July 2021

Human-centered Cybersecurity Research — Anthropological Findings from Two Longitudinal Studies

Anwesh Tuladhar
University of South Florida

Follow this and additional works at: https://digitalcommons.usf.edu/etd

Part of the Computer Sciences Commons, and the Social and Cultural Anthropology Commons

Scholar Commons Citation

This Dissertation is brought to you for free and open access by the USF Graduate Theses and Dissertations at Digital Commons @ University of South Florida. It has been accepted for inclusion in USF Tampa Graduate Theses and Dissertations by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact digitalcommons@usf.edu.
Human-centered Cybersecurity Research — Anthropological Findings from Two Longitudinal Studies

by

Anwesh Tuladhar

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Xinming Ou, Ph.D.
Adriana Iamnitchi, Ph.D.
Jarred Ligatti, Ph.D.
Yu Zhang, Ph.D.
Daniel Lende, Ph.D.

Date of Approval:
July 20, 2021

Keywords: Ethnography, Transportation Security, Secure Development, Situated Learning

Copyright © 2021, Anwesh Tuladhar
Dedication

To my family, who have believed in me more than I believe in myself.
Acknowledgments

I am extremely grateful to have Dr. Xinming Ou as my advisor who has guided me through my Ph.D. journey. He has always been patient, helpful, understanding, and insightful from the very beginning. His ability to show you the way without telling you where to go at times is uncanny.

I would like to thank Dr. Adriana Iamnitchi, Dr. Daniel Lende, Dr. Jay Ligatti, and Dr. Yu Zhang for serving on my committee and Dr. Nathan Fisk for chairing the defense. I am delighted to have had the opportunity to work with everyone on my committee. I would also like to thank Dr. Raj Rajagopalan and Dr. John McHugh for providing feedback on the SOUPS paper.

I would also like to acknowledge the impact of our research group, Argus Lab, throughout my time here at USF. I have learned a great deal, from research to life in general, from this group. Thank you Sathya, Fengguo, Yuping, Daniel (Xiaolong), Richard, Jaleel, Armin, Louis (Guojun), and Parvaneh for making everything more enjoyable. Thank you again to Fengguo for boosting my technical skills, resume (helped me get my first internship :D), and above all my confidence to take on the challenges of Ph.D.

A big thank you to all my friends — Love, Sulav, Rajeev, Upama, Ghulam Jilani, Ahmad, Hernan, J.B., (to name a few) for all the time spent together from coffee breaks to playing soccer to having delicious (sometimes unbearably spicy) food.

I also owe great gratitude to my partner in life, Sanju G.C., who has been there encouraging, motivating, and supporting me all along. Thank you for being the yin to my yang ✨
This work was supported by the U.S. National Science Foundation under Grant No. 1638301, 1801633, and 1622402.
Table of Contents

List of Figures ........................................................................................................ iv

Abstract ................................................................................................................. v

Chapter 1: Introduction .......................................................................................... 1
  1.1 Reading Guide ................................................................................................. 6

Chapter 2: Research Methodology ........................................................................ 7
  2.1 Participant Observation ................................................................................. 7
  2.2 Data Analysis .............................................................................................. 8
  2.3 Research Ethics ........................................................................................... 9

Chapter 3: Study 1 — Vehicular Transportation System ..................................... 10
  3.1 Background .................................................................................................. 10
  3.2 Research Team ............................................................................................ 12
  3.3 Fieldwork Setting ....................................................................................... 12
  3.4 Methodology .............................................................................................. 13
  3.5 Training ....................................................................................................... 13
    3.5.1 Transportation System Overview ....................................................... 15
  3.6 Security Evaluation ..................................................................................... 17
    3.6.1 Intersection ......................................................................................... 17
    3.6.2 Signal Cabinet ...................................................................................... 18
    3.6.3 Application Security ............................................................................ 20
    3.6.4 Network Security ................................................................................ 21
  3.7 Anthropological Observations and Findings ............................................. 22
    3.7.1 Challenges in Communication ............................................................ 22
    3.7.2 Need for Systematization of Knowledge ........................................... 23
  3.8 Systematization Framework ......................................................................... 24
    3.8.1 Approach .............................................................................................. 25
    3.8.2 Component View .................................................................................. 26
      3.8.2.1 Vehicle ......................................................................................... 27
      3.8.2.2 Intersection ................................................................................. 27
      3.8.2.3 Roadside Unit (RSU) ................................................................. 27
      3.8.2.4 Transportation Management Center (TMC) ............................. 27
      3.8.2.5 Cloud End-point ......................................................................... 28
    3.8.3 Technological View .............................................................................. 28
      3.8.3.1 Sensing ........................................................................................ 28
      3.8.3.2 Control ........................................................................................ 29
      3.8.3.3 Inference ....................................................................................... 29
      3.8.3.4 Applications ............................................................................... 29
      3.8.3.5 Communication ......................................................................... 30
3.8.4 Stakeholders Involved ................................................... 30

3.9 Systematization of Knowledge .................................................... 31
  3.9.1 Attacks on Sensing ..................................................... 32
    3.9.1.1 Types of Sensing Technologies ................................ 32
    3.9.1.2 Attack Categories ............................................. 33
    3.9.1.3 Impacts of Attacks on Sensing ................................. 34
    3.9.1.4 Possible Mitigations ........................................... 35
  3.9.2 Attacks on Control ..................................................... 36
    3.9.2.1 Types of Control Technologies................................. 36
    3.9.2.2 Attack Categories ............................................. 37
    3.9.2.3 Impacts of Attacks on Control................................. 39
    3.9.2.4 Possible Mitigations ........................................... 40
  3.9.3 Inference Technologies ................................................ 41
    3.9.3.1 Types of Inference Technologies ............................... 41
    3.9.3.2 Attack Categories ............................................. 42
    3.9.3.3 Impacts of Attack on Inference................................ 43
    3.9.3.4 Possible Mitigations ........................................... 44
  3.9.4 Applications ............................................................ 45
    3.9.4.1 Types of Applications......................................... 45
    3.9.4.2 Attack Categories and Impacts................................ 45
    3.9.4.3 Possible Mitigations ........................................... 47
  3.9.5 Communication Technologies ........................................... 48
    3.9.5.1 Types of Communication Technologies ......................... 48
    3.9.5.2 Attack Categories ............................................. 49
    3.9.5.3 Possible Mitigations ........................................... 51

3.10 Discussions ..................................................................... 52
  3.10.1 Emerging Threats ..................................................... 53

3.11 Related Work................................................................... 55

3.12 Conclusion...................................................................... 56

Chapter 4: Study 2 — Secure Software Development ......................................... 57
4.1 Background ...................................................................... 57
4.2 Fieldwork Setting ................................................................ 58
  4.2.1 The Development Team ................................................ 59
  4.2.2 Methodology ........................................................... 59
4.3 Software Development Processes and Challenges Facing Secure Development ..... 60
  4.3.1 Sprint Planning ........................................................ 60
  4.3.2 Development Workflow ............................................... 61
    4.3.2.1 Design ........................................................ 62
    4.3.2.2 Implementation ............................................... 63
    4.3.2.3 Continuous Integrations/Code Analysis......................... 64
    4.3.2.4 Developer Testing ............................................. 65
    4.3.2.5 Code Review .................................................. 65
    4.3.2.6 Post Development Testing ..................................... 66
4.3.3 Product Release ........................................................ 67
4.4 A Shift in Secure Development Practice .......................................... 67
  4.4.1 Little Impact at First................................................... 67
  4.4.2 Making Progress ........................................................ 69
    4.4.2.1 Security Scrum Poker ......................................... 70
List of Figures

Figure 3.1 Overview of the Transportation Ecosystem........................................ 15
Figure 3.2 Signal Cabinet ................................................................... 18
Figure 3.3 Systematization Approach........................................................ 26
Figure 3.4 False Data Injection Attack Against the I-SIG Inference System........... 43
Figure 4.1 The Learning Cycle .............................................................. 84
Abstract

Cybersecurity is a pressing issue. Researchers have proposed numerous security solutions over the years in order to combat security issues but it is still common to find known, well understood security issues in production environments. In this thesis, I seek to find the underlying reasons to why existing security solutions and best practices are not consistently applied and how to improve the utilization of secure best practices. To this end, I adopt the anthropological research method of long term participant observation and embed myself in real-world settings in order to understand the existence of security issues and the perception of security from a “native’s point of view”.

First, I conducted an in-depth, six-month embedding in a traffic management center (TMC) of a mid-size city in the U.S. to gain first-hand knowledge of the cyber-security issues in vehicular transportation systems, which is a multi-disciplinary field with combined contributions from civil & transportation engineering, traffic engineering, electrical engineering, communications engineering, and computer science. We identify the existence of silos of different disciplines, making it difficult to understand and communicate the security impact one can have in the context of the whole transportation ecosystem. Based on our observations, we present a systematization framework which identifies key components, technologies, and stakeholders in the whole ecosystem which forms the basis for understanding attack scenarios, their impacts and mitigations. This methodology helps to put security analysis into the context of the transportation ecosystem and provides a common platform for communication to help break down the silos existing both in research and in practice.
Next, I conducted an eight-month long ethnographic study of a software development company to explore if and how a development team adopts security practices into the development lifecycle. This effort involved working as a software engineer and participating in all development activities as a new hire would. During the fieldwork, I observed a positive shift in the development team’s practice regarding secure development. Our analysis of data indicates that the shift can be attributed to enabling all software engineers to see how security knowledge could be applied to the specific software products they worked on. I also observed that by working with other developers to apply security knowledge under the concrete context where the software products were built, developers who possessed security expertise and wanted to push for more secure development practices (security advocates) were more effective in achieving this goal. Our data analysis point to an interactive learning process where software engineers acquire knowledge, apply it in practice, and contribute to the team, leading to the creation of a set of preferred practices which is often collectively referred to as “company culture.” This learning process can be understood through the lens of the situated learning framework, where it is recognized that knowledge transfer happens within a community of practice, and applying the knowledge is the key in individuals (software engineers) acquiring it and the community (the company) embodying such knowledge in its practice. Our data show that enabling a situated learning environment for security gives rise to security-aware software engineers. I discuss the roles of management and security advocates in driving the learning process to start a security culture in a software company.
Chapter 1: Introduction

“If you think technology can solve your security problems, then you don’t understand the problems and you don’t understand the technology.” — Secrets and Lies by Bruce Schneier [87]

Cybersecurity is a not solely a technical problem but a socio-technical problem where human factors play a crucial role in the security of the system. Over the years, the focus on human factors in security has seen the rise of the field of usable security which has yielded ample improvements in security of software products as security designs have been catered more towards the end-users. With tremendous growth in the number of software products and, with it, security vulnerabilities, the usable security community also expanded their focus from end-users to software developers. As not all software developers were seen to have adequate security expertise [118], onus was placed on improving security tools such as APIs of programming libraries, development tools, and vulnerability discovery tools [2, 38, 110, 111, 118]. Even with the development of numerous security tools to assist software developers in writing secure code, the same (or similar) security bugs and vulnerabilities have kept recurring in production environments even when known secure solutions are readily available. In this thesis, I seek to understand why known security issues keep recurring and existing security knowledge and solutions are not applied in practice, and how to dampen this trend.
Early work from 2002 by Gonzalez and Sawicka [35] postulate the necessity to collaborate with companies and organizations in order to better understand the role of human factors in security of systems. In order to answer the questions above, we collaborate with organizations in two distinct problem domains:

1. Vehicular transportation system: We collaborate with a transportation management center (TMC), which is an operational center for the city-wide traffic management.

2. Software development: We collaborate with a development team of a software company.

Sundaramurthy et al. [94, 95] were the first to adopt the anthropological research method of long-term participant observation to study security operations centers (SOCs). This study provided deep insights into the problem of burnout in SOCs [94] and proposed technical solutions to improve operational efficiency [95]. Inspired by their success, we also adopt long-term participant observation to conduct both our studies. Chapter 2 describes the details of data collection using participant observation and subsequent analysis using general inductive approach.

First, in Chapter 3, I describe my six-month long embedding at a TMC of a mid-sized city in the U.S. I first describe the field work setting and my first-hand experience of the daily activities conducted by a TMC staff. This is followed by the non-invasive security evaluation of the technologies utilized in the TMC. This evaluation was restrictive due to the potential consequences of working in a live system overlooking over 550 traffic signals. Nonetheless, it gave me the opportunity to identify existing security issues, verify the threats identified in the literature, and understand the TMC operators perspective towards security. On analysis of the fieldnotes, we make a key observation that the existence of silos of different engineering disciplines makes it difficult to understand and communicate security issues and their impacts in the context of the whole trans-
portation ecosystem. We also find that the research efforts in this domain are often fragmented, targeting only specific segments of the system, and hence fail to capture the overarching context of the whole transportation system. This failure to communicate security issues and their impacts on the transportation system contributes to the failure to utilize the available secure solutions. Based on these observations, we find that a coherent framework capturing the key components, technologies, and stakeholders will benefit the management and operation of such systems as well as the design process of future systems and system components.

The contributions of this study at the TMC are as follows:

1. In Section 3.7, I present the observations from the embedding in the TMC which led to a two-tiered framework to study and evaluate the security posture of the transportation system as a whole, presented in Section 3.8

2. In Section 3.9, this framework is utilized to organize existing transportation security knowledge found in literature, to understand existing threats, attack techniques, mitigations, and the stakeholders responsible for prevention and mitigation. We use this process to identify common causes for the discovered vulnerabilities and evaluate the suggested countermeasures where possible.

   In Chapter 4, I present the second study, which is focused on the implementation of secure development practices in a software development company. I joined the development team as a new hire and fully participant in all development activities such as scrum meetings, design/implementation discussions, code review, testing, documentation, and so on. I was fortunate to join the company two months after the management decision to implement secure software development lifecycle (S-
SDLC) which provided us the opportune moment to conduct research into whether and how such a push for security may result in concrete positive changes in the development processes.

In Section 4.3 and 4.4, I present the observations of eight months of fieldwork including the evolution of the S-SDLC, the role of a security expert within the development team, identification of challenges faced, and how they were overcome in time to lead to a positive shift in the development team’s practice regarding secure development. Our analysis of the fieldnotes reveal that an interactive learning process exists within the development team where software engineers acquire knowledge, apply it in practice, and contribute to the team, leading to the establishment of a set of preferred practices which is often collectively referred to as the “team culture.” This learning cycle is central to the successful establishment of secure preferred practices and, hence, start a security culture within the team. The learning process within the development team is described in Section 4.5 and the key enablers of the positive shift in secure development is described in Section 4.6. We also provide recommendations to software development companies based on our findings in Section 4.8.

The main contributions of this study of a software development team are as follows:

1. We identify an important factor in establishing secure development practices in a software company — the role of situated learning [62] that forms an integral part of software engineers’ work. Rather than assuming structured processes on their own can solve security problems, we examine the context for learning about security within the development environment and analyze how it shapes the workflows followed by individual software engineers. Our analysis of data shows that what was driving the positive shift in the development team’s security awareness can be explained by the learning dynamics existent therein, in particular the software engineers being able to identify the applicability of the security knowledge within the
context of the everyday work they perform. The situated learning dynamics could drive the
team into a set of agreed-upon knowledge and the associated practices, becoming the “pre-
ferred practices” for dealing with specific security concerns. We hence identify a way to start
a secure coding culture in a development team.

2. Our data also indicate that the presence of a security expert working within the development
team is instrumental in driving the situated learning cycle for security. It appears that when
such security experts are part of the development team, and their actions foster the learning
process, the adoption of secure coding practices become more readily accepted by the team.
In particular, we find it important that security knowledge be offered within the context of
the team’s concrete work.

Both of these longitudinal studies provided me the opportunity to observe real-world oper-
ations and interact with professionals in the field. By studying the problem within the context of
where it occurs, I was able to extract valuable insights that are otherwise difficult to obtain through
snapshots-in-time studies such as controlled experiments or through interviews/surveys which rely
on self-reported data. This approach also benefits from the fact that it allows us to study peo-
ple as a collective, as opposed to individually, which helps us uncover different social dynamics
as well. Hence, I believe long-term participant observation through collaborations with compa-
nies/organizations is a valuable tool to better understand the role of human-factors in security of
systems.
1.1 Reading Guide

The rest of the thesis is organized as follows: Chapter 2 describes the anthropological research method utilized to collect data through participant observation and data analysis using general inductive approach. Chapter 3 describes the fieldwork at a transportation management center and details the systematization framework proposed. Chapter 4 describes the fieldwork at a software development company and explores the implications of situated learning in establishing a security culture in a software development team.
Chapter 2: Research Methodology

This thesis describes the findings from long-term studies at two different locations, namely, a Transportation Management Center (TMC) and a Software development company. The primary research method employed for both of these studies was long-term participant observation — a qualitative research method developed by anthropologists and sociologists to study human behavior in their natural setting in order to gain a better understanding of local practices and their culture. This chapter describes participant observation, data collection, and data analysis within the context of our research. We also discuss the limitations of this approach compared to other approaches. The specifics of each fieldwork are discussed in the corresponding chapters.

2.1 Participant Observation

Participant observation is a qualitative research method in which the researcher spends a long period of time (typically over a year) as a member of the community being studied. Being embedded within the community allows the researcher to participate and observe day-to-day activities of the people within the community. Such constant involvement within the community allows the researcher to obtain strong understanding of the local practices from a native perspective.

The participant observer in this research was a computer science Ph.D. student with background in security and ample industry experience. In preparation for this research, the computer science Ph.D. student was enrolled in applied research methods class from the Department of Anthro-
pology and additionally underwent extended training in participant observation under the guidance of the anthropologist in the research team.

2.2 Data Analysis

The participant observer (PO) maintained descriptive field notes on daily work activities and interactions with other employees in the field. These notes included “as is” descriptions and avoided personal opinions or commentary. The raw field note data was analyzed using the general inductive approach [97], augmented by specific techniques for qualitative data analysis [10]. After a period of data collection, the research team met weekly to discuss and reflect on the observations made so far. The team went over the events of the past week and discussed the events concerning local practices, security practices, and the relevant interactions. The PO tagged the raw field data based on the patterns and themes that emerged during these discussions. Any unanswered questions and/or missing information during these discussions then guided the future observations in the field. The weekly iteration of data collections followed by in-depth discussions led to the refinement of the emerging categories used for coding. Then, as broader themes were conceptualized, the PO started to write analytic notes summarizing each theme and documenting ideas and analysis of each along with supporting data from relevant sources in addition to the raw fieldwork data. After the end of the fieldwork, the research team continued further analysis of data through extensive discussions to draw out the major implications of the observations.
2.3 Research Ethics

The study was reviewed and approved by the Institutional Review Board (IRB). Researchers explained the study goals to participants and obtained verbal informed consent from participants. Field notes were anonymized, as well as discussions during weekly research meetings.
3.1 Background

Vehicle transportation infrastructure is undergoing fundamental changes with the converging of traditional closed systems and the Internet-driven new technologies to improve efficiency, mobility, safety, comfort, and convenience. It is a multi-disciplinary field with combined contributions from the civil & transportation engineering, traffic engineering, electrical engineering, communications engineering, and computer science and engineering. With rapid advancements in technology and improved connectivity, introduction of connected and autonomous vehicles is leading the technology-driven revolution of the transportation systems. The increased connectivity within and between vehicles, and between vehicles and the transportation infrastructure is driving the transformation at an accelerated pace. These advancements have made possible vast improvements and created an enormous market for rapid deployment of feature-rich vehicles and infrastructure equipment. Like in all domains, the dramatic increase in the use of technology and connectivity also opens up new threats from cyber attacks.

A significant challenge in understanding the cybersecurity risk in a system like vehicle transportation is that it requires the understanding of an ecosystem consisting of multiple non-trivial physical systems as well as the various stake holders involved in their design and operation. A vehicle is made by manufacturers and driven by humans (and/or computer programs) on the road, whose road-side infrastructures are built and maintained by public entities such as municipal and
state governments. These days both cars and infrastructures can communicate with third-party vendors in the cloud, providing/receiving information that impact their operations. With the influx of technology in every component and increase in the connectivity between them, the influence of each component on the other is greater than ever before. The dependencies thus created means that security vulnerabilities are no longer isolated to a particular component, thus making it much harder to contemplate without understanding the whole transportation ecosystem. It is not surprising that security risks in such a complex and inter-connected system can be both numerous and nuanced. Existing work in vehicle transportation cybersecurity has tended to focus on parts of the overall ecosystem, e.g., vehicles, traffic lights, etc. While (not surprisingly) many attack avenues were discovered, it is not often clear why these problems are there in the first place (beyond blaming users or developers), how to prevent them from being introduced, and how to mitigate them through cost-effective methods if elimination is not a practical option. To answer these questions a holistic framework to systematize cybersecurity issues in the vehicle transportation ecosystem is beneficial. With such a framework one can have a better understanding of the potential vulnerabilities when designing a system, have a quicker grasp of the risk when operating on the systems, and have a more meaningful perspectives on how to mitigate them in reality.

This chapter describes the six month long embedding at a transportation management center (TMC) of a mid-size city in the U.S. This effort provided first-hand knowledge of the inner workings of the vehicle transportation ecosystem through direct access to a real-world TMC and interactions with people from multiple engineering disciplines including transportation, traffic engineering, computer and communications, and others. This chapter starts by describing the specifics of the fieldwork setup before moving on to the observations at the TMC. The following

---

1This chapter was published as a technical report [102]
section looks at the cyber-security risks prevalent in the vehicular transportation system. Finally, we describe and discuss our systematization framework identifying the key components, technologies, and stakeholders in the whole ecosystem and forming the basis of understanding attack scenarios and their mitigations.

3.2 Research Team

The research team for this work comprised of me, the participant observer (PO), a computer science professor specializing in systems security, specialists in civil and transportation engineering including a Ph.D. student, a post doctoral researcher, and a professor in civil engineering, and a professor in anthropology.

3.3 Fieldwork Setting

A TMC is typically operated by local and state authorities and covers a specific geographic area based on its jurisdiction. A TMC provides a direct lens for understanding the domain knowledge required for security evaluation of the entire transportation ecosystem. We conducted in-depth fieldwork with researchers embedded in the TMC, run by a medium-size U.S. city operating over 550 traffic signals with the goal of optimizing the travel time throughout the city. A combination of fiber optics cables, twisted copper wire, and wireless communication connected hundreds of traffic cabinets to the central TMC. The TMC housed dedicated servers to run software applications used by traffic engineers to monitor real-time traffic and operate signal cabinets. It used an in-house test cabinet to train new employees and test new signal timings.
The PO spent a few days every week (roughly five hours each visit) depending on the TMC schedule, working primarily with seven members of the team, for a duration of six months, observing daily operational activities and participating where possible. The PO first went through training on basic traffic engineering and various tools used during daily operations. The PO then participated in daily operations which included collecting travel time data, creating/updating signal timing, operation of reversible lanes, monitoring and analyzing signal timing reports, and incident response (failures, accidents, and client complaints).

3.4 Methodology

The fieldwork data was analyzed using the general inductive approach described in Section 2. The descriptive field notes was used to record the daily work, the systems used in various tasks, and interactions with the engineers at the TMC. Given that one initial goal of the embedding was to understand the system, the PO also maintained separate notes focused on the technical information and dimensions related to vehicle transportation system security. Both the anthropological and the technical field notes were subject to the analysis and helped develop plans for subsequent research visits. This fieldwork was part of a larger interdisciplinary project and was reviewed and approved by the Institutional Review Board (IRB). The findings reported here are limited to the knowledge gained by the PO through access to the physical artifacts and the interactions with the professionals at the TMC.

3.5 Training

The first few weeks of the fieldwork was training on the traffic engineering fundamentals focused on performing signal timing calculations. The traffic engineers on duty trained the PO on
the fundamentals of signal timing calculations and demonstrated the systems and tools that they used on a daily basis to maintain efficient signal timing throughout the city. The PO was then assigned to assist in re-timing the signal plan for a major corridor along four intersections and carried out the following tasks:

1. Collected travel time data during the morning peak hour (9:30 AM — 10:30 AM) from the field along the corridor.

2. Calculated appropriate parameters for the signal controller to use such as clearance times for pedestrians to provide maximum safety margins, as well as the offset times required for coordinated signal timings along the corridor.

3. Programmed the new signal timing in the test signal controller. Any new signal timing was first programmed in the test cabinet and left to run for at least 24 hours before reprogramming the on-field signal cabinet.

During this process, the PO gained first hand experience with the systems in place at the intersection for vehicle detection, the internals of the traffic cabinet including signal controller, malfunction management unit (MMU), network devices to connect back to the TMC, and the applications used by the traffic engineers to monitor and configure the signal controllers. All of the intersections are connected to the central TMC by either twisted copper pair wires, fiber-optics cables, or wireless connections and forms the air-gapped communication network. Based on the type of connection, the city operates two distinct systems to monitor and configure the traffic signals throughout the city: (1) Metropolitan Traffic Control System (MTCS) is an older intersection controller application which runs on the Microsoft DOS operating system and is deployed to manage the
traffic signals at intersections which are only connected via twisted copper pair wires. (2) Econolite Centracs [25] which is an Advanced Traffic Management System (ATMS) solution and is used to manage the traffic intersections connected via fiber optics or wireless links. Both of these applications were deployed in in-house servers. Dedicated workstations are also assigned to use these applications. These workstations are separate from the internet-connected personal workstations to maintain the air-gapped transportation network.

3.5.1 Transportation System Overview

Based on our observations, we visualize the transportation system as an ecosystem of various different technologies as in Figure 3.1. The main component is the millions of vehicles on the road, including specialized public transit and emergency response vehicles. Vehicles dominate the transportation ecosystem in terms of both the number and the technological advancement and research. Vehicles have evolved from traditional mechanical systems to complex computing-mechanical systems consisting of hundreds of sensors, dozens of control devices from electronic control units, or...
ECUs, to infotainment systems, and now to connected and automated vehicles (CAVs) capable of communicating amongst each other and with the infrastructure and cloud services that provides large-scale data analytics and inference capabilities. Dozens of vendors contribute to the manufacturing and development of vehicles, driven by the demands from end-user customers.

The vehicles, end-users, and the transportation infrastructure interface with each other mainly through the devices on the road intersections such as vehicle detectors, traffic lights, and dynamic message signs (DMS). This interface capability is extended by the use of mobile devices and various applications and further improved by the introduction of connected vehicles. The input interface for the transportation system are the sensing technologies, feeding data into the roadside component. Based on these inputs, the roadside equipment in turn control the output interfaces.

To maintain efficient and safe traffic flow throughout, the roadside equipment are connected to each other and to the TMC through various communication channels (fiber optics, wireless, etc.). Daily monitoring and efficient operation of the transportation system is the responsibility of the TMC operators. TMC may host several servers supporting the operations of the roadside units and various applications utilized by the operators to perform their daily tasks. They might also be connected to other third-party cloud services through virtual private networks (VPNs).

Transportation system is a critical infrastructure of the modern society with practically everyone utilizing its services. Hence, regulators at various levels enforce standards and rules to ensure safe and secure functioning of this system. This includes standards and rules imposed on the end-users (traffic laws), operators (procedures to follow) and vendors (types of technology to use and standards to maintain).
3.6 Security Evaluation

After the completion of training, we conducted a non-invasive security analyses, with a particular focus on how the in-house testbed worked in the context of the overall management of the transportation system. This gave us the opportunity to verify whether threats identified in the broader literature were present in a real-world setting. However, operating on a live system with potential for severe consequences restricted our experimentation but at the same time offered opportunities to understand the security posture of a real system as well as understand the operators view towards security.

3.6.1 Intersection

During our fieldwork we learned that incorrect sensing data was a concern for the TMC engineers. While the concerns were mostly for malfunction, deliberate attacks would worsen the situation.

One existing problematic scenario is pedestrian button being stuck in the ON position, equivalent to receiving fake sensing data. The maximum impact occurs at intersections between a major corridor and a rarely used side-street activated only on vehicle/pedestrian detection. This causes unnecessary red lights on the main corridor while the side street is always serviced even when no vehicles/pedestrians are present. The impact is harder to observe as operators have to analyze travel time data to observe sub-optimal traffic conditions.

In another scenario we found in the TMC, a public complaint reported that the left turn green phase was being skipped even when vehicles were present in the lane during evening time windows. It turned out that the skipped phase was on the east-west corridor of an intersection using
video-based detection, which suffered from a sun-glare at evening time causing missed detections. Yan et al. [122] demonstrated missed detections in millimeter-ware (MMW) radars (used in Tesla cars) and video-based detection. While this work is focused on sensors on vehicles, similar attacks can be effective for sensors in intersections as well.

3.6.2 Signal Cabinet

Figure 3.2 shows the test signal cabinet. These are present at the side of the roads with signalized intersections. Although the signal cabinets are locked, they utilize standardized locks with Corbin style #2 keys which are readily available for purchase. We also found that although signal controllers provide password protection and access control capabilities, they are not utilized in practice. The reasoning behind both of these is the fact different field operators need to access the cabinets which makes physical key and credential management cumbersome for daily operations. Ghena et al. [32] report that even when authentication is enabled, they utilize the default settings.
We also found that, with physical access, the intersection could be sent to all-way flashing red state trivially by: (i) removing a load switch, or (ii) updating the signal timing on the controller to violate safety conditions such as minimum yellow time, minimum all red time, and no conflicting greens. In this case, the Malfunction Management Unit (MMU)/Cabinet Monitor Unit (CMU) sends the intersection to all-way flashing red to maintain the safety property. MMU identifies conflict status through hard-wired programming card [72] or a serial memory key (datakey) [49]. However, we found that with the knowledge of how the permissive states are defined in MMU/CMU, availability of modifiable hard-wire programming card for training purposes, and physical access to the unit, adversaries can modify or replace it to allow unsafe conflicting green state. This, combined with signal timing manipulation can create catastrophic conditions.

During the fieldwork, with access to the network, we were able to gain root access on Econolite signal controllers as it exposed ssh service setup with default user name and password. Using this we could trigger system reboots, modify/remove essential programs, or carry out denial-of-service (DoS) attacks on the controller. Although not used by the TMC, we were informed that some of the cabinets are equipped with WiFi dongles which allows the field operators to get on the network to access to the controllers without opening cabinet. This would even allow remote exploits of the same kind.

Although MMU/CMU maintains safe conditions and requires physical tampering for complete bypass, unsafe conditions can still be created by pushing the signal timing to the edge of conflict condition. During our fieldwork we were able to program a signal timing with one second flickering green light without triggering the MMU. This is possible since the MMU only checks for minimum all red and yellow phases and has no restriction for the green phase. This, at the very
least, can lead to confusion and annoyance to the public at intersections with low speed limit but may lead to fatal accidents when the speed limits are higher and the vehicles are unable to stop in time. This result shows that attackers can drive intersections into unsafe signal conditions, even with MMU/CMU present as the final line of defense.

During our embedding, we were able to disrupt the signal timing synchronization on a test signal controller by short-circuiting two pins on the GPS, thereby sending a false signal to reset the controller’s local clock to the programmed reference time without triggering any faults or alerts on the system. This can greatly impact the traffic flow. While in our experiments we did have physical access to the RSU, with the rise of network-connected field devices, remote access to the devices from close proximity, or even through the network is possible. With such access, adversaries can miscalibrate sensors, install malicious firmware, or gain access to other devices that trust it.

3.6.3 Application Security

The Metropolitan Traffic Control System (MTCS) is an older intersection controller application running on Microsoft DOS which does not support any modern security features and does not even require basic authentication. This is also inadequate from an operational security point of view, as the weak monitoring system means that attacks can go unnoticed. ASC3 (kdClient, utility, configurator) is a set of tools to provide direct connection to a ASC/3 signal controller for configuring pin assignments and signal timing [24]. Any signal controller on the network can be connected to based on IP address without authenticating.

More modern applications do provide some security mechanisms such as authentication and additional features useful for operational security. Centracs is used for configuration and monitoring
of Econolite controllers throughout the network [25]. For operational security, it provides device log management, generates alerts for fault conditions, and can generate reports for further analysis if required. It also tracks configuration updates along with user information. DYNAC’s DynGate [51] is for operation of reversible lanes. It provides interfaces to inference system for scanning the roadway for vehicles, and control technology for operation of lane barriers/gates and corresponding signs. This is a safety critical application with potential to prevent (or create with malicious intent) devastating conditions.

3.6.4 Network Security

We found that although the transportation communication network is air-gapped from direct internet access, external access to the backend servers is provided to the third-party vendors for management and maintenance of the application servers.

We also found that third-party applications such as connected signals app [19], which displays real-time traffic light predictions at each intersection, receive data directly from the TMC. This was accomplished via a Raspberry Pi [130] device directly connected to the transportation network in order to transmit the signal timing. We were assured that no incoming traffic was accepted from that device. Such cases undermine the air-gapped assumption of the transportation communication network and changes the threat landscape.

Other monitoring applications include BlueTOAD [99] which utilizes data collected from traveller’s bluetooth-enabled devices at various intersections. Waze Traffic View [114] and Waycare [113] utilize crowd sourced and other open data sources. Although these applications have less
security implications, adversarial access can misinform operators which can hide issues or lead to incorrect decision making.

3.7 Anthropological Observations and Findings

Section 3.6 described security issues prevalent in a real-world transportation management system. This helped us identify cyber-security threats as well as verify some of the threats reported in literature. During this process, the PO also maintained descriptive field-notes which gave us insights into different perceptions of security, threats, and mitigations within the professionals working in the TMC. Next, we discuss our experiences and findings.

3.7.1 Challenges in Communication

Perhaps the most important experience during this fieldwork is that efficient communication presented a major challenge during the fieldwork. Field operators and computer/civil/traffic engineers were all working in the same environment. The perspective of the transportation infrastructure was different for each individual. Take, for example, a discussion on signal cabinets and security. The field operator focused on how a person can access the signal cabinet; the traffic engineer focused on the potential undesired conditions that may arise due to changes in the signal timings; the computer engineer focused on the signal controller, software running on it, and network connection; and the civil engineer focused on the planning aspect and the impact it may have on the entire road network.

In this case, the field operator asked “can attackers physically access the signal cabinets?” The traffic engineer asked “what damage can attackers cause if given access to signal cabinets?” These and others are not direct quotes.

22
computer engineer asked “can attackers gain access to devices in signal cabinets?” And the civil engineer asked “what can be the impact of such malicious actions?” All these points of view are important to understand in order to perform a security risk analysis of the transportation system.

Throughout the fieldwork, we interacted with people from multiple disciplines including transportation, traffic engineering, computer and communications engineering. Each of these fields have a unique role to play in the design, implementation, and operation of the transportation system. We observe the existence of silos of each discipline, making it difficult to understand and communicate the security impact one can have in the context of the whole transportation ecosystem. This echoes what we find in the relevant research literature, where in many cases security issues identified stem from assumptions made about other aspects of the ecosystem, regardless of whether such assumptions can hold or not.

3.7.2 Need for Systematization of Knowledge

When analysing the field notes, we found that cybersecurity-related discussions were better facilitated when the relevant technologies were considered in an abstract way. At times, discussions focused solely on the purpose of the relevant technology in the current transportation system managed by the TMC, often getting into disciplinary minutiae that did not promote interdisciplinary understanding of security concerns. In contrast, more general discussions of problems facilitated risk analysis because people could draw on and apply knowledge about cyber attacks in other systems to their own. For example, when discussing cyber attacks on wireless in-pavement vehicle detectors, the initial response was that those were not used here and hence was not of concern. On further discussion, abstracting away the wireless in-pavement vehicle detectors to just “sensors” allowed the discussion to proceed leading to the understanding that a “sensor” (any sensor) compromised by
any means can lead to missed detections or false detections. This then led to risk assessment in the sensors deployed in this TMC.

On further reflection, we realized that the abstraction of technology not only helped in communicating security issues to experts of different disciplines, but also could aid security analysis. The abstracted technology view helped to drive the discussion of cybersecurity issues in vehicle transportation system. But then people involved in the TMC needed to connect the abstract view of specifics, in particular how the potential effects of compromised technologies could affect the specific transportation ecosystem managed by the TMC. Relating contextual information to abstract issues mattered in whether people acted or not to address security concerns. We believe that systematizing security issues that may arise will not only benefit the management and operation of such systems, but also the design process of future systems and system components which are undergoing rapid technological advancement.

3.8 Systematization Framework

Based on our experiences during the fieldwork, we find that systematizing security knowledge about vehicle transportation system can benefit from a two-tiered structure, the first tier describing the overall transportation ecosystem, and the second tier describing the various technologies involved. We find that applying this two-tiered framework can help communicate security issues to different disciplines and understand existing threats, attack techniques, mitigations, and stakeholder responsibilities. It could also benefit systematic risk/security analysis in existing systems or the design of future systems.
3.8.1 Approach

The transportation system is a technological ecosystem: a complex interconnected network of multiple components interacting with each other, utilizing various types of independent and interdependent technologies which are influenced by different stakeholders at various stages of development. We propose a systematization approach to provide a holistic context for the transportation ecosystem. The holistic view makes it easier to capture the interactions between the elements of the ecosystem and provides a coherent framework to evaluate the security properties of the ecosystem.

Our systematization utilizes a two-tier view of the transportation ecosystem to facilitate discussion and understanding. We call the first tier component view, whereby we categorize the transportation ecosystem into components, providing the logical separation of the various aspects of the ecosystem. This is the logical dimension of the ecosystem. The second tier is called technological view, whereby we identify the key classes of technologies that enable the functionality of these components. This is the functional dimension of the ecosystem. To aid security analysis, we identify key security issues faced by these classes of technologies, common mitigation approaches used, and the stakeholders involved. This approach is illustrated in Fig. 3.3. We have presented this systematization approach to a different state-level TMC that operated in the same city, as well as to a transportation research institute where people from diverse domains attended. In each of these experiences, we found that the systematization was well received and helped to open up discussion of security issues from people in multiple disciplines.

We want to emphasize that both these dimensions are important in our systematization. The logical dimension provides the context in which a component resides in the overall transportation ecosystem, thus the same type of security issues will have varying implications based on differing
### System Description

<table>
<thead>
<tr>
<th>Component View</th>
<th>Technological View</th>
<th>Attack Categories</th>
<th>Mitigations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicles</td>
<td>Sensing</td>
<td>Sensing vulnerabilities</td>
<td>Patching</td>
</tr>
<tr>
<td>Intersection equipment</td>
<td>Control</td>
<td>Service vulnerabilities</td>
<td>Device replacement</td>
</tr>
<tr>
<td>Roadside equipment (RSU)</td>
<td>Inference</td>
<td>Weak/no authentication</td>
<td>Fundamental Changes</td>
</tr>
<tr>
<td>Transportation Management Center (TMC)</td>
<td>Applications</td>
<td>Weak/no encryption</td>
<td>Regulations</td>
</tr>
<tr>
<td>Cloud-end points</td>
<td>Communication</td>
<td>Programming vulnerabilities</td>
<td></td>
</tr>
</tbody>
</table>

### Security Analysis

**Stakeholders**

- End-users
- Vendors
- Operators
- Regulators

Figure 3.3: Systematization Approach. *System description*: The transportation ecosystem is separated into components which are described using the abstraction of enabling technologies in order to capture a holistic point-of-view without having to know the implementation details. *Security analysis*: Identified common attack categories and possible mitigation strategies. *Stakeholders*: The parties involved in maintaining a safe and secure transportation ecosystem.

 contexts. The functional dimension captures the similarities among the possible attacks on the same type of technologies, despite the fact that they could be used under different contexts and thus with different security impacts. We believe this two-tier approach to look at security issues will allow one to both generalize those problems based on their similarities, and at the same time not lose the intricacy and diversity of possible attacks when it comes to the interactions among components in the overall system.

### 3.8.2 Component View

Based on the roles they play, the transportation ecosystem can be divided into the following components.
3.8.2.1 Vehicle

Vehicles, traditionally the “users” of the transportation infrastructure, now form an integral part of the transportation ecosystem. The connected infrastructure relies on trajectory data from vehicles which, in turn, rely on the infrastructure for safety advisory and navigation.

3.8.2.2 Intersection

Transportation infrastructure and vehicles/pedestrians exchange information at the intersections and hence intersections serve as an input/output interface. Infrastructure receives input from vehicles/pedestrians using detection systems, vehicle-to-infrastructure (V2I) communication, video surveillance, toll gantries, etc. Information is conveyed back to vehicles/pedestrians using traffic lights, dynamic message signs (DMS), infrastructure-to-vehicle communication, and mobile applications.

3.8.2.3 Roadside Unit (RSU)

RSU hosts devices responsible for safe and efficient operation of the intersection. These include signal controllers that manage the signal timings, Malfunction Management Unit (MMU) that ensures safety conditions, and network equipment for communicating with other intersections and the TMC.

3.8.2.4 Transportation Management Center (TMC)

TMC is a regional hub that serves as the mission control for urban transportation and highway networks. TMC operators collect real-time data and combine with other operational and control data in order to monitor roadways, proactively optimize traffic conditions, provide inci-
dent management, disseminate traveler information, and coordinate with other authorities for daily traffic, special events, accidents and emergencies.

3.8.2.5 Cloud End-point

Modern systems require heavy computational capabilities and rely on cloud-based infrastructure to host them. Existing usage of such services include vehicle diagnostics, real-time navigation, fleet management, and data-driven traffic monitoring services, with more emerging.

3.8.3 Technological View

Each component relies upon one or more technologies to function. In many cases, similar functionality is required across different components. Hence, based on the function provided, we group them into different classes of technologies: sensing, control, inference, application, and communication. Examining the transportation ecosystem based on these abstractions of technologies provides an opportunity to understand common attack patterns across components. This provides the basis for cybersecurity discussions amongst experts from different domains, that would otherwise be inhibited by technical details involving the differing domains. In addition, contextualization of those technologies within the transportation ecosystem aids in identifying attack goals, attack paths, and impacts that would otherwise be overlooked. An example of this is further discussed in Section 3.9.3.2. We describe each of these technologies below.

3.8.3.1 Sensing

The class of technologies used to detect the surrounding environment are grouped under this category. This also includes communication-based sensing such as vehicle-to-infrastructure (V2I)
devices. Sensing technologies facilitates translation of physical world inputs to electrical/digital signals. Sensing inputs are utilized in various components of the transportation system to improve efficiency and/or safety. For example, vehicles have hundreds of built-in sensors to facilitate features such as collision detection, automatic lane keeping, blind-spot detection and more. On the road, sensors are used for vehicle detection, toll enforcement, monitoring, and others.

3.8.3.2 Control

The class of technologies which facilitates the control functionality directly through operation of actuators or indirectly by informing end-users to take certain actions. These technologies usually consists of small computing devices. For example, electronic control units (ECU) in vehicles and signal controllers in the RSU.

3.8.3.3 Inference

The class of technologies used to extract insights from data collected through both sensing and communication. Increased use of sensor technologies and connectivity gives rise to vast amounts of data. Making use of these data points presents a huge challenge on its own. New challenges are present in utilizing the vast amounts of data for real-time operations such as in autonomous vehicles and signal timing optimizations using vehicle-to-infrastructure (V2I) communications. Real-time V2I data collection and utilization also presents privacy related challenges.

3.8.3.4 Applications

Mobile, web, and desktop applications used throughout the vehicle transportation ecosystem are grouped under this category. They allow humans to interface with other technologies. For
example, end-users use applications to view inferred insights like optimal navigation, and TMC operators use applications to configure and manage different field devices remotely.

3.8.3.5 Communication

Communication and networking technologies are grouped under this category. Communication channels between different technologies enable them to work in unison as a single unit. The connectivity can be within components or across components. Such connectivity can be provided using different standardized, custom built, or proprietary technologies. Latency, bandwidth, and reliability requirements dictate the type of communication channel used. For example, vehicles communicate with RSUs using dedicated short range communication (DSRC) while RSUs are connected to the TMC using optical fiber cables.

3.8.4 Stakeholders Involved

The transportation ecosystem is influenced by a number of stakeholders at various stages, each with their own responsibilities. We categorize the stakeholders as follows.

1. *End-users* the public using the roadways and the transportation technologies available to them.

2. *Vendors* software and hardware providers for the transportation ecosystem.

3. *Operators* individuals responsible for daily operation and maintenance of the transportation system.

3.9 Systematization of Knowledge

We use the systematization framework introduced in Section 3.8 to present the existing literature uniformly, across the two dimensions discussed above. The transportation infrastructure is an amalgamation of multiple disciplines and as such the body of research work is also spread across these different domains. Hence, we do not aim to provide a complete survey and instead focus on covering the breadth of this field, extracting key aspects (common attack techniques, proposed mitigations, stakeholders involved) for each dimension. We supplement this with real-world examples from our field experience and online resources where applicable.

The literature review itself was an iterative process with the initial reviews conducted in tandem with the fieldwork. When we discuss various security issues with the TMC personnel, we conducted extensive search for relevant articles, both peer-reviewed and otherwise, using search engines. These articles then further drove the discussions during the fieldwork as it brought forward relevant research topics that could be discussed within the context of the TMC. Where a large number of works covering a topic are available, we included those considered to be significant based on the content, number of citations, and publication venues. For articles that cover an important aspect in the ecosystem, even such papers are not highly cited or did not appear in highly ranked venues, we still included them if they provided valuable information. To make it easier for readers to navigate the selected articles, we provide an online portal to view them within our framework: https://transportation-sok.github.io/
3.9.1 Attacks on Sensing

3.9.1.1 Types of Sensing Technologies

Most sensing technologies are used in intersections for vehicle detection:

1. Traditional: Inductive loop, magnetometer, electro-magnet sensors, infrared sensors, microwave radar, etc.

2. Communication-based: ISM band radios (900 MHz / 4.9 GHz / 5.8 GHz), vehicle to infrastructure (V2I) communication.


These sensing technologies detect vehicles and pedestrians to control signal timing. Data collected are also aggregated to measure volume, speed, and travel time which are further analyzed, the insights from which can inform system-wide signal timing performance improvement [109]. Specialized sensors can detect radio waves emitted from transit buses, trains, and emergency vehicles to give them priority in passing. Automatic vehicle identification systems are used for toll collection, red light violation, etc. There are also sensors for highway ramp metering, measuring truck weight, and the increasing trend of sensors for monitoring the surrounding environment such as air quality.

We also consider time synchronization a type of sensing. This is used in roadside unit (RSU), and could be implemented through microwave links, GPS, or more modern computing technologies [27]. It is necessary for important tasks such as switching of signal timing plans based on time-of-day, and coordinating signal timing across multiple intersections along a corridor to achieve continuous traffic flow [108].
3.9.1.2 Attack Categories

Regardless of the concrete implementation of these sensing technologies, the goals of the adversary are: (a) to cause erroneous readings, (b) to disrupt the detection (Denial of service (DoS)/jamming).

Pre-acquisition attacks occur when adversaries deliberately forge/alter/introduce data/signals the sensors rely upon, leading to erroneous readings, malfunction, or jamming. Attacks include capture, replay, delay, signal forgery, flooding, and jamming. Both vehicles and RSU’s use location data from GPS, which are vulnerable to pre-acquisition attacks including replay, data spoofing, and jamming [18, 80, 112] leading to incorrect navigation [54, 83]. Pre-acquisition attacks against vehicle detection systems can induce both false positive and false negative results. Petit et al. [82] showed erratic detection by cameras through targeting the exposure control of the camera with bursts of light. Yan et al. [122] demonstrated missed detections (blinding attacks) in millimeter-ware (MMW) radars (used in Tesla cars) and video-based detection. For MMW, electromagnetic waves in the same frequency band as the sensors (76–77 GHz) flood the receiver, whereas video-based, LED and visible laser light sources flood the camera to blind the sensors. While these works are focused on sensors on vehicles, similar attacks can be effective for sensors in intersections. For communication-based detections, adversaries can conduct pre-acquisition attacks through attacking the communication channel (Section 3.9.5.2). Chen et al. [16] demonstrated such a practical pre-acquisition attack on a V2I system, the Intelligent Traffic Signal System (I-SIG), by assuming control of the source vehicle and sending fake trajectory data to the infrastructure. This attack bypassed the security credential management system (SCMS) [116], [107] of I-SIG by compromising an authenticated data transmitter (i.e., the vehicle) and fooled the sensor into accepting false data.
*Attack on sensing device* occur through direct manipulation of the sensing device with physical or remote access can lead to miscalibration causing erroneous readings, data injection/rejection, or device damage. The attack on GPS disrupting the signal timing disruption discussed in Section 3.6.2 is an example of this attack category. Such sensor disrupting attacks are also considered by Monteuuis et al. [69] in their attacker model for connected and automated vehicles (CAVs).

*Post-acquisition attacks* occur when adversaries exploit the sensed data in-transit as it is transmitted to other local/remote entities. These attacks are usually facilitated by weaknesses in the communication channel and/or on the receiving end. Cerrudo [14] found that the access point that received data from in-pavement wireless vehicle detectors built by Sensys Networks [73] did not require any authentication, allowing false data injection attacks. Obermaier et al. [76] found that weak authentication in surveillance cameras allowed the adversaries to impersonate as the camera to the cloud service and trigger motion detection events, inject forged video streams, or deny the camera of the cloud service completely (DoS attack). Such attacks on traffic cameras can lead inference systems to extract incorrect traffic flow data.

### 3.9.1.3 Impacts of Attacks on Sensing

Attacks on sensing impacts the performance of downstream inference and control technologies leading to undesirable impact on traffic flow and safety. During our fieldwork we learned that incorrect sensing data was a concern for the TMC engineers. While the concerns were mostly for malfunction, deliberate attacks would worsen the situation. As discussed in Section 3.6.1, inaccurate or false pedestrian detections at actuated intersections can cause unwanted relight stoppages even when no vehicles or pedestrians are present. Similar missed detections, as in the public complaint report, can lead to expected green phase to be skipped which can cause queuing effect in the
corresponding lanes and worsen traffic conditions as travel time increases. Pre-acquisition attack discussed earlier by Yan et al. [122] could also achieve similar effect at intersections with video-based detection.

Ernst et al. [27], using simulation, evaluate the impact of destabilizing time synchronization in a six intersection coordinated corridor and find that travel time through the corridor can grow linearly as time passes, causing significant queuing. The attack on GPS discussed in Section 3.6.2 can lead to this undesired outcome.

3.9.1.4 Possible Mitigations

Pre-acquisition attacks are difficult to identify and prevent as they occur outside the system boundary. Improving data validation and noise reduction could thwart some of these attacks [121, 124], but may not always be effective. Use of redundant sensing devices or technologies is often proposed [1, 63], but incurs financial cost for additional devices and computational cost to merge multiple data sources and may not always be viable.

To mitigate attacks on sensing devices, access to devices need to be hardened using existing security measures. For instance, the security issues disclosed by Cerrudo [14] were not new to the security community and were fixable using textbook security measures. This work exposed the lack of attention to security from vendors and the real world consequence. (1) More than 200,000 vulnerable devices (> $100 million) were deployed throughout many countries; (2) the issue was not simply solvable by patching since communication was not encrypted, and updates were not signed, making the patching process itself an attack vector; (3) the only real solution would be device replacement, complicated by the fact that these devices were buried under pavement and
meant to operate for decades. It would also require coordination with other agencies that deal with
the pavement, and the resulting costs would be prohibitive.

In summary, sensing drives modern capabilities of the transportation ecosystem, but it also
opens it up to input from untrusted sources. As such, it is the first line of defense. Vendors so
far have largely failed to implement basic security measures and are often shown to be unaware
of threats or neglecting them altogether. Retroactive solutions are not only costly but in some
cases infeasible. Hence, vendors bear the responsibility of designing products with security at the
forefront.

3.9.2 Attacks on Control

3.9.2.1 Types of Control Technologies

The functions performed by control technologies in the transportation infrastructure include:

1. Implementing signal timing: Signal controller

2. Actuating visual interfaces: DMS controller

3. Conflict monitoring: Specialized devices such as Malfunction Management Unit (MMU) or
   Cabinet Monitor Unit (CMU). It checks for conflicting greens or violation of minimum red
   and yellow times, and when such conditions are detected, turns the intersection into all-way
   flashing red.

   To perform these functions, control technologies have I/O interfaces to receive data from
sensing (direct or communication-based through V2I receivers), or intermediate devices such as
Serial Interface Unit (SIU) or Bus Interface Unit (BIU) which converts I/O signals to/from 24V
sources. The computing component is typically a general-purpose computer with standard processor and Linux-based operating system [7]. The traffic flow directives is then transmitted through the following means:

1. Visual-based: Traffic signals and dynamic message signs (DMS)
2. Communication-based: Infrastructure-to-vehicle (I2V) communication
3. Cloud-based: Devices (embedded in vehicles or carried by humans) receiving directives from the cloud.

3.9.2.2 Attack Categories

A computing unit accessible through various communication channels provides attractive targets for adversaries. By launching common cyber attacks, adversaries can gain control of the system or use it to infiltrate into other parts of the ecosystem. Adversaries can do so through:

1. Direct access: For example, front panel of the device or upload from external storage (data key or USB sticks);
2. Proximity access: For example, through local communication channels like WiFi using mobile or desktop applications, or from other devices connected to the local network;
3. Remote access: For example, through the wide area networks or Internet.

The feasibility of attacks through all these levels of access have previously been demonstrated in vehicles [15, 55], [36, 40, 60].

Attack through direct access represent threats where adversaries require physical access. Such threats often tend to be dismissed as unrealistic. As such the devices under the “protection” of
these RSUs are deployed with inadequate security considerations. In reality, such incidents do occur as the control panels and RSUs use shared standardized locks (Corbin style with #2 keys) with keys readily available for purchase [27], [98]. As discussed in Section 3.6.2, user authentication and access control capabilities available in signal controllers are not utilized in practice. Even when used, Ghena et al. [32] report that default settings are not updated in RSUs. Similarly, Kelarestaghi [52] report multiple cases of hacked DMS controllers with adversaries successfully altering the message displayed. These hacks only required layman knowledge with most simply exploiting the use of default credentials to update the messages displayed [47, 57, 58, 93, 117]. These controllers were also found to expose a vulnerable service which allowed credentials to be updated without having to authenticate first.

Attacks against the devices in the signal cabinet with physical access discussed in Section 3.6.2 are examples of this attack category. We demonstrated how the intersection can be trivially sent to all-way flashing red state, sub-optimal signal timing conditions, and even catastrophic unsafe conditions like flickering all-way green conditions.

Attack through proximity and remote access represent threats where the adversaries do not require physical access to the devices. Proximity and remote-range exploitations are enabled by I/O interfaces that expose vulnerabilities such as no/weak authentication in remote access services (commonly associated with embedded or IoT devices). Ghena et al. [32] and Zhang et al. [125] identify the following issues in RSU devices: (i) weak authentication: default and hardcoded credentials, and (ii) vulnerable services: exposed ssh, ftp and telnet services, remote login (rlogin), remote task management service, and debug service using Wind river DeBug protocol (WBD) — all could be exploited to access to the signal controller devices. Similarly, DMS is reported to expose vulnerable
services including open telnet ports, publicly accessible web-interfaces, and use of insecure protocols such as Simple Network Management Protocols (SNMP) [47, 57, 58, 93, 117]. As discussed in Section 3.6.2, we were able to gain root access on Econolite signal controllers as it exposed ssh service setup with default user name and password. Such attacks are also possible without direct access when the signal cabinet is quipped with wireless dongles.

With proximity or remote access to the signal controllers possible, we can remotely achieve unsafe conditions of one second flickering green light without triggering the MMU/CMU as discussed in Section 3.6.2. Independently, Ning et al. [74] achieved a stronger attack on MMU/CMU by carefully timing the conflicting greens down to 200ms, which is the transient time required to trigger a conflict state. This results in a flickering green and solid green on two conflicting routes, highly likely to lead to severe crashes in the intersection. These results show that remote attacks can drive intersections into unsafe signal conditions, even with MMU/CMU present as the final line of defense.

3.9.2.3 Impacts of Attacks on Control

Following impacts are observed through attacks on visual interfaces:

1. Performance and safety degradation: Displaying unsafe/unreliable information on the DMS regarding route information, real-time traffic conditions, variable speed limits, and weather conditions can lead to unnecessary traffic diversions or unsafe road conditions.

2. Distractions: Studies have shown that the credibility of DMS is important to achieve efficient operations as well as ensure safety of workers with automated work zone information systems
(AWIS) [78, 81], [23]. Even distracting messages can have long-term impact from reduced trust in and future ignorance of important safety messages.

Following impacts are observed through attacks on signal controllers and conflict monitors:

3. Unsafe conditions: Assumptions regarding safety guarantees through MMU/CMU is jeopardized as discussed above. This can lead to fatal accidents.

4. Inefficient signal timing causes increased delays and further produces environmental and financial consequences with increased emission and fuel consumption [92], [11, 84]. Additionally, with the increased connectivity, a single compromised network connected device will grant adversaries access into the transportation ecosystem [74].

3.9.2.4 Possible Mitigations

Physically securing RSUs with separate locks and use of strong authentication mechanisms in the devices within can harden against direct attacks. However, in practice such measures are turned down as it leads to operational hindrance for field operators to manage hundreds of keys and passwords. Use of RFID-based locks [68] might be another solution but requires updating hundreds of cabinets, a financial burden for municipalities. Apart from that, we also have to account for extreme cases such as power outage due to unforeseen circumstances in which case physical keys might be preferred. Mitigation of remote attacks is possible (in theory) simply by following best practices. Disabling unnecessary vulnerable services, avoiding use of hard-coded or weak authentication mechanisms, using virtual private network (VPN), and minimizing network exposure in general should harden the security posture of the transportation infrastructure [32], [47].
In summary, while control technologies are built on top of well established operating systems and software, common vulnerabilities, some even trivial, still dominate. Vendors and operators alike have not adopted available counter measures and instead relied heavily on the previously closed nature of the transportation infrastructure. Emerging threats are no longer limited to performance degradation. Safety critical attacks, previously thought to be impossible, have been demonstrated. Unlike sensing, which tend to be deployed and relied upon for decades without any changes, control technologies regularly interact with operators. Vendors, operators, and regulators all have a role to play in mitigating existing issues and improving the security posture.

3.9.3 Inference Technologies

3.9.3.1 Types of Inference Technologies

Inference technologies derive insights from both online real-time data, and offline historical data, to inform control and provide insights to operators. Telemetry data from dozens of sensors in vehicles are utilized for fleet management, route optimization, and predictive vehicle diagnostics [17]. Within RSUs, Connected vehicle (CV) based Intelligent Traffic Signal System (I-SIG) calculates optimal signal timings at intersections in real time based on trajectory data collected by V2I sensors at the intersections. At TMCs, inference systems aggregate information from multiple sources to produce insights that assist operators in traffic monitoring, incidence response, and system-wide planning [20], [25, 67, 113]. Data sharing agreements between TMCs, end-users, and third parties facilitate commercial applications for end-users and operators. Several real-time traffic signal prediction and advisory applications have been developed such as GreenDrive [126] and SignalGuru [56]. Mobile applications such as Connected Signals’ Enlighten [19] provides red
light countdowns and green-wave speed$^3$ advisories to drivers based on inferences from predictive models [33, 34]. Waycare [113] combines data from multiple sources to provide real-time traffic monitoring with predictions of future congestion and potential accident risk areas.

### 3.9.3.2 Attack Categories

Adversaries can attack the various inference technologies to (a) influence the insights extracted, (b) force real-time prediction to miss deadlines, rendering them useless, or (c) disable the system altogether. Vulnerabilities can exist in the inference system because of flawed algorithms and/or flawed implementations. Adversaries can exploit them either through direct attacks, enabled through vulnerabilities of the host platform on which the inference system runs, or through indirect attacks, where vulnerabilities in a different component are exploited causing malicious data to flow into the inference system.

**Direct attacks** represent threats where adversaries require physical access. The feasibility of attacks through physical access depends on the location of the host platform. Such attacks can be limited to insider attacks for TMC/cloud-endpoint components. Host platforms provide core services such as data ingestion APIs and remote administration to the inference system, all of which are attack surfaces for adversaries. Similar paradigm is found in the IoT domain, where security vulnerabilities are found to be exposed through weak authentications, insecure implementation of APIs, and mis-configurations [6].

**Indirect attacks** represent threats where adversaries do not require direct physical access. Adversaries can manipulate input data sources to inference technologies by attacking another component of the transportation ecosystem. These attacks are difficult to consider during the system  

$^3$Speed at which vehicles are likely to encounter continuing green lights in a coordinated corridor.
Figure 3.4: False Data Injection Attack Against the I-SIG Inference System. The figure shows I-SIG enabled RSU with additional context information for the vehicle. Arrow shows the data-flow path from the vehicle to the I-SIG system. In this case, vehicle’s control technology responsible for sensor data transmission is compromised, allowing malicious data-flow through to otherwise secure RSU.

design as the input sources are typically out of the designer’s control and often assumed to be trustworthy. Chen et al. [16] discovered that certain edge-case input data are not handled optimally by the inference system used in Intelligent Traffic Signal System, or I-SIG. Crafted malicious data can be injected into the system through the pre-acquisition attack discussed in Section 3.9.1.2, leading to inefficient signal timing to be generated. The attack path can be visualized using our systematization framework as shown in Figure 3.4. I-SIG is required to ingest untrusted data and compute optimal signal timing within two seconds. Ideally, data validation/noise reduction mechanisms should be in place to reject malicious input. But with only limited computational power available, a less optimal algorithm was implemented, introducing this vulnerability.

3.9.3.3 Impacts of Attack on Inference

Based on the use case of inference systems, attacks can have impacts on both performance and safety. Both performance and safety impacts are higher in real-time systems. The attack on the
above I-SIG system resulted in 23.4% degradation in travel time of vehicles through the attacked intersection compared to the base system without it [16]. This is a significant drop in performance, negating the benefits of the system. Unreliable insights generated from false data injections also impact the decision making process of the TMC operators and lead to performance and safety impacts.

The impacts are catastrophic in real-time safety critical applications such as in CV and self-driving cars. For example, Adbo et al. [1] report that critical information such as vehicle’s lane position and platoon identification number are not used when merging two platoons, allowing fake data to initiate merge between two platoons in different lanes to proceed. Hence, use of such data driven technologies in the vehicle transportation system must consider the possibilities of attacks on the inference systems through direct or indirect means.

3.9.3.4 Possible Mitigations

1. The inference system should be resilient against malicious inputs. Here trade-offs may have to be made like in the I-SIG example above. A holistic system evaluation is necessary to make informed trade-off decisions. Any assumptions made must then be reflected in the design of the other components of the ecosystem so that necessary security measures can be implemented. Dedinsky et al. [21] propose a system with video as a redundant data source to identify false data injections from connected vehicles. But such solutions require fundamental design changes and do not account for the limited resource constraint and other problems that may arise, such as, which data source to trust in case of conflicting information.
2. Secure API endpoints: API design should account for the underlying communication channel and incorporate appropriate authentication and encryption mechanisms.

3. Secure the underlying host platform: As with control technologies, unnecessary services should not be exposed, to reduce the attack surface.

In summary, inference systems provide critical insights for control and operator decision making. Incorrect insights can cause adverse impacts on performance and safety. Threats are particularly difficult to identify when these systems do not directly interact with external sources. Suggested mitigations include identifying critical data sources, sanitizing inputs, and augmenting it with redundant sources where possible.

3.9.4 Applications

3.9.4.1 Types of Applications

Mobile, web, and desktop applications are companions to the other technologies in the ecosystem. They provide output interfaces for monitoring in the TMC and advisory information such as speed limits and route planning to end users. They also perform critical tasks such as relaying information to the cloud, and providing configuration and management capabilities for sensing, control, and inference.

3.9.4.2 Attack Categories and Impacts

Ease of access to applications makes them attractive targets for adversaries. Security issues with mobile, web, and desktop applications, and the proper mitigations are well known to the security community and are not discussed in detail here. Instead, we shed light on the applications
used in the transportation ecosystem, the threats they pose, and the security considerations they make (if any) based on our fieldwork and appropriate literature.

*Vehicle and intersection components* have many end-user applications. For example, navigation applications are widely used by end-users. These relay location data to cloud-based inference systems which provides opportunities for post-acquisition attacks. Implicit trust in the applications by cloud counterparts facilitates false data injection attacks which can lead to manipulation of real-time traffic data and sub-optimal routing [50, 91], [100]. With the emergence of CV, applications have more safety critical utility. For example, Connected signals app [19] displays real-time traffic light predictions at each intersection. This changes the threat landscape from inconveniences to the users to potential safety hazards such as traffic violations and accidents.

*RSU components* also have applications to configure or manage devices, some with additional proximity or remote access capabilities. Ning et al. [74] find that devices implicitly trust the applications allowing a signal controller’s datakey to be flashed without authentication using an application developed based on the specifications provided online. Such implicit trust is dangerous and can be used to eavesdrop, leak configuration settings, user credentials, or reverse engineer communication protocols involved [14, 74].

Applications used in the *TMC* allow network wide management of signal controllers, DMS, toll gantries, reversible lanes, and so on. Others provide visibility into the entire system for traffic monitoring and data analytics to assist daily operations. These are powerful applications, the access to which can facilitate all the attacks discussed in Section 3.9.2.2 and securing them is paramount. As discussed in Section 3.6.3, TMC utilize many applications lacking basic security mechanisms such as use of authentication. Existence of a variety of applications utilizing technologies ranging
from decades ago such as the Microsoft DOS operating system to modern artificial intelligence based technologies such as Waycare [113] also presents a challenge to secure the infrastructure in the TMC as a single weak link can allow attackers into the system.

Monitoring applications include BlueTOAD [99] which utilizes data collected from traveller’s bluetooth-enabled devices at various intersections. Waze Traffic View [114] and Waycare [113] utilize crowd sourced and other open data sources. Although these applications have less security implications, adversarial access can misinform operators which can hide issues or lead to incorrect decision making.

3.9.4.3 Possible Mitigations

Mitigation for the shortcomings of legacy systems such as MTCS is to simply replace it with the latest technology. But in practice, this requires both device replacement and fundamental changes since it requires updating the backbone communication channel from twisted copper pair to fiber optics, and replacing all the cabinets on heavy traffic roads with the latest ones. This is not only cost prohibitive, but also requires coordination between several organizations as it would require construction work on the road, planning from the communication network perspective, and all the while providing smooth traffic flow through the city. This difficulty of upgrading legacy systems poses the primary challenge of securing the transportation ecosystem.

Vendors must practice secure software development practices and use vulnerability discovery tools where applicable. Assumption of closed-source nature of the transportation ecosystem for security is not valid as proven by acquisition of hardware and software artifacts through social engineering [14]. Implementation of appropriate authentication and secure data exchange mechanisms
with use of encryption and proper handling of cryptographic keys is a must. In addition to the normal secure application development practices, vendors must evaluate operational requirements and use cases to identify specific threats, impacts, and usability concerns in the transportation ecosystem. Different operational constraints in the transportation ecosystem means that the most prevalent solution might not be the most suitable.

In summary, implicit trust between technologies and companion applications is an enabler of attacks. These issues are also seen in the IoT domain but the level of risk is considerably more in the transportation ecosystem as applications can configure and manage the core technologies. The usability constraints also vary compared to the IoT domain in general as multiple operators require simultaneous access. Vendors must adhere to the established security principles, while considering domain specific challenges, to develop secure applications.

3.9.5 Communication Technologies

3.9.5.1 Types of Communication Technologies

In vehicles, multiple sensors and ECUs communicate with varying requirements on timing, bandwidth, and priority using standardized protocols such as Control Area Network (CAN) as well as proprietary ones. In RSUs, devices are connected using direct point-to-point connections such as synchronous data link control (SDLC) serial bus, or local area networks (LAN). National Transportation Communications for Intelligent Transportation System Protocol (NTCIP) facilitates communication between multiple TMCs and Advanced Transportation Controllers (ATCs) [75]. With the introduction of connected and autonomous vehicles (CAVs), vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and infrastructure-to-vehicle (I2V) communications are currently
being tested in the field [29], [103]. Vehicle-to-anything (V2X) communications is being standard-
ized as dedicated short range communication (DSRC) in the U.S. by IEEE [53] and as ITS-G5 in Europe by ETSI [28]. With the proliferation of electric vehicles, roadside charging units have introduced vehicle-to-grid (V2G) communication through wired power-line-communication (PLC) which provide complex interactions such as automatic billing [9].

3.9.5.2 Attack Categories

A typical claim for “security” is that the transportation network is air-gapped/isolated from the Internet. This assumption does not hold. A simple search on the Shodan [88] search engine reveals many Internet-connected signal controllers. During our fieldwork, we found that third-
party VPN access to the internal servers is common for maintenance, upgrades, and use of cloud services. This necessitates complete trust in the third party to maintain the isolation. At the TMC, workstations connected to the transportation network were in general separate from the personal workstations connected to the Internet. But in practice some workstations were connected to both networks for ease of use.

Vendors tend to claim security of products by use of proprietary protocols. Claims of security through obscurity using proprietary protocols is weak as reverse engineering of protocols is common [13, 90].

Considering the communication mechanisms and security properties, attacks can be catego-
rized into: (i) attacking weak crypto; and (ii) compromising network nodes.

Weak cryptography represents a major threat to the communication mechanisms. Propri-
etary protocols developed by vendors often omit security considerations and do not use authen-
tication and encryption mechanisms. This opens the possibilities for both passive eavesdropping and active man-in-the-middle (MITM) attacks [1, 14]. Cerrudo [14] reports that vendors of wireless vehicle detectors deliberately chose not to implement authentication and encryption when designing their protocol, claiming that the proprietary nature of the protocol led to the clients and vendors deciding against additional security. Researchers were able to capture the traffic within the range of the IEEE 802.15.4 2.4 GHz transceiver, reverse engineer the protocol, and carry out post-acquisition false data injection attacks. Additionally, the over-the-air update system was neither encrypted nor signed, potentially allowing firmware update worms against the system. Even with cryptographic measures, flaws in the protocol design allow cryptanalysis and replay attacks. Roufa et al. [48] identify that the static data packet format used in the tire pressure management system (TPMS) protocol was flawed allowing cryptanalysis and replay attacks. They were able to trigger alerts by flipping a single bit, and track the vehicles by retrieving vehicle identifiers from the data packets. Baker et al. [9] find that V2G communication for EV charging use unencrypted communication channels allowing extraction of personal identifiers through eavesdropping. Such identifiers can be used for financial gain through payment systems or user tracking. Ghena et al. [32] find that proprietary protocols used for communication between RSUs over the ISM band (5.8 GHz or 900 MHz) have similar shortcomings of lack of encryption, information leaking data packet formats, and use of simple transmission schemes such as frequency hopping spread spectrum (FHSS).

Compromising network nodes is another possibility for adversaries to exploit the communication mechanisms. The transportation industry has taken steps towards incorporating security in this front. DSRC is not secure by design, but IEEE 1609.2 standard looks to secure the communications by use of public key infrastructures (PKI). Laurendeau et al. [61] and Kreilein [59] note
that, even with cryptographic measures, DSRC can still be an enabler for attacks to the transportation ecosystem through deception attacks, denial of service (DoS) attacks (using false messages for spoofing or jamming), cryptographic attacks (private keys still accessible through OBD-II ports in vehicles), malware exploitation (using DSRC as a transmitter), and V2X exploitation (due to deployment of infrastructure without security architecture). USDOT has also defined a message security standard, the Secure Certificate Management System (SCMS) which ensures that vehicles and RSUs have valid certificates during communication [12]. However, a compromised node with valid certificates can still bypass these security measures [16]. Another major application of V2V communication is platooning. Adbo et al. [1] find deficiencies in the V2V Platoon Management Protocol (PMP) and demonstrate that adversaries can cause platoons to merge over large unsafe distances, slow down through fake obstacle detections, engage unsafe cross-lane merges, and even take over a platoon and lead it to dangerous situations.

3.9.5.3 Possible Mitigations

Implementing secure communication protocols is known to be a difficult problem as even the most widely used protocols like TLS/SSL have previously been found to have vulnerabilities [66]. Standardized protocols should be used whenever possible. Open specifications (vs proprietary) for protocols allow experts to discover issues and improve them [74]. Hence, when appropriate, vendors should opt for well established protocols over custom ones. Use of cryptographic measures is always suggested when custom protocols have to be created. Additional consideration regarding data packet formats, consistency checks are also necessary to avoid cryptanalysis and other side channel attacks [48]. Suitable replacements or upgrades to widely used existing vulnerable protocols should also be considered [64]. Applications involved in communication must also be secured. This includes
use of up-to-date implementations of cryptographic libraries, and secure storage of cryptographic keys. Legacy applications should also be patched to use updated protocols and enable encryption.

In summary, a myriad of communication protocols exist in the transportation ecosystem. For many purposes, the industry has taken steps to develop open standards which facilitate interoperability as well as allow expert to study and validate them. But security is not considered as a major factor from the beginning and only implemented as an add-on, often with use of TLS/SSL. Worrisome is that when security is not imposed by the underlying specification or compliance regulations, vendors consistently opt-out of implementing it. Hence, security-aware design must be adopted and operators and regulators must demand higher security standards from the vendors.

3.10 Discussions

Within the transportation ecosystem, the roadside infrastructure is still in its infancy of technological advancement. This often means that the impact of cyber vulnerabilities is not as high yet. However, we are able to identify a number of weaknesses that portend much graver impacts should the system move quickly into the connected vehicle/autonomous vehicle paradigm, as many contend is happening. First, there is an over reliance on the isolated nature of the transportation infrastructure for security. Second, there is a race for rapid development of feature-rich products. Third, stakeholders leading the development often do not have adequate technology background, with many vendors evolving from developing mostly electrical systems to modern computer applications and cloud-based services. Lastly, usability always trumps security. In an operational environment such as the transportation infrastructure where many operators are not “cybersecurity savvy”, operational convenience is even more emphasized than cybersecurity practices.
As seen from Section 3.8, the vehicle transportation ecosystem is an intricate one where different technologies rely on each other to perform their tasks. Hence, the need for security might not be obvious as identifying attack goals, attack paths, and impacts can be difficult. To answer “why someone would attack a device?” one needs to not only know what the device does, but how it interacts with other devices in the ecosystem. Understanding the interactions between devices also helps identify potential attack paths. This information is not obvious to the developers working only on a single technology in the transportation ecosystem, as evidenced by the vendor’s decision to not include authentication and encryption on wireless detectors discussed in Section 3.9.5.2. The impacts of attacks and incentives to achieve them are also not easily observable. The primary impact is congestion and, even with the development of a successful exploit, its impact can only be shown through simulation models, which is not as “eye-catching” as real demonstrations of potential harm as shown by vehicular hacks. But congestion is a real problem. A study conducted by INRIX in 2019 estimates that drivers spent 99 hours in traffic per year in the U.S., which amounts to an average cost of $1,377 per driver ($88 billion annual cost) based on the FDOT’s time loss valuation [84]. Until recently, the impact on safety was thought to be limited as conflict monitors were assumed to protect against conflicting greens by sending the intersection into conflict flash [32]. The flickering green attacks weakened this major safety assumption [74].

3.10.1 Emerging Threats

The transportation ecosystem is a critical infrastructure and is going through a rapid technology-driven overhaul. With increasing involvement of technology in mobility of both vehicles and pedestrians, the impacts of cyber attacks is only rising. Most of the applications being tested in the three CV pilots in the U.S. are designated as safety features — 12 of 15 in New York [104],
9 of 13 in Tampa [105], and 5 of 5 in Wyoming [106]. Once these are deployed and relied upon, cyber attacks will have even bigger safety implications. As we have seen, mitigating issues after deployment in the transportation system has many barriers — large-scale deployment, difficulty in access, financial restrictions, and involvement of multiple agencies. Hence, future deployments of technologies must first be put through a thorough security evaluation.

Security is considered a niche domain that requires expert knowledge and yet deploying a secure system requires collective effort from everyone involved. From the perspective of a user, it is the responsibility of the vendors to develop adequate security measures in all technologies. But to maintain security, users also need to utilize the available security measures. For the vendors, conducting a thorough security evaluation not only requires technical expertise but also domain knowledge from multiple disciplines to extensively identify attack goals, attack paths, and potential impacts. City-wide deployments of transportation infrastructures utilize technologies from multiple different vendors that have to work in unison. Hence, regulatory bodies also need to step in so that security standards are maintained by all the vendors involved. We believe that our systematization approach helps to breakdown the silos of each discipline and get everyone involved in the security discussion by abstracting away technical details into five core technologies which can then be used to describe any part of the transportation system. This provides the necessary context required to identify attack goals, uncover attack paths, and reason about attack impacts. Security experts can then use this information to flesh out the details in the development phase.
3.11 Related Work

Ghena et al. [32] partnered with a local transportation agency to conduct a thorough vulnerability analysis of the signal controller and associated communication channels. To the best of our knowledge, this is the first such analysis and presents important findings such as remote manipulation of signal timing. Ning et al. [74] partnered with a local municipality to perform vulnerability analysis for vehicle traffic signal system. They demonstrate an attack that practically achieves all-way-green state at intersections, breaking the safety assumptions previously thought to be provided by CMU/MMU. Ernst et al. [27] present a framework for threat assessment of traffic cabinets with four levels of access and evaluate the potential impact using simulation models. Abdo et al. [1] present a systematic analysis of safety and performance impacts of attacks on CV platooning protocols. Dominic et al. [22] propose a risk assessment framework suitable for threat modelling in cooperative automated driving applications. Baker et al. [9] broaden the scope of vehicular transportation ecosystem and investigate EV charging infrastructure and security issues in the corresponding vehicle-to-grid communication. In comparison to these prior works on transportation infrastructure security, our unique contribution is creating a systematization framework which encapsulates the transportation ecosystem, and using it to organize/communicate the security issues in a manner that is at once abstract and context-aware.

Ethnographic fieldwork has been used to study security operations centers (SOC) [94, 96]. Our ethnographic fieldwork in the TMC was inspired by this prior work, but had a different type of purpose and outcome – using the insights gained during the fieldwork, we formulated a systematization framework for the broader field of transportation system cybersecurity.
3.12 Conclusion

We present a systematization framework for understanding and communicating cyber security knowledge in the vehicle transportation ecosystem. The framework views the ecosystem through two dimensions: components and technologies. This two-tiered view allows us to discuss security issues that are both common in pattern across multiple components, and intricate and context-specific due to inter-component interactions. This provides a coherent framework through which to communicate security risks to the various stakeholders, and a useful tool for security evaluations to derive insights into attack goals, attack paths, and potential impacts.
Chapter 4: Study 2 — Secure Software Development

4.1 Background

A wide range of research has addressed how to best establish secure development practices within a software development team/company. The standard approach is to use a secure development model to formulate a suitable secure software development lifecycle (S-SDLC) [31, 46, 123, 133]. Despite the success of S-SDLC in its originating company, successful establishment of secure development practices has remained more difficult for the software industry at large [46]. In general, some companies are unwilling to place code security on a level playing field with business considerations such as time-to-market of the product. There are also companies who make an effort to deliver secure code through the adoption of secure development life cycles but have not been able to do so effectively. Reasons for such failures have been identified as lack of security knowledge in developers, lack of available resources, lack of usable security, improper use of security APIs, etc [2, 38, 118, 120]. Use of security tools is often suggested as a way to help alleviate such problems by catching developer mistakes before they land in the product. However the adoption of security tools into development itself can remain an issue [119]. Some studies allude to the notion of security mindset or security culture within the company as the influential factor in driving these secure development practices [8, 44, 131]. But what is a security culture? What are the benefits it provides and how can a company start to develop such a culture?
This chapter describes the eight month long ethnographic study of a software development company which adopted the secure software development lifecycle (S-SDLC) approximately two months prior to the PO joining the development team. This provided an invaluable opportunity for the research team to examine whether and how secure development practices can take hold in a software development team when there is buy-in from the management. The chapter starts by describing the specifics of the fieldwork setup before moving on to the observations of the company’s overall software development processes and challenges facing secure development throughout the fieldwork. The following section discusses the observation and analysis of the positive shift in the development team’s practices regarding secure development. Finally, we describe and discuss the role of an interactive learning process in the creation of a set of preferred practices within the team which defines the “culture” of the team and how this process can be leveraged to give rise to security-aware developers and start a security culture in a software company.

4.2 Fieldwork Setting

The research team for this work comprised of me, the participant observer (PO), two computer science professors specializing in systems security and programming languages, and a professor of anthropology.

The fieldwork was conducted at a software development company headquartered in the United States with offices throughout the world. The researcher was embedded in a development team responsible for two security-related products developed by the company, referred to as P1 and P2 in this paper. Product P1 provided network access control capabilities. Product P2 was a cloud-based service for secure remote access. Due to the ongoing COVID-19 pandemic, the mode
of work varied between work-from-home and on-premise. The company followed local government guidelines; mandated work-from-home when stay-at-home order was in effect, and provided the flexibility of either work-from-home or on-premise otherwise. For on-premise work, the company followed all safety precautions by reorganizing the office setup to socially distance cubicles and providing masks and hand sanitizers. All meetings were held through video conferencing even when on premise. The advantages of being on premise were ease of access to the test environment and ability to start impromptu discussions and meetings when necessary.

The PO spent three days per week at the company for a duration of eight months. The PO participated in all activities a regular software engineer at the company would, such as sprint planning, scrum meetings, bug fixes, feature design/implementation/testing, and code reviews. Ample industry experience and a background in security allowed the PO to integrate quickly into the daily work.

4.2.1 The Development Team

The main participants were five software engineers (SWEs) in the development team, two network engineers, two support engineers, two sales/customer relations representatives, one quality assurance engineer (QAE), one graphic designer, and one vice president (VP), who also oversaw the management of the two products. All SWEs had at least 1.5 years’ experience within the company, with two having more than five years’ experience. The QAE joined during the fieldwork.

4.2.2 Methodology

The fieldwork data was analyzed by the research team utilizing the general inductive approach described in Section 2. The descriptive anthropological fieldnotes was also supplemented
with the following data points which enriched the data analysis in order to draw out the major implications of our observations to secure software development practices:

- Technical notes: For all the security incidents encountered, tracked the process of identification, technical details, and the progress made towards mitigating the issues.

- Relevant artifacts: Code, tickets, internal wiki pages.

4.3 Software Development Processes and Challenges Facing Secure Development

Approximately two months prior to the researcher joining the development team, management instructed the team to employ secure software development lifecycle (S-SDLC). This provided an invaluable opportunity for the research team to examine whether and how secure development practices can take hold in a software development team when there is buy-in from the top. In this section, we describe our observations of the company’s overall software development processes and challenges facing secure development throughout our fieldwork. In Section 4.4 we focus our discussion on observations and analysis of the shift in the development processes as a result of the management push for S-SDLC.

The company adopted a sprint-based agile development model. An issue-tracking tool was used for planning and tracking the development progress throughout a sprint. We describe this process below.

4.3.1 Sprint Planning

Everyone in the team was free to create a ticket for any work that was not already tracked and would be added to the backlog queue. However, only the lead SWEs could approve the ticket
for development. In addition, the VP and customer facing specialists could add feature tickets based on company vision and customer requests/feedback. Each week the VP and the lead sales representative, along with the lead SWEs had a prioritization meeting where the new tickets were discussed, approved/rejected for future development, with the approved tickets ranked based on priority. For each sprint, the SWEs and QAE conducted a sprint planning meeting where the highest priority tickets from the backlog were discussed and assigned story points representing the estimated complexity/amount of work. Story points for each ticket were agreed upon by the whole team using SCRUM poker [137], where each SWE and QAE anonymously assigned story points based on their understanding of the required work. When the assigned scores varied widely, a discussion was held to allow each SWE/QAE to explain the reasoning for their scores and SCRUM poker was re-done, until everyone converged on a common score. A total of 60–70 story points were targeted for each sprint, which allowed for a small number of additional high-priority tickets or unforeseen issues to be included in the sprint at a later time.

4.3.2 Development Workflow

A short 20–30 minute scrum meeting was held every morning to provide brief updates on the progress from the previous day, any issues/roadblocks encountered, and goals for the current day. The meeting was led by the lead SWE and included all SWEs, the QAE, the graphic designer, and the VP. Additional meetings were called by individuals as required to discuss ticket requirements, design issues, knowledge transfer for codebase, or testing strategies. We next discuss the stages of development and the challenges facing secure development in the context of each stage.
4.3.2.1 Design

A high-level design discussion was held during the sprint planning. For simple tickets, the SWE assigned took the responsibility of finalizing the design. For more complex tickets, discussions were held with the appropriate team members. In some cases, a wiki page with suggested alternatives was requested before such discussions.

Including security as a part of the design consideration presented the following challenges.

- Challenges regarding security knowledge of SWEs: During the design stage the main focus was to achieve functional correctness and performance considerations when applicable. When the features dealt with sensitive information, security became a necessity. Yet, secure design practices were not always the focus and instead assumed protection through “security functions” such as authentication and authorization. For example, one SWE when asked if the input attributes should be validated:

  “I’m not sure I’d worry too much about that. This form is authorized for admin only, so customers won’t be changing this attribute themselves. Trying to validate that they’ve provided a valid <redacted> attribute feels kind of complicated…”

Secure design must determine relevant threats (through threat modeling) and consider all aspects of the software, but it is a tall order to require all SWEs to have such knowledge and skills.

- Challenges in understanding contextual knowledge by a security expert: One of the software engineers in the team had a significant background in security (called SecSWE hereafter). He was able to identify the relevant security risks and propose secure design solutions. However,
although he had been working with the team for more than 1.5 years, the knowledge of the
minutiae of how the product operated was lacking and the proposed solutions were not always
directly applicable for the product at hand. Since there was no one-size-fits-all solution to
security problems, secure design required in-depth knowledge of both security and the product.

4.3.2.2 Implementation

SWEs picked up tickets from the sprint plan to implement. Again, the first priority of
SWEs was to implement the functional requirements of the tickets. We observed the following
secure development challenges in this stage.

- Challenges regarding security knowledge in SWEs: Even with security considerations in the
design phase, the actual implementation of code could expose vulnerabilities if the SWE was
not capable of defensive programming and unaware of secure development practices. The
main challenge is that the SWE needs to be able to identify the potential security risks in
the code that he/she is writing. Other factors such as reliance on frameworks, or incorrect
use of frameworks or APIs can also lead to insecure implementation. Such lack of knowledge
in SWEs cannot be simply compensated by the presence of a security expert in the team.
Usually in such cases the identification of security issues shifts further down the development
process during code analysis or security testing and presents additional challenges — the issue
might be missed altogether, fixing the security could require significant code changes or even
design changes, or there might not be enough time in the sprint to fix the issues which might
lead to the ticket being excluded from the sprint.
• Challenges in applying security knowledge in practice: In certain cases SWEs were able to identify potential security risks and also had the necessary knowledge to resolve them, but chose not to do so. We concluded this based on our data where we observed that during discussions regarding security issues, often times some SWEs were able to propose the solutions, but did not apply them in practice. This could be attributed to multiple reasons such as lack of time, reliance on security functions (such as authentication and authorization), or security of code considered “invisible” to the customer compared to the feature itself.

4.3.2.3 Continuous Integrations/Code Analysis

Once the implementation was complete, the source code was pushed to the remote repository where automated builds were carried out by the continuous integration (CI) pipelines. These pipelines executed unit/integration tests and code analysis tools such as SonarQube [134] (later Black Duck [135]) on the feature branch.

• Challenges on fully utilizing available tools: We found that the available tools were not utilized to their full capabilities. SonarQube was not maintained and mostly only relied upon for simply lint and code quality checks. The team was asked to set up Black Duck, a tool that analyzes the use of third-party open-source libraries in the codebase and provides information on licenses and known security vulnerabilities, into the CI pipeline as a part of the secure development effort. During discussions on the initial results of the scan, one SWE remarked: “We use quite a few out-dated packages. I would be surprised if this tool didn’t report any issues.” Black Duck was setup on management’s request, and the scans were initially enabled by default in the CI pipeline. But the resulting build failures in the Black Duck stage prevented
SWEs from merging in code. The tool was then disabled by default and a separate ticket was filed to track and address the vulnerabilities discovered.

4.3.2.4 Developer Testing

Before a ticket was assigned for code review, SWEs made sure that all automated tests were passing, deployed the updated product on a test environment, and performed their own testing. These tests were usually targeted towards functionality rather than security. Once functionality was verified, the ticket was updated with the steps to replicate the test plan and assigned for code review with the creation of a pull request.

4.3.2.5 Code Review

Two SWEs were assigned for code review. Usually one SWE provided thorough review while the other would just sanity-check the code. Depending on the complexity of the feature, the reviewers may perform quick functionality tests on top of going through the code changes. Any missing pieces, mistakes, inconsistency or departure from existing best practices in the coding pattern were set up as tasks to be addressed before the ticket was marked as “done.” We observed the following challenges in this stage.

- Challenges in consistently performing code review: Occasionally, when the changes were required urgently, code review was essentially skipped with the SWE just describing the changes made to others and asking if anyone objected to the approach. This could lead to potentially identifiable issues propagating to the production code base.
• Challenges in thoroughly performing security review: Although SWEs provided good feedback during code review, the suggested changes were based on internal best practices and patterns followed in other similar modules in the product. However, a thorough security code review requires more in-depth security knowledge and experience which was lacking in the SWEs. SecSWE however was able to provide specific security-related feedback. A potential API misuse of a crypto-library (bouncycastle [136]) was identified by SecSWE during code review. While addressing this comment, it was discovered that the API misuse could have caused memory leaks leading to out of memory conditions.

4.3.2.6 Post Development Testing

The QAE, who was hired during our fieldwork, prepared thorough test plans for each ticket in the sprint and carried them out on the test build. We observed the following challenges at this stage.

• Challenges in acquiring contextual knowledge by QAE: Although the QAE had years of prior experience, he was new to the team and the products, and hence required assistance to set up test environments and understand the specifics of the product before he could create strategic test plans. The QAE also had a security background and showed interest to learn and practice security-oriented testing along with SecSWE. But he expressed lack of time and in-depth knowledge about the product as reasons not to do so at that time.
4.3.3 Product Release

At the end of a sprint, a build of the product including all implementations in the sprint was deployed within the company for up to a week; then release notes were written and the product was released. The customers were required to opt-in for the updates, after which the support team executed the remote update procedures.

4.4 A Shift in Secure Development Practice

Shortly after the researcher joined, one SWE from each product team was assigned to be a member of a “virtual” application security engineering team and tasked to help drive security improvements for the product. This was part of the secure software development lifecycle (S-SDLC) effort that was kicked off before we joined. The designated SWE performed security-related tasks in addition to the normal sprint work. SecSWE was assigned this role for his team.

4.4.1 Little Impact at First

During the first three months of the fieldwork, the only security-related work fell into two new categories of tickets created as part of the S-SDLC efforts.

- **CSF tickets**: security-related tasks guided by the NIST Cybersecurity Framework (CSF) [127].
- **ASVS tickets**: compliance with OWASP Application Security Verification Standard (ASVS) framework [129], for web facing application components. Level-1 of the framework requires that web applications do not expose any easily discoverable vulnerabilities included in the OWASP top ten [128] or other similar lists.
These tickets were not included in the sprint plan. SecSWE and another developer (SWE1) were tasked to work on these tickets alongside the sprint work. Both SecSWE and SWE1 worked on these tickets individually, and the only updates about this work was provided briefly during morning scrum. These tickets were referred to as “burning cycles” and often the updates on these efforts carried little information:

- "I completed another CSF yesterday."
- "I knocked off a couple of CSF tickets."
- "I talked with <management personnel> about some CSF work and what is expected."
- "I will be catching up on some neglected CSF work and write up some wikis."
- "Maybe I will pick up some CSF stuff in my spare time."
- "My changes are in PR. I will next work on ASVS tickets while I wait for reviews."
- "I am working on a P2 ticket and also doing some ASVS audits."
- "I finished 4 or 5 ASVS tickets."
- "I have a lot of ASVS work to do."

When talking to SecSWE on how the security work is going, he remarked:

“I don’t know. It takes a lot of work for this ASVS stuff, looking at all the code, testing, researching... I feel like we are putting all of this effort and time on this but nothing is being done about it you know.”
Although significant effort was put on resolving the CSF and ASVS tickets, we did not observe any impact on the development workflow as a whole.

4.4.2 Making Progress

During the third month of the fieldwork, SecSWE started to work on threat models for both products. In order to gain an in-depth understanding of the product architectures, a meeting was requested with the lead SWEs. However, as P2 was pending a major release during this time, the meeting was postponed for after the release and SecSWE was asked to focus on P1 till then. Nonetheless, he first shared the initial threat model for P1 with the team for feedback which garnered greater visibility on the security work in the development team as a whole and initiated discussions on the communication patterns between the different microservices in the product. SecSWE also documented the security issues in a wiki page in order to facilitate the pending discussions for the threat modeling work.

Prior to the threat modeling work, two security tickets had also been logged: (1) The PO discovered that the same key pair was reused for all customers when P1 was setup as a high availability (HA) pair. (2) On further investigation, SecSWE discovered another instance of key reuse problem in establishing connections to the cloud server. The threat modeling work also initiated discussions and feedback from other SWEs concerning these issues.

Another key mismanagement issue was discovered where a private key was exposed in a publicly accessible server. A wiki was created detailing the potential misuse cases of this error. The initial response from other SWEs was that this server, while Internet facing, was not advertised to the public as it was mainly used to distribute software updates. Discussions on the potential
misuse cases of this issue in particular garnered positive interest in security work with the lead SWE remarking:

“I am excited about the work SecSWE is doing.”

4.4.2.1 Security Scrum Poker

With several security tickets logged, SecSWE suggested to have a meeting specifically to discuss these tickets before the next sprint. Prior to the meeting, everyone was asked to review the tickets and corresponding wiki pages for discussion. SecSWE also introduced the DREAD risk assessment scheme [65] and the security scrum poker (akin to scrum poker) in order to assess the estimated risk of the discovered vulnerabilities. The goal was for the entire team to converge on a risk score for each ticket, discuss the rationale behind the scores in case of mismatch to clarify everyone’s understanding of the issue, and ultimately use the risk scores to prioritize the security tickets.

4.4.2.2 Putting Security into Development Context

Three security scrum pokers were held during the fieldwork. In the first meeting, two SWEs tended to score lower than the others. As with scrum poker, in case of mismatch the SWEs were asked to explain the rationale behind their scores. This brought forward any misunderstanding of the discussed issue and allowed the group to clarify them. After a couple of iterations, one of the SWEs kept having varying scores and tried to move on to another ticket by agreeing with the others’ scores but the lead SWE remarked: “You cannot just do that. Either you have to defend the score or tell us why you changed your mind.” The whole team agreed that the meeting was very
fruitful in clarifying their understanding of the issues and/or the proposed solutions with the SWEs remarking:

- “That was more productive than I expected.”
- “I really liked this session and the discussions cleared things up. I am excited to see where this effort leads.”

These discussions led to contextual analysis of the discovered issues (what is the risk in the system?). They helped uncover root causes of existing issues and bring forward discussion on potential solutions, trade-offs for alternatives, and potential road-blocks in implementing them. Importantly, these discussions were useful to SecSWE as well.

The discussions between SecSWE and the lead SWE led to the understanding of how and why the private key ended up in the public-facing server in the first place — it turned out that previously P1 was distributed to the customers using Preboot Execution Environment (PXE) boot over the network. Although this method had not been used for several years, it was still used internally to quickly deploy test environments. As setting up internal test environments did not require the PXE boot kickstarter script to be on a public facing server, it was subsequently moved to an internal server during the fieldwork. This task required collaboration between SecSWE, lead SWEs, as well as the networking engineers to implement, test, and deploy. For the cases of reused keys, short-term solutions of limiting users to only required commands while restricting shell access altogether were proposed. A longer-term goal to set up a per-deployment key management and distribution mechanism was also discussed. During the fieldwork only the task to research the approach was created.
After the first security scrum poker, SecSWE asked others to also report any security issues they found. During the course of the fieldwork 15 security tickets were created that were not related to ASVS or CSF. The following are the categories of vulnerabilities discovered during the fieldwork.

- Mismanagement of cryptographic keys and certificates.
- Lack of access control to remote assets
- Improper handling of passwords
- Unencrypted application update channel
- Remote code execution (RCE)
- Cross-site scripting (XSS)
- Privilege escalation
- Hard-coded credentials
- SQL and command injection
- Mis-configured SAML (Security Assertion Markup Language) authentication

These issues were discussed in at least one security scrum poker meeting. SecSWE and the PO also developed proof-of-concept (PoC) attacks for application-level vulnerabilities such as remote code execution, XSS, Privilege escalation, and SQL and command injection which helped drive further discussions. Out of the 15 security tickets identified, 8 were approved for development after going through both the security scrum poker and the prioritization stages. Six of the approved
tickets were included in a sprint plan. The researcher asked for SecSWE’s opinion on the increased focus on security. The response was:

“I am surprised by the increased focus on security as well. They were not at all interested in these stuff before...I had already reported some of these issues before, although I didn’t have time to make PoCs for it. But it’s good that we have some attention now.”

4.4.3 Challenges in Security Ticket Prioritization

Although work was done to identify security issues, getting them prioritized for development still presented challenges.

- **Security tickets were not considered “real”:** Purely addressing existing security issues or improving security in existing code/infrastructure was not considered as “real.” In one sprint planning meeting after a few security tickets were discussed and included in the sprint, the lead SWE remarked: “Okay now let’s include some real tickets in here as well.” The basis for this point of view seemed to be that security improvements made to existing features or to the infrastructure were not visible to the customers.

- **Security tickets had higher story points:** Many security tickets were voted to have high story points and hence would not leave room to include other feature-driven tickets. The reasons for higher scores include:
  - **Technical challenges:** Security tickets required more research and experimentation to figure out the most suitable solution for the product.
- **Dependencies**: Fixing existing security issues required identifying all use cases of the vulnerable feature and the impacts of the changes on the product. Finding dependencies itself was time consuming as documentations may be outdated, and additional developer and QAE testing would be needed.

- **Implied changes in processes**: SWE/support/QAE may be relying on the vulnerable features and may not want to change. SWE may need to provide viable alternatives.

  “Before we move on with the fix, we need to first find out if there are undocumented use cases of these things. This is not uncommon with the support team to have some automated scripts which might rely on some access or some feature and we do not want to break them.” This could lead to additional work.

- **Legacy systems**: Older systems already deployed at customer sites may still need to be supported. In such cases, alternative solutions needed to be provided or both new and old systems needed to be supported. Some security holes may be impossible to resolve because of initial bad design. One ticket was blocked due to this very reason as around 20 customer sites were yet to be migrated to the updated system.

- **Meeting Customer Requirements**: Customers were unwilling to allow change of existing features. During a discussion for changing the rule specification UI, which introduced command injection vulnerability, one SWE mentioned that they had already tried to remove that feature before as the product already had an updated alternative built in. But the customers were unwilling to migrate to the new feature as it meant that they had to transition all the existing rules to the new format and they were unwilling to do so. SWE said that he already knew what this customer would say:
“If there are security issues then that is your problem and you need to fix it without
taking away my features.”

- **New customer requests:** During the course of a sprint, new high-priority customer tickets may be received. In such cases the security tickets would be de-prioritized, as happened to two security tickets included in the sprint plan.

4.4.4 Security-aware SWEs

After the introduction of security scrum pokers, there was an increase in security-related discussions outside the meetings as well. These ranged from humorous comments — “SecSWE is not going to be happy if you do that.” or “He is the security police now!? <laughs>” to positive reactions for including security tickets during prioritization meetings: “SecSWE and <the researcher> are pretty good with security.”

- **Security considerations in other tickets:** In addition to the security tickets, security considerations were made in three other feature tickets.

  1. **User-side error reporting for failed certification validation.** The researcher was assigned this ticket to which led to a major refactoring of the code and use of an updated single library for performing uniform certificate validation throughout P1.

  2. **Enabling use of new certificate for SAML authentication without requiring application restart.** A certificate reuse misconfiguration was discovered while working on this ticket. Code was refactored to allow proper configuration changes as well as utilize the refactored certificate validation library.
3. **Sending real-time alerts to customers.** This ticket required enabling communication between the on-premise customer deployment and the cloud server. This opportunity was used to improve the access control on the cloud server by allowing access to only the required commands and disabling shell access altogether.

- **Potential security issues identified:** Security issues were also brought up and discussed by other SWEs.

  1. An SWE discussed potential XSS vulnerabilities in another team’s application while working with them, and advocated for the other team to consider upgrading a programming framework to the latest stable version.

  2. Input validation was added in multiple modules proactively by SWEs working on a ticket with UI changes. Often they asked (in person or over Slack) if validation code was already implemented in the module or where to look for reference validation code. In cases where validation was complicated like in the P1 lead SWEs proposed how the validation could be done.

    “Do we have any input validation code that is used both by `<microservice1>` and `<microservice2>`? If so, do you remember where it is located?”

    “...I know that it’s not a priority for management to validate input that is supposed to be entered only by support, but it doesn’t cost much.”

- **Security considerations in design:** A feature requested by a high-priority customer required the ability to access internal configuration options otherwise hidden behind the application for an unorthodox use case of product P1. The initial design for the feature had not considered
security risks with the assumption that this feature would only be accessible by the administrative account, which belongs to the support team. SecSWE pointed out that such design could potentially expose command injection and privilege escalation vulnerabilities and started a discussion on the feature, which led to the finding that the original design had overestimated the access requirements to implement the desired functionality. The initial design was then shelved with a follow up design discussion scheduled to allow time to gather information for a more secure approach.

4.4.4.1 What Was Driving the Change

On analysis of the fieldnote data, we find that the positive shift in the development team’s security awareness can be attributed to the software engineers being able to identify the applicability of the security knowledge within the context of the everyday work they performed. We observe that by working along with others in the team to apply security knowledge under the concrete context of the software products, the software engineers became attentive to security risks when similar situations were encountered later. When a considerable number of discussions had taken place on a security-related topic, the group ended up with an agreed-upon set of knowledge and the associated set of practices became the “preferred practice” for dealing with this security concern. At this stage, considering this specific aspect of code security became the group’s “habit.” Later if some SWE in the team needed to work on a relevant part of the code but lacked this specific piece of security knowledge, they would seek guidance from others in the team, in the same manner as they would with other types of development tasks. They then learned and executed the preferred practice of the group.
Our analysis of data shows that what was driving the positive change was the learning dynamics existent in the development team. The initial lack of visible impact from management pushing for adopting S-SDLC was because the CSF and ASVS tickets were detached from the SWEs' regular work, and thus the relevant security knowledge did not have much opportunity to be directly applied in their work. It turned out that application of knowledge was the key driver for learning in an environment like a development team. Later on when SecSWE started to use security scrum poker in the threat modeling work, and involved all SWEs in the discussions, the security knowledge became concretized and contextualized. This drove a learning cycle within the team that allowed the SWEs to start obtaining relevant security knowledge and become more security aware.

We find that understanding the learning dynamics in the development team is crucial to effectively push for secure development practices. In fact, making developers more security aware is no different than cultivating their knowledge in any other aspect of software development. Our data indicate that, to establish a security culture in a development team, it might be helpful to follow the same learning dynamics that drive how culture forms for that community.

4.5 Learning in a Development Team

The analysis of our fieldnote data yielded a model that explains the establishment and evolution of preferred practices in a development team and hence the progression of its culture. In summary, the development team is a situated learning [62] environment where the process of learning drives the creation and evolution of preferred practices. When SWEs needed assistance, they acquired the necessary knowledge from the team. As they performed their task and applied the knowledge in practice, it provided the necessary platform to further drive the process of learning
and started to make contributions to the group. This process iterated over many cycles, until the group reached a point of saturation where the knowledge developed within the team was sufficient to facilitate progression in the task at hand. When this process was applied in practice, it not only led to professional growth of the SWE but also served as validation for the knowledge which then became a part of the current culture of practice.

4.5.1 Subject Matter Experts (SMEs)

As is common in software industry these days, the products the company built were vast entities and no single SWE knew the details of all aspects of a product. Multiple dimensions of knowledge were required within the development team in order to build the software, and the in-depth knowledge of each dimension was scattered between different SWEs in the team. An SWE can be the subject matter expert (SME) for some dimensions while at the same time being a novice in others.

When an SWE had the most in-depth knowledge on a topic within a development team, they were often called a subject matter expert (SME) of that particular dimension of knowledge. Although everyone in the team may have a good understanding on the topic, the SME was the one who understood the underlying details of the implementation. When an SWE started to work on a task new to them, they first went (or were directed to go) to the SME on the team. This created an implicit hierarchy within the team based on the dimension of knowledge under consideration, which facilitated the flow of knowledge within the team. This hierarchy transcended job titles. For example, despite holding a junior position in the company, a new hire who had worked on a task could immediately become the SME on certain pieces of knowledge associated with the task, and any future queries related to these pieces would first be directed towards them.
We observe the existence of SMEs throughout our data. When trying to set up a test environment for a new router device, an SWE asked the group: “I have read through the documentation but I still cannot get it to work in our test environment. Can anyone help me out?” He was directed to one of the network engineers: “Normally, I just go and ask <network engineer>. I do that even before going through the documentation. 99% of the time, he knows what to do and I trust him.”

When trying to get access to a development infrastructure, the researcher asked the lead SWE: “I need to access the CA server to test this feature. How do I get access?” Lead SWE: “You should go ask SWE1. He just cleaned up the access list for the CSF thing.”

When the lead SWE was asked the details of an existing script: “Full disclosure, I have no idea how that script works. <Former employee> implemented it and no one has had to make changes till now. But <support engineer> should provide you more information. They are the ones who use it.” In this case, although the SME is no longer within the company, the workflow established through the use of the automation script still remained and the next most knowledgeable person took responsibility of it.

Our data shows that the roles of SMEs, the knowledge on each dimension, and the preferred set of practices were not static but were developed and evolved within the development environment. When there were multiple potential SMEs on a topic, the responsibility could be passed on to the others as well. In some cases this also led to more official transfers of duties within the team. For example, when dealing with customer issues, the lead SWE was pulled into multiple meetings between the customer support team and the clients. Overwhelmed by the work, the team internally discussed the possibility of having another SWE who was working on the problem module for the
past months to take over some of the client discussions, with some guidance from the lead SWE. After reaching an agreement, this was then communicated to the management for future meetings.

4.5.2 Establishment of Preferred Practices

The development team tended to have established preferences for activities that were carried out repeatedly. We observed team preferences for coding styles, debugging techniques, code reviews, ways of dealing with the IT department, use of scripts/tools for tasks, etc. We also observe that these preferred practices were usually tried and tested approaches of doing things within the team and were communicated to other SWEs in the team as needed. For example, preferred coding styles were communicated through the code itself while any unwarranted deviations were communicated through code reviews and reverted back to the preferred way. Any changes made to improve the existing style were also communicated through code and code review. Other preferences could be communicated mainly through discussions between the SWEs whether in a one-on-one or group setting. Usually an SWE sought help from the group using language like “Got a second for a rubber ducky?”, “Can I borrow some of your time <SWE>?”. SWEs were encouraged to hold these discussions in the group chat as there could be more “eyes” on the problem and the solutions could be reached more quickly. These discussions also allowed for the preferred practices to evolve and improve as issues or better options were identified.

These preferred practices became a part of the group knowledge and tended to stick through generations of employees. In such cases some of the in-depth knowledge might be lost with the employee leaving but the preferred practices continued.

- “That is a script that <former employee> developed and we still use it.”
“That playbook was written by a former employee. I know what it does but I am not sure if it uses this script internally. I would have to go read through the code but it gets the job done.”

4.5.3 A Situated Learning Environment

Through analysis of the field notes we find that the roles of “SME” and “learner”, assumed by different SWEs for different dimensions of knowledge, drove a learning cycle within the team. This interactive activity of learning was the core process through which preferred practices were established within the team.

The pattern of learning observed here is not new. The concept of learning, not through a teacher/learner dyad, but as a situated activity where a learner not only acquires knowledge from the experts (“old-timers”) and their peers but does so while participating and contributing in a community of practice is referred to as situated learning [62, 85, 115]. Learning, in this view, is not simply a process of transfer or assimilation of knowledge from the expert (SME) to learners (SWEs), but rather a generative process where each “reproduction cycle” from “learner” to “old-timer” leaves a trace in the community of practice, in both its social structure and physical, linguistic and symbolic artifacts.

The development team is a dynamic situated learning environment with a wide range of knowledge to be acquired and mastered. Based on the dimensions of knowledge under consideration, SWEs simultaneously perform multiple roles of learning practitioner, aspiring expert, status subordinate, or sole responsible agent [62]. The everyday activity of software development provided situated opportunities to learn, defining the “learning curriculum” for the task that SWEs were
performing. As an SWE sought to learn from the team, different SWEs enacted different roles to drive a learning cycle to reproduce the existing culture of practice.

- “Are you guys available for a zoom to discuss the DNS cache changes for the data viz stuff?”
- “Alright type gurus. I’m trying to make an interface that is a Map between two sets of constants. I’m not allowed to do what I posted above. Suggestions? . . .”

Contradictions also arise as a part of this interactive social process as learners start to contribute. Working on resolutions to these contradictions leads to a renewed practice in the community, i.e., preferred practices are established and evolved as SWEs go through the learning cycle.

In this vein, creating a secure development culture is the process of making secure coding practices into the preferred practices of the development team. Thus, facilitating situated learning regarding security within the development team, is key.

4.5.4 The Learning Cycle

Figure 4.1 shows the interactions of an SWE with the development team as he/she progressed from the role of a learner to an SME. At any given time, an SWE could assume different roles for different dimensions of knowledge. For example, the SecSWE could be an SME on certain secure coding practices, but at the same time a learner on some technical details about a particular aspect of the product. External resources were accessible at anytime throughout the process of learning. We first describe the different roles an SWE could assume in this learning cycle.
• **Learner**: An SWE started out as a learner acquiring knowledge, the preferred set of practices, from the team. Learners also looked to external resources on their own, especially for a completely new aspect on which there was no existing knowledge in the team yet. Such acquisition of knowledge only made a difference in the team’s practice when the SWE applied the knowledge in the practice, whereby he/she started to contribute to the team’s knowledge and progressed in the path of professional growth.

• **Knowledgeable SWE**: The application of acquired knowledge in the context of daily practice by the SWE served an important purpose — it provided the basis to have contextual discussions whereby the SWE was able to make contributions to the group. The resulting iterations of
the interactive learning process led to a convergence in understanding of the knowledge within
the group, and thereby the establishment of preferred practices.

- **SME**: When the learning cycle reached saturation and no new contributions were made to the
  community of practice, the SWE was able to assume the role of SME. In terms of legitimate
  peripheral participation [62], the SWE had reached full participation for that particular di-
  mension of knowledge space. The learning cycle then continued for that dimension with other
  SWEs filling in the role of learner and knowledgeable SWE.

  We find that an effective learning cycle went through the following stages to create, maintain,
  and grow the preferred practices through multiple generations of employees.

1. **Acquisition**: The most accessible and credible source for a learner was the SME on the topic of
   interest. They provided access to the current culture of practice to the new learner. The level
   of knowledge available in the team varied depending on factors such as education, prior expe-
   rience, applicability in the daily work, and so on. When the knowledge within the team was
   sufficient, the learning cycle simply reinforced the current preferred practices, as new learners
   continued buying into it. In case of insufficient expertise within the team, a new knowledge
   requirement was created which led to individual/group research on the topic through external
   resources. This could also be facilitated by a new member joining the team who possessed the
   lacking knowledge.

   - **Security Implication**: the expertise levels of the SMEs on security within the team deter-
     mine the team’s preferred practice in secure coding.
2. **Application:** Acquired knowledge needed to be applied in daily practice to drive the learning process. This was a critical step in the learning cycle; without application in daily work, the knowledge was limited to the individual SWE and never became a part of the preferred practice. On the other hand, applicability led to both individual and team growth as the applied knowledge was immediately shared to the peers through development activities like scrum meetings, design/implementation discussions, code review, testing, documentation, and so on. This provided two important driving forces that helped propagate the learning cycle: (1) a shared motivation to solve problems, and (2) the shared context of the work practices which everyone was aware of. These facilitated bi-directional discussions as opposed to a teacher-student scenario as is often perceived as how transfer of knowledge happens.

- **Security Implication:** SWEs’ security knowledge, like all other knowledge, needs to be grown with application. This works well for security, since the best time to apply security knowledge is when the code is being written (as opposed to applying security knowledge to fix vulnerabilities later on).

3. **Contributions:** As SWEs put knowledge into practice, they were able to contribute to the group based on their experience and findings from daily practice. This knowledge exchange through application led to the growth and evolution of the whole team with the increase in existing knowledge on the topic. When there was an established/preferred/agreed upon knowledge base on that given topic, the knowledge in the group reached a level of saturation, and it became a part of the preferred practices.

- **Security Implication:** When security becomes part of the preferred practice, all SWEs in the team will be security aware while writing new code. We observe that the first
successful step towards implementing an effective S-SDLC and creating a security culture in the development team was the rise of security-aware SWEs. After all, if the SWEs are capable of writing secure code, it will make a real change in the final products’ security. As was pointed out in prior literatures, fixing security bugs retroactively is costly and often encounters resistance from the development team [45, 79]. Companies would be better off to prevent, as much as possible, security vulnerabilities from being introduced in the first place.

4.6 Revisiting the Shift towards Security

We now identify the key enablers of the positive shift in secure development we observed during the fieldwork.

4.6.1 Setting Security as a Goal

Past experience suggested that management support was an important factor in the successful implementation of S-SDLC [46]. Our observations supported this. Due to cost in terms of time and efforts required, security was easily perceived as an “obstruction” to the daily practice of SWEs and hence the learning cycle for this dimension of knowledge did not evolve at first. We found that management played an important role to set security as a goal, making it a part of the deliverable. Doing so ensures that security knowledge is not something that overwhelms SWEs but simply applicable to daily practice, eliminating a critical barrier to drive the learning cycle.
4.6.2 Applying Security Knowledge in Context

Having the management directive and support for secure coding was necessary but not sufficient to eliminate the barrier to adopting secure development practices. Secure coding requires a wide range of security knowledge, and providing adequate education and awareness was pointed out as one major challenge in successfully implementing S-SDLC [46]. While the company provided SWEs virtual training for secure coding and there were also various guidelines and wiki pages the SWEs could access, applying the acquired knowledge in everyday work required expertise in both security and the contextual knowledge of the existing code base. Finding this connection was challenging for SWEs. Without application, the knowledge gained from training was at best internalized by individual SWEs, but remained detached from their daily practice. The SWEs effectively considered security-related tasks as secondary tasks, separate from their primary practice. To overcome this, a bottom up support was also needed to make real progress. In our fieldwork we found that such bottom up support happened through the learning cycle identified in the previous section. The threat modeling and associated security scrum poker meetings, which involved all SWEs, provided the opportunity for the SecSWE to put the relevant security knowledge into the concrete context of the software being built. This started the learning dynamics that enabled all SWEs to progress on the “security dimension.”

4.6.3 The Role of Security Advocates

The work of SecSWE played an important role in facilitating the learning cycle and making security into part of the development team’s preferred practices. SecSWE was a “security advocate” [42] even before the management pushed to implement S-SDLC. He worked in the devel-
opment team, and was also assigned to be a part of the virtual security team, providing additional security resources. Analyzing our data, we find that this structure added more value to security advocacy, making other SWEs more receptive to his advice as they started to consider it “part of his job.” Working on the same team provided an important factor in demonstrating the applicability of the security knowledge in the context of the daily practice. This facilitated SecSWE to contribute knowledge as applicable to daily practice, helping to drive a productive learning cycle, which was beneficial to both the rest of the team and SecSWE himself. Through this interactive learning process, SecSWE was able to better understand the necessary details of the product which allowed him to apply his security knowledge in a more context-aware manner. Further iterations of this learning cycle led to more security-aware SWEs in the team.

4.7 Limitations

Our work is limited by a few factors. First, our findings are based on the fieldwork data collected by a single researcher. Although the researcher had prior training and experience in conducting participant observation research, the collected data are shaped by the researcher’s positionality (his age, gender, position in the company, and so forth). For example, the researcher did not have as many interactions with customer service and upper management because of his position in the company. However, the researcher did build an overall understanding of the company during the research, and the results were extensively discussed with the broader research group during analysis to better account for any inherent biases in the data. Second, our findings are based on the observations of a single company with a particular size and structure. Although we believe the development team is representative of one in a mid-sized software development company, the specific challenges of adopting secure development practices and how they were/were not overcome
may not be directly applicable and generalizable to every company. As such, the model of how a culture is developed within a software development team might not be comprehensive. Nevertheless, during data analysis, the team paid particular attention to how results related to common problems faced in security and software development to ensure that the findings could be relevant to other companies.

4.8 Recommendations for Companies

Our findings suggest a potentially useful strategy for a small to medium sized company. Having a security expert as a part of the development team, participating and advocating for security at every stage of the development process, is beneficial in starting a security culture. This not only helps cultivate security-aware developers, but also helps the security expert identify security issues and collectively converge to secure practices that are best suited for the project at hand. Development of the relevant security knowledge in conjunction with the regular software development skills promotes secure coding practices which, overtime, become a part of the team culture. Our research also observed the effect management had in facilitating the positive shift. Even though the initial efforts focused on the compliance tickets were not effective, the fact that management made security an explicit goal provided the opportunity for the security advocates to experiment different strategies that eventually led to positive results.

4.9 Related Work

Our fieldwork was conducted in the backdrop of the company starting to implement a secure development lifecycle, a concept first articulated by Howard and Lipner [46]. This seminal work highlighted the importance of education and training in creating S-SDLC. Our findings further indi-
cate that understanding the learning dynamics, in particular how preferred practices are established within a software development team through the situated learning framework, can be instrumental in creating positive changes in secure development.

There is a long line of study on developers' role in software security. Some used psychological techniques [77]. Others used surveys and interviews [4, 8, 39, 70, 89, 120] as well as study of code artifacts [4, 70]. More recently, researchers have used secure coding competitions [86, 110] and controlled experiments [3, 4, 5, 37, 71] to study the problem. Our work is unique in that we use long-term participant observation conducted in a real company. The longitudinal study based on real-world observations allows us to obtain deep insights that are otherwise hard to come out through snapshots-in-time study or self-reported data.

Palombo et al. [79] used ethnographic methods to study a software company’s secure development processes. The authors indicated that a co-creation model where security experts working inside the development team could produce positive changes in secure development processes. Our work revealed the role of learning dynamics in pushing for positive shift in adopting secure development processes. The role situated learning plays in starting a secure development culture is consistent with the co-creation model.

The SecSWE in our study can be viewed as a “security advocate”, which has been extensively discussed in recent studies [41, 42, 43]. Our findings on the role of team culture in security awareness of SWEs echoes that from prior studies. Assal and Chiasson [8] explored how security best practices are integrated into the software development lifecycles and found that company culture is an influential factor in adoption of security practices. Haney et al. [44] carried out in-depth interviews to understand cryptographic development and testing practices in organizations and found that
rigorous secure development and testing practices are guided by a strong security culture within organizations. They also identify that security experts within the team are critical influences in the security culture of an organization and in supporting less-experienced personnel. Our findings confirm the important role security advocates play in starting a security culture, and further provide guidance on how to make security advocates’ work effective, through understanding the underlying learning dynamics that drive the formation of a development team’s culture.

There are also past work that examined the effect of learning from experience in software development [30], and work that analyzed open-source software development using the situated learning framework [26]. Our work focuses on secure development, and our research findings are consistent with these earlier works which focused on learning’s role in software development in general.

4.10 Conclusion

We present an ethnographic study of secure development processes in a software company. Our research was able to observe the unfolding of implementing a secure development life cycle in the company. Data analysis shows that a positive shift in developers’ security awareness resulted from underlying situated learning dynamics, where security knowledge is constantly applied in the concrete work of the development team. This process drives the establishment of secure coding practices as the preferred practices of the team, essentially establishing a secure development culture. We find that a security expert working within the development team could be instrumental in driving this positive shift.
Chapter 5: Conclusion

The role of human-factors is critical in improving the security of any system. The reasons of adoption, or failure there of, is usually very nuanced and difficult to understand without gaining first hand experience. This thesis advocates for the use of long term participant observation research in order to understand the underlying subtleties of security issues by presenting the findings from two longitudinal studies.

The study of a transportation management center found that the existence of silos of different engineering disciplines contributes to the lack of holistic understanding of the complex transportation ecosystem. We present a systematization framework which facilitates the communication of security knowledge and helps breakdown these silos.

The study of a software development company found that software development team’s culture is defined by a set of preferred practices which is developed over a period of time through the underlying interactive learning dynamics within the team. We provide recommendations regarding the role of management and security advocates within the team in leveraging the learning dynamics to give rise to security-aware development and promote secure preferred practices within the team as a first step towards developing a security culture.
References


[49] ITS Cabinet Standard. ITS.


[75] NTCIP. NTCIP Published Standards.


[103] USDOT. Intelligent Transportation Systems - Connected Vehicle Pilot Deployment Program.


[105] USDOT. Intelligent Transportation Systems - Connected Vehicle Pilot Deployment Program (Tampa).


Appendix A: Codebook

Coding of the fieldnote data followed the general guidelines of inductive approach [97] and grounded theory [10]. It was an iterative process and started after the first three months of the fieldwork. The research team held weekly meetings to discuss and reflect on the data collected. Themes and patterns emerged from those discussions and various codes were used to tag the content in the raw fieldnote. Below is the list of codes used.

- Bug discovery
- Bug discovery:internal
- Communication issue
- Compliance:asvs
- Compliance:csf
- Compliance:csf:thirdparty
- Compliance:encryption
- Compliance:phishing
- Cross product issue
- Customer pressure
• Feature pressure
• Forgotten issue
• Ignored issue
• Infra
• Infra:legacy
• Infra:security
• Learn
• Learn:best practice
• Learn:figure out
• Learn:peer programming
• Learn:review
• Policy change
• Preferred practice:code
• Preferred practice:support
• Preferred practice:workflow
• Remote work issues
• SME
• SME:handover
• SME:new

• Secure development

• Security-aware

• Threat modeling

• Threat modeling:dread

• Training

• Workflow change
Appendix B: Copyright Permissions

The following copyright permission is for the use of material in Chapter 4 which have appeared in [101].

AUTHORIZATION AND LICENSE TO PUBLISH

A major mission of the USENIX Association (“USENIX”) is to provide for the creation and dissemination of new knowledge. Unlike most publishers, USENIX permits authors to retain their ownership of the copyrights in their works. We therefore require each author to provide written consent giving USENIX the right to publish his or her work. Therefore, the parties agree as follows:

1. Grant of Rights to Publish. You hereby grant to The USENIX Association (“USENIX”), the right to publish and re-publish the paper provisionally entitled

   Paper Title: An Analysis of the Role of Situated Learning in Starting a Security Culture in a Software Company

   (the “Paper”) electronically and in audiovisual formats (the “Rights”). USENIX’s rights are non-exclusive, worldwide, perpetual and irrevocable, but also non-sublicensable and non-transferable. Among the rights you hereby grant to USENIX is the right to be the first publisher of the Paper. USENIX initially plans to publish the Paper in the following publication: Proceedings of the Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021) (the “Original Publication”) and to post the files indefinitely for display on the USENIX website, but may also otherwise disseminate the Paper, as well as the oral presentation of the Paper, on media, as a live broadcast, and in any other electronic or audiovisual format chosen by USENIX. Copyrights are retained by the authors.

2. Posting. You retain all rights in the Paper subject to the non-exclusive rights granted to USENIX in Section 1 above.

3. Notice Request. USENIX will include the following notice on the first page of the Original Publication: “Copyright to the individual works is retained by the author[s].”

4. Subsequent Publication by Author. You must include a notice of original publication by USENIX in any subsequent publication of the Paper or a revised version of the Paper.

5. Warranties. You warrant that:
   a. You are the sole author and owner of the Paper and the rights pertaining thereto or, if the Paper is a work of joint authorship, you are signing as to your legal rights as joint author to give USENIX permission to publish the Paper. If the Paper is owned by your employer, you warrant that you have obtained permission to enter into this agreement and to grant the rights herein described to USENIX.
   b. USENIX does not need the permission of any other party to exercise the rights granted herein.
   c. The Paper does not contain any defamatory or other unlawful content.
   d. The Paper does not infringe upon the rights of others.
   e. The Paper and its contents are original to you.
   f. The Paper is not currently submitted for publication elsewhere and has not been previously published elsewhere.
   g. If the Paper contains any materials protected by copyright law, written permission from the copyright holders has been obtained, copies are attached to this document, and proper credit has been given in the Paper. If using a Creative Commons license, you have fulfilled all licensing requirements.

6. Right to Withdraw Access. USENIX reserves the right to remove content and publications under its control from public access at any time for any reason. Occasionally, USENIX receives notices alleging that the content it publishes may infringe third-party rights or may otherwise violate some law or regulation. USENIX generally does not take a position on the merits of such allegations. Rather, USENIX will inform the author or speaker of the allegations, refer the complainant to the author or speaker, and may remove the identified content or publication from public access. USENIX may restore the material, at its discretion, when it deems appropriate. The author or speaker will be solely responsible for resolving the complaint. USENIX also may send the notice containing the allegations to Lumen (lumendatabase.org) and/or similar organizations.

USENIX Code of Conduct Guidelines

Speakers are responsible for the content of their presentations, but USENIX requests that speakers be cognizant of potentially offensive actions, language, or imagery, and that they consider whether it is necessary to convey their message. If they decide to include it, USENIX asks that they warn the audience, at the beginning of the talk, and provide them with the opportunity to leave the room to avoid seeing or hearing the material.
Appendix C: Institutional Review Board Authorization

The following is the IRB approval for the ethnographic study.

June 30, 2021

Daniel Lende
Department of Anthropology
4202 E. Fowler Avenue, SOC 107
Tampa, FL 33620

Dear Dr. Daniel Lende:

On 6/30/2021, the IRB reviewed and approved the following protocol:

<table>
<thead>
<tr>
<th>Application Type:</th>
<th>Continuing Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRB ID:</td>
<td>Pro00036117 CR000002</td>
</tr>
<tr>
<td>Review Type:</td>
<td>Expedited 7</td>
</tr>
<tr>
<td>Funding:</td>
<td>National Science Foundation</td>
</tr>
<tr>
<td>IND, IDE, or HDE:</td>
<td>None</td>
</tr>
</tbody>
</table>

Approved Protocol and Consent(s)/Assent(s):
- IRB Protocol Version 1 June 2018.docx;
- Verbal Consent Form Version 2.pdf;

Approved study documents can be found under the 'Documents' tab in the main study workspace. Use the stamped consent found under the 'Last Finalized' column under the 'Documents' tab.

The IRB approved the protocol from 7/24/2021 to 7/24/2022. Within 45 days of 7/24/2022, submit a continuing review/study closure request in BullsIRB by clicking Create Modification/CR.

If continuing review approval is not granted before the expiration date of 7/24/2022, approval of this protocol expires on that date.

In conducting this protocol you are required to follow the requirements listed in the INVESTIGATOR MANUAL (HRP-103).

Sincerely,
Gina Larsen
IRB Manager