CTI-FOR-CPS: SECURING CYBER-PHYSICAL SYSTEMS VIA ADVANCED CYBER THREAT INTELLIGENCE METHODS

A Dissertation

by

EFRÉN DARÍO LÓPEZ MORALES

BS, Universidad Autónoma de Guadalajara, 2012
MS, Arizona State University, 2020

Submitted in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in

GEOSPATIAL COMPUTER SCIENCE

Texas A&M University-Corpus Christi
Corpus Christi, Texas

August 2025

August 2025

CTI-FOR-CPS: SECURING CYBER-PHYSICAL SYSTEMS VIA ADVANCED CYBER THREAT INTELLIGENCE METHODS

A Dissertation

by

EFRÉN DARÍO LÓPEZ MORALES

This dissertation meets the standards for scope and quality of
Texas A&M University-Corpus Christi and is hereby approved.

Carlos Rubio-Medrano, PhD
Chair

Jose Baca, PhD
Committee Member

Tianxing Chu, PhD
Committee Member

Mehdi Sookhak, PhD
Committee Member

Catherine Schumann, PhD
Graduate Faculty Representative

August 2025

ABSTRACT

Many services that make our modern society work are possible thanks to Cyber-Physical Systems (CPS). These services include electricity generation and distribution (Industrial Control Systems (ICS)), Global Positioning System (GPS) and remote sensing (satellites), and transportation (connected and autonomous vehicles). Many CPS rely on and buttress geospatial data such as GPS location, satellite imagery, and light detection and ranging (LiDAR) point clouds. Due to their importance, CPS have been the target of cyberattacks that aim to disrupt our society. One of the tools that we can leverage to protect CPS is cyber threat intelligence (CTI). CTI is threat information aggregated, transformed, analyzed, interpreted, or enriched to understand a threat actor's motives, targets, and attack behaviors. However, current CTI on CPS is limited as current methods cannot collect and analyze data that can be converted into useful CTI thus leaving CPS exposed to further attacks.

This dissertation addresses this problem by developing new methods that advance the state of the art in CTI processing and collection phases, specifically for cyber-physical systems which we call *CTI-for-CPS*. We close the above research gap by introducing three novel research projects that push the boundaries of CTI-for-CPS, specifically, industrial control systems (ICS), space systems (SS), and connected and autonomous vehicles (CAV). The first contribution is a novel threat taxonomy for programmable logic controllers (PLCs) used in ICS which are well-established CPS, to improve how we categorize and analyze threats in ICS. The second contribution is a satellite honeypot, a novel approach that allows us to gather empirical CTI data on threat actors' techniques targeting satellites. The third and final contribution is a CAV cybersecurity sandbox. This sandbox simulation allows us to test cyberattacks on one or multiple CAVs to collect raw CTI data that can later be analyzed, marking a significant step forward in our understanding of cyber threats targeting CAVs.

These three contributions introduced in this dissertation present novel approaches to collect, aggregate, and analyze data to produce valuable CTI-for-CPS. Failing to advance the state of the

art of CTI-for-CPS risks leaving CPS exposed as we will not understand the latest techniques that adversaries use to target CPS, hindering our ability to develop effective countermeasures. Without proper countermeasures in place, the security of our critical infrastructure is at risk of cyberattacks.

DEDICATION

This dissertation is dedicated to my wife, Erandi, and my daughter, Sara. Without their love and support, I would not have been able to complete it.

ACKNOWLEDGEMENTS

I want to express my heartfelt gratitude to my advisor, Dr. Carlos Rubio-Medrano, who has been my mentor since 2019 and who introduced me to the amazing world of scientific research. His guidance and support have been with me since the very first day of my Ph.D. and throughout every stage of it. I am especially grateful for his patience, constant encouragement, and the high standards he set so that I would become a better researcher and person.

I am grateful to all of my committee members, Dr. Baca, Dr. Chu, and Dr. Sookhak, for sharing their expert knowledge in their respective research areas, which greatly enriched my Ph.D. education. Their insightful feedback and steadfast support were instrumental in shaping and improving this dissertation.

I thank Dr. Álvaro Cárdenas at the University of California, Santa Cruz, and Dr. Ali Abbasi at the CISPA Helmholtz Center for Information Security in Germany for being outstanding mentors and collaborators. I am deeply grateful for their generosity in hosting me at their respective institutions, for their guidance and teaching, and for helping me broaden my knowledge across a range of research topics in which they are truly world-class experts.

Last but not least, I thank my labmates Jacob, Syed, Ruben, and Azim for their friendship, support, interesting and deep discussions, memorable trips, and all the great times we spent working on research and enjoying life beyond academia.

TABLE OF CONTENTS

Page

ABSTRACT iv

DEDICATION vi

ACKNOWLEDGEMENTS vii

TABLE OF CONTENTS viii

LIST OF FIGURES xii

LIST OF TABLES xiv

CHAPTER I: INTRODUCTION 1

1.1 Problem Statement 3

1.2 Proposed Work 3

1.3 Dissertation Outline 4

CHAPTER II: BACKGROUND 5

2.1 Cyber-Physical Systems 5

2.1.1 Industrial Control Systems (ICS) 5

2.1.2 Satellite Systems 6

2.1.3 Connected and Autonomous Vehicles (CAVs) 6

2.2 Cyber Threat Intelligence (CTI) 6

2.2.1 Data Collection (Phase 2) 8

2.2.2 Information Processing (Phase 3) 8

2.2.3 Cyber Threat Intelligence of Cyber-Physical Systems 8

CHAPTER III: THREAT TAXONOMY FOR INDUSTRIAL CONTROL SYSTEMS 10

3.1 Introduction 10

3.2 Background 12

3.2.1 PLC System Environment 12

3.2.2 PLC Basic Components 13

3.2.3 PLC Communication Protocols 15

3.2.4 SoftPLCs 16

3.3 Research Questions and Methodology . . . 16
3.3.1 Overview of Research Questions . . . 16
3.3.2 Knowledge Systematization Methodology . . . 17
3.4 Classification and Evaluation Criteria . . . 20
3.4.1 ICS Threat Taxonomy . . . 20
3.4.2 Attack Classification . . . 21
3.4.3 Attack Complexity . . . 22
3.4.4 Attack Impact . . . 24
3.4.5 Defense Classification . . . 24
3.4.6 Defense Deployability . . . 25
3.4.7 Defense Robustness . . . 25
3.5 Overview of Attacks . . . 26
3.6 Overview of Defenses . . . 31
3.7 Research Gaps . . . 33
3.8 Discussion . . . 38
3.9 Related Work . . . 40
3.10 Chapter Conclusion . . . 40
3.10.1 Scientific and Grey Literature Resources . . . 41
3.10.2 MITRE Technique and Sub-Technique Model Example . . . 41
3.10.3 PLCs' Larger Context and Underlying Architecture . . . 42
3.10.4 PLC Memory Blocks . . . 42
3.10.5 PLC Built-in Security Features . . . 44
3.10.6 Real-World PLC Attacks . . . 46
3.10.7 Excerpt of the ICS^2 Matrix . . . 47
CHAPTER IV: SATELLITE HONEYPOT . . . 49
4.1 Introduction . . . 49
4.2 Background . . . 52

4.2.1 Types of Honeypots . . . 53
4.2.2 Honeypot's State of the Art . . . 53
4.2.3 Anatomy of a Satellite Mission . . . 54
4.2.4 Satellite Architecture . . . 57
4.2.5 Small Satellite Protocol Ecosystems . . . 58
4.2.6 Space Systems' Tactics, Techniques and Procedures (TTPs) . . . 60
4.3 Threat Model . . . 60
4.4 HoneySat High-level Design . . . 60
4.4.1 HoneySat's Design Objectives . . . 60
4.4.2 HoneySat's Design Principles . . . 61
4.4.3 Theory of Operation . . . 62
4.5 Evaluation . . . 63
4.5.1 Experimental Questions . . . 63
4.5.2 TTP Interaction Experiment . . . 64
4.5.3 SmallSat Operators Survey . . . 66
4.5.3.1 Survey Structure . . . 67
4.5.3.2 Participants . . . 68
4.5.3.3 Methodology and Key Results . . . 68
4.5.4 Internet Interaction Experiment . . . 70
4.5.5 Case Study: Generic CCSDS Mission Honeypot . . . 72
4.6 Discussion and Future Work . . . 73
4.7 Chapter Conclusion . . . 74
4.7.1 Interaction Sequences and Exploits . . . 74
4.7.2 Deployment and Security Hardening Implementation . . . 76
CHAPTER V: CONNECTED AND AUTONOMOUS VEHICLES SANDBOX . . . 78
5.1 Introduction . . . 78
5.2 Background . . . 81

5.2.1 Sandboxing 81
5.2.2 Vehicle-to-Cloud (V2C) Infrastructure 82
5.2.3 MQTT Protocol 83
5.2.4 Connected and Autonomous Vehicles' Tactics, Techniques and Procedures (TTPs) 84
5.2.5 Established CAV Threat Models 85
5.3 Our Focus: Vehicle-to-Cloud Threat Model 86
5.4 *Grand Hack Auto*: A Threat Analysis Framework for V2C 87
5.4.1 Design Objectives 88
5.4.2 Implementation 89
5.5 *HYDRA*: A Synthetic Malware for V2C 90
5.5.1 Design Objectives 91
5.5.2 Implementation 92
5.6 Evaluation 95
5.6.1 Experimental Questions 95
5.6.2 Physical Impact Experiments 96
5.6.3 Systematic Threat Analysis 99
5.6.4 Cloud Integration Experiment 101
5.7 Discussion 104
5.7.1 Related Work 104
5.7.2 Limitations 105
5.8 Ethical Considerations 105
5.9 Chapter Conclusion 106
CHAPTER VI: CONCLUSION AND FUTURE WORK 108
REFERENCES 110

LIST OF FIGURES

Page

2.1 Cyber Threat Intelligence Life Cycle. This dissertation focuses on Phases 2 and 3 (blue) 7

3.1 A Generalized PLC Architecture. Based on [1, 2]. 14

3.2 Our systematization methodology. Based on [3]. 18

3.3 Literature focused on PLC Security per Year. 18

3.4 PLC-centered Threat Model. Based on [4]. 22

3.5 Comparison of Research Share vs Market Share. 34

3.6 The 15 Most Common PLC Models. 34

3.7 Defense vs Attack Methods per Target Component. 35

3.8 Attack Methods according to their Tactic. 36

3.9 Defense Methods per Mitigation Category. 37

3.10 Papers evaluated using SoftPLCs vs HardPLCs. 39

3.11 Example of an ICS process with PLCs in Level 1: Controller Network as defined by the Purdue Model. Based on [5, 6]. 44

3.12 Relationship between Critical Infrastructure (CI), Operational Technology (OT), Cyber-Physical Systems (CPS), Industrial Control Systems (ICS), Supervisory Control and Data Acquisition (SCADA), and PLCs. Based on [7]. 45

3.13 Condensed version of the ICS^2 Matrix. It includes nodes from the MITRE ATT&CK for ICS Matrix (Blue), the MITRE ATT&CK Enterprise (Red) and our additions (Yellow). . 48

4.1 The components commonly found in a satellite mission in the context of the space and ground segments. 55

4.2 HoneySat's Theory of Operation. 62

4.3 Architecture of dockerized Generic CSP Honeypot. 76

4.4 Screenshot of the YAMCS web interface, displaying a TM packet, returned from the Generic CCSDS Mission Honeypot. 76

5.1 Basic MQTT network configuration for CAVs. In this example, a vehicle's MQTT client publishes some sensor data via a topic to the broker and then a cloud MQTT client subscribes to this topic to get the sensor data. In the same way the cloud MQTT client publishes some entertainment data to the broker and the vehicle's MQTT client subscribes to the entertainment topic to get the corresponding data. 82
5.2 The two threat model scenarios considered for *Grand Hack Auto*. ① the target CAV communicates with a legitimate cloud provider that has been compromised. ② the target CAV communicates with a malicious MQTT client running on a C2 infrastructure controlled by an attacker. 87
5.3 *Grand Hack Auto* framework architecture. The framework leverages the CARLA simulator ① which generates a world where the vehicles move and each vehicle has sensors (e.g. GNSS) attached to them. Each vehicle uses the CARLA API ② to send and receive data to its own MQTT client ③. All the vehicles' MQTT clients connect to a malicious broker which can be local or cloud-based ④ and this broker is connected to the malware ⑤ that exploits the multiple features of our analysis framework. 88
5.4 *HYDRA* synthetic malware's architectural design. 91
5.5 Remote command attack example that showcases the effects of the malicious MQTT command on the speed and acceleration of a CAV. 98
5.6 2D trajectory plot of two vehicles in a coordinated collision scenario. 99
5.7 Latency comparison between the local network broker and two cloud brokers, HiveMQ and Mosquitto. 103

LIST OF TABLES

Page

1.1 Relationship between dissertation chapters and papers 2

3.1 A Summary of PLC Attack and Defense Methods (Network, OS and Runtime Components). **D**=DNP3, **G**=GOOSE, **S7**=S7COMM, **I**=IEC 104, **UA**=OPC-UA, **P**=Profinet, **E**=EtherNet/IP, **M**=Modbus TCP, **F**=FINS, **S**=Syslog, **B**=Beckhoff, **BP**=Backpane, **R**=SoC Register, **RT**=Runtime, **O**=OpenPLC, **C**=CODESYS / CODESYS upload protocol, **N**=Not Specified, **DE**=Detection, **PR**=Prevention, **RE**=Recovery 27

3.2 A Summary of PLC Attack and Defense Methods (Control Logic, Firmware, I/O, Memory and CPU Components). **S7**=S7COMM, **UA**=OPC-UA, **P**=Profinet, **E**=EtherNet/IP, **M**=Modbus TCP, **T**=TriStation, **SM**=SoMachine, **U**=USB Port, **MC**=Memory Card, **X**=Others, **O**=OpenPLC, **C**=CODESYS, **N**=Not Specified, **DE**=Detection, **PR**=Prevention, **RE**=Recovery 28

3.3 Proposed technique definition of Exfiltration over Covert Channel according to MITRE ATT&CK: Design and Philosophy [8] 43

3.4 Attack Vectors per Access Level. 46

4.1 Comparison of Existing Honeypots and HoneySat. 54

4.2 Tactics and techniques supported by HoneySat compared to the ones included in the SPACE-SHIELD matrix that are applicable to virtual honeypots. 66

4.3 Summary of questions in Section 4.5.3, *satellite honeypot operation tasks* of the survey (48 total). 67

4.4 Exposed Telnet interactions received. 71

4.5 Tactics, Techniques and Procedures' Interaction Experimental Exploits. 75

5.1 Comparison of Threat Models in Connected and Autonomous Vehicles (CAVs) 83

5.2 Tactics and techniques supported by *Grand Hack Auto* compared to the ones included in the Automotive Threat Matrix (ATM). 100

5.3 Evaluation of TTPs from the Automotive Threat Matrix using the CAV Sandbox 102

5.4 Comparison of Related Work to Grand Hack Auto 104

CHAPTER I: INTRODUCTION

The critical infrastructure on which our society depends upon, such as the electrical grid, is buttressed by Cyber-Physical Systems (CPS). CPS are systems that integrate computational and physical components in order to carry out a physical process. CPS allow complex physical processes to become more efficient as they integrate computer networks and novel features such as data analytics and cloud connectivity [9]. Examples of CPS include ICS used for food production, Space Systems used for Earth's remote sensing, and connected and autonomous vehicles (CAVs) used for transportation.

However, along with CPS' capabilities come cybersecurity risks, specifically, the risk of cyber threats and attacks that can compromise these systems' safety, reliability, and integrity. A successful cyberattack on a CPS has the unique characteristic of physically endangering real-world assets such as nuclear facilities in the case of ICS, roads in the case of CAVs, and satellite imagery availability in the case of satellites. Compromising such assets might result in millions of dollars' worth of damage or, even worse, the loss of human life. One of the most infamous instances of a successful CPS cyberattack is the 2011 Stuxnet malware attack. Stuxnet sabotaged centrifuges used to enrich nuclear fuel, causing significant physical damage [10]. Since Stuxnet, several other CPS cyberattacks have occurred; for example, in 2017, researchers gained control over vulnerable features of Tesla and BMW autonomous cars through Wi-Fi and browser vulnerabilities using previously unknown exploits [11].

One key component of an effective cybersecurity strategy for CPS is cyber threat intelligence (CTI) [12, 13]. CTI refers to the process of collecting, analyzing, and disseminating information about cyber threats and adversaries to inform defensive measures and decision-making [14]. By leveraging CTI, organizations can gain insights into emerging threats, adversary tactics, and vulnerabilities specific to various computer systems, including CPS, enabling them to proactively mitigate risks and enhance their security.

This dissertation introduces *CTI-for-CPS*, a methodology to advance the state of the art of the collection and processing phases of the cyber threat intelligence lifecycle, specifically on

ICS, satellites, CAVs, which are crucial examples of CPS. The problem to be addressed in this dissertation is developing new methods that advance the state of the art in cyber threat intelligence processing and collection phases for cyber-physical systems. Although there has been previous efforts to improve CTI-for-CPS [15], current methods for processing and collecting CTI for CPS are limited [16]. For example, there is no satellite honeypot that allows us to collect raw cyber threat intelligence data. Current processing and collecting methods are focused on commodity computers.

Chapter #	**Publication**	**Remarks**
1	–	Introduction and dissertation objectives and contributions
2	–	Background
3	López-Morales, Efrén, Ulysse Planta, Carlos Rubio-Medrano, Ali Abbasi, and Alvaro A. Cardenas. "SoK: Security of Programmable Logic Controllers." In 33rd USENIX Security Symposium (USENIX Security 24), pp. 7103-7122. 2024.	Published
4	López-Morales, Efrén, Ulysse Planta, Gabriele Marra, Carlos González, Jacob Hopkins, Majid Garoosi, Elías Obreque, Carlos Rubio-Medrano, and Ali Abbasi. "HoneySat: A Network-based Satellite Honeypot Framework." arXiv preprint arXiv:2505.24008 (2025).	Under review
5	López-Morales, Efrén, and Carlos Rubio-Medrano. "Grand Hack Auto: A Vehicle-to-Cloud Threat Analysis Framework for Connected and Autonomous Vehicles."	Under review
6	–	Conclusion and future work

Table 1.1
Relationship between dissertation chapters and papers

1.1 Problem Statement

The problem to be addressed in this dissertation is to study and develop new methods that advance the state of the art in cyber threat intelligence processing and collection phases, specifically for cyber-physical systems. Current methods for processing and collecting cyber threat intelligence of cyber-physical systems are limited [12]. For example, no satellite honeypot currently allows us to collect raw cyber threat intelligence data. Current processing and collecting methods are focused on commodity computers. For example, there is ample research done on cyber threat intelligence methods in social media security, cloud computing security, and more [17]. However, these methods are not directly transferable to cyber-physical systems as they have different architectures and purposes. Cyber threat intelligence is crucial for developing countermeasures and security strategies. Failing to develop effective cyber threat intelligence processing and collecting methods for cyber-physical systems would result in poor threat intelligence that would limit our capacity to develop effective countermeasures and security strategies.

1.2 Proposed Work

The main goal of this dissertation is to advance the state-of-the-art CPS CTI. Specifically, we are interested in advancing the two Phases of the CTI lifecycle depicted in Fig. 2.1: data collection (Phase 2) and data processing (Phase 3). To accomplish this goal, we divided our research into three objectives.

1. Develop a novel taxonomy of threats against PLCs and ICS in general. This new taxonomy requires an extensive literature review of the known threats targeting PLCs. It will allow us to better process CTI in Phase 2 of the CTI life cycle. PLCs and ICS are well-established CPS and would give us insight on how to improve CTI for ICS and CPS more broadly. This literature review and taxonomy will help us identify and understand research gaps in the following two objectives.

2. Develop the first satellite honeypot. Honeypots allow us to collect raw threat intelligence data for satellite systems. To develop our honeypot, we leveraged the knowledge gained on

objective 1; specifically, from the literature review, we learned that honeypots are a powerful tool for collecting raw data. This objective is based on the results obtained from Objective 1, specifically, the research gaps learned from the literature review.

3. Finally, develop a CAV security framework. This framework will leverage the sandbox methodology, a sandbox is a testing environment that isolates untested and untrusted code and protects critical resources, such as production servers, from testing changes that could have unintended consequences in the production environment. A CAV sandbox will allow us to test TTPs and gather threat intelligence raw data.

1.3 Dissertation Outline

This dissertation is organized into six chapters. The current chapter, Chapter 1 introduces the research motivation, problem statement, overarching objectives, and key contributions. Chapter 2 introduces the necessary background concepts that will be referenced during the rest of the manuscript. Chapters 3 through 5 each present a major contribution, and are based on the publications produced during the course of the Ph.D. program, shown in Table 1.1. Chapter 3 proposes a taxonomy for securing ICS, providing foundational insights for classifying and assessing attacks and countermeasures. Chapter 4 presents the design, and evaluation of a satellite honeypot system, which was developed to study adversarial behavior in space systems. Chapter 5 introduces a CAV sandbox framework that integrates realistic vehicle simulation with cloud-based threat scenarios. Finally, Chapter 6 concludes the dissertation by summarizing the research findings and describing future directions.

CHAPTER II: BACKGROUND

2.1 Cyber-Physical Systems

CPS include physical and computational components and integrate physical processes with computing and networking capabilities to offer otherwise unavailable levels of automation and connectivity [9]. Examples of CPS include Industrial Control Systems (ICS), Connected and Automated Vehicles (CAV), and satellites. CPS makes our modern society work by enabling critical infrastructure services such as electricity generation and distribution (ICS), Global Positioning System (GPS), remote sensing (satellites), and transportation (CAV). Many CPS rely on sensors and actuators that enable the CPS to sense the physical world around it and control physical equipment that can change the physical environment. These sensors include LiDAR, GPS, and temperature [18] and they feed different types of data to the CPS. For example, LiDAR provides point data that a CAV can use to sense its environment. Satellites use sensors such as GPS to sense its latitude and longitude. The term CPS was coined in late 2006 by the NSF in the United States [19] and has seen copious research since, including security research. Indeed, Presidential Policy Directive 21 recently revamped intelligence sharing on critical infrastructure. However, there is a research gap in the literature regarding CPS cyber threat intelligence [20]. This work will focus on the security of three CPS: industrial control systems (ICS), satellites, and connected and autonomous vehicles (CAVs).

2.1.1 Industrial Control Systems (ICS)

ICS manages and controls critical utilities such as the power grid, water treatment plants, and transportation systems. ICS comprises multiple components, including sensors, actuators, and Programmable Logic Controllers (PLCs). PLCs control industrial processes such as the ones used in water treatment plants by running special programs known as control logic. Control logic reads inputs from sensors and outputs instructions to actuators based on the control logic and input data [21].

2.1.2 Satellite Systems

Satellites are complex CPS designed to withstand outer space conditions and tasked with specific missions. These missions include Earth's remote sensing and GPS location, among others. Satellites' architectures are opaque and varied, but most include the following five subsystems [22]. 1) The Command and Data Handling System (CDHS) manages all satellite functions, such as telecommand parsing. The CDHS works alongside an onboard computer (OBC) and flight software (FS). 2) The Electrical Power System (EPS) manages power generation, supply, and distribution to and from the other satellite subsystems. 3) The communications (COM) subsystem consists of an antenna and a radio which communicates with the ground station. 4) The Attitude Determination and Control System (ADCS) allows the satellite to control its three-dimensional position. This is useful when the satellite needs to point in a specific direction, for example, toward the sun. The ADCS uses sensors and actuators to determine and change the satellite's attitude. 5) The payload is the subsystem used to accomplish the satellite's mission. The payload can vary widely depending on the mission, for example, it can include red, green and blue (RGB) or near-infrared cameras for remote sensing missions.

2.1.3 Connected and Autonomous Vehicles (CAVs)

Connected and Autonomous Vehicles (CAVs) leverage multiple technologies to communicate with nearby vehicles and infrastructure to provide features such as vehicle automation to drive decision-making [23]. CAVs can be self-driving vehicles that use artificial intelligence or computer systems to drive themselves without human operators and are connected via cellular or other networks to send and receive data from other CAVs, transportation infrastructure such as green lights and even pedestrians.

2.2 Cyber Threat Intelligence (CTI)

CTI is the process of collecting, analyzing, and disseminating information about cyber threats and adversaries to inform defensive measures and decision-making. CTI is produced during the CTI life cycle (depicted in Fig. 2.1), which includes five Phases. 1) Requirement planning, 2) data

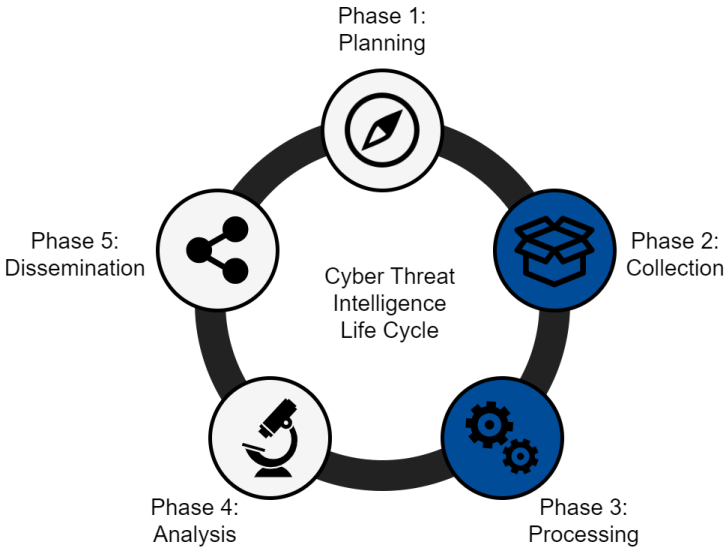

Figure 2.1
Cyber Threat Intelligence Life Cycle. This dissertation focuses on Phases 2 and 3 (blue)

collection, 3) information processing, 4) intelligence analysis, and 5) dissemination [14]. Each of these stages leverages different methods to accomplish its objectives. Requirement planning involves defining what type of intelligence needs to be collected and how the intelligence will be used. Data collection involves identifying threat intelligence sources to start raw data collection. Raw data can be obtained in various ways, including web scrapping, deploying honeypots, running breach and attack simulations among others [24]. Data processing involves structuring raw data, discarding noisy or irrelevant data, and categorizing the data using models and taxonomies. Data analysis involves analyzing the information obtained in Phase 3 to understand threat actors' tactics, techniques, and procedures (TTPs). Lastly, data dissemination involves deploying the intelligence obtained in Phase 4 into an intelligence repository or database that can be used by a threat intelligence tool such as Security Information and Event Management frameworks or SIEMs.

Recently, industry and government institutions have integrated CTI as a key part of their cybersecurity strategy; these include Cisco, Microsoft, and National Security Agency [25], to name a few. These CTI strategies have been focused mainly on commodity computer and enterprise security. For example, Microsoft's threat intelligence efforts are focused on threats such as social engineering, e.g., phishing, cloud computing, and ransomware [26].

2.2.1 Data Collection (Phase 2)

Collection includes those activities related to data acquisition required to satisfy the requirements specified in the collection strategy. This includes developing and coordinating collection sensors, exercising authoritative control of specific collection operations, monitoring the overall satisfaction of requirements, and assessing the effectiveness of the collection plan to satisfy the original and evolving intelligence needs. These sensors or sources of data are varied and include multiple techniques [27]. For example, honeypots [15], malware analysis repositories such as VirusTotal [28], and social media threat exchanges such as ThreatExchange by Meta [29].

2.2.2 Information Processing (Phase 3)

Raw collected data is converted into forms readily used by decision-makers, intelligence analysts, and other consumers, as depicted in Fig. 2 [13]. Processing and exploitation include data classification, editing, conversion and correlation, document and media translation, and reporting the results of these actions to analysis and production elements. Processing and exploitation may be federated or performed by the same element that collected the data [19]. One of the most important parts of information processing is data classification because big amounts of data can be separated and processed independently. To classify raw collected data, several threat taxonomies have been proposed [30, 22, 31]. These taxonomies define and classify the different ways in which adversaries can attack a particular system. For example, the MITRE ATT&CK® framework is a taxonomy of attacks that target Enterprise computers such as Windows and MacOS [32].

2.2.3 Cyber Threat Intelligence of Cyber-Physical Systems

CTI has become a mature and well-developed discipline in the context of enterprise IT and commodity computing. This has resulted in a plethora of commercial CTI platforms developed by important technology companies. These include Microsoft's Defender Threat Intelligence [33], IBM's X-Force [34], Google's Cyber Threat Intelligence [35] and Cisco's Talos [36]. These platforms are often backed by large-scale telemetry, and a robust ecosystem of tools for ingestion, enrichment, and response.

In contrast, CTI tailored for CPS is both limited in scope and fragmented across sectors. Most existing work is narrowly focused on ICS, with Dragos being one of the main companies that offer ICS-specific products for CTI in addition to producing threat reports and malware analyses specific to that domain [37, 38]. Even within ICS, public CTI offerings are sparse, and lack standardization. Beyond ICS, other critical CPS domains, such as connected and automated vehicles (CAVs) and satellites, have received little attention in either academic or commercial CTI efforts. This contrast between commodity computer CTI and CPS CTI highlights a critical research gap, as CPS increasingly underpin critical infrastructure and real-world safety, the lack of tailored CTI leaves them vulnerable to attacks that traditional IT-centric intelligence cannot address.

CHAPTER III: THREAT TAXONOMY FOR INDUSTRIAL CONTROL SYSTEMS

Abstract

Billions of people rely on essential utility and manufacturing infrastructures such as water treatment plants, energy management, and food production. Our dependence on reliable infrastructures makes them valuable targets for cyberattacks. Among the prime targets for adversaries attacking physical infrastructures are Programmable Logic Controllers (PLCs), because they connect the cyber and physical worlds. In this dissertation chapter, we conduct the first comprehensive systematization of knowledge that explores the security of PLCs: we present an in-depth analysis of PLC attacks and defenses and discover trends in the security of PLCs from the last 15 years of research. Additionally, we identify and point out research gaps that if left ignored could lead to new catastrophic attacks against critical infrastructures.

3.1 Introduction

Programmable Logic Controllers, or PLCs, are small rugged computers that have programmable memory to store functions such as timers and logic gates. These functions can control *physical* machines, such as water pumps or centrifuges [1]. PLCs are rather unique systems due to two main characteristics: First, they act as a bridge that connects the cyber and physical worlds; therefore any attack carried out on them can have an immediate effect in the real world. Second, they have opaque, heterogeneous architectures that are different from traditional computer architectures, comprising multiple firmware, which makes them difficult to secure [39].

Before the introduction of PLCs, physical processes were controlled by *relay* panels. Relays are switched either *on* or *off* by an electric current, and thus they can be used in logic circuits. Ladder logic was created as a way to configure relay panels to automatically launch actuation commands based on sensor information. As industrial processes became more complex and varied, relays were not enough to meet the new requirements. An evolution of PLCs, first based on microprocessors, then on networks, and more recently on modern technology paradigms, such as virtualization, has introduced multiple benefits, but at the same time increased the attack surface of these systems.

The first PLC, the *Modicon 084*, was introduced in 1968 [40]. Unlike relays, PLCs could be programmed and reprogrammed to adjust to the process requirements without any physical changes to the control system. A flurry of programming languages emerged, extending ladder logic and introducing new paradigms. Eventually, in 1993, the IEC 61131-3 standard unified these multiple languages into basic standards for PLC programming languages [41].

The first PLCs communicated with the physical world as well as with other computers via analog or serial communications. As computer networks became more reliable and available in the IT world, Ethernet communications were introduced to PLCs. This network connectivity keeps increasing. For example, the Siemens S7-1500 PLC includes up to 4 Ethernet ports, whereas the legacy S7-300 needed an expansion module to support Ethernet connectivity. At the same time, network accessibility now allows remote attackers to deploy classical network attacks against PLCs.

Nowadays, with the advent of Industry 4.0 and IIoT [42], PLCs are going through another paradigm shift bringing even more functionalities like cloud integration, web services, and virtualization. New players like CODESYS and OpenPLC have also entered the market, challenging the long-standing practice of using proprietary hardware and proprietary software products that dominated the PLC industry. These changes pose many open questions about the security of PLCs to the research community.

In order to advance the security of PLCs and the systems they interact with, plenty of research has been produced in the past decades. In particular, research output increased after the term Cyber-Physical System (CPS) was coined in late 2006 by the NSF in the United States [19]. However, the community still lacks an up-to-date general understanding of where the security of PLCs stands and what directions should (or should not) be taken in the future.

To address this challenge, we introduce a comprehensive analysis of the security of PLCs by integrating knowledge from multiple fragmented origins (scientific and grey literature), comprising many existing attack and defense methods in the literature. In addition, we introduce a novel threat model as well as classification and evaluation criteria for summarizing and structuring the existing knowledge about PLC security.

In summary, we make the following contributions:

- We provide a systematization of the literature that consists of **133 papers**, which include **119 attack methods**, and **70 defense methods** (one paper might include one or more attack or defense methods).
- We present a comprehensive taxonomy[1] that integrates the scientific literature with the existing ATT&CK for ICS Matrix in collaboration with MITRE ATT&CK®.
- We identify important PLC security research gaps and discuss future research directions and recommendations.
- We provide **three public tools** to facilitate and foster research and collaboration on this topic. **1)** Our full systematization dataset[2], which other researchers can use to replicate our results and perform their own analyses. **2)** A PLC security artifacts repository[3], which acts as a centralized database for artifacts updated by the community. **3)** A reference graph[4] that enables researchers to further analyze the reviewed literature in this SoK.

3.2 Background

We now introduce some relevant background topics on PLCs that will be leveraged in future sections.

3.2.1 PLC System Environment

PLCs [43] are an essential component of most physical critical infrastructures such as water treatment systems and gas pipelines. PLCs are commonly found in Industrial Control Systems and Supervisory Control and Data Acquisition (SCADA) systems. In the last few years, two new concepts have emerged in the context of Cyber-Physical Systems: Industrial IoT (IIoT) and Induystry 4.0. IIoT is a subset of IoT concerned with connecting industrial assets, e.g., PLCs, with

[1] https://github.com/efrenlopezm/ics2matrix
[2] https://github.com/efrenlopezm/plc-sok-dataset
[3] https://github.com/efrenlopezm/plcsecurityartifacts
[4] https://www.researchrabbitapp.com/collection/public/E6XRY0186G

information systems and business processes. Industry 4.0, on the other hand, is a subset of IIoT and refers to the use of Internet technologies to improve production efficiency by employing smart services in smart factories [42]. Figure 3.12 illustrates the relationship between the above concepts. Furthermore, IIoT and Industry 4.0 have changed PLCs in two ways: First, they have introduced support for modern network protocols, which we discuss further in Sec. 3.2.3. Second, they have changed the hardware and software architectures to support virtualization and compatibility. We further discuss these topics in Sec. 3.2.4.

PLCs' industrial environment, however, remains largely the same and typically includes the following [4]:

Actuator. A hardware component that moves or operates a device in the physical world. Examples of actuators include valves, motors, and piezoelectric actuators.

Sensor. A device that generates an electrical analog or digital signal that represents a physical property of a process. Examples include temperature or magnetic field sensors.

Engineering Station. A general-purpose computer that is used to write the control logic or ladder logic code for the PLC to execute. It is usually connected to the PLC so that the compiled control logic program can be uploaded.

Human Machine Interface (HMI). The hardware or software used to interact with the PLC, e.g, a physical control panel with buttons and lights or a software display.

We further discuss PLCs' underlying architecture and environment in Appendix 3.10.3.

3.2.2 PLC Basic Components

As shown in Fig. 3.1, a typical PLC has the following basic components that are susceptible to attacks:

Control Logic. A control logic program contains the instructions that the PLC executes to interact with its environment. The IEC 61131-3 standard [41] specifies the syntax and semantics of a unified suite of programming languages for PLCs. Control programs are written in one of these supported languages, e.g., Structured Text, and then compiled into machine code or bytecode. For instance, Siemens PLCs run proprietary MC7 bytecode [44] compiled by the SIMATIC S7

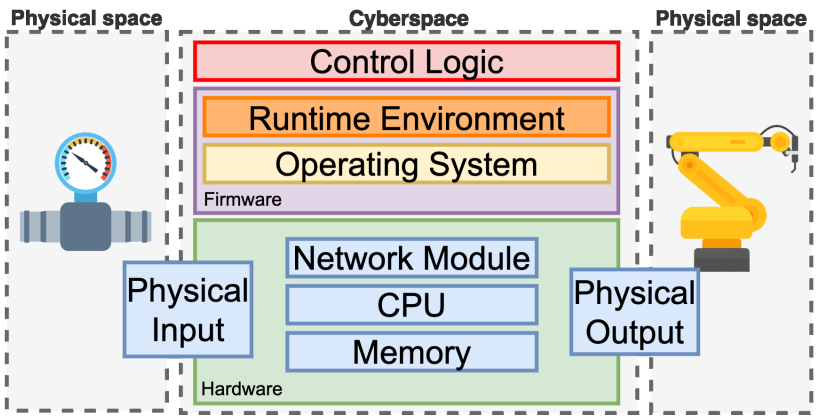

Figure 3.1
A Generalized PLC Architecture. Based on [1, 2].

Manager software [45].

Runtime Environment. The runtime environment executes the process control code [2] and interacts with the I/O modules. It can be proprietary, e.g., Schneider, or open source like the OpenPLC runtime [46].

Operating System. Most PLCs have a Real-Time Operating System (RTOS) [47]. RTOS are operating systems that meet strict processing time requirements and support real-time applications. Vendors support a variety of RTOS in their platforms. For example, Siemens supports the Nucleus RTOS [48] and VxWorks [49].

Firmware. The firmware bridges the gap between the PLC hardware and software. While simple PLCs might run applications as bare metal (without an OS) [50, 51], modern PLCs use the firmware under an RTOS. Firmware can be upgraded via SD cards or through a network connection [52].

CPU. The CPU interprets the input signals and executes the logic instructions saved in memory. The CPU chassis has slots where other components may be attached, e.g., a network module. It may also include USB ports and SD Card slots.

Memory Unit. It stores the program that the CPU will execute along with input data. The memory unit may include different types of memory blocks, which are further discussed in Appendix 3.10.4.

Network Module. Modern PLCs can have one or more ports to communicate with the supervisory control network (regular computers monitoring the process) or the fieldbus (actuators and sensors).

Physical I/O Modules. These include input modules with metal pins that receive information (via a voltage or current analog signal) from sensors. The output modules send analog data to actuators such as servo motors.

PLC Scan Cycle. It is the cycle in which the PLC reads the sensor inputs, executes the current control logic program and updates the output to the actuators. It is measured in ms and should stay constant. If its time increases, PLCs implement a watchdog timer that sends the PLC to a *Fault* mode [53, 54].

3.2.3 PLC Communication Protocols

As discussed in the introduction, PLCs are becoming more connected via modern communication protocols. Many of these protocols were not designed to include strong security features [55], allowing for the proliferation of vulnerabilities that make the PLCs running them susceptible to attacks [56, 57]. Protocols commonly used in practice include, but are not limited to, the following:

Fieldbus. PLCs use Fieldbus [58] protocols to talk to sensors and actuators. Historically these communications were done through serial-based interfaces or analog signals. Sample standards include Profibus, CAN bus, Modbus, and DeviceNet. They all differ in features and implementation, resulting in limited compatibility [59, 60].

Supervisory Network Protocols. PLCs communicate with other controllers or classical computers through a variety of proprietary and standardized protocols. These protocols use the IEEE 802.3 Ethernet standard [61] in industrial environments [62]. For example, the EtherNet/IP protocol combines IEEE 802.3 and the TCP/IP Suite [63]. Some are designed to operate on LANs (they do not use IP addresses but only MAC addresses), such as GOOSE, while others use TCP/IP and can be used on LANs and WANs, such as Modbus TCP.

Industry 4.0 Protocols. These protocols include support for cloud connectivity, compatibility between devices, and security features. Examples include MQTT (Message Queuing Telemetry

Transport) [64] and OPC-UA (Open Platform Communications-Unified Architecture) [65]. Currently, some PLC models support Industry 4.0 protocols [66], whereas older PLCs can be retrofitted to support them [67].

3.2.4 SoftPLCs

The term SoftPLC has been used in the scientific literature since the late 1990s [68, 69, 70]. Although it has not been well defined, generally a SoftPLC is a software runtime environment that executes PLC programs, for example, programs that follow the IEC 61131-3 standard [41]. The runtime is portable and compatible with multiple hardware that can range from microcontrollers to cloud servers [71]. The two major SoftPLC projects are CODESYS [72] and OpenPLC [71]. CODESYS supports approximately a thousand different device types from more than 500 manufacturers [73]. OpenPLC, first proposed in 2014 [74], includes a runtime and an editor, and it is compatible with 18 platforms including Windows and Linux [71]. One of the biggest differences between OpenPLC and CODESYS is that OpenPLC's runtime is open source [46], unlike that of CODESYS. For the remainder of this work, we will refer to traditional PLCs as *HardPLCs*, in order to differentiate them from SoftPLCs.

3.3 Research Questions and Methodology

In this section, we first elaborate on the research questions, and we then describe the methodology we followed to systematize knowledge by collecting, analyzing, and evaluating works in the literature.

3.3.1 Overview of Research Questions

RQ-1: ***What are the attack methods against PLCs?*** We aim to categorize and analyze attack methodologies targeting PLCs introduced in the literature in the last 17 years.

- **RQ-1.1:** *Which components of the PLC are targeted?* We aim to identify what internal components of a PLC, e.g., the CPU described in Sec. 3.2.2, are being targeted, in an effort to identify patterns such as the components that have received the most attention in the literature.
- **RQ-1.2:** *How difficult is it to deploy attacks?* We also aim to identify the level of effort

required by attackers to successfully carry out attacks against PLCs. In Sec. 3.4, we elaborate on a threat model and classification criteria, including the potential attack vectors as well as the level of access required by attackers, e.g., Internet access, needed to deploy an attack.

- **RQ-1.3:** *What is the impact of deploying attacks?* Finally, we are interested in identifying the impact that attacks against PLCs may have if they are deployed successfully. This includes the level of disruption achieved, e.g., modifications to control logic programs as shown in Sec. 3.2.1.

RQ-2: *What are the defense methods to protect PLCs?*

In addition to attacks, we want to systematize the PLC defenses proposed in the literature in the last 17 years.

- **RQ-2.1:** *Which components of the PLC are protected?* We want to identify which of the components introduced in Sec. 3.2.2 are the focus of defenses to understand which of them have received less attention and may therefore remain unprotected.

- **RQ-2.2:** *How difficult is it to deploy defenses for PLCs?* We also study the level of effort required to deploy defense mechanisms for PLCs. As shown in Sec. 3.4.5, this includes the organizational effort from administrators and/or operators to modify and adjust a given PLC/ICS Environment and the performance overhead.

- **RQ-2.3:** *Are there enough defenses addressing reported attacks for PLCs?* Finally, we are interested to know if enough defenses exist in the literature to counteract the attacks we have identified as a part of this work. This way, we aim to identify research gaps and discuss the directions of future work as we describe in Sec. 3.7 and Sec. 3.8.

3.3.2 Knowledge Systematization Methodology

To answer the Research Questions raised in Sec 3.3.1, we use the systematization methodology shown in Fig. 3.2. First, we perform a systematic literature review. Second, based on the data

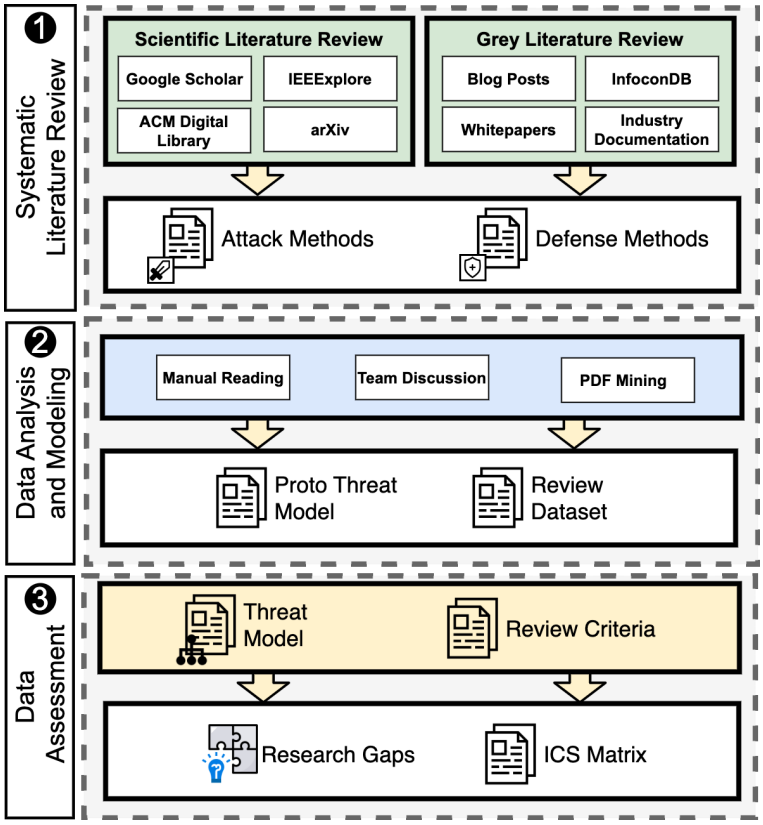

Figure 3.2
Our systematization methodology. Based on [3].

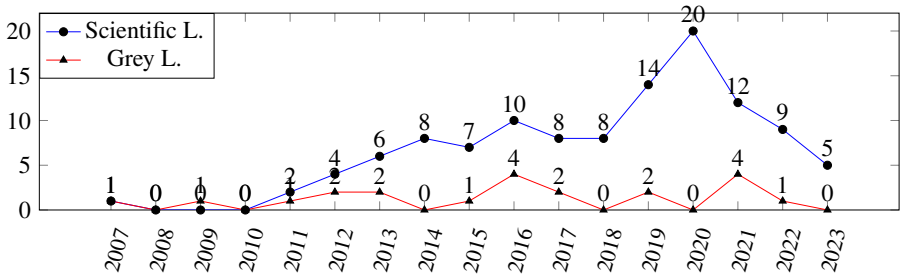

Figure 3.3
Literature focused on PLC Security per Year.

obtained in the literature review, we perform data analysis and modeling. Finally, we assess the data.

❶ **Systematic Literature Review**. We review both scientific and grey literature to collect PLC attack and defense methods. We also define the final SoK scope based on the specified inclusion criteria.

Scientific Literature Review. We consider scientific literature to mean the literature that is

based on the scientific method that uses evidence to develop conclusions. It uses previous literature to develop theories and hypotheses while taking care to cite the authors and tools that are used. For the scientific literature review, we carried out the following steps: **1)** Search Querying. We used search queries based on keywords such as *"plc"*, *"ics"*, and *"security"*. The complete list of keywords can be found in Appendix 3.10.1. **2)** Consulting literature resources. We used these queries to search the selected resources such as Google Scholar and the ACM Digital Library. The complete list of consulted resources can be found in Appendix 3.10.1. **3)** Applying inclusion criteria. After collecting all these papers, we include them only if they meet the following three criteria. A) The paper must include the term "PLC" or "Programmable Logic Controller". B) It proposes an attack or defense method. C) It includes an experimental evaluation of an attack or defense with at least one PLC. **4)** Applying snowballing. We applied the snowballing technique [75] on all the papers that matched the above criteria to find papers that went unnoticed during the initial search.

Grey Literature Review. We consider grey literature as literature with limited distribution (i.e., not included in academic publishing libraries). It includes unpublished reports, policy documents, white papers, and technical reports [76]. To perform our grey literature review, we followed the same steps used in the scientific literature review, except that in step **2)** (consulting literature resources) we instead searched in the grey literature resources listed in Appendix 3.10.1.

Final SoK Scope. As a result of the scientific and grey literature reviews we collected 133 papers with the earliest being from 2007 and the latest from 2023. Although we searched for papers before 2007, we did not find any that met our criteria. To the best of our knowledge, and based on our exhaustive research, there was no research on designing new attacks or defenses for PLCs before 2007. Fig. 3.3 shows the final number of scientific and grey papers included in our selection criteria per year of publication, depicting an overall increase in PLC security research over the years, peaking in 2020.

❷ **Data Analysis and Modeling**. In this phase, we first read and analyzed each paper to extract important security-relevant information such as attack vectors, PLC models and manufacturers,

and PLC target component. Second, we matched each of the attack and defense methods to their corresponding MITRE technique, subtechnique or mitigation category. The MITRE framework is further discussed in Sec. 3.4.1. Third, we recorded this information in a spreadsheet. The result of this data analysis and modeling was the identification of the building blocks for a first draft of our threat model depicted in Fig. 3.4 and the classification criteria discussed in Sec. 3.4., e.g., the access level and PLC target component.

❸ **Model Assessment**. In this phase, we evaluate the data, threat model, and criteria developed in ❷ to produce the SoK contributions. We use these results to summarize our findings and identify the research gaps discussed in Sec. 3.7. Finally, we evaluate the results to produce the ICS threat taxonomy discussed in Sec. 3.4.1.

3.4 Classification and Evaluation Criteria

In this section, we present the criteria to classify and evaluate the works considered for this SoK. These criteria and the symbols introduced alongside them will later be used in Tables 3.1 and 3.2. Our first research question RQ-1 is addressed in Sec. 3.4.2, Sec. 3.4.3, and Sec. 3.4.4. Our second research question RQ-2 is addressed in Sec. 3.4.5, Sec. 3.4.6 , and Sec. 3.4.7.

3.4.1 ICS Threat Taxonomy

To better classify attack and defense methods in the following sections, we extended the MITRE ATT&CK for ICS Matrix [77] and the Hybrid ATT&CK Matrix [78] to create the ICS^2 Matrix, a taxonomy of threats against PLCs and ICS. The taxonomy includes adversary tactics that describe "what" is the adversary's goal and attack techniques that describe "how" the adversary can complete their goal. Additionally, it includes mitigations that prevent a technique from being successfully executed. The ICS^2 Matrix incorporates the scientific knowledge accumulated during the past 17 years of PLC security research by adding 6 new attack techniques and 5 new mitigation categories based on the literature reviewed in this SoK. Due to space restrictions, we provide a condensed version of our taxonomy in Fig. 3.13. The full ICS^2 Matrix version is publicly available[5] for the security community to use and extend.

[5] `https://github.com/efrenlopezm/ics2matrix`

Matrix Integration. One of the main limitations of the MITRE ATT&CK for ICS Matrix [77] is that it does not incorporate techniques based on security research findings but instead focuses on techniques based on real-world ICS cyberattacks. For example, the "system firmware" technique [79] includes a reference to Triton discussed in Appendix 3.10.6. However, it does not include any references to any of the PLC Firmware research listed in Table 3.2. This was also confirmed by MITRE during one of our communications. Our ICS^2 Matrix fills this gap. To integrate our systematization findings with the existing MITRE for ICS Matrix, we needed to add techniques not already covered by MITRE. To develop these new techniques, we used MITRE's official guidelines [8]. This document outlines the information required to create a new MITRE-style technique, for example, a matching "tactic" and "procedure example". In Appendix 3.10.2 we describe in detail how we developed and included one of these techniques.

3.4.2 Attack Classification

Considering RQ-1.1, the following criteria classify attacks according to the taxonomy technique, e.g., Adversary-in-the-Middle, the basic components that are targeted, e.g., the CPU or Memory, and the attack vectors that can be ultimately used against a PLC.

Target PLC Component. Each attack method was classified according to the PLC Basic Component that is the primary target, following the description previously shown in Sec. 3.2.2. Such information was retrieved based on the descriptions explicitly provided by the paper's authors.

Attack Category. Each attack method was sorted into one *technique* category, based on our ICS^2 Matrix. The technique categories allowed us to determine what kind of defenses might (or might not) counter them.

Attack Vector. When launching an attack against a PLC, there might be one or more vectors or paths available to the adversary to deliver the payload. The list of attack vectors we considered are shown in Table 3.4. Most of the time the attack vector is a vulnerability in the implementation of a network protocol, e.g., S7comm. However, there are other vectors like inserting an SD card into the PLC chassis, for instance, the work by Garcia et al. [80].

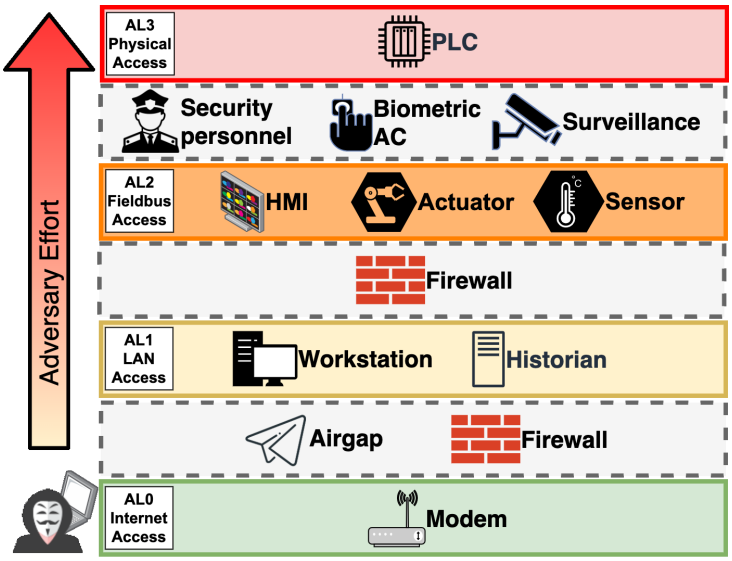

Figure 3.4
PLC-centered Threat Model. Based on [4].

3.4.3 Attack Complexity

Following RQ-1.2, the criteria shown next are intended to understand, qualify, and quantify how complex it is for an attacker to carry out an attack against a target PLC.

Environment Knowledge. This criterion evaluates how much knowledge of the cyber-physical system environment where the PLC lives, e.g., the system topology discussed in Sec. 3.2.1, is required to launch the attack. An attack may require *zero knowledge* (○), e.g., a DoS[81] attack targeting the PLC network module only requires an IP address and does not require any environment knowledge. *Partial knowledge* (◑) may require basic information only, e.g., a high-level description of the physical process. CaFDI [82], for example, requires limited local information about the substation configuration. Finally, *extensive knowledge* (●) requires detailed information about the cyber-physical environment. For example, SABOT[83] requires the adversary to encode their understanding of the system behavior into a specification.

Available Source Code. This criterion evaluates whether the attack source code is publicly available (●) or not (○).

Access Level. We also consider the level of access an attacker needs to carry out the suggested

attacks. Our threat model, shown in Fig. 3.4, is divided into four distinct *Access Levels* (ALs) labeled from AL0 to AL3.

Adversaries might try to move through the ICS environment in order to reach the PLC and deliver their exploit payload. Overall, our threat model helps us answer two main questions: *1) What access level is required to attempt the attack?*, and *2) What attack surfaces are available to the adversary?*

AL0: Access to PLC via the Internet (0). An adversary may be able to access a PLC through an Internet-compatible protocol, e.g., S7comm. This is usually a private network where a remote SCADA computer connects to a PLC via an industrial protocol such as DNP3. Attacks may also be launched if a PLC is publicly exposed on the Internet. At first, this might seem like a ludicrous idea: Why would anybody make a PLC publicly available over the Internet? However, a preliminary Shodan search suggests that the number of publicly accessible PLCs (excluding Shodan-identified honeypots) is over 6,500 as of September 2023[6]

AL1: Access to Supervisory LAN (1). This level requires an attacker to have access to the LAN network of the industrial process. This can be achieved if the attacker compromises workstation computers, data historians, or similar operational computers. Some industrial protocols run only on Ethernet and are non-routable (such as GOOSE), so an attacker targeting a vulnerable GOOSE stack on a PLC will have to do so in the same LAN.

AL2: Access to PLC Fieldbus Network (2). As mentioned in Sec. 3.2.3, fieldbus represents the lower level of communications between the PLC and nearby field equipment such as actuators and sensors. An attacker can compromise any of these devices to exploit a vulnerability in the fieldbus code implementation. The attacker can also launch Adversary-in-the-Middle attacks in this network, as fieldbus communications are rarely authenticated in practice.

AL3: Physical Access to PLC (3). Lastly, this level assumes the adversary has bypassed environmental and physical protection measures (e.g., locked doors) of the target. Having physical access to the PLC may be the most difficult scenario for an adversary but can still be achieved

[6] `https://www.shodan.io/search?query=plc+-%22792%2F71644%22`.

by malicious insiders or disgruntled employees. An attacker with physical access can use a JTAG interface, as shown in Table 3.4.

3.4.4 Attack Impact

Following RQ-1.3, the criteria below are intended to understand and quantify the potential impact of a successfully carried out attack, e.g., the *payoff* in return of the effort invested, as discussed in Sec. 3.4.3.

Potential Damage. This criterion evaluates the *immediate* damage inflicted to the physical operation of the ICS. The damage is *limited* (1) if the attack does not change the operation of the process but instead aims to collect information or gain unauthorized access, for example, a password sniffing attack [84]. The potential damage is *substantial* (2) if the attack can stop the industrial process partially or completely, e.g., a DoS attack [85]. The potential damage is *severe* (3) if the attack is able to insidiously control the physical process, e.g., a logic bomb or firmware modification attack, where the attacker can launch arbitrary control commands to the system (similar to Stuxnet, discussed in Appendix 3.10.6).

HardPLC Targets. An attack may affect a *single* PLC model (○), e.g., the Siemens SIMATIC S7-300, or may affect *multiple* PLC models of the same manufacturer (◑), e.g., the Rockwell's Allen-Bradley 1100 and 1400 models. Also, an attack may affect *multiple* PLC models from two or more manufacturers (●), e.g., Siemens' SIMATIC S7-1500 and Modicon's M221. An attack that is effective against various devices is an indicator of the severity of the problem, while an attack focusing on a single device might be contained.

SoftPLC Targets. Finally, this criterion indicates if a SoftPLC was used to evaluate the proposed attack. An attack may have been evaluated using *CODESYS* (C), *OpenPLC* (O) or *no SoftPLC* at all (-) (i.e., the attack only focused on HardPLCs).

3.4.5 Defense Classification

After describing our criteria for attacks, we now focus on defenses by looking at RQ-2.1.

Defense Category. Each defense method was sorted into one *mitigation* category, based on our ICS^2 Matrix. The mitigation categories allowed us to determine what kind of attacks might (or

might not) be countered.

Defense Vector. The defense vector is the path used in order to stop the attack payload from being delivered. The available defense vectors are defined in our Threat Model for PLCs in Table 3.4. Most of the time, the attack and defense vectors will match. However, they can differ. For example, *Rajput et al.* use the JTAG interface as a defense vector [86].

3.4.6 Defense Deployability

Following RQ-2.2, the following criteria evaluate how difficult it is to deploy a defense method to protect PLCs.

PLC Overhead. This criterion quantifies the overhead that the defense method incurs on the PLC itself, which can be either *zero* (1), *negligible* (2), or *considerable* (3).

Infrastructure Overhead. This criterion quantifies the cost that comes with setting up the required infrastructure to implement a defense method. That may involve *no changes* (1), or it may involve *either* an infrastructure change, e.g., setting up new VLANs or an additional hardware component, e.g., bump-in-the-wire (2). Finally, a defense may involve *both* infrastructure changes and hardware components (3).

Maintenance. This criterion quantifies the level of post-deployment maintenance required by a given defense method. A defense method may require *no maintenance* (1), *sporadic maintenance* (2), or *constant maintenance* (3).

Source Code Availability. This criterion evaluates whether the defense source code is publicly available (●) or not (○).

3.4.7 Defense Robustness

Following RQ-2.3, we also characterize how the defense mitigates attacks.

Defense Stage. Defenses can be categorized in three different stages of the security process. 1) *Prevention Defense* (PR) aims to reduce the possibility of an incident *before* a known vulnerability is exploited. 2) *Detection Defense* (DE) aims to identify and alert about a *current or recent* attack. 3) *Recovery Defense* (RE) aims to reduce the damage of an attack *after* it has been carried out successfully.

Effectiveness. The defense method's effectiveness is indicated by its accuracy. The accuracy values are taken from the research paper itself. If the paper did not report or specify the accuracy, it is marked as *N/S* (not specified). We selected accuracy because it is the most reported metric across the defense method papers. We recognize that this metric does not work for all defense methods, however, it is the most frequently reported quantitative metric available in the current literature. We further discuss this problem in Sec.3.8.

HardPLC Scope. This criterion evaluates the number of different PLC models that are protected by the defense. The defense may protect a *single* PLC model (○), e.g., the Siemens SIMATIC S7-300, or it may protect *multiple* PLC models of the same manufacturer (◑), e.g., the Rockwell's Allen-Bradley 1100 and 1400 models. Also, a defense may protect *multiple* PLC models from two or more manufacturers (●), e.g., Siemens' SIMATIC S7-1500 and Modicon's M221.

SoftPLC Scope. This criterion indicates if any SoftPLC was used to evaluate a defense, either *CODESYS* (C), *OpenPLC* (O) or *no SoftPLC* at all (-).

3.5 Overview of Attacks

By using the systematization methodology presented in Sec. 3.3 and the criteria discussed in Sec. 3.4, we obtained Tables 3.1, and 3.2. These tables use the *PLC target component* categorization defined in Fig. 3.1.

We now illustrate examples of attacks per target component. Due to the large number of papers in our SoK (133 papers), we cannot discuss all of them. To select the papers systematically, we picked the two papers with the most citations from each category. We used Google Scholar to perform the search on June 3, 2023. In the next paragraphs, we discuss the insights provided by these highly-cited papers.

Attacks that Target the PLC Communications Module. *Urbina et al.* [139] introduce an AitM attack between the PLC and the field devices (AL2). An interesting observation of that paper is that field networks tend to follow a ring topology rather than the typical star topology of switched Ethernet networks; therefore, an AitM attack does not need to use ARP (Address Resolution Protocol) spoofing or similar techniques to place itself between two devices. It just

Attacks											Defenses											
				Complexity			Impact							Deployability					Robustness			
Target Component	Method	Category (Technique)	Attack Vector	Access Level	Env Knowledge	Source Code	P. Damage	HPLC Targets	SPLC Targets		Method	Category (Mitigation)	Defense Vector	PLC Overhead	Infra Overhead	Maintenance	Source Code	Effectiveness	D. Strategy	HPLC Scope	SPLC Scope	
Network	Attacks on situational awareness[87]	Adversary-in-the-Middle	P	1	●	○	2	○	-		Cross-layer fingerprinting[88]	Network Intrusion Pre.	D	1	1	1	○	92.8	DE	○	-	
	Concealment Attack[89]		M	1	○	●	2	○	-		DFA-based intrusion detection[90]		M	1	2	1	○	99.8	DE	○	-	
	Controller Eavesdropping[91]		P	1	○	○	1	●	O		Encrypted Traffic IDS[92]		I	1	2	1	○	N/S	DE	-	-	
	Controller Packet Tampering[91]		P	1	○	○	3	●	O		IDS for S7 networks[93]		S7	1	2	1	○	99.8	DE	○	-	
	False Data Injection Attack[94]		M	0	○	○	2	●	-		Model-based anomaly detection[95]		S7	1	2	1	○	96	DE	○	-	
	Man in the middle[96]		E	1	○	○	1	○	-		Physical fingerprinting[88]		D	1	1	1	○	92.8	DE	○	-	
	OPC-UA Rogue Client[97]		UA	0	○	○	2	○	-		PLC Watermarking[54]		E	1	2	1	○	N/S	DE	●	-	
	Replay attack[84]		P	0	○	○	2	○	-		Embedding Encryption[98]	Encrypt Network Traffic	M	1	1	1	○	N/S	PR	-	O	
	Replay attack[99]		F	0	○	○	2	○	-		Hash Authentication[100]		F	1	1	1	○	N/S	PR	○	-	
	SDN Enabled MitM[91]		P	1	○	○	1	●	O		LVST (LabView SSH Tunnel)[101]		E	1	2	1	○	N/S	PR	●	-	
	ISO-TSAP Replay Attack[102]		P	0	○	○	2	◑	-		AES-256 Implementation[103]		M	1	1	1	●	N/S	PR	-	O	
	Third-party Eavesdropping[91]		P	1	○	○	1	●	O		PLCrypto[104]		E	1	1	1	●	N/S	PR	○	-	
	Change the IP[105]	Denial of Service	E	0	○	●	2	○	-		PLC-Sleuth[106]	Network Intrusion Pre.	S7	1	2	1	○	100	DE	○	-	
	Control engine attack[107]		M	0	○	○	2	●	-		Semantic IDS[108]		M	1	2	1	○	N/S	DE	-	-	
	Denial of Service Attack[81]		P	0	○	○	2	○	-		Telemetry IDS[109]		M	1	2	1	○	99.5	DE	-	-	
	DoS clearing Flow Table[91]		P	0	○	○	2	○	O		Zeus[110]		E	1	2	2	○	98.9	DE	○	-	
	Flow Rule Blocking[91]		P	0	○	○	2	○	O		Side-channel Anomaly Detection[111]	Out-of-Band Comms Channel	E	1	1	2	○	N/S	DE	○	-	
	UDP reflect attack[99]		F	0	○	○	2	○	-		Anomaly detection[112]		N	1	2	2	○	90	DE	●	-	
	Authentication Bypass[113]	Modify Auth. Process	E	1	○	○	1	○	-		Arcade.PLC[114]	Control Logic Verification	S7	1	1	1	○	N/S	PR	○	-	
	Cryptographic attack[99]		F	1	○	○	1	○	-		ShadowPLCs[115]	Network Intrusion Pre.	S7	2	1	2	○	97.3	DE	○	-	
	Replay attack[116]		S7	0	○	○	1	◑	-		Snapshooter[117]		S	2	2	1	○	N/S	DE	-	O	
	S7 Authentication Bypass[102]		P	0	○	○	1	◑	-		Traffic Data Classification[118]		S7	1	1	1	○	99.7	PR	○	-	
	Unauthorized password updating[84]		S7	0	○	○	1	○	-		vBump[119]		G	1	3	2	●	N/S	PR	○	C	
	SoMachine Authentication[96]	Hardcoded Credentials	M	0	○	○	1	○	-													
	Credentials from storage[96]		E	0	○	○	1	◑	-													
	subverting read/write-protection[96]		P	0	○	○	1	○	-													
	subverting write-protection[96]		P	0	○	○	1	○	-													
	Command packet flooding[120]	Network Denial of Service	E	0	○	○	1	○	-		Modbus/TCP Firewall[121]	Filter Net. Traffic	M	1	2	1	○	N/S	PR	○	-	
	Modbus Flooding Attack[122]		M	0	○	○	2	○	-													
	Network Flooding Attack[85]		M	0	○	○	2	●	-													
	UDP flooding Attack[123]		M	0	○	○	2	○	-													
	Password reset attack[96]	Brute Force	M	0	○	○	1	○	-		Shade[124]	Network Intrusion Pre.	E	2	1	1	●	N/S	DE	●	-	
	CLIK password attack[125]		M	0	○	●	1	○	-		Host Anomaly Detecton[2]		M	2	1	1	○	N/S	DE	●	-	
	Dictionary Attack[126]		B	0	○	○	1	○	-		Argus[127]		E	1	3	3	○	N/S	DE	-	-	
	Dump module code[105]	Data from Local System	E	0	○	●	1	○	-		PLC-PROV[128]	Validate Program Inputs	N	1	2	2	○	N/S	DE	-	C	
	Memory Logic Attack[102]		P	0	○	●	1	○	-		PLCPrint[129]		S7	1	1	1	○	95	DE	●	-	
	Crash 1756-ENBT module[105]	Device Restart/Shutdown	E	0	○	●	2	○	-		ABAC model for PLC[130]	Authorization Enforcement	S7	1	2	1	○	N/S	PR	●	-	
	Reset 1756-ENBT module[105]		E	0	○	●	2	○	-		FINS detection rules[131]	Network Allowlists	F	1	2	1	○	N/S	PR	◑	-	
	Leak Modbus data[132]	Exfiltration Side-channel	M	0	○	○	1	○	C													
	Leak OPC-UA Process Data[132]		UA	0	○	○	1	○	C													
	Passive network scanner[133]	Network Conn. Enumeration	M	1	○	○	1	●	-													
	Port scanner[134]		P	0	○	●	1	○	C													
	SOCKS Proxy[134]	Connection Proxy	P	0	○	○	1	○	C		SDN-enabled automatic response[135]	Network Intrusion Pre.	E	1	2	1	●	N/S	DE	-	-	
	New ADMIN account[126]	Exp. for Credential Access	B	0	○	○	1	○	-		WeaselBoard[136]	Exploit Protection	BP	3	3	3	○	N/S	PR	●	-	
	Getting a Shell on the PLC[102]	Exp. for Privilege Escalation	P	0	○	○	3	◑	-		Memory Access Taintedness[137]	Exploit Protection	M	-	1	1	○	N/S	DE	-	-	
	Password stealing[84]	Network Sniffing	P	0	○	○	1	○	-		SCADA Protocol Obfuscation[138]	Encrypt Net. Traffic	M	1	1	1	○	N/S	PR	○	-	
	Wireless Fieldbus MiTM[139]	Wireless Sniffing	E	3	○	○	1	○	-		Secure Logging for ICS[140]	Encrypt Net. Traffic	S	1	3	1	○	N/S	PR	○	-	
OS	3S CoDeSys Tools[141]	Exp. for Privilege Escalation	C	1	○	●	2	-	C		ECFI[142]	Exploit Protection	RT	2	2	1	○	N/S	PR	○	O	
	Remote arbitrary code execution[143]		P	0	○	○	2	◑	-													
	Arbitrary Code Execution[144]	Exp. for Client Execution	R	3	○	○	2	○	-													
Runtime	Ghost in the PLC[145]	Block I/O Communication	R	3	◑	○	3	●	C		GhostBuster[146]	Exploit Protection	RT	2	1	1	○	N/S	DE	○	C	
	Unauthenticated file read/write[147]	Data from Local System	C	1	○	○	2	○	C													
	XML bomb[147]	Modify Parameter	C	1	○	○	2	○	C													

Table 3.1

A Summary of PLC Attack and Defense Methods (Network, OS and Runtime Components).
D=DNP3, **G**=GOOSE, **S7**=S7COMM, **I**=IEC 104, **UA**=OPC-UA, **P**=Profinet, **E**=EtherNet/IP, **M**=Modbus TCP, **F**=FINS, **S**=Syslog, **B**=Beckhoff, **BP**=Backpane, **R**=SoC Register, **RT**=Runtime, **O**=OpenPLC, **C**=CODESYS / CODESYS upload protocol, **N**=Not Specified, **DE**=Detection, **PR**=Prevention, **RE**=Recovery

Attacks											Defenses											
				Complexity			Impact						Deployability					Robustness				
Target Component	Method	Category (Technique)	Attack Vector	Access Level	Env Knowledge	Source Code	P. Damage	HPLC Targets	SPLC Targets		Method	Category (Mitigation)	Defense Vector	PLC Overhead	Infra Overhead	Maintenance	Source Code	Effectiveness	D. Strategy	HPLC Scope	SPLC Scope	
Control Logic	Advanced Stealthy Injection Attack[148]	Modify Control Logic	P	0	◑	●	3	○	-		Safety Verification[149]	Control Logic Verification	S7	2	2	1	○	N/S	DE	○	-	
	CLIK[125]		SM	1	○	●	3	○	-		CPLCD[150]		S7	1	1	1	○	N/S	DE	○	-	
	Data Execution Attack[151]		M	0	○	●	2	○	-		Trusted Safety Verifier[152]		N	1	2	1	○	N/S	DE	-	-	
	Fragmentation and Noise Padding[151]		M	0	○	●	2	●	-		Detection Manipulation[153]		X	1	1	1	○	N/S	DE	○	-	
	Ladder Logic Bomb[154]		U	3	●	○	3	○	-		PLCloud[155]		S7	1	3	3	○	N/S	RE	○	-	
	Time-of-Day (TOD) interrupt attack[156]		S7	0	◑	○	3	○	-		PLC Guard[157]		S7	1	2	2	○	N/S	PR	○	-	
	Control logic injection attack[158]	Modify Program	X	0	◑	○	3	○	○		PLC-VBS[159]	Vulnerability Scanning	S7	1	1	1	○	N/S	PR	○	-	
	Dynamic Malware Payloads[160]		N	1	○	○	3	-	-		ICSPatch[161]	Update Control Logic	N	1	2	1	●	100	PR	◑	C	
	Executing arbitrary ladder logic[147]		N	0	○	○	2	○	-		D-Box[162]	Limit Access to MCU R.	X	2	1	1	●	N/S	PR	○	-	
	Immediate Failure Attack[163]		T	1	◑	○	2	○	-													
	Latent Failure Attack[163]		T	1	●	○	3	○	-													
	SABOT[83]		N	1	●	○	3	-	-													
	Ladder Logic Exfiltration[164]	Exfiltration over Side-channel	U	3	◑	○	1	○	-													
	Leak Application data[132]		N	0	○	○	1	-														
	S7-1200 Download Attack[57]	Program Download	S7	0	○	○	2	◑	-													
	S7-1500 Download Attack[57]		S7	0	○	○	2	○	-													
	Denial of Engineering Operations[165]	Adversary-in-the-Middle	E	1	○	●	2	○	-		Optimised Datablocks[166]	Encrypt Sensitive Info.	S7	2	2	1	○	N/S	PR	○	-	
	CaFDI[82]	Brute Force I/O	N	2	◑	○	3	-	-													
	Time-of-Day Attack[167]	Change Operating Mode	S7	1	◑	○	2	○	-		PLC redundancy framework[168]	Redundancy of Service	E	2	3	2	○	N/S	PR	○	-	
	LogicLocker[169]	Data Encrypted for Impact	M	0	○	○	2	●	-		Quad-redundant PLC[170]		X	2	3	2	○	N/S	PR	○	-	
	Denial of Decompilation Attack[171]	Denial of Service	S7	0	○	○	2	●	-		Digital twin-based simulation[172]		N	1	3	3	●	N/S	RE	-	-	
	Stable Perturbation Attack[173]	ICS Sector Discovery	N	1	◑	●	3	○	-													
	Evil PLC Attack[174]	Lateral Tool Transfer	X	0	◑	○	3	●	-													
	Targeted Manipulation of FB Operation[175]	Manipulate I/O Image	N	1	○	○	3	●	-													
Firmware	Bricking the device[147]	System Firmware	N	1	○	○	2	○	-		AttkFinder[176]	Process Vul. Discovery	E	1	1	1	●	96	PR	●	-	
	Compromise System Functions[177]		SM	1	○	○	3	○	-		VETPLC[178]		E	1	1	1	○	N/S	PR	○	○	
	Decrypting Siemens Simatic firmware[102]		P	1	○	●	2	◑	-		Similo[179]		E	1	2	2	○	100	RE	●	-	
	Firmware leakage[180]		U	3	○	○	1	○	-		Logging input simulation[181]		E	1	2	1	○	99.9	DE	○	-	
	Firmware modification attack[50]		U	3	○	○	3	○	-		CPAC[182]	Auth. Enforcement	X	2	2	2	○	N/S	PR	○	-	
	Persistent denial-of-service attack[183]	Modify Program	E	0	○	○	2	○	-		Firmware verification tool[184]	Code Signing	X	1	1	1	○	N/S	PR	○	-	
	Remotely-triggered DoS[183]		E	0	○	○	2	○	-													
	Time-based denial-of-service attack[183]		E	0	○	○	2	○	-		PLCDefender[185]	Attestation	E	2	3	1	○	98	PR	●	○	
	Flash Update[105]	Activate Firmware Update Mode	E	0	○	●	2	○	-													
	HARVEY[80]	Data from Debug Port	MC	3	●	○	3	○	-													
	Shutting down the PLC[147]	Device Restart/Shutdown	N	1	○	○	2	○	-													
	Rogue Firmware Load[186]	Module Firmware	E	1	○	○	2	●	-		SNIFU[187]	Code Signing	U	2	2	1	○	100	PR	○	-	
I/O	Leak I/O Process Data[132]	Exfiltration over Side-channel	N	2	○	○	1	-	-		ORRIS[86]	Antivirus/Antimalware	X	2	2	1	●	85.6	DE	○	C	
	Blinkware[188]		X	3	◑	○	1	○	-													
	Analog Emissions attack[189]		N	3	◑	○	1	○	-													
	PHYCO[190]		U	3	◑	○	1	○	-													
	Escalated Privilege I/O Command[113]	Adversary-in-the-Middle	E	2	○	○	2	○	-		Hidden Sensor Measurements[191]	Network Intrusion Detection	X	1	2	1	○	N/S	DE	○	-	
	I/O Command attack[113]		E	1	○	○	2	○	-		Physics-based attack detection[192]		E	1	1	1	○	N/S	DE	○	-	
	Wireless Control[193]	Spoof Reporting Message	N	3	○	○	2	◑	-		Blockchain Monitoring [194]	Encrypt Network Traffic	E	1	3	2	○	N/S	PR	○	-	
	Malicious Expanders[193]		N	3	○	○	2	○	-													
	False sequence attack[195]	Brute Force I/O	N	2	●	○	3	-	-		PCAT[196]	Validate Program Inputs	E	1	2	1	○	N/S	PR	-	-	
	Manipulated Variable[197]	Manipulation of Control	X	2	◑	○	2	○	-		Smart I/O Modules[198]		M	1	2	1	●	N/S	PR	-	-	
	OPC UA Supply Chain Attack[199]	Supply Chain Compromise	UA	0	◑	○	2	○	-		PAtt[200]	Attestation	X	2	2	1	○	97	PR	○	-	
	Backdoor Attack[201]	Unauthorized Command Message	N	2	○	●	2	-	-													
Memory	Leak Crypto secrets[132]	Exfiltration over Side-channel	N	1	○	○	1	-	-													
	WaterLeakage[202]		S7	0	○	○	1	○	-													
	PLC-Blaster[203]	Change Operating Mode	S7	0	○	○	2	○	-		Mitigate Malicious Disruption[204]	Redundancy of Service	N	2	1	1	○	N/S	DE	-	O	
	Clear PLC memory[84]	Data Destruction	P	0	○	○	2	○	-		CPS Twinning[205]		M	1	3	3	●	N/S	RE	○	-	
	ICS-BROCK[206]	Data Encrypted for Impact	U	3	○	○	2	○	-		Armor PLC[207]		N	1	3	1	○	N/S	DE	-	O	
	Memory Dump[177]	Data from Debug Port	SM	1	○	○	1	○	-													
	Exfiltrate FB Variables[175]	Exfiltration over ICS Protocol	P	0	○	○	1	◑	-													
	Storage Based Covert Channel[175]	Fallback Channels	P	0	○	○	1	◑	-													
	DB content manipulation[208]	Modify Parameter	S7	0	◑	○	2	○	-													
	ROP Attack[209]	Process Injection	M	0	○	○	2	○	-													
	False Command Injection Attack[210]	Unauthorized Command Message	M	0	○	○	2	○	○													
CPU	Forcing a CPU Stop[105]	Device Restart/Shutdown	E	0	○	●	2	○	-		WeaselBoard[136]	Exploit Protection	BP	3	3	3	○	N/S	PR	●	-	
	Crash CPU[105]		E	0	○	●	2	○	-		C2[211]		N	2	2	1	○	N/S	PR	○	-	
	CPU Stop and Start Attack[102]		P	0	○	●	2	◑	-													
	S7-1200 Start/Stop Attack[57]	Change Operating Mode	S7	0	○	○	2	◑	-													
	S7-1500 Start/Stop Attack[57]		S7	0	○	○	2	○	-													
	ASIC Reverse Engineering	Supply Chain Compromise	X	3	○	○	3	○	-													

Table 3.2
A Summary of PLC Attack and Defense Methods (Control Logic, Firmware, I/O, Memory and CPU Components).

S7=S7COMM, **UA**=OPC-UA, **P**=Profinet, **E**=EtherNet/IP, **M**=Modbus TCP, **T**=TriStation, **SM**=SoMachine, **U**=USB Port, **MC**=Memory Card, **X**=Others, **O**=OpenPLC, **C**=CODESYS, **N**=Not Specified, **DE**=Detection, **PR**=Prevention, **RE**=Recovery

needs to inject the attack between the two attacked devices. An AitM attacker can then send false sensor readings to the PLC or false control commands to actuators.

Wardak et al. [84] introduce another network attack, which focuses on password sniffing on the network interface between a workstation and the PLC (AL1) in Fig. 3.4. The authors show that several of the connections between the workstations and the PLCs are not encrypted (or authenticated). Consequently, passwords can be sniffed and then used to gain access to protected actions in the PLC (e.g., the attacker can send start and stop commands to the PLC).

Attacks that Target the Control Logic. Two of the most important challenges for modifying the control logic of a PLC are 1) how to infect the PLC without being detected and 2) how to hide the infection from the engineering workstation. To address challenge 1, *Yoo and Ahmed* [151] propose two control logic infection attacks that can bypass network intrusion detection systems. In the first attack, they bypass intrusion detection systems that look for transfers of control logic (compile code) by injecting control code in data blocks (used for the transfer of data such as counters) and then modifying the execution pointer to execute code in data blocks. The second attack uses fragmentation and noise to further obfuscate these control logic transfers to the PLC.

Kalle et al. [125] target both challenges (infection and stealthiness). They consider a different problem in code injection, where they assume they have a binary they want to modify. They then develop a decompiler transforming low-level control logic to a high-level instruction list to help them inject the malicious code before recompiling it into a binary that can be uploaded to the vulnerable PLC. They also develop a virtual PLC that interfaces with the engineering workstation (via an AiTM attack); this virtual PLC then sends previously captured network traffic of the original uninfected control logic back to the workstation.

Attacks that Target the PLC CPU. As discussed in Appendix 3.10.5, PLCs generally have three CPU operating modes: "STOP," "RUN," and "PROGRAM." Attacks targeting the PLC CPU focus on disabling the CPU remotely (AL3 or AL2) with STOP commands. One of the earliest examples of these attacks launched against Siemens PLCs by impersonating the workstation is the work of Beresford [102]. While Siemens released cryptographic protections for these connections

that would prevent these attacks, more recent work reverse-engineered the cryptographic protocol. *Biham et al.* [57] were able to create a rogue engineering station that could remotely start or stop these newer PLCs by compromising a vulnerable key exchange protocol.

Passwords can protect CPU modes of operation, but a compromised password can still enable remote attacks. Higher-end PLCs protect their CPU modes with physical methods, such as using a physical switch or a physical key. These methods are further discussed in Appendix 3.10.5.

Attacks that Target the PLC Firmware. Firmware modification is one of the most powerful attacks in any platform, as the attacker can control access to the input and output modules of the PLC while remaining undetected. *Basnight et al.* [50] pioneered methods on PLC firmware reverse-engineering and how to develop modified firmware as a proof of concept.

The most cited firmware attack paper in our study is Harvey [80]. The authors extracted firmware images from the update packages of the PLC vendor's website and the PLC memory through the JTAG interface of the PLC processor. Then they identified the subroutines that allowed them to modify the inputs and outputs for the PLC. To upload the firmware, they rely on vulnerabilities to protect remote firmware update functions (AL0, AL1 or AL2 in Fig. 3.4) or directly through the JTAG interface (AL3 in Fig. 3.4).

Attacks that Target the PLC I/O. The two most cited papers targeting the inputs and outputs of the PLC focus on how to use input or output physical signals as covert channels to exchange information. For example, *PHYCO* [190] proposes a method where two compromised PLCs can talk to each other, even when a firewall exists between them. One PLC can send an output to increase power generation, and the other PLC can read the increased generation as a bit of information. Adversaries can also use the PLC I/O to exfiltrate data. *Krishnamurthy et al.* [189] illustrate how a PLC sending control commands to a motor can leak information about the status of a chemical plant.

Attacks that Target the PLC Runtime. An illustrative example of runtime attacks is provided by *Abbasi et al.* [145], by introducing Pin Control Attacks. This attack involves tampering with the SoC configuration within the PLCs, aiming to disrupt the communication between the PLC runtime

and the hardware peripherals. By implementing this attack, the adversary severs the connection between the runtime software and the physical world, allowing them to manipulate the control data within the actual physical process.

Runtime attacks can also target the availability of the system; for example, *Gjendemsjø et al.* [147] proposed an XML Bomb Attack that causes the PLC runtime to crash by modifying an XML file.

Attacks that Target the PLC Operating System. There are not a lot of papers focusing on attacking the OS of PLCs. In their comprehensive study of the attack surface of a PLC, *Abbasi et al.* [144] touch upon an intriguing aspect of PLC security: the vulnerability of the Siemens Adonis Real-Time Operating System. Although their primary focus lies on the bootloader security of Siemens PLCs, they also explore security weaknesses inherent in the Adonis RTOS.

Attacks that Target the PLC Memory. Some network or control logic attacks target the memory of the PLC as part of their infection chain by modifying memory blocks or by getting memory dumps. The most cited efforts focused on developing worms stored in memory [203, 202].

3.6 Overview of Defenses

Defenses Focusing on Network Inspection. Since most of the attacks in the literature focus on network exploits, it is natural to expect several defenses in the network as well. Some of the first and most popular network intrusion detection systems proposed for industrial networks model the highly-periodic network traffic as a deterministic finite automaton (DFA) and propose to raise alerts when the network traffic does not follow the learned DFA [93]. Other approaches for protecting the networks of PLCs include adding cryptographic protections and machine learning for anomaly detection and prevention [98].

Control Logic Defenses. Ensuring that the control program running on the PLC is verified and correct is an old area of research, as it is not necessarily related to security but focuses on providing safety guarantees for the operation of a process [114]. This area has received renewed attention in the security community. An example of this line of research is illustrated by *McLaughlin et al.* [152], which proposed a Trusted Safety Verifier approach that verifies the control logic code

through a bump-in-the-wire before it is executed by the PLC.

CPU Defenses. One way to detect attacks against the CPU module is to monitor all communications being exchanged between the modules of a PLC. WeaselBoard [136] is a backplane analysis system that forwards all inter-module traffic to an analysis system to detect a variety of attacks. This approach is more general than just detecting CPU attacks, but because it captures information directly from the CPU, we think it fits better in this category. Other papers on CPU defenses focus on resiliency. For example, *Luo et al.* [170] proposed Quad-Redundand PLC, a redundancy framework to provide resiliency after one CPU is attacked, aiming for a second CPU to keep the system running.

Firmware Modification Defenses. Bump-in-the-wire defenses are general mechanisms where an extra device is placed between the sender and the receiver. This device checks that the data or code sent to the receiver is correct. Some of the more popular firmware defenses are based on this architecture, where the proxy device is used to inspect firmware updates before they are installed in the PLC [187, 184].

I/O Defenses. The two most-cited defenses against the inputs and outputs of the PLC include a defense strategy [192] and a prevention mechanism [211]. The work of Urbina et al. [192] focuses on using physical models of the process under control to detect inconsistencies between the inputs and outputs. On the other hand, C^2 [211] focuses on preventing malicious outputs from a compromised PLC from reaching actuators. C^2 proposes a way to express and enforce security policies, and it also denies actions that violate the policy.

Runtime Defenses. ECFI [142] is a control-flow integrity monitoring runtime protection for real-time PLCs. Fundamentally, ECFI achieves this by segregating control-flow verification of the PLC runtime software from control-flow tracing, ensuring the preservation of real-time requirements crucial for PLC operations. Distinguishing itself from other control-flow monitoring systems, ECFI refrains from terminating the runtime process; instead, it promptly notifies the operator of any detected control-flow violations. This approach strikes a balance between protection and timeliness, allowing for effective threat detection without disrupting the overall system func-

tionality. On the other hand, Ghostbuster [146] serves as a defense mechanism designed to identify Pin Control Attacks targeting PLC runtimes [145]. Implemented as a kernel driver, Ghostbuster operates in two distinct modes to ensure comprehensive protection. In Kernel mode, it actively detects alterations made to the SoC debug registers, enabling the detection of elusive "Ghost in the PLCs" I/O intercepts. On the other hand, to thwart user-mode pin control attacks, Ghostbuster employs a distinct strategy. It diligently monitors the SoC configuration of the Pin Control Subsystem, constantly searching for configuration violations that could sever the connection between the PLC runtime and the physical world. By adopting these proactive measures, Ghostbuster effectively fortifies PLC systems against Pin Control Attacks and reinforces the security of the runtime environment.

OS Defenses. Another IT protection being considered is malware detection at the OS level [86], which requires new hardware (Power Debug PRO) and could be circumvented using data hooking or gaining kernel privilege level.

Memory Defenses. Applying protections like ASLR is now common in most IT infrastructure, but similar approaches are being explored for PLCs. For example, *Robles-Durazno et al.* [166] proposed Optimized Datablocks, an approach in which the allocation of the datablock data is randomized such that the attacker does not know its exact location, making attacks more difficult as with other moving target defenses.

3.7 Research Gaps

We now focus on identifying relevant insights, gaps, and recommendations for future work by analyzing the data from Tables 3.1 and 3.2.

Most of the Attacks Require Zero Environment Knowledge. We found that 82% (97/119) of the attacks that may cause either *limited* (1) (36% (35/97)), *substantial* (2) (56% (54/97)), or *severe* (3) (8% (8/97)) damage require *zero knowledge* (O) of the environment in which the PLC lives, following the description presented in Sec. 3.2.1 and the criteria discussed in Sec. 3.4.3. This may imply that adversaries can potentially launch non-trivial attacks against power grids, chemical plants, water treatment plants, etc. by targeting PLCs without having to invest time performing

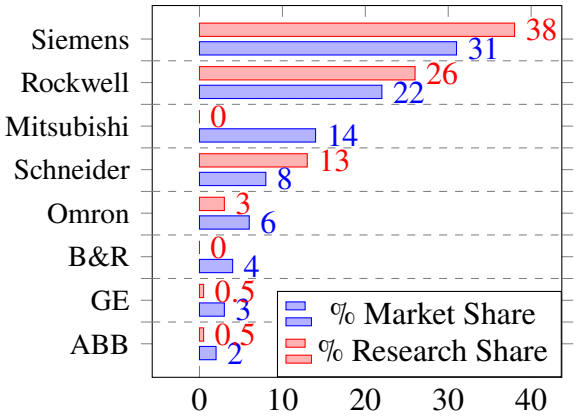

Figure 3.5
Comparison of Research Share vs Market Share.

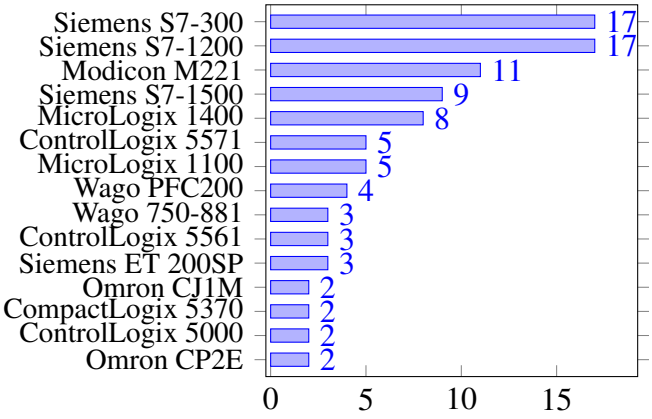

Figure 3.6
The 15 Most Common PLC Models.

reconnaissance or learning the specifics of actuators, sensors, etc. Moreover, we were able to identify that only a few defense methods, e.g., *Formby et al.* [2] and *Bellettini et al.* [138], make use of such important environmental information. We therefore recommend that future defenses make use of environment knowledge in their strategy to increase their effectiveness.

The Security of Important PLC Brands Has Not Been Explored. We found that the security of some important PLC platforms widely used in practice has been ignored. As shown in Fig. 3.5, the *market* share percentage of important PLC manufacturers such as Mitsubishi, Omron, ABB, and GE is considerably higher than their *research* share, e.g., the number of papers explicitly using them for attack/defense evaluation purposes. For example, even though Mitsubishi PLCs account for 14% of the global market, they contribute to 0% of the research share in our review (they only

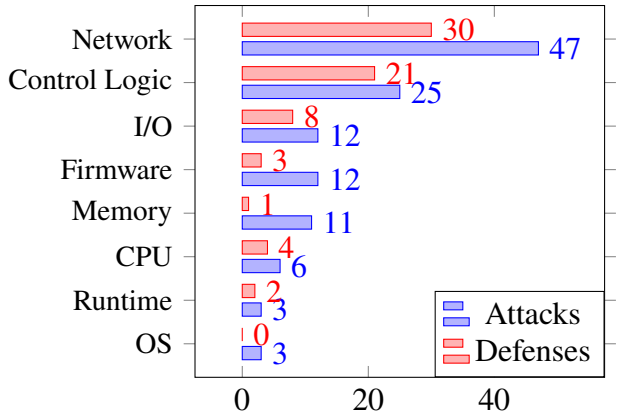

Figure 3.7
Defense vs Attack Methods per Target Component.

appear once in our Grey literature survey [212]).

This means that there are potentially thousands of PLCs deployed in the world whose security has not been explored or may not have been adequately understood. We therefore recommend future lines of work specifically addressing these devices, exploring the effectiveness of existing attacks and defenses as discussed in this work, as well as the introduction of newer security methods tailored for them.

Lack of Defenses at the Recovery Stage. We found that most defense methods are designed for the Prevention (PR) and Detection (DE) stages discussed in Sec. 3.4.7, 47% (33/70) and 47% (33/70) out of 70 respectively, whereas the Recovery (RE) stage accounts for only 6% (4/70). This means that in the event of a successful attack there are limited options to recover and bring the PLC back to operation.

Most Attacks and Defenses are Evaluated on a Small Subset of PLCs. Our results show that the top 5 most common PLC models in our literature review (as shown in Fig. 3.6) account for 40% of all studies. This may result in a narrow understanding of PLC security that excludes the rest of the PLC models in the market. Additionally, we found that 80% (95/119) of attack methods and 81% (57/70) of defense methods were evaluated using a single HardPLC model (○). This means that most attack and defense methods are shown to work with a single HardPLC model, making it unclear whether or not the defense method can be generalized to other PLC models and

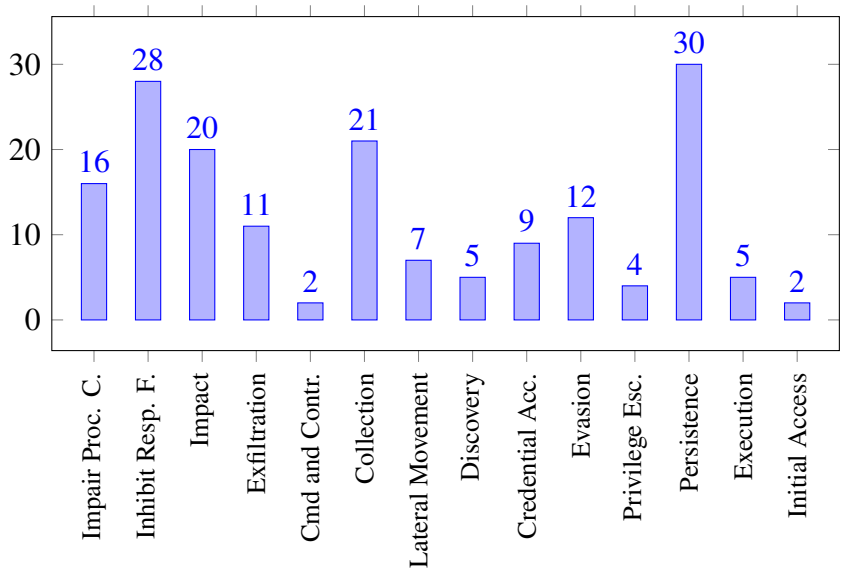

Figure 3.8
Attack Methods according to their Tactic.

manufacturers. Therefore, we recommend that future research include an evaluation of multiple PLCs.

Important Tactics have Little to No Research. Our ICS^2 Matrix includes 14 Tactics. However, as Fig. 3.8 shows, most of the attack methods focus on Impair Process Control, Inhibit Response Function, Exfiltration, Collection, and Persistence. On the other hand, important Tactics like Command and Control, Lateral Movement, Evasion, Credential Access, and Initial Access have not been investigated.

One reason that might explain why these tactics have not been explored in previous research is that they are too specific or custom-fit. For example, the Command and Control tactic includes techniques such as *Commonly Used Port* and *Standard Application Layer Protocol*, which depend on specific network protocols and ports. This specific focus on network ports might limit the scope of research that can be carried out for this tactic.

Most Mitigation Strategies have Little to No Research. We found defense methods that match 17 Mitigation categories. However, as shown in Fig.3.9, the majority of these defenses fall into the Network Intrusion Prevention, Encrypt Network Traffic and Control Logic Verification categories. The remaining 14 Mitigation categories have only 4 or fewer defenses. These include important cat-

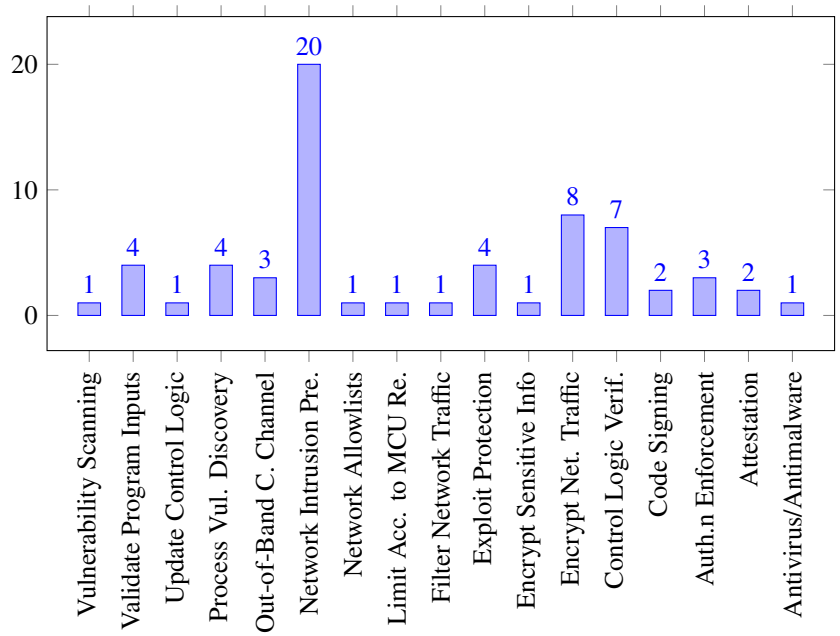

Figure 3.9
Defense Methods per Mitigation Category.

egories like Antivirus/Antimalware, Vulnerability Scanning, and Attestation, which are especially important given the rise of ICS malware such as the ones described in Appendix 3.10.6.

Weaknesses of State-of-the-Art Defenses Based on the results shown in Tables 3.1 and 3.2, we identified the following three major weaknesses when it comes to defending PLCs: **1)** No ransomware detection. Even though researchers have shown that ransomware attacks against PLCs are possible [169, 206] and there are multiple documented ransomware attacks against ICS, there is no available research focused on how to detect and stop PLC ransomware. To address this research gap, future research should introduce new defense methods that take advantage of state-of-the-art malware and ransomware detection techniques such as sandbox detection [213] and static and dynamic analysis [214]. **2)** No web-based malware detection. Research shows that attackers target PLCs' web interfaces[215, 15] and that it is possible to compromise PLCs via their web applications [216]. However, there is no available research focused on how to detect and stop malware targeting PLCs' web interfaces. **3)** No Exfiltration over Covert Channel Detection. This is one of the new techniques that we introduced in our ICS^2 Matrix (Fig. 3.13), which includes methods such as *PHYCO* [190]. Introducing this technique lays the groundwork for identifying the

need for mitigations against such techniques. Currently, there is no known mitigation for this type of attack. Indeed, in 2014 *Garcia et al.* wrote: "There has been no detection solution capable of identifying such hidden communications in the physical power system," [190] which still holds true.

3.8 Discussion

We now discuss research challenges that have not attracted enough attention and may become relevant as PLCs evolve.

Reproducible Research. Based on our analysis and the results presented in Tables 3.1 and 3.2, few defense and attack methods provide publicly available research artifacts. During our literature review, we searched for research artifacts for each paper. We searched the paper itself on Google and the author's website. Using this method, we were able to find the source code for 19 papers (only 16%). This limits the reproducibility of attacks and defenses. In an attempt to find the artifacts for papers without openly available code, we contacted 91 authors via email requesting their research artifacts, and we received 16 responses (17.6%). Ultimately only 3 shared a research artifact. The other 13 did not share the source code for the following reasons: **1)** The project was completed long ago or the first author moved on to a different institution (30%). **2)** There were funding or distribution restrictions (25%). **3)** The authors were working on it and will publish it later (15%). **4)** There were no plans to release it to the public (30%). 16% is a low number of papers with artifacts, and this does not even consider if the source code has good documentation or if these public resources are easy to run. Therefore, we encourage researchers to release their PLC security artifacts so that research can be replicated and built upon and to leverage our PLC security artifacts repository discussed in Sec. 3.1 to disseminate their artifacts. We acknowledge that given the criticality of PLCs it is not always possible to release artifacts at the time of publication if at all. For example, the US' Cybersecurity & Infrastructure Security Agency (CISA) recommends researchers "provide us a reasonable amount of time to resolve the issue before you disclose it publicly [217]."

A second challenge that compounds the problem of lack of reproducible research is the absence

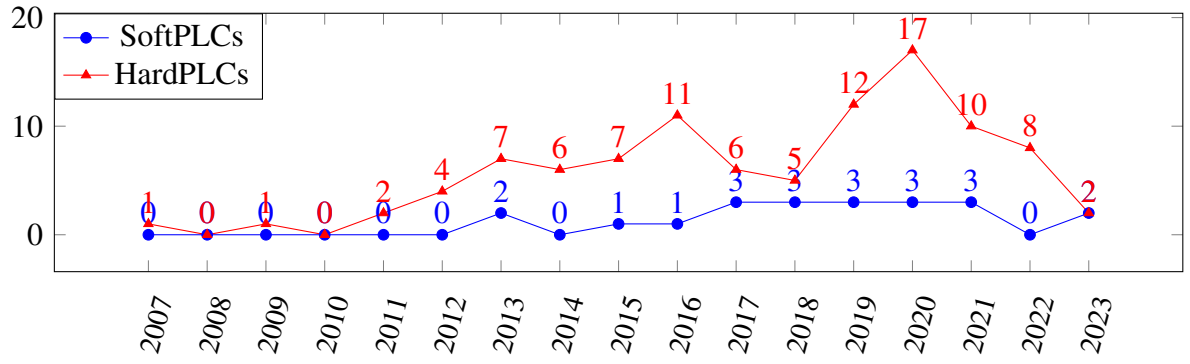

Figure 3.10
Papers evaluated using SoftPLCs vs HardPLCs.

of standardized performance metrics. Based on the results in Tables 3.1 and 3.2, few defense methods reported detailed PLC overhead or effectiveness metrics as described in Sec. 3.4.7. This makes comparing and building upon previous research almost impossible. To overcome this challenge we recommend that future defense methods report detailed quantifiable metrics. First, papers that propose methods such as Intrusion Detection Systems (IDS) should report a complete and detailed confusion matrix. The work by *Salehi et al.* [185] provides an excellent example. Second, future research on defense methods should report detailed and quantifiable benchmark metrics, e.g., memory overhead. Existing benchmarks such as *BenchmarkIoT* [218] can be customized for PLCs to overcome this challenge.

Transition from HardPLCs to SoftPLCs. As discussed in the introduction, PLCs are going through a paradigm shift with support for new protocols and new functionalities. Support for SoftPLCs appears to be increasing. While SoftPLCs currently have a minimal market share, large manufacturers like Siemens and Rockwell are preparing hardware-agnostic products[219]. This paradigm shift has also reached the scientific community. As shown in Fig. 3.10, PLC security research has diverged into two different strands: evaluations that use HardPLCs and evaluations that use SoftPLCs.

Considering this trend, future research should be aimed at addressing the following challenges: **1)** Developing transitional defense methods that secure both HardPLCs and SoftPLCs. For example, developing *bump-in-the-wire* solutions that are compatible with HardPLCs and SoftPLCs. This may

involve evaluating research on both HardPLCs and SoftPLCs. **2)** Investigating defense mechanisms available for SoftPLCs that were previously unavailable for HardPLCs. For example, HardPLCs' proprietary nature limited their access control and encryption capabilities by incorporating weak encryption protocols. SoftPLCs' less opaque architecture provides the opportunity to solve this problem. **3)** Investigating both attack and defense methods that are possible only with SoftPLCs. For example, CODESYS' SoftPLCs can be connected to the cloud [220], which opens new ways to collect data that can be used to train machine learning-based Intrusion Detection Systems.

3.9 Related Work

SoKs on PLCs. To the best of our knowledge, the work by *Sun et al.* [221] is the only SoK focused on attacks and defenses on the control logic of PLCs. However, our work is different in two ways. First, *Sun et al.* focuses on PLCs' control logic, while we take a more comprehensive approach that includes 8 additional PLC components. Second, our SoK includes papers with practical evaluations, while *Sun et al.* focus on formal or theoretical research.

PLC Honeypots. While previous work includes approaches for PLC honeypots [222, 15], their focus is intelligence gathering rather than attacking PLCs.

Fuzzing and Binary Reverse Engineering. There is also noticeable research on fuzzing and reverse engineering for ICS and PLCs in the literature [223, 224, 225, 226, 227]. Our focus, however, is not on software analysis or patching but on outlining specific attacks.

Embedded Controllers. While they have received most of the attention in the security literature, PLCs are not the only embedded equipment in ICS. Remote Terminal Units (RTUs) and Intelligent Electronic Devices (IEDs) are even more prevalent in energy transmission systems and substation automation [228, 229, 230]. We need further research into the security of these other embedded controllers.

3.10 Chapter Conclusion

In this dissertation chapter, we provide a novel threat taxonomy for Industrial Control Systems (ICS). We also pointed out research gaps that should be tackled in the future so that the security of PLCs can be better understood, thus helping avoid future attacks against ICS and PLCs. We hope this

systematization is useful to newcomers to the field as well as experienced PLC researchers looking to contextualize their work. As a part of Future Work, we are developing an experimental testbed as discussed in Sec. 3.2.1. This way, different attack and defense strategies can be replicated to provide experimental evidence on their effectiveness, deployability, and robustness, thus ultimately complementing the results provided in this work.

Appendix A:

3.10.1 Scientific and Grey Literature Resources

The seven digital libraries queried during our scientific literature review are as follows: ACM Digital Library[7], arXiv[8], dblp[9], Google Scholar[10], IEEExplore[11], USENIX Papers Search[12], and NDSS Symposium Search[13].

The three main sources we queried during our grey literature review are as follows: Digital Bond Archives[14], InfoconDB[15], and Google[16].

Search keywords.

```
TC-1: "plc", "programmable␣logic␣controller", "scada", "cps",
TC-2: "cyber-physical␣system", "cyber␣physical␣system", "iiot",
TC-3: "industrial␣internet␣of␣things", "industry␣4.0",
TC-4: "industrial␣control␣system", "ics", "embedded␣system",
TC-5: "attack", "threat", "vulnerability", "defense"
```

3.10.2 MITRE Technique and Sub-Technique Model Example

As an example of how our ICS^2 Matrix can be used to categorize ICS-specific attacks proposed by researchers, we will have a look at the work by *Krishnamurthy et al.* [189] They proposed an attack

[7] https://dl.acm.org/
[8] https://arxiv.org/
[9] https://dblp.org/
[10] https://scholar.google.com/
[11] https://ieeexplore.ieee.org/Xplore/home.jsp
[12] https://www.usenix.org/publications/proceedings/
[13] https://www.ndss-symposium.org/
[14] https://dale-peterson.com/digital-bond-archives/
[15] https://infocondb.org/
[16] https://www.google.com/

strategy where a malware-infected PLC can exploit the acoustic emissions generated by a motor controlling a valve in a feedback control loop within an ICS. This covert acoustic channel enables the malware to secretly transmit sensitive information, such as proprietary controller parameters and system passwords, to a remote receiver. This attack does not disrupt the stability, performance, or signal characteristics of the closed-loop process, making it stealthy and effective at exfiltrating data from the compromised ICS.

We categorized this attack under the *Exfiltration* tactic but could not find a fitting technique. Thus, we propose the addition of the technique *Exfiltration over a Covert Channel*. This way of data exfiltration bypasses the techniques *Filter network Traffic* and *Data Loss Prevention*, as covert channels utilize unintended means of communication by definition. To mitigate covert channels requiring the propagation of wireless signals, *Minimize Wireless Signal Propagation* represents a defense, but due to the different forms that covert channels can take, complete mitigation proves difficult, which puts this technique in the category *Mitigation Limited or Not Effective*. Table 3.3 shows the new technique definition motivated by the example above. This technique definition follows MITRE's official guidelines [8].

3.10.3 PLCs' Larger Context and Underlying Architecture

In this Appendix, we expand on the environment in which PLCs operate. Specifically, we use the Purdue model [231] to describe the network architecture of an ICS process and where PLCs fit in.

As we discuss in Sec. 3.1, PLCs control physical machines or actuators, such as pumps. However, these actuators do not exist in a vacuum. They are one component of many that work together to complete a larger industrial process, for example, a water treatment process. Figure 3.11 depicts a sample of a water treatment process ICS using the Purdue Model.

3.10.4 PLC Memory Blocks

PLCs have specialized memory blocks that store different types of data. These blocks are made available to applications that perform operations like program upload and download [1] and can be classified as:

Data Item	
Name	Exfiltration over Covert Channel
Tactic	Exfiltration
Data Sources	
Description	Adversaries may attempt to exfiltrate data via a covert channel such as analog emissions of physical instrumentation. For example, actuators, sensors, and mechanical structures. These analog emissions can be acoustic or electromagnetic. In some circumstances, the adversary needs to have physical access to the ICS to measure the analog signal using an antenna, for example. The physical medium or device could be used as the final exfiltration point or to hop between otherwise disconnected systems.
Asset	Field Controller/RTU/PLC/IED
Defense Bypassed	Data Loss Prevention, Filter Network Traffic
Contributor	Efrén López Morales
Procedure Example	Krishnamurthy, Prashanth, et al. "Process-aware covert channels using physical instrumentation in cyber-physical systems." [189] IEEE Transactions on Information Forensics and Security 13.11 (2018): 2761-2771.
Mitigation	Minimize Wireless Signal Propagation, Mitigation Limited or Not Effective
Detection	

Table 3.3
Proposed technique definition of Exfiltration over Covert Channel according to MITRE ATT&CK: Design and Philosophy [8]

Data Blocks (DB). These blocks are used to store data that will later be used by a program. Different data types can be stored (e.g., Boolean, byte, integer) [232].

System Data Blocks (SDB). These blocks contain PLC configuration information [10, 232], listing PLC model, firmware version, IP address, and information about the attached add-ons (e.g., communication processors, frequency converters). SDBs are automatically created and compiled by the PLC and cannot be modified by the user.

Organization Blocks (OB). These blocks are the interfaces between the operating system and the user program [116, 233]. OBs execute when events occur (e.g., at CPU startup, clocked executions, errors, hardware interrupts). Several other OBs serve specific roles.

Function Blocks (FB). Finally, FBs hold standard executable code blocks, which are written in any of the IEC 61131 standard programming languages [10, 232].

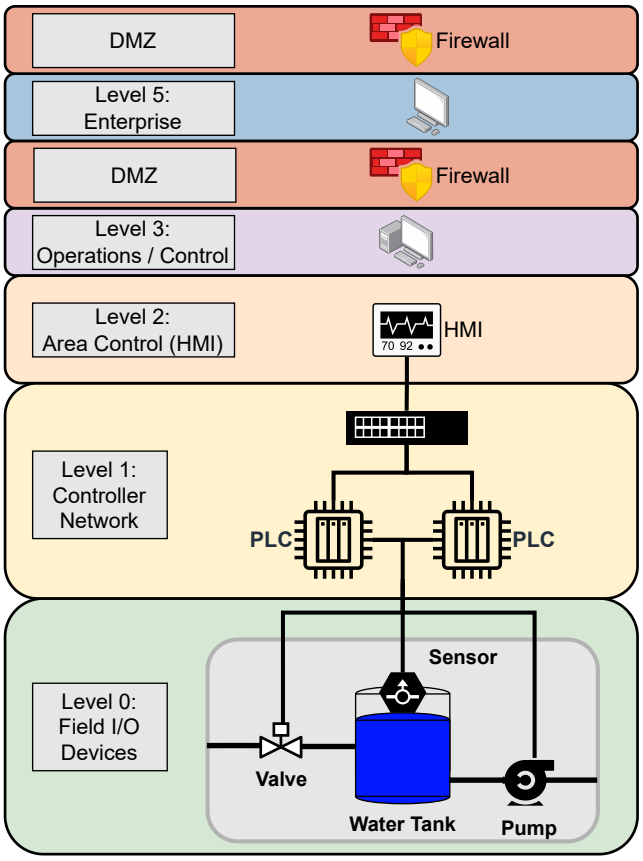

Figure 3.11
Example of an ICS process with PLCs in Level 1: Controller Network as defined by the Purdue Model. Based on [5, 6].

3.10.5 PLC Built-in Security Features

Access Control. Most PLCs include built-in access control features. For example, Siemens and Allen-Bradley PLCs can be configured with a password that restricts changes to the CPU configuration[234, 235]. However, these built-in features have been shown to be ineffective[84].

Encryption. Some PLCs also include encryption features for memory blocks, for example, Siemens allows the encryption of specific memory blocks [236]. Another encryption functionality involves the use of hashing to detect changes. Some Rockwell and Siemens PLCs support hashing [237]. Additionally, some of the PLC Industrial Ethernet protocols incorporate built-in encryption. However, like the access control features, they have been shown to be vulnerable [116].

Operational Modes. PLCs have different operational modes intended for different scenarios.

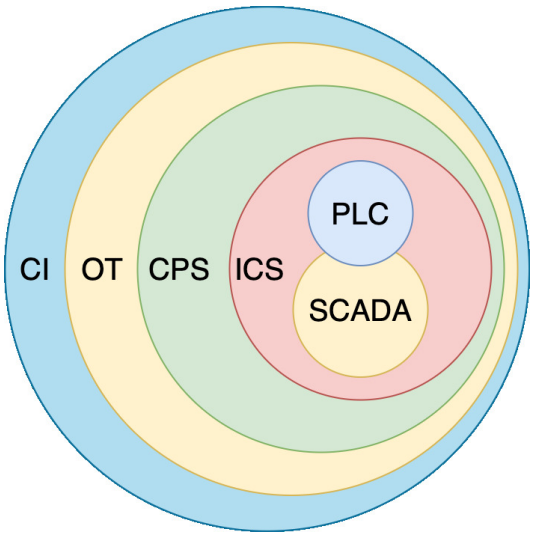

Figure 3.12
Relationship between Critical Infrastructure (CI), Operational Technology (OT), Cyber-Physical Systems (CPS), Industrial Control Systems (ICS), Supervisory Control and Data Acquisition (SCADA), and PLCs. Based on [7].

For example, to modify the control logic program of a production PLC, it is necessary to change or *switch* the PLC operational mode from *Run* to *Stop* and then to *Program* [238]. In practice, there exist two ways to switch the operational mode of a PLC: either to use the PLC management software, e.g., SIMATIC STEP 7 for Siemens PLCs, or to manually switch the operational mode using a physical *key* inserted in the PLC chassis, which overrides the software option just described.

Overall, the most common operational modes are:

- *Run Mode.* This mode is used to execute the control logic program of the PLC. Input from sensor devices is monitored, and output is sent to actuator devices. Ideally, a production PLC should always be set to *Run* [238, 239].
- *Stop Mode.* In this mode, the PLC stops reading inputs and stops the control logic program execution. Typically, a PLC must be switched to *Stop* before it can be re-configured by means of the *Program* mode [240, 238].
- *Program Mode.* In this mode, the PLC control logic program is loaded, modified, or deleted. While in this mode, all outputs from the PLC are stopped [238, 239].

Disabling Unused Protocols. Some PLCs have the option to disable network services to reduce

unnecessary attack surfaces. For example, Siemens PLCs allow for the integrated web server to be disabled [241], preventing the exploitation of web server-based vulnerabilities [242, 102].

Monitoring HMI Data. HMIs are useful to visualize trends about the performance of PLCs, as they can plot the PLC Scan Cycle, the uptime, and shut down and restarts, which can be helpful to detect an ongoing attack [237].

Access Level	Vector
AL3: Physical	• JTAG Port • USB Port • Key Switch • Backplane • Memory Card
AL2: Fieldbus	• PROFIBUS • DeviceNet • RS-232 Serial • EtherNet/IP
AL1: LAN	• GOOSE • Workstation • SoMachine (Schneider) • TriStation (Schneider) • LS Proprietary (LS Electric)
AL0: Internet	• Modbus TCP • DNP3 • IEC 104 • OPC UA • EtherNet/IP • S7comm • Webserver (HTTP/HTTPS) • SNMP

Table 3.4
Attack Vectors per Access Level.

3.10.6 Real-World PLC Attacks

We summarize real-world attacks that showcase the pressing need to improve the security of PLCs.

Stuxnet. Stuxnet is the first-ever documented malware for PLCs. At the time, it set itself apart from previous malware by showing a high level of sophistication, a deep understanding of industrial processes, and the use of four *zero-day* exploits [10]. After compromising a computer with the software for programming PLCs, the malware uploaded its malicious control program to the target

PLCs. In particular, the attack targeted Siemens 315 and 417 and made them damage centrifuges while reporting that everything was normal [243].

Triton. This malware, also known as TRISIS and HatMan [21, 244] was identified in 2017 after a petrochemical facility in Saudi Arabia was shut down. After compromising an engineering workstation, Triton was able to launch a dropper (trilog.exe) to deliver backdoor files to Safety Instrumented System (SIS) PLC. The first backdoor file was a 0-day exploit that allowed the attackers to inject the second file into the PLC's memory. With a program in the memory of the PLC, the attackers could have control of the device. The attackers were unable to take full control of the system because an error in the PLC caused a system shutdown.

Pipedream Toolkit. At the time of writing, Pipedream (also known as Incontroller) is the latest documented malware that specifically targets PLCs [245]. This is not a single purpose malware but a modular framework that includes multiple exploits that target different PLCs. Once the attackers compromise a computer in the control network, Pipedream can be used to scan and compromise Scheider Electric PLCs, OMROM Sysmac NEX PLCs, and Open Platform Communications Unified Architecture (OPC UA) servers. Pipedream is believed to have been developed by a nation state [246].

Crashoverride. Also known as Industroyer [247, 230], Crashoverride is believed to have caused the power outage in Ukraine's capital in December of 2016 [248]. A sophisticated malware designed to disrupt ICS networks used in electrical substations, it targeted the Open Platform Communications Data Access protocol or OPC-DA for short, which defines how data can be transferred to and from PLCs.

3.10.7 Excerpt of the ICS^2 Matrix

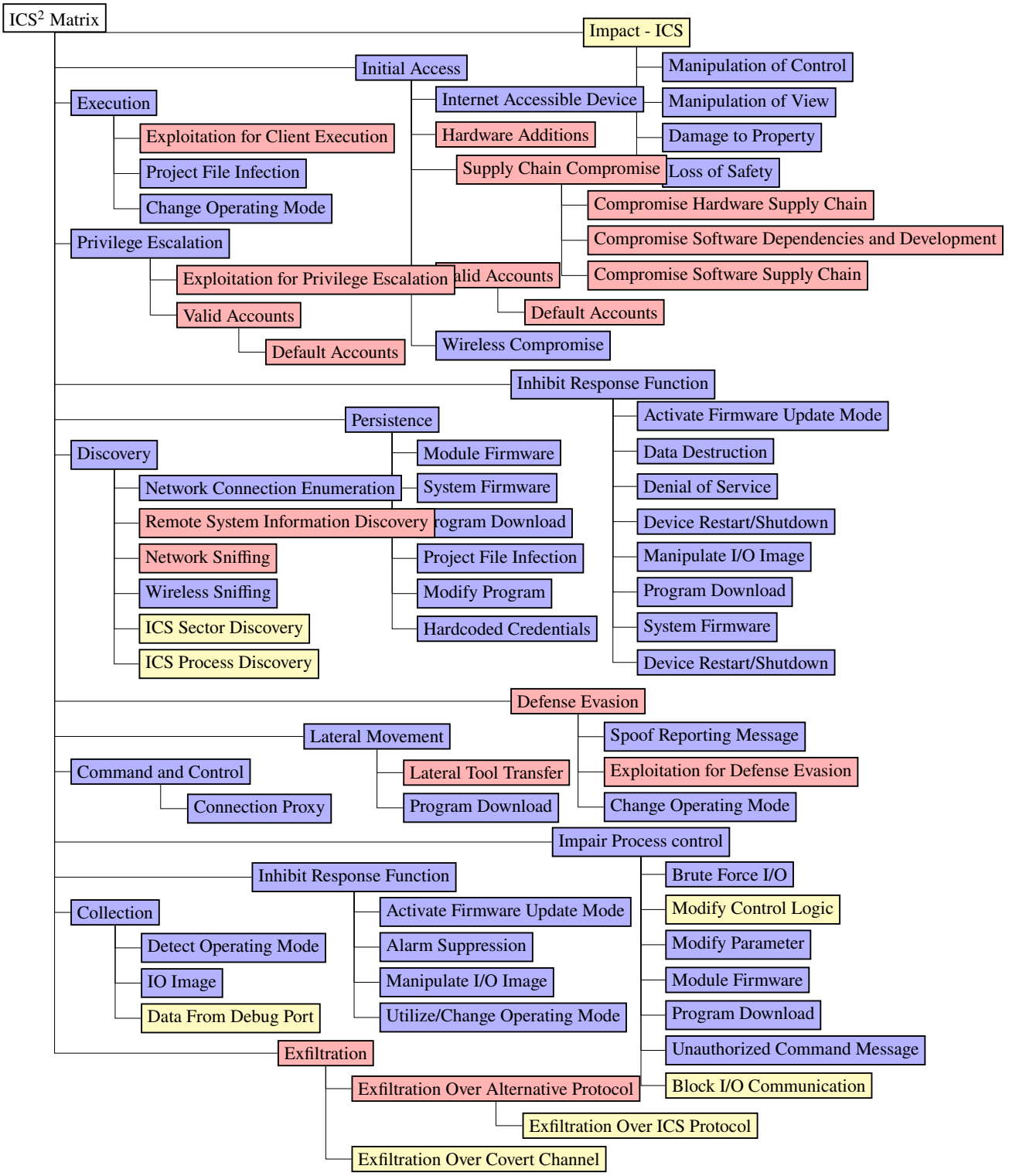

Figure 3.13
Condensed version of the ICS^2 Matrix. It includes nodes from the MITRE ATT&CK for ICS Matrix (Blue), the MITRE ATT&CK Enterprise (Red) and our additions (Yellow).

CHAPTER IV: SATELLITE HONEYPOT

Abstract

Satellites are the backbone of several mission-critical services, such as GPS that enable our modern society to function. For many years, satellites were assumed to be secure because of their indecipherable architectures and the reliance on security by obscurity. However, technological advancements have made these assumptions obsolete, paving the way for potential attacks, and sparking a renewed interest in satellite security. Unfortunately, to this day, there is no efficient way to collect data on adversarial techniques for satellites, which severely hurts the generation of security intelligence.

In this dissertation chapter, we present HoneySat, the first high-interaction satellite honeypot framework, which is fully capable of convincingly simulating a real-world CubeSat, a type of Small Satellite (SmallSat) widely used in practice. To provide evidence of the effectiveness of HoneySat, we surveyed experienced SmallSat operators currently in charge of active in-orbit satellite missions.

Results revealed that the majority of satellite operators (71.4%) agreed that HoneySat provides realistic and engaging simulations of CubeSat missions. Further experimental evaluations also showed that HoneySat provides adversaries with extensive interaction opportunities by supporting the majority of adversarial techniques (86.8%) and tactics (100%) that target satellites. Additionally, we also obtained a series of real interactions from actual adversaries by deploying HoneySat on the internet over several months, confirming that HoneySat can operate covertly and efficiently while collecting highly valuable interaction data.

4.1 Introduction

Artificial satellites are complex devices designed to withstand outer space conditions. They serve multiple purposes or types of *missions* that include position and navigation, e.g., the Global Positioning System (GPS) constellation [249], Earth observation, e.g., the Sentinel constellation [250], and broadband internet service, e.g., the Starlink constellation. In addition, spacecraft can range from being as massive as thousands of kilograms, such as the International Space Station

(ISS) to one kilogram CubeSats. As a result, the software and hardware components that make up a specific spacecraft vary greatly. Likewise, satellite missions' vary a lot depending on the owner's budget. For example, university missions are smaller as they have a limited budget. A cyberattack on a satellite or satellite constellation (group of satellites) could have disastrous consequences on a global scale, which is difficult to comprehend. Such an attack could lead to the cessation of air traffic and widespread communication blackouts. It could also cause food shortages and the freezing of financial transactions [251]. Furthermore, such an attack could exacerbate the Kessler Syndrome [252], a scenario in which collisions between satellites and debris in orbit create a cascade effect, generating even more debris, jeopardizing future satellite launches and operations.

In this increasingly vulnerable environment, the probability of a successful satellite cyberattack continues to rise. This is driven by three key trends [253]: first, satellite deployments have increased at an unprecedented pace. For instance, while an average of 82 launches took place between 2008 and 2017, as many as 197 launches occurred in 2023 alone, each typically carrying multiple satellites [254]. This surge is partly fueled by the rise of cheaper commercial off-the-shelf (COTS) components and the availability of more Space Launch Vehicles (SLVs), which makes access to orbit affordable for smaller institutions, such as universities [255]. Second, ground station technology has become significantly more affordable (and sometimes open source), greatly lowering the communication barrier with satellites [256]. Thus, a broader range of malicious actors can now communicate with satellites. Third, satellite engineers and operators continue to rely on *protocol obscurity* practices, such as hiding specialized knowledge about the transmission protocol implemented on satellites [22].

Although satellite security research has gained increased attention [22, 257, 258, 259], the relationship between outer space and cyberspace remains poorly understood [260]. At the same time, space-focused threat intelligence remains sparse, and our current methods for identifying tactics, techniques, and procedures (TTPs) used against these critical systems are limited. MITRE ATT&CK currently tracks 152 threat groups but shows only one that targets satellites explicitly [261], highlighting a significant data gap. Although the volume of reported cyber incidents

in the space sector has grown, these reports rarely provide sufficient detail [251]. As a result, the security community has limited visibility into adversarial activity aimed at space infrastructure.

Honeypots' ability to collect real-world cyberattack data makes them an ideal solution to this problem [262]. A honeypot is a decoy computer system intended to lure and entice malicious actors to interact with it [263]; all the while, the honeypot logs all the interactions the attackers make. This log data can later be analyzed to discover new and existing TTPs.

Since the release of the first honeypot, the Deception Toolkit, in 1997 [264], a wide range of honeypots with ever-increasing capabilities have been introduced. These honeypots are used by universities, companies, and nation-states worldwide [265, 266, 267, 268, 269]. Honeypots are also used to deter malicious actors from attacking different types of systems, from industrial control systems [269] to social media platforms [270]. However, as of the time of writing, *there is no space-sector specific honeypot* in the literature.

In this dissertation chapter, we present the first satellite honeypot in the literature, *HoneySat* to attract and analyze adversaries who attack space infrastructures over the Internet, a commonly observed threat vector [271, 272, 273, 274]. HoneySat is a modular, high-interaction honeypot framework that realistically simulates a complete satellite system (ground infrastructure and Satellite). Specifically, HoneySat simulates *Small Satellites* or *SmallSats* which are spacecraft with a mass of less than 180 kilograms[275]. CubeSats for example, are SmallSats. Additionally, as part of HoneySat, we developed the *Satellite Simulator*, a Python project to provide generic simulation functionality for users to populate satellite honeypots with believable data. Unlike existing Cyber-Physical Systems (CPS) honeypots for Industrial Control Systems (ICS) or Unmanned Aerial Vehicles (UAVs), HoneySat simulates telemetry and telecommand of a real satellite and simulated payload behavior. This capability enables the creation of complete testing environments for satellite software that integrate a variety of simulated subsystems and sensors. For instance, Satellite Simulator can simulate orbital mechanics (e.g., position tracking, attitude adjustment) and electrical power systems (e.g., power generation, consumption, and distribution) alongside other subsystems.

We leveraged our framework to create honeypots of real-world CubeSat missions. Our results, backed by our survey of satellite operators, show that HoneySat's simulation is highly realistic. Our framework is able to simulate an entire real satellite mission, provides realistic telemetry, and supports real telecommands.

In summary, we make the following contributions:

- A novel framework, HoneySat, a high-interaction, extensible honeypot for small satellites (Sec. 4.4).
- The Satellite Simulator, that simulates the physical processes, sensors, and subsystems necessary for a realistic satellite honeypot [276].
- The results of a survey of experienced satellite operators that provide valuable insights into how realistically our honeypot performs, as well as experimental evidence demonstrating that the HoneySat framework can simulate small satellites, collect rich real-world interaction data, and be customized for multiple satellites (Sec. 4.5).

This work was a joint effort with equal contribution which resulted in a publication titled "HoneySat: A Network-based Satellite Honeypot Framework [276]". For the purposes of dissertation delineation, I (Efrén Darío López Morales) contributed the design of the threat model, and the high-level design of the honeypot. My collaborator, Ulysse Planta at the CISPA Helmholtz Center for Information Security, contributed the detailed technical design, and the implementation of the system. We coordinated closely during evaluation, experimentation and writing.

4.2 Background

This section lays out key background concepts that are relevant to satellite honeypots. For honeypots: a description of the different existing types (Sec. 4.2.1), as well as the current state-of-the-art (Sec. 4.2.2). For satellites: their operation (Sec. 4.2.3), their architecture (Sec. 4.2.4), existing Protocol Ecosystems (Sec. 4.2.5), and tactics, techniques, and procedures (TTPs, Sec. 4.2.6).

4.2.1 Types of Honeypots

Honeypots are categorized by interaction levels according to the interaction opportunities they provide. The two main types of honeypots are low-interaction and high-interaction.

Low-Interaction Honeypots. These honeypots offer minimal interaction, simulating real systems through scripts or finite-state machines. Their advantages are ease of setup and maintenance due to low resource consumption, and a reduced risk of adversarial takeover. However, they provide limited interaction opportunities to adversaries which limits the interaction data they provide. Low-interaction honeypot examples include Conpot [277] and Honeyd [278].

High-Interaction Honeypots. These honeypots offer extensive interaction opportunities via emulation or advanced simulations [278]. Their main advantage is providing adversaries with almost limitless interactions, enabling them to provide extensive interaction data. However, they pose a high risk of adversarial takeover as adversaries have more opportunities to hijack the honeypot [279]. High-interaction honeypot examples include Cowrie [280] and HoneyPLC [269].

4.2.2 Honeypot's State of the Art

The literature on honeypots includes hundreds of implementations that simulate a diverse set of computer systems [281, 282]. From classic implementations that simulate a host's TCP/IP stack, such as Honeyd [278], to modern approaches that integrate social media applications, such as HoneyTweet [270]. However, there is no satellite honeypot in the literature. In the absence of a satellite honeypot, we now examine the honeypot approaches most related to satellites: Industrial Control Systems (ICS) [283, 14] and Unmanned Aerial Vehicles (UAV), a.k.a., *drones* summarized in Table 4.1.

Satellite systems like ICS must be aware of some physical process, e.g., its position, the sun's position, etc, via *sensors* to acquire data about the physical world. Several ICS honeypots have simulated these physical processes. For example, ICSnet [284] and HoneyICS [285] simulate several components such as programmable logic controllers (PLCs), and actuators such as water valves.

HoneyDrone [286] is a UAV honeypot that integrates different simulations, e.g., Ardupilot, to

Table 4.1
Comparison of Existing Honeypots and HoneySat.

Keys: ✓ = Supported; × = Not Supported.

Honeypot/ Feature	Interaction Level	Included Protocols	Physics Sims	Extensibility
Conpot [277]	Low	9	0	✓
HoneyPLC[269]	High	3	0	✓
ICSPot [288]	High	4	1	×
HoneyICS[285]	High	2	1	✓
HoneyDrone [286]	Medium	4	1	×
HoneySat	High	4	6	✓
Addressed in Section	[276], [276]	[276], [276]	[276], [276]	4.5.5

recreate multiple attack scenarios, including the UAV ground control station. Although UAV honeypots share some similarities with satellites [287], satellite honeypots require additional physical simulations and involve a more complex operation environment, which we discuss in Sec. 4.2.3.

4.2.3 Anatomy of a Satellite Mission

We now describe the components of a satellite mission. Due to satellite missions' complexity, we explain each component and match it with one of the numbers in Fig. 4.1. Every satellite mission includes the *ground segment* from which satellite operators control the satellite and the *space segment* which includes the satellite itself.

Ground Segment ①. The Ground Segment (GS) covers the terrestrial supporting infrastructure required for a successful satellite operation. It consists of a ground station, responsible for exchanging data with the spacecraft, the computational and network infrastructure required for communication, but also the systems to operate the satellites, e.g., servers, databases, and user interfaces [22]. Several ground stations can be connected in a network and coordinated as part of one GS. The GS includes the Ground Segment Software (GSS) that helps operators schedule and send commands and visualize data that is sent back in response.

Space Segment ②. The space segment comprises a satellite or a constellation of satellites. A satellite is launched into orbit and then establishes communications with the ground segment. During regular operations, satellites may communicate through one or multiple ground stations [22].

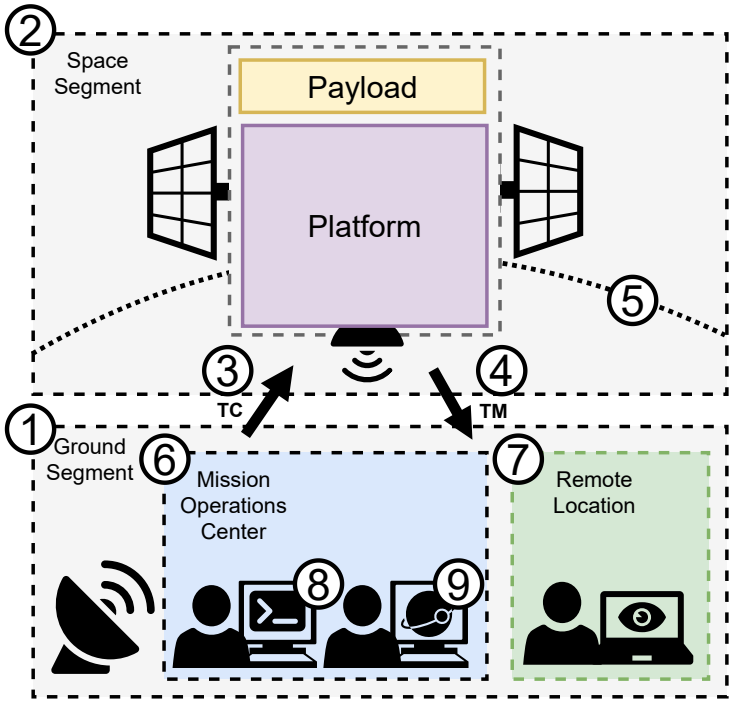

Figure 4.1
The components commonly found in a satellite mission in the context of the space and ground segments.

Telecommands (TC) ③ and Telemetry (TM) ④. The basic data flow between the space and ground segments are TC and TM [289]. TM is the data the satellite sends to the ground station which may contain the satellite's status, responses to previous commands, or payload data [289]. TCs are used to operate the satellite and are transmitted and encapsulated in a space protocol (see Sec. 4.2.5). The design and implementation of TCs varies depending on the satellite mission. From a security perspective, TCs are particularly important as an attacker that can send valid TCs to a satellite can fully take over the mission [22].

Orbital Pass ⑤. A satellite and its ground station can communicate *only* during an orbital pass. An orbital pass, or more commonly a *pass*, is when the satellite rises above a ground station's horizon and becomes available for communication. A pass's duration and timing depend on the satellite's orbit characteristics and any obstructing objects between the satellite and the ground station, e.g., mountains [290]. In some cases, such as geostationary orbit (GEO), the satellite can be above the horizon constantly. The occurrence of passes can be calculated using the spacecraft's

orbital elements. A common representation of the orbital elements is the two-line element set (TLE) [291].

Satellite Mission Operations ⑥. Satellite mission operations vary widely depending on the owning organization, budget, and technology. Nevertheless, they share some commonalities, which we now describe.

A mission's operation involves a team of operators that use GSS to operate a satellite or a satellite constellation and ensure the mission's success [292]. Operations are carried out in a Mission Operations Center (MOC), where satellite operators sit at their workstations to manage TCs sent to the satellite(s).

Satellite operators may also remotely operate satellites ⑦ by connecting to the ground segment using tools such as desktop sharing like VNC (Virtual Network Computing) [256], or even rely on autonomous or semi-autonomous operations [293].

Satellite operations include two main activities: satellite tracking and TC generation and scheduling. Satellite tracking calculates the satellite's position in orbit and controls the ground station(s)' tracking antenna to establish communication between the ground and space segments. TC generation and scheduling crafts command to be programmed and sent to the satellite so it can perform different functions, e.g., read systems status data, schedule orbital maneuvers, schedule payload operations, and download payload data.

Ground (Segment) Software (GSS). GSS allows satellite operators to carry out the satellite mission's routine operations as described in the previous paragraph. GSS are very diverse. Some satellite missions develop their own GSS while others use open-source [294] or buy proprietary GSS like GSWeb [295]. There are two main types of GSS: Mission Control Software (MCS) and Ground Station Control Software (GSCS).

Mission control software ⑧ manages TCs and scripts to be sent to the satellite and can display TM. For example, ESA's SCOS-2000 is an MCS that provides generic functionality that can be customized for a specific mission to cover the functions required for TM reception and processing and TC verification [296]. Some satellite missions develop their own MCS. For example, the

SUCHAI mission team developed their MCS application to send and receive TC/TM [294].

Ground station control software (GSCS) ⑨ helps satellite operators track and visualize the satellite's orbit and provide detailed information about each satellite's pass. For example, Gpredict is a popular open-source GSCS that performs real-time satellite tracking and orbit prediction [297]. Gpredict can also interface with and control the radio system and antenna rotor during the satellite pass.

4.2.4 Satellite Architecture

Satellite architectures are varied and complex, however, here we describe the most common terminology depicted in Fig. 4.1's space segment ②.

When referring to satellite architecture there is a distinction between the *platform* that facilitates the successful operation of the satellite's day to day-to-day activities and the *payload*. The platform provides the possibility to run a payload that fulfills the purpose of these missions. This payload differs based on the goal of the mission and can range from instruments to measure physical properties to communications systems providing wireless connectivity.

Platform: The platform, or satellite bus, is a system composed of custom-designed or off-the-shelf subsystems necessary for critical satellite operations. These include, among others, the Attitude Determination and Control System (ADCS) to maintain the satellite's orientation (i.e. attitude) and position to keep the satellite pointed towards antennas or solar panels illuminated; the Electrical Power Subsystem (EPS) for managing power generation and distribution; the Communication Subsystem (COMM) and the Command and Data Handling (C&DH) subsystems to facilitate communications for receiving TC and sending TM and controlling the satellite operations.

All of these subsystems are then controlled via TCs sent to the satellite. TCs may be, depending on the protocol ecosystem and complexity of the subsystem, interpreted by a central C&DH System or merely forwarded to the recipient subsystem for further processing [298]. This managerial duty is left to Flight Software (FS) running on an embedded system of differing complexity. Some examples include NASA's Core Flight System (cFS) [299] and F' (F Prime) [300], KubOS [301], the German Aerospace Center (DLR)'s OUTPOST [302], and the University of Chile's SUCHAI

FS [303]. Attackers aim to gain the ability to send Telecommands to the Flight Software, as this typically grants control over all spacecraft subsystems.

Payload. The payload is the equipment that the satellite employs to fulfill its mission. Due to satellites' varied missions, payloads are heavily customized [304]. For example, if a satellite's mission is remote sensing, its payload may include an infrared camera [305]. craft's essential functions, acts on TCs, and handles internal data transmitted to and from other systems [298]. The CDHS is comprised of Flight Software (FS) running on an OBC [22].

4.2.5 Small Satellite Protocol Ecosystems

SmallSat missions can often be categorized into the following categories by the adoption of protocols and their corresponding philosophies.

Cubesat Space Protocol. The Cubesat Space Protocol (CSP) family of protocols is a one-stop solution for Small missions [306]. There are not a lot of choices left to the operator/developer and there are two vendors mainly offering components using this protocol and some that offer compatibility solutions [295]. If a mission built on CSP is expanded by a commercial subsystem it will plug in without issue after configuration.

CSP is implemented as an open-source C library called *libCSP* [306]. CSP follows the TCP/IP model, including transport and routing protocols and multiple layer 2 interfaces such as I2C (Inter-Integrated Circuit), CAN (Controller Area Network), and ZeroMQ (ZMQ) for transmission on TCP/IP networks [307].

In CSP the ground segment interfaces and satellite subsystems are part of a CSP network. Sending TC is as simple as sending a CSP packet with an address corresponding to a node that is part of the satellite. The configured static routing tables of the nodes on the network will then cause the packet to be passed to the ground station and transmitted to the satellite over RF where a system on board will route the packet to its destination. In case the satellite is not currently doing a pass, the packet is dropped.

The straightforwardness of CSP makes it particularly interesting for the honeypot use case as the internal structures of any CSP-based mission will look very similar, so distinguishing two missions

is difficult as they will use the protocols in a very similar way.

There are also default services running on standardized ports that are enabled by default when building libCSP [306], for example for network diagnostic (ping) or basic operations (rebooting).

In CSP missions, nodes on the network typically feature a command-line-based interface for configuration and debugging purposes [295]. This may allow someone with access to it to interact with the current node and other nodes on the network.

CCSDS Space Communication Protocols. The CCSDS Space Communication protocols are a vast set of standardized protocols used for different purposes in space communications. A major CCSDS protocol relevant for SmallSat TM and TC is called spacepacket. This protocol is used in combination with the ECSS Packet Utilization Standard (PUS) to define how TCs and TM are encoded and transported. The PUS defines services (and thus sets of TC/TM) for functionality that satellite missions likely require, including large data transfer or event reporting [308]. Moreover, mission designers can tailor the PUS standard by selecting a subset of services and sub-services relevant to their mission needs. Furthermore, it is possible to define custom Services or implement numerous custom functionalities.

Proprietary Protocols. In addition to these widespread ecosystems, vendor-specific sets of protocols may be usable on top of more standard protocols or transport higher-layer packets of known protocols. Some basic small satellites may also have entirely ad hoc protocols that are limited to use with a single mission or a set of satellites by a specific institution. These proprietary protocols are not ideal candidates for constructing honeypots as an ecosystem of one mission that is unique to a small set of missions will look out of place for other missions, preventing generalization.

The choice of ecosystem also influences the choice of GSS as an MCS usually focuses on a single protocol stack. In addition to the space protocol ecosystems above, satellite operators use well-known network protocols to connect with the mission's ground segment services. These protocols include Telnet, SSH, VNC, FTP, and HTTP. Telnet, SSH, and VNC are used to remotely connect to the MOC workstations. HTTP is used for web interfaces that operators use to visualize mission data, and FTP is used to download TM data from the mission data repository. If an

adversary were to compromise any of these services, they could use it as a stepping stone to target the satellite itself.

4.2.6 Space Systems' Tactics, Techniques and Procedures (TTPs)

Tactics, Techniques, and Procedures (TTPs) describe the behavior of a malicious actor in a structured scheme to understand how they might execute an attack [309, 310]. Several frameworks have been introduced to standardize space systems' TTPs. There are two MITRE-style frameworks, the SPARTA matrix [311] and the SPACE-SHIELD matrix [312]. These matrices list and describe space security-specific tactics and techniques such as *initial access* and *ground segment compromise*.

4.3 Threat Model

Following Fig. 4.1, we assume an adversary willing to compromise a satellite can only interact with the space segment simulation by sending TCs from the ground segment first. To gain initial access to the ground segment, an adversary needs to connect via one of exposed network protocols depicted in Sec. 4.2.5, namely, VNC, Telnet, TCP/IP, etc., which correspond with the operational protocols used in real satellite missions.

From there, an adversary may try to launch different commands to take full control and/or compromise the services offered by the satellite's mission. Finally, the modeling of physical radio communication between the space and ground segments, which is commonly used in practice, is considered out of scope and left for future work.

4.4 HoneySat High-level Design

In this section, we explain the objectives we aim to achieve (Sec. 4.4.1) and the design principles we follow to meet such objectives (Sec. 4.4.2).

4.4.1 HoneySat's Design Objectives

Our design aims to achieve the following objectives:

DO-1 **Capability to Capture Rich Interaction Data.** As explained in Sec. 4.1, the number one objective of any honeypot is to capture interaction data from which we can derive knowledge

on adversaries' TTPs. As such, our first objective for HoneySat is to have capability to capture rich interaction data.

DO-2 **Provide Deception.** As we discussed in Sec. 4.1, honeypots' nature must remain *covert* to entice adversaries into interacting with it. As such, our second objective is for HoneySat's nature to remain hidden from adversaries.

DO-3 **Provide Extensibility and Customizability.** A framework's main purpose is to provide generic functionality that can be customized to meet the user's needs, in HoneySat's case, we must be able to support multiple SmallSats. For example, a particular SmallSat may use CSP or CCSDS ecosystems. Additionally, each SmallSat mission has a particular orbit that must be customizable. As such, our third objective is for HoneySat to provide extensibility and customizability.

4.4.2 HoneySat's Design Principles

To meet our objectives, we selected the following principles:

DP-1 **High-Interaction Simulation.** As we described in Sec. 4.2.1, high-interaction honeypots give adversaries the same or almost identical interaction opportunities as a real satellite. Thus, they can log many of the TTPs discussed in 4.2.6. For these reasons, we selected high-interaction simulation as our first design principle. This design principle is based on design objective DO-1.

DP-2 **Realistic Simulation.** As we discussed in Sec. 4.2.2, satellites keep track of their orbit location, the position of the sun, among others. These details must be simulated by our honeypot; otherwise, they may alert adversaries that they are interacting with a fake system. This design principle is based on design objective DO-2.

DP-3 **Modularity.** As we explained in Sec. 4.2, satellites are complex and diverse systems. There is no one-size-fits-all solution. To tackle this problem, we designed HoneySat to be modular. Modularity allows us to insert, remove, or change any part of our honeypot framework

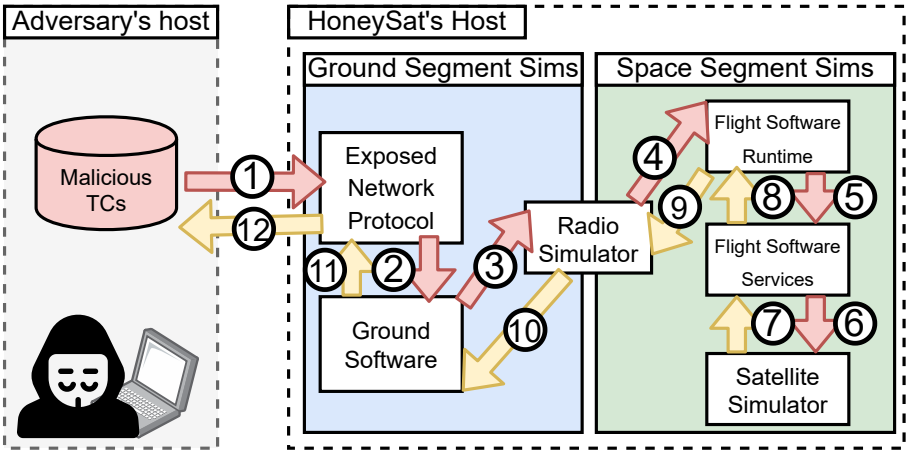

Figure 4.2
HoneySat's Theory of Operation.

without implementing a new honeypot from scratch. This design principle is based on our objective DO-3.

4.4.3 Theory of Operation

In this section, we provide a brief overview of how the theory of operation works by following the numbers in Fig. 4.2.

An adversary gets initial access via one of the *Exposed Network Protocols* ①. On interaction with the *Simulated Ground Segment*, an attacker is presented with access to the *Ground Software* ②. The configuration of this ground software is defined by the *Ground Configuration*, which allows the ground software to look like one of many different missions from its protocol ecosystem to the attacker. The attacker can then interact with the *Ground Software*, while their actions are reported to the *Log Database*. They might want to try to gain more privileges on the ground segment or try to send TC to the space segment. Instead of deploying a ground station with an RF transmitter and associated hardware, we employ the *Radio Simulator* ③ to handle all simulated radio frequency communications.

When an attacker uses the ground segment to send a valid TC or raw packets, the *Radio Simulator* checks whether the satellite configured in the *Satellite Personality* is currently passing

over the designated ground station location. If so, the *Radio Simulator* forwards the TC to the *Flight Software Runtime* ④. When the *Flight Software Runtime* receives a command, it will run the corresponding *Flight Software Service* to compute a response ⑤. In case it requires either changes to an on-board system's state or a sensor value to execute the TC, it will invoke the *Satellite Simulator* for this ⑥. The *Satellite Personality* contains the configuration used by the *Satellite Simulator*. Once a TC is processed, the resulting TM is routed back through the *Radio Simulator* to either the attacker or the ground software used by the attacker ⑦-⑫, and is also recorded in our *Log Database*.

As we stated in Sec.4.1, my collaborator, Ulysse Planta at the CISPA Helmholtz Center for Information Security, contributed the detailed technical design, and the implementation of the system and these are available in our publication titled "HoneySat: A Network-based Satellite Honeypot Framework [276]".

4.5 Evaluation

This section lists three experimental questions designed to test HoneySat's alignment with our design objectives. Next, we present four sets of experiments that provide empirical evidence confirming that our design objectives have been achieved within HoneySat. We describe each experiment's environment, methodologies, and results.

4.5.1 Experimental Questions

These questions aim to determine whether or not HoneySat's meets the design objectives outlined in Sec. 4.4.1.

Q-1 **Can HoneySat offer extensive interaction opportunities to adversaries?**

Since capturing data on varied techniques is the purpose of any honeypot, we explore the capabilities of HoneySat, as described in Sec. 4.4. This question is related to design objective DO-1 and is addressed in Sec. 4.5.2 and 4.5.4.

Q-2 **Can HoneySat simulate a SmallSat mission well enough to deceive adversaries?**

After enticing an adversary HoneySat must keep its true nature hidden, we need to simulate

the satellite's communication and physics characteristics described in Sec. 4.2.4. This question relates to design objective DO-2 and is answered in Sec. 4.5.3.

Q-3 **Can HoneySat simulate different SmallSat missions?**

HoneySat's customization is important because it would allow users of our framework to implement their own honeypots. This question relates to design objective DO-3, and we answer it in Sec. 4.5.5.

To answer the above research questions we conducted four experiments. First, in Sec. 4.5.2 we craft multiple attacks in a controlled environment to quantify the level of interaction that HoneySat provides. Second, in Sec. 4.5.3 we conduct a survey with experienced satellite operators to evaluate HoneySat's realism. Third, in Sec. 4.5.4 we deploy HoneySat and expose it to the Internet to test its deception capabilities. Fourth and final, in Sec. 4.5.5 we add an additional protocol ecosystem to HoneySat to test its extensibility potential.

4.5.2 TTP Interaction Experiment

This experiment seeks to answer Q-1 by quantifying the different interactions that HoneySat provides. To achieve this, we leveraged the SPACE-SHIELD matrix (Version 2.0) [312] discussed in Sec. 4.2.6. SPACE-SHIELD provides a collection of adversary tactics and techniques for space systems. In this experiment, we determined the number of tactics and techniques that HoneySat supports. SPACE-SHIELD consists of 14 tactics and 62 techniques. However, not all of them are applicable to a virtual, network-based honeypot such as HoneySat. Applicability is defined by the limitations of a virtual honeypot which does not implement any real hardware. For example, the technique *Compromise Hardware Supply Chain* involves "replacing a hardware component in the supply chain with a custom or counterfeit part" which is out of the scope of a virtual honeypot. Taking this into consideration, 14 tactics, and 33 techniques are applicable to HoneySat.

Experiment Description. Our experimental environment included two hosts. One host running HoneySat and the adversary host. The HoneySat host ran Ubuntu 23.10 and was configured with the SUCHAI-2 satellite and ground personalities. The adversary host ran a Telnet client. Both

hosts were connected to the same network.

Experiment Methodology. We designed one interaction or exploit for each of the applicable 33 techniques for our honeypot. The interactions involved ground segment simulations such as the web interface. The exploits were simple, using one TC, or complex with multiple TCs involved. The interactions were designed as examples of how the techniques in the SPACE-SHIELD matrix can be implemented in our honeypot.

Experiment Results. We crafted 16 exploits and 17 interactions following the above methodology. For example, the technique *T1007 - System Service Discovery*'s description reads "Adversaries may try to gather information about registered local system services. Adversaries may obtain information about services using tools and OS utility commands. Adversaries may use the information from System Service Discovery during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions."

Based on this description, we designed the following exploit to capture information about the satellite's running processes:

TC-1: `1: obc_system ps -aux > ps.log`

TC-2: `1: tm_send_file 10 ps.log`

Similarly, technique *T2014 - Backdoor Installation* reads "An attacker can interfere with the hardware or the software, integrating or modifying the existing software, hardware configuration, or the transponder configuration to permit himself future access to the resource". Therefore, we designed the following exploit to upload a malicious script to the satellite software:

TC-1: `tm_send_file backdoor.sh 1`

TC-2: `1: obc_system ./recv_files/backdoor.sh`

Due to space limitations, we do not describe all the exploits and interactions here. However, the complete exploit and interaction list is available in Table 4.5 in Appendix 4.7.1. The overall results for this experiment are depicted in Table 4.2. The key findings of our experiment are shown below, providing strong evidence for answering question Q-1 in the affirmative and design objective DO-1

Tactics	SPACE-SHIELD Techniques (Applicable to Virtual Honeypots)	HoneySat Supported Techniques
Reconnaissance	2	2
Resource Development	2	2
Initial Access	2	2
Execution	2	2
Persistence	2	2
Privilege Escalation	2	1
Defense Evasion	4	4
Credential Access	3	3
Discovery	2	2
Lateral Movement	4	1
Collection	2	2
Command & Control	2	2
Exfiltration	2	1
Impact	7	7
Total	38	33

Table 4.2
Tactics and techniques supported by HoneySat compared to the ones included in the SPACE-SHIELD matrix that are applicable to virtual honeypots.

as achieved.

Key findings Q-1

- HoneySat supports 86.8% of the SPACE-SHIELD matrix techniques possible in a virtual satellite honeypot.
- HoneySat supports 100% of the SPACE-SHIELD matrix tactics.

4.5.3 SmallSat Operators Survey

Evaluating the realism of a satellite honeypot is challenging for two main reasons. First, as we discussed in Sec. 4.1, satellites, including SmallSats are very diverse. Second, there is no established metric or tool, such as Nmap's OS detection [313], to quantify the level of realism of our honeypot.

To evaluate the realism and deception capabilities of HoneySat, we surveyed *experienced, real-world SmallSat operators*. Because operators interact with real-world SmallSat missions on a daily basis they are *experts* and thus are the best population to rigorously evaluate HoneySat.

4.5.3.1 Survey Structure

We divided our survey into five sections. The first section collected essential background information about the participants, while the second part focused on their professional experience. In this second section, participants were asked whether they worked in industry, government, or academia and to self-report their cybersecurity skills as well as satellite-related skills (e.g., satellite hardware engineering). The third section of the survey probed participants' satellite operation experience, including how many missions they had operated and which tools they used.

The largest section of the survey included 48 questions about *satellite honeypot operation tasks*. Participants watched eight 1–2 minute recordings of HoneySat 's VNC view, each showcasing the ground and satellite personality of a real CSP-based CubeSat mission (based on the SUCHAI-2 CubeSat [314]). Every recording highlighted a different feature of HoneySat by replicating a real-world satellite mission operation, as discussed in Section 4.2. After each recording, participants answered questions designed to assess how realistic they found that particular aspect of the honeypot. Table 4.3 lists all of the satellite operations demonstrated in these recordings. Finally, once participants had seen multiple elements of our honeypot, the survey concluded with an overall evaluation section. Both the honeypot operation tasks and the overall evaluation used 5-point Likert scale [315] questions to measure participants' positive or negative reactions.

Satellite Operation	**Evaluated Component**	**No. Qts.**
Telecommand Scheduling	Mission Control SW	6
Pass Prediction	Ground Station Control SW	6
Antenna Positioning	Ground Station Control SW	6
EPS Telemetry	Satellite Simulator	6
Temperature Telemetry	Satellite Simulator	6
ADCS Telemetry	Satellite Simulator	6
RGB Camera Telecommand	Satellite Simulator	6
Ping Test	Radio Simulator	6

Table 4.3
Summary of questions in Section 4.5.3, *satellite honeypot operation tasks* of the survey (48 total).

4.5.3.2 Participants

We conducted the survey via Qualtrics and distributed the survey directly to *active SmallSat mission operators* from previously identified missions. In total, we received responses from 14 satellite operators who have between 1 and 5 years of experience operating satellites and have operated between 2 and 7 unique missions. In terms of demographics, 21.4% (3/14) of participants were female, 71.4% (10/14) male and 7.1% (1/14) preferred not to answer. 57.1% (8/14) belonged to the 18-24 age group, 35.7% (5/14) to 25-34 and 7.1% (1/14) to 35-44. 28.5% (4/14) of participants' highest level of education was high school, 35.7% (5/14) bachelor's degree, 14.2% (2/14) master's degree and 21.4% (3/14) Ph.D. In regards to geographic location, 78.5% (11/14) of participants were located in Europe, 14.2% (2/14) in North America and 7.1% (1/14) in South America.

Recruiting participants was challenging due to the rarity and specialized nature of the required expertise. Over a span of four months, we reached out to national and international institutions, as well as private corporations involved in operational satellite missions to identify suitable participants.

4.5.3.3 Methodology and Key Results

In the survey, we aimed to evaluate three key aspects of our honeypot. First, whether the ground segment simulations are realistic; second, whether the space segment simulations are realistic; and third, whether HoneySat, as a whole, provides a convincing and realistic system.

Before describing the results it is important to emphasize that the participants were *informed* that the recordings showed a simulation of a satellite mission and not of a real mission.

Ground Segment Realism. To understand if HoneySat's ground segment is realistic we showed participants recordings of different satellite operations (discussed in Sec. 4.2.3) performed with HoneySat, to showcase the ground segment components listed in Table 4.3.

For example, one of the recordings shows the *pass prediction* operation which involves calculating when and where a satellite will be visible or within range of a specific ground station. This operation is performed using the ground station control software. After the participants watched a recording of this operation, we asked them to rate their perceived level of realism. 85.7% of par-

ticipants agreed or strongly agreed that the *pass prediction* operation done in HoneySat resembles that of a real mission.

Another relevant operation is the *telecommand scheduling* operation which involves the planning and organization of the commands to be sent to the satellite during a pass. 42.8% of participants strongly agreed that the *telecommand scheduling* operation performed in HoneySat resembles that of a real mission mentioning the use of a terminal-based mission control software. Interestingly, 35.7% of participants neither agreed nor disagreed, citing that they do not use a command line tool but instead a GUI. These results indicate that some of the surveyed operators use a command line-based mission control software while others use a GUI-based confirming the diversity of means of operation in Sec. 4.2.3.

In summary, according to our survey, most participants perceive the ground segment simulated by HoneySat as comparable to that of a real satellite mission. While a few participants mentioned that they expected to see a GUI-based mission control software instead of a command line-based one, this is something that we expected, as the means of satellite operations vary greatly among missions. Nevertheless, thanks to HoneySat's modularity HoneySat can be extended to use a GUI-based mission control software such as YAMCS [316].

Space Segment Realism. In a similar manner, to understand if HoneySat's space segment is realistic, we showed participants recordings of different satellite operations that make use of HoneySat's space segment simulations to showcase multiple components of the space segment listed in Table 4.3. For example, one of the recordings shows the *EPS Telemetry* operation which involves the collection of data from the Electrical Power Subsystem discussed in Sec. 4.2.4. This operation is performed using the mission control software to send telecommands and download the latest EPS telemetry. After the participants watched a recording of this operation performed by HoneySat, we asked them to rate their perceived level of realism of the telemetry output shown during the operation. 57.1% of participants agreed or strongly agreed that the telemetry shown in the EPS Telemetry operation resembles that of a real mission mentioning realistic EPS values such as voltage, and temperature. 14.2% neither agreed nor disagreed mentioning that the EPS temeletry

was presented in a different format. Finally, 28.5% disagreed or strongly disagreed mentioning that the telemetry included less values than the mission that they had operated. These results indicate that the EPS telemetry generate by the Satellite Simulator is considered realistic by the majority of participants.

We also asked participants to asses the *ping test* operation. This operation is a diagnostic procedure used to verify the communication link between a ground station and the satellite. It involves sending CSP packets between two CSP nodes, namely the ground segment and the spacecraft. After the participants watched a recording of this operation performed on HoneySat, we asked them to rate their perceived level of realism. 64.2% of participants agreed or strongly agreed that the ping test operation resembles that of a real mission citing the realistic response times. 21.4% of participants neither agreed nor disagreed stating that in their experience the ping test also downloads additional telemetry. Finally, 14.2% disagreed or strongly disagreed noting that in their mission the ping command has a different name. The results of the *ping test* operation indicate that HoneySat's radio simulator (which controls the communication between the ground and space segments) is considered to realistically simulate the communication of a real satellite mission by the majority of participants. A significant key finding of our survey is shown below.

Overall, the results obtained provide convincing evidence for answering question Q-2 in the affirmative and design objective DO-2 as achieved.

Key finding Q-2

71.4% of surveyed satellite operators agreed or strongly agreed with the following statement: *"If I did not know this was a honeypot simulation of a CubeSat satellite mission, I would have believed it to be an actual CubeSat satellite mission."*

4.5.4 Internet Interaction Experiment

This experiment explores HoneySat's capabilities to entice external actors in the wild by deploying our honeypot over the Internet. To accomplish this, we leveraged the Shodan search engine. Shodan is a search engine for Internet-connected devices that scans the whole Internet and indexes

Honeypot	Source IP	Cmds Received	Engaged Time
Honeypot 1	Egypt	4	2 hr
Honeypot 2	Sweden (Tor)	9	5 min
Honeypot 1	France (Tor)	6	4 min
Honeypot 3	USA	8	3 min

Table 4.4
Exposed Telnet interactions received.

exposed servers and their TCP ports. Shodan then reads the banner information for each port. For example, it gathers the web headers and Telnet login banners [317]. Additionally, Shodan tags servers as honeypots if the servers have too many open ports [269].

Experiment Description. We deployed and configured five HoneySat instances over the Internet and exposed TCP port 24 for the Telnet server and 80 for the web interface so that anybody can interact with it and allow Shodan to index our server and its banners. We deployed four HoneySat instances in total.

Experiment Results. Our honeypot servers were successfully indexed by Shodan. Both Telnet and web banners were captured, and none of them were tagged as honeypots.

Our honeypots successfully enticed external actors and captured 4 distinct sessions (show in Table 4.4) via the exposed Telnet protocol. The exposed Telnet protocol was implemented so that a human had to type *"activate"* in the terminal before sending commands. This feature was implemented to filter out Internet Telnet bots. As such, the commands that we describe below were sent by a human and not a crawler or Internet bot.

We now describe the commands received in the first interaction session shown in Table 4.4.

1. *help:* Shows the ground software available commands.
2. *ls:* The Linux ls command is not a valid command in the honeypot ground software.
3. *fp_show:* Prints the list of commands in the current flight plan (The flight plan refers to a pre-programmed sequence of tasks, and commands that guide the satellite's operation).
4. *com_debug:* Prints the current CSP network configuration including the current CSP node, interfaces and routing table.

The commands *fp_show* and *com_debug* are *flight software-specific commands* which suggests that the adversary was successfully deceived by our honeypot and purposefully sent flight software commands instead of haphazardly sending unrelated commands. Another factor that indicates that the adversary was deceived is that they engaged with our honeypot over a period of two hours indicating that they did not immediately identified it as a honeypot.

Although the data captured is limited to 16 commands, it is important to emphasize that satellite systems are niche targets, and it is exceptionally rare to see unsolicited, satellite-specific interactions on the public internet, so the fact that HoneySat attracted any targeted commands at all suggests it was successful in appearing convincing to at least some adversaries. Second, the nature of the interactions matters as much as the quantity: the commands we received were structured, mission-relevant, and plausibly malicious, indicating intentional engagement rather than random scanning or generic bot activity. Third, our goal was to demonstrate feasibility and validate that a virtual satellite honeypot can elicit realistic and domain-specific attacker behavior.

In summary, these results provide strong evidence for answering question Q-2 in the affirmative and design objective DO-2 as achieved.

Key finding Q-2

Human adversaries interacted with three of our HoneySat-powered honeypots during four distinct Telnet sessions resulting in *16 satellite mission-specific commands* captured.

4.5.5 Case Study: Generic CCSDS Mission Honeypot

In this case study we are interested in testing the extensibility capabilities of HoneySat to support additional ecosystems (discussed in Sec. 4.2.5) by adding a *second ecosystem* to HoneySat, namely the CCSDS ecosystem.

Experiment Description. We selected the CCSDS ecosystem because it is a standard protocol suite used by other SmallSats [22]. To accomplish this, we leveraged an open-source CCSDS ecosystem-based flight software framework, RACCOON OS [318], and YAMCS, an open-source Mission Control software framework with built-in support for PUS [316].

Methodology. Building upon our HoneySat framework implementation, we enhanced the system by integrating various components of the RACCOON OS and the YAMCS framework.

Results. We successfully extended HoneySat to support the CCSDS ecosystem. Regarding the ground segment we implemented the *exposed network protocol* using YAMCS' built-in web interface (Fig. 4.4), for the *mission control software* we used YAMCS's built-in Mission Control Software [316], for the *radio simulator* we used RACCOON's communication application and for the *ground configuration* and *logging repository* we again used YAMCS built-in features.

Focusing on the space segment, we implemented the satellite *flight software runtime* using the RACCOON framework, and for the *Flight Software Services*, we configured the RACCOON framework to connect it to the YAMCS's MCS on the ground segment. The *satellite personality* and *logging repository* were based on the existing HoneySat implementations.

The majority of the effort involved in extending HoneySat to support the CCSDS ecosystem was dedicated to understanding the ecosystem itself, including components such as PUS. Additionally, we had to analyze the RACCOON flight software code and the YAMCS framework documentation. The only extra implementation that was required was the Flight Software Services which we modified using Rust. Other than that we reused several elements of HoneySat like the satellite personality and the Satellite Simulator.

In summary, these results provide strong evidence for answering question Q-3 in the affirmative and design objective DO-3 as achieved.

Key findings Q-3

Out of the box HoneySat supports CSP and CCSDS, the two most widely used standard space ecosystems, and was evaluated by simulating *3 real-world SmallSats*.

4.6 Discussion and Future Work

Limitations. Currently, the functionality of some subsystems within HoneySat's Satellite Simulator are constrained by the quality of the data provided. For example, the resolution of Earth's images generated by our camera payload depends on the resolution of its source data

(USGS [319]). Consequently, creating a honeypot for a satellite with a high-resolution camera payload would not be feasible without addressing the underlying issues of data source quality. Additionally, HoneySat is currently designed as a virtual honeypot and does not support radio link attacks such as jamming.

Broader Applications of HoneySat. While originally designed as a honeypot, our framework offers the flexibility to support a range of applications beyond its initial purpose. One promising use case is the development of digital twins for satellite systems, enabling the simulation of real-world satellite subsystems and communication scenarios. Furthermore, HoneySat can be integrated into cyber range environments to enhance training programs and cybersecurity simulation exercises. Lastly, HoneySat serves as an educational tool, providing researchers, industry professionals, and security enthusiasts with an opportunity to explore and learn about satellite architecture and operations.

4.7 Chapter Conclusion

Although we have yet to witness a Stuxnet-like cyberattack on space systems, security researchers and professionals need to develop effective countermeasures to secure satellites. In this dissertation chapter, we introduced HoneySat, the first satellite honeypot that provides a much-needed alternative source of empirical data on attackers' TTPs. We created a satellite honeypot that realistically simulates an entire satellite system, including its sensors and subsystems.

We performed several experiments that provide strong evidence that the framework can obtain rich interaction data, can be extended and customized, and simulate a satellite to keep its honeypot nature hidden from potential adversaries. Finally, we hope that security researchers and professionals take advantage of HoneySat's open-source implementation and use it as a foundation not only for satellite honeypot deployments but also for simulation, education, and training applications.

Appendix B

4.7.1 Interaction Sequences and Exploits

Table 4.5 includes all the interaction sequences and exploits we performed during the experiments in Sec. 4.5.2.

Table 4.5
Tactics, Techniques and Procedures' Interaction Experimental Exploits.

Tactic	Technique	ID	Subsystem	Exploit
Reconnaissance	Active Scanning (RF/Optical)	T2001	Threat Model Limitation	N.A.
Reconnaissance	Gather Victim Mission Information	T2002	Web Interface	Use the Web Interface to gather mission documentation.
Reconnaissance	Gather Victim Org Information	T1591	Web interface	Use the Web Interface to gather mission documentation.
Reconnaissance	In orbit proximity intelligence	T2029	Threat Model Limitation	N.A.
Reconnaissance	Passive Interception (RF/Optical)	T2004	Threat Model Limitation	N.A.
Reconnaissance	Phishing for Information	T1598	Tangential	N.A.
Resource Development	Acquire or Build Infrastructure	T1583	Telnet interface	Acquire ground segment using the Telnet service.
Resource Development	Compromise Account	T2038	Tangential	N.A.
Resource Development	Compromise Infrastructure	T1584	Threat Model Limitation	N.A.
Resource Development	Develop/Obtain Capabilities	T2007	Ground software. Flight software.	Exploit OS, libraries or software vulnarabilities. Deploy custom CSP application to forge TC/TM.
Initial Access	Direct Attack to Space Communication Links	T2008	Ground software. Flight software.	Use the ground software to send/receive TC/TM. Deploy custom CSP application to forge TC/TM.
Initial Access	Ground Segment Compromise	T2030	Telnet interface. Ground software.	Use the telnet interface to access the ground software.
Initial Access	Supply Chain Compromise	T1195	Tangential	N.A.
Initial Access	Trusted Relationship	T2039	Threat Model Limitation	N.A.
Initial Access	Valid Credentials	T2009	Tangential	N.A.
Execution	Modification of On Board Control Procedures modification	T2010	Ground software. Flight software.	Upload a script to the satellite software: tm_send_file code.py 1 1: obc_system python recv_files/code.py
Execution	Native API	T1106	Ground software. Flight software.	Execute shell commands or delete system files: 1: obc_system <command> 1: obc_rm -r $HOME
Execution	Payload Exploitation to Execute Commands	T2012	Tangential	N.A.
Persistence	Backdoor Installation	T2014	Ground software. Flight software.	Upload a script to the satellite software: tm_send_file backdoor.sh 1 1: obc_system ./recv_files/backdoor.sh
Persistence	Key Management Infrastructure Manipulation	T2013	Tangential	N.A.
Persistence	Pre-OS Boot	T1542	Ground software. Flight software.	Use obc_system, obc_rm, obc_mkdir commands. Upload/modify an OS configuration file. Start/stop/schedule execution of services/daemons.
Persistence	Valid Credentials	T2009	Tangential	N.A.
Privilege Escalation	Become Avionics Bus Master	T2031	Implementation limitation	N.A.
Privilege Escalation	Escape to Host	T1611	Docker. Virtual machine.	Escape the container/VM with previously crafted exploits.
Defense Evasion	Impair Defenses	T1562	Ground software. Flight software.	Send commands to change operation mode. 1: drp_set_var_name obc_opmode 0
Defense Evasion	Indicator Removal on Host	T1070	Ground software. Flight software.	Use commands to remove artifacts, logs, etc.: 1: obc_rm <path>
Defense Evasion	Masquerading	T2040	Ground software. Flight software.	Use commands to upload artifacts modify system settings: tm_send_file articaft 1 1: obc_mv artifact /etc/config/artifact
Defense Evasion	Pre-OS Boot	T2041	Ground software. Flight software.	Use obc_system, obc_rm, obc_mkdir commands. Upload/modify an OS configuration file. Start/stop/schedule execution of services/daemons.
Credential Access	Adversary in the Middle	T2042	Ground software. Flight software.	Deploy a CSP application to capture/inject CSP packets.
Credential Access	Brute Force	T2043	Ground software. Flight software.	Brute force valid TC parameters: 1: obc_ebf <KEY>
Credential Access	Communication Link Sniffing	T2044	Ground software.	Scape the ground software or docker and run tcpdump. Deploy a CSP application to capture/inject CSP packets.
Credential Access	Retrieve TT&C master/session keys	T2015	Tangential	N.A.
Discovery	Key Management Policy Discovery	T2032	Tangential	N.A.
Discovery	Spacecraft's Components Discovery	T2034	Ground software. Flight software.	Send TC to redirect satellite logs to ground segment: 1: log_set 5 2 10
Discovery	System Service Discovery	T1007	Ground software. Flight software.	Capture running processes information 1: obc_system ps -aux >ps.log 1: tm_send_file 10 ps.log
Discovery	Trust Relationships Discovery	T2033	Tangential	N.A.
Lateral Movement	Compromise a Payload after compromising the main satellite platform	T2045	Implementation Limitation	N.A.
Lateral Movement	Compromise of satellite hypervisors	T2017	Docker. Virtual machine.	Escape the container/VM with previously crafted exploits.
Lateral Movement	Compromise the satellite platform starting from a compromised payload.	T2046	Implementation Limitation	N.A.
Lateral Movement	Lateral Movement via common Avionics Bus.	T2016	Implementation Limitation	N.A.
Collection	Adversary in the Middle	T1557	Ground software. Flight software.	Deploy a CSP application to capture/inject CSP packets.
Collection	Data from link eavesdropping	T2018	Ground software. Flight software.	Scape the ground software or docker and run tcpdump. Deploy a CSP application to capture/inject CSP packets.
Command and Control	Protocol Tunnelling	T2047	Ground software. Flight software.	Deploy an malicious application that sends data over CSP
Command and Control	Telecommand a Spacescraft	T2019	Ground software. Flight software.	Use the ground software to send TC: 1: com_ping 1
Command and Control	TT&C over ISL	T2048	Threat Model Limitation	N.A.
Exfiltration	Exfiltration Over Payload Channel	T2021	Implementation Limitation	N.A.
Exfiltration	Exfiltration Over TM Channel	T2022	Ground software. Flight software.	The atacker deploys a custom CSP node or a backdoor
Exfiltration	Optical link modification	T2037	Threat Model Limitation	N.A.
Exfiltration	RF modification	T2036	Threat Model Limitation	N.A.
Exfiltration	Side-channel exfiltration	T2035	Threat Model Limitation	N.A.
Impact	Data Manipulation	T2054	Ground software. Flight software.	Send TC to modify/reset TM database: 1: drp_set_var_name drp_ack_ads 10000000 1: drp_reset_payload 1 1010 1: drp_reset_status 1010
Impact	Ground Segment Jamming	T2050	Threat Model Limitation	N.A.
Impact	Loss of spacecraft telecommanding	T2055	Ground software. Flight software.	Send TC to change communication paramters. Modify network configuration in the ground station.
Impact	Permanent loss to telecommand satellite	T2027	Ground software. Flight software.	Send TC to destroy filesystem: 1: obc_system rm -rf –no-preserve-root /
Impact	Resource damage	T2028	Threat Model Limitation	N.A.
Impact	Resource Hijacking	T1496	Ground software. Flight software.	Upload a script to the satellite software: tm_send_file code.py 1 1: obc_system python recv_files/code.py
Impact	Saturation of Inter Satellite Links	T2052	Threat Model Limitation	N.A.
Impact	Saturation/Exhaustion of Spacecraft Resources	T2053	Ground software. Flight software.	Send TC to create a reset loop: 1: fp_set_cmd_dt 10 2147483647 10 obc_reset
Impact	Service Stop	T1489	Ground software. Flight software.	Send TC to lauch a fork bomb or a reset loop 1: obc_system :(){ :—:& };: 1: fp_set_cmd_dt 10 2147483647 10 obc_reset
Impact	Spacecraft Jamming	T2049	Threat Model Limitation	N.A.
Impact	Temporary loss to telecommand satellite	T2026	Ground software. Flight software.	Send TC to make the system unresponsive 1: obc_system sleep 3600
Impact	Transmitted Data Manipulation	T2024	Threat Model Limitation	N.A.

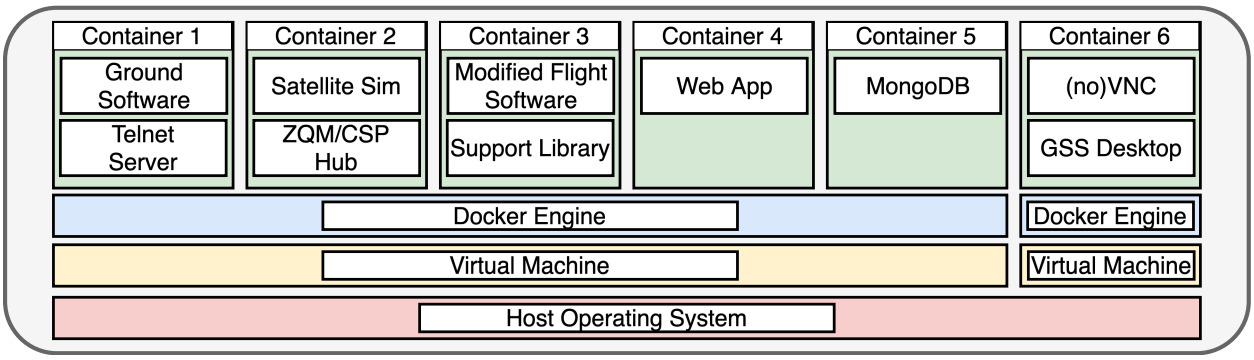

Figure 4.3
Architecture of dockerized Generic CSP Honeypot.

4.7.2 Deployment and Security Hardening Implementation

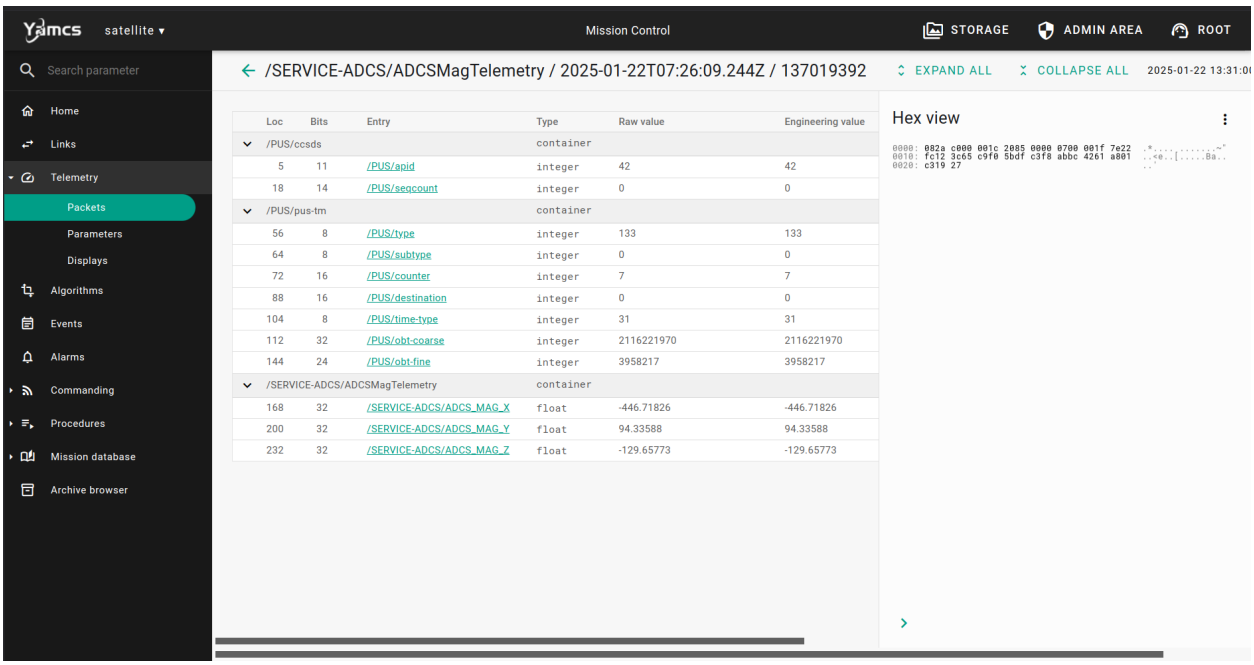

Figure 4.4
Screenshot of the YAMCS web interface, displaying a TM packet, returned from the Generic CCSDS Mission Honeypot.

As we mentioned in Sec. 4.2.1, high-interaction honeypots such as HoneySat present a high risk of adversary takeover. To mitigate this risk, we implemented HoneySat with two sandboxing layers.

Virtual Machine. VMs provide the highest isolation level among sandboxing techniques. We implemented HoneySat in VM environments to leverage VMs' robust security.

Containerization. After we completed HoneySat's development, we used Docker Compose [320] to containerize each of our framework's applications. Specifically, we created four different containers depicted in Fig. 4.3. Containerizing HoneySat provides two benefits. First, it creates another sandboxing layer that prevents adversaries from using our honeypot to access the underlying system [279]. Second, it proves a convenient and flexible way to deploy HoneySat.

Firewall. In addition the previous two measures we also enabled and configured a standard firewall to allow only the required TCP ports that our honeypot uses. This improves our deployment security by blocking any connections to ports that might be vulnerable to documented attacks.

CHAPTER V: CONNECTED AND AUTONOMOUS VEHICLES SANDBOX

Abstract

Connected and Autonomous Vehicles (CAVs) are a core component of future transportation, adopting paradigms such as Vehicle-to-Cloud (V2C) to provide data analytics and customized mobility services, i.e., entertainment for passengers. This innovative approach has also provided adversaries with a new attack surface to compromise both the security and safety of CAVs, potentially putting human lives at risk. However, despite its daunting importance, this novel threat model remains underexplored in the literature.

To address this research gap, this dissertation chapter presents the first systematic threat analysis of the V2C threat model. To accomplish this, we introduce two novel systems. First, *Grand Hack Auto*, the first interactive CAV sandbox environment that simulates the required V2C infrastructure for threat analysis, and *HYDRA*, the first synthetic malware for CAVs, designed to implement multiple CAV-specific Tactics, Techniques and Procedures (TTPs). TPP characterize adversaries' behavior in a structured manner and help us model, understand, and plan countermeasures against such threats.

Our results show that exploiting V2C connectivity has serious consequences as they may allow for adversaries to orchestrate multi-vehicle collision incidents over the Internet. Additionally, *Grand Hack Auto* and *HYDRA* demonstrate a high level of robustness by realism by successfully integrating them with two enterprise level cloud environments, HiveMQ and Mosquitto. Additionally, our systematic threat analysis shows that our frameworks supports 15 techniques that rely on the V2C threat model. These results demonstrate that *Grand Hack Auto* and *HYDRA* are highly-effective tools that may allow for better protections against threats on V2C-enabled CAVs.

5.1 Introduction

Connected and Autonomous Vehicles (CAVs) have been forecast as the future of transportation. As an example, the U.S. Department of Transportation (USDOT) considers CAVs as a central technology in its development plans [321]. Also, several companies including Tesla, Waymo, and

Honda have developed their own models of CAVs in the past few years [322]. However, CAVs have been shown to be vulnerable to cyberattacks [323]. For example, Tesla CAVs are vulnerable to relay attacks which allows malicious actors to unlock and steal cars [11].

Compounding this problem is the fact that CAVs are increasingly connected to the cloud due to the Vehicle-to-Cloud (V2C) connectivity paradigm [324], which allows for CAVs to provide users with new services such as entertainment.

In such regard, the MQTT network protocol has become the most popular solution among CAV Original Equipment Manufacturers (OEMs) to deliver V2C services [325, 326] due to its low network overhead and its ability to maintain sessions even in deadzones without cellular coverage. As an example, BMW's DriveNow service uses MQTT as a part of its backbone cyberinfrastructure [327]. Moreover, MQTT has become the *de facto* standard to provide CAVs *mobility services* [325] including, but not limited to, messaging services, data analytics and command processing. To deliver these services, OEMs rely on IoT cloud providers who provision the necessary infrastructure (storage, processing, networking), for example, HiveMQ's MQTT Broker [328] and Amazon Web Services' AWS IoT Core product [324]. Although MQTT provides built-in security features such as authentication and encryption [329] these are not always configured or even enabled [330]. In 2018, 99.71% of the MQTT Internet deployments, as listed by the Shodan search engine, did not use encryption [331].

While extensive research has been conducted on securing in-vehicle networks such as CAN bus, the security of V2C communication channels remains poorly understood, thus potentially providing adversaries with a new attack surface to attack CAVs. In this dissertation chapter we explore the emerging *V2C threat model* by analyzing the security implications of MQTT-based communication between CAVs and cloud services, highlighting the tactics, techniques and procedures (TTPs) that adversaries may exploit and their real-world consequences for CAVs. For example, a V2C-enabled technique is *Internet Communication* [30] where adversaries use a compromised Internet connection for command and control and to exfiltrate the vehicle's data.

To accomplish this, we introduce *Grand Hack Auto*, the first CAV interactive customizable

framework designed to analyze attacks and malware that take advantage of the *V2C threat model*. *Grand Hack Auto* leverages sandboxing, a well-known method, modeling an isolated, safe environment to study malicious software, which has been shown to be an essential tool for Cyber Threat Intelligence (CTI) gathering and is instrumental in preventing the spread of malicious software across a production system, in this case, a CAV [25].

Grand Hack Auto simulates the physical processes and properties of CAVs (e.g., speed and acceleration), the MQTT architecture that connects vehicles to the cloud (e.g., clients and brokers), and the V2C network infrastructure required to test adversarial behavior (e.g., network configuration and logging).

In addition, we also introduce *HYDRA*, the first CAV synthetic malware designed to work as a "Swiss army knife" to implement multiple MQTT adversarial tools such as reconnaissance and Command and Control (C2) [332].

HYDRA also implements CAV-specific attacks such as the *Coordinated Multi-Vehicle Collision* attack [333], which abuses the V2C threat model to orchestrate CAV crashes. *HYDRA* also allow us to gather data to understand the TTPs available in V2C, thus providing a valuable tool that allows us to understand the impact that an attack might have on V2C-enabled CAVs and how countermeasures might be further developed as future work.

In summary, we make the following contributions:

- We introduce *Grand Hack Auto*, a novel threat analysis framework for cloud-based malware and attacks that target CAVs (Sec. 5.4).
- We present *HYDRA*, the first V2C-focused synthetic malware, designed to exploit the *V2C threat model* using the MQTT protocol (Sec. 5.5).
- We leverage both *Grand Hack Auto* and *HYDRA* to explore the *V2C threat model* for CAVs and systematically threat-analyze the possible TTPs that it enables by systemically testing the techniques documented in the Automotive Threat Matrix (ATM) [30] (Sec. 5.6.3).

- Finally, we provide evidence of the robustness and realism of both *Grand Hack Auto* and *HYDRA* by evaluating them on *real-world enterprise IoT cloud infrastructure.* Our evaluation results confirm that our framework is robust and supports two cloud providers (Sec. 5.6.4).

In an effort to support open and reproducible science, *Grand Hack Auto* and *HYDRA* are open sourced and available online[1,2]. As *HYDRA* is designed as an adversarial tool, we took ethical considerations into account to make *HYDRA* open source which we explain in detail in Sec. 5.8.

5.2 Background

We start by providing relevant background topics on CAVs that will be referenced in later sections: sandboxing (Sec. 5.2.1), V2C infrastructure (Sec. 5.2.2), the MQTT protocol (Sec. 5.2.3), tactics, techniques and procedures (TTPs) (Sec. 5.2.4), and CAV threat models (Sec. 5.2.5).

5.2.1 Sandboxing

Sandboxing is a well-established security approach that isolates code execution within a controlled and restricted environment to prevent unintended or malicious interactions with the host system [25]. It has been widely used in the literature for analyzing potentially harmful software without risking real-world assets or infrastructure. In the context of malware research, sandboxes provide a layer of protection, enabling repeatable and observable experiments on malicious behaviors while ensuring system integrity [279].

Sandboxing is particularly relevant to Cyber-Physical Systems (CPS) such as CAVs, where malicious code can impact both digital systems and physical operations and it has been used in the context of Industrial Control Systems (ICS) [334] and space systems [279]. Without an isolated testing environment, even synthetic malware could lead to safety hazards, or network propagation during experimentation.

Finally, sandbox environments provide CTI valuable data that organizations, such as OEMs, can use to develop and improve their systems' security countermeasures and overall security stand-

[1] https://github.com/efrenlopezm/cav-sandbox-mqtt
[2] https://github.com/efrenlopezm/hydra-mqtt

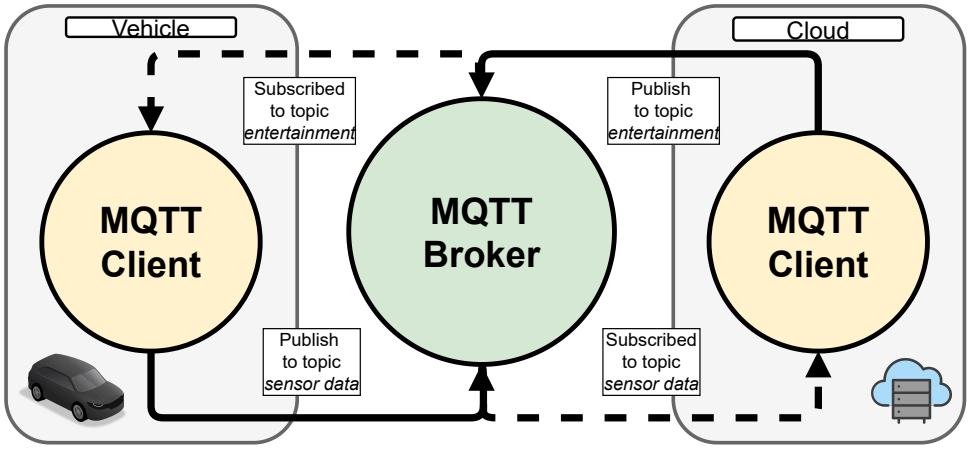

Figure 5.1
Basic MQTT network configuration for CAVs. In this example, a vehicle's MQTT client publishes some sensor data via a topic to the broker and then a cloud MQTT client subscribes to this topic to get the sensor data. In the same way the cloud MQTT client publishes some entertainment data to the broker and the vehicle's MQTT client subscribes to the entertainment topic to get the corresponding data.

ing [25]. These data may include malware behavior, network communication patterns generated by the malware, and vulnerabilities the malware tries to exploit.

5.2.2 Vehicle-to-Cloud (V2C) Infrastructure

Vehicle-to-Cloud or V2C refers to the ability of a vehicle to connect and exchange data with the cloud [335]. V2C infrastructure enables real-time data exchange, such as telemetry, sensor data, and control commands between vehicles and centralized cloud systems, and ultimately allows for a range of applications, including remote diagnostics and over-the-air (OTA) updates. V2C depends on cloud platforms that provide on-demand storage, computational resources, and data processing. These platforms enable CAVs to offload computationally intensive tasks (e.g., machine learning inference) and access shared data from other vehicles and infrastructure. Major cloud providers offer products specifically tailored for V2C, including Amazon AWS IoT Core [324], HiveMQ [328], Azure IoT hub [336], and Google Cloud for Automotive [337]. Most of these cloud platforms communicate with CAVs via the MQTT network protocol discussed in Sec. 5.2.3.

Table 5.1
Comparison of Threat Models in Connected and Autonomous Vehicles (CAVs)

Aspect	Intra-Vehicular	Vehicle-to-Everything (V2X)	Vehicle-to-Cloud (V2C)
Communication Scope	Internal vehicle networks (CAN, LIN, FlexRay)	Nearby vehicles, infrastructure, pedestrians (DSRC, C-V2X)	Remote cloud services (MQTT, REST, WebSockets)
Primary Interface	Wired, internal	Wireless, local (short-range RF)	Internet-based (TCP/IP)
Adversary Proximity	Physical access (OBD-II, ECU)	Local proximity (radio range)	Remote (Internet-accessible)
Attacker Capabilities	Physical presence, access to internal bus, reverse engineering ECUs	RF hardware, proximity spoofing tools, signal jammers	Internet access, compromised credentials, cloud API or broker exploitation
Example Attacks	CAN injection, ECU spoofing, diagnostics abuse	Fake hazard alerts, GPS spoofing, Sybil, jamming	Command injection, fleet-wide broker compromise
Attack Scale	One vehicle per attacker	Multiple vehicles in vicinity	Individual or fleet-wide impact
Latency Constraints	Real-time (ms-level)	Low-latency, safety-critical	Moderate-latency tolerated
Security Focus	Message integrity, ECU authentication	Secure message signing, spectrum resilience	Broker hardening, topic access control, data trust

5.2.3 MQTT Protocol

MQTT or Message Queuing Telemetry Transport is a lightweight publish/subscribe messaging protocol for the Internet of Things (IoT) [338] that is ideal for connecting remote devices with a small code footprint and network bandwidth. MQTT is used in multiple industries, including automotive, and telecommunications. As featured in Fig. 5.1, the MQTT protocol architecture includes three main elements:

- **MQTT Broker.** Brokers are intermediary servers that route messages between clients. Brokers keep track of subscriptions and publishing messages to subscribed clients and are deployed in the cloud or on-premises and can be scaled to support millions of devices and messages per second.
- **MQTT Client.** Clients publish or subscribe to messages to the *broker*. A client can publish a

message to a topic, which is a logical channel to which the message is sent. Subscriber clients receive messages that are published to a topic. Clients can also act as both a publisher and a subscriber.

- **MQTT Topic.** An MQTT topic is a filter the broker uses to deliver messages. The broker filters all connected clients according to their subscriptions and forwards the message to these subscribers. Clients identify the messages to which they want to receive by subscribing to a particular topic.

Finally, MQTT includes two optional security mechanisms: authentication and encryption [329]. Authentication is achieved using a username and password that each client must send to the broker. However, these credentials are sent as plain text. In order to encrypt all communication including authentication credentials, MQTT uses TLS. However, TLS encryption is not always feasible for devices with limited resources [339].

Despite these security features being built-in, they are not always configured correctly. For example, as of February 2025, using the Shodan search engine reveals half a million IoT devices running MQTT's unencrypted TCP port 1883[3].

5.2.4 Connected and Autonomous Vehicles' Tactics, Techniques and Procedures (TTPs)

Tactics, Techniques, and Procedures (TTPs) describe the behavior of a malicious actor in a structured way to understand how they can execute an attack [309, 310]. TTPs are commonly presented as matrices. There are many such matrices according to the type of system they describe. For example, there are matrices for enterprise systems [32], industrial control systems [283], and space systems [340]. In this work, we leverage the Automotive Threat Matrix (ATM) which includes automotive-specific adversary TTPs based on real-world observations and automotive exploit research [30]. The ATM matrix lists and describes 13 vehicle security-specific tactics and techniques such as the *Affect Vehicle Function* tactic, which refers to adversaries trying to affect vehicle functions and systems, i.e., movement control, audio, and displays. Each tactic may include

[3] `https://www.shodan.io/search?query=mqtt+port%3A1883&page=3`

several different techniques, for example, the *Internet Communication* technique occurs when an adversary can use a compromised Electronic Control Unit (ECU) (embedded system in a vehicle that processes sensor data sends commands to actuators.) internet connection for command and control and data exfiltration [30].

5.2.5 Established CAV Threat Models

Security research in CAVs has primarily focused on two well-known threat surfaces: the intra-vehicular network and external communication with nearby entities also known as vehicle-to-everything (V2X), which have guided security research for many years. Table 5.1 provides a high-level comparison of these threat models which we now discuss.

- **Intra-vehicular Threat Model.** The intra-vehicular threat model is best represented by the Controller Area Network (CAN) bus, which as the backbone of intra-vehicular communication, enabling ECUs to exchange real-time messages for controlling physical functions such as braking, acceleration, and steering. The canonical threat model assumes adversaries gains physical or logical access to the CAN bus via an OBD-II port, a compromised ECU, or a remote attack chain [341]. The intra-vehicular threat model can be further divided between the Conventional Remote Attacker Model (CRAM) and the Enhanced Remote Attacker Model (ERAM) [323].

- **Vehicle-to-Everything (V2X) Threat Model.** V2X encompasses communication between the vehicle and nearby infrastructure, and other vehicles, typically over Dedicated Short-range Communications (DSRC) or Cellular-V2X protocols [342]. The V2X threat model allows adversaries to inject or intercept messages in the wireless domain, create phantom vehicles, or exploit timing vulnerabilities.

While these threat models have received extensive attention, emerging communication channels between vehicles and cloud service, e.g., V2C, have not been systematically analyzed, despite growing reliance on cloud-based functionality in modern CAV systems [327].

5.3 Our Focus: Vehicle-to-Cloud Threat Model

We now define the threat model on which this work focuses: the *V2C threat model*. This threat model has become a reality due to the increasingly connectivity between CAVs and cloud infrastructure discussed in Sec. 5.1. This threat model assumes that the target CAV(s) already implements a V2C communication protocol such as MQTT, for example some BMW CAVs already do this [327]. This is relevant as having an existing MQTT implementation helps any MQTT-based malware or adversary to stealthily operate, because the malicious traffic will use the same ports as the legitimate MQTT process. The CAV communicates with a cloud server, e.g., an MQTT broker, either on an encrypted or unencrypted channel and sends data such as GNSS latitude and longitude, speed or acceleration. Finally, the cloud server or broker consumes the data sent from the CAV, for example it can consume the GNSS data to show nearby gas stations. The cloud server can also send data and commands to the CAV. For example, it can send warnings if the CAV is going over the speed limit and can send commands to the CAV such as a throttle command if the CAV's speed reaches dangerous levels.

We further divide this threat model into two different scenarios. These scenarios are illustrated in Fig. 5.2:

- **Scenario 1: Compromised Legitimate Cloud Infrastructure.** In this scenario, the CAV establishes a connection with a legitimate cloud broker, such as a HiveMQ-hosted MQTT service. The CAV is properly configured and uses the intended broker; however, the broker's infrastructure has been compromised. The attacker may have obtained valid credentials, or gained insider access to the cloud environment which often happens [343]. Common vulnerabilities that could enable this scenario include misconfigured access controls (e.g., public-facing brokers without authentication), overly permissive topic access policies, unpatched MQTT broker software with known vulnerabilities, or the exposure of weak or hardcoded API keys and tokens used to authenticate broker access. Additional risks include improper multi-tenancy isolation, where one customer's misconfiguration or breach may expose others' data, and insider threats, where an

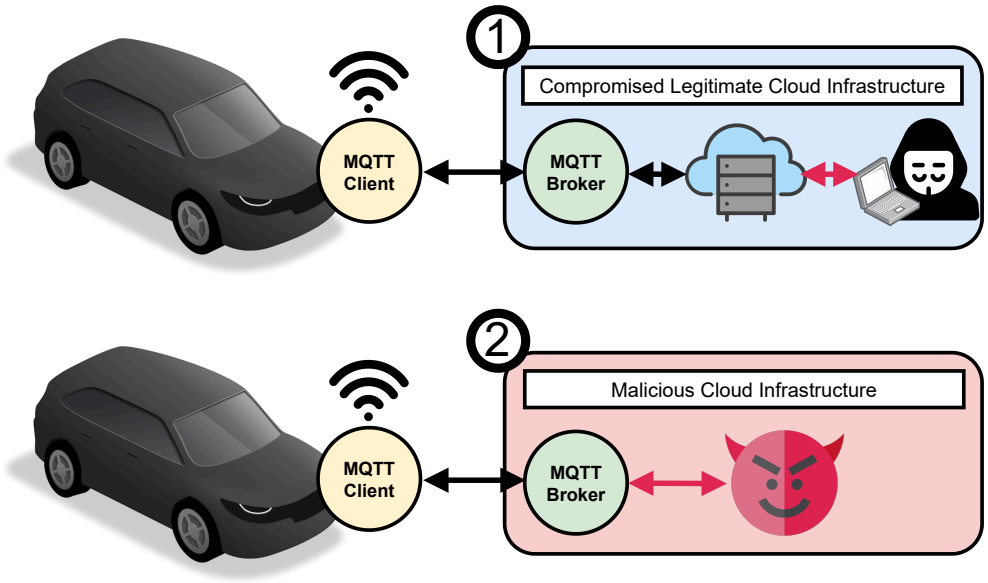

Figure 5.2
The two threat model scenarios considered for *Grand Hack Auto*. ① the target CAV communicates with a legitimate cloud provider that has been compromised. ② the target CAV communicates with a malicious MQTT client running on a C2 infrastructure controlled by an attacker.

employee or contractor with broker admin access.

- **Scenario 2: Malicious Cloud Infrastructure.** In this scenario, the CAV connects to a broker that is fully controlled by the attacker. This may occur due to misconfiguration, DNS spoofing, or exploitation of CAV software vulnerabilities. A specific example would be an attacker that exploits vulnerabilities in the CAV's MQTT client implementation allowing them to inject new broker configurations or redirect communication flows at runtime without changing static configurations. The attacker sets up a rogue MQTT broker that impersonates a trusted service or leverages permissive client settings that allow arbitrary broker connections. This scenario has already been documented in the Industrial Control Systems (ICS) domain [344].

5.4 *Grand Hack Auto*: A Threat Analysis Framework for V2C

As mentioned in Sec. 5.2.1, sandboxes are an effective way of analyzing malware behavior because they provide a realistic, isolated environment that stops the malware from affecting other systems or hosts. However, the level of realism that a particular sandbox provides varies depending

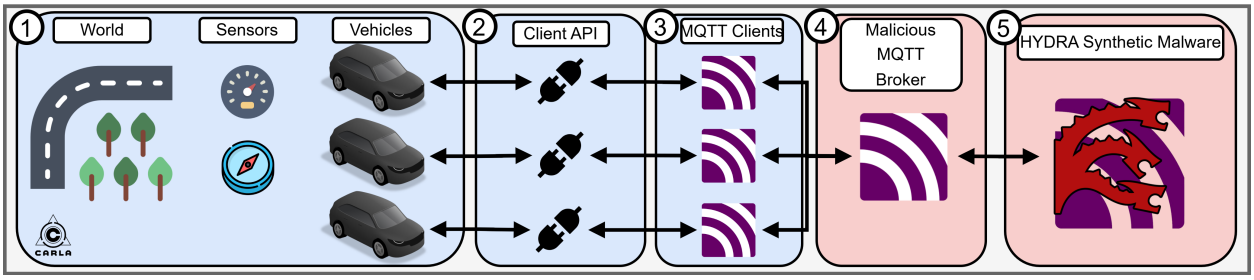

Figure 5.3
Grand Hack Auto framework architecture. The framework leverages the CARLA simulator ① which generates a world where the vehicles move and each vehicle has sensors (e.g. GNSS) attached to them. Each vehicle uses the CARLA API ② to send and receive data to its own MQTT client ③. All the vehicles' MQTT clients connect to a malicious broker which can be local or cloud-based ④ and this broker is connected to the malware ⑤ that exploits the multiple features of our analysis framework.

on the malware itself. Because of this, CPS virtual sandboxes have become popular [334, 345] as they allow us to monitor the malware's behavior and its impact on the physical process.

5.4.1 Design Objectives

With that in mind, *Grand Hack Auto* is designed to achieve the following objectives:

- **(DG-1) Isolation.** *Grand Hack Auto* must be completely isolated from any other network so that any unknown or unforeseen malware behavior does not affect other systems.
- **(DG-2) Modularity.** *Grand Hack Auto* must be modular so that different modules can be attached and detached depending on the requirement. This allows for extensibility to analyze future malware.
- **(DG-3) Network Traffic Capture.** *Grand Hack Auto* must provide a network traffic capture capability so that the malware behavior can be studied using network forensics.
- **(DG-4) Logging.** *Grand Hack Auto* must provide built-in logging capabilities so that the interactions between the sandbox and the malware to be analyzed can be logged into organized and easy to analyze data.

- **(DG-5) Secure Coupling.** *Grand Hack Auto* must allow easy integration and separation of infected or malicious applications.

- **(DG-6) Physical CAV Simulation.** *Grand Hack Auto* must provide a realistic simulation of connected vehicles so that the malware can consume realistic data and the real-world consequences of the malware payload can be observed.

5.4.2 Implementation

Based on the aforementioned design principles, Fig. 5.3 depicts the high-level architecture of *Grand Hack Auto*, which are described as follows:

- **Physical Simulation Module.** The physical simulation replicates the required elements found in the CAV environment, including the CAVs themselves, traffic lights, and roads. The simulation is capable of simulating or "spawning" multiple CAVs so that the interactions between them are analyzed. The simulation provides important data points related to each CAV, for example, GNSS coordinates, acceleration and velocity. We implemented the physical CAV simulation leveraging the CARLA simulator [346] and writing a CARLA client application that spawns one or more vehicles. Each vehicle has a GNSS sensor attached to generate GNSS data and each vehicle has an independent MQTT client that continuously draws sensor data such as acceleration and velocity. The CARLA client application was implemented in Python using CARLA's Python API. This module follows DG-2 and DG-6 design objectives.

- **MQTT Clients Module.** The MQTT clients are the clients that are attached to each of the CAVs spawned in the physical simulation. These clients publish data generated from each simulated CAV to the broker and subscribe to topics to receive data.

  We implemented vehicles' MQTT client in Python using the CARLA API [347] and the Eclipse Paho MQTT package [348]. This was done from scratch as CARLA does not include built-in MQTT support. The client includes a predefined set of commands that are then translated to the local vehicle commands. For example, the remote command "break" is translated to

"throttle=0.0 brake=1.0" for the CARLA API. However, this predefined set of commands can be updated dynamically by communicating with the malicious broker. We implemented this by having the client subscribe to a special topic called "update". This allows the client to download new payloads from the malicious broker. This module follows DG-2 design objective.

- **MQTT Broker Module.** The MQTT broker publishes the cloud data discussed in Sec. 5.2.2 to the CAVs. Additionally, the broker is vulnerable malicious activities as it can be used by a malicious MQTT client or malware to communicate with the CAV as we discussed in Sec. 5.3. We implemented the malicious MQTT broker using Eclipse Mosquitto [349] and created a configuration file to customize the broker to support local or cloud connections. This module follows DG-2 and DG-6 design objectives.

- **Malware Module.** This module hosts the malware that is being analyzed and that will interact with the rest of the sandbox. Because there has not been any documented CAV or V2C malware, in this module we host our synthetic malware *HYDRA*, which we discuss in Sec. 5.5. In order to isolate each part of the sandbox we used two sandboxing layers [279], first a virtual machine (VMWare) and containerization (Docker). To facilitate the network traffic capture we leveraged Docker's networking features to route all sandbox traffic through a virtual network interface. This interface can be used by network forensic tools such as Wireshark for further analysis. Finally, logging was implemented using a Python class that logs the sandbox's relevant data including acceleration, speed, x and y positions, GNSS latitude and longitude and latency. Each data is organized as a CSV file for each of the vehicles in the simulation. This module follows the D-1, DG-2, DG-3, DG-4, DG-5 and DG-6 design objectives.

5.5 *HYDRA*: A Synthetic Malware for V2C

Synthetic malware has been used as an alternative to understand how a specific system might be affected and what countermeasures can be developed to mitigate such attacks. For example, *PLCinject* is an offensive tool that injects Siemens' Programmable Logic Controller (PLC) mal-

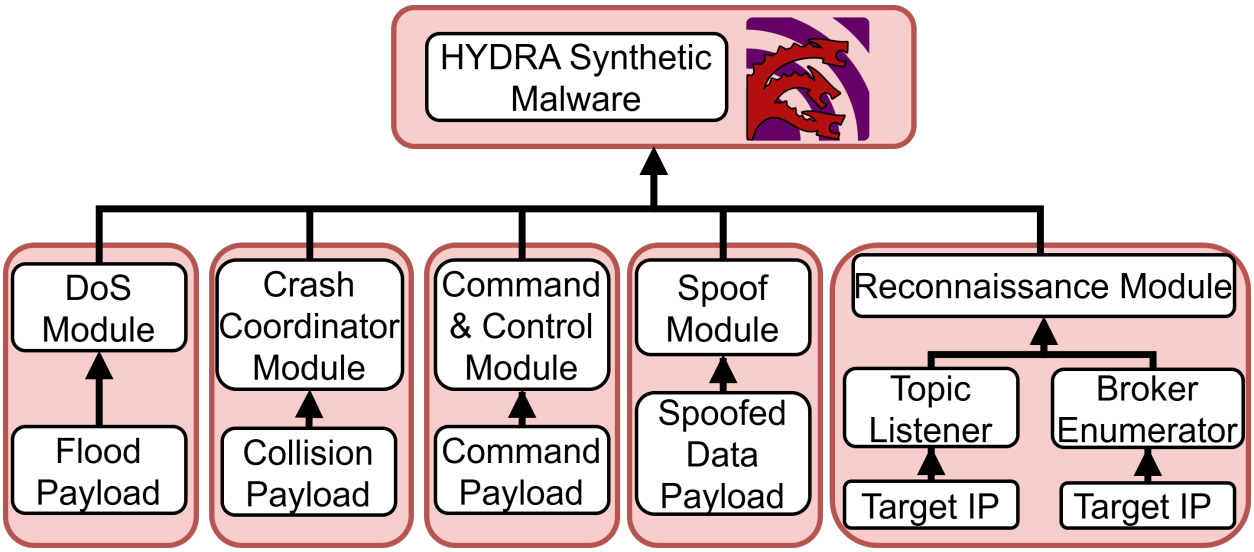

Figure 5.4
HYDRA synthetic malware's architectural design.

ware [350], *LogicLocker* is a synthetic ransomware designed for PLCs [169], and *IronSpider* is a synthetic malware that targets webserver vulnerabilities in PLCs [216].

However, to the best of our knowledge, there is no documented real-world malware that specifically targets CAVs. To address this, we now present the design considerations as well as the architectural implementation of *HYDRA*, the first synthetic malware for V2C-enabled CAVs.

5.5.1 Design Objectives

There are some examples of real-world malware that target similar types of CPS that may give us an idea of how a CAV-specific malware might look like:

IOCONTROL [344] is a malware that targets Supervisory Control and Data Acquisition (SCADA) devices including PLCs to execute commands such as arbitrary code execution, port scan, and more. IOCONTROL achieves these capabilites by communicating with a Command & Control (C2) server over MQTT. [344]. Conversely, PIPEDREAM [351] is a "Swiss army knife" malware that includes tools for reconnaissance, manipulation, and disruption of PLCs. These tools can scan for new devices, brute force passwords, sever connections, and then crash the target device. PIPEDREAM even provides adversaries with an interface for manipulating the targeted

Algorithm 1 Crash Coordinator

Initialize empty map `positions` to store car states
for each telemetry message received **do**
    Extract car ID from topic
    Parse and store (x, y, yaw) in `positions[car_id]`
end for
if positions received from all cars **then**
    Compute crash point:
        $x_{\text{crash}} = \frac{1}{N} \sum_{i=1}^{N} x_i$
        $y_{\text{crash}} = \frac{1}{N} \sum_{i=1}^{N} y_i$
    for each car **do**
        Compute desired yaw toward crash point
        Calculate yaw error and steering
        Set throttle based on steering angle
        Choose command (e.g., `forward`, `left`, `right`)
        Publish control command to MQTT
    end for
end if

devices [351].

Following this, the design objectives of *HYDRA* are:

- **(DH-1) Modularity.** *HYDRA* must be modular so that different adversarial tools, e.g., reconnaissance tools, can be integrated to implement multiple techniques used in the V2C threat model and that can be tested on *Grand Hack Auto*.
- **(DH-2) Support V2C Protocol.** *HYDRA* must support a V2C network protocol, specifically MQTT, as it is used in real-world CAVs and cloud servers.
- **(DH-3) CAV-specific Attacks.** *HYDRA* must make use of the V2C threat model to implement malicious tools that are specifically designed for CAVs. For example a coordinated multi-vehicle collision attack.

5.5.2 Implementation

We implemented *HYDRA* using Python and the Paho MQTT library [348]. The implementation features modular classes so that new tools can be added or removed from the toolkit. Architecturally,

as depicted in Fig. 5.4, *HYDRA* has two main components based on the aforementioned design objectives:

Malicious Broker. The malicious broker acts as a Command & Control server and is capable of subscribing and publishing to the malicious client. When subscribing, the broker automatically saves the exfiltrated data published by the client and when acting as a publisher, the broker sends commands to the client. The logic of these commands are implemented in the toolkit. This module follows DH-1 and DH-2 design objectives.

Toolkit. *HYDRA*'s toolkit is designed as a modular command line application that includes multiple tools that implement different techniques which are based on the ATM matrix discussed in Sec. 5.2.4. The current design includes the following five modules:

- *Denial of Service (DoS) Module*: This module is designed to overwhelm the CAV's MQTT client with a high volume of requests in a short time period of time to disrupt the legitimate MQTT mobility services operation. This module implements the TTP *Denial of Service on Vehicle Function* technique, and was implemented by rapidly opening a large number of connections, each with a unique MQTT client ID, thus leading the broker to allocate a session state for each client, leading to the Denial of Service [352]. This module follows DH-1 and DH-2 design objectives.

- *Crash Coordinator Module*: This module orchestrates coordinated multi-vehicle collisions within *Grand Hack Auto*, and is designed to emulate a malicious actor who gains control over multiple CAVs and manipulates their trajectories to converge at a given crash point. This is designed to implement the TTP *Affect Vehicle Function* tactic. To implement this module, it first passively listens to telemetry data from multiple vehicles over MQTT. The module continuously receives position and orientation information (e.g., x, y, yaw) published by each vehicle specifically, from topics such as */vehicle1/telemetry/yaw* and computes a crash point based on the average location of all participating vehicles. Using this target point, the system calculates individualized steering and throttle commands that guide each vehicle toward the crash point. These control commands

are then published to vehicle-specific MQTT control topics, for example */vehicle1/command*. Algorithm 1 sequentially describes these steps. This module follows DH-1, DH-2 and DH-3 design objectives.

- *Command & Control (C2) Module*: This module is designed to both exfiltrate or infiltrate data from/to the CAV MQTT client topics. Additionally, it sends commands to the CAV via the malicious MQTT broker. The C2 module allows *HYDRA* to send commands such as "break" to force a vehicle to change its speed dramatically, potentially causing harm to the vehicle and its passenger(s). This module is designed to implement the TTP *Command and Control* tactic and was implemented by allowing structured control messages to be sent from the cloud to a connected autonomous vehicle (CAV) via MQTT. Commands are formatted as JSON objects and include a timestamp to emulate real-time interactions. These messages are published to relevant MQTT topics, such as those responsible for vehicle control, and may contain instructions like throttle adjustments, for example the */vehicle/control* topic. This module follows DH-1, DH-2 and DH-3 design objectives.

- *Spoof Module*: This module sends spoofed messages disguised as if they came from a legitimate cloud service. The goal is to deceive the recipient (either the vehicle or the cloud) into accepting and acting on the messages as if they were authentic. For example, sending a spoofed GNSS message. This module implements the TTP *Defense Evasion* tactic and was implemented by creating a fake or "spoofed" topic that publishes malicious data, for example the */vehicle/sensors/gnss* topic. The *HYDRA* broker then publishes data in JSON format with fake GNSS coordinates. This module follows DH-1, and DH-2 design objectives.

- *Reconnaissance Module*: This module implements the TTP *Network Service Scanning* technique and is designed to scan the network for available MQTT clients and brokers and enumerate them. Additionally, it discovers the available topics so that *HYDRA* can subscribe to them. The recon module includes two tools, the topic listener and the broker enumerator. The topic listener is designed to discover all the MQTT topics in a particular network. Discovering the available

topics is important because they reveal the type of data being transmitted between a client and a broker. For example, if a topic called "vehicle/gnss" is discovered, subscribing to it might reveal the GNSS data from a particular vehicle. The topic listener was implemented to passively discover active MQTT topics by subscribing to all available topics using the wildcard subscription pattern "#" [353]. After registration, it configures the MQTT client to listen to every message published on the broker. This approach enables *HYDRA* to perform reconnaissance of MQTT topic namespaces without prior knowledge of topic names.

The broker enumerator lists the available brokers in a given network and shows several details of the MQTT implementation running on them, for example, the MQTT version, QoS support and uptime. This information is important so that further attacks or exploits can be crafted based on this initial recon information.This tool was implemented to try to connect to the most common MQTT ports including 1883, 8883, 9001 and the special SYS-Topics which are special meta topics that an MQTT broker can use to publish information about the broker itself and its MQTT client sessions [353]. This module follows DH-1, and DH-2 design objectives.

5.6 Evaluation

We now evaluate both *Grand Hack Auto* and *HYDRA* by first laying out the experimental questions we investigate (Sec. 5.6.1), and then performing three sets of experiments (Sec. 5.6.2, Sec. 5.6.3, Sec. 5.6.4).

5.6.1 Experimental Questions

Q-1 **What is the impact of V2C-based malware on decision-making for CAVs and its physical effects?**

As CAVs are Cyber-Physical Systems (CPS), cyberattacks targeting them can have physical, real-world consequences for the CAV themselves and their environment. As such, we are interested in exploring the possible physical consequences of abusing the *V2C threat model*. This research question is addressed in Sec. 5.6.2.

Q-2 **What are the TTPs that *Grand Hack Auto* and *HYDRA* support in the context of the *V2C threat model*?**

Since *Grand Hack Auto* and *HYDRA* were developed to investigate the V2C threat model, we are interested in systematically analyzing the threats that such a model enables. As such, we provide a structured, realistic view of how adversaries can exploit the V2C threat model using the TTPs and mapping attacker behavior across multiple stages of a V2C attack, such as reconnaissance, initial access, command and control. This research question is addressed in Sec. 5.6.3.

Q-3 **Are *Grand Hack Auto* and *HYDRA* robust enough to operate on a real-world enterprise cloud infrastructure?**

As this work focuses on the V2C threat model, we are interesting in experimenting weather *Grand Hack Auto* and *HYDRA* are robust enough to handle real-world cloud environments, such as HiveMQ, discussed in Sec. 5.2.2. This research question is addressed in Sec. 5.6.4.

5.6.2 Physical Impact Experiments

Motivation. In order to demonstrate the physical consequences of abusing the V2C threat model, we conducted two experiments that affect the physical behavior of one or more CAVs in *Grand Hack Auto*: a *remote acceleration and breaking attack* and a *coordinated multi-vehicle collision attack*. The goal of the remote acceleration and braking attack is to disrupt the operation of a single vehicle by remotely injecting drive commands over the V2C channel. The second attack, coordinated multi-vehicle collision, demonstrates the potential of a more complex adversarial behavior involving multiple CAVs. By manipulating the MQTT telemetry and control data across several vehicles, the attacker can steer them into converging paths, triggering a deliberate crash, showing how V2C channels can be leveraged to orchestrate collisions.

To carry out these attacks, the attacker must have network-level access to the MQTT broker, this could be because of misconfigurations (e.g., unauthenticated brokers), or insider access as discussed in Sec. 5.3. Additionally, the vehicles must be subscribed to control topics or accepting drive commands from cloud brokers, for example, a compromised legitimate broker as shown in

Fig. 5.2. These assumptions are supported by the fact that, as we discussed in Sec. 5.1, the vast majority of MQTT deployments are insecure either because they are unencrypted, or improperly configured.

We implemented these two attacks as they abuse the V2C threat model to perform attacks that create dangerous situations for the vehicle's passengers and any pedestrians. These underscore the potential consequences of exploiting the V2C threat model and the necessity for experimental environments to test and analyze these types of attacks.

Methodology. We deployed both *Grand Hack Auto* and *HYDRA* inside a VM with Ubuntu 20.04.6 LTS, 16 GB RAM, and 4 vCPUs. Using our *Grand Hack Auto* configuration file, we configured each vehicle MQTT client to connect to the local network broker on port 1883. After establishing connectivity between *Grand Hack Auto* and *HYDRA*, we launched both the remote acceleration and breaking and a coordinated multi-vehicle collision attacks.

In the *remote acceleration and braking attack* scenario, we sent two unauthorized messages using *HYDRA* to our sandbox, one to force the vehicle to accelerate and another one to brake unexpectedly.

In the *coordinated multi-Vehicle collision attack*. we leveraged *HYDRA*'s Crash Coordinator module discussed in Sec. 5.5. Before launching the attack, we used the topic listener tool discussed in Sec. 5.5.2 to identify the available vehicles and their topics. After this, we configured the crash coordinator module with two vehicles' topics and ran it. The Crash Coordinator ran as a background service within *HYDRA* and subscribed to the previously defined telemetry topics to gather the required data, vehicle's location (x, y) and orientation (yaw). Upon receiving this data, the Crash Coordinator calculated the centralized crash point and finally published the resulting commands to each vehicle's MQTT control topic */vehicle1/control*. These control messages are consumed by *Grand Hack Auto*, which applies them directly to the CARLA vehicle interface, resulting in coordinated motion toward the calculated impact location.

Results.

The results of the *remote acceleration and braking attack* are depicted in Fig. 5.5. The

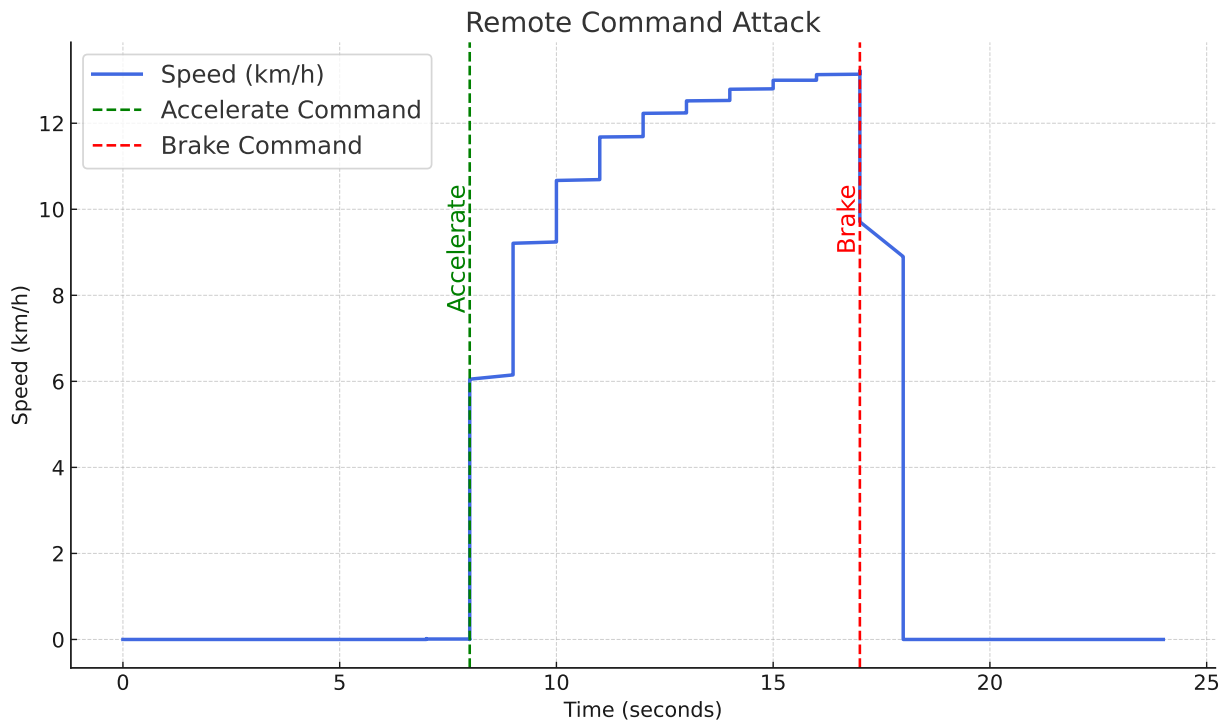

Figure 5.5
Remote command attack example that showcases the effects of the malicious MQTT command on the speed and acceleration of a CAV.

commands were successfully sent from *HYDRA*'s C2 module and caused the vehicle in *Grand Hack Auto* to first accelerate rapidly and then break suddenly. This attack which may lead to collisions, and endanger the vehicle's passenger or pedestrians.

The results of the *coordinated multi-vehicle collision attack.* are depicted in Fig. 5.6. *HYDRA*'s crash coordinator module successfully orchestrated the crash between two vehicles in *Grand Hack Auto* by sending a series of acceleration and steering commands. These eventually led to both vehicles crashing at the crash point shown in Fig. 5.6.

The results from both attacks provide evidence of the impact that V2C-based malware, i.e., *HYDRA*, can have on decision-making processes in CAVs and their corresponding physical effects. Together, these experiments reveal that V2C-based malware poses a direct threat to both individual vehicle autonomy and broader traffic safety, thus providing strong evidence towards answering Q-1.

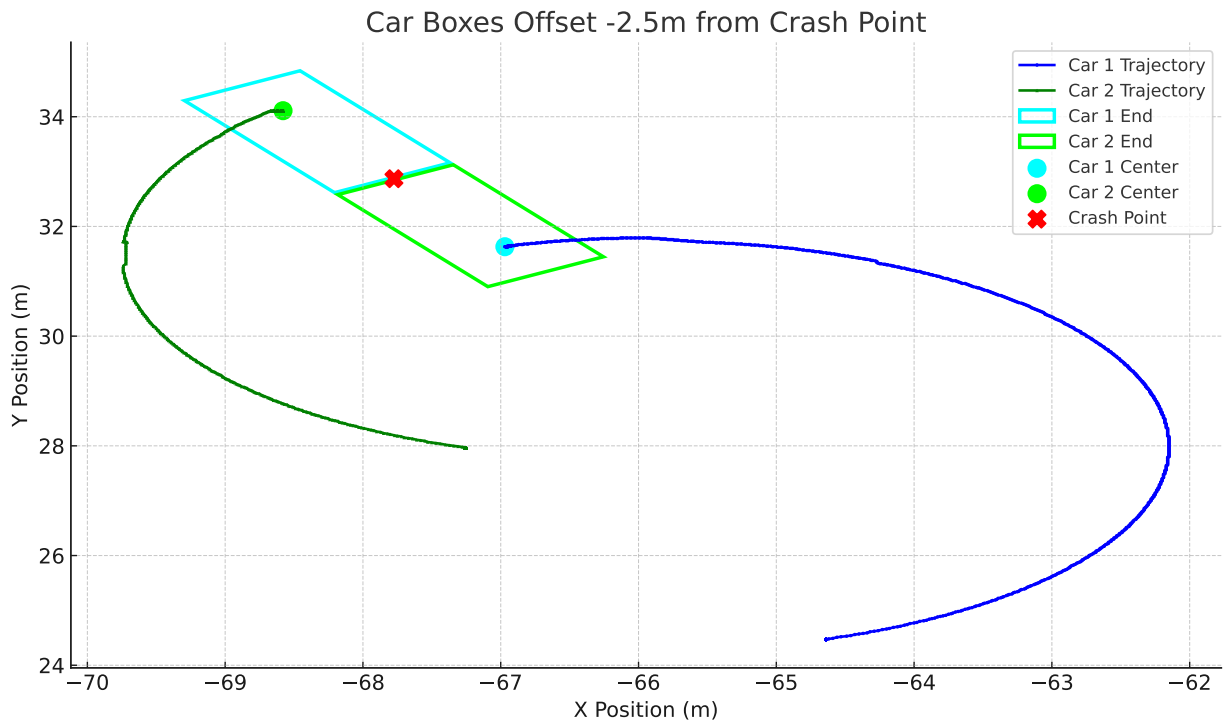

Figure 5.6
2D trajectory plot of two vehicles in a coordinated collision scenario.

5.6.3 Systematic Threat Analysis

Motivation. Understanding the range of TTPs that *Grand Hack Auto* supports is essential for evaluating its effectiveness to analyze V2C threats. The goal of this experiment is to assess and quantify which techniques *Grand Hack Auto* and *HYDRA* can support within the V2C threat model as defined by the ATM matrix [30].

Methodology. Using the same environment as we did in Sec. 5.6.2, we leveraged *HYDRA* to evaluate the TTPs that *Grand Hack Auto* supports. We systematically tested each one of *HYDRA*'s tools (Sec. 5.5) and matched them to their respective technique. We executed each *HYDRA* module within *Grand Hack Auto* using Docker containers to carry out V2C-based attacks. The objective of each experiment was to verify whether the corresponding TTP from the Automotive Threat Matrix could be effectively reproduced within *Grand Hack Auto*. Each experiment was successful if *Grand Hack Auto* demonstrated the ability to deliver the intended physical or logical effect of the

ATM Matrix Tactics	*Grand Hack Auto* Supported Techniques
Manipulate Environment	3
Initial Access	0
Execution	0
Persistence	0
Privilege Escalation	0
Defense Evasion	0
Credential Access	1
Discovery	5
Lateral Movement	0
Collection	1
Command & Control	1
Exfiltration	1
Affect Vehicle Function	2
Reconnaissance	1
Total	15

Table 5.2
Tactics and techniques supported by *Grand Hack Auto* compared to the ones included in the Automotive Threat Matrix (ATM).

attack (e.g., vehicle acceleration). Additionally, each module's behavior was evaluated to ensure fidelity to the semantics of the corresponding ATM technique, thus validating that *Grand Hack Auto* accurately supports the techniques.

Results. We successfully tested and completed attacks using *HYDRA*'s five modules which account for 15 techniques. Table 5.3 lists every technique covered and how it corresponds to our results. To determine this correspondences we used the descriptions of each of the techniques as stated in the ATM matrix. For example, *HYDRA*'s broker enumerator module corresponds to: *Process Discovery*, as it allows "Adversaries to attempt to get information about running processes on a system" as shown in Listing 5.1.

Our findings reveal that a wide range of TTPs are not only feasible in a connected vehicle

```
TC-1: $ hydra.py --ip 127.0.0.1 --broker-enum --verbose
TC-2: Starting MQTT reconnaissance on 127.0.0.1:1883
TC-3: Enumerating broker information...
TC-4: Scanning for open MQTT-related ports...
TC-5: Open MQTT port detected: 1883
TC-6: Open MQTT port detected: 9001
TC-7: Connected with result code 0
TC-8: BrokerEnumerator connected. Attempting to gather more details...
TC-9: Broker Info: $SYS/broker/version -> mosquitto version 2.0.20
```

Listing 5.1
Sample output from *HYDRA*'s broker enumerator feature
which is part of the recon module.

context but can also be executed using lightweight messages through a standard MQTT broker. The experiments validate that *Grand Hack Auto*, coupled with synthetic malware like *HYDRA*, is an effective platform for studying cyber-physical impacts of vehicular threats, thus providing convincing evidence towards answering Q-2. Moreover, this TTP-based evaluation can serve as a foundation for future work in intrusion detection, risk assessment, and the development of countermeasures focused on vehicle-to-cloud attack surfaces.

5.6.4 Cloud Integration Experiment

Motivation. To evaluate the realism and adaptability of *Grand Hack Auto* in real-world, operational environments, we integrated it with two real-world cloud-based MQTT brokers. First, we selected *HiveMQ Cloud* [328], a popular enterprise IoT cloud platform that provides on-demand, MQTT broker deployments. Second, we selected Eclipse Mosquitto [349], an open-source broker implementation of the MQTT protocol that has been used by enterprise applications such as Cisco's Splunk [354] and IBM App Connect Enterprise [355].

Experiment Description. We deployed *Grand Hack Auto* and *HYDRA* on the same environment as Sec. 5.6.2, however, for this experiment we configured each vehicle MQTT client to connect to the HiveMQ and Mosquitto cloud servers on port 1883. After establishing the cloud connectivity between *Grand Hack Auto* and *HYDRA*, we evaluated a crash coordinator scenario involving two vehicles across across three different brokers: HiveMQ, Mosquitto and the local network broker.

Table 5.3
Evaluation of TTPs from the Automotive Threat Matrix using the CAV Sandbox

ATM Tactic	ATM Technique	HYDRA Module Scenario	Sandbox Outcome
Manipulate Environment	Downgrade to Insecure Protocols	Initiated an unencrypted MQTT session on port 1883	Packet capture reveals plain text traffic from vehicles
Manipulate Environment	Jamming or Denial of Service	Used DoS module to overwhelm MQTT client	Vehicles' MQTT client unable to send or receive data
Manipulate Environment	Manipulate Communications	Used Spoof module to manipulate GNSS data	Vehicles' GNSS data is incorrect
Credential Access	Network Sniffing	Initiated a packet capture from malicious broker to sniff topic data	Vehicle's owner data is leaked
Discovery	Location Tracking	Used topic listener tool to gather GNSS data and track location	Vehicle's location is compromised
Discovery	Network Service Scanning	Used broker enumerator to discover the current MQTT version	Vehicle's MQTT service is disclosed
Discovery	Process Discovery	Used broker enumerator to discover the MQTT process	Vehicle's MQTT process is disclosed
Discovery	System Information Discovery	Used broker enumerator to discover the MQTT uptime and version	Vehicle's MQTT uptime is disclosed
Discovery	System Network Connections Discovery	Used broker enumerator to discover the MQTT number of connections	Vehicle's MQTT connections discovered
Collection	Network Sniffing	Initiated a packet capture from malicious broker to sniff topic data	Vehicle's data is compromised
Command and Control	Internet Communication	Use C2 module to send commands to vehicles	Vehicle receives malicious commands to accelerate or break
Exfiltration	Internet Communication	Used topic listener to read and save telemetry data	Vehicle's data is exfiltrated
Affect Vehicle Function	Denial of Service on Vehicle Function	Used DoS module to overwhelm MQTT client	Vehicle cannot receive neccesary data to perform fnctions
Affect Vehicle Function	Abuse Standard Diagnostic Protocol for Affecting Vehicle Function	Used crash coordinator module to affect vehicle's function	Vehicle's function fails and causes a crash
Reconnaissance	Gather Target Information – from Vehicle	Used Reconnaissance module to gather data such as software details	Vehicle's data is compromised

Results. Our initial *Grand Hack Auto* implementation was tested on a local network broker, however, in order to support live HiveMQ and Mosquitto cloud servers we had to modify both *Grand Hack Auto* and *HYDRA*. Specifically, we had to handle multiple MQTT options including Quality of Service (QoS) and topic string encoding. These upgrades allowed for *Grand Hack Auto* to correctly support these enterprise cloud brokers.

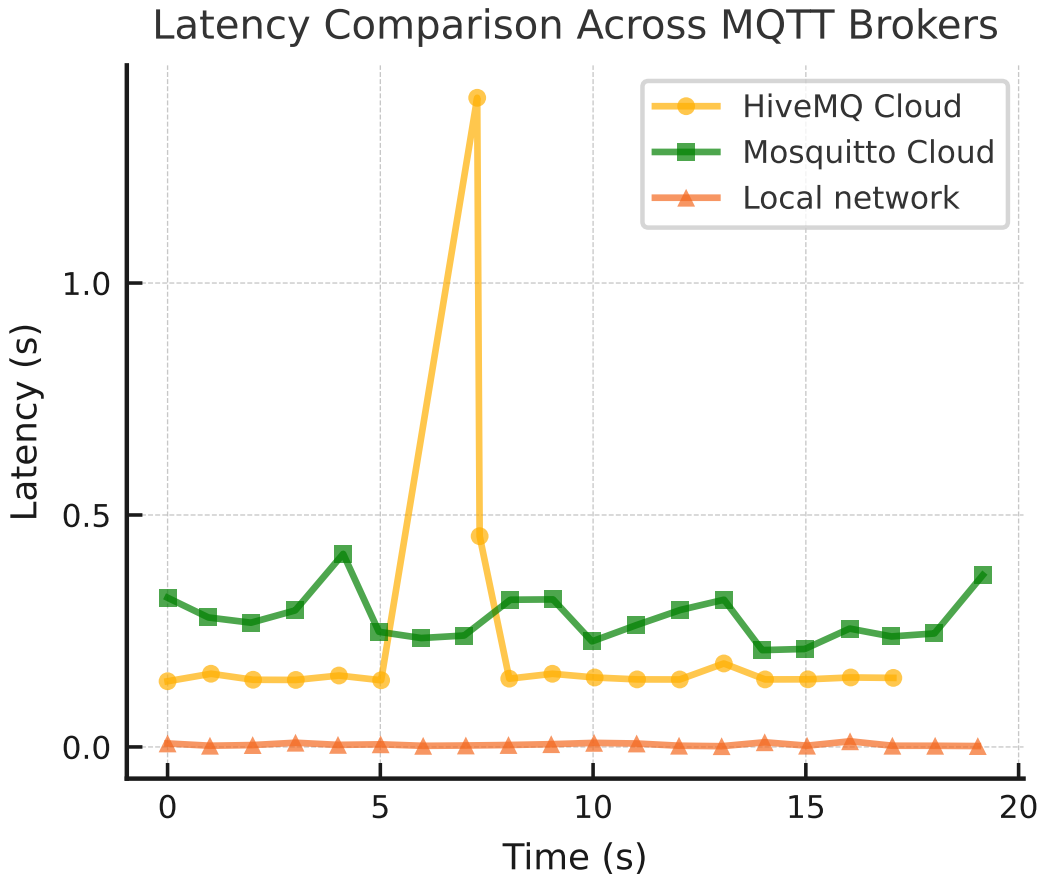

Figure 5.7
Latency comparison between the local network broker and two cloud brokers, HiveMQ and Mosquitto.

Grand Hack Auto successfully maintained stable connections to both cloud brokers, demonstrating interoperability with commercial MQTT cloud services. As expected the latency for both HiveMQ and brokers was higher than the local network's as can be seen in Fig. 5.7. The HiveMQ latency average was 0.236 seconds while the Mosquitto latency was 0.278 seconds, however, the HiveMQ latency spiked considerably at 1.399 seconds while Mosquitto maintained a relatively stable latency with the maximum value recorded at 0.416 seconds.

However, *Grand Hack Auto* was resilient enough to handle these fluctuations without breakdowns in behavior or connectivity.

These results answer Q-3 and indicate that *Grand Hack Auto* and *HYDRA* are indeed robust

Table 5.4
Comparison of Related Work to Grand Hack Auto

System	Type	Security Focus	V2C Focus	Enterprise Cloud Support	Open Source
Wang et al. [356]	Digital Twin	✗	✓	✗	✗
Chunheng et al. [345]	Sandbox	✓	✗	✗	✗
High Mobility [357]	Sandbox	✗	✗	✓	✗
Simutack [358]	Simulation	✓	✗	✗	✓
Grand Hack Auto	Sandbox	✓	✓	✓	✓

enough to operate on real-world enterprise cloud infrastructure and are not limited to local testing environments.

5.7 Discussion

In this section we discuss comparisons with previous work (Sec. 5.7.1), and the limitations of our approach (Sec. 5.7.2).

5.7.1 Related Work

Table 5.4 provides a comparison between our approach and previous works in the literature. More specifically, *Grand Hack Auto* and *HYDRA* can be further compared with previous approaches as follows:

Wang et al. introduced a vehicle digital twin focusing in V2C that synchronizes the data from real-world vehicles to a cloud-based digital twin [356]. Although this work implements V2C, it is used for a different purpose. Specifically, the V2C infrastructure is used to synchronize and save the digital twin whereas *Grand Hack Auto* uses V2C as part of its threat model.

The Cloud-based Sandbox is used to detect the false data injection attacks on CAVs [345]. This sandbox was implemented using Matlab and VISSIM, a traffic simulation environment. However, this approach does not include any network protocol implementation or the physical simulation of the CAVs' sensors.

High Mobility's Sandbox includes a car simulator that exposes vehicle data via an API. Practitioners can draw data from it for different purposes, for example, data analytics via the MQTT

protocol [357]. However, High Mobility's sandbox does not include any security features and it is focused on vehicle sensor data analysis. Additionally, it is available only as a commercial product with no trial version.

Simutack is an open-source attack simulation framework [358] that generates realistic attack scenarios for automotive security testing and it is built leveraging three existing simulation frameworks, CARLA [347], SUMO [359] and OMNeT++ [360]. Simutack provides attack scenarios focused on intra-vehicular and V2X networks.

5.7.2 Limitations

Currently, our framework leverages the CARLA simulator for the physical process simulation. However, CARLA does not simulate the CAV intra-vehicular hardware and software and thus it is not possible to analyze low-level malware using our framework. This limitation could be overcome by extending the CARLA simulator to include intra-vehicular simulations. Nevertheless, as mentioned in Sec. 5.3, *Grand Hack Auto* is focused on the V2C threat model and not on the intra-vehicular networks such as a controller area network (CAN) [323].

Another limitation is that our approach only supports one V2C protocol at the moment: MQTT. We selected MQTT as it is the *de facto* standard for V2C communications, as we discussed in Sec. 5.1. Having said that, as *Grand Hack Auto* is highly modular, additional protocols such as WebSockets [361] could be added in the future.

5.8 Ethical Considerations

In this dissertation chapter we presented *HYDRA*, an offensive tool designed to simulate malware behaviors in V2C systems for the purpose of advancing research on threat modeling for connected autonomous vehicles (CAVs). We are committed to conducting and disseminating research while adhering to ethical standards and responsible disclosure practices. For this, we referenced the USENIX Security '25 Ethics Guidelines [362]. In accordance with such guidelines, we assessed the dual-use nature of our work. While *HYDRA* emulates adversarial behavior, it is explicitly designed for use in controlled simulation environments, i.e., *Grand Hack Auto*, and does not target real-world vehicle firmware.

Additionally, we disclose and point out the adversarial nature of *HYDRA* by including the following warning message every time the malware is used: *"WARNING: This tool is intended for RESEARCH and EDUCATIONAL purposes only. Unauthorized deployment on live or production systems is strictly prohibited. Use responsibly and ensure compliance with local laws and regulations."*

Finally, *HYDRA*'s source code is released in the interest of transparency, reproducibility, and community contribution and we welcome collaboration and feedback from the community to ensure this tool is used ethically and constructively.

Transitioning from the simulated environment of Grand Hack Auto and synthetic malware like HYDRA to real-world CAV security assessment would present significant technical and regulatory challenges beyond the ethical concerns addressed in the paper. Technically, the lack of access to proprietary vehicle firmware, ECU interfaces, and production-grade V2C configurations makes it difficult to replicate attacks on physical vehicles. From a regulatory standpoint, testing on live vehicles may violate legal restrictions on vehicle tampering and pose safety risks. Additionally, targeting enterprise IoT brokers, such as HiveMQ, for testing purposes could breach provider terms of service and impact shared infrastructure. Overcoming these hurdles would require collaboration with OEMs, the use of controlled testbeds, and adherence to formal safety and legal frameworks for real-world experimentation.

5.9 Chapter Conclusion

In this dissertation chapter, we presented *Grand Hack Auto*, a framework for simulating and analyzing V2C threats on CAVs, and *HYDRA*, the first synthetic malware targeting the V2C threat model. Through systematic experimentation using real-world IoT cloud platforms, we demonstrated that *Grand Hack Auto* can realistically emulate CAV behavior and that *HYDRA* can effectively implement a variety of TTPs, including complex attacks such as coordinated multi-vehicle collisions. By open-sourcing our tools, we hope that other researchers can extend both *Grand Hack Auto* and *HYDRA* to continue research and the development of countermeasures that can secure CAVs. For future work we plan to investigate countermeasures for securing dynamic

command update channels in V2C systems, for example, using methods such as authentication, and policy enforcement mechanisms for MQTT topics.

CHAPTER VI: CONCLUSION AND FUTURE WORK

This dissertation addressed critical challenges in CTI for CPS by introducing the concept of *CTI-for-CPS*, which was applied through three new methodologies to improve the collection and processing phases of the CTI life cycle. The research targeted three representative domains of CPS: industrial control systems (ICS), satellite systems, and connected and autonomous vehicles (CAVs), each of which is vital to the infrastructure and security of modern society.

The first contribution was a comprehensive systematization of the literature surrounding Programmable Logic Controllers (PLCs), resulting in a novel ICS threat taxonomy. This taxonomy filled gaps in the existing MITRE ATT&CK for ICS Matrix and provided the foundation for more rigorous threat modeling and countermeasure evaluation in ICS.

The second contribution was the design and evaluation of HoneySat, the first high-interaction satellite honeypot. HoneySat is both realistic and effective at capturing adversarial interactions in space systems. It collected valuable TTP data and received positive evaluations from satellite operators and real-world adversarial interactions over the Internet, supporting the framework's realism and utility.

The third contribution was *Grand Hack Auto*, the first sandbox for V2C threats in CAVs. In combination with the *HYDRA* synthetic malware, the sandbox enabled experimentation with adversarial tactics and evaluated MQTT-based threats on connected vehicles. The results revealed serious attack potential, including coordinated crashes, and validated the sandbox's utility for cybersecurity research.

Together, these contributions advance the state of the art in CTI for CPS by demonstrating novel ways to gather, classify, and analyze threat intelligence in domains where traditional CTI techniques are ineffective. They show that with domain-specific frameworks and tools, it is possible to build a stronger foundation for defending critical CPS infrastructure.

Future work will focus on advancing both the analysis and dissemination phases of the CTI lifecycle for CPS. While this dissertation has primarily addressed the collection and processing phases, further CTI life cycles phases remain unexplored and are an open challenge. In the case

of the analysis phase, one direction is the development of automated analysis tools that take the processed data and find more specific examples of potential vulnerabilities. These tools could use machine learning to detect potential threats, infer attacker objectives, and assess operational impact. In space systems, such tools might analyze threat indicators or anomalies in radio, communications. For industrial systems, they might analyze CTI with control loop anomalies or actuator misbehavior.

Regarding the dissemination phase of CTI for CPS, sharing CTI with relevant stakeholders in a secure, timely, and context-aware manner is of paramount importance. This includes creating tailored CTI formats for different CPS sectors, defining role-based access to sensitive information, and leveraging dissemination protocols like TAXII [363]. In the case of satellites, special considerations include limited bandwidth, intermittent connectivity, and the diverse organizational ecosystem, e.g., commercial, governmental, and academic missions. Future systems should support automated, policy-driven dissemination that ensures the right information reaches the right entity without overwhelming them with irrelevant or overly technical data. Together, improved analysis and dissemination phases of CTI for CPS would help close the loop in the CTI lifecycle, enabling CPS operators to respond more quickly and effectively to the latest threats.

REFERENCES

[1] W. Bolton, *Programmable logic controllers*. Newnes, 2015.

[2] D. Formby and R. Beyah, "Temporal execution behavior for host anomaly detection in programmable logic controllers," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1455–1469, 2019.

[3] P. Ladisa, H. Plate, M. Martinez, and O. Barais, "Sok: Taxonomy of attacks on open-source software supply chains," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2022, pp. 167–184.

[4] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ics) security–rev. 2," *NIST Special Publication*, vol. 800, no. 82, 2015.

[5] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Comput. Surv.*, vol. 51, no. 4, jul 2018. [Online]. Available: https://doi.org/10.1145/3203245

[6] K. Stouffer, M. Pease, C. Tang, T. Zimmerman, V. Pillitteri, and S. Lightman, "Guide to operational technology (ot) security," *National Institute of Standards and Technology: Gaithersburg, MD, USA*, vol. NIST SP 800-82 Rev. 3, Sep. 2023.

[7] E. L. Morales, C. E. Rubio-Medrano, A. Doupé, R. Wang, Y. Shoshitaishvili, T. Bao, and G.-J. Ahn, *HoneyPLC: A Next-Generation Honeypot for Industrial Control Systems*. Cham: Springer International Publishing, 2023, pp. 145–181. [Online]. Available: https://doi.org/10.1007/978-3-031-16613-6_8

[8] "MITRE ATT&CK®: Design and philosophy," https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf, accessed: 12-10-2023.

[9] National Institute of Standards and Technology, "Csrc topic: cyber-physical systems — csrc," Jun. 2020. [Online]. Available: https://csrc.nist.gov/topics/applications/cyber-physical-systems

[10] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, no. 6, p. 29, 2011.

[11] A. Greenberg, "Teslas can still be stolen with a cheap radio hack—despite new keyless tech," May 2024. [Online]. Available: https://www.wired.com/story/tesla-ultra-wideband-radio-relay-attacks/

[12] S. Caltagirone, "Industrial Control Threat Intelligence," 2018.

[13] M. Dempsey, "JP 2-0, Joint Intelligence," Oct 2013.

[14] E. López-Morales, "Securing cyber-physical systems via advanced cyber threat intelligence methods," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 5119–5121. [Online]. Available: https://doi-org.manowar.tamucc.edu/10.1145/3658644.3690865

[15] E. López-Morales, C. Rubio-Medrano, A. Doupé, Y. Shoshitaishvili, R. Wang, T. Bao, and G.-J. Ahn, "Honeyplc: A next-generation honeypot for industrial control systems," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 279–291. [Online]. Available: https://doi.org/10.1145/3372297.3423356

[16] S. Caltagirone, "Industrial control threat intelligence," Dragos, Inc., Whitepaper, 2018. [Online]. Available: https://www.dragos.com/wp-content/uploads/Industrial-Control-Threat-Intelligence-Whitepaper.pdf

[17] C. Sauerwein, D. Fischer, M. Rubsamen, G. Rosenberger, D. Stelzer, and R. Breu, "From threat data to actionable intelligence: an exploratory analysis of the intelligence cycle implementation in cyber threat intelligence sharing platforms," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, 2021, pp. 1–9.

[18] K. Kim, J. S. Kim, S. Jeong, J.-H. Park, and H. K. Kim, "Cybersecurity for autonomous vehicles: Review of attacks and defense," *Computers & security*, vol. 103, p. 102150, 2021.

[19] "Cyber-Physical Systems Workshop," https://web.archive.org/web/20080517071555/http://varma.ece.cmu.edu/cps/, 2006, accessed: 27-09-2023.

[20] The White House, "Presidential policy directive – critical infrastructure security and resilience (ppd-21)," Feb. 2013, accessed: 2025-05-26. [Online]. Available: https://obamawhitehouse.archives.gov/the-press-office/2013/02/12/presidential-policy-directive-critical-infrastructure-security-and-resil

[21] T. ICS-CERT, "Mar-17-352-01 hatman-safety system targeted malware (update b)," https://www.cisa.gov/sites/default/files/documents/MAR-17-352-01%20HatMan%20-%20Safety%20System%20Targeted%20Malware%20%28Update%20B%29.pdf, Feb. 2019, accessed: 02-10-2023.

[22] J. Willbold, M. Schloegel, M. Vögele, M. Gerhardt, T. Holz, and A. Abbasi, "Space odyssey: An experimental software security analysis of satellites," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 1–19.

[23] California Department of Transportation, "Connected and automated vehicles (cav)," 2023, accessed: 2025-05-26. [Online]. Available: https://dot.ca.gov/programs/traffic-operations/cav

[24] SafeBreach Labs, "Operationalizing threat intelligence with breach and attack simulation," 2022, accessed: 2025-05-26. [Online]. Available: https://www.safebreach.com/blog/operationalizing-threat-intelligence-with-breach-and-attack-simulation

[25] B. Lenaerts-Bergmans, "What is cybersecurity sandboxing?" Sep. 2023. [Online]. Available: https://www.crowdstrike.com/en-us/cybersecurity-101/threat-intelligence/cybersecurity-sandboxing/

[26] Microsoft, "What is cyber threat intelligence?" 2023, accessed: 2025-05-26. [Online]. Available: https://www.microsoft.com/en-us/security/business/security-101/what-is-cyber-threat-intelligence

[27] H. Slatman, "Awesome threat intelligence," 2018, accessed: 2025-05-26. [Online]. Available: https://github.com/hslatman/awesome-threat-intelligence

[28] VirusTotal, "Virustotal file upload service," 2025, accessed: 2025-05-26. [Online]. Available: https://www.virustotal.com/gui/home/upload

[29] Meta Platforms, Inc., "Threatexchange documentation," 2024, accessed: 2025-05-26. [Online]. Available: https://developers.facebook.com/docs/threat-exchange

[30] AutoISAC, "Atm," Jan. 2025. [Online]. Available: https://atm.automotiveisac.com/

[31] The Aerospace Corporation, "Sparta: Space attack research and tactic analysis," 2023, accessed: 2025-05-26. [Online]. Available: https://sparta.aerospace.org/

[32] The MITRE Corporation, "Att&ck matrix for enterprise," Feb. 2025. [Online]. Available: https://attack.mitre.org/

[33] Microsoft, "Microsoft defender threat intelligence," 2024, accessed: 2025-06-05. [Online]. Available: https://www.microsoft.com/en-us/security/business/siem-and-xdr/microsoft-defender-threat-intelligence

[34] IBM, "Ibm x-force threat intelligence," 2024, accessed: 2025-06-05. [Online]. Available: https://www.ibm.com/x-force

[35] Google Cloud, "Google cloud threat intelligence," 2024, accessed: 2025-06-05. [Online]. Available: https://cloud.google.com/security/products/threat-intelligence?hl=en

[36] Cisco Talos Intelligence Group, "Cisco talos intelligence," 2024, accessed: 2025-06-05. [Online]. Available: https://talosintelligence.com/

[37] I. Dragos, "Crashoverride: Analysis of the threat to electric grid operations," *Online]: https://dragos. com/blog/crashoverride/CrashOverride-01. pdf*, 2017.

[38] Dragos Inc., "Cyber threat intelligence," 2024, accessed: 2025-06-05. [Online]. Available: https://www.dragos.com/cyber-threat-intelligence/

[39] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ics) security - supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc)," 2011-06-07 2011, https://www.nist.gov/publications/guide-industrial-control-systems-ics-security-supervisory-control-and-data-acquisition. Accessed: 27-09-2023.

[40] B. C. Brusso, "50 years of industrial automation [history]," *IEEE Industry Applications Magazine*, vol. 24, no. 4, pp. 8–11, 2018.

[41] M. Tiegelkamp and K.-H. John, *IEC 61131-3: Programming industrial automation systems*. Springer, 2010.

[42] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE transactions on industrial informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[43] V. R. Segovia and A. Theorin, "History of control history of plc and dcs," *University of Lund*, 2012.

[44] N. Naumenko, "Simatic mc7 code," https://gitlab.com/nnaumenko/mc7, accessed: 02-10-2023.

[45] Siemens, "The intelligent choice for your automation task: Simatic controllers," accessed: 2021-12-8.

[46] T. Alves, "OpenPLC Runtime version 3," https://github.com/thiagoralves/OpenPLC_v3, Sep. 2023, accessed: 27-09-2023. [Online]. Available: https://github.com/thiagoralves/OpenPLC_v3

[47] J. A. Stankovic and R. Rajkumar, "Real-time operating systems," *Real-Time Systems*, vol. 28, no. 2-3, pp. 237–253, 2004.

[48] "Siemens embedded software board support packages," https://static.sw.cdn.siemens.com/siemens-disw-assets/public/5TLWV4zZ3yuS84kwVz3Tqb/en-US/Siemens_SW_embedded_board_support_packages_v9_Nov2022.pdf, Nov. 2022, accessed: 02-10-2023.

[49] "Siemens chooses vxworks to power its high-performance industrial pcs," https://www.windriver.com/themes/Windriver/pdf/Siemens-Customer-Success-Story.pdf, accessed: 03-10-2023.

[50] Z. Basnight, J. Butts, J. Lopez Jr, and T. Dube, "Firmware modification attacks on programmable logic controllers," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 76–84, 2013.

[51] J. Zaddach and A. Costin, "Embedded devices security and firmware reverse engineering," *Black Hat USA*, 2013, https://media.blackhat.com/us-13/US-13-Zaddach-Workshop-on-Embedded-Devices-Security-and-Firmware-Reverse-Engineering-Slides.pdf. Accessed: 03-10-2023.

[52] "How do you perform a firmware update of the distributed IO in STEP 7 / PCS 7?" https://support.industry.siemens.com/cs/document/109783755/how-do-you-perform-a-firmware-update-of-the-distributed-io-in-step-7-pcs-7-?dti=0&pnid=14342&lc=en-US, accessed: 02-10-2023.

[53] R. Automation, "Logix 5000 Controllers Tasks, Programs, and Routines," https://literature.rockwellautomation.com/idc/groups/literature/documents/pm/1756-pm005_-en-p.pdf, p. 69, 2022, accessed: 27-09-2023.

[54] C. M. Ahmed, M. Ochoa, J. Zhou, and A. Mathur, "Scanning the cycle: Timing-based authentication on plcs," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 886–900. [Online]. Available: https://doi.org/10.1145/3433210.3453102

[55] V. M. Igure, S. A. Laughter, and R. D. Williams, "Security issues in scada networks," *computers & security*, vol. 25, no. 7, pp. 498–506, 2006.

[56] Y. Xu, Y. Yang, T. Li, J. Ju, and Q. Wang, "Review on cyber vulnerabilities of communication protocols in industrial control systems," in *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*. IEEE, 2017, pp. 1–6.

[57] E. Biham, S. Bitan, A. Carmel, A. Dankner, U. Malin, and A. Wool, "Rogue7: Rogue engineering-station attacks on s7 simatic plcs," *Black Hat USA*, 2019, https://i.blackhat.com/USA-19/Thursday/us-19-Bitan-Rogue7-Rogue-Engineering-Station-Attacks-On-S7-Simatic-PLCs-wp.pdf. Accessed: 27-09-2023.

[58] J.-P. Thomesse, "Fieldbus technology in industrial automation," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1073–1101, 2005.

[59] S. Kolla, D. Border, and E. Mayer, "Fieldbus networks for control system implementations," in *Proceedings: Electrical Insulation Conference and Electrical Manufacturing and Coil Winding Technology Conference (Cat. No. 03CH37480)*. IEEE, 2003, pp. 493–498.

[60] M. Felser and T. Sauter, "The fieldbus war: history or short break between battles?" in *4th IEEE international workshop on factory communication systems*. IEEE, 2002, pp. 73–80.

[61] "IEEE 802.3 ETHERNET," https://www.ieee802.org/3/, accessed: 02-10-2023.

[62] P. S. Marshall and J. S. Rinaldi, *Industrial Ethernet*. ISA, 2004.

[63] "EtherNet/IP™ | ODVA Technologies | Industrial Automation," https://www.odva.org/technology-standards/key-technologies/ethernet-ip/, accessed: 27-09-2023.

[64] "Mqtt: The standard for iot messaging," https://mqtt.org/, 2022, accessed: 02-10-2023.

[65] "Unified Architecture - opc foundation," https://opcfoundation.org/about/opc-technologies/opc-ua/, accessed: 03-10-2023.

[66] "Cloud Connectivity – IoT PLC: Controllers with MQTT," https://www.wago.com/us/open-automation/cloud-connectivity/iot-plc-controller-with-mqtt, accessed: 27-09-2023.

[67] H. Haskamp, F. Orth, J. Wermann, and A. W. Colombo, "Implementing an opc ua interface for legacy plc-based automation systems using the azure cloud: An icps-architecture with a retrofitted rfid system," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2018, pp. 115–121.

[68] V. Saarimäki, N. Siltala, P. Partanen, J. Vihinen, and R. Tuokko, "Experiences in different fieldbuses used together with pc-based control systems," in *Fieldbus Technology*. Springer, 1999, pp. 164–169.

[69] "SoftPLC," https://en.everybodywiki.com/SoftPLC, Aug. 2020, accessed: 03-10-2023.

[70] D. Dietrich, P. Neumann, and H. F. Schweinzer, "Fieldbus technology: Systems integration, networking, and engineering," in *Proceedings of the Fieldbus Conference FeT'99 in Magdeburg, Federal Republic of Germany*. Springer, 1999.

[71] "1.1 OpenPLC Overview – OpenPLC," https://openplcproject.com/docs/openplc-overview/, accessed: 27-09-2023.

[72] "CODESYS Runtime," https://www.codesys.com/products/codesys-runtime.html, accessed: 27-09-2023.

[73] "CODESYS Inside," https://www.codesys.com/the-system/codesys-inside.html, accessed: 27-09-2023.

[74] T. R. Alves, M. Buratto, F. M. De Souza, and T. V. Rodrigues, "Openplc: An open source alternative to automation," in *IEEE Global Humanitarian Technology Conference (GHTC 2014)*. IEEE, 2014, pp. 585–589.

[75] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '14. London, England, United Kingdom: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2601248.2601268

[76] V. S. Conn, J. C. Valentine, H. M. Cooper, and M. J. Rantz, "Grey literature in meta-analyses," *Nursing research*, vol. 52, no. 4, pp. 256–261, 2003.

[77] "Matrix | MITRE ATT&CK®," https://attack.mitre.org/matrices/ics/, accessed: 02-10-2023.

[78] O. Alexander, "In Pursuit of a Gestalt Visualization: Merging MITRE ATT&CK® for Enterprise and ICS to Communicate," *MITRE ATT&CK®*, Sep. 2020, https://medium.com/mitre-attack/in-pursuit-of-a-gestalt-visualization-merging-mitre-att-ck-for-enterprise-and-ics-to-communicate-3523daa7b580. Accessed: 27-09-2023.

[79] "System firmware, technique t0857 - ics — mitre att&ck®," https://attack.mitre.org/techniques/T0857/, accessed: 04-11-2023.

[80] L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. A. Mohammed, and S. A. Zonouz, "Hey, my malware knows physics! attacking plcs with physical model aware rootkit." in *Network and Distributed System Security Symposium*, 2017.

[81] E. N. Ylmaz, B. Ciylan, S. Gönen, E. Sindiren, and G. Karacayılmaz, "Cyber security in

industrial control systems: Analysis of dos attacks against plcs and the insider effect," in *2018 6th international istanbul smart grids and cities congress and fair (icsg)*. IEEE, 2018, pp. 81–85.

[82] S. McLaughlin and S. Zonouz, "Controller-aware false data injection against programmable logic controllers," in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 2014, pp. 848–853.

[83] S. McLaughlin and P. McDaniel, "Sabot: Specification-based payload generation for programmable logic controllers," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. Raleigh, North Carolina, USA: Association for Computing Machinery, 2012, p. 439–449. [Online]. Available: https://doi.org/10.1145/2382196.2382244

[84] H. Wardak, S. Zhioua, and A. Almulhem, "Plc access control: a security analysis," in *2016 World Congress on Industrial Control Systems Security (WCICSS)*. IEEE, 2016, pp. 1–6.

[85] M. Niedermaier, J.-O. Malchow, F. Fischer, D. Marzin, D. Merli, V. Roth, and A. Von Bodisco, "You snooze, you lose: measuring {PLC} cycle times under attacks," in *12th USENIX Workshop on Offensive Technologies (WOOT 18)*, 2018.

[86] P. H. N. Rajput, E. Sarkar, D. Tychalas, and M. Maniatakos, "Remote non-intrusive malware detection for plcs based on chain of trust rooted in hardware," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 369–384.

[87] M. Krotofil and J. Larsen, "Rocking the pocket book: Hacking chemical plants," *DEFCON Conference*, 2017, https://media.defcon.org/DEF%20CON%2023/DEF%20CON%2023%20presentations/DEF%20CON%2023%20-%20Marina-Krotofil-Jason-Larsen-Rocking-the-Pocketbook-Hacking-Chemical-Plants-WP-UPDATED.pdf. Accessed: 02-10-2023.

[88] D. Formby, P. Srinivasan, A. M. Leonard, J. D. Rogers, and R. A. Beyah, "Who's in control of your control system? device fingerprinting for cyber-physical systems." in *Network and Distributed System Security Symposium*, 2016.

[89] A. Erba, R. Taormina, S. Galelli, M. Pogliani, M. Carminati, S. Zanero, and N. O.

Tippenhauer, "Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems," in *Annual Computer Security Applications Conference*, ser. ACSAC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 480–495. [Online]. Available: https://doi.org/10.1145/3427228.3427660

[90] N. Goldenberg and A. Wool, "Accurate modeling of modbus/tcp for intrusion detection in scada systems," *international journal of critical infrastructure protection*, vol. 6, no. 2, pp. 63–75, 2013.

[91] J. Gardiner, A. Eiffert, P. Garraghan, N. Race, S. Nagaraja, and A. Rashid, "Controller-in-the-middle: Attacks on software defined networks in industrial control systems," in *Proceedings of the 2th Workshop on CPS&IoT Security and Privacy*, ser. CPSIoTSec '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 63–68. [Online]. Available: https://doi.org/10.1145/3462633.3483979

[92] M. Hoeve, "Detecting intrusions in encrypted control traffic," in *Proceedings of the First ACM Workshop on Smart Energy Grid Security*, ser. SEGS '13. Berlin, Germany: Association for Computing Machinery, 2013, p. 23–28. [Online]. Available: https://doi.org/10.1145/2516930.2516945

[93] A. Kleinman and A. Wool, "Accurate modeling of the siemens s7 scada protocol for intrusion detection and digital forensics," *The Journal of Digital Forensics, Security and Law: JDFSL*, vol. 9, no. 2, p. 37, 2014.

[94] T. Choi, G. Bai, R. K. Ko, N. Dong, W. Zhang, and S. Wang, "An analytics framework for heuristic inference attacks against industrial control systems," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2020, pp. 827–835.

[95] P. Kreimel, O. Eigner, and P. Tavolato, "Anomaly-based detection and classification of attacks in cyber-physical systems," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, 2017, pp. 1–6.

[96] A. Ayub, H. Yoo, and I. Ahmed, "Empirical study of plc authentication protocols in industrial

control systems," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 383–397.

[97] N. O. T. Alessandro Erba, Anne Müller, "Resting on feet of clay: Securely bootstrapping opc ua deployments," *Black Hat Europe*, 2021, https://i.blackhat.com/EU-21/Wednesday/EU-21-Erba-Resting-On-Feet-Of-Clay-Securely-Bootstrapping-OPC-UA-Deployments.pdf. Accessed: 27-09-2023.

[98] T. Alves, R. Das, and T. Morris, "Embedding encryption and machine learning intrusion prevention systems on programmable logic controllers," *IEEE Embedded Systems Letters*, vol. 10, no. 3, pp. 99–102, 2018.

[99] N. Sayegh, A. Chehab, I. H. Elhajj, and A. Kayssi, "Internal security attacks on scada systems," in *2013 Third International Conference on Communications and Information Technology (ICCIT)*. IEEE, 2013, pp. 22–27.

[100] J. Gao, Y. Li, Y. Li, T. Gong, F. Xu, Q. Zhao, H. Jiang, and J. Jia, "An effective defense method based on hash authentication against mode-switching attack of ormon plc," in *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*. IEEE, 2022, pp. 976–979.

[101] P. Plesowicz and P. Laszczyk, "Evaluation of secure signal transmission in automatic control using ssh tunneling," in *2013 18th International Conference on Methods & Models in Automation & Robotics (MMAR)*. IEEE, 2013, pp. 672–677.

[102] D. Beresford, "Exploiting siemens simatic s7 plcs," *Black Hat USA*, vol. 16, no. 2, pp. 723–733, 2011, https://media.blackhat.com/bh-us-11/Beresford/BH_US11_Beresford_S7_PLCs_WP.pdf. Accessed: 27-09-2023.

[103] T. Alves, T. Morris, and S.-M. Yoo, "Securing scada applications using openplc with end-to-end encryption," in *Proceedings of the 3rd annual industrial control system security workshop*. IEEE, 2017, pp. 1–6.

[104] Z. Yang, Z. Bao, C. Jin, Z. Liu, and J. Zhou, "Plcrypto: A symmetric cryptographic library

for programmable logic controllers," *IACR Transactions on Symmetric Cryptology*, pp. 170–217, 2021.

[105] R. Santamarta, "Attacking controllogix," https://drive.google.com/file/d/1Ch_1PvDYd1QYcdhZr9NhtpuSGcbnhW1v/, 2012, accessed: 27-09-2023.

[106] Z. Yang, L. He, P. Cheng, J. Chen, D. K. Yau, and L. Du, "{PLC-Sleuth}: Detecting and localizing {PLC} intrusions using control invariants," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 333–348.

[107] S. A. Qasim, A. Ayub, J. Johnson, and I. Ahmed, "Attacking the iec 61131 logic engine in programmable logic controllers," in *International Conference on Critical Infrastructure Protection*. Springer, 2021, pp. 73–95.

[108] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel, "Through the eye of the plc: Semantic security monitoring for industrial processes," in *Proceedings of the 30th Annual Computer Security Applications Conference*, ser. ACSAC '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 126–135. [Online]. Available: https://doi.org/10.1145/2664243.2664277

[109] S. Ponomarev and T. Atkison, "Industrial control system network intrusion detection by telemetry analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 252–260, 2015.

[110] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu, "Watch me, but don't touch me! contactless control flow monitoring via electromagnetic emanations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1095–1108. [Online]. Available: https://doi.org/10.1145/3133956.3134081

[111] S. Dunlap, J. Butts, J. Lopez, M. Rice, and B. Mullins, "Using timing-based side channels for anomaly detection in industrial control systems," *International Journal of Critical Infrastructure Protection*, vol. 15, pp. 12–26, 2016.

[112] R. Zhou, Z. Wang, X. Ji, and W. Xu, "Anomoly detection for plc based on magnetic side

channel," in *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2)*. IEEE, 2020, pp. 3817–3821.

[113] R. Grandgenett, W. Mahoney, and R. Gandhi, "Authentication bypass and remote escalated i/o command attacks," in *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, 2015, pp. 1–7.

[114] S. Biallas, J. Brauer, and S. Kowalewski, "Arcade. plc: A verification platform for programmable logic controllers," in *2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2012, pp. 338–341.

[115] L. Junjiao, X. Lin, C. Xin, W. Hui, H. Li, H. Yan, S. Jiawei, S. Zhiqiang, and L. Sun, "Shadowplcs: A novel scheme for remote detection of industrial process control attacks," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[116] C. Lei, L. Donghong, and M. Liang, "The spear to break the security wall of S7CommPlus," *Black Hat Europe*, p. 12, 2017, https://www.blackhat.com/docs/eu-17/materials/eu-17-Lei-The-Spear-To-Break%20-The-Security-Wall-Of-S7CommPlus-wp.pdf. Accessed: 02-10-2023.

[117] C. Jin, S. Valizadeh, and M. van Dijk, "Snapshotter: Lightweight intrusion detection and prevention system for industrial control systems," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2018, pp. 824–829.

[118] H. Lan, X. Zhu, J. Sun, and S. Li, "Traffic data classification to detect man-in-the-middle attacks in industrial control system," in *2019 6th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 2020, pp. 430–434.

[119] N. O. Tippenhauer, B. Chen, D. Mashima, and D. M. Nicol, "Vbump: Securing ethernet-based industrial control system networks with vlan-based traffic aggregation," in *Proceedings of the 2th Workshop on CPS&IoT Security and Privacy*, ser. CPSIoTSec '21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, p. 3–14. [Online]. Available: https://doi.org/10.1145/3462633.3483983

[120] R. Grandgenett, R. Gandhi, and W. Mahoney, "Exploitation of allen bradley's implementation

of ethernet/ip for denial of service against industrial control systems," in *Proceedings of the 9th International Conference on Cyber Warfare and Security, ICCWS 2014, Academic Conferences and Publishing International Limited, Reading, UK*, 2014, pp. 58–65.

[121] W. Shang, Q. Qiao, M. Wan, and P. Zeng, "Design and implementation of industrial firewall for modbus/tcp." *J. Comput.*, vol. 11, no. 5, pp. 432–438, 2016.

[122] S. Bhatia, N. S. Kush, C. Djamaludin, A. J. Akande, and E. Foo, "Practical modbus flooding attack and detection," in *Proceedings of the Twelfth Australasian Information Security Conference (AISC 2014)[Conferences in Research and Practice in Information Technology, Volume 149]*. Australian Computer Society, 2014, pp. 57–65.

[123] M. Khadpe, P. Binnar, and F. Kazi, "Malware injection in operational technology networks," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2020, pp. 1–6.

[124] H. Yoo, S. Kalle, J. Smith, and I. Ahmed, "Overshadow plc to detect remote control-logic injection attacks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2019, pp. 109–132.

[125] S. Kalle, N. Ameen, H. Yoo, and I. Ahmed, "Clik on plcs! attacking control logic with decompilation and virtual plc," in *Binary Analysis Research (BAR) Workshop, Network and Distributed System Security Symposium (NDSS)*, 2019.

[126] G. Bonney, H. Höfken, B. Paffen, and M. Schuba, "Ics/scada security analysis of a beckhoff cx5020 plc," in *2015 International Conference on Information Systems Security and Privacy (ICISSP)*. IEEE, 2015, pp. 1–6.

[127] S. Adepu, S. Shrivastava, and A. Mathur, "Argus: An orthogonal defense framework to protect public infrastructure against cyber-physical attacks," *IEEE Internet Computing*, vol. 20, no. 5, pp. 38–45, 2016.

[128] A. Al Farooq, J. Marquard, K. George, and T. Moyer, "Detecting safety and security faults in plc systems with data provenance," in *2019 IEEE International Symposium on Technologies for Homeland Security (HST)*. IEEE, 2019, pp. 1–6.

[129] M. M. Cook, A. K. Marnerides, and D. Pezaros, "Plcprint: Fingerprinting memory attacks in programmable logic controllers," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3376–3387, 2023.

[130] S. Gowdanakatte, I. Ray, and S. Hilde Houmb, "Attribute based access control model for protecting programmable logic controllers," in *Proceedings of the 2022 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*, ser. Sat-CPS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 47–56. [Online]. Available: https://doi.org/10.1145/3510547.3517926

[131] J. Gao, J. Li, H. Jiang, Y. Li, and H. Quan, "A new detection approach against attack/intrusion in measurement and control system with fins protocol," in *2020 Chinese Automation Congress (CAC)*. IEEE, 2020, pp. 3691–3696.

[132] D. Tychalas and M. Maniatakos, "Special session: Potentially leaky controller: Examining cache side-channel attacks in programmable logic controllers," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*. IEEE, 2020, pp. 33–36.

[133] A. Wedgbury and K. Jones, "Automated asset discovery in industrial control systems-exploring the problem," in *3rd International Symposium for ICS & SCADA Cyber Security Research 2015 (ICS-CSR 2015) 3*, 2015, pp. 73–83.

[134] J. Klick, S. Lau, D. Marzin, J.-O. Malchow, and V. Roth, "Internet-facing plcs-a new back orifice," *Blackhat USA*, pp. 22–26, 2015, https://www.blackhat.com/docs/us-15/materials/us-15-Klick-Internet-Facing-PLCs-A-New-Back-Orifice-wp.pdf. Accessed: 02-10-2023.

[135] A. F. Murillo, L. F. Cómbita, A. C. Gonzalez, S. Rueda, A. A. Cardenas, and N. Quijano, "A virtual environment for industrial control systems: A nonlinear use-case in attack detection, identification, and response," in *Proceedings of the 4th Annual Industrial Control System Security Workshop*, ser. ICSS '18. San Juan, PR, USA: Association for Computing Machinery, 2018, p. 25–32. [Online]. Available: https://doi.org/10.1145/3295453.3295457

[136] J. Mulder, M. Schwartz, M. Berg, J. R. Van Houten, J. Mario, M. A. K. Urrea, A. A. Clements, and J. Jacob, "Weaselboard: zero-day exploit detection for programmable logic

controllers," *Sandia report SAND2013-8274, Sandia national laboratories*, 2013.

[137] C. Bellettini and J. L. Rrushi, "Vulnerability analysis of scada protocol binaries through detection of memory access taintedness," in *2007 IEEE SMC Information Assurance and Security Workshop*. IEEE, 2007, pp. 341–348.

[138] C. Bellettini and J. Rrushi, "Scada protocol obfuscation: A proactive defense line in scada systems," in *SCADA Security Scientific Symposium*, vol. 154. Digital Bond Press, 2007.

[139] D. Urbina, J. Giraldo, N. O. Tippenhauer, and A. Cardenas, "Attacking fieldbus communications in ics: Applications to the swat testbed," in *Proceedings of the Singapore Cyber-Security Conference (SG-CRC) 2016*. IOS Press, 2016, pp. 75–89.

[140] S. Schorradt, E. Bajramovic, and F. Freiling, "On the feasibility of secure logging for industrial control systems using blockchain," in *Proceedings of the Third Central European Cybersecurity Conference*, ser. CECC 2019. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3360664.3360668

[141] R. Wightman and D. Peterson, "Project Basecamp 3S CoDeSys Vulns and Tools," https://web.archive.org/web/20130806010310/http:/www.digitalbond.com/tools/basecamp/3s-codesys/, Aug. 2013, accessed: 03-10-2023.

[142] A. Abbasi, T. Holz, E. Zambon, and S. Etalle, "Ecfi: Asynchronous control flow integrity for programmable logic controllers," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017, pp. 437–448.

[143] T. Keren, "The Race to Native Code Execution in PLCs," https://claroty.com/blog/research-race-to-native-code-execution-in-plcs, accessed: 03-10-2023.

[144] A. Abbasi, T. Scharnowski, and T. Holz, "Doors of Durin: The Veiled Gate to Siemens S7 Silicon," *Black Hat Europe*, p. 53, 2019. [Online]. Available: https://i.blackhat.com/eu-19/Wednesday/eu-19-Abbasi-Doors-Of-Durin-The-Veiled-Gate-To-Siemens-S7-Silicon.pdf

[145] A. Abbasi and M. Hashemi, "Ghost in the plc designing an undetectable programmable logic controller rootkit via pin control attack," *Black Hat Europe*, vol. 2016, pp. 1–35, 2016.

[146] A. Abbasi and A. Genuise, "Ghost in the plc vs ghostbuster: on the feasibility of detecting pin control attack in programmable logic controllers," in *Ghost in the PLC vs GhostBuster*. Eindhoven University of Technology, 2017.

[147] M. Gjendemsjø, "Creating a weapon of mass disruption: Attacking programmable logic controllers," Master's thesis, Institutt for datateknikk og informasjonsvitenskap, 2013.

[148] W. Alsabbagh and P. Langendörfer, "A stealth program injection attack against s7-300 plcs," in *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, vol. 1. IEEE, 2021, pp. 986–993.

[149] L. Garcia, S. Zonouz, D. Wei, and L. P. De Aguiar, "Detecting plc control corruption via on-device runtime verification," in *2016 Resilience Week (RWS)*. IEEE, 2016, pp. 67–72.

[150] K. Yau and K.-P. Chow, "Plc forensics based on control program logic change detection," *Journal of Digital Forensics, Security and Law*, vol. 10, no. 4, p. 5, 2015.

[151] H. Yoo and I. Ahmed, "Control logic injection attacks on industrial control systems," in *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer, 2019, pp. 33–48.

[152] S. E. McLaughlin, S. A. Zonouz, D. J. Pohly, and P. D. McDaniel, "A trusted safety verifier for process controller code." in *Network and Distributed System Security Symposium*, vol. 14, 2014.

[153] J. Lee, H. Choi, J. Shin, and J. T. Seo, "Detection and analysis technique for manipulation attacks on plc control logic," in *Proceedings of the 2020 ACM International Conference on Intelligent Computing and Its Emerging Applications*, ser. ACM ICEA '20. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3440943.3444742

[154] N. Govil, A. Agrawal, and N. O. Tippenhauer, "On ladder logic bombs in industrial control systems," in *Computer Security*. Springer, 2017, pp. 110–126.

[155] H. Senyondo, P. Sun, R. Berthier, and S. Zonouz, "Plcloud: Comprehensive power grid plc

security monitoring with zero safety disruption," in *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 2015, pp. 809–816.

[156] W. Alsabbagh and P. Langendörfer, "Patch now and attack later-exploiting s7 plcs by time-of-day block," in *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2021, pp. 144–151.

[157] J.-O. Malchow, D. Marzin, J. Klick, R. Kovacs, and V. Roth, "Plc guard: A practical defense against attacks on cyber-physical systems," in *2015 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2015, pp. 326–334.

[158] W. Alsabbagh, C. Kim, and P. Langendörfer, "Good night, and good luck: A control logic injection attack on openplc," EasyChair, Tech. Rep., 2023.

[159] S. Maesschalck, A. Staves, R. Derbyshire, B. Green, and D. Hutchinson, "Snakes and ladder logic: Plc-vbs, a plc control logic vulnerability discovery tool," *arXiv preprint arXiv:2206.06669*, 2022.

[160] S. McLaughlin, "On dynamic malware payloads aimed at programmable logic controllers," in *Proceedings of the 6th USENIX Conference on Hot Topics in Security*, ser. HotSec'11. USA: USENIX Association, 2011, p. 10.

[161] P. H. Narayan Rajput, C. Doumanidis, and M. Maniatakos, "Icspatch: Automated vulnerability localization and non-intrusive hotpatching in industrial control systems using data dependence graphs," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2022.

[162] A. Mera, Y. H. Chen, R. Sun, E. Kirda, and L. Lu, "D-box: Dma-enabled compartmentalization for embedded applications," in *Network and Distributed System Security Symposium*, 2022.

[163] B. Lim, D. Chen, Y. An, Z. Kalbarczyk, and R. Iyer, "Attack induced common-mode failures on plc-based safety system in a nuclear power plant: practical experience report," in *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 2017, pp. 205–210.

[164] A. David and L. George, "Exfiltrating reconnaissance data from air-gapped ics/scada netowrks," *Black Hat Europe*, 2017, https://www.blackhat.com/docs/eu-17/materials/eu-17-Atch-Exfiltrating-Reconnaissance-Data-From-Air-Gapped-Ics-Scada-Networks.pdf. Accessed: 02-10-2023.

[165] S. Senthivel, S. Dhungana, H. Yoo, I. Ahmed, and V. Roussev, "Denial of engineering operations attacks in industrial control systems," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 319–329. [Online]. Available: https://doi.org/10.1145/3176258.3176319

[166] A. Robles-Durazno, N. Moradpoor, J. McWhinnie, G. Russell, and I. Maneru-Marin, "Plc memory attack detection and response in a clean water supply system," *International Journal of Critical Infrastructure Protection*, vol. 26, p. 100300, 2019.

[167] W. Alsabbagh and P. Langendörfer, "No need to be online to attack-exploiting s7-1500 plcs by time-of-day block," in *2022 XXVIII International Conference on Information, Communication and Automation Technologies (ICAT)*. IEEE, 2022, pp. 1–8.

[168] E. F. M. Josephlal, S. Adepu, Z. Yang, and J. Zhou, "Enabling isolation and recovery in plc redundancy framework of metro train systems," *International Journal of Information Security*, vol. 20, no. 6, pp. 783–795, 2021.

[169] D. Formby, S. Durbha, and R. Beyah, "Out of control: Ransomware for industrial control systems," in *RSA conference*, vol. 4, 2017.

[170] J. Luo, M. Kang, E. Bisse, M. Veldink, D. Okunev, S. Kolb, J. G. Tylka, and A. Canedo, "A quad-redundant plc architecture for cyber-resilient industrial control systems," *IEEE Embedded Systems Letters*, vol. 13, no. 4, pp. 218–221, 2020.

[171] N. Zubair, A. Ayub, H. Yoo, and I. Ahmed, "Control logic obfuscation attack in industrial control systems," in *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, 2022, pp. 227–232.

[172] M. Dietz, M. Vielberth, and G. Pernul, "Integrating digital twin security simulations

in the security operations center," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ser. ARES '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3407023.3407039

[173] E. Sarkar, H. Benkraouda, and M. Maniatakos, "I came, i saw, i hacked: Automated generation of process-independent attacks for industrial control systems," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 744–758. [Online]. Available: https://doi.org/10.1145/3320269.3384730

[174] M. Sapir, U. Katz, N. Moshe, S. Brizinov, and A. Preminger, "EVIL PLC ATTACK: WEAPONIZING PLCS," *Team82 Research*, p. 60, 2022, https://claroty-statamic-assets.nyc3.digitaloceanspaces.com/resource-downloads/team82-evil-plc-attack-research-paper-1661285586.pdf. Accessed: 27-09-2023.

[175] B. Green, R. Derbyshire, M. Krotofil, W. Knowles, D. Prince, and N. Suri, "Pcaad: Towards automated determination and exploitation of industrial systems," *Computers & Security*, vol. 110, p. 102424, 2021.

[176] J. H. Castellanos, M. Ochoa, A. A. Cardenas, O. Arden, and J. ZHOU, "Attkfinder: Discovering attack vectors in plc programs using information flow analysis," in *24th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2021)*, ser. RAID '21. San Sebastian, Spain: Association for Computing Machinery, 2021, p. 235–250. [Online]. Available: https://doi.org/10.1145/3471621.3471864

[177] N. Zubair, A. Ayub, H. Yoo, and I. Ahmed, "Pem: Remote forensic acquisition of plc memory in industrial control systems," *Forensic Science International: Digital Investigation*, vol. 40, p. 301336, 2022.

[178] M. Zhang, C.-Y. Chen, B.-C. Kao, Y. Qamsane, Y. Shao, Y. Lin, E. Shi, S. Mohan, K. Barton, J. Moyne *et al.*, "Towards automated safety vetting of plc code in real-world plants," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 522–538.

[179] S. A. Qasim, J. Lopez, and I. Ahmed, "Automated reconstruction of control logic for programmable logic controller forensics," in *International Conference on Information Security*. Springer, 2019, pp. 402–422.

[180] S. Bitan and A. Dankner, "sOfT7: Revealing the Secrets of the Siemens S7 PLCs," *Black Hat USA*, p. 36, 2022, https://i.blackhat.com/USA-22/Wednesday/US-22-Bitan-Revealing-S7-PLCs.pdf. Accessed: 27-09-2023.

[181] G. Walkup, S. Etigowni, D. Xu, V. Urias, and H. W. Lin, "Forensic investigation of industrial control systems using deterministic replay," in *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2020, pp. 1–9.

[182] S. Etigowni, D. J. Tian, G. Hernandez, S. Zonouz, and K. Butler, "Cpac: Securing critical infrastructure with cyber-physical access control," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, ser. ACSAC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 139–152. [Online]. Available: https://doi.org/10.1145/2991079.2991126

[183] C. Schuett, J. Butts, and S. Dunlap, "An evaluation of modification attacks on programmable logic controllers," *International Journal of Critical Infrastructure Protection*, vol. 7, no. 1, pp. 61–68, 2014.

[184] L. McMinn and J. Butts, "A firmware verification tool for programmable logic controllers," in *International Conference on Critical Infrastructure Protection*. Springer, 2012, pp. 59–69.

[185] M. Salehi and S. Bayat-Sarmadi, "Plcdefender: Improving remote attestation techniques for plcs using physical model," *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7372–7379, 2020.

[186] D. Peck and D. Peterson, "Leveraging ethernet card vulnerabilities in field devices," in *SCADA security scientific symposium*, 2009, pp. 1–19.

[187] H. Benkraouda, M. A. Chakkantakath, A. Keliris, and M. Maniatakos, "Snifu: Secure

network interception for firmware updates in legacy plcs," in *2020 IEEE 38th VLSI Test Symposium (VTS)*. IEEE, 2020, pp. 1–6.

[188] D. Tychalas, A. Keliris, and M. Maniatakos, "Stealthy information leakage through peripheral exploitation in modern embedded systems," *IEEE Transactions on Device and Materials Reliability*, vol. 20, no. 2, pp. 308–318, 2020.

[189] P. Krishnamurthy, F. Khorrami, R. Karri, D. Paul-Pena, and H. Salehghaffari, "Process-aware covert channels using physical instrumentation in cyber-physical systems," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2761–2771, 2018.

[190] L. Garcia, H. Senyondo, S. McLaughlin, and S. Zonouz, "Covert channel communication through physical interdependencies in cyber-physical infrastructures," in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 2014, pp. 952–957.

[191] J. Giraldo, D. Urbina, C. Tang, and A. A. Cardenas, "The more the merrier: Adding hidden measurements to secure industrial control systems," in *Proceedings of the 7th Symposium on Hot Topics in the Science of Security*, ser. HotSoS '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3384217.3385624

[192] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg, "Limiting the impact of stealthy attacks on industrial control systems," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, p. 1092–1105. [Online]. Available: https://doi.org/10.1145/2976749.2978388

[193] J. FitzPatrick, "The tao of hardware, the te of implants," *Black Hat, USA*, 2016, https://www.blackhat.com/docs/us-16/materials/us-16-FitzPatrick-The-Tao-Of-Hardware-The-Te-Of-Implants-wp.pdf. Accessed: 02-10-2023.

[194] M. K. Choi, C. Y. Yeun, and P. H. Seong, "A novel monitoring system for the data integrity of

reactor protection system using blockchain technology," *IEEE Access*, vol. 8, pp. 118 732–118 740, 2020.

[195] M. Xiao, J. Wu, C. Long, and S. Li, "Construction of false sequence attack against plc based power control system," in *2016 35th Chinese Control Conference (CCC)*. IEEE, 2016, pp. 10 090–10 095.

[196] S. Shrivastava, G. R. MR, and A. Mathur, "Pcat: Plc command analysis tool for automatic incidence response in water treatment plants," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 2151–2159.

[197] M. K. Alexander Bolshev, "Never trust your inputs: Causing 'catastrophic physical consequences' from the sensor (or how to fool adc)," *Black Hat Asia*, 2016, https://www.blackhat.com/docs/asia-16/materials/asia-16-Bolshev-Never-Trust-Your-Inputs-Causing-Catastrophic-Physical-Consequences-From-The-Sensor.pdf. Accessed: 27-09-2023.

[198] H. Pearce, S. Pinisetty, P. S. Roop, M. M. Kuo, and A. Ukil, "Smart i/o modules for mitigating cyber-physical attacks on industrial control systems," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4659–4669, 2019.

[199] M. Hildebrandt, K. Lamshöft, J. Dittmann, T. Neubert, and C. Vielhauer, "Information hiding in industrial control systems: An opc ua based supply chain attack and its detection," in *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*, ser. IH&MMSec '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 115–120. [Online]. Available: https://doi.org/10.1145/3369412.3395068

[200] H. R. Ghaeini, M. Chan, R. Bahmani, F. Brasser, L. Garcia, J. Zhou, A.-R. Sadeghi, N. O. Tippenhauer, and S. Zonouz, "{PAtt}: Physics-based attestation of control systems," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*, 2019, pp. 165–180.

[201] T. Walita, A. Erba, J. H. Castellanos, and N. O. Tippenhauer, "Blind concealment from reconstruction-based attack detectors for industrial control systems via backdoor attacks,"

in *Proceedings of the 9th ACM Cyber-Physical System Security Workshop*, ser. CPSS '23. Melbourne, VIC, Australia: Association for Computing Machinery, 2023, p. 36–47. [Online]. Available: https://doi.org/10.1145/3592538.3594271

[202] A. Robles-Durazno, N. Moradpoor, J. McWhinnie, and G. Russell, "Waterleakage: A stealthy malware for data exfiltration on industrial control systems using visual channels," in *2019 IEEE 15th International Conference on Control and Automation (ICCA)*. IEEE, 2019, pp. 724–731.

[203] R. Spenneberg, M. Brüggemann, and H. Schwartke, "Plc-blaster: A worm living solely in the plc," *Black Hat Asia*, vol. 16, pp. 1–16, 2016, https://www.blackhat.com/docs/asia-16/materials/asia-16-Spenneberg-PLC-Blaster-A-Worm-Living-Solely-In-The-PLC-wp.pdf. Accessed: 03-10-2023.

[204] U. Chatterjee, P. Santikellur, R. Sadhukhan, V. Govindan, D. Mukhopadhyay, and R. S. Chakraborty, "United we stand: A threshold signature scheme for identifying outliers in plcs," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–2.

[205] M. Eckhart and A. Ekelhart, "Towards security-aware virtual environments for digital twins," in *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*, ser. CPSS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 61–72. [Online]. Available: https://doi.org/10.1145/3198458.3198464

[206] Y. Zhang, Z. Sun, L. Yang, Z. Li, Q. Zeng, Y. He, and X. Zhang, "All your plcs belong to me: Ics ransomware is realistic," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2020, pp. 502–509.

[207] W. Zhang, Y. Jiao, D. Wu, S. Srinivasa, A. De, S. Ghosh, and P. Liu, "Armor plc: A platform for cyber security threats assessments for plcs," *Procedia Manufacturing*, vol. 39, pp. 270–278, 2019.

[208] R. D. Marina Krotofil, "Greetings from the '90s: Exploiting the design of industrial controllers in modern settings," *Black Hat Europe*, 2021, https:

//i.blackhat.com/EU-21/Wednesday/EU-21-Krotofil-Greetings-from-the-90s-Exploiting-the-Design-of-Industrial-Controllers-in-Modern-Settings.pdf. Accessed: 02-10-2023.

[209] A. Ayub, N. Zubair, H. Yoo, W. Jo, and I. Ahmed, "Gadgets of gadgets in industrial control systems: Return oriented programming attacks on plcs," in *2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2023, pp. 215–226.

[210] W. Alsabbagh, S. Amogbonjaye, D. Urrego, and P. Langendörfer, "A stealthy false command injection attack on modbus based scada systems," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*. IEEE, 2023, pp. 1–9.

[211] S. McLaughlin, "Cps: Stateful policy enforcement for control system device usage," in *Proceedings of the 29th Annual Computer Security Applications Conference*, ser. ACSAC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 109–118. [Online]. Available: https://doi.org/10.1145/2523649.2523673

[212] M. Cheng and S. Yang, "Taking Apart and Taking Over ICS & SCADA Ecosystems: A Case Study of Mitsubishi Electric," *DEF CON 29*, 2021, https://media.defcon.org/DEF%20CON%2029/DEF%20CON%2029%20presentations/Mars%20Cheng%20Selmon%20Yang%20-%20Taking%20Apart%20and%20Taking%20Over%20ICS%20%26%20SCADA%20Ecosystems%20-%20%20A%20Case%20Study%20of%20Mitsubishi%20Electric.pdf. Accessed: 03-10-2023.

[213] T. Vidas and N. Christin, "Evading android runtime analysis via sandbox detection," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '14. Kyoto, Japan: Association for Computing Machinery, 2014, p. 447–458. [Online]. Available: https://doi.org/10.1145/2590296.2590325

[214] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, "Dynamic malware analysis in the modern era—a state of the art survey," *ACM Comput. Surv.*, vol. 52, no. 5, sep 2019. [Online]. Available: https://doi.org/10.1145/3329786

[215] T. Sasaki, A. Fujita, C. H. Gañán, M. van Eeten, K. Yoshioka, and T. Matsumoto, "Exposed infrastructures: Discovery, attacks and remediation of insecure ics remote management

devices," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 2379–2396.

[216] R. Pickren, T. Shekari, S. Zonouz, and R. Beyah, "Compromising industrial processes using web-based programmable logic controller malware," in *Network and Distributed System Security Symposium (NDSS)*, 2024.

[217] "Vulnerability disclosure policy template — cisa," https://www.cisa.gov/vulnerability-disclosure-policy-template, accessed: 14-11-2023.

[218] N. S. Almakhdhub, A. A. Clements, M. Payer, and S. Bagchi, "Benchiot: A security benchmark for the internet of things," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019, pp. 234–246.

[219] "Soft PLCs: Revisiting the industrial innovator's dilemma," https://iot-analytics.com/software-based-plcs-revisiting-industrial-innovators-dilemma/, Oct. 2021, accessed: 03-10-2023.

[220] "Codesys automation server — codesys store international," https://store.codesys.com/en/codesys-automation-server.html, accessed: 08-11-2023.

[221] R. Sun, A. Mera, L. Lu, and D. Choffnes, "Sok: Attacks on industrial control logic and formal verification-based defenses," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 385–402.

[222] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, "A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2351–2383, 2021.

[223] D. Tychalas, H. Benkraouda, and M. Maniatakos, "Icsfuzz: Manipulating i/os and repurposing binary code to enable instrumented fuzzing in ics control applications," in *30th USENIX Security Symposium (USENIX Security 21)*, Virtual Event, 2021.

[224] M. Niedermaier, F. Fischer, and A. von Bodisco, "Propfuzz—an it-security fuzzing framework for proprietary ics protocols," in *2017 International conference on applied electronics (AE)*. IEEE, 2017, pp. 1–4.

[225] T. Wang, Q. Xiong, H. Gao, Y. Peng, Z. Dai, and S. Yi, "Design and implementation of

fuzzing technology for opc protocol," in *2013 Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, 2013, pp. 424–428.

[226] D. Fang, Z. Song, L. Guan, P. Liu, A. Peng, K. Cheng, Y. Zheng, P. Liu, H. Zhu, and L. Sun, "Ics3fuzzer: A framework for discovering protocol implementation bugs in ics supervisory software by fuzzing," in *Annual Computer Security Applications Conference*, ser. ACSAC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 849–860. [Online]. Available: https://doi.org/10.1145/3485832.3488028

[227] A. Keliris and M. Maniatakos, "ICSREF: A framework for automated reverse engineering of industrial control systems binaries," in *Network and Distributed System Security Symposium (NDSS)*, 2019.

[228] N. Ortiz, A. A. Cardenas, and A. Wool, "Scada world: An exploration of the diversity in power grid networks," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 8, no. 1, pp. 1–32, 2024.

[229] D. Sullivan, E. Luiijf, and E. J. Colbert, *Components of industrial control systems*. Springer, 2016.

[230] L. Salazar, S. Castro, J. Lozano, K. Koneru, E. Zambon, B. Huang, R. Baldick, M. Krotofil, A. Rojas, and A. A. Cardenas, "A tale of two Industroyers: It was the season of darkness," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 162–162.

[231] T. J. Williams, "The purdue enterprise reference architecture," *Computers in industry*, vol. 24, no. 2-3, pp. 141–158, 1994.

[232] S. AG, "Training document for the company-wide automation solution totally integrated automation (tia)," https://www.automation.siemens.com/sce-static/learning-training-documents/classic/advanced-programming/b04-data-blocks-en.pdf, accessed: 27-09-2023.

[233] ——, "System software for s7-300/400 system and standard functions vol-

ume 1/2," https://cache.industry.siemens.com/dl/files/604/44240604/att_67003/v1/s7sfc_en-EN.pdf, accessed: 27-09-2023.

[234] "How can you protect your S7 program with a password for an S7-300 and ET 200 CPU?" https://support.industry.siemens.com/cs/document/10154913/how-can-you-protect-your-s7-program-with-a-password-for-an-s7-300-and-et-200-cpu-?dti=0&lc=en-US, accessed: 02-10-2023.

[235] R. Automation, "Logix 5000 Controllers Security," https://literature.rockwellautomation.com/idc/groups/literature/documents/pm/1756-pm016_-en-p.pdf, p. 58, 2022, accessed: 27-09-2023.

[236] "Encrypted block protection for fbs and fcs from step 7 v5.5," https://support.industry.siemens.com/cs/document/45632073/how-do-you-install-the-improved-block-protection-for-fbs-and-fcs-in-step-7-v5-5-onwards-?dti=0&lc=en-AO, accessed: 02-10-2023.

[237] "Top 20 Secure PLC Coding Practices," https://plc-security.com/, accessed: 03-10-2023.

[238] M. Hoffman and G. Cedillo, "Value of PLC Key Switch Monitoring to Keep Critical Systems More Secure | Dragos," https://www.dragos.com/blog/industry-news/value-of-plc-key-switch-monitoring/, Aug. 2021, accessed: 03-10-2023.

[239] "ControlLogix System User Manual," https://literature.rockwellautomation.com/idc/groups/literature/documents/um/1756-um001_-en-p.pdf, accessed: 02-10-2023.

[240] "Hardware configuration changes in RUN mode," https://cache.industry.siemens.com/dl/files/505/14040505/att_47225/v1/cir_en.pdf, accessed: 02-10-2023.

[241] "Examples for the SIMATIC S7-1200 / S7-1500 Web Server," https://cache.industry.siemens.com/dl/files/496/68011496/att_959527/v2/68011496_Examples_for_S7WebServer_DOC_v21_en.pdf, p. 85, 2018, accessed: 02-10-2023.

[242] S. A. Milinković and L. R. Lazić, "Industrial plc security issues," in *2012 20th Telecommunications Forum (TELFOR)*. IEEE, 2012, pp. 1536–1539.

[243] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security and Privacy*, vol. 9, no. 3, p. 49–51, may 2011. [Online]. Available: https://doi.org/10.1109/MSP.2011.67

[244] A. Dragos, "Trisis malware: Analysis of safety system targeted malware," https://www.dragos.com/resource/trisis-analyzing-safety-system-targeting-malware/, 2017, accessed: 27-09-2023.

[245] I. Dragos, "Chernovite's pipedream malware targeting industrial control systems (ics)," https://www.dragos.com/blog/industry-news/chernovite-pipedream-malware-targeting-industrial-control-systems/, 2022, accessed: 27-09-2023.

[246] Cybersecurity and I. S. A. (CISA), "Apt cyber tools targeting ics/scada devices," https://www.cisa.gov/uscert/ncas/alerts/aa22-103a, 2022, accessed: 02-10-2023.

[247] J. Slowik, "Anatomy of an attack: Detecting and defeating crashoverride," *VB2018, October*, 2018.

[248] K. E. Hemsley, E. Fisher *et al.*, "History of industrial control system cyber incidents," Idaho National Lab.(INL), Idaho Falls, ID (United States), Tech. Rep., 2018.

[249] United States Space Force, "Global positioning system ¿ space operations command (spoc) ¿ display," Feb. 2023. [Online]. Available: https://www.spoc.spaceforce.mil/About-Us/Fact-Sheets/Display/Article/2381726/global-positioning-system

[250] European Union Agency for the Space Programme, "Copernicus — eu agency for the space programme," Apr. 2024. [Online]. Available: https://www.euspa.europa.eu/eu-space-programme/copernicus

[251] V. Charlotte and P. Walter, "A world without satellite data as a result of a global cyber-attack," *Space Policy*, vol. 59, p. 101458, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0265964621000503

[252] J. Drmola and T. Hubik, "Kessler syndrome: System dynamics model," *Space Policy*, vol. 44, pp. 29–39, 2018.

[253] M. Holmes, "The growing risk of a major satellite cyber attack," May 2023.

[Online]. Available: https://interactive.satellitetoday.com/the-growing-risk-of-a-major-satellite-cyber-attack/

[254] M. Semanik and P. Crotty, "U.s. private space launch industry is out of this world," Nov. 2023. [Online]. Available: https://www.usitc.gov/publications/332/executive_briefings/ebot_us_private_space_launch_industry_is_out_of_this_world.pdf

[255] R. L. Staehle, B. Anderson, B. Betts, D. Blaney, C. Chow, L. Friedman, H. Hemmati, D. Jones, A. Klesh, P. Liewer *et al.*, "Interplanetary cubesats: opening the solar system to a broad community at lower cost," NASA, Tech. Rep., 2012.

[256] S. L. O. California Polytechnic State University, "Earth station - polysat," Apr. 2024. [Online]. Available: https://www.polysat.org/earth-station

[257] T. Scharnowski, F. Buchmann, S. Wörner, and T. Holz, "A case study on fuzzing satellite firmware," in *Workshop on the Security of Space and Satellite Systems (SpaceSec)*, 2023.

[258] U. Planta, J. Rederlechner, G. Marra, and A. Abbasi, "Let me do it for you: On the feasibility of inter-satellite friendly jamming," in *2024 Security for Space Systems (3S)*, 2024, pp. 1–6.

[259] J. Willbold, T. Cloosters, S. Wörner, F. Buchmann, M. Schloegel, L. Davi, and T. Holz, " Space RadSim: Binary-Agnostic Fault Injection to Evaluate Cosmic Radiation Impact on Exploit Mitigation Techniques in Space ," in *2025 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2025, pp. 1047–1063. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00139

[260] J. Pavur and I. Martinovic, "Building a launchpad for satellite cyber-security research: lessons from 60 years of spaceflight," *Journal of Cybersecurity*, vol. 8, no. 1, p. tyac008, 2022.

[261] The MITRE Corporation, "Groups — mitre att&ck," Apr. 2024. [Online]. Available: https://attack.mitre.org/groups/

[262] T. Holz and F. Raynal, "Detecting honeypots and other suspicious environments," in *Proceedings from the sixth annual IEEE SMC information assurance workshop*. West Point, NY, USA: IEEE, 2005, pp. 29–36.

[263] F. Cohen, "The use of deception techniques: Honeypots and decoys," *Handbook of Information Security*, vol. 3, no. 1, pp. 646–655, 2006.

[264] ——, "Deception toolkit," Mar. 1998. [Online]. Available: http://all.net/dtk/

[265] L. Franceschi-Bicchierai, "Thousands of new honeypots deployed across israel to catch hackers," Nov. 2023. [Online]. Available: https://techcrunch.com/2023/11/20/thousands-of-new-honeypots-deployed-across-israel-to-catch-hackers/

[266] M. Burgess, "A clever honeypot tricked hackers into revealing their secrets," Aug. 2023. [Online]. Available: https://www.wired.com/story/hacker-honeypot-go-secure/

[267] S. Hilt, F. Maggi, C. Perine, L. Remorin, M. Rösler, and R. Vosseler, "Caught in the act: Running a realistic factory honeypot to capture real threats," *Trend Micro Research*, 2020.

[268] Shared Threat Intelligence for Network Gatekeeping and Automated Response (STINGAR), "About - stingar," Apr. 2024. [Online]. Available: https://stingar.security.duke.edu/about-2/

[269] E. López-Morales, C. Rubio-Medrano, A. Doupé, Y. Shoshitaishvili, R. Wang, T. Bao, and G.-J. Ahn, "Honeyplc: A next-generation honeypot for industrial control systems," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 279–291. [Online]. Available: https://doi.org/10.1145/3372297.3423356

[270] B. Acharya, M. Saad, A. E. Cinà, L. Schönherr, H. D. Nguyen, A. Oest, P. Vadrevu, and T. Holz, "Conning the crypto conman: End-to-end analysis of cryptocurrency-based technical support scams," in *2024 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2024. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00156

[271] N. Boschetti, N. G. Gordon, and G. Falco, "Space cybersecurity lessons learned from the viasat cyberattack," in *ASCEND 2022*. Aerospace Research Central, 2022, p. 4380.

[272] R. Bisping, J. Willbold, M. Strohmeier, and V. Lenders, "Wireless signal injection attacks on VSAT satellite modems," in *33rd USENIX Security Symposium (USENIX Security 24)*.

Philadelphia, PA: USENIX Association, Aug. 2024, pp. 6075–6091. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/bisping

[273] G. Kavallieratos and S. Katsikas, "An exploratory analysis of the last frontier: A systematic literature review of cybersecurity in space," *International Journal of Critical Infrastructure Protection*, p. 100640, 2023.

[274] J. Willbold, M. Schloegel, R. Bisping, M. Strohmeier, T. Holz, and V. Lenders, "Vsaster: Uncovering inherent security issues in current vsat system practices," in *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2024, pp. 288–299.

[275] NASA, "What are smallsats and cubesats?" [Online]. Available: https://www.nasa.gov/what-are-smallsats-and-cubesats/

[276] E. López-Morales, U. Planta, G. Marra, C. González, J. Hopkins, M. Garoosi, E. Obreque, C. Rubio-Medrano, and A. Abbasi, "Honeysat: A network-based satellite honeypot framework," 2025, arXiv:2505.24008 https://arxiv.org/abs/2505.24008. [Online]. Available: https://arxiv.org/abs/2505.24008

[277] J. Vestergaard, "mushorg/conpot: Ics/scada honeypot," Mar. 2024. [Online]. Available: https://github.com/mushorg/conpot

[278] N. Provos and T. Holz, *Virtual honeypots: from botnet tracking to intrusion detection*. Boston, MA, USA: Pearson Education, 2007.

[279] G. Marra, U. Planta, P. Wüstenberg, and A. Abbasi, "On the feasibility of cubesats application sandboxing for space missions," in *Workshop on the Security of Space and Satellite Systems (SpaceSec)*, 2024.

[280] M. Oosterhof, "cowrie/cowrie: Cowrie ssh/telnet honeypot https://cowrie.readthedocs.io," Apr. 2024. [Online]. Available: https://github.com/cowrie/cowrie

[281] N. Ilg, P. Duplys, D. Sisejkovic, and M. Menth, "A survey of contemporary open-source honeypots, frameworks, and tools," *Journal of Network and Computer Applications*, vol.

220, p. 103737, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S108480452300156X

[282] J. Nazario, "paralax/awesome-honeypots: an awesome list of honeypot resources," Mar. 2024. [Online]. Available: https://github.com/paralax/awesome-honeypots

[283] E. López-Morales, U. Planta, C. Rubio-Medrano, A. Abbasi, and A. A. Cardenas, "SoK: Security of programmable logic controllers," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 7103–7122. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/lopez-morales

[284] L. Salazar, E. López-Morales, J. Lozano, C. Rubio-Medrano, and A. A. Cárdenas, "Icsnet: A hybrid-interaction honeynet for industrial control systems," in *Proceedings of the Sixth Workshop on CPS&IoT Security and Privacy*, ser. CPSIoTSec'24. New York, NY, USA: Association for Computing Machinery, 2024, p. 68–79. [Online]. Available: https://doi.org/10.1145/3690134.3694813

[285] M. Lucchese, F. Lupia, M. Merro, F. Paci, N. Zannone, and A. Furfaro, "Honeyics: A high-interaction physics-aware honeynet for industrial control systems," in *Proceedings of the 18th International Conference on Availability, Reliability and Security*, ser. ARES '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3600160.3604984

[286] J. Daubert, D. Boopalan, M. Mühlhäuser, and E. Vasilomanolakis, "Honeydrone: A medium-interaction unmanned aerial vehicle honeypot," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–6.

[287] U.S. Geological Survey, "Uncrewed aerial systems — u.s. geological survey," Feb. 2004. [Online]. Available: https://www.usgs.gov/calval/uncrewed-aerial-systems

[288] M. Conti, F. Trolese, and F. Turrin, "Icspot: A high-interaction honeypot for industrial control systems," in *2022 International Symposium on Networks, Computers and Communications (ISNCC)*, 2022, pp. 1–4.

[289] The European Space Agency, "Esa - telemetry & telecommand," Mar. 2013. [Online]. Available: https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Onboard_Computers_and_Data_Handling/Telemetry_Telecommand

[290] L. Wood, "Introduction to satellite constellations: orbital types, uses and related facts," Jul. 2006. [Online]. Available: https://savi.sourceforge.io/about/lloyd-wood-isu-summer-06-constellations-talk.pdf

[291] D. A. Vallado and P. J. Cefola, "Two-line element sets–practice and use," in *63rd International Astronautical Congress*. Naples, Italy: International Astronautical Federation, 2012, pp. 1–14.

[292] NASA Goddard Space Flight Center, "Flight training: Introduction." [Online]. Available: https://solc.gsfc.nasa.gov/modules/missionops/mainMenu_textOnly.php

[293] J. A. Gutierrez Ahumada, K. Doerksen, and S. Zeller, "Automated fleet commissioning workflows at planet," in *AIAA/USU Conference on Small Satellites, Technical Session 12: Constellation Missions, SSC1-XII-04*, Logan, UT, 2021. [Online]. Available: https://digitalcommons.usu.edu/smallsat/2021/all2021/214/

[294] Space and Planetary Exploration Laboratory, University of Chile, "SPEL / SUCHAI-Flight-Software · GitLab," Feb. 2024. [Online]. Available: https://gitlab.com/spel-uchile/suchai-flight-software

[295] GomSpace, "Nanocom ms100 datasheet," Mar. 2021. [Online]. Available: https://gomspace.com/UserFiles/Subsystems/datasheet/gs-ds-nanocom-ms100-13.pdf

[296] M. Merri, A. Ercolani, D. Guerrucci, V. Reggestad, and D. Verrier, "Cutting the cost of esa mission ground software," May 2007. [Online]. Available: https://www.esa.int/esapub/bulletin/bulletin130/bul130g_merri.pdf

[297] A. Csete, "Gpredict: Free, real-time satellite tracking and orbit prediction software," Dec. 2023. [Online]. Available: https://oz9aec.dk/gpredict/

[298] B. D. Yost, "Nasa ssri knowledge base — detailed design and analysis ¿

subsystem design ¿ command and data handling," Jun. 2023. [Online]. Available: https://s3vi.ndc.nasa.gov/ssri-kb/topics/32/

[299] D. McComas, J. Wilmot, and A. Cudmore, "The core flight system (cfs) community: Providing low cost solutions for small spacecraft," in *Annual AIAA/USU Conference on Small Satellites*. Logan, UT: Utah State University, University Libraries, 2016.

[300] R. Bocchino, T. Canham, G. Watney, L. Reder, and J. Levison, "F prime: An open-source framework for small-scale flight software systems," in *AIAA/USU Conference on Small Satellites, Advanced Technologies II, SSC-18-XII-04*, Logan, UT, 2018. [Online]. Available: https://digitalcommons.usu.edu/smallsat/2018/all2018/328/

[301] Xplore, Inc, "kubos/kubos," Feb. 2024, original-date: 2018-01-18T14:55:14Z. [Online]. Available: https://github.com/kubos/kubos

[302] DLR, "Open modUlar sofTware PlatfOrm for SpacecrafT," 2022. [Online]. Available: https://www.github.com/dlr-ry/outpost-core

[303] C. E. Gonzalez, C. J. Rojas, A. Bergel, and M. A. Diaz, "An architecture-tracking approach to evaluate a modular and extensible flight software for cubesat nanosatellites," *IEEE Access*, vol. 7, pp. 126 409–126 429, 2019.

[304] The European Space Agency, "Esa - about payload systems," Apr. 2024. [Online]. Available: https://www.esa.int/Enabling_Support/Space_Engineering_Technology/About_Payload_Systems

[305] NASA, "What is remote sensing? — earthdata," Aug. 2019. [Online]. Available: https://www.earthdata.nasa.gov/learn/backgrounders/remote-sensing

[306] Y. Shoji, "libcsp/libcsp," Feb. 2024. [Online]. Available: https://github.com/libcsp/libcsp

[307] The ZeroMQ authors, "ZeroMQ — get started," 2024. [Online]. Available: https://zeromq.org/get-started/

[308] M. M. Sam Cooper, "Ccsds mission operations," https://indico.esa.int/event/62/contributions/2797/attachments/2307/2667/1235_-_mission-operation-services---future-trends_Presentation.pdf, [Accessed 20-01-2025].

[309] M. Raza, "What are ttps? tactics, techniques & procedures explained," Apr. 2024. [Online]. Available: https://www.splunk.com/en_us/blog/learn/ttp-tactics-techniques-procedures.html

[310] National Institute of Standards and Technology (NIST), "tactics, techniques, and procedures (ttp) - glossary," Apr. 2024. [Online]. Available: https://csrc.nist.gov/glossary/term/tactics_techniques_and_procedures

[311] The Aerospace Corporation, "Sparta," Apr. 2024. [Online]. Available: https://sparta.aerospace.org/

[312] E. S. Agency, "ESA SPACE-SHIELD," 2023. [Online]. Available: https://spaceshield.esa.int/#

[313] Nmap, "Os detection," Jan. 2025. [Online]. Available: https://nmap.org/book/man-os-detection.html

[314] E. Kulu, "SUCHAI 2 @ Nanosats Database — nanosats.eu," https://www.nanosats.eu/sat/suchai-2, 2022, [Accessed 29-04-2024].

[315] Qualtrics, "What is a likert scale?" Jan. 2025. [Online]. Available: https://www.qualtrics.com/experience-management/research/likert-scale/

[316] Space Applications Services, "Yamcs Mission Control," https://yamcs.org, 2025, [Accessed 23-01-2025].

[317] J. Matherly, "Complete guide to shodan," *Shodan, LLC (2016-02-25)*, vol. 1, 2015.

[318] P. W. José Manual Diez, Fabian Krech, "Raccoon os," https://gitlab.com/rccn, [Accessed 20-01-2025].

[319] U.S. Geological Survey, "Earthexplorer — u.s. geological survey," Nov. 2022. [Online]. Available: https://www.usgs.gov/tools/earthexplorer

[320] Docker Inc., "Docker: Accelerated container application development," Apr. 2024. [Online]. Available: https://www.docker.com/

[321] US Department of Transportation, "Automated Vehicles Comprehensive Plan," https:

//www.transportation.gov/sites/dot.gov/files/2021-01/USDOT_AVCP.pdf, 2021, [Online; accessed 06-March-2025].

[322] Tesla, "Autopilot," Aug. 2025. [Online]. Available: https://www.tesla.com/autopilot

[323] Z. Tang, K. Serag, S. Zonouz, Z. B. Celik, D. Xu, and R. Beyah, "Eracan: Defending against an emerging can threat model," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1894–1908. [Online]. Available: https://doi.org/10.1145/3658644.3690267

[324] Amazon Web Services, Inc., "Secure iot gateway, iot gateway device - aws iot core - aws," Feb. 2025. [Online]. Available: https://aws.amazon.com/iot-core/

[325] HiveMQ, "Why mqtt has become the de facto standard for the connected car," Oct. 2020. [Online]. Available: https://www.hivemq.com/blog/mqtt-standard-for-connected-car/

[326] A. Mehran and D. Ulus, "Runtime verification containers for publish/subscribe networks," 2024. [Online]. Available: https://arxiv.org/abs/2408.06380

[327] HiveMQ, "Bmw car-sharing application relies on hivemq for reliable connectivity," Apr. 2024. [Online]. Available: https://www.hivemq.com/case-studies/bmw-mobility-services/

[328] ——, "Hivemq – unlock the value of your data with mqtt," Feb. 2025. [Online]. Available: https://www.hivemq.com/

[329] ——, "Authentication with username and password - mqtt security fundamentals," Mar. 2024. [Online]. Available: https://www.hivemq.com/blog/mqtt-security-fundamentals-authentication-username-password/

[330] M. Hernandez, "Whispers of the machines: Exposing mqtt hidden talks," Aug. 2024. [Online]. Available: https://www.protect.airbus.com/blog/exposing-mqtt-hidden-talks

[331] M. S. Harsha, B. M. Bhavani, and K. Kundhavai, "Analysis of vulnerabilities in mqtt security using shodan api and implementation of its countermeasures via authentication and acls," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2018, pp. 2244–2250.

[332] CrowdStrike, "Command and control (c&c) attack: Definition and prevention," https://www.crowdstrike.com/en-us/cybersecurity-101/cyberattacks/command-and-control-cac-attack/, 2024, accessed: 2025-05-15.

[333] J. Zhou, X. Wu, Y. Lv, X. Li, and Z. Liu, "Recent progress on the study of multi-vehicle coordination in cooperative attack and defense: an overview," *Asian Journal of Control*, vol. 24, no. 2, pp. 794–809, 2022.

[334] L. Salazar, S. R. Castro, J. Lozano, K. Koneru, E. Zambon, B. Huang, R. Baldick, M. Krotofil, A. Rojas, and A. A. Cardenas, "A tale of two industroyers: It was the season of darkness," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024, pp. 312–330.

[335] Amazon Web Services, Inc., "Cm-s02 vehicle connectivity management," Feb. 2025. [Online]. Available: https://docs.aws.amazon.com/wellarchitected/latest/connected-mobility-lens/cm-s02-vehicle-connectivity-management.html

[336] Microsoft, "Communicate with an iot hub using the mqtt protocol," Feb. 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/iot/iot-mqtt-connect-to-iot-hub

[337] Google, "Google cloud for automotive," Feb. 2025. [Online]. Available: https://cloud.google.com/solutions/automotive?hl=en

[338] MQTT.org, "Mqtt: The standard for iot messaging," Dec. 2024. [Online]. Available: https://mqtt.org/

[339] HiveMQ, "Tls/ssl - mqtt security fundamentals," Mar. 2024. [Online]. Available: https://www.hivemq.com/blog/mqtt-security-fundamentals-tls-ssl/

[340] E. S. Agency, "ESA SPACE-SHIELD," 2023. [Online]. Available: https://spaceshield.esa.int/#

[341] H. Wen, Q. A. Chen, and Z. Lin, "Plug-N-Pwned: Comprehensive vulnerability analysis of OBD-II dongles as a new Over-the-Air attack surface in automotive IoT," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 949–965. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/wen

[342] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher,

A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *20th USENIX security symposium (USENIX Security 11)*, 2011.

[343] J. Nadeau, "83% of organizations reported insider attacks in 2024," Nov. 2024. [Online]. Available: https://www.ibm.com/think/insights/83-percent-organizations-reported-insider-threats-2024

[344] Claroty, "Inside a new ot/iot cyberweapon: Iocontrol," Dec. 2024. [Online]. Available: https://claroty.com/team82/research/inside-a-new-ot-iot-cyber-weapon-iocontrol

[345] C. Zhao, J. S. Gill, P. Pisu, and G. Comert, "Detection of false data injection attack in connected and automated vehicles via cloud-based sandboxing," *Trans. Intell. Transport. Sys.*, vol. 23, no. 7, p. 9078–9088, Jul. 2022. [Online]. Available: https://doi.org/10.1109/TITS.2021.3090361

[346] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[347] The CARLA team, "carla 0.9.15," Nov. 2023. [Online]. Available: https://pypi.org/project/carla/

[348] The Eclipse Foundation, "paho-mqtt 2.1.0," Apr. 2024. [Online]. Available: https://pypi.org/project/paho-mqtt/

[349] Eclipse Mosquitto, "Eclipse mosquitto," Jan. 2025. [Online]. Available: https://github.com/eclipse-mosquitto/mosquitto

[350] SCADACS, "Plcinject," Aug. 2015. [Online]. Available: https://github.com/SCADACS/PLCinject

[351] Dragos Inc., "Chernovite's pipedream malware targeting industrial control systems (ics)," Apr. 2022. [Online]. Available: https://www.dragos.com/blog/industry-news/chernovite-pipedream-malware-targeting-industrial-control-systems/

[352] U. Morelli, I. Vaccari, S. Ranise, and E. Cambiaso, "Dos attacks in available mqtt implementations: Investigating the impact on brokers and devices, and supported anti-dos protections," in *Proceedings of the 16th International Conference on Availability, Reliability*

and Security, ser. ARES '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3465481.3470049

[353] R. Light, "Mosquitto man page," Nov. 2021. [Online]. Available: https://mosquitto.org/man/mosquitto-8.html

[354] Splunk LLC, "Monitoring and logging mqtt topic messages using eclipse mosquitto," Nov. 2024. [Online]. Available: https://lantern.splunk.com/Splunk_Platform/UCE/Technology%2C_Communications%2C_and_Media/Monitoring_and_logging_MQTT_topic_messages_using_Eclipse_Mosquitto

[355] IBM, "Quality of service and connection management," Jun. 2023. [Online]. Available: https://www.ibm.com/docs/en/integration-bus/10.0?topic=bus-quality-service-connection-management

[356] Z. Wang, X. Liao, X. Zhao, K. Han, P. Tiwari, M. J. Barth, and G. Wu, "A digital twin paradigm: Vehicle-to-cloud based advanced driver assistance systems," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–6.

[357] K. Valdek, "The one true sandbox for connected car data," Feb. 2023. [Online]. Available: https://www.high-mobility.com/blog/the-one-true-sandbox-for-connected-car-data

[358] A. Finkenzeller, A. Mathur, J. Lauinger, M. Hamad, and S. Steinhorst, "Simutack - an attack simulation framework for connected and autonomous vehicles," in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, 2023, pp. 1–7.

[359] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

[360] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008, pp. 1–10.

[361] HiveMQ, "Understanding the differences between mqtt and websockets for iot,"

https://www.hivemq.com/blog/understanding-the-differences-between-mqtt-and-websockets-for-iot/, 2022, accessed: 2025-05-15.

[362] USENIX: The Advanced Computing Systems Association, "Usenix security '25 ethics guidelines," Jun. 2024. [Online]. Available: https://www.usenix.org/conference/usenixsecurity25/ethics-guidelines

[363] OASIS Cyber Threat Intelligence (CTI) Technical Committee, "Oasis cyber threat intelligence (cti) documentation," https://oasis-open.github.io/cti-documentation/, 2024, accessed: 2025-06-23.