cost of the **HackComms** attack is quantified in terms of the monetary cost incurred to prepare the attack; we estimate the cost to be $50,000, as calculated in [56]. Assuming that there is no secondary track detection system as a safety countermeasure, the attack will not be detected.

To model the attack on the CBI server, we defined an attack step **CutCable** that can be performed on a **ProgrammableDevice**. If a **ProgrammableDevice** d_1 has *networkConnections* with another **ProgrammableDevice** d_2, then the adversary can perform the **CutCable** attack step to change the *controlSetting* of d_2. The cost of the **CutCable** attack is the same as that of the **Damage** attack. The attack will always be detected, because it is obvious to the defender that the effect of the attack is malicious.

3.8.2 Updated System Instance Diagram

Finally, we modify our system instance diagram from Figures 2.3 and 2.4 to model the additional assets and relations that affect system safety. The modified system instance diagram is shown in Figures 3.11 and 3.12.

We add a `Laptop` device that *controls* the `Train`. The `Laptop` can be used to control the `Train` only after the **HackComms** attack has succeeded. We also add a `Points` device that represents the point machines in the system. We separate the `Signaling Server` device into the `ATS Server` and the `CBI Server`, because the two servers control different components of the system, i.e., the `Train` and `Points`, respectively. So the `ATS Server` *controls* the `Train`, and the `CBI Server` has a *networkConnection* to the `Points`.

We expand on the **TrainCollision** goal of Chapter 2 to consider attacks on the point machines. The goal can be achieved by either switching the value of the *controlSetting* state variable of either the `Train` **Device** or the `Points` **Device** from 1 (On) to 0 (Off).

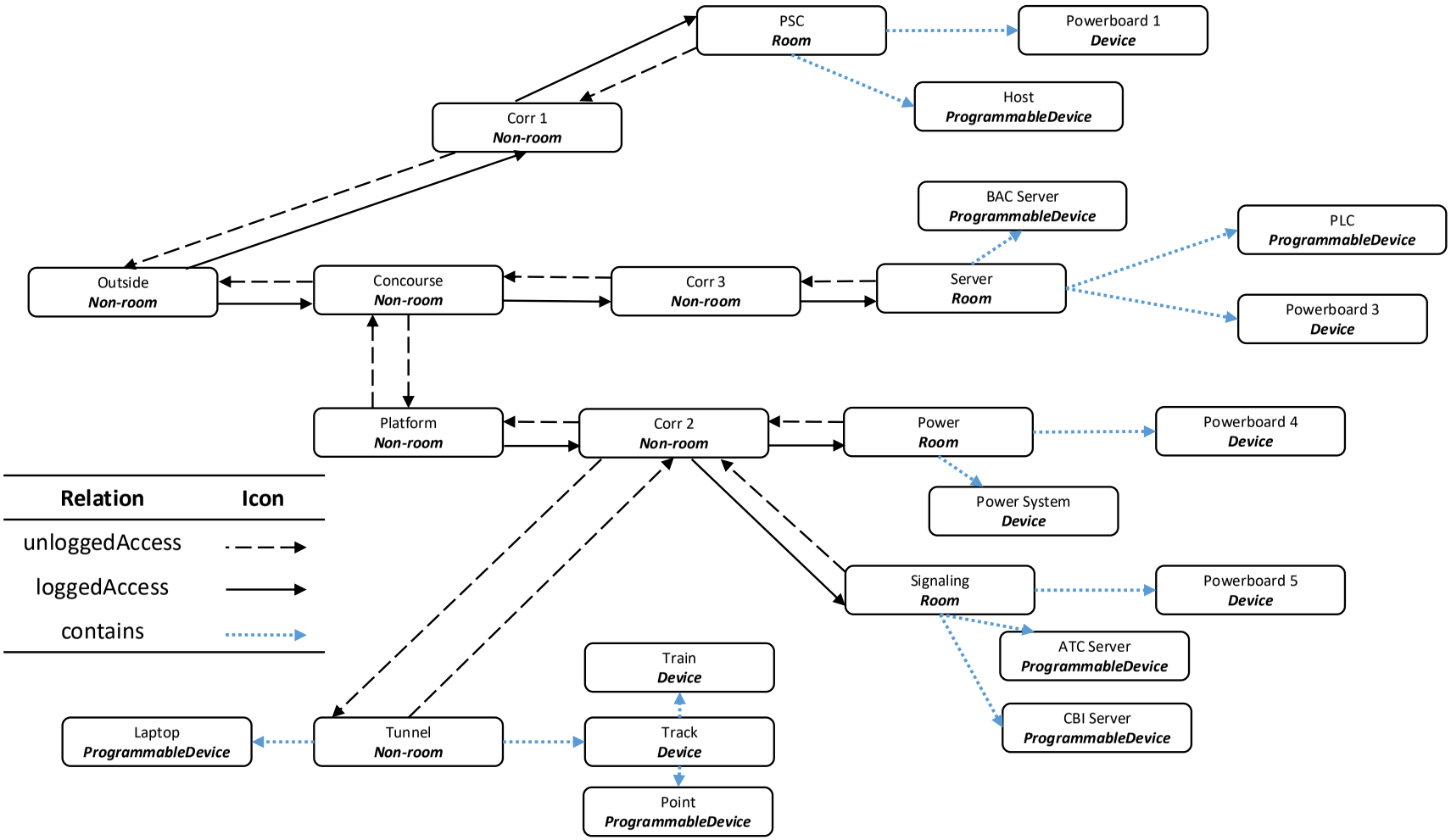


Figure 3.11: The physical station architecture view of the system instance diagram.

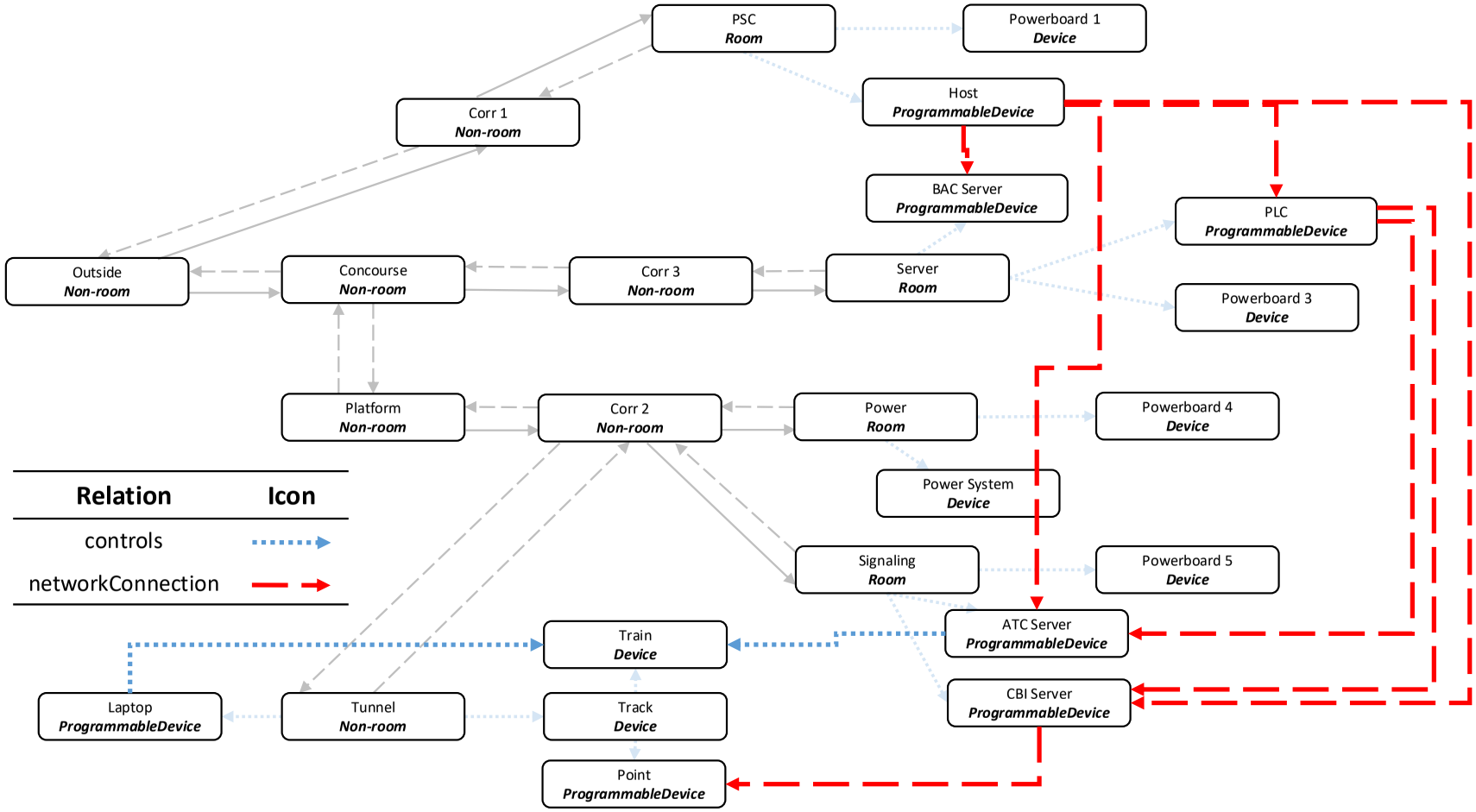


Figure 3.12: The relations among the devices in the physical station.

3.8.3 Adversary Profile

Previously, we modeled different adversaries who possess varying amounts of physical and cyber access to the system and compared their attack paths through the rooms in the station. Some of the adversaries are unable to affect system safety because they do not have the required physical or cyber access to the devices.

Instead, in this chapter, we want to determine what course of action an outsider can take to compromise system safety. Based on the results of this chapter, we define three courses of action for the adversary: (1) enter the rooms in the station through brute force, (2) bribe an insider with the appropriate physical and cyber access, and (3) inject messages to the train from public areas.

We define two additional attack steps in the ontology that represent the adversary's gaining of the appropriate skills and access to the system. We define a **BruteForce** attack step that results in the adversary's gaining of a **BreakLock** skill. The cost of the attack step is quantified in terms of the monetary costs associated with preparing the attacks. An adversary can enter the station rooms using brute force by (1) breaking directly through doors, (2) stealing a staff member's access card, or (3) forging an access card. The investment cost for the adversary includes purchase of attack tools to gain access to station rooms and power tools to access the equipment inside the rooms. We estimate the cost of the **BruteForce** attack step to be $1,000. The **BreakLock** skill gives the adversary permission to perform the **Move**, **Damage**, and **CutCable** attack steps. However, the adversary will always be detected, because staff members will notice through video surveillance that an outsider has entered staff-only areas.

We also define a **BribeInsider** attack step that results in the adversary's gaining of physical (**Space_X** access elements, where **X** is any of the three subsystems) and cyber access (**Access_X** access elements) to the system. The investment cost for the adversary consists of bribing an insider with money, so we estimate the cost of the attack step to be $500,000. Since the insider is an authorized user of the system, the adversary will not be detected when performing the **Move**, **DirectControl**, **RemoteControl**, **Login**, and **NetworkMove** attack steps.

3.8.4 Experiment Results

We want to determine what course of action the adversary will take when he or she is concerned with being stealthy. We therefore regenerated the attack graphs, using the updated ontology and system instance diagrams. The adversary's stealthiness is represented

by a preference weight that is associated with physical and financial cost. That physical and financial cost is what the adversary stands to lose if the attack campaign is detected by a defender. Table 3.4 shows the resulting metrics for each attack campaign.

We varied the preference weight from \$0 to \$5,000,000 and observed which attack paths the adversary takes. The simulation results show that for weights between \$0 and \$500,000, the adversary chose the **BruteForce** attack step and entered the **Signaling** room to perform the **CutCable** attack on the connection between the *CBI Server* and the *Points* machine. If the weight was above \$500,000, the adversary chose the **BribeInsider** attack step and used the physical access gained to enter the same **Signaling** room to **Login** to the *ATS Server* and **RemoteCtrl** the *Train*.

Our results show that when the adversary does not care about being detected, he or she will choose to use brute force to enter the station. Then, the adversary will always be detected by station operators who monitor the system using video surveillance. However, a stealthy adversary would opt to expend more money on bribing an insider to compromise the system. Although attacking train communications is less costly than bribing an insider, it is also less likely to succeed, and thus will not be chosen by the adversary. Our findings echo the opinions of railway system experts [56]. Therefore, in Chapters 4 and 5, we will look at defense countermeasures to detect abnormal behavior by insiders.

Table 3.4: Calculated metrics for each attack campaign. The different attack steps are listed.

Attack Campaign	**Cost**	**Attack Steps**
BruteForce	\$1,006	**CutCable:** CBI Server
BribeInsider	\$500,013	**Login:** ATS Server **RemoteCtrl:** Train
HackComms	\$170,220	**HackComms:** Laptop **RemoteCtrl:** Train

3.9 CONCLUSION

It is important to analyze the safety of a cyber-physical system that is under threat from different attacker models. In this chapter, we modeled the cyber-physical system by using state automata and we specified an adversary's attacks by using model templates that can be applied to a state automaton representing normal system operations. We defined the circumstances under which an attack may be carried out and represented the attack in UPPAAL.

We described a safety analysis we conducted for a railway signaling system subjected to both outsider and insider threats. We applied our attack model templates to the railway system and focused on modeling attacks on the network messages and cyber commands issued to physical devices.

Our results show how the different combinations of attack capabilities with respect to the attacker's position in the system can affect system safety. Our analysis of system safety also provides insight into which safety countermeasures can deter attacks. We also reapplied our approach from Chapter 2 to analyze which of the attacks that affect system safety would be used by a stealthy adversary. Our results show that we should focus on detecting abnormal behavior of insiders in order to prevent stealthy adversaries from compromising system safety.

CHAPTER 4: SOCIO-TECHNICAL SYSTEM DEFENSE: DETECTING ABNORMAL PHYSICAL MOVEMENT

Insider threats are a top concern of all organizations because they are common and can have severe consequences. In a recent report on IBM's Cyber Security Intelligence Index [60], insiders were found to account for 60% of the attacks launched in 2015, with 75% of those attacks involving malicious intent and the remaining 25% being inadvertent.

However, insider threats are difficult to detect, since the adversary already has physical and cyber access to the organization's assets. Many state-of-the-art research efforts [61] and state-of-the-practice tools [62, 63] focus on the cyber aspect of insider attacks by analyzing the user's cyber footprint (e.g., logins and file accesses). However, an organization's defense mechanisms are only as strong as its weakest link. By failing to consider the physical aspect of users' behavior, an organization not only leaves itself unable to detect precursor physical behavior that could facilitate future cyber attacks, but also opens itself up to less tech-savvy attacks such as vandalism and theft [64].

Thus, physical security plays a crucial role in an organization's overall defense posture. This is especially true for critical infrastructure systems such as power grids and transportation systems in which a physical breach can have major real-world effects. Therefore, in this chapter, we address insider threats in critical infrastructures by detecting potential precursor behavior of users in the physical domain. More specifically, we study the physical movement behavior of users for any suspicious patterns that may manifest as a result of malicious intent.

4.1 BACKGROUND AND MOTIVATION

To address physical security, building access controls [65] are often used to limit the areas that users can access based on their role in the organization; they normally do so through a relatively static assignment of a set of locations to the user's tracking device (e.g., RFID tag or access card). When a user moves between spaces (e.g., tapping a card at a door), information about this movement is logged.

Although building access control restricts the spaces that a user is able to access, it is merely the first step towards physical security. As with other access control solutions, it faces the same problem of being overly permissive [66]. But denying access to rarely accessed rooms is a costly solution, as it places the burden on administrators to grant every access request, which can lead to severe consequences, especially in time-critical situations (e.g., maintenance). Even with a restrictive set of granted permissions, the access control solutions

in place do not take into account the context of a user's access.

Thus, we focus on developing a more advanced behavior-monitoring capability that detects abnormalities in a user's movement within an organization's buildings. We envision that when our detection system is put into use by a company, there will be existing movement data logs that have been collected for the present users in that company. Those logs will be used as training data by our algorithm to build the user movement models. The building of the user movement models can be conducted during the installation of the intrusion detection system through feeding in of the existing data logs. Then, our intrusion detection system can be put into use immediately and be able to process door access in real time.

Currently, a railway system staff member would need to look through the physical access logs manually in order to detect malicious behavior. We aim to reduce the amount of manual effort with our detection system, which automatically presents a smaller subset of potentially malicious physical accesses in real time to the staff member. The staff member can then focus his or her attention on the smaller subset, using video surveillance to corroborate evidence of malicious activity. To aid the decision-making, we can also supplement the suspicious accesses with a model of the users' normal behavior.

In Chapter 2, we examined the effect of the addition of such defenses on the overall security posture of an organization. The results of the analysis in that section indicate that this type of defense capability will increase the strength of the organization's security. Thus, it is an important defense mechanism that deserves more attention from researchers and security practitioners.

In this chapter, we explore how physical access logs collected from a railway transit system can be used to develop a more advanced behavior-monitoring capability for the purposes of detecting abnormalities in a user's movement. In particular, we aim (1) to determine the feasibility of characterizing the movement behavior of users in a complex real-world system, (2) to develop techniques that can be applied to this detection problem, and (3) to identify ways to integrate real-time detection into physical security.

We provide a systematic approach for tackling those issues in a way that can be generalized to a diverse set of systems. We observe that since an organization consists of users who have a diverse set of roles, the movement patterns of users in different roles may vary vastly because of their job needs. Instead of proposing a single technique to model all users, we construct a methodical approach that selects the appropriate model based on the context of the organizational role and learns that model from historical data. More specifically, we propose metrics to determine the feasibility of modeling the behavior of certain users in a system. We then construct models that factor in contextual information such as time and location, and show that the model can be used in an online manner. We propose two different

models and evaluate their ability to represent normal user movement behavior by comparing the detection rates of the framework when it is using those models. This study is supported by a set of real-life card tap logs that we collected from our industrial collaborator.

In summary, the contributions in this chapter are as follows:

- We define a framework that characterizes a user's physical movement behavior and learns models of the user's behavior by using historical data. More specifically, we propose a metric based on the entropy of time series that quantifies the regularity of user movement in order to differentiate user movement behaviors. By choosing different models for users with irregular behavior instead of regular behavior, we show that integration of logs of relevant device state is needed to improve detection performance for those users.

- We propose two different Markov models to represent the physical movement of a user. These Markov models differ with respect to the amount of information they represent about the physical paths taken by users.

- We evaluate our framework using a year and two months of real-world card tap logs obtained from railway transit stations. We show that we can detect suspicious movement behavior in an online manner before the final destination is reached.

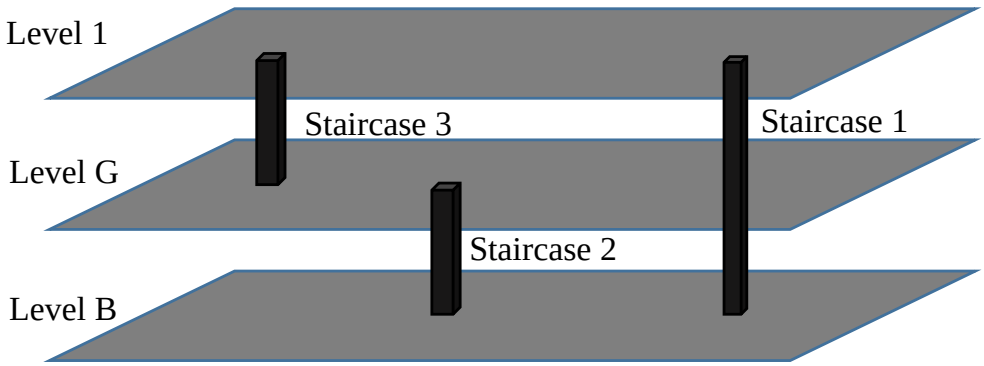
4.2 DEFINITIONS AND SYSTEM MODEL

We define a system $sys = (U, Env)$ (e.g., an enterprise organization or critical infrastructure systems) as the collection of users U who work for it, and the environment Env that contains the system's assets. The environment Env consists of both the physical and cyber domains of the system. The physical domain is composed of the building and the physical assets within it. The cyber domain consists of the networked computer system and its digital assets. The cyber and physical domains are interrelated, but we focus only on the physical domain in this chapter.

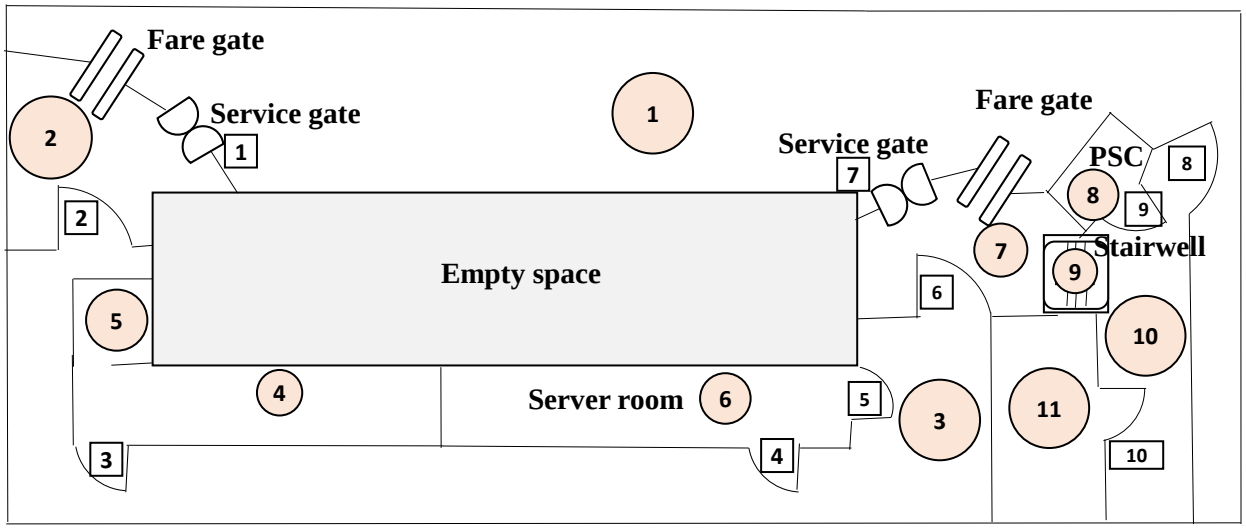
We represent the building topology as a directed graph $G = (S, E)$ in which the set of vertices represents the spaces in the building. A directed edge $e(v_1, v_2)$ represents possible movement from v_1 to v_2.[1] For example, the floor plan in Figure 4.1b is represented as the graph in Figure 4.2.

The set of spaces S can be divided into two partitioning subsets: rooms $\mathbf{R}$, and common areas $\mathbf{C}$ (e.g., staircases, corridors), i.e., $S = \mathbf{R} \cup \mathbf{C}$, and $\mathbf{R} \cap \mathbf{C} = \phi$. The edges are

[1]This implies that if $e(v_1, v_2)$ exists, the backward edge $e(v_2, v_1)$ also exists in G.



(a)



(b)

Figure 4.1: The building topology of a railway station. (a) The different levels of the railway station, with staircases connecting two or more levels. (b) A small sample floor plan of one of the levels. The PSC room is the Passenger Service Center.

labeled with the access door codes that are associated with user access. The edges can be weighted to reflect a metric quantifying the relationship between the spaces, i.e., $w : E \to \mathbb{R}$. For example, $w(e(u, v))$ can represent the physical distance between spaces u and v. In Section 4.4, we will define w more precisely and use the edge weights to determine the shortest paths between spaces.

The state of the system at time t is defined as $State_t(sys) = (L^t, Access_t, Q_t)$, where

- $L^t \subseteq S^{|U|}$ is the location of the users in the system;
- $Access_t \subseteq U \times S \times S$ is the set of accesses, where an access is a triple (u_i, s_1, s_2) or $s_1 \to_{u_i} s_2$; and

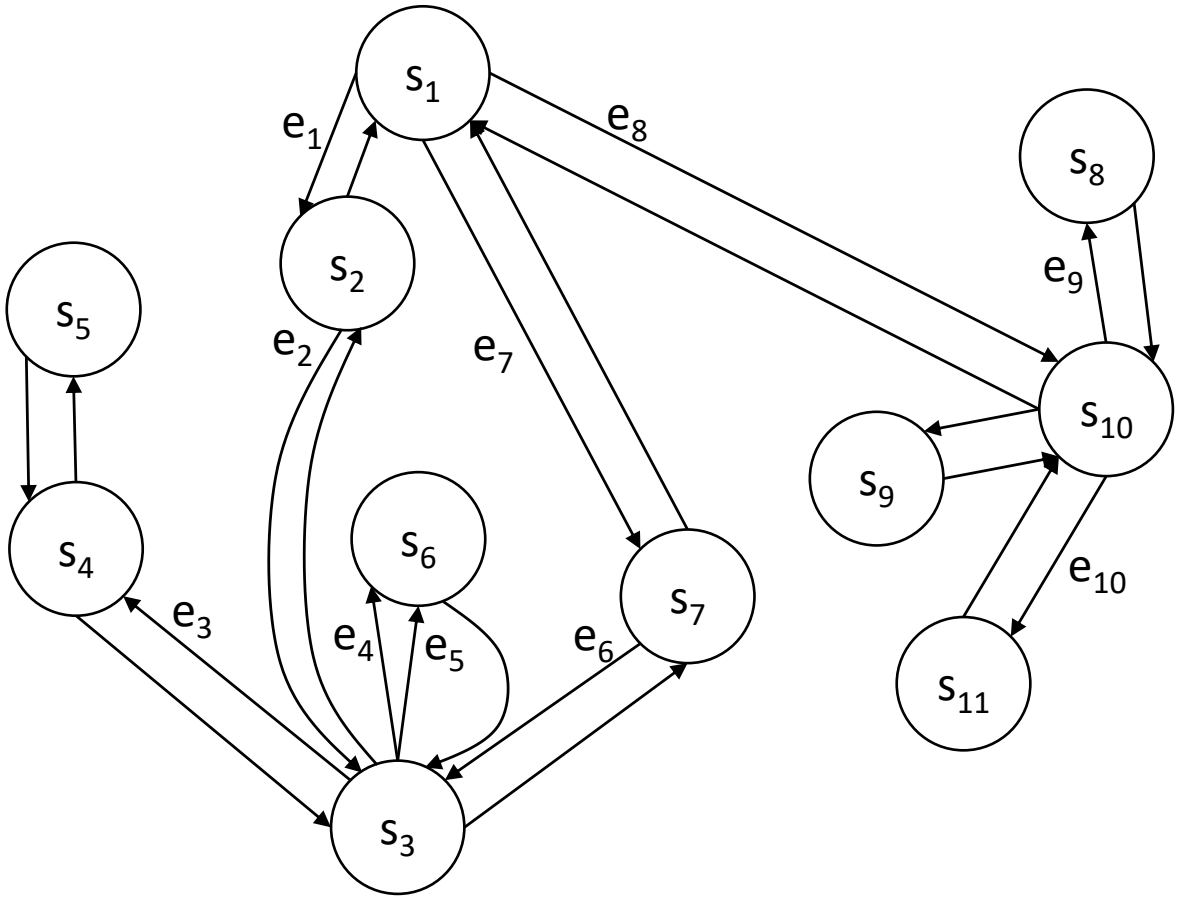


Figure 4.2: Graph representation of the railway station. Each labeled edge represents a card reader on the door separating the spaces (vertices).

- Q_t is the state of the environment, including the condition of the physical and cyber topology and assets (e.g., malfunctioning devices, changes in networking access).

In our case study, a railway station consists of a single building that may house one or multiple railway lines through it. The general public accesses the railway lines by passing through fare gates in the concourse area and moving to the platform. Figure 4.1 depicts the topology of the railway station in our case study.

In addition to the concourse and platform area, the railway station contains many rooms hidden from the public eye that house the equipment necessary to maintain the running of the station and its portion of the railway track. The rooms are distributed throughout the station on multiple levels. The railway staff can access those spaces only by tapping their access cards at readers on the doors. Although most of the doors inside the staff-only spaces have card readers, there are a number of doors that allow free access. Different stations have different floor plans, and the number of rooms within a station may vary. However, all the

stations share the same types of rooms (e.g., power supply room).

Multiple rooms may share the same functionality. For each room, we label the corresponding vertex with the subsystem associated with it, $Subsys(v) = \{OP, POW, EC, TM, ME\}$, where the description of each subsystem is as follows.

OP (Station operation control): e.g., the *Passenger Service Center* (PSC), server room, office, and storeroom.

POW (Power control): e.g., traction power room and *Uninterrupted Power Supply* (UPS) room.

EC (Environment control): e.g., water chiller room and ventilation fan room.

TM (Transport mechanical equipment control): e.g., escalator rooms and platform screen door room.

ME (Mechanical equipment): e.g., lighting control room and pump room.

Unlike an enterprise system for which the office building has a simple, systematic layout across all levels (e.g., a single corridor branching out to multiple rooms), a railway station has a complex, asymmetrical layout. There are multiple paths with varying lengths that a user can take to get from one room to another. That implies that topology is an important factor in determining whether a user's physical movement is anomalous. In addition, the railway transit system has diverse user roles (e.g., station operators and power maintenance staff). The job scopes of such users vary in terms of work shifts, responsibilities, and work locations, all of which affect their physical movement behavior. Even users in the same role exhibit different movement behaviors based on their assigned duties and personal habits.

The building access control system that is in place offers a limited view of users' physical movement. Since card readers may fail and certain doors are not outfitted with card readers, we are unable to determine a user's full movement trajectory. A user may also tailgate, i.e., follow closely behind, another user, and thus the access will remain invisible to us. (We tackle that problem in Chapter 5.) Therefore, it is challenging to detect deviations in a user's movement behavior. We address the problem in the next section by integrating knowledge of the system layout and by learning models of users' behaviors from historical card tap data.

4.3 MALICIOUS MOVEMENT DETECTION FRAMEWORK: OFFLINE PHASE

In this section, we describe our framework that systematically analyzes users' physical movement logs to detect malicious insiders. The framework dissects the problem into three parts: understanding the characteristics of users' behaviors, learning a suitable model representation, and using the model together with knowledge of the system layout to estimate the probability of an abnormal access. The framework consists of an offline and an online phase, as shown in Figure 4.3.

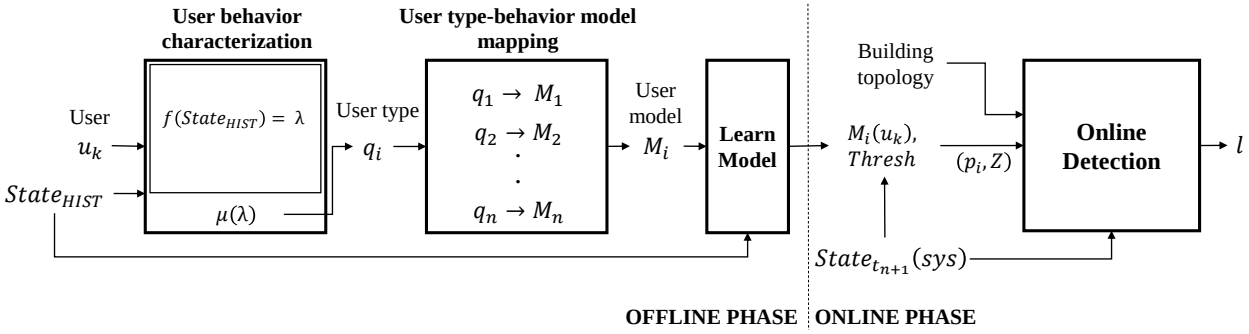


Figure 4.3: The framework is divided into offline and online phases, where the models learned from the offline phase are fed into the online phase.

The offline phase consists of two stages: characterization of users based on their past movement behavior, and construction of models based on users' characteristics and past movement. The input to this phase is the training data that describe the historic system states $State_{Hist} = State_{t_{-1}}(sys) \ldots State_{t_{-n}}(sys)$, and the output of this phase is a collection of tuples $(\mathcal{M}, \textbf{Thresh})$, in which $\mathcal{M}$ is a model representing the movement behavior of a user and $\textbf{Thresh} : Access_{Past} \rightarrow \mathbb{R}$ is a function that takes in a set of accesses and returns a real number that represents the threshold value, below which the probability score of a movement will be considered anomalous.

4.3.1 User Types

The first stage of the offline phase is to distinguish between different users by using their past movement behavior. Typical access control systems assign roles to users based on the sets of rooms that they need to access. However, these roles do not directly reflect the user behavior. Instead, we propose to categorize users according to how they move within a building.

We define the different types of user behavior $\mathbb{T}$ based on the users' "reasons" for movement, where "reason" refers to the context that facilitates users' movement patterns. These

reasons can be inferred from the structure and numerical values of the historical system state pertaining to the users. We define a function $f : State_{Hist} \rightarrow \mathbf{F}$ that extracts features from the historical system state. We then calculate a function $\mu : \mathbf{F} \rightarrow \mathbb{T}$ over the feature vector returned by f and output the user type or "reason."

Application: In our railway station case study, there are two main types of user movement behavior, $\mathbb{T} = \{q_1, q_2\}$. The first type, q_1, involves users who have a very regular movement behavior, and the primary members are station operators. Station operators work a fixed set of hours in the station, and, because of their job scope, their movement patterns are fairly consistent. They remain in the *Passenger Service Center* (PSC) to assist the public and monitor the state of the station, visit storerooms and staff rooms, and clock in and out.

The second type of users, q_2, is those whose movement is triggered when an event occurs. This applies to maintenance staff who visit rooms to conduct maintenance and repairs on the equipment. Different maintenance staff members are in charge of different subsystems (e.g., power supply or signaling), and thus they access different sets of rooms in the station.[2]

In order to categorize a user into q_1 or q_2, we extract a feature based on the approximate entropy of a time series [67] constructed using the collected historical access data, or training data, $f(Access_{Hist}) = ln(C_m/C_{m+1}) = \lambda_e \in \mathbf{F}$, where C_m is the prevalence of repetitive patterns of length m in the time series. Each subsequence of length m in the sequence is compared to other subsequences. If the number of similar subsequences is high, then C_m is large. This metric has been shown to be able to quantify the predictability of user movement [68, 69]. We choose m to be 3, which provides a good metric for characterizing our trace as shown in Section 4.6.

If the user's entropy value is lower than $\mathbf{E}$, then the user has more predictable movement. Thus, the user belongs to q_1; otherwise, the user belongs to q_2. In other words,

$$\mu(\lambda_e) = \begin{cases} q_1 & \text{if } \lambda_e < \mathbf{E} \\ q_2 & \text{otherwise} \end{cases} \tag{4.1}$$

where $\mathbf{E}$ is a numerical threshold. The choice of parameter $\mathbf{E}$ is discussed in Section 4.6.

[2]The same principle may apply to other systems too. E.g., a security guard doing rotations in a building belongs to q_1, and a technical support staff member who goes to an office when his or her assistance is required belongs to q_2.

4.3.2 User Behavior Models

Next, we construct behavior models for each behavior type $q \in \mathbb{T}$ defined earlier. Since each user is motivated to move within the building for different reasons, it is not possible to specify a single model for all users' behavior. Such a model would be inherently biased towards a certain set of users and perform badly for others.

Instead, for each $q \in \mathbb{T}$, we select an appropriate modeling technique $\mathcal{M} \in \mathbb{U}_{\mathcal{M}}$ from a large set of possible modeling techniques $\mathbb{U}_{\mathcal{M}}$. The model should leverage q's distinct characteristics and provide insight into the likelihood that a user will access a room given the current system state $State_{t_{n+1}}(sys)$.

For each user u of type $q \in \mathbb{T}$, we learn the model by analyzing the historic system states $State_{Hist}$ described by the training data in order to assign probabilities to the rooms in $\mathbf{R}$. Finally, we describe the construction of the **Thresh** : $Access_{Past} \rightarrow \mathbb{R}$ function in Section 4.4 for ease of understanding.

Application: Users belonging to q_1 have a lower entropy value $\lambda_e < \mathbf{E}$ than those belonging to q_2. This implies that q_1 users' movement patterns are more predictable and repetitive than those of q_2 users'. Thus, we choose to represent a user's movement behavior with a Markov model.

Given the historic system states $State_{Hist}$ (obtained from the training data), we learn the Markov model of a user u. We can reconstruct the full movement sequence $Seq_{Hist}(u) = S_1 \dots S_n$ from $Access_{Hist}$. The sequence $Seq_{Hist}(u)$, or $Seq(u)$ for short, can be divided into segments such that a period of inactivity (of more than 3 hours) separates any two segments, i.e., $Seq(u) = Seg^1_{1,j} Seg^2_{j+1,k} \dots Seg^m_{p,n}$ where $\sum_{i \in \{1,\dots,m\}} |Seg^i| = n$ and $\forall i \in \{1, \dots, m\}, Seg^i = S_g \dots S_h, Seg^{i+1} = S_{h+1} \dots S_j$; then, the accesses $(u, S_{h-1}, S_h) = a_t$, $(u, S_h, S_{h+1}) = a_{t+T}$ imply that $T \geq 3$ hours.

There are many ways to represent that movement sequence as a Markov model. In this dissertation, we present two different ways to express a user's movement sequence in the form of a Markov model. Specifically, we define (1) a Markov model that captures only the sequence of rooms that are visited, $\mathcal{M}_r$, that is the model defined in [70]; and (2) a Markov model that, in addition to the visited rooms, also captures the sequence of common areas visited on the path to those rooms, $\mathcal{M}_s$.

The model $\mathcal{M}_r$ is less restrictive than $\mathcal{M}_s$ because, unlike $\mathcal{M}_s$, it does not describe the likelihood of a user's favoring different paths to a room. Thus, $\mathcal{M}_r$ allows more freedom for a user to move within a building without having his or her movement flagged as suspicious. (E.g., a user could take a different path to the staff room because he or she stopped by a store

in the opposite direction.) However, such freedom comes at the cost of potentially missing malicious movement that deviates from the user's normal path. For example, an attacker may choose to take a path that is less populated to avoid meeting other staff members. We will see in Section 4.6 how these two models, $\mathcal{M}_r$ and $\mathcal{M}_s$, compare in terms of false positive and false negative rates.

We define the Markov model $\mathcal{M}_r$ as having:

- a state space that is the set of rooms $\mathbf{R}$,
- a transition probability matrix P with the entry $p_{ij} = \frac{\#r_i c_1 \ldots c_n r_j \in Seq}{\#r_i \in Seq}$ as the normalized frequency with which the user visits r_i and then r_j, and
- an initial probability vector π with $\pi_i = \frac{\#Seg_{jk} = (c_j \ldots c_n r_i \ldots)}{\#Seg}$,

where $c_1 \ldots c_n \in \mathbf{C}$.

On the other hand, the states in the Markov model $\mathcal{M}_s$ are represented by a tuple (r_{prev}, s), where r_{prev}, similar to the state in $\mathcal{M}_r$, is the previous room visited, and $s \in S$ is the space that the user is currently in. For example, a movement sequence $c_0 \ldots c_n r_0 c_{n+1} \ldots c_m r_1$ where $c_i \in \mathbf{C}, i \in \{0, \ldots, m\}$ can be expressed as the following state transitions: $(\cdot, c_0) \rightarrow (\cdot, c_1) \cdots \rightarrow (\cdot, r_0) \rightarrow (r_0, c_{n+1}) \rightarrow (r_0, c_{n+2}) \cdots \rightarrow (r_0, r_1)$. We therefore define the Markov model $\mathcal{M}_s$ as having:

- a state space that is the set of tuples $(\mathbf{R}, S)$,
- transition probability matrix P with the entry $p_{ij} = \frac{\#r_i c_1 \ldots c_n s_k s_j \in Seq}{\#r_i c_1 \ldots c_n s_k}$ as the normalized frequency with which the user is in state $i = (r_i, s_k)$ and then $j = (r_m, s_j)$, and
- an initial probability π with $\pi_i = \frac{\#Seg^j = (s_i \ldots)}{\#Seg}$.

However, the users belonging to q_2 have less regular movements and may change movement patterns based on events in the system. We therefore combine the Markov model with additional contextual knowledge about the states of the devices in the rooms. More specifically, we annotate each state associated with a room with a Boolean variable indicating whether a device failure has occurred within that room.

4.4 MALICIOUS MOVEMENT DETECTION FRAMEWORK: ONLINE PHASE

The online phase involves determining, based on the behavior models derived from the offline phase, whether a user's access is an abnormality. We thus advance the state of the art

of physical security systems by attempting to detect anomalous accesses as early as possible so that the appropriate response actions, such as blocking access to a room, can be taken. The inputs to this phase are the tuple $(\mathcal{M}, \textbf{Thresh})$ from the offline phase and the current state of the system $State_{t_{n+1}}(sys)$. The output of this phase is a real number in $\mathbb{R}$ that indicates the degree of abnormality of the access.

The current system state $State_{t_{n+1}}(sys)$ includes the location of a user u_i, $L_i^{t_{n+1}} = s_1 \in S$ and the physical access that is being made, $A = (u_i, s_1, s_2) \in Access_{t_{n+1}}$. In other words, the user is moving from s_1 to s_2. Using our knowledge of the building topology, the behavior model $\mathcal{M}$, and the current state, we can calculate a probability that the current access fits into the user's normal movement behavior. That probability score is compared to the threshold value returned from the application of the **Thresh** function on the past accesses $Access_{Past} = Access_{t_0} \ldots Access_{t_n}$. If the probability score falls below the threshold value, then the access is marked as anomalous.

Application: During the online phase, we keep track of certain properties of the past system state. For the Markov model $\mathcal{M}_r$, we keep track of the room that the user has last accessed: $r_L \in \mathbf{R}$, where $Seq_{Past} = s_1 \ldots r_L c_1 \ldots c_m$. Then, based on the transition probability matrix, we can extract the set of probabilities associated with the next possible visited room $NextRoom = \{r_i \in \mathbf{R} | p_{Li} > 0\}$. Since the Markov model $\mathcal{M}_r$ does not directly assign transition probabilities to the non-room spaces, we instead derive the probability score of an access by determining whether the user's movement is bringing him or her closer to the next possible room to be visited.

In other words, given the access A, we want to determine all the rooms that the user is likely to access. We first find all the rooms that are reachable from s_2, i.e., $P_T = \{r_i \in NextRoom | \exists (s_2, s_i, \ldots s_{i+m} r_i)\}$, where $(s_2, s_i, \ldots s_{i+m} r_i)$ is a sequence of vertices on the path from s_2 to r_i. For all such vertices $r_i \in P_T$, we decide whether the user is likely to access r_i by moving to s_2 from s_1. If it's easier to access r_i from s_2, then we consider r_i one of the rooms that u_i is likely to move to next.

To decide whether r_i is easily accessed, we calculate path lengths by using the weights on the edges, i.e., $d(s_i, s_j) = \min_{\forall (s_1 \ldots s_m)} w(e(s_i, s_1)) + w(e(s_m, s_j)) + \sum_{k=1}^{m-1} w(e(s_k, s_{k+1}))$. We calculate the shortest path from s_1 to r_i, $d(s_1, r_i)$, and compare it to the shortest path from s_1 to r_i through s_2, $d(s_1, s_2) + d(s_2, r_i)$. If the shortest path through s_2 is similar in length to the shortest path $d(s_1, r_i) \approx d(s_1, s_2) + d(s_2, r_i)$, then we consider r_i a possible room that the user wants to access.

In this dissertation, we choose a simple assignment of weights to edges. That assignment of weights can easily be changed depending on users' preferences for taking certain types of

paths. We assign all edges a weight of 1, with the exception of edges that connect different levels of the building (i.e., staircases, elevators, and escalators). We assume that users prefer to take as few staircases as possible, so we assign a weight of 10 to those edges that connect different levels.

With the resulting shortlisted set of rooms, we sum up the likelihoods of those rooms $\sum_{r_i \in P_T} p_{Li}$ to obtain a final score.

Although we describe the construction of the **Thresh** function in this section for ease of understanding, the construction is actually performed in the offline stage alongside the construction of the movement models. The **Thresh** function maps the past accesses $Access_{Past}$ (or, more specifically, a property of the past accesses that is kept track of in the online phase) to a threshold value, below which the probability of a movement is considered anomalous. For the Markov model $\mathcal{M}_r$, **Thresh** takes in the room last accessed, r_L, and returns $percentile_x(p_L)$, i.e., $\textbf{Thresh}(r_L) = percentile_x(p_L)$. The reasoning behind that threshold is that since there are rooms that the user may seldom visit, transitions through them may be deemed anomalous. We therefore take as the threshold a certain percentile value, x, of the probability distribution associated with the room r_L, p_L. The percentile value can be changed by practitioners based on the system requirements; a higher value reduces the false positives but potentially allows malicious movements to be missed, while a lower value catches more malicious movements but increases the false positives. We compare the probability score to the threshold value returned by the function $Z = \textbf{Thresh}(r_L)$, and if the score is below Z, access A is deemed anomalous. The algorithm for that phase is given below in Algorithm 4.1.

Algorithm 4.1 OnlineDetection algorithm for $\mathcal{M}_r$ Markov model

Require: $(L_i^{t+1}, A = s_1 \rightarrow s_2) \in State_{t+1}, r_L = Property(Access_{Past})$
  function OnlineDetection($State_{t+1}, \mathcal{M}_r, Thresh, Property(Access_{Past})$)
    $score \leftarrow 0; Z \leftarrow Thresh(r_L)$
    $NextRoom \leftarrow \mathcal{M}_r(r_L)$
    for all $r_i \in NextRoom$ **do**
      $shortestlen \leftarrow GetShortestPath(s_1 \rightarrow r_i)$
      $len \leftarrow GetShortestPath(s_2 \rightarrow r_i) + w(e(s_1, s_2))$
      if $len < shortestlen \times k$ **then** $score \leftarrow score + p_{Li}$ **end if**
    end for
    if $s_2 \in \mathbf{R}$ **then** $r_L \leftarrow s_2$ **end if**
    if $score < Z$ **then return** *Anomaly* **end if**
  end function

For the Markov model $\mathcal{M}_s$, instead of calculating the probability of the current access, we calculate the probability that the sequence of movement steps is within normal user movement behavior. In other words, for a sequence of Markov states $b_0 \rightarrow b_1 \cdots \rightarrow b_n$, the probability of such a sequence is $\prod_{i=0}^{n} p_{b_i b_{i+1}}$. Since we are calculating that probability in an online fashion, we only need to keep track of the previous accumulated product, $\prod_{i=0}^{n-1} p_{b_i b_{i+1}}$, and the previous state b_{n-1}.

The **Thresh** function, in that case, takes in the length of the movement sequence, in addition to the starting state $b_0 = (r_0, s_0)$. For each state $b = (r_i, s_j)$ in the transition probability matrix, we use the movement sequence from $Access_{Hist}$, obtained from the training data, to determine the typical probability distribution of a user who starts in state b and moves k steps within the building. More precisely, $\textbf{Thresh}(b_0, k) = percentile_x(\{\prod_{i=0}^{k-1} p_{b_i b_{i+1}} | b_o \ldots b_k \in Seq_{Hist}\})$. In the next section, we vary the percentile value to examine the trade-off between the false positive and true positive rates for a given Markov model.

Then we can compare the threshold value returned by the function $Z = \textbf{Thresh}(b_0, k)$ with the accumulated probability score for the current access. If the probability score is below the threshold Z, then the access is deemed anomalous.

However, since our simple algorithm relies on the accumulation of probabilities, the presence of low probability values earlier in the movement sequence will directly affect the probability score of future accesses. In other words, the appearance of anomalies in the movement sequence would cause the probability of the movement chain to drop below the threshold even after the user's movement returns to normal, thus falsely marking future accesses as being equally anomalous.

To prevent that phenomenon, we divide the accumulated probability score (and threshold value) by the path length, i.e., $\frac{1}{k} \prod_{i=0}^{k} p_{b_i b_{i+1}}$. This division allows us to average out the effect of the low probability values.

However, even with that division, the probability of the movement chain after the anomalous movement cannot immediately bounce back to above the threshold value. Instead, it slowly increases in value. To prevent the low-probability values from influencing future accesses in the movement chain, we restart our calculations and start a new accumulation of a movement chain immediately after the anomalous accesses. To pinpoint the access immediately after the anomalous accesses, we compare the previous probability score $score_k$ with the current access's probability score $score_{k+1}$. If $score_k < score_{k+1} < Z$, that implies that the transition probability associated with the current access is high enough to cause the average to increase, so the current access is most probably not anomalous. The algorithm using the Markov model $\mathcal{M}_s$ is given below in Algorithm 4.2. Both the algorithms for M_r

and M_s are of complexity $O(1)$.[3]

Algorithm 4.2 OnlineDetection algorithm for the $\mathcal{M}_s$ Markov model.

Require: $(L_i^{t+1}, A = S_1 \rightarrow S_2) \in State_{t+1}, (r_L, score_{prev}, chainlen, prevState, b_0) = Property(Access_{Past})$

function OnlineDetection($State_{t+1}, \mathcal{M}_s, Thresh, Property(Access_{Past})$)

$currState \leftarrow (r_L, s_2)$

$Z \leftarrow Thresh(b_0, chainlen + 1)$

$score \leftarrow \frac{(score_{prev} \times chainlen) \times \mathcal{M}_s(prevState \rightarrow currState)}{chainlen+1}$

if $score < Z, score \leq score_{prev}$ **then return** Anomaly **end if**

if $score < Z, score > score_{prev}$ **then**

$score_{prev} \leftarrow 0$

$chainlen \leftarrow 0$

$r_L \leftarrow \phi, prevState \leftarrow \phi$

$b_0 \leftarrow currState$

else

$score_{prev} \leftarrow score$

$chainlen \leftarrow chainlen + 1$

if $s_2 \in$ **R then** $r_L \leftarrow s_2$ **end if**

$prevState \leftarrow currState$

end if

end function

Finally, for the q_2 users, we include another step in the online detection algorithm that correlates the remaining suspicious accesses with logs about the device state after all the accesses have been vetted through the Markov model. One can see intuitively that if a device in room R_d fails and then a physical access into R_d is logged, that physical access should be considered non-malicious. Therefore, when a device failure in a room is logged, we update the Boolean variable of the state in the Markov model that is associated with that room to indicate that movement into that room may be impending. Then, when an access is made to that room, we run it through the OnlineDetection algorithm. If it returns with a suspicion alert, then we check whether the Boolean variable associated with the room is checked. If so, then we deem that access as non-malicious.

[3]For Algorithm 4.1, we can precompute the probability scores for each space in the offline phase, thus removing the complexity of computing shortest paths during real-time analysis.

4.5 EMPIRICAL ANALYSIS OF DATA

We use a real-world data set containing physical accesses to a railway station in a city. The data set reflects a period from June 2016 to August 2017. The station has 62 rooms, and the data include a total of 141,357 card taps (or accesses) made by 590 users. While we focus on one station in this dissertation, the whole railway line consists of 33 stations, 12 of which are interchange stations. We estimate that the average number of card taps per hour over all the stations is approximately 450, whereas the highest number of card taps per hour is around 1,200. Those high numbers pose a significant challenge if the associated logs need to be examined manually.

The data set contains the following information regarding physical accesses: (1) date and time, (2) door code, (3) user identification, and (4) type of access (legal entry, failed entry, clock-in, or clock-out). When the access is a failure, that implies either that the user's card had expired or that the user did not have permission to access the room. Those failed accesses serve as ground truth for known abnormal accesses. Clock-ins and clock-outs represent the users' card taps at a time punch clock inside a room in the station.

In explaining the design of our detection framework and user behavior model in Section 4.3, we made a number of claims about the spatial and temporal properties of user movement behavior. In this section, we substantiate those claims by analyzing the physical movement data set, and justify our modeling choices.

4.5.1 Movement Independence

First, we justify our decision to model user movement with first-order Markov chains. We make the following two claims in support of our decision: (1) a user's next movement (or access) is well-predicted using only recent movement history (i.e., that user movement behavior is Markovian); and (2) it is sufficient to use just the current state to determine a user's subsequent movement (i.e., the Markov models should be *first-order*).

We substantiate both claims by using chi-square tests for Markov chain analysis as described in [71]. We ran a multi-stage test in which each n-th stage attempted to reject the null hypothesis that the data conformed to a Markov chain of order higher than n. Acceptance of the null hypothesis for the n-th test would signify that a user's subsequent movement depended only on the previous n states in the user's Markov chain (inclusive of the current state), and rejection of the null hypothesis would signify that there was likely some higher-order dependence in the data. More precisely, we have three null hypotheses:

$$H0 : \chi^2 = \sum_{x_t, x_{t+1} \in \mathbf{ST}} \frac{(N(x_t, x_{t+1}) - \frac{N(x_t)N(x_{t+1})}{n-1})^2}{\frac{N(x_t)N(x_{t+1})}{n-1}} \tag{4.2}$$

$$H1 : \chi^2 = \sum_{x_t, x_{t+1}, x_{t+2} \in \mathbf{ST}} \frac{(N(x_t, x_{t+1}, x_{t+2}) - \frac{N(x_t, x_{t+1})N(x_{t+1}, x_{t+2})}{N(x_{t+1})})^2}{\frac{N(x_t, x_{t+1})N(x_{t+1}, x_{t+2})}{N(x_{t+1})}} \tag{4.3}$$

$$H2 : \chi^2 = \sum_{x_t, x_{t+1}, x_{t+2}, x_{t+3} \in \mathbf{ST}} \frac{(N(x_t, x_{t+1}, x_{t+2}, x_{t+3}) - \frac{N(x_t, x_{t+1}, x_{t+2})N(x_{t+1}, x_{t+2}, x_{t+3})}{N(x_{t+1}, x_{t+2})})^2}{\frac{N(x_t, x_{t+1}, x_{t+2})N(x_{t+1}, x_{t+2}, x_{t+3})}{N(x_{t+1}, x_{t+2})}} \tag{4.4}$$

where $\mathbf{ST}$ is the state space of the Markov chain, $N(x_i, \ldots, x_{i+m}) = |\{t | X_t = x_i, \ldots, X_{t+m} = x_{i+m}\}|$, and X_t represents the t-th state of the user movement sequence Seq_u. The first null hypothesis $H0$ tests for independence between the Markov states. Null hypothesis $H1$ tests for first-order dependence, and null hypothesis $H2$ tests for second-order dependence. We expect that each user's movement sequence should reject $H0$, thus implying that there is dependence on recent movement history (Claim 1), but accept $H1$, thus implying that subsequent movements are best predicted using only the current state (Claim 2).

For each user, we convert his or her historic movement sequence Seq_{Hist} into a sequence of Markov states, X_t. In the case of the Markov model $\mathcal{M}_r$, that would be a sequence of rooms that the user accesses. In the case of the Markov model $\mathcal{M}_s$, that would be a sequence of (room,space)-tuples. Then, we calculate the chi-squared values for each of the three null hypotheses and based on the rejection or acceptance of the hypothesis, reach a decision on which order of Markov chain statistically fits the user's movement sequence. More precisely, the order of the user's Markov chain is

$$n = \begin{cases} 0, & \text{if } p_0 > 0.15 \\ 1, & \text{if } p_0 \leq 0.15, p_1 > 0.01 \\ 2, & \text{if } p_0 \leq 0.15, p_1 \leq 0.01, p_2 > 0.01 \\ \geq 3 & \text{otherwise} \end{cases} \tag{4.5}$$

where p_0, p_1, p_2 are the p-values obtained from the three chi-squared tests, respectively.

The results in Table 4.1 show that when we examine users' room movements ($\mathcal{M}_r$ Markov models), we find that over 84% of users are best modeled by zero-order or first-order models, and when we examine (room,space)-movements ($\mathcal{M}_s$ Markov models), we find that over 95% of users are best modeled by zero-order or first-order models. Note that any user best modeled by a zero-order model (i.e., with no dependence between movements) can be accurately

modeled by a first-order model that is a complete graph with identical next-state transition probabilities from each state. Thus, both of our assumptions hold for the vast majority of users. We also find that the few users who exhibit second-order and higher dependence in their $\mathcal{M}_s$ models have a high entropy, with a minimum value of 1.04, which indicates that their movement behavior is relatively aperiodic and therefore unlikely to achieve good performance with our approach.

Table 4.1: The percentage of users who fit an n-th order Markov model $\mathcal{M}_r$ and $\mathcal{M}_s$.

		Order			
		0	**1**	**2**	≥ 3
Model	$\mathcal{M}_r$	19.0	64.6	10.9	5.5
	$\mathcal{M}_s$	7.5	89.1	1.4	2.0

4.5.2 Consistency of Movement Paths

In this subsection, we justify our decision to model the sequence of spaces (or the path) taken by users in $\mathcal{M}_s$. We claim that users tend to follow a small set of distinct paths, and thus those paths are a property or feature of their normal movement behavior; any deviation from those paths should be considered anomalous. We would like to show empirically that users habitually follow the same path when moving from a given room to another, only occasionally choosing a different path.

For each user, we group his or her historic physical accesses Seg_{Hist} by the sequence of doors visited between two rooms. In other words, we obtain tuples of $(u_i, r_j, r_k), u_i \in U, r_j, r_k \in \mathbf{R}$, and each tuple is associated with the collection of trips made between room r_j and r_k by u_i, $Trip(u_i, r_j, r_k) = T^i_{j,k} \subseteq P$. We define a trip as the sequence of doors that are accessed, $p = (d_1, d_2 \ldots d_n) \in P, d_i \in DoorCode$, and two trips are equal only if their sequences are exactly the same. We can then define the collection of distinct trips for each tuple $UniqTrip^i_{j,k} = \{p_1, p_2 \in T^i_{j,k} | \forall p_1, p_2 \in UniqTrip, p_1 \neq p_2\}$.

In total, there are 2,273 (u_i, r_j, r_k)-tuples. For each tuple, we calculate the following:

- $Overlap = \frac{|\{d \in p_i \cap p_j | p_i, p_j \in UniqTrip, i \neq j\}|}{|\{d \in p_i | p_i \in UniqTrip\}|}$, the fraction of doors that are shared among the unique trips,
- $FrequentedTrip = \{p \in UniqTrip \,|\, \frac{|t \in T | t = p|}{|T|} > \frac{1}{|UniqTrip|}\}$, the unique trips that appear more than average, i.e., the unique trips that appear more than $\frac{1}{x}$ times, where x is the total number of unique trips; and

- $ConcentratedTrip = \frac{1}{|T|} \sum |\{t|t = p, p \in FrequentedTrip\}|$, the fraction of all trips that are in $FrequentedTrip$.

We divide the 2273 (u_i, R_1, R_2)-tuples into three categories:

- Tuples with $|UniqTrip| \leq 5$,
- Tuples with $|UniqTrip| > 5$ and $ConcentratedTrip > 0.8$, and
- Tuples with $|UniqTrip| > 5$ and $ConcentratedTrip \leq 0.8$.

The first category is tuples that have fewer than 5 unique trips. We find that 96.9% of the tuples fall under this category. The average of $Overlap$ over those tuples is 0.42, which implies that the unique trips do share common doors that are accessed.

The second category is tuples for people who have more unique trips, but for whom more than 80% of the total trips are associated with a small subset of unique trips. There are 27 tuples in the second category. All of them have $|FrequentedTrip| \leq 5$, except for one tuple with $|FrequentedTrip| = 6$ and one with $|FrequentedTrip| = 7$. Thus, fewer than 7 unique trips account for a majority of the trips. Those unique trips also have a higher average overlap, $Overlap = 0.74$. That implies that although there are more unique trips than in the first category of tuples, the paths are quite similar.

Finally, the third category consists of tuples whose trips are more varied. It contains 44 tuples. For those tuples, $FrequentedTrip \leq 4$, and those trips that are not in $FrequentedTrip$ account for less than 16% of the total trips T. Even though there is a larger spread of unique trips, the average overlap between those trips is high, $Overlap = 0.72$. Thus, the unique trips were very similar and differed only in terms of the sequence or repetition of some accessed doors. Of the 44 tuples, only 4 have a lower overlap, $Overlap \leq 0.5$. For those tuples, the less frequented trips $p \notin FrequentedTrip$ appeared only once or twice; so those unique trips were very rarely taken.

In conclusion, our analysis informs us that users tend to take a small set of unique paths, and when we delved into the sequence of doors accessed, we found that those unique paths share a high overlap. Thus, the paths that users take can be characterized as part of their normal movement behavior patterns, and any deviations from those paths would be empirically anomalous.

4.5.3 Temporal Behavior

Finally, we examine the temporal behavior of users' physical movements to determine whether such information can be used in tandem with spatial behavior to characterize users'

movement sequences.

First, we examine the time of day of users' accesses to test for the presence of time-of-day patterns in the users' movement behavior that could be used to detect anomalous movements. Figure 4.4 shows the distribution of all the users' accesses over the time of day. We see that the accesses are spread throughout the day, with multiple small surges of activity at certain times of the day. That implies that the users do not share a single, well-defined schedule, though users do seem to be more active during the normal workday (7 AM to 8 PM). Thus, we further examine the users' accesses individually.

In our dataset, we have accesses that represent users' clock-ins and clock-outs. Users with such accesses are more likely to have regular schedules in the system, since their work hours are explicitly logged.

We find that 29% of the users have clock-ins and/or clock-outs in their card tap logs. However, all but 4 of those users did not consistently perform a clock-in and/or clock-out when they entered and left the station. Further, when we examined those users' time-of-day accesses (seen in Figure 4.6), we found that while some of the users have more well-defined shift timings, their shifts are not consistent and change frequently.

For example, Figure 4.5a shows the the time-of-day distribution of accesses for one of the users with clock-ins and clock-outs. There are three big spikes in the user's overall

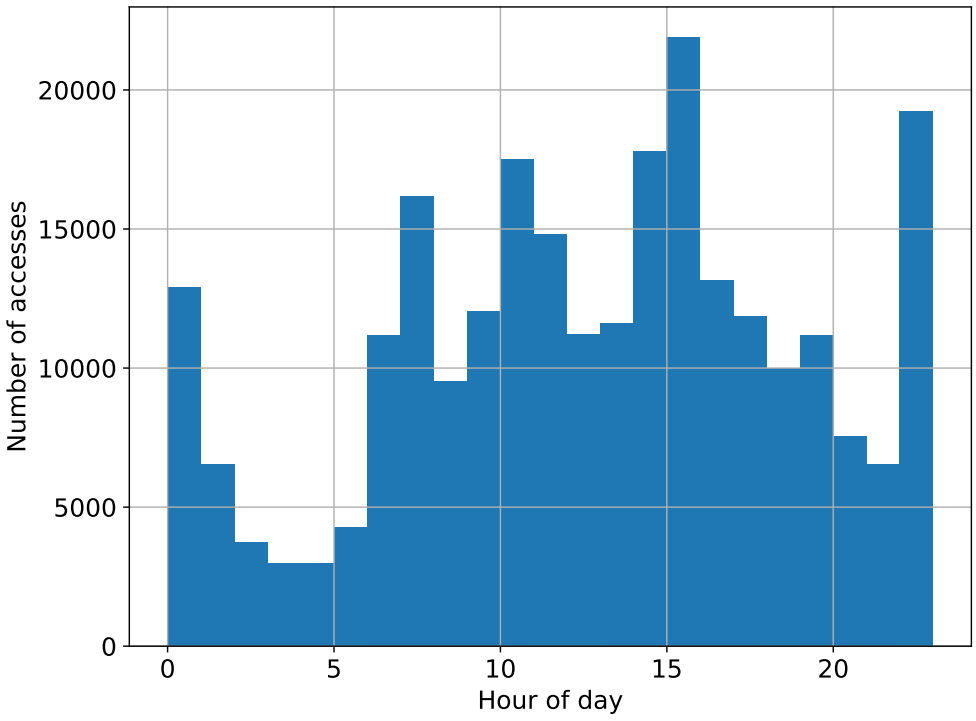
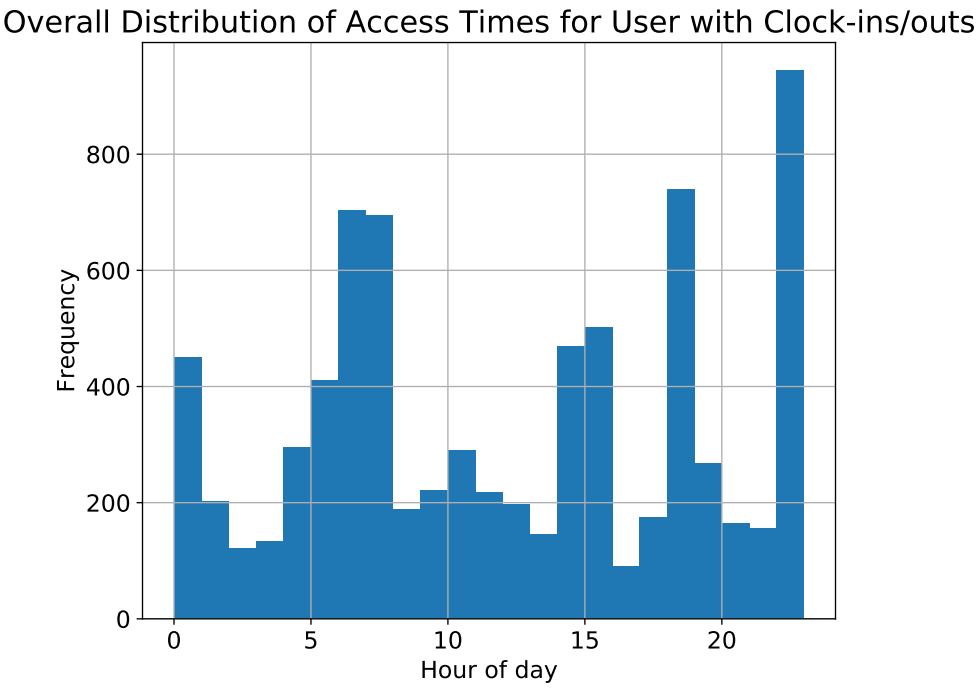
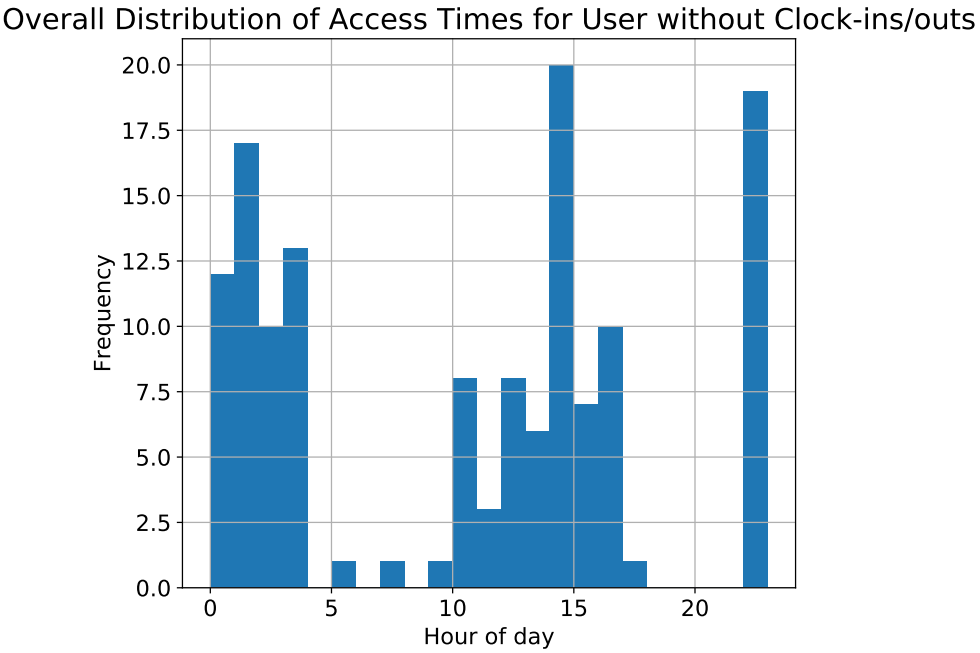


Figure 4.4: The distribution of all users' accesses over the time of day.

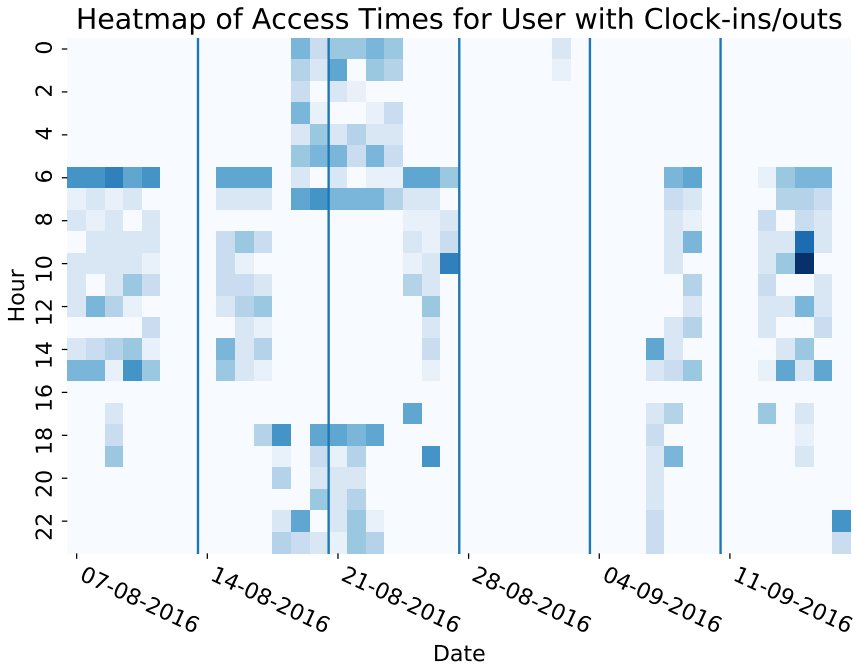


(a)

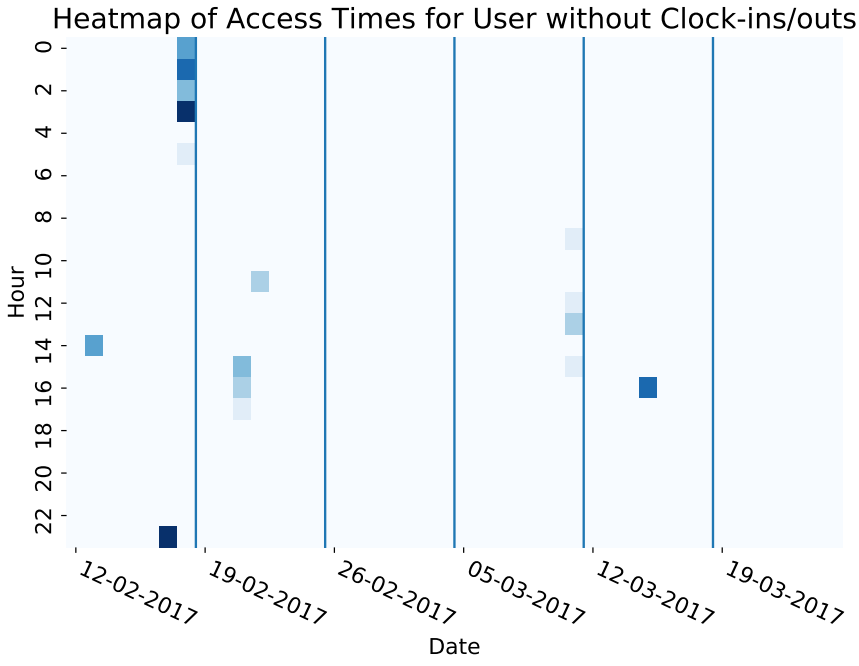


(b)

Figure 4.5: The overall distribution of the access times for (a) a user with clock-ins and clock-outs, and (b) a user with no clock-ins or clock-outs.



(a)



(b)

Figure 4.6: The distribution of the access times over a few days of (a) a user with clock-ins and clock-outs, and (b) a user with no clock-ins or clock-outs.

distribution, which implies that the user likely has three different shift schedules in the logs. However, we see in Figure 4.6a that the shifts do not have consistent lengths and that their timings can overlap (e.g., 6am—8pm and 6pm—8am). Furthermore, the changes in shifts do not follow a consistent periodic behavior. Users who did not have any clock-ins or clock-outs were similarly very sporadic in their behavior, as shown in Figures 4.5b and 4.6b. Without additional information about the actual assignment of shifts to users, it would be a nontrivial task to predict timing information for users, and that would limit the utility of using temporal behavior in detecting malicious accesses. Thus, while using time-of-day behavior might provide additional information, we decided not to use it in this work, opting instead to look only at spatial behavior. As we will show in the subsequent sections, our spatial-based detector performs very well on the users with clock-ins and/or clock-outs, so we believe that the utility of including temporal behavior in our model would be limited.

4.6 EVALUATION: DETECTION PERFORMANCE

In this section, we utilize the real-world data set to demonstrate the effectiveness of our framework in our railway transit station case study. We determine the detection capability of our proposed behavior models, and we examine the possibility of detecting malicious movement in an online manner.

4.6.1 Experiment Setup

We simulated malicious movement in order to conduct a more thorough assessment of the detection ability. For each user, we injected accesses into the testing data to simulate an adversary's deviations from normal movement.

We chose a few points within the sequence of movement accesses in the testing data representing the time at which the adversary decides to move off-course. Then, for each point, S_i, we randomly selected a target room $R_T \in \mathbf{R}$ that the adversary will visit, because almost all rooms house critical assets and can be potential targets. For the sake of simplicity, we assume that the adversary will take the shortest path to reach the room R_T from the current point S_i, because the adversary would want to tap through as few doors as possible to reduce the possibility of detection.

We calculated the shortest path from S_i to R_T as $S_i e_1 S_1 \dots S_n e_n R_T$. For each edge e_i, $i \in [1, n]$ that has a door code, we added an injected access $A_i = S_i \rightarrow S_{i+1}$ that represents the logging of the adversary's movement as he or she uses the acquired access control device to move through the spaces protected by card readers.

We can repeat that process to choose a series of rooms that the attacker visits (to consider the case when an attacker wanders around the space to do a site survey and explore potential attack opportunities, or when an attacker needs to access a few rooms in sequence to achieve his or her goal). We can also restrict the choice of rooms to only those rooms that have been visited by the user in the historic card tap logs, to simulate a stealthier attacker. We set the number of rooms visited by the attacker to 1; in the next section, we describe how we experimented with increasing the number of rooms visited by the attacker.

We set the rate at which we inject data (or choose points within the testing data) at 5%. In the next section, we discuss experiments in which we varied the percentage of malicious accesses by changing the rate at which we injected data.

We split the data set into 80% training and 20% testing subsets and performed 10-fold cross-validation. We conducted the experiments on a Windows 7 Home Premium machine with a 2.7 GHz CPU core and 4 GB of RAM.

4.6.2 Results

In this subsection, we present the evaluation results for our approach from Section 4.3 based on the card tap data set from the railway station. We also compare our approach to a baseline method that is a simple extension of building access controls. The baseline method keeps a list of spaces $Visited = \{s \in Seq_{Hist}(u_i)\}$ that has been visited by a user u_i, and any card tap by u_i that led to a space not contained in $Visited$ is marked as malicious. In other words, the baseline method is similar to the whitelisting in building access controls, but limits the user's movement further to only those spaces in the historic card tap logs that the user has visited before.

Implementation Performance

We evaluated the running time of both the offline and online phases. The average running times over all users of the construction of Markov models in the offline phase were 1.18 s and 12.4 s, respectively, for the $\mathcal{M}_r$ and $\mathcal{M}_s$ Markov models, whereas the average running times over all physical accesses of the OnlineDetection function in the online phase were 0.2 ms and 0.07 ms, respectively, for the $\mathcal{M}_r$ and $\mathcal{M}_s$ Markov models.

The running time of the offline phase for the $\mathcal{M}_s$ Markov model is longer than that of $\mathcal{M}_r$ because of the larger amount of information needed to construct the **Thresh** function. On the other hand, the running time of the online phase for the $\mathcal{M}_r$ Markov model is longer than that of $\mathcal{M}_s$ because of the calculation of shortest paths in the OnlineDetection algorithm.

The offline phase can be conducted sporadically during system downtime, whereas the online phase is fast enough to be executed in a real-time manner.

The average size over all the users of the constructed $\mathcal{M}_r$ Markov models is 3 vertices and 6 edges, with a maximum size of 28 vertices and 77 edges. On the other hand, the average size over all the users of the $\mathcal{M}_s$ Markov models is 16 vertices and 28 edges, with a maximum size of 162 vertices and 251 edges.

Detection Capability

We varied the percentile value used to select the threshold value in **Thresh** for the two different Markov models and plotted their *receiver operating characteristic* (ROC) curves. The ROC curve plots the true positive rate against the false positive rate obtained using different percentile values and is used to analyze the ability of a binary classifier. The closer the curve is to the top left-hand corner of the plot, the better a classifier (or in this case, our intrusion detection solution) is. The ROC curves are shown in Figure 4.7. The *areas under the curve* (AUCs) of the $\mathcal{M}_r$ ROC curve and the $\mathcal{M}_s$ ROC curve are 0.80 and 0.85, respectively. Those ROC curves show that both models achieve relatively high accuracy rates, although at the slight cost of an increased false positive rate. Based on the higher-peaked curve and the AUC value, the $\mathcal{M}_s$ model does a better job at classification than the $\mathcal{M}_r$ model. For the results detailed in the following paragraphs, we picked a 95th percentile value as the threshold for both the $\mathcal{M}_r$ and $\mathcal{M}_s$ models, for fair comparison.

Out of the 98,818 card taps, our $\mathcal{M}_r$ Markov model approach marked 9,187 as suspicious, whereas the $\mathcal{M}_s$ Markov model approach marked 12,929 as suspicious. Hence, either way, the practitioner's effort would have been reduced by over 85%. Table 4.2 shows the number of card taps over the ten testing subsets. We can see that large fractions of the injected accesses and malicious ground truth data are detected as malicious. The numbers of false positives are also low and fairly constant over the ten subsets.

On average, over the ten testing subsets, the $\mathcal{M}_r$ approach gives a false positive rate of 0.09 and a false negative rate of 0.28, while the $\mathcal{M}_s$ approach gives a false positive rate of 0.13 and a false negative rate of 0.20. In comparison, the baseline method had a false positive rate of 0.03 and a false negative rate of 0.35.

Since the $\mathcal{M}_s$ Markov model incorporates more information about the user's normal movement behavior, its false negative rate is lower, since we can more easily detect deviations from that normal behavior. However, that comes at the cost of a slightly higher false positive rate. On average, about 30% of the users had a lower false positive rate, with an average decrease of 0.35 and a maximum decrease of 1.0 relative to rates found with the $\mathcal{M}_r$ model.

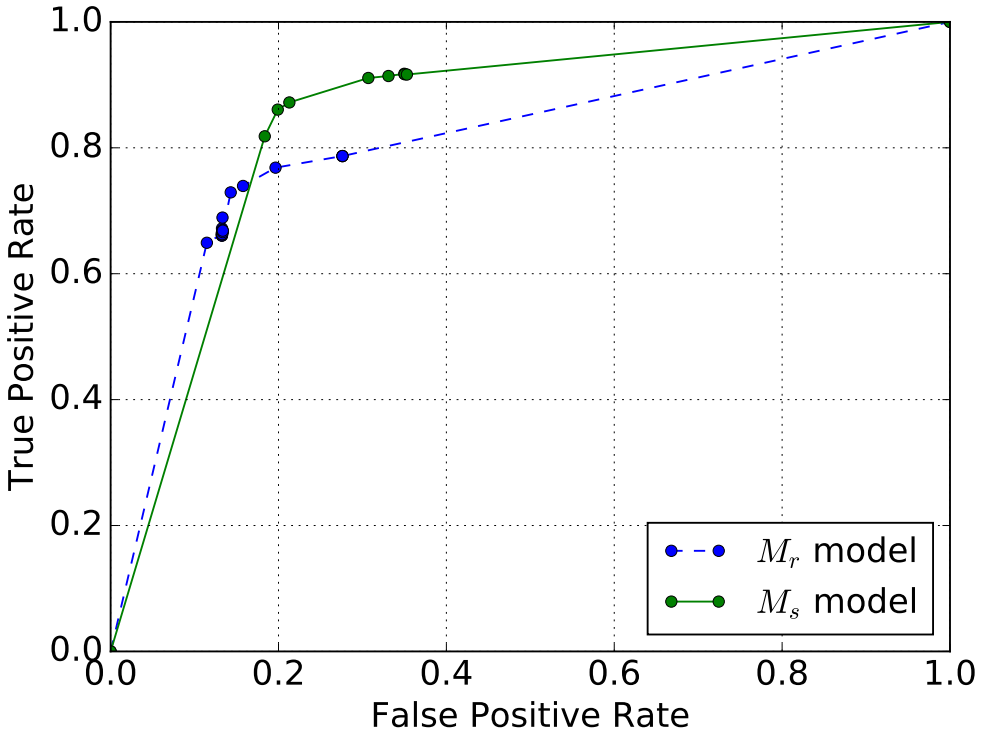


Figure 4.7: The ROC curves for the $\mathcal{M}_r$ and $\mathcal{M}_s$ Markov models.

Table 4.2: The number of physical accesses over time divided into three sections representing the valid (or non-malicious) accesses, the malicious ground truth accesses, and injected accesses.

	Invalid			**Injected**			**Valid**		
	Total	M_r	M_s	**Total**	M_r	M_s	**Total**	M_r	M_s
15/7	57	47	49	1067	756	874	9963	1145	1693
31/8	67	44	49	1044	762	844	9122	642	1066
15/10	63	36	38	1134	832	888	9542	881	1155
30/11	21	17	18	1036	749	816	9138	742	1188
15/1	27	24	25	1215	884	996	10428	985	1204
28/2	35	21	27	1237	923	1011	11174	1214	1659
15/4	35	28	30	1292	902	1004	11017	961	1355
31/5	39	29	29	1250	875	991	9687	781	1096
15/7	60	40	38	1159	830	948	9162	905	1205
31/8	45	34	37	1180	858	949	9136	931	1308

On the other hand, the baseline method has the lowest false positive rate, but at the cost of a higher false negative rate. In particular, we find that if an attacker carefully selects his or her path by moving only to previously visited rooms, the baseline method will not be able to identify any of those paths (i.e., its false negative rate will be 100%). In comparison, our

solution can still raise an alarm if the path covers any unusual transitions among previously visited rooms. When the number of rooms visited by the attacker is 1, our false negative rate for the $\mathcal{M}_r$ Markov model is relatively high (i.e., 0.28) because a substantial fraction of the generated malicious paths are indistinguishable from legitimate paths that a user actually traveled before. In other words, since most of the generated paths are short, it becomes impossible in a certain fraction of cases to differentiate anomalous and normal movement behavior.

To study how the length of the attacker's path affects the performance of our approach, we varied the number of visited rooms in the path; the results are presented in Figure 4.8. The baseline method is still unable to detect any of these malicious paths, regardless of their lengths. In contrast, the probabilities that our Markov model approaches will detect a path increase as the path grows longer. As seen in Figure 4.8, the $\mathcal{M}_s$ Markov model performs better than the $\mathcal{M}_r$ Markov model.

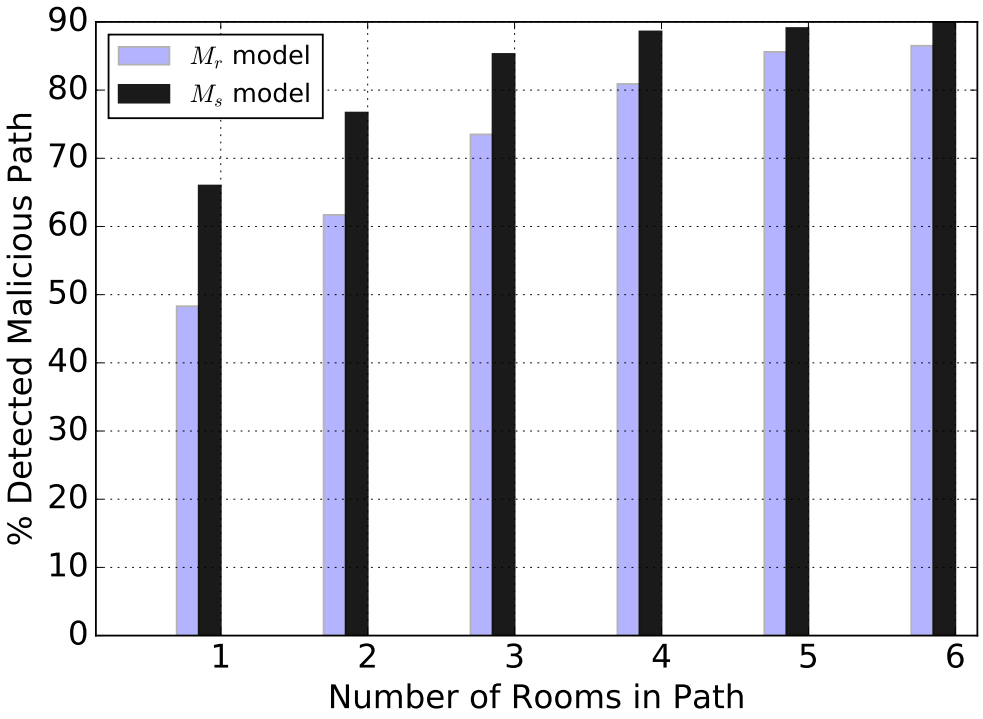


Figure 4.8: The percentage of detected malicious paths vs. number of rooms in the path for the $\mathcal{M}_r$ Markov model and the $\mathcal{M}_s$ Markov model.

We also varied the rate at which we injected malicious accesses into the testing data and plotted the resulting false positive and false negative rates obtained from using the $\mathcal{M}_s$ Markov model. The plot is shown in Figure 4.9. We can see that the false positive rate increases almost linearly with the percentage of malicious accesses that were injected into

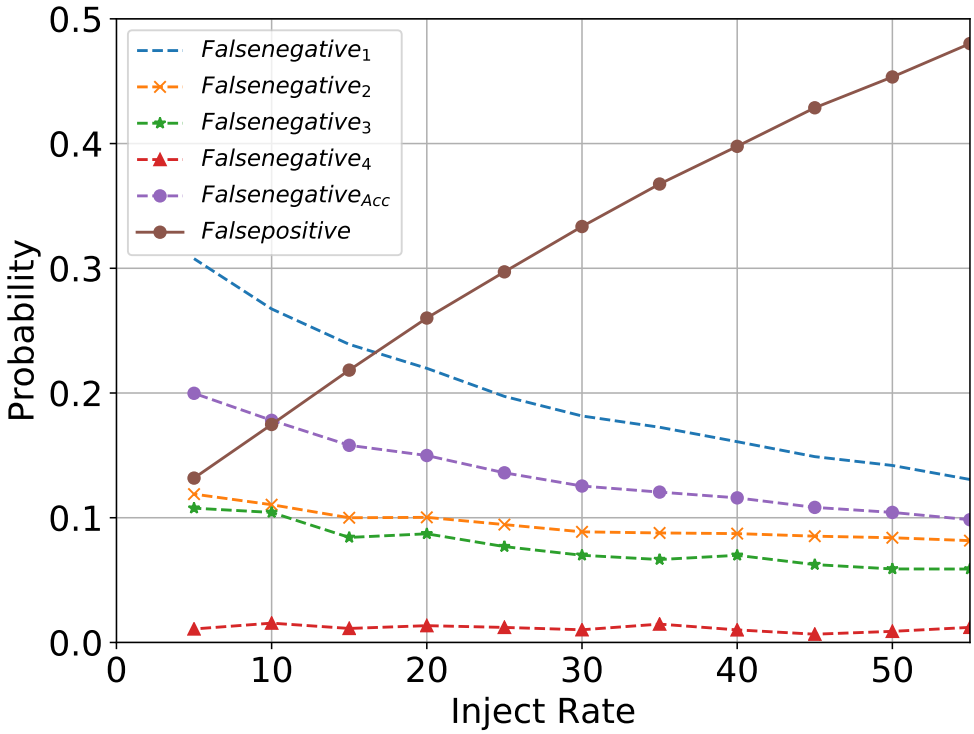


Figure 4.9: The false positive and false negative rates for the $\mathcal{M}_s$ model vs. the rate of injection of malicious accesses. The solid line represents the false positive rate, whereas the other dashed lines are the false negative rates for each malicious access on the path and the accumulative false negative rate.

the testing data.

For every point of injection into the data, A_iA_{i+1}, where A_i and A_{i+1} are accesses and the point of injection is between the two of them, the resulting sequence of physical movement accesses would be $A_iM_1 \dots M_rA_{i+1}$, where M_i, $i \in [1, r]$ are the injected malicious accesses ending in a room, $M_r = S_i \to R_m$.

If we consider the access $A_{i+1} = S_j \to S_{j+1}$, the previous visited room is now R_m, which is a random room in the station according to our simulation strategy. Therefore, the current state would be (R_m, S_{j+1}). The likelihood that this state belongs in the Markov model $\mathcal{M}_s$ and has a high transition probability from the previous state is low, so our detection algorithm will mark it as a malicious access. Then, for every point of injection into A_iA_{i+1}, the originally non-malicious access A_{i+1} would be marked as malicious instead of normal. The false positive rate would therefore be correlated with the rate of injection of data.

On the other hand, the false negative rate decreases slightly as additional malicious data are injected. In particular, the false negative rate for the first malicious access on the path, M_1, decreases while the false negative rates for the second, third, and fourth malicious

accesses (M_2, M_3, M_4, respectively) remain relatively steady. That shows that the more active an adversary is in moving around the station (equivalent to a higher injection rate of malicious accesses), the higher our chance of detecting the malicious movement early on.

Finally, we determine the amount of training data that are required for the algorithm to construct an accurate model of normal movement behavior. We see in Figure 4.10 that the larger the number of days that a user is active in the training data, the lower the false positive rates obtained. More precisely, the upper bound on the false positive rate is seen to decrease linearly with an exponential increase in the number of distinct (but not necessarily consecutive) days of activity. In particular, the vertical line in the figure shows that 37 active days of data are needed to achieve at worst a false positive rate of 0.4. Therefore, our detection system requires around a month's worth of active accesses for training, which, in practice, is a reasonable amount of data to obtain.

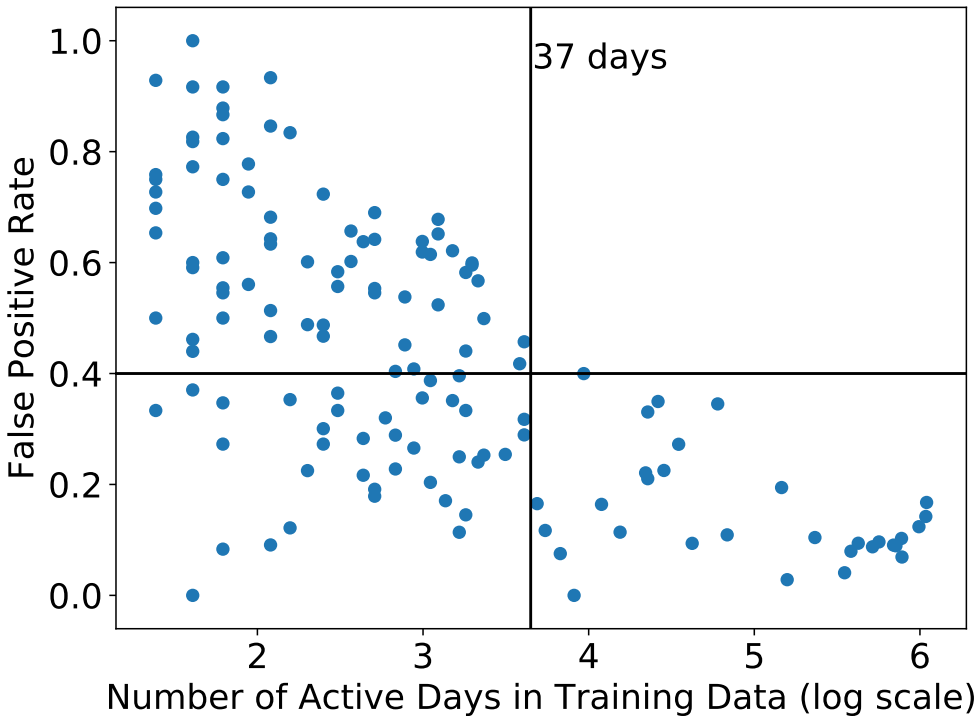


Figure 4.10: The false positive rates vs. the number of days that a user is active in the training data.

Comparison with Different Models

We want to determine how accurately different models can be used in our detection system. We used the F2 score, which measures a weighted average of the precision and recall of the

detection system's accuracy. Since it is more important to detect anomalies (i.e., suspicious movement behavior) than to reduce false positives, we chose the F2 score because it weighs the recall more than the precision.

In particular, we compared the first-order Markov model $\mathcal{M}_s$ with the first-order Markov model $\mathcal{M}_r$, a second-order Markov model $\mathcal{M}_s^2$, and several Bayesian network models.

Bayesian network models are probabilistic graphical models that encode the dependencies between states. They are a richer model than Markov chains because they allow for a larger set of dependencies among states and are not restricted to the single directed chain of dependencies of the Markov model.

We constructed Bayesian network models whose nodes represent the sequence of spaces visited by the user. For example, a Bayesian network with three nodes N_1, N_2, N_3 represents the first, second, and third spaces visited by the user, respectively. While a Markov chain would have N_3 dependent on N_2 and N_2 dependent on N_1, a Bayesian network could have any other combination of dependencies (as long as it resulted in an acyclic graph).

We used the Bayes Server tool [72] to learn a Bayesian network model. For a given number of nodes in the network, x, we fed training data for each user into the tool. The training data are a collection of moving windows of size x over the historic access data of a user. The tool learns the structure of the network and the parameters for the dependencies among the nodes. We varied the number of nodes in the Bayesian network to be between 2 and 5.

Then, in the online phase, given the current movement sequence $Seq = S_1 S_2 \dots S_n$, we calculated the log-likelihood of the sequence, $log(p(S_1 S_2 \dots S_n))$, and the conflict score of the sequence, $\text{Conf}(Seq) = log((p(S_1)p(S_2)...p(S_n))/P(Seq))$. The **Thresh** function takes in the training data and returns the 95th percentile of the log-likelihood, and the 100th percentile of the conflict scores of those data given the network model. The function definition is shown below.

$$\textbf{Thresh}(Access_{Past}) = percentile_{95}(log(P(Access_{Past}))), percentile_{100}(\text{Conf}(Access_{Past})) \tag{4.6}$$

We can compare the threshold values returned by the function $Y, Z = \textbf{Thresh}(Access_{Past})$ with the log-likelihood and conflict score of the current movement sequence. If the log-likelihood of the current movement sequence falls below Y or the conflict score falls above Z, then the access S_n is deemed anomalous.

We calculated the F2 score for each model used in the detection system. The results are shown below in Table 4.3. The higher the F2 score, the more accurate the model is in capturing the users' movement behavior.

We can see from the table that as the number of nodes in the Bayes network increases, the

F2 score decreases. That corroborates our previous findings that most users' movements have only first-order dependence. We also note that the 2-node Bayes network model performs the best, followed closely by our first-order Markov model M_s.

The 2-node Bayes network model has one dependence relation, since there are only 2 nodes in the model. Then, the log-likelihood of a 2-node sequence in the Bayes network model is equivalent to the probability of a 2-node sequence in M_s. However, our model M_s considers the cumulative probability of longer sequences. That implies that accumulating a probability of sequences longer than 2 spaces would not increase the accuracy of the model. In future work, we will investigate the types of users to which that observation applies and further improve our model M_s to accommodate that insight.

Table 4.3: The false positive rate, false negative rate, and F2 score for each of the models.

	False Positive Rate	**False Negative Rate**	**F2 score**
M_r	0.09	0.28	0.66
M_s	0.13	0.20	0.69
M_s^2	0.19	0.15	0.67
2-node Bayes	0.12	0.16	0.72
3-node Bayes	0.15	0.19	0.67
4-node Bayes	0.19	0.23	0.62
5-node Bayes	0.21	0.27	0.58

Online Detection

We have shown the running time of our algorithm to be on the order of milliseconds. Since the complexity of the algorithm is constant, it is also scalable. Therefore, our approach can work fast enough to process physical accesses in real-time. In addition, we want to study how early on a malicious path of a certain length can be detected, because we want to analyze how effective our approach is at raising an alert in real-time before an attacker reaches his or her destination room. We considered injected paths with a length of 4, and the false negative rates for the first, second, third, and fourth injected accesses in the path were 0.49, 0.3, 0.12, and 0.04, respectively, for the $\mathcal{M}_r$ Markov model and 0.32, 0.12, 0.09, and 0.02, respectively, for the $\mathcal{M}_s$ Markov model. That shows that our approach is able to detect malicious paths (with a certain minimum length) with high confidence, and even before an attacker has reached the destination room.

4.7 EVALUATION: MODEL SELECTION

In this section, we evaluate our decision to choose different models to represent user movement behavior based on the users' categorization. First, by evaluating our usage of the entropy metric, we answer the question of whether the movement behavior of users can be characterized effectively in a complex system. Then, we determine whether integration of information about device state can increase the accuracy of our detection approach for a particular class of user.

User Characterization

First, we evaluate whether the entropy metric defined in Section 4.3 is suitable for characterizing user behaviors. If the entropy metric can differentiate user behaviors, then the Markov models constructed for users with lower entropy ($q_1 \in \mathbb{T}$) will have a better detection capability (lower false positive and negative rates) than those constructed for users with higher entropy. If so, that would substantiate our choice of using Markov models for q_1 users; in the next subsection, we evaluate the augmented Markov models for q_2 users and consider how they improve on the base Markov model.

We plot the entropy vs. false positive rates for the $\mathcal{M}_r$ Markov model in Figure 4.11[4]. Each point in Figure 4.11 represents a single user in one testing data subset. One user should map to a maximum of 10 points in the plot.

We can see that users whose entropy is below 0.7 have a lower false positive rate, on average, than users whose entropy is above 0.7. We also note that users with clock-ins and/or clock-outs have a lower entropy value in general and a lower false positive rate. The implication is that there is some correlation between the regularities of spatial and temporal movement, which corroborates our claim that our approach works better with users who have more periodic behavior. We can therefore set our entropy threshold $\mathbf{E}$ to 0.7 in order to distinguish between the user types. Then, 29% of the users would belong to q_1 and 71% to q_2. Although fewer users are categorized under q_1, these users account for 63% of the card taps.

However, several outliers show a high false positive rate for q_1-type users. We studied each of them individually and found that there were reasons why the accesses were marked as suspicious. The users represented by triangles in Figure 4.11 accessed rooms that they had not previously visited, whereas the users represented by squares had a small testing set (fewer than 10 card taps), so their false positive rates are disproportionately high. In actual

[4]The results for the $\mathcal{M}_s$ Markov model are similar, and we therefore do not show them here.

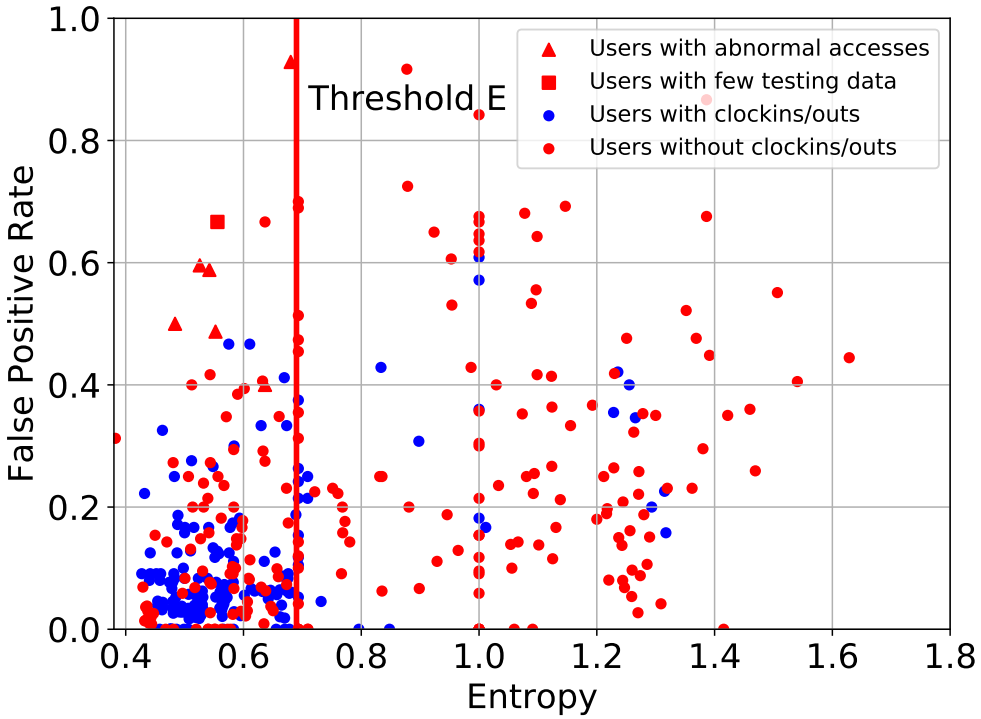


Figure 4.11: The distributions of false positive rates according to the entropy of users' movements. Blue points represent users with clock-ins and/or clock-outs, snd red points represent users without clock-ins or clock-outs.

operation, the training data set and set of real-time card taps will be much larger, so there won't be outliers.

Integration of Device State

In Section 4.3, we proposed to correlate logs documenting device state with physical access logs in order to decrease the false positives for the q_2-type users. Since the device-state logs in our case study are collected separately from the physical access logs, we have to assume that the timing information in the device-state logs and physical access logs is synchronized.

The logs collected for different types of device (e.g., breakers or lights) are different, and thus the amount of information about the device's state that can be extracted varies. We extracted the textual description and alarm values that indicated device failure and searched the device logs for failure incidents.

In particular, there are four rooms in the station for which we could identify the failure of a device from the logs with particularly strong confidence. These rooms contained devices that controlled the environment in the station (e.g., air chiller and water pumps).

With the additional step of correlating device failures with physical accesses, we reduced the false positives for a subset of the users by an average of 0.45 for the four rooms. This preliminary result shows that we can use additional logs regarding the system environment to determine whether an access was malicious.

4.8 RELATED WORK

In this section, we discuss the related work spanning domains from physical movement tracking and prediction to anomaly detection of physical movement and cyber events.

There has been a substantial amount of work on use of cyber logs (e.g., network flows and system logs) to profile users and detect events of interest. For example, Kent et al. proposed *authentication graphs* [73] to profile user behavior and detect threats using computer authentication logs in an enterprise network. Specifically, the authors analyze a 4-month log of authentication events in an enterprise network that ran a Microsoft Windows Active Directory authentication system. They observe distinguishable graph attributes for different user roles and propose to use such differences to help improve user behavior profiling and misuse detection. In contrast, our work focuses on physical access logs, for which physical-world factors, like space and time, directly impact the correlation among different access events. Our data set also tends to be more noisy and less complete than a log collected from an authentication system. Despite those differences, we too observe the importance of distinguishing different user roles.

In the area of physical access control, there has been work in the route anomaly detection area that looked at people or objects moving in a geographical space that was not delineated by rooms [74–76]. Pallotta and Jousselme [74] and Radon et al. [75] both detect deviations in the trajectory of a vessel in the maritime domain. Their approaches use contextual information, such as the speed of the vessel and weather information, in order to predict the next location of a vessel. However, in the maritime domain, the source and destination of the vessel are already known beforehand, and the anomalies are assumed to arise from the differences in trajectories. That is unlike our work, in which we focus on an indoor setting that has unpredictable destinations for each user.

Dash et al. [76] use mobile data to predict the movement of people in a geographical region. They construct multiple Dynamic Bayesian network models, each of which includes different granularities of context (e.g., day of the week vs. time of day). They predict the next visited location by analyzing the results obtained from each of those models. Unlike their completely data-driven approach of applying all models before computing the best result, we propose a more guided approach in which one first chooses the appropriate model

based on an understanding of a person's past movement data.

Lin et al. [77] also use mobile data for user verification by constructing *Hidden Markov Models* (HMMs) that represent each user's mobility patterns. Those HMMs are constructed from cell tower data and ambience features of WiFi. Using the sequential probability ratio test, the authors decide whether the current sequence of locations adheres well to the user's HMM model and not to other users' HMM models. Although their approach allows for real-time authentication of a user, the result of the analysis at each time point may be inconclusive if the evidence of the user's movement does not conform more strongly to the user's HMM model than to other users' HMM models. Our approach, however, produces a decisive yes-or-no answer for each user movement and does not compare the movement to other users' models. In other words, we take a more conservative approach and treat any deviations from the user's movement model as suspicious behavior. This allows the security practitioner to more quickly identify malicious behavior when it appears.

In contrast to the works described above, we consider the more restrictive setting of indoor location tracking, which reduces the amount of noise in the data and allows us to identify a user's location with more confidence. Because of physical barriers that prevent a user from moving uninhibited from one space to another, the paths that a user can take are also limited.

For indoor physical access, there has been work in both movement prediction and anomaly detection. In the movement prediction domain, Gellert and Vintan [78] use HMMs to predict a user's next location. They use real-world physical access data of four users from a single floor of an office building; notably, the size and topology of the building are very small. Their data set contains only four users. Their results show that a simple Markov model of order 1 gives the best performance. Koehler et al. [79] expand on Gellert's work by using ensemble classifiers to predict how long a user will stay at a given location.

In the anomaly detection domain, different techniques to detect differences in a user's movement have been proposed. Graph models have been studied by Eberle and Holder [80] and Davis et al. [81]. Eberle and Holder [80] detect structural anomalies by extracting common subgraph movement patterns [82]. However, they only consider simplified physical layouts and do not distinguish among different user roles. Davis et al. [81] search labeled graphs for both structural and numeric anomalies and apply their approach to physical access logs in an office building.

Other models, ranging from finite state machines to specific rules, have also been studied. Liu et al. [83] model the normal movements of devices as transitions in finite state machines. Unlike us, they focus on the movement of devices (instead of people) in a hospital setting, where their main goal is to detect missing-device events. Biuk-Aghai et al. [84] focus on

suspicious behavioral patterns, including temporal, repetitive, displacement, and out-of-sequence patterns. Those patterns involve only the time interval between movements and the reachability of locations rather than the sequence of locations that were visited.

Finally, patents from IBM [85] and Honeywell [86] present the general designs of ways to use physical access data to detect potential security incidents. However, they do not discuss detailed designs for dealing with complicated building topology and user roles, and do not provide experimental studies involving real-world traces.

4.9 CONCLUSION

One way in which organizations address insider threats is through physical security. However, the state of the art in building access control is lacking. In this chapter, we studied the use of physical access logs for detecting malicious movement within a building. We proposed a systematic framework that uses knowledge of the system and its users in order to analyze physical access logs, and we applied the framework to a real-world data trace of physical accesses in railway stations.

First, we characterized users by using a set of metrics that take historical physical access data as input. In particular, we defined a metric based on the approximate entropy of a time series that measures the regularity of user movement. Each user type is mapped to a behavior model, and the details of the model are learned through use of the user's past physical accesses. More specifically, we defined two different Markov models, which represent different amounts of information about the user's movement sequences. Finally, we developed an online detection algorithm that takes the behavior model and the building topology as input, and returns a score indicating the likelihood that the user's access is anomalous.

We applied our framework to a real-world data trace of physical accesses in railway stations. The results show that our proposed Markov models perform well in representing user movement behavior. In particular, inclusion of information about the path taken between rooms in the model helps the detection algorithm flag more malicious movement at the cost of a slightly increased false positive rate. The results also show that our framework can be applied in an online fashion to monitor and flag suspicious movement before an insider reaches the final destination room.

However, physical access logs are only able capture certain scenarios of human movement behavior. In real life, people can use physical or social means to circumvent the building access control system and remain undetected by sensors in the building. In the following chapter, we will address that limitation and propose ways to overcome it.

CHAPTER 5: SOCIO-TECHNICAL SYSTEM DEFENSE: DETECTING TAILGATING

In critical infrastructure facilities such as power substations, airports, and railway stations, physical security is of the utmost importance, in part because the equipment located within those buildings must be safeguarded from unauthorized insider threats [65, 87]. Physical security is typically enforced using building access control systems that limit human movement in indoor settings. The data collected from such systems are used for a variety of analyses, including detection of unauthorized or abnormal physical movement, as we showed in Chapter 4. Those analyses rely on the assumption that the cyber data on physical movement reflect actual movement scenarios.

However, that assumption is not true in real life, since people can circumvent a building access control system, so that their movements will not be captured by sensors. Then, our detection approach in Chapter 4 would not be able to identify those movements. In this chapter, we address that limitation of building access control systems by developing approaches that use physical constraints of human movement in indoor settings to identify instances of movement that escape detection by existing building access control systems.

5.1 BACKGROUND AND MOTIVATION

Building access control systems use door movement sensors and card readers to capture the movement of people. Those devices can be circumvented by physical and social means, e.g., by "tailgating", which means following an authorized person. Tailgating is a serious issue because it could allow unauthorized users into critical spaces, and that can constitute a security or safety violation. Thus, it is important (1) to determine the extent to which building access control systems can capture tailgating, and (2) to leverage additional data sources to identify the scenarios that cannot be captured by those systems.

In this chapter, we define physical constraints on human movement that we then use to infer tailgating from data collected by building access control systems. We study the limitations of those cyber data and propose solutions that use physical data sources to complement those data to identify potential physical security violations.

We base our study on the same building access control system discussed in Chapter 4, which monitors the movement of staff members and visitors in a railway station. The card tap logs collected by that system contain information about entries (card taps) and exits (door movement) to a space. We also collected physical and cyber data from other sources in that railway station, such as sign-in log book entries, to complement the card tap logs.

Specifically, the contributions in this chapter are as follows:

- We study how tailgating manifests in real life based on observations of tailgating in a railway station over a 6-day period. We use our observations and domain knowledge to infer movement behavior from card tap logs.
- We specify two physical constraints on human movement that are based on building topology and space occupancy, which we then use to drive our approach.
- We use the topological constraints on human movement to implement a checker that identifies tailgating when it finds a discontinuity in a person's movement trajectory. We thus found 3,999 instances of tailgating in a railway station during a 17-month period.
- We use the constraints on space occupancy to develop several approaches based on comparison of the numbers of card taps and egress events in the card tap logs, and show that there remain scenarios that are not visible in those logs; e.g., a visitor who tailgates in and out of a space remains invisible.
- We show that the limitations of card tap logs can be mitigated in part by using physical data sources such as manual sign-in logs to supply information regarding expected room occupancy. For example, we found that in 36 of the 80 visits to a server room, visitors noted their presence in the sign-in logs, but were invisible in the card tap logs.

5.2 RELATED WORK

Building access control systems typically use card readers and door movement sensors to provide dashboard information about users' accesses and door movement [87, 88]. That information can be used to track the locations of users, prevent unauthorized access, and detect violations of access control policy [70, 78, 79, 89]. However, building access control systems do not capture all potential user movement, e.g., tailgating.

Tailgating is a vulnerability that can result in theft, damage of property, and other unauthorized activity that is harmful to the system, and it has been found to occur at a rate of 40 to 60% in some office buildings, according to one report [90]. Measures such as employee education and installation of turnstiles [91] have been taken to prevent tailgating, as recommended by various guidelines, including the NERC CIP-014 [92]. However, physical mechanisms like turnstiles are not enforceable throughout a secure building, and it is hard to ensure that all employees will follow rules, even when they have been provided with training.

There has been work on detecting tailgating by using different technologies, such as video surveillance and additional sensors and badges. Advances in computer vision have allowed video surveillance to be further automated to detect and classify motion trajectories [93–95]. In particular, such algorithms can be used in tandem with radio-frequency identification (RFID) based technologies to detect tailgating [96]. Indoor location systems track users by using networks of beacons on access doors, and RFID tags [97] or phone apps [98–100] that are located on each user.

The limitation of those technologies is that they require additional installations of physical equipment (like antennas) and distribution of tracking devices, and they also introduce privacy and security concerns [101]. Unlike those approaches, we use existing data from building access control systems. We view video camera surveillance as a complementary solution and alternative data source to be used as part of our approach to identifying instances of tailgating.

5.3 CASE STUDY: RAILWAY STATION

We use the same railway station used in Chapter 4 for our case study. That railway station contains 62 rooms that house the equipment necessary to maintain the running of the station and its portion of the railway track. Below, we describe the data collected by that building access control system, and other supplementary information regarding the presence of people within the station.

5.3.1 Data Sources

We collected the following cyber and physical data about human movement and system events: (1) card tap logs, (2) station sign-in logs, (3) equipment room sign-in logs, (4) maintenance schedules, (5) manual observations of human movement, and (6) device event logs. The time periods of the collected data for each data source overlap and are shown in Figure 5.1.

The card tap logs, station sign-in logs, and device event logs contain data about the entire station. Both the equipment room sign-in logs and maintenance schedule contain data about events in the server room, which is labeled in Figure 5.2. Finally, the manual observations contain data about human movement to the doors highlighted in grey in Figure 5.2.

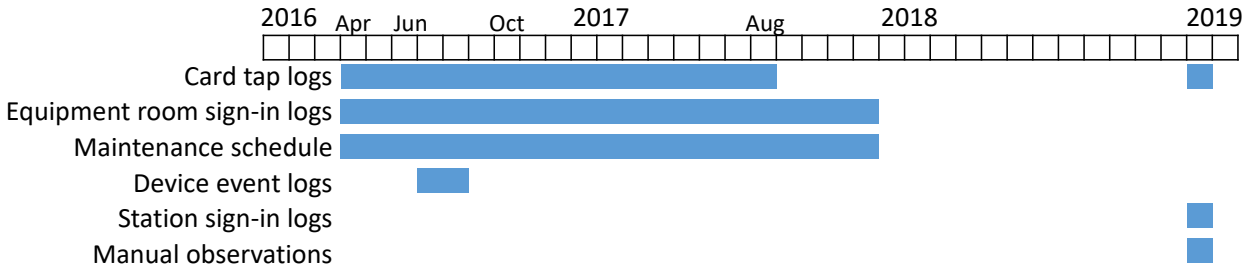


Figure 5.1: The shared timeline of the collected data from the railway station.

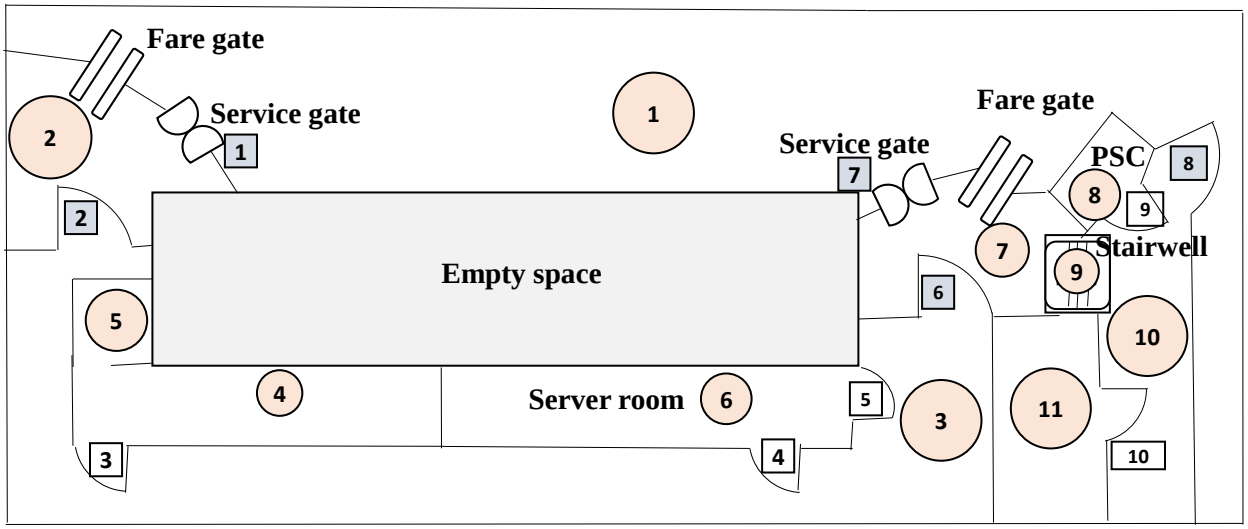


Figure 5.2: A small sample floor plan of Level G of the railway station. The doors that we observed are marked with grey squares.

Card Tap Log

We have card tap logs collected by the physical access system. They contain information about card taps and door movement events in the railway station. The events took place between April 2016 and August 2017, and during the month of January 2019. A total of 298,799 card taps were made by 781 users.

The logs contain the following information regarding physical movement: (1) timestamp; (2) doorcode; (3) card number; (4) user identification; (5) type of event (**Legal Access** (legal entry), **Invalid Attempt** (failed entry), **Free Egress**, **Clock In**, or **Clock Out**); and (6) condition of door (**Door Open Fail**, **Door Close Fail**). A failed entry implies either that the user's card has expired or that the user does not have permission to access the room. Clock-ins and clock-outs represent card taps at a time punch clock inside a room in the station. The condition of the door reflects special situations in which the door has either been left open for more than 10 seconds (**Door Close Fail**), or has not been opened

after someone tapped his or her card (**Door Open Fail**).

Station Sign-in Log

When visitors or staff members enter the railway station, they are first required to sign in at the *Passenger Service Center* (PSC). They record their pertinent details manually in a log book that is then signed by the station operator inside the PSC. At the end of their visit, they need to sign out at the PSC by recording their sign-out time which is then signed by the station operator. We have collected 97 entries in the log book from January 7 to 13, 2019.

That log book contains the following information: (1) name, (2) user identification, (3) the department to which the user belongs, (4) purpose of visit, (5) date of visit, (6) time of sign-in, (7) time of sign-out, (8) staff member who witnessed sign-in, and (9) staff member who witnessed sign-out. Visitors are always escorted by a railway staff member and are assigned a visitor pass with no credentials.

Equipment Room Sign-in Log

Each equipment room in the station has a log book. Every person who enters an equipment room needs to record his or her visit in that log book. At the end of their visits, people must record their sign-out times. We collected 143 entries to a server room in the railway station that took place between April 2016 and December 2017.

The log book contains the following information: (1) name, (2) the department to which the user belongs, (3) purpose of visit, (4) date of visit, (5) time of sign-in, (6) time of sign-out, and (7) name of accompanying staff member. Only staff members who are custodians of a room have card access to that room.

Maintenance Schedule

The equipment in the rooms is periodically maintained by teams of staff members. We have collected data about the maintenance schedule for the server room that contains the following information: (1) date of maintenance, and (2) devices to maintain. The schedule covers dates between April 2016 and December 2017.

Manual Observations

To complement the data that we collected, we conducted a small experiment to elucidate the movement of people in real life as they perform their tasks in the railway station. Given the sensitive nature of the data collection, we observed the movement of people only from public areas such as the concourse and platform. From our vantage point, we gathered a total of 86 movement events through 5 doors from January 7 to 13, 2019 for a total of 8 hours. Those 5 doors serve either as an (1) entrance to the station, or (2) entrance to corridors that lead to equipment rooms, including the server room, as shown in Figure 5.2. We recorded the following information: (1) date and time of movement, (2) doorcode, (3) description of people moving, (4) type of movement (entry or exit), and (5) identity of person who taps his or her card.

Device Events Log

The equipment in the rooms is constantly monitored, and any change in its state is logged in a database. We have collected the logs for the equipment in all the rooms in the station for the period of July and August of 2017. The logs contain the following information on events pertinent to the devices: (1) timestamp, (2) device identification, (3) event description, (4) event type, (5) value associated with event, (6) severity level of event, and (7) operator handling the event.

5.3.2 Data Preprocessing

Coalescing. In the card tap log, we found occurrences of multiple **Legal Access** (by the same card) or **Free Egress** events at the same door within the span of a few seconds. Consecutive **Legal Access** events within such a short period are due to a person's tapping of a card more than once to open the door. Similarly, closely spaced **Free Egress** events occur because of multiple movements of the door as someone pushes it open to exit. Since those events are repetitive instances of a single action, we coalesce each of those groups of events into a single entry. After analyzing the distribution of inter-event times, we set the threshold for coalescing at 20 seconds.

Identifying users across logs. We want to associate a user's presence in the server room with his or her card tap activity. Therefore, we cross-referenced the names of the custodial staff members and accompanying staff members in the equipment room sign-in log with the

user identities that appear in the card tap logs from the time period indicated by the sign-in log.

Grouping of visits. Multiple entries in the equipment room sign-in log sometimes represent a single visit by a group of people. Since we want to analyze the movement activity during each visit separately, we grouped the 143 log entries into incidents representing single visits, based on the time period and purpose of the visit. There were a total of 80 incidents, of which 27 were visits by a single person, and 53 were visits by a group of people.

Relating observations to logs. We wanted to determine how the movement we observed manifested in card tap logs. Specifically, we wanted to test our hypothesis that if a door is held open for long, i.e., a **Door Close Fail** event, that implies that tailgating has occurred. So we cross-referenced our observations with the card tap logs by using the recorded timestamps. We found that of the 12 times a **Door Close Fail** event occurred in the card tap logs during the period of our observations, only 3 of those instances corresponded to tailgating. Thus, that disproves our hypothesis and shows that we need a more involved approach to identify tailgating.

5.4 PHYSICAL CONSTRAINTS ON HUMAN MOVEMENT

Based on our observations and domain knowledge, we found that the events in card tap logs are not indicative of the number of people moving or the direction in which they moved. A **Legal Access** event implies only that the person who tapped his or her card was outside the door and was likely to enter that space. A **Free Egress** event implies that at least one person was inside the space prior to the egress event. A **Clock In** or **Clock Out** event implies that the person who tapped his or her card was in the room containing the clock puncher. So we can only infer the possible locations of people from those events, and thus cannot identify tailgating directly from the card tap logs. Instead, we define two physical constraints on human movement that allow us to use the card tap logs to identify tailgating:

1. *A person who wants to tap at a card reader B in a space that is protected by another card reader A must first tap at A before B.* A violation of this constraint means that the person tailgated into the space.

2. *Before a **Legal Access** or **Invalid Attempt** event at card reader (or door) A, the occupancy of the space outside the door must be nonzero. Before a **Free Egress** event at door A, the occupancy of the space behind the door must be nonzero.* Thus,

by tracking the occupancy of a space based on card tap log events, we can identify tailgating when there is a violation of the constraint.

5.5 USING BUILDING TOPOLOGY TO INFER TAILGATING

In this section, we use Constraint 1 defined in Section 5.4, that is, a person can only move through a series of connected spaces, to infer tailgating from card tap logs. We can build a person's movement sequence by tracking the **Legal Access**, **Invalid Attempt**, **Clock In**, and **Clock Out** events, as discussed in Section 5.4. Then, we use the building topology to determine the reachability of the spaces containing the card readers. If there is a gap in the sequence that required the person to access an additional card reader to reach the space, it implies that tailgating happened there.

We employ the same representation of building topology used in Chapter 4. More concretely, we represent the building topology as a directed multigraph $G = (S, E)$ in which the set of vertices S represents the spaces in the building. A directed edge $e_i = (v_1, v_2)$ represents possible movement from v_1 to v_2. The edges are labeled with doorcodes, $label(e)$, if there is a card reader bordering the two spaces. We use the example of the floor plan in Figure 5.2, which is represented as a graph in Figure 4.2.

We use the graph G to determine all possible pairs of doorcodes that can occur in a movement sequence without violating Constraint 1. First, we find all possible simple paths that a person may take between any pair of spaces.

$$Paths = \{(s_i e_i s_{i+1} \dots s_{i+n} e_{i+n} s_j) | \forall s_i, s_j \in S, e_i = (s_i, s_{i+1}) \in E\} \tag{5.1}$$

The edges along those paths are $Paths_e = \{(e_i \dots e_{i+n}) | (s_i e_i s_{i+1} \dots s_{i+n} e_{i+n} s_j) \in Paths\}$. Some of those edges may not have a doorcode; for example, moving out of a room does not involve tapping a card. We therefore remove those edges to get $D = \{(e_i \dots e_{i+m}) \subseteq p \in Paths_e | \forall e, \exists label(e)\}$. Then, the possible pairs of doorcodes in a movement sequence are $Dc = \{(label(e_j), label(e_k)) | (e_j, e_k) \subseteq D\}$.

If there is a pair of doorcodes in the movement sequence that does not exist in Dc, $(label(e_i), label(e_j)) \notin Dc$, then the person has tailgated between those spaces. The set of possible doors that the person skipped is $\{(label(e_{i+1}) \dots label(e_{i+n})) | (e_i, e_{i+1} \dots e_{i+n}, e_j) \in D\}$.

We found a total of 3,999 instances of a person not tapping his or her card at a door. 201 out of 781 people had at least one instance of such a violation. An example of a violation was a person who tapped at e_7 and then at e_5, skipping e_6.

To verify that those instances were indeed indications that someone skipped a card tap and not artifacts of a disused card reader, we checked that all the possible skipped doorcodes had at least one occurrence of a **Legal Access** recorded in the card tap logs. We found that there always was, and thus that the 3,999 instances were occurrences of tailgating in the station.

We found that a majority of the missing card taps were to staircases or doors that lead into the station. However, a sizable amount of missing card taps were to corridors that lead to critical equipment rooms (indicated in red in Figure 5.3), although some of those corridors are potentially more tightly controlled, and thus have fewer missing card taps. Thus, our approach can be used to discover which areas in a building have higher occurrences of tailgating.

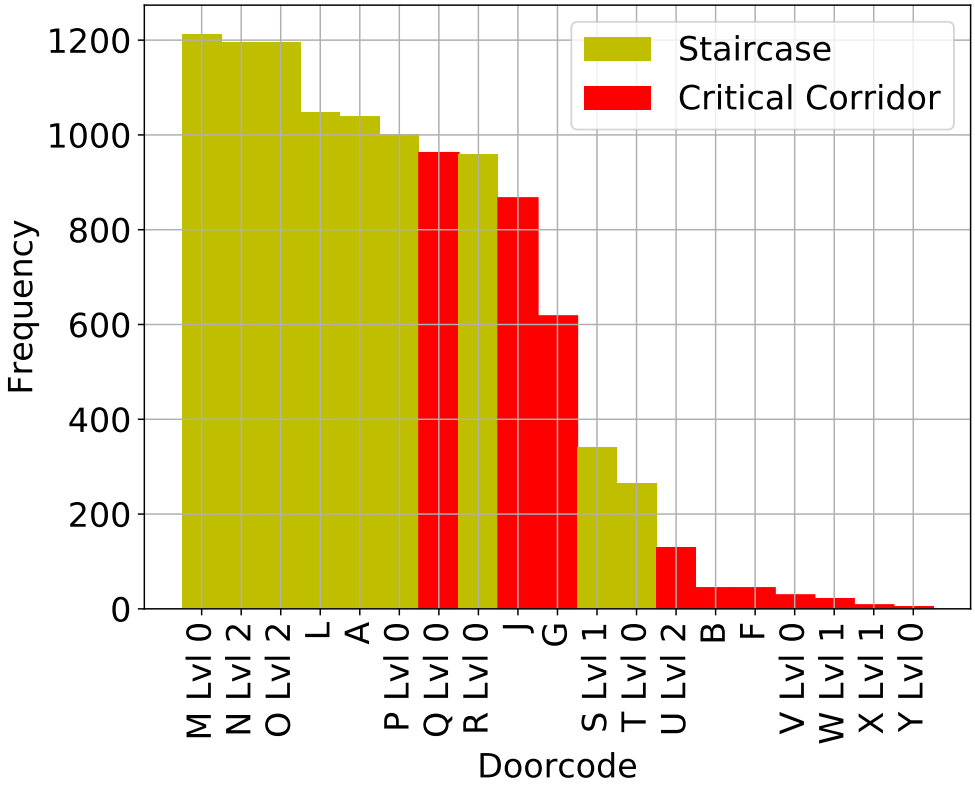


Figure 5.3: Frequency of occurrences of violations at different locations.

However, that approach only considers card taps. It does not consider egress events or cases where a person tailgates into several spaces and finally a room (e.g., e_7, e_6, e_5). We will tackle this issue in Section 5.6, where we consider both card tap and egress events.

5.6 TRACKING ROOM OCCUPANCY TO INFER TAILGATING

In this section, we use Constraint 2 defined in Section 5.4, which pertains to tracking the occupancy of a space, to infer tailgating. However, card tap logs do not provide information about the number of people moving into or out of a space, which is important for understanding the effect of movement on occupancy change. We propose to classify all possible movement scenarios and use our manual observations to identify the most common scenarios for the purpose of assigning movement counts to the events in the card tap logs.

5.6.1 Classification of Physical Movement Behavior

The events in the card tap log that pertain to the movement of people are **Legal Access** and **Free Egress**. Since those events are not indicative of the actual movement of people, it is possible for an arbitrary number of people to enter and exit the space.

We define the classes of movement scenarios as the different combinations of counts of people moving **In** and **Out** of the room for each **Legal Access** and **Free Egress** event, i.e., each class is labeled as **Access-In*-Out*** or **Egress-In*-Out***, where * represents a quantifier (**0**, **1**, or **N**) for the number of people moving in the given direction.

For the **Access** class, the person associated with the tapping event may or may not move **In** to the space. Independent of that person, an arbitrary number of people can move **In** or **Out** of the space. Thus, the set of possible quantifiers for the **Access** class is **Access-In{0,1}{0,N}-Out{0,N}**.

For the **Egress** class, the person who pushed the door open may or may not move **Out** of the space. Independent of that person, an arbitrary number of people can move **In** or **Out**. However, for **Egress**, unlike the **Access** class, there is no information on who pushed the door open, so we only differentiate between **0**, **1**, and **N** people moving **Out**; therefore, the set of possible quantifiers for the **Egress** class is **Egress-In{0,N}-Out{0,1,N}**. All movement scenario classes are shown in Table 5.1.

We used our manual observations to relate each movement scenario class to real-life situations, which we describe in Table 5.1. We annotated each class with the number of occurrences we found in our observations.

We found occurrences of the **Access-In00-OutN** class that happened at the service gates because those were the only doors that required someone to tap his or her card (**Legal Access**) to exit the space. Therefore, events that occur at the service gates cannot differentiate between a person entering and exiting the station.

We also found occurrences of the **Access-In00-Out0** class that corresponded to a person's

Table 5.1: Classes of movement scenarios annotated with the likelihood of the class's occurrence (**LH**), the number of occurrences of that class we found in our 86 observation events (**Found**), and a description of how that class manifests in real-life situations (**Situation**).

LH	Found	Class	Situation
2	0	Access-In10-OutN	People exit coincidentally as someone taps card
5	28	Access-In10-Out0	Someone taps card and enters
2	***0***	***Access-In1N-OutN***	***Someone taps card and lets people in because of social courtesy or to let visitors in; people exit coincidentally***
5	***10***	***Access-In1N-Out0***	***Someone taps card and lets people in because of social courtesy or to let visitors in***
1	***0***	***Access-In0N-OutN***	***Someone taps card for other people to enter but doesn't enter; people exit coincidentally***
2	***1***	***Access-In0N-Out0***	***Someone taps card for other people to enter but doesn't enter***
2	13	Access-In00-OutN	Someone taps card and no one enters; people exit coincidentally
1	2	Access-In00-Out0	Someone taps card and no one enters or exits
4	***0***	***Egress-InN-Out1***	***Someone pushes door open to exit; people enter coincidentally***
3	***0***	***Egress-InN-OutN***	***Someone pushes door open for group of people to exit; people enter coincidentally***
3	***3***	***Egress-InN-Out0***	***Someone pushes door open for people to enter***
5	23	Egress-In0-Out1	Someone pushes door open and exits
5	6	Egress-In0-OutN	Someone pushes door open for group of people to exit
1	0	Egress-In0-Out0	Someone pushes door open and no one enters or exits

tapping of his or her card at the service gate again to ensure that the gate was properly locked.

Finally, the one occurrence of the **Access-In0N-Out0** class happened when a staff member from the inside tapped at the service gate to allow another staff member to enter the station.

By the definition of tailgating, the movement scenarios that involve the movement of a group of people into a space are the only classes that constitute tailgating; we italicize them in Table 5.1. We are particularly concerned with identifying instances of those classes.

We can see from Table 5.1 that of the movement classes that correspond to tailgating, the **Access-In1N-Out0** and **Egress-InN-Out0** classes happen the most frequently. Thus, we will focus our efforts on identifying instances of those classes. We also find that the most common classes of movement are **Access-In10-Out0** and **Egress-In0-Out1**. So we can assume that a **Legal Access** and **Free Egress** event corresponds to the movement of a single person.

5.6.2 Using Card Tap Logs to Identify Tailgating

Based on our domain knowledge, we leverage the constraint that the occupancy of a room must be 0 at the end of a working day. In tracking the occupancy of a space, we use the assumption from Section 5.6.1 that one event in the card tap logs corresponds to the movement of one person. We base our approach on the physical constraints that there should be no one in an equipment room before and after a visit (presuming that visits don't overlap). Then, if no tailgating occurs, we expect that the number of entries will be equal to the number of exits for that room for each day.

We count the number of entries to a room s_r (**Legal Access** to any door that leads to the room $e = (s, s_r)$) and subtract the number of exits from that room (**Free Egress** from any door that leads to the room $e = (s, s_r)$). If the result is negative, there are more exits than entries, which implies that someone has tailgated into the room.

We define an approach, *sum*, that performs that calculation at the end of the day as shown in Algorithm 5.1. We applied *sum* to the server room for the time periods recorded in the equipment room sign-in log so we could compare the results with the analyses later, in Section 5.6.3. The *sum* approach identified 18 occurrences of tailgating.

However, the *sum* approach does not take into account the space occupancy during the day. So we defined an approach, *ctr*, that extends *sum* by keeping a running count of the number of entries and exits to the room. The *ctr* approach is shown in Algorithm 5.2. Then, by Constraint 2 in Section 5.4, we can identify tailgating as soon as the room occupancy

falls below 0. The *ctr* approach identified 2 more occurrences of tailgating than *sum* did, and thus performed better than *sum* because it checks the number of accesses and egresses at each event. However, those two approaches do not take into account people who tailgate both in and out of the room. We tackle that issue in Section 5.6.3.

Algorithm 5.1 SUM algorithm

Require: A card tap log is a sequence of events, $X = (x_0 \dots x_n)$. For each event x_i, there is a $User$ and $AccessType$ field, $[User, Type] \in x_i$.

function SUM(X)

if $|x_i[Type] == LegalAccess| - |x_i[Type] == FreeEgress| < 0$ **then return** tailgating

end function

Algorithm 5.2 CTR algorithm

Require: A card tap log is a sequence of events, $X = (x_0 \dots x_n)$. For each event x_i, there is a $User$ and $AccessType$ field, $[User, Type] \in x_i$.

function CTR(X)

$score \leftarrow 0$

for all $x_i \in X$ **do**

if $x_i[Type] == LegalAccess$ **then** $score \leftarrow score + 1$ **end if**

if $x_i[Type] == FreeEgress$ **then** $score \leftarrow score - 1$ **end if**

if $score < 0$ **then return** tailgating **end if**

end for

end function

5.6.3 Using Sign-in Logs in Tandem with Card Tap Logs

A railway station consists mainly of staff rooms, storerooms, and equipment rooms. We focus on equipment rooms because they contain critical equipment for the running of the system. The equipment rooms have sign-in logs that contain information about the number of people in a group during a visit, so we can use that information to determine room occupancy more accurately.

Ideally, the number of people in the sign-in log would be equal to the number of people who tapped their cards. However, there can be policy violations in which a person taps to enter the room but is not in the sign-in log, and we indeed found instances of missing names of accompanying staff members in the sign-in logs. On the other hand, if a person appears

in the sign-in log but did not tap to enter the room, the *sum* and *ctr* approaches will fail to identify tailgating when there were instances of **Egress-In0-OutN** classes during the visit.

Therefore, we propose the *ctr-eqp* approach, which extends the *ctr* approach to account for the number of visitors tailgating into the room. The algorithm for *ctr-eqp* is shown in Algorithm 5.3. First, we check that the number of people in the sign-in log is equal to the number of people associated with the **Legal Access** events. If the sign-in log shows more people than the number who tapped their cards, someone must have tailgated into the room. Then, we keep a running count of entries and exits, much like the *ctr* approach, except that on the first **Legal Access** event, the running count is further increased by the number of visitors instead of just 1.

Algorithm 5.3 CTR-EQP algorithm

Require: A card tap log is a sequence of events, $X = (x_0 \dots x_n)$. For each event x_i, there is a $User$ and $AccessType$ field, $[User, Type] \in x_i$. Equipment room sign-in logs have visits annotated with the total number of people visiting, $total\_people$, and the number of those people who have the visitor role $num\_visitors$.

```
function CTR-EQP(X, total_people, num_visitors)
    if |x_i[User]| < total_people then return tailgate end if
    score ← 0
    for all x_i ∈ X do
        if x_i[Type] == LegalAccess then
            if score == 0 then score ← score + num_visitors end if
            score ← score + 1
        end if
        if x[Type] == FreeEgress then score ← score − 1 end if
        if score < 0 then return tailgate end if
    end for
end function
```

The *ctr-eqp* approach identified 22 more occurrences of tailgating than *ctr* did, as shown in Figure 5.4. Thus, the tailgating instances identified by the *ctr-eqp* approach are a superset of the instances identified by the *sum* or *ctr* approach, which shows that inclusion of additional data sources like the sign-in logs help to identify tailgating.

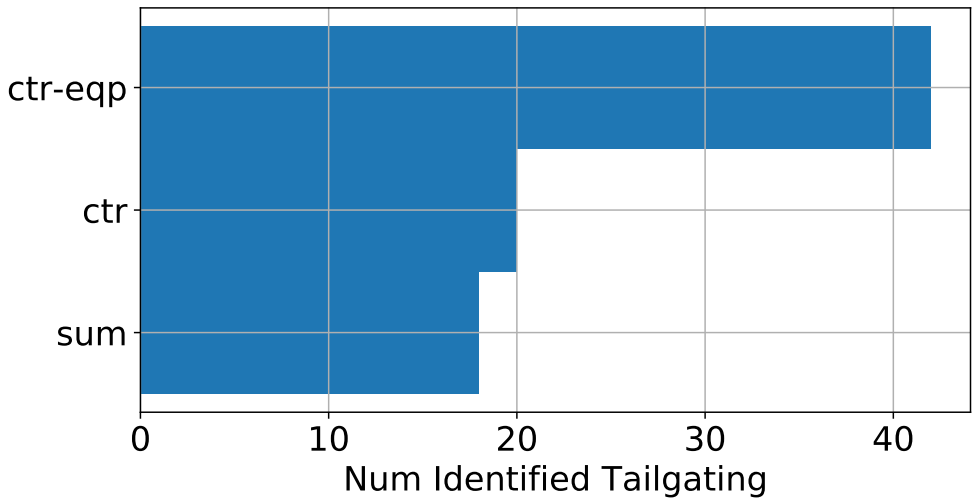


Figure 5.4: Identified occurrences of tailgating by *sum*, *ctr*, and *ctr-eqp*.

5.7 DISCUSSION

Dataset Limitations. Manual sign-in logs are often not fully accurate or complete. In particular, we found that around 60% of the accesses to the server room were not logged as visits in the sign-in logs, and 5 of the 33 dates in the maintenance schedule do not have corresponding entries in the sign-in logs. In addition, the names of all the accompanying staff members are not always written in the log, and we had to infer those missing staff members using the card swipe logs. The sign-in and sign-out times in the logs are sometimes missing or incomplete and are just a rough estimate of the duration of the visit. Thus, it is important to understand and acknowledge the limitations of the sign-in log and perform further cross-checks of the information in that log with other forms of data sources.

Approach Generality. Our topology-based approach is applicable to any building since it is based on physical limitations of human movement in an indoor setting. Of our room occupancy-based approaches, *ctr* is also applicable to any building because it capitalizes on the constraint that occupancy of a space should not fall below 0. On the other hand, *sum* and *ctr-eqp* approaches rely on domain knowledge and additional data sources, in the case of *ctr-eqp*.

Attacker Model and Limitations. If an attacker tailgates into any space in the station, they need to have first signed in at the station entrance with a staff member and entered into the space with a non-malicious staff member. We assume that the staff member will ensure that the attacker has logged their entry in the sign-in log book. However, our approaches

do not distinguish between "benign" tailgating and "suspicious" tailgating that is the result of malicious behavior. As we see in our results, a large portion of the tailgating instances that we uncovered were due to visitors who needed to be escorted by railway staff members. In those cases, tailgating was necessary for them to do their jobs. While it can be argued that such instances need not be identified, we strongly believe that it is essential to uncover them, because they give insight into potential policy violations (e.g., a visitor's borrowing of a staff member's card for convenience).

In our analysis of the identified tailgating instances, we also found interesting quirks in the card tap logs. For example, one of the visits to the server room had only a single **Free Egress** event in the card tap log, with no corresponding **Legal Access** event within the span of a few days. Occasionally, a **Legal Access** event had no user identification associated with it, and in one instance, an **Invalid Attempt** event resulted in entry of people into the room. Discussions with our railway system collaborator suggested that the user may have been tapping at the reader with multiple cards and the reader read the wrong card.

5.8 SECURITY ANALYSIS: EVALUATING OUR SYSTEM DEFENSE APPROACHES

In Section 3.8, we used the ADVISE model to identify the likely course of action an adversary will take to compromise the safety of the system. Our findings showed that a stealthy adversary would bribe an insider to use his or her credentials to gain physical access to the system. Thus, in this chapter and in Chapter 4, we proposed approaches to detect abnormal physical movement behavior by insiders. In this section, we want to analyze the extent to which our proposed approaches protect the system against such attack campaigns.

5.8.1 Updated ADVISE Model

We use the results from our proposed detection systems to update the ADVISE model in Section 3.8. Specifically, we assign detection probabilities to the **Move** attack step in the ontology.

If the insider uses his or her physical access card to enter the station, then our approach in Chapter 4 will attempt to detect abnormal sequences of card taps. In Section 4.6.2, we calculated the false negative rates for the first, second, third, and fourth accesses of a user. Thus, we can annotate each directed path in the system instance diagram with the corresponding detection probability, as shown in the second column of Table 5.2.

Otherwise, if the insider tailgates another staff member into the station, then our approach in Chapter 4 is unable to detect the physical movement. According to protocol, the insider

Table 5.2: Detection probabilities for the **Move** attack step.

Move To	**Detection Probability**	
	Chapter 4	**Chapter 5**
Concourse	0.68	0
Corr 1	0.68	0
PSC	0.88	0
Corr 2	0.88	0
Corr 3	0.88	0
Server	0.91	1
Power	0.91	1
Signaling	0.91	1
Platform	0	0

and staff member would record their presence in the log book located within the equipment room. Then our *ctr-eqp* approach from Section 5.6.3 will always detect the tailgating instance, but only after the attack has taken place.

Our *ctr* and *sum* approaches may still be able to detect the tailgating instance if the insider and staff member have a combination of accesses and exits that do not fulfill the constraints we defined. However, we are unable to assign concrete detection probabilities without further ground truth data to statistically verify the accuracy of those approaches. We therefore assume the worst-case scenario in which the *ctr* and *sum* approaches are unable to detect the tailgating instance.

We create a separate system instance diagram to model the detection probabilities of the *ctr-eqp* approach as shown in the third column of Table 5.2. Only the **Move** attack steps to a **Room** will have a detection probability of 1, whereas other **Move** attack steps will have a detection probability of 0. Since the *PSC* is the only **Room** that is not an equipment room, the **Move** attack step to the *PSC* will have a detection probability of 0.

5.8.2 Experiment Results

We want to determine whether our detection approaches will alter the attack campaign chosen by the adversary. In Section 3.8, we found that a stealthy adversary who stands to lose more than $500,000 will choose the **BribeInsider** attack step.

We varied the preference weight (or the adversary's stealthiness) from $0 to $5,000,000. The simulation results show that if the insider uses his or her physical access card, then the payoff of bribing an insider is consistently lower than that of using brute force to enter the station, because the insider is almost as likely to be detected as an outsider. Since our

detection system from Chapter 4 operates in an online manner, the time to detection would be comparable to that taken by a staff member observing video surveillance. Therefore, our online detection approach from Chapter 4 increases the security of the system.

Our results show that for weights between \$0 and \$3,300,000, the adversary chooses the **BruteForce** attack step and performs the **CutCable** attack on the connection between the *CBI Server* and the *Points* machine. If the weight is above \$3,300,000, the adversary chooses to perform the **HackComms** attack step on the *Laptop* and performs the **RemoteCtrl** attack on the *Train*.

Previously, only adversaries who stood to lose less than \$500,000 would choose to enter the station via brute force. Those adversaries who stood to lose more than \$500,000 would choose to bribe an insider. So with the addition of our detection system, more adversaries would choose to enter the station via brute force. Those adversaries would be detected while performing the attack steps. Only a very stealthy adversary (e.g., a nation-state adversary) would attempt to hack the communications between the train and the servers, despite the large amount of time needed to collect the data needed.

However, if the insider is able to tailgate behind another staff member, then the payoff of bribing an insider is consistently higher than that of hacking the communications of the train. Our results show that if the weight is above \$500,000, the adversary will perform the **BribeInsider** attack step and tailgate into the **PSC** to **Login** to the *Host* and **RemoteCtrl** the *Train*.

We can, however, develop our detection approach further to infer tailgating instances for other spaces, and the *PSC* in particular, using other sources of information. In particular, we can identify cyber logins to terminals in the PSC by using the device event logs. Then, if we find a user login to the terminal in the PSC but no card tap associated with that user, that implies the user has tailgated into the space. Therefore, we can assign a detection probability of 1 to the **Move** attack step into the PSC.

With that improvement to our detection approach, the payoff of bribing an insider to tailgate another staff member into the station is now consistently lower than that of entering the station through brute force. Then, our results show that for weights between \$0 and \$3,300,000, the adversary chooses the **BruteForce** attack step, and for weights above \$3,300,000, the adversary chooses the **HackComms** attack step. Therefore, the inclusion of our detection approach has also incentivized the adversary away from bribing an insider.

The drawback to our detection approach is that the defender can infer that tailgating, and therefore the attack, has occurred only after the attack has taken place. However, that post-mortem detection analysis can inform the defenders to whom they should attribute the attack. We also believe that the time to detection for our approach would be short because

the forensics team only needs to obtain the station sign-in log book, equipment room sign-in log books, and card tap logs. Corroboration of those data does not take much time, since there are not many accesses to equipment rooms in a day. In comparison, forensic analyses of the terminals take much more time to detect the **DirectControl** attack step by an insider if he or she has inserted malware to erase or hide traces of malicious activity in the host logs.

5.9 CONCLUSION

Physical security is essential to protect the equipment inside critical infrastructure facilities. However, building access control systems that rely on card readers and door movement sensors are inadequate to prevent tailgating behavior that potentially violates physical access control policies.

In this chapter, we developed approaches to detect tailgating that use existing data sources in railway stations. We defined several physical constraints on human movement that drove our approach to inferring tailgating from card tap logs. We analyzed the movement trajectories derived from the card tap logs and found 3,999 instances of tailgating throughout the building. Then, we narrowed our focus to critical equipment rooms and proposed two approaches that keep track of room occupancy using only the card tap logs, and another approach that uses equipment room sign-in logs together with the card tap logs to identify tailgating. Our results show that the equipment room sign-in logs are very useful in identifying tailgating.

We also reapplied our approach from Chapter 2 to analyze how our detection approaches from this chapter and Chapter 4 would affect the decisions made by adversaries. Our results show that our detection approaches would incentivize the adversaries to either attack the communication messages to the train or, in most cases, perform brute-force attacks to enter the station instead of bribing insiders. Therefore, our detection approaches have strengthened the security of the system.

CHAPTER 6: CONCLUSION

Today, the increased interconnections among devices in critical infrastructure systems have improved the system's efficiency at the cost of a larger attack surface. Malicious adversaries are now able to launch both physical and cyber attacks to compromise the safety of systems, resulting in real-world consequences that impact citizens. Thus, it is important to protect such critical infrastructure systems from both physical and cyber threats.

In this dissertation, we presented a model-driven framework that combines information from the cyber, physical, and human domains to analyze a system's state and defend against malicious physical attacks. Specifically, our contributions are (1) a novel ontology to represent cyber-physical systems and cyber and physical actions by an adversary for security analysis, (2) a strategic approach for representing combinations of cyber actions of an adversary for safety verification, (3) the first large-scale analysis of human movement for detection of malicious behavior, and (4) the first real-life study of identification of tailgating. We applied our contributions to a railway system case study and showed that it supports our thesis statement, which is that the protection of critical infrastructure systems requires an integration of security analysis, safety analysis, and system defense that is enabled by the cross-domain models that we developed. In this chapter, we briefly review our contributions and describe further avenues of work.

6.1 REVIEW OF CONTRIBUTIONS

First, we presented a novel ontology that represents the cyber and physical system components, the relationships among them, and the cyber and physical actions of a human actor, for the purposes of security analysis. We modeled a railway station using concepts from the ontology and input that model into the ADVISE tool, which then generated an attack execution graph automatically. We used that attack execution graph to analyze the potential attack paths of an adversary and showed that the addition of a defense system for physical movement of humans would help to strengthen the security of the system.

Then, we presented a safety analysis of the railway system that identified potential cyber threats to the signaling system. We used networks of timed automata to model the cyber-physical control feedback loop of the signaling system that drives system service. We developed a set of transformations on those state automata to represent different combinations of cyber actions taken by a human adversary. Then, we performed model checking to identify which combinations of attacks would violate system safety. We found potential

threat vectors that could be utilized by outsider and insider adversaries and showed the effects of several safety countermeasures in mitigating those threat vectors. We reapplied our security analysis with the addition of those cyber threat vectors and showed that adversaries would respond to safety countermeasures by exploiting physical and social means rather than cyber means to compromise system safety.

To strengthen the security of the system, we presented two detection systems to identify suspicious physical movement of humans. Our first detection system identifies abnormal sequences of card taps by humans by employing socio-technical models to represent normal patterns of movement behavior of human actors in an indoor setting. We used historic physical access logs collected from building access control systems to build Markov models of normal movement behavior for each human actor. Then, in real-time, we screened physical card taps and compared the current sequence of card taps to those models to detect deviations from the models. We showed that our detection system can detect suspicious movement with increasing likelihood as the length of the suspicious movement sequence increases. However, our detection system is unable to capture suspicious movement that does not involve the tapping of an access card.

Thus, we developed our second detection system that uses cyber and physical data sources to identify tailgating movement behavior. We identified physical constraints on human movement based on our domain knowledge. Then, we developed several approaches that track the physical location of human actors and identify instances of tailgating when those physical constraints are violated. We showed that our detection system can identify certain scenarios of tailgating behavior and that inclusion of additional physical data sources helps to build a more complete user location model. We reapplied our security analysis and showed that our detection systems can incentivize adversaries to pursue a cyber-attack campaign that would take considerable time and effort instead of bribing insiders.

6.2 FUTURE DIRECTIONS

There are many avenues of future work that we can pursue to advance the state of safety and security in critical infrastructure systems. We will now discuss the different technical improvements that can be done for each chapter's contribution. Then, we propose further extensions of the work and conclude with a discussion about the future of our proposed framework.

6.2.1 Technical Improvements

In Chapter 2, we constructed an ontology for cyber-physical systems that models a system's cyber and physical components, and cyber and physical actions on that system. The generated attack execution graph is only as complete as that ontology. Therefore, our ontology could be extended in the future to model various types of devices and possible attack steps and their consequences.

We currently model physical consequences of a human action as attack steps in the ontology that are performed by an adversary. However, the physical consequences of an action can take effect immediately and are not themselves actions performed by a human actor. The physical consequences could be modeled instead by composing the attack execution graph with *stochastic activity networks* (SANs), fault trees, or hybrid attack graphs that model the system functions.

It is difficult to perform such a composition with the current version of the ADVISE tool, because the tool's simulation of the adversary's decision-making process is unable to view the effects of certain actions outside the attack execution graph generation and would not be able to account for complex interactions between different models. Thus, the adversary's modeled decision-making process would not have the foresight to attempt certain actions. One could explore the different ways in which the tool's simulation algorithm could have full visibility into the workings of the system to better inform the adversary's decision-making process. We envision that this feature will be part of future releases of the tool.

In Chapter 3, we used statistical model checking to verify the safety of the system. Although statistical model checking provides a good measure of system safety, we want to be able to say with certainty that safety is guaranteed under certain conditions. One avenue of further work is to reduce model complexity by removing the differential equation that represents the train movement and discretizing certain elements of the model. One could also bound the model to prevent state-space explosion. Then, UPPAAL's built-in model checker could be used to explore the full state space.

We modeled attacks on network messages and commands in Chapter 3. Other attacks on the control feedback loop, such as the traction power supplied to the track, relevant signals sent to the train, and the environmental conditions of the railway tunnel that are controlled by computing devices in the station, could be considered.

In Chapter 4, we compared our Markov models to Bayesian network models and found that 2-node Bayesian network models also had high accuracy in detecting abnormal movement sequences. One could analyze the users' movement sequences further to determine under what conditions Markov or Bayesian network models should be used.

Finally, in Chapter 5, we conducted an observation study to determine how often each movement scenario occurs in real life. That study could be extended to obtain more ground truth data and verify the results of our tailgating detection approaches. Our approaches could also be extended to look at events at other locations instead of only a specific space. For example, if a person taps somewhere outside a room, we can infer that he or she has left the room, but that only allows us to associate egresses with that person.

6.2.2 Extending Approach Generality

In our dissertation, we have chosen several key components of the railway system to model and analyze. In particular, we considered one of the 13 railway stations on the railway line and performed a security analysis. We also collected various cyber and physical data logs from that station, including physical access logs and equipment room sign-in logs. Although all of the railway stations are similar in terms of the types of rooms and devices contained within them, the physical architecture, number of rooms, and devices differ from one station to another.

Therefore, our analyses could be extended to all the railway stations. In particular, our system instance diagram from Chapter 2 could be extended to the full railway system, and one could evaluate how different station architectures affect the ability of adversaries to achieve their goals. One could also analyze how different physical architectures affect the movement of humans, especially those railway staff who visit different stations to perform their duties.

The goal of our detection systems for physical movement is to identify abnormal movement behavior. However, not all abnormal behavior is the result of malicious intent. For example, our tailgating detection approach can only determine that tailgating has happened and is unable to identify when that tailgating happened or whether that tailgating event can be attributed to benign or malicious purposes. Other information about the system state could be used to provide more context for the movement behavior. We collected several types of cyber and physical data that were not put to full use in our dissertation. For instance, we used the device event logs to extract information about device failures in 4 rooms of a station to enhance the movement models for maintenance staff. That work could be extended to use device event logs to determine when people are interacting with the devices in a room. The main challenge in using that data source is that of correctly attributing changes in the device state to physical interaction with a person instead of automated physical processes or remotely controlled actions. Device event logs could also be combined with cyber login data to determine whether a person is attempting to perform malicious actions on a device.

We also collected station sign-in logs that identify the people who should be in the station at any given point in time. Much like the equipment room sign-in logs, that knowledge can help us find instances of tailgating. However, it is harder to use station sign-in logs than equipment room sign-in logs because we only have the information that a certain person is in the building at a given time; we cannot pinpoint his or her location. Thus, building topology could be used to track the movement of all users in the building to correlate the **Legal Access** and **Free Egress** events.

In Chapter 3, we analyzed the safety of a diamond crossing by taking into consideration the possible attacks on the network protocol and commands issued to point machines. One could also investigate safety in other locations on the railway line, such as railway yards, to gain a better understanding of the overall system safety. We could reapply our model to those locations by modifying the track topology model and routing table functions in our server automaton. While we have not considered system availability in this dissertation, our analysis could be extended to include the impact of attacks on system service.

6.2.3 Towards a Comprehensive Framework

We envision that our framework for security analysis, safety analysis, and system defense can be applied to other critical infrastructure systems to provide insight into the threats posed by different adversary profiles and suggest countermeasures to put in place.

Our framework abstracts model elements to reduce the amount of system practitioners' effort needed to construct the models needed for analysis. For example, when there are changes in the system or updates to the possible attacks, a system practitioner need only modify the ontology and system model in Chapter 2 instead of the attack execution graph. In our works' current state, the models we discussed in different chapters do not share common definitions. Thus, work needs to be done to integrate common pieces of information, such as the system's architecture, which is used in Chapters 2, 4, and 5.

Once the inputs to each approach have been consolidated, the system-specific inputs of our approaches can be divided into three components: system architecture and service, system data, and adversary profiles. In particular, if system practitioners want to apply our framework to their critical infrastructure systems, they will perform the following steps.

1. Model the system's physical and cyber network architecture and interdependencies, using the classes and relations in the base ontology.

2. Use networks of timed automata to model the control feedback loop that drives system service.

3. Input system logs, including building access control logs, to the physical defense system.

4. Model the different adversary profiles and goals expressed in terms of the system model.

5. Restrict the potential combination of attacks on the system based on domain knowledge.

Once they have done so, our security analysis approach from Chapter 2 will use the system model from step 1, the adversary profiles and goals from step 4, and the attacks from step 5 to generate the attack execution graph. Our safety analysis approach from Chapter 3 will use the model from step 2, the adversary profiles and goals from step 4, and the attacks from step 5 to verify the safety of the system. Finally, our system defense approaches from Chapters 4 and 5 will use the system model from step 1 and the system logs from step 3 to generate models of user behavior and user location.

6.3 CONCLUDING REMARKS

There are many complex interactions in critical infrastructure systems that can be exploited by attackers to compromise system safety. During the course of our work, we found that modeling the intricacies of human behavior in the context of the interaction between the human actor and the system is the most difficult task in our modeling approaches. The reason is that it is hard to claim completeness of the possible malicious scenarios that may happen in a system.

In our dissertation, we model the preconditions and effects of a human action. It is important also to model the duration of a human action because it affects the time to response of detection systems and system defenders. However, we model only the duration of a human action in Chapter 3, because we restricted the possible sets of scenarios that can occur. It is difficult to specify a suitable range for the duration of a human action because different adversaries may take different amounts of time to accomplish an action. In addition, there may be multiple human actors performing malicious actions to achieve their goal. Adversaries may also employ deception techniques or actions to mask their malicious actions. Therefore, the modeling of human-initiated cyber and physical actions on a system is a continuing research area that requires the fostering of open collaborations among domain experts, system operators, and researchers in an effort to better understand and protect such systems.

We have learned many valuable lessons from interacting with our railway system collaborators. Our discussions about potential attacker scenarios exposed differences of opinion

with regard to the ability of an attacker to achieve certain attack goals. While our collaborators were focused on the practical possibilities, we were more concerned with the theoretical possibilities. The results of our security analysis did show that our collaborators were correct in assuming the limits of certain attackers. However, the theoretical attacks that we exposed were important for our collaborators to be aware of.

Our collaborators were also open to providing data, for which we are very grateful. Those data allowed us to identify problems with their system and present solutions and analyses to them for potential deployment in the future.

In conclusion, we believe that protection of critical infrastructure systems is a team effort by domain experts, system operators, and researchers that involves development of analytic techniques enabled by models that integrate the cyber, physical, and human domains of the system.

REFERENCES

[1] Department of Homeland Security, "Homeland security presidential directive 7: Critical infrastructure identification, prioritization, and protection," https://www.dhs.gov/homeland-security-presidential-directive-7, September 22, 2015.

[2] D. Kushner, "The real story of Stuxnet," IEEE Spectrum, vol. 50, no. 3, pp. 48–53, 2013.

[3] K. Zetter, "Inside the cunning, unprecedented hack of Ukraine's power grid," Wired, March 3, 2016. [Online]. Available: https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid

[4] Department of Defense, "System safety," https://www.system-safety.org/Documents/MIL-STD-882E.pdf, May 11, 2012.

[5] J. Smith, S. Russell, and M. Looi, "Security as a safety issue in rail communications," in Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software, vol. 33. Australian Computer Society, Inc., 2003, pp. 79–88.

[6] N. Howe, "Cybersecurity in railway signalling systems," Institution of Railway Signal Engineers News, vol. 236, pp. 1–4, 2017.

[7] W. G. Temple, Y. Wu, B. Chen, and Z. Kalbarczyk, "Reconciling systems-theoretic and component-centric methods for safety and security co-analysis," in Computer Safety, Reliability, and Security, S. Tonetta, E. Schoitsch, and F. Bitsch, Eds. Cham: Springer, 2017, pp. 87–93.

[8] W. Young and N. Leveson, "Systems thinking for safety and security," in Proceedings of the 29th Annual Computer Security Applications Conference. ACM, 2013, pp. 1–8.

[9] G. Macher, A. Höller, H. Sporer, E. Armengaud, and C. Kreiner, "A combined safety-hazards and security-threat analysis method for automotive systems," in Computer Safety, Reliability, and Security, F. Koornneef and C. van Gulijk, Eds. Cham: Springer International Publishing, 2015, pp. 237–250.

[10] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," ACM Computing Surveys, vol. 46, no. 4, pp. 55:1–55:29, Mar. 2014.

[11] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy and survey of intrusion detection system design techniques, network threats and datasets," 2018. [Online]. Available: http://arxiv.org/abs/1806.03517

[12] B. Chen, C. Schmittner, Z. Ma, W. G. Temple, X. Dong, D. L. Jones, and W. H. Sanders, "Security analysis of urban railway systems: The need for a cyber-physical perspective," in Proc. 33rd International Conference on Computer Safety, Reliability, and Security, F. Koornneef and C. van Gulijk, Eds. Springer International Publishing, 2015, pp. 277–290.

[13] C. Tan, "Mysterious signal fault hits Circle Line again," The Straits Times, November 3, 2016. [Online]. Available: http://www.straitstimes.com/singapore/transport/mysterious-signal-fault-hits-circle-line-again

[14] D. Sim, "How the Circle Line rogue train was caught with data," Datagovsg, November 30, 2016. [Online]. Available: https://blog.data.gov.sg/how-we-caught-the-circle-line-rogue-train-with-data-79405c86ab6a

[15] C. Tan, "Joint news release by the Land Transport Authority (LTA) & SMRT - Circle Line signaling problems caused by intermittent failure of signaling hardware on train," Land Transport Authority, November 11, 2016. [Online]. Available: https://www.lta.gov.sg/apps/news/page.aspx?c=2&id=89defe38-427b-4cfe-b69d-4e35ca7d0f64

[16] G. Baker, "Schoolboy hacks into city's tram system," The Telegraph, January 11, 2008. [Online]. Available: http://www.telegraph.co.uk/news/worldnews/1575293/Schoolboy-hacks-into-citys-tram-system.html

[17] S. Grad, "Engineers who hacked into L.A. traffic signal computer, jamming streets, sentenced," Los Angeles Times, December 1, 2009. [Online]. Available: http://latimesblogs.latimes.com/lanow/2009/12/engineers-who-hacked-in-la-traffic-signal-computers-jamming-traffic-sentenced.html

[18] Y. Sin, "NSL disruption: Water pumping system malfunction at Bishan MRT station caused tunnel to flood, says LTA," The Straits Times, October 8, 2017. [Online]. Available: http://www.straitstimes.com/singapore/transport/nsl-disruption-water-collecting-in-tunnels-was-due-to-malfunction-in-water

[19] R. Sim, "Disruptions, flooding, fake work records: How systemic are SMRT's cultural issues?" The Straits Times, November 5, 2017. [Online]. Available: http://www.straitstimes.com/singapore/transport/how-systemic-are-smrts-cultural-issues

[20] E. LeMay, "Adversary-driven state-based system security evaluation," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, Illinois, 2011.

[21] R. A. Serrano and E. Halper, "Sophisticated but low-tech power grid attack baffles authorities," Los Angeles Times, February 11, 2014. [Online]. Available: http://www.latimes.com/nation/la-na-grid-attack-20140211-story.html

[22] J. Pagliery, "Sniper attack on California power grid may have been 'an insider,' DHS says," CNN, October 17, 2015. [Online]. Available: http://money.cnn.com/2015/10/16/technology/sniper-power-grid/

[23] Ponemon Institute LLC, "Cybersecurity in Operational Technology: 7 Insights You Need to Know," Tenable, Tech. Rep., 2019.

[24] B. Feddersen, K. Keefe, W. H. Sanders, C. Muehrcke, D. Parks, A. Crapo, A. Gabaldon, and R. Palla, "Enterprise security metrics with the ADVISE meta model formalism," in Proc. 9th International Conference on Emerging Security Information, Systems, and Technologies, Venice, Italy, Aug. 23, 2015, pp. 65–66.

[25] S. Jajodia, S. Noel, and B. O'Berry, "Topological analysis of network attack vulnerability," in Managing Cyber Threats: Issues, Approaches, and Challenges, V. Kumar, J. Srivastava, and A. Lazarevic, Eds. Springer, 2005, pp. 247–266.

[26] S. Jajodia, S. Noel, P. Kalapa, M. Albanese, and J. Williams, "Cauldron mission-centric cyber situational awareness with defense in depth," in Proc. of the Military Communications Conference 2011, Nov. 2011, pp. 1339–1344.

[27] E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke, "Model-based security metrics using ADversary VIew Security Evaluation (ADVISE)," in Proc. 8th International Conference on Quantitative Evaluation of SysTems, Aachen, Germany, Sept. 5–8, 2011, pp. 191–200.

[28] J. Gash, "Physical operating environment: How the cyber-electromagnetic environment fits," in Canadian Military Journal, 2012, vol. 12, no. 3, pp. 28–34.

[29] M. Krötzsch, F. Simančík, and I. Horrocks, "A description logic primer," Computing Research Repository, 2012. [Online]. Available: http://arxiv.org/abs/1201.4089

[30] S. More, M. Matthews, A. Joshi, and T. Finin, "A knowledge-based approach to intrusion detection modeling," in Proc. IEEE Symposium on Security and Privacy Workshops. IEEE, 2012, pp. 75–81.

[31] B. Shepard, C. Matuszek, C. B. Fraser, W. Wechtenhiser, D. Crabbe, Z. Güngördü, J. Jantos, T. Hughes, L. Lefkowitz, M. Witbrock, D. Lenat, and E. Larson, "A knowledge-based approach to network security: Applying Cyc in the domain of network risk assessment," in Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence, vol. 3. AAAI Press, 2005, pp. 1563–1568.

[32] H. Holm, K. Shahzad, M. Buschle, and M. Ekstedt, "P^2 CySeMoL: Predictive, probabilistic cyber security modeling language," IEEE Transactions on Dependable and Secure Computing, vol. 12, no. 6, pp. 626–639, Nov, 2015.

[33] S. Marrone, R. J. Rodriguez, R. Nardone, F. Flammini, and V. Vittorini, "On synergies of cyber and physical security modelling in vulnerability assessment of railway systems," Computers & Electrical Engineering, vol. 47, pp. 275–285, 2015.

[34] The TREsPASS Project, "Technology-supported risk estimation by predictive assessment of socio-technical security," https://www.trespass-project.eu/, 2017.

[35] C. W. Probst, J. Willemson, and W. Pieters, "The attack navigator (invited)," in Graphical Models for Security - Revised Selected Papers, ser. Lecture Notes in Computer Science, S. Mauw, B. Kordy, and S. Jajodia, Eds. Berlin: Springer Verlag, February 2016, vol. 9390, p. 1–17.

[36] P. J. Hawrylak, M. Haney, M. Papa, and J. Hale, "Using hybrid attack graphs to model cyber-physical attacks in the smart grid," in Proc. of 5th International Symposium on Resilient Control Systems, Aug, 2012, pp. 161–164.

[37] J. Rouvroye and E. van den Bliek, "Comparing safety analysis techniques," Reliability Engineering & System Safety, vol. 75, no. 3, pp. 289–294, 2002.

[38] E. Denney and G. Pai, "Evidence arguments for using formal methods in software certification," in Proc. IEEE International Symposium on Software Reliability Engineering Workshops, 2013, pp. 375–380.

[39] "UPPAAL," http://www.uppaal.org/, accessed: 2018-05-20.

[40] A. Borälv, "The industrial success of verification tools based on Stålmarck's method," in Computer Aided Verification, O. Grumberg, Ed. Springer, 1997, pp. 7–10.

[41] A. Fantechi, "Twenty-five years of formal methods and railways: What next?" in Software Engineering and Formal Methods, S. Counsell and M. Núñez, Eds. Cham: Springer, 2014, pp. 167–183.

[42] S. Busard, Q. Cappart, C. Limbrée, C. Pecheur, and P. Schaus, "Verification of railway interlocking systems," in Proc. 4th International Workshop on Engineering Safety and Security Systems, 2015, pp. 19–31.

[43] M. Oz and O. Kaymakci, "An automatic formal model generation and verification method for railway interlocking systems," Gazi University Journal of Science, vol. 30, no. 2, pp. 133–147, 2017.

[44] H. Wang, F. Schmid, L. Chen, C. Roberts, and T. Xu, "A topology-based model for railway train control systems," IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 2, pp. 819–827, 2013.

[45] S. Qiu, M. Sallak, W. Schön, and Z. Cherfi-Boulanger, "Availability assessment of railway signalling systems with uncertainty analysis using statecharts," Simulation Modelling Practice and Theory, vol. 47, pp. 1–18, 2014.

[46] S. Ghosh, P. Dasgupta, C. Mandal, and A. Katiyar, "Formal verification of movement authorities in automatic train control systems," in Proc. International Conference on Railway Engineering, 2016, pp. 1–8.

[47] P. di Tommaso, F. Flammini, A. Lazzaro, R. Pellecchia, and A. Sanseviero, "The simulation of anomalies in the functional testing of the ERTMS/ETCS trackside system," in Proc. 9th IEEE International Symposium on High-Assurance Systems Engineering, 2005, pp. 131–139.

[48] X. Han, T. Tang, J. Lv, and H. Wang, "Failure analysis of Chinese train control system level 3 based on model checking," in Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification, T. Lecomte, R. Pinger, and A. Romanovsky, Eds. Cham: Springer International Publishing, 2016, pp. 95–105.

[49] Q. Cappart, C. Limbrée, P. Schaus, J. Quilbeuf, L. Traonouez, and A. Legay, "Verification of interlocking systems using statistical model checking," in Proc. IEEE 18th International Symposium on High Assurance Systems Engineering, 2017, pp. 61–68.

[50] R. Bloomfield, M. Bendele, P. Bishop, R. Stroud, and S. Tonks, "The risk assessment of ERTMS-based railway systems from a cyber security perspective: Methodology and lessons learned," in Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification, T. Lecomte, R. Pinger, and A. Romanovsky, Eds. Cham: Springer International Publishing, 2016, pp. 3–19.

[51] M. Rocchetto and N. O. Tippenhauer, "Towards formal security analysis of industrial control systems," in Proc. of the ACM Asia Conference on Computer and Communications Security. ACM, 2017, pp. 114–126.

[52] D. Dolev and A. Yao, "On the security of public key protocols," IEEE Transactions on Information Theory, vol. 29, no. 2, pp. 198–208, 1983.

[53] M. Rocchetto and N. O. Tippenhauer, "CPDY: Extending the Dolev-Yao attacker with physical-layer interactions," in Formal Methods and Software Engineering, K. Ogata, M. Lawford, and S. Liu, Eds. Cham: Springer International Publishing, 2016, pp. 175–192.

[54] M. Puys, M.-L. Potet, and A. Khaled, "Generation of applicative attacks scenarios against industrial systems," in Foundations and Practice of Security, A. Imine, J. M. Fernandez, J.-Y. Marion, L. Logrippo, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2018, pp. 127–143.

[55] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, "UPPAAL SMC tutorial," International Journal on Software Tools for Technology Transfer, vol. 17, no. 4, pp. 397–415, 2015.

[56] T. Chothia, M. Ordean, J. de Ruiter, and R. J. Thomas, "An attack against message authentication in the ERTMS train to trackside communication protocols," in Proc. of the Asia Conference on Computer and Communications Security. ACM, 2017, pp. 743–756.

[57] S. Y. Chang, B. A. N. Tran, Y. C. Hu, and D. L. Jones, "Jamming with power boost: Leaky waveguide vulnerability in train systems," in Proc. IEEE 21st International Conference on Parallel and Distributed Systems, 2015, pp. 37–43.

[58] M. Heddebaut, S. Mili, D. Sodoyer, E. Jacob, M. Aguado, C. P. Zamalloa, I. Lopez, and V. Deniau, "Towards a resilient railway communication network against electromagnetic attacks," in Proceedings of the Transport Research Arena 5th Conference: Transport Solutions from Research to Deployment, 2014, pp. 1–10.

[59] I. Lopez and M. Aguado, "Cyber security analysis of the European train control system," IEEE Communications Magazine, vol. 53, no. 10, pp. 110–116, 2015.

[60] "A survey of the cyber security landscape," Cyber Security Intelligence Index, IBM X-Force Research, 2016.

[61] M. Salem, S. Hershkop, and S. J. Stolfo, "A survey of insider attack detection research," in Insider Attack and Cyber Security: Beyond the Hacker. Springer, 2008, pp. 69–90.

[62] Alien Vault, "Insider threat detection software," https://www.alienvault.com/, 2016.

[63] Tripwire, "Insider threat security & detection," http://www.tripwire.com/, 2016.

[64] CERT Insider Threat Center, "Insider threat and physical security of organizations," https://insights.sei.cmu.edu/insider-threat/2011/05/insider-threat-and-physical-security-of-organizations.html, 2011.

[65] M. E. Luallen, "Managing insiders in utility control environments," SANS Institute, Tech. Rep., 2011. [Online]. Available: https://www.sans.org/reading-room/whitepapers/analyst/managing-insiders-utility-control-environments-34960

[66] L. Bauer, L. F. Cranor, R. W. Reeder, M. K. Reiter, and K. Vaniea, "Real life challenges in access-control management," in Proc. ACM SIGCHI Conference on Human Factors in Computing Systems, 2009, pp. 899–908.

[67] S. M. Pincus, "Approximate entropy as a measure of system complexity." Proceedings of the National Academy of Sciences, vol. 88, no. 6, pp. 2297–2301, 1991.

[68] X. Li, "Using complexity measures of movement for automatically detecting movement types of unknown GPS trajectories," American Journal of Geographic Information System, vol. 3, no. 2, pp. 63–74, 2014.

[69] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," Science, vol. 327, no. 5968, pp. 1018–1021, 2010.

[70] C. Cheh, B. Chen, W. G. Temple, and W. H. Sanders, "Data-driven model-based detection of malicious insiders via physical access logs," in Proc. 14th International Conference on Quantitative Evaluation of Systems, N. Bertrand and L. Bortolussi, Eds. Springer International Publishing, 2017, pp. 275–291.

[71] P. Billingsley, "Statistical methods in markov chains," The Annals of Mathematical Statistics, vol. 32, no. 1, pp. 12–40, 1961.

[72] "Bayes Server," https://www.bayesserver.com/, accessed: 2018-10-20.

[73] A. D. Kent, L. M. Liebrock, and J. C. Neil, "Authentication graphs: Analyzing user behavior within an enterprise network," Computers & Security, vol. 48, pp. 150–166, 2015.

[74] G. Pallotta and A. L. Jousselme, "Data-driven detection and context-based classification of maritime anomalies," in Proc. 18th International Conference on Information Fusion, 2015, pp. 1152–1159.

[75] A. N. Radon, K. Wang, U. Glasser, H. Wehn, and A. Westwell-Roper, "Contextual verification for false alarm reduction in maritime anomaly detection," in Proc. IEEE International Conference on Big Data, 2015, pp. 1123–1133.

[76] M. Dash, K. K. Koo, J. B. Gomes, S. P. Krishnaswamy, D. Rugeles, and A. Shi-Nash, "Next place prediction by understanding mobility patterns," in Proc. IEEE International Conference on Pervasive Computing and Communication Workshops, 2015, pp. 469–474.

[77] M. Lin, H. Cao, V. Zheng, K. C.-C. Chang, and S. Krishnaswamy, "Mobility profiling for user verification with anonymized location data," in Proceedings of the 24th International Conference on Artificial Intelligence. AAAI Press, 2015, pp. 960–966.

[78] A. Gellert and L. Vintan, "Person movement prediction using hidden Markov models," Studies in Informatics and Control, vol. 15, no. 1, pp. 17–30, 2006.

[79] C. Koehler, N. Banovic, I. Oakley, J. Mankoff, and A. K. Dey, "Indoor-ALPS: An adaptive indoor location prediction system," in Proc. ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2014, pp. 171–181.

[80] W. Eberle and L. Holder, "Anomaly detection in data represented as graphs," Intelligent Data Analysis: An International Journal, vol. 11, no. 6, pp. 663–689, 2007.

[81] M. Davis, W. Liu, P. Miller, and G. Redpath, "Detecting anomalies in graphs with numeric labels," in Proc. 29th ACM Conf. on Information and Knowledge Management, 2011, pp. 1197–1202.

[82] W. Eberle, L. Holder, and J. Graves, "Detecting employee leaks using badge and network IP traffic," in IEEE Symposium on Visual Analytics Science and Technology, October 2009, unpublished.

[83] C. Liu, H. Xiong, Y. Ge, W. Geng, and M. Perkins, "A stochastic model for context-aware anomaly detection in indoor location traces," in Proc. IEEE 12th International Conference on Data Mining, 2012, pp. 449–458.

[84] R. P. Biuk-Aghai, Y.-W. Si, S. Fong, and P.-F. Yan, "Individual movement behaviour in secure physical environments: Modeling and detection of suspicious activity," in Behavior Computing, L. Cao and P. S. Yu, Eds. Springer, 2012, pp. 241–253.

[85] M. J. Hoesl, "Integrated physical access control and information technology security," U.S. Patent No. 6641090 B2, granted on June 17, 2014.

[86] H. Khurana, V. Guralnik, and R. Shanley, "System and method for insider threat detection," U.S. Patent No. 8793790 B2, granted on July 29, 2014.

[87] "AlertEnterprise Guardian Physical," https://www.alertenterprise.com/products-EnterpriseGuardianPIAMandBadging.php, accessed: 2019-03-20.

[88] "IDentitycard PremiSys Security Management Dashboard," https://www.identicard.com/access-control/premisys-security-management-dashboard/, accessed: 2019-03-20.

[89] J. P. Boyer, K. Tan, and C. A. Gunter, "Privacy sensitive location information systems in smart buildings," in Proc. of the 3rd International Conference of Security in Pervasive Computing. Springer, 2006, pp. 149–164.

[90] AlliedUniversal Security Services, "Security tailgating (aka piggybacking)," Security, Resiliency and Technology Integration Forum, Tech. Rep. [Online]. Available: http://www.alliedbarton.com/Portals/0/SRC/WhitePapers/Security%20Tailgating%20-%20Best%20Practices%20in%20Access%20Control.pdf

[91] L. Fennelly and M. Perry, Physical Security: 150 Things You Should Know. Butterworth-Heinemann, 2017.

[92] C. Vinson, J. Hallenstein, R. L. Fisher, and S. D. Perusquia, "Review of physical security protection of utility substations and control centers," State of Florida Public Service Commission Office of Auditing and Performance Analysis, Tech. Rep. [Online]. Available: http://www.psc.state.fl.us/Files/PDF/Publications/Reports/General/Electricgas/Physical_Security_2014.pdf

[93] T. K. Ho, K. Matthews, L. O'Gorman, and H. Steck, "Public space behavior modeling with video and sensor analytics," Bell Labs Technical Journal, vol. 16, no. 4, pp. 203–217, March 2012.

[94] H. Liu, S. Chen, and N. Kubota, "Intelligent video systems and analytics: A survey," IEEE Transactions on Industrial Informatics, vol. 9, no. 3, pp. 1222–1233, Aug. 2013.

[95] K. F. MacDorman, H. Nobuta, S. Koizumi, and H. Ishiguro, "Memory-based attention control for activity recognition at a subway station," IEEE MultiMedia, vol. 14, no. 2, pp. 38–49, April 2007.

[96] R.-S. Hsiao, T.-X. Chen, C.-H. Kao, H.-P. Lin, and D.-B. Lin, "An intelligent access control system based on passive radio-frequency identification," Sensors and Materials, vol. 29, no. 4, pp. 355–362, 2017.

[97] J. Toledo-Castro, P. Caballero-Gil, N. Rodríguez-Pérez, I. Santos-González, and C. Hernández-Goya, "Beacon-based fuzzy indoor tracking at airports," in Multidisciplinary Digital Publishing Institute Proceedings, vol. 2, no. 19, 2018, pp. 1255:1–1255:9.

[98] "AnyPlace," http://anyplace.cs.ucy.ac.cy/, accessed: 2018-05-20.

[99] "infSOFT," https://www.infsoft.com/industries/airports/features, accessed: 2018-05-20.

[100] J. Xiong and K. Jamieson, "Arraytrack: A fine-grained indoor location system," in Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation, 2013, pp. 71–84.

[101] K. Conger, R. Fausset, and S. F. Kovaleski, "San Francisco bans facial recognition technology," The New York Times, May 14, 2019. [Online]. Available: https://www.nytimes.com/2019/05/14/us/facial-recognition-ban-san-francisco.html