POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Methods for Assessing the Cybersecurity Posture of Critical Infrastructure via Testbed-Based Experimentation

CHRISTOPHER NEAL
Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie informatique

Mai 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Methods for Assessing the Cybersecurity Posture of Critical Infrastructure via Testbed-Based Experimentation

présentée par **Christopher NEAL**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Nora BOULAHIA CUPPENS, présidente
José FERNANDEZ, membre et directeur de recherche
Andrea LODI, membre et codirecteur de recherche
Antoine LEMAY, membre et codirecteur de recherche
Tarek OULD-BACHIR, membre
Sylvain LEBLANC, membre externe

ACKNOWLEDGEMENTS

To begin, I would like to thank Dr. José Fernandez, Dr. Andrea Lodi, and Dr. Antoine Lemay for their guidance and support as supervisors over the course of this research.

Additionally, a number of individuals were instrumental in realizing this research as collaborators and facilitators. I would like to express my gratitude to Mehdi Taobone, Militza Jean, Dr. Ranwa Al Mallah, Dr. Khalid Laaziri, Nader Ammari, Marielba Urdaneta, Jean-Yves De Miceli, Dr. Marthe Kassoufe, Dr. David Barrerra, Dr. Hanane Dagdougui, Dr. Véronique St-Supery, Dr. Thomas Dean, Dr. Fan Zhang, David Trask, Marienna MacDonald, Matthew Daley, Karthik Thiyagarjan, Richard Doucet, Peter Adamson, and Nathan Bruno.

This research was made possible with financial and materiel support from the Laboratoire de Sécurité des Systèmes Informatiques (SecSI), the Canada Excellence Research Chair in Data Science for Real-Time Decision Making (CERC4DSM), the Hydro-Quebec Research Institute (IREQ), the Institute for Data Valorization (IVADO), and the Canadian Nuclear Laboratories (CNL) National Innovation Centre for Cyber Security (NICCS).

Lastly, I would like to thank my family and friends whom have been by my side along this journey.

RÉSUMÉ

Les infrastructures critiques (IC) désignent les actifs et les processus permettant le fonctionnement d'une société. Elles incluent divers secteurs dont la production d'énergie, les systèmes d'alimentation électrique, les réseaux de transport, le trafic aérien, l'approvisionnement alimentaire, l'approvisionnement en eau, les services de santé publique, les manufactures, les services financiers et les services de sécurité. De plus en plus, ces systèmes sont surveillés et contrôlés par des systèmes numériques connectés aux réseaux informatiques. Il existe alors une surface d'attaque potentielle grandissante qui peut être exploitée par des concurrents capables de cyberattaques. Compte tenu de l'importance des IC dans la société, la réussite d'une cyberattaque contre elles aurait un impact direct et sérieux. Récemment, le besoin grandissant de protection des IC contre les cyberattaques s'est reflété par une croissance accrue de la recherche portant sur la protection contre ces attaques.

La construction de bancs d'essai répliquant les systèmes du monde réel et simulant des cyberattaques permet l'étude de problèmes de cybersécurité ouverts liés à ces systèmes. Un banc d'essai représente donc un système réel qui lui, peut être composé d'un mélange de systèmes matériels et logiciels réels, émulés et simulés. L'utilisation de ces bancs d'essai en cybersécurité permet d'étudier les capacités de l'attaquant ainsi que la posture défensive d'un système grâce à des expériences reproductibles, sans mettre les systèmes réels en danger. Cette thèse décrit trois études de cas sur bancs d'essai portant sur la cybersécurité liés aux systèmes des centrales nucléaires (CN), aux microréseaux électriques (ME) et au contrôle du trafic aérien (CTA). Ces différents domaines représentent tous un exemple de système cyber-physique (SCP), là où un processus physique est contrôlé et surveillé par une infrastructure de calcul numérique. Cette thèse fera référence à ces systèmes en tant qu'environnement IC-SCP.

L'étude de cas de CN démontre un banc d'essai où la simulation logicielle de CN s'intègre à un système de contrôle du niveau de la chaudière nucléaire physique, qui lui est géré par deux contrôleurs programmables logiques. Une attaque du type «homme dans le milieu» est menée pour fournir des valeurs de processus falsifiées aux contrôleurs, ce qui provoque un effet malveillant dans le système. Cette étude démontre comment les processus physiques d'une CN sont sensibles aux cyberattaques.

L'étude de cas de ME présente un banc d'essai qui simule un ME résidentiel et qui est géré par une contrôleur simulé qui résout un programme linéaire d'entiers mixtes. Des attaques simulées modifient alors les données d'état fournies et les actions de contrôle déterminées par

le contrôleur. Les agents d'apprentissage par renforcement sont entrainés à déterminer les données d'état falsifiées idéales à transmettre au contrôleur afin qu'il effectue une action sous-optimale. Ces attaques sont présentées comme une méthode de découverte de vulnérabilités dans les algorithmes de contrôle et peuvent généralement être réutilisées contre d'autres algorithmes de contrôle industriel.

L'étude de cas de CTA illustre un banc d'essai simulé qui génère des rapports de position du trafic aérien à partir d'installations radars et d'antennes. Les attaques simulées génèrent des messages de type «Automatic Dependent Surveillance-Broadcast» (ADS-B) falsifiés, créant des avions falsifiés sur l'écran de CTA. Ce travail décrit une solution défensive basée sur une ontologie qui utilise des requêtes de type «SPARQL Protocol and RDF Query Language» (SPARQL) afin d'identifier des messages ADS-B anormaux qui vont à l'encontre des contraintes de l'aviation.

Bien que les résultats de ces expériences constituent des contributions uniques, le thème principal de ce travail analyse les connaissances acquises à partir des études de cas pour parfaire notre compréhension des caractéristiques de conception des bancs d'essai et des méthodologies de développement. Un processus exploitant une nouvelle taxonomie pour permettre de classifier les bancs d'essai pour la cybersécurité IC-SCP est proposé. Ce processus se résume par les étapes suivantes: l'identification des exigences nécessaires de l'expérimentation pour répondre à une question de recherche spécifique, l'utilisation de la taxonomie pour développer un plan du banc d'essai répondant aux besoins des expérimentations et finalement, le développement d'un banc d'essai pour répondre à l'artefact de conception de manière itérative.

Ce travail représente un pas vers la sécurisation des environnements IC-SCP. Alors que le paysage technique évolue et que de nouvelles menaces émergent, la recherche sur ce sujet se poursuit. Les bancs d'essai développés ici seront réutilisés par les chercheurs travaillant sur ces projets. Pour ceux qui ne peuvent avoir accès aux bancs d'essaies, cette recherche fournit un aperçu des techniques de développement de bancs d'essai et des meilleures techniques. Bien que ce travail utilise des bancs d'essai pour effectuer une analyse soit offensive, soit défensive, cette recherche ouvre la porte à de travaux futurs où ces deux types d'analyse se feraient de façon concomitante lors d'exercices d'équipe rouge versus bleu, par exemple. Celles-ci pourraient être menées par des opérateurs humains ou, de façon plus intéressante, par des agents basés sur l'intelligence artificielle (IA) pouvant adapter leurs stratégies de manière automatisée. Alors que l'IA continue de proliférer, les cyber-conflits du futur seront fort probablement menés, en partie du moins, par des agents automatisés. Il est donc impératif de comprendre comment de tels conflits peuvent survenir dans les environnements IC-SCP.

ABSTRACT

Critical Infrastructure (CI) refers to the assets and processes which enable the functioning of society. This includes a number of sectors, such as, energy production, electrical power systems, transportation networks, air traffic, food supply, water supply, public health services, manufacturing, financial services, and security services, to name a few. These systems are increasingly monitored and controlled by digital systems connected to computer networks and there is an ever-increasing attack surface which can potentially be exploited by cyber-capable adversaries. As the number of cyberattacks against CI steadily increases, there is a demand for researchers to investigate how to protect CI from adversaries, since a successful attack can have serious impact on the welfare of society.

A method for addressing this pressing issue is to construct testbeds, replicating portions of CI, in order investigate open cybersecurity concerns related to real-world systems. A testbed is a representation of a real-world system that can be comprised of a mixture of actual, emulated, and simulated, hardware and software systems. Performing cybersecurity research with testbeds allows for the ability to understand attacker capabilities and the defensive posture of a system through repeatable experiments, without endangering operational systems.

This thesis describes a testbed case study for a Nuclear Power Plant (NPP), a Microgrid (MG), and an Air Traffic Control (ATC) system. Although these are different domains, each are an example of a Cyber-Physical System (CPS), where physical processes are controlled and monitored by digital-based computational infrastructure. This research refers to this setting as a CI-CPS environment. While the findings from these experiments constitute unique contributions, the primary theme of this work analyzes the insights gained from the case studies to advance our understanding of testbed design traits and development methodologies.

The NPP case study demonstrates a Hardware-in-the-Loop (HIL) testbed which integrates an NPP software simulation with a Boiler Level Control (BLC) system, controlled by a Programmable Logic Controller (PLC). A Man-in-the-Middle (MITM) attack is conducted to provide falsified process values to the PLC, causing a malicious effect in the BLC system. This study demonstrates how physical processes in an NPP are susceptible to cyberattacks, additionally, the data generated from the attack has been collected for a separate research project in collaboration with the International Atomic Energy Agency (IAEA).

The MG case study presents a simulated testbed environment of a residential MG, managed by a custom Mixed Integer Linear Programming (MILP) controller. Simulated attacks are

conducted that modify the state data provided to and the control actions determined by the controller. Additionally, Reinforcement Learning (RL) agents are trained to determine ideal falsified state data to pass to the controller, so that it performs the most sub-optimal action. These attacks are presented as part of a framework for uncovering vulnerabilities in control algorithms and can be reused generally against other industrial control algorithms.

The ATC case study illustrates a simulated testbed which generates the position reports, from radar and antenna installations, of air traffic travelling in a controlled airspace. Simulated attacks generate falsified Automatic Dependent Surveillance-Broadcast (ADS-B) messages causing Ghost Aircraft to appear on an ATC display. This work outlines an ontology-based defensive solution that uses SPARQL Protocol and RDF Query Language (SPARQL) queries to identify anomalous ADS-B messages violating aviation-related constraints. The feasibility of a cyber-physical ontology-based defensive tool is demonstrated, however, the SPARQL query process has been shown to require future efforts to increase the speed of detection.

The experiences gained from working with these testbeds has highlighted the need for a concrete testbed development methodology. A process is proposed that leverages a novel taxonomy for classifying CI-CPS cybersecurity testbeds. The process involves identifying the experiment requirements needed to investigate a particular research question, using the taxonomy to develop a blueprint of the testbed to meet the needs of the experiments, and iteratively developing the testbed to meet the design artefact. The taxonomy has been used to classify the three testbed case studies, after the fact, to demonstrate its applicability in capturing the essence of CI-CPS cybersecurity testbeds. Future work suggests to validate the framework by using it to drive the development of a new testbed.

This research has taken a step towards securing CI-CPS environments, however, as the technical and threat landscape of these systems continues to evolve, there is ongoing work to be done by researchers and practitioners. The developed testbeds will be reused by researchers continuing these projects; for those that can not access these testbeds, this research provides insights into testbed development techniques and best practices. While this work uses testbeds to either perform offensive or defensive analysis, this research calls for future work to consider experiments where these occur at the same time in Red vs. Blue team exercises. These could be conducted by human operators or, more interestingly, by Artificial Intelligent (AI)-based agents who can adapt their strategies in an automated fashion. As AI continues to proliferate, cyber-conflicts of the future will likely be carried out by automated agents to some degree and there is a need to understand how such conflicts may occur in CI-CPS environments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

AI	Artificial Intelligence
AIA	Attack Impact Analysis
ADS-B	Automatic Dependent Surveillance-Broadcast
AMI	Advanced Metering Infrastructure
API	Application Programming Interface
APT	Advanced Persistent Threat
ARP	Address Resolution Protocol
ATC	Air Traffic Control
BLC	Boiler Level Control
BPP	Bilevel Programming Problem
CI	Critical Infrastructure
CNL	Canadian Nuclear Laboratories
CPS	Cyber-Physical System
CSV	Comma-Separated-Values
CVE	Common Vulnerabilities and Exposures
DCSA	Defensive Computer Security Architecture
DDoS	Distributed Denial of Service
DDS	Data Distribution Service
DoS	Denial of Service
DER	Distributed Energy Resource
DMS	Distribution Management System
DNN	Deep Neural Network
DNO	Distribution Network Operator
DNP3	Distributed Network Protocol 3
DTA	Defensive Tools Analysis
E&T	Education and Training
EMS	Energy Management System
ESS	Energy Storage System
FDI	False Data Injection
FSD	Flight Simulation Data
GPS	Global Positioning System
HIL	Hardware-in-the-Loop
HMI	Human-Machine Interface

IAEA	International Atomic Energy Agency
ICS	Industrial Control System
ICT	Information and Communication Technology
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IP	Internet Protocol
IT	Information Technology
IVA	Infrastructure Vulnerability Analysis
LAN	Local Area Network
LSTM	Long Short-Term Memory
MAC	Media Access Control
MDP	Markov Decision Process
MG	Microgrid
MGCC	Microgrid Central Controller
MGO	Main Grid Operator
MILP	Mixed Integer Linear Programming
MITM	Man-in-the-Middle
ML	Machine Learning
NICCS	National Innovation Centre for Cyber Security
NIST	National Institute of Standards and Techonology
NPP	Nuclear Power Plant
OT	Operational Technology
OT-SOC	Operational Technology Security Operation Center
OWL	Web Ontology Language
PCAP	Packet Capture
PCC	Point of Common Coupling
PDC	Phasor Data Concentrator
PERA	Purdue Enterprise Reference Architecture
PLC	Programmable Logic Controller
PMU	Phasor Measurement Unit
PSR	Primary Surveillance Radar
PT	Penetration Testing
PWR	Pressurized Water Reactor
RDF	Resource Description Framework
RES	Renewable Energy Source
REST	Representational State Transfer

RL	Reinforcement Learning
RSS	Received Signal Strength
SCADA	Supervisory Control and Data Acquisition
SDR	Software Defined Radio
SIEM	Security Information and Event Management
SOC	State of Charge
SPARQL	SPARQL Protocol and RDF Query Language
SSR	Secondary Surveillance Radar
TCP/IP	Transmission Control Protocol and Internet Protocol
URI	Uniform Resource Identifier
USB	Universal Serial Bus
VATSIM	Virtual Air Traffic Simulation Network
VFD	Variable Frequency Drive
W3C	World Wide Web Consortium
WAN	Wide Area Network

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

Critical Infrastructure (CI) refers the assets and processes that are instrumental in ensuring the operation of a society [1]. This encompasses a broad range of sectors, such as, energy production, electrical power systems, transportation networks, air traffic, food supply, water supply, public health services, manufacturing, financial services, and security services, to name a few [2]. These systems increasingly utilize digital components to raise productivity and reduce costs, however, they are interconnected across computer networks and can potentially be compromised by adversarial actors [3, 4]. There is thus a need to examine cybersecurity concerns related to these systems, since an attack can have grave implications on the welfare of populations and the environment.

The research presented in this work strives to address this issue through testbed-based cybersecurity experimentation related to three CI case studies and provides conclusions from these efforts to aid in the advancing of the security of CI as a whole. These case studies examine security issues related to Nuclear Power Plants (NPPs), Microgrids (MGs), and Air Traffic Control (ATC) systems. These are all examples of Cyber-Physical Systems (CPSs), where physical processes are controlled, monitored, coordinated, and integrated through the aid of computing infrastructure [5]. This work has an emphasis on cybersecurity for CPS-based CI, which this work refers to as a CI-CPS environment. Consequently, this work does not consider cybersecurity issues related to CI environments where there are no physical processes, such as banking institutions or medical record systems.

A key component of this research is the use of testbeds to perform cybersecurity experimentation. A testbed is a replication of an environment, whether it be simulated, emulated, physically represented, or some mixture of these [6]. Since it is impractical to perform cybersecurity experimentation on actual operational systems, where there can be serious real-world consequences, researchers use testbeds to represent some subset of the reality of an environment to perform experiments and draw conclusions. This work specifically uses testbeds to represent large-scale physical processes, along with the data that is generated by their supporting digital-based computing infrastructure.

Through analyzing the insights gained from developing testbeds and conducting experiments in the NPP, MG, and ATC case studies, this work addresses the primary intended contribution of this research. Namely, this work proposes a CI-CPS cybersecurity testbed design methodolgy that leverages a novel taxonomy called POILA. The name of the taxonomy is derived from the first letters of the five main classes of information that is represented. These

classes are the following; Physical Process Architecture, Operational Technology Architecture, Information Technology Architecture, Logistics Architecture, and Activity Classification. These classes capture the essence of a CI-CPS environment's hierarchies of computational infrastructure required to manage a physical process and it is proposed that these are depicted in a cybersecurity testbed. The testbed design methodology suggests to first determine an open cybersecurity research problem in a particular CI-CPS environment, define what requirements are needed from a testbed to perform the necessary experiments to answer the question, use the POILA taxonomy to construct a design of the intended testbed, construct the testbed to meet the POILA design, and, finally, perform the experiments.

The demand for such a process became evident while performing the individual NPP, MG, and ATC testbed case studies. In all three cases a large amount of time was spent building the testbeds without a firm grasp of what the finished state would look like and what the specific experiments would be. Ultimately, this limited the total amount of time that was actually spent performing experiments. This proposed process can be reused by future researchers to ensure that testbeds are developed with the most efficacy to meet their intended goals and so that they can more profoundly address their intended particular open cybersecurity challenges. As impending cyberthreats continue to mount, improving the testbed development process is an important challenge so that prosperous experiments can be conducted with wise expenditures of limited time and resources.

To get to this point, this introductory chapter motivates the need for such research by highlighting the growing threats facing CI-CPS environments, demonstrates how advancing the process behind testbed-driven experimentation is invaluable in combating these challenges, and lays out the roadmap for the remainder of this work. This chapter is organized as follows; Section 1.1 outlines the state of the CI-CPS cybersecurity landscape, Section 1.2 introduces in more detail how testbeds can be used for cybersecurity experimentation, Section 1.3 provides the problem definition and the overall goals for this work, and Section 1.4 details the organization of this dissertation.

1.1 Critical Infrastructure Cyberthreat Landscape

CI-CPS environments are undergoing a transtion from the use of legacy analog and electromechanical-based equipment to more modern Information and Communication Technology (ICT) infrastructure, resulting in an increased coupling between cyber-based digital technologies with physical equipment [3, 7]. Whether through the construction of new, primarily, all-digital builds or the upgrade of facilities through technological eras, the Opera-

tional Technology (OT) networks within an CI-CPS are increasingly taking the shape of and being integrated with traditional enterprise Information Technology (IT) networks.

OT refers to the hardware, software, and communications protocols used for the monitoring as well as controlling of industrial processes [8]. In general, there are process values reported by sensors and control actions are invoked to drive the process towards some ideal setpoints. OT systems comprise components such as Programmable Logic Controllers (PLCs), Supervisory Control and Data Acquisition (SCADA) systems, and communications protocols such as Modbus or the Distributed Network Protocol 3 (DNP3). Contrarily, IT traditionally refers to data-centric computing, such as data storage (e.g. databases), operating systems, enterprise networks, software applications, personal computers, and physical servers, to name a few.

There is a heightened entanglement of computational philosophies within CI-CPS environments, introducing a number of cybersecurity challenges as capable adversaries have an ever increasing assortment of attack vectors through ICT infrastructure to harm the reliable operation of physical processes [9]. This section provides a subset of notable CI-CPS cyberincidents, describes several motivations for attacks, and outlines some future trends in CI-CPS cybersecurity.

1.1.1 Notable Events

The events described here are a subset of notable events in the history of cyberattacks against CI-CPS environments, often resulting in some impact to physical systems. In many of these cases, the groups which have perpetuated the attack are not officially known. This work does not attempt to attribute the origin of such attacks, instead the point is to highlight the occurrence of such events.

In 2000, a former disgruntled employee of a water treatment facility in Maroochy Shire, Australia, released 800,000 litres of raw sewage by maliciously controlling SCADA systems and suppressing alarm systems [10,11]. The attacks occurred over a period of several months, resulting in significant environmental damages to marine life, local waterways, and producing an unbearable smell to the local residents. The attacks were eventually discovered when suspicious computer equipment was discovered through a routine traffic violation stop after an attack had occurred.

In 2003, the Slammer Worm (also known as Sapphire) spread across the Internet, exploiting a known buffer-overflow vulnerability in Micorsoft's SQL Server to infect hosts, notably the Davis-Besses NPP in Ohio, United States [12, 13]. Slammer is often considered the fastest computer worm in history, as it impacted approximately 90% of vulnerable hosts within 10

minutes and affected many sectors. Although a patch was provided by Microsoft six months prior to the attack, hosts were infected because they did not install the updates. The attack did not directly target the NPP, however, the worm managed to infiltrate the plant through a misconfigured firewall and disabled safety monitoring systems for several hours. Fortunately, there was no physical impact since the plant was shut down at the time for maintenance.

In 2007, the Idaho National Laboratory demonstrated how a cyberattack could damage physical components of the electrical grid [14, 15]. The attack exploited a vulnerability to open and close a circuit breaker in a diesel generator, out of phase with the connected power system, causing the generator to eventually explode. Although the attack was a controlled experiment, it showed that much of the legacy equipment and protocols operating the electrical grid are vulnerable to being attacked.

The Baku-Tbilsi-Ceyhan oil pipeline in Turkey exploded in 2008, due to a what was initially reported to be a cyberattack, causing 30,000 barrels of oil to be exposed into the environment above a water aquifer [7, 16, 17]. The attackers were presumed to have gained access to the control systems through vulnerabilities in wireless video-surveillance and safety systems. The attack allegedly involved suppressing alarms, deleting video surveillance footage, and compromising control equipment to increase the pressure in the pipeline. Operators only became aware of the explosion 40 minutes after the incident, perhaps due to extensive obfuscation tactics on the part of the attackers, when security personnel eventually witnessed flames. In the years following the event, there has been dispute as to what extent malicious cyber-based activities led to the explosion [18], however, this uncertainty over the attribution of the explosion puts into question the resiliency of such systems against well-orchestrated cyberattacks.

The Stuxnet Worm, discovered in 2010, is considered as an important shift in cybersecurity and geopolitics, as it is the first publicly known cyberwarfare weapon [7, 17, 19, 20]. Unlike previous attacks which maliciously accessed and controlled systems, this was a malware that was designed to attack specifically targeted equipment. Stuxnet infected several industrial sites in various countries, but the most notable was the Natanz uranium-enrichment facility in Iran. The malware was introduced to the facility by an unauthorized Universal Serial Bus (USB) drive and propagated through the network via unpatched vulnerabilities. The ultimate effect of the attack severely damaged centrifuges by causing them to spin slightly faster at regular intervals, while masquerading the attack by displaying normal operations on SCADA monitoring systems.

In 2014, an alleged military exercise caused aircraft to temporarily vanish from controller terminals in Austria, Germany, the Czech Republic, and Slovakia on at least two occasions [21].

Although this was not attributed to a cyberattack, it puts into question the resiliency of ATC systems to targeted attacks.

A German steel mill was attacked in 2014 preventing a blast furnace from shutting down and causing significant physical damages [17]. The attackers gained access to the plant's office network through a social engineering and spearphising campaign, then traversed into the production network, and ultimately inflicted malicious control over physical processes.

The Ukrainian power grid has been directly attacked on two separate occasions resulting in the loss of electrical power [14, 17, 19, 22]. In 2015, three separate electrical distribution companies were targeted which resulted in 30 distribution substations being shut off and approximately 230,000 people were without power for several hours [22]. This coordinated attack occurred over several phases. Login credentials were stolen through spearphising emails equipped with BlackEnergy malware, SCADA systems were infiltrated and remotely switched off, and file systems were destroyed using KillDisk malware [22]. This attack was significant since it was the first publicly acknowledged attack to result in power outage. In 2016, a malware, known as CrashOverride, switched off power provided by a transmission substation to one fifth of the population of Kiev. The discovery of CrashOverride is significant since it is considered to be a malware not tailor-made for a specific target, but is a general purpose weapon to be reused across a variety of industrial systems [14, 23].

In 2017, the Triton malware was discovered at a Saudi Arabian petrochemical plant [24, 25]. This was the first known instance of malware where attackers targeted safety instrumentation system, which would prevent shutting down industrial processes in the case of a catastrophe. This malware is believed to have been created with the purpose of destroying infrastructure to such a degree that there would be loss of human life. The malware was discovered due to what is considered to be an error from attackers who had infiltrated the site's systems.

In 2019, the Kudankulam NPP located in India discovered malware on its IT networks [26,27]. The malware is considered as benign from a operational safety standpoint with its intent perhaps being for reconnaissance. However, evidence suggests that the plant was specifically targeted since the generalized malware was repurposed for this particular facility [28].

In December 2020, it was reported that a major United States software firm known as SolarWinds was subject to a massive supply-chain attack that affected a large number of government agencies, private companies, and critical infrastructure installations across the globe [29–32]. The attack is alleged to have been initiated in early 2020 when hackers were able to infiltrate SolarWinds and compromise the source code of a popular IT management software system known as Orion. When SolarWinds provided Orion software updates to its clients, the software unwittingly installed a backdoor that allowed remote access of the

updated systems to the attackers. The attack is reported to have gone unnoticed for several months. The compromise to the Orion software was initially discovered by a private cybersecurity firm known as FireEye, which was also subjected to the attack. At the time of writing, much of the media narrative has focused on the degree to which the attack was used to perform espionage on government agencies, however, the full intent and extent of the attack is still being uncovered. This attack has many important ramifications for geopolitics and cybersecurity; for the sake of this dissertation, the attack highlights the extent to which critical infrastructure is vulnerable to motivated and dedicated attackers.

A timeline of the presented events is provided in Figure 1.1. The figure highlights for each confirmed event; a commonly-used identification, the affected CI-CPS sector, location, and year.

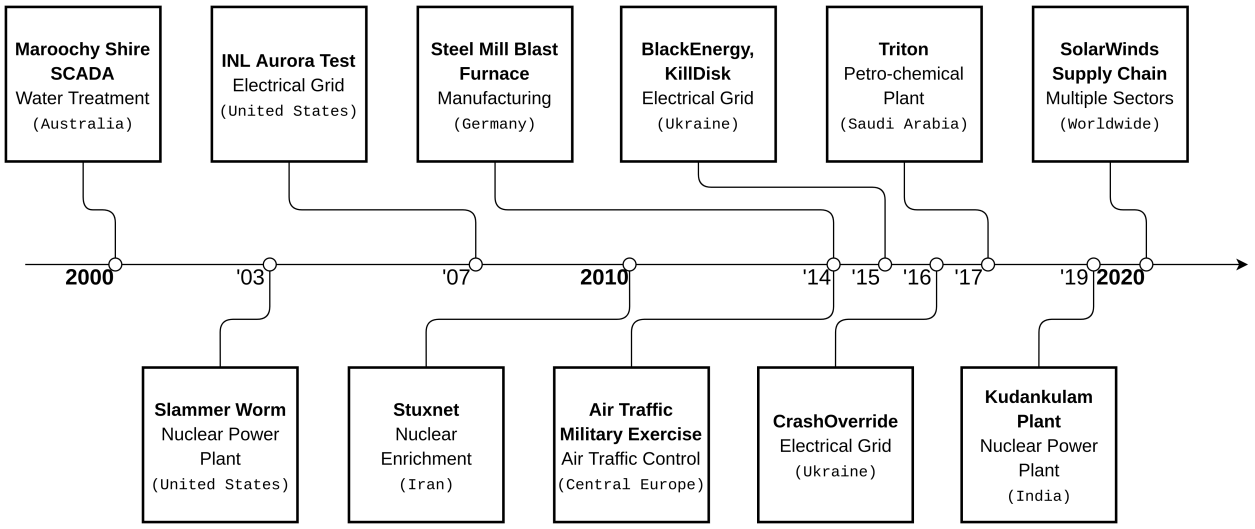

Figure 1.1 Timeline of Notable ICS-CPS Cybersecurity Events

The attacks presented here are just a subset of attacks that have been occurring against CI and it does not appear to be slowing down. A survey conducted in 2015 by the Organization of American States and the security firm Trend Micro reported that from a sample of 500 CI operators in North and South America, 53% have reported increased number of attacks, 54% have had attackers attempt to manipulate equipment, 40% have had attackers attempt to shut down their systems, 44% have had attackers attempt to delete or destroy data, and 76% of respondents noted an increase in the sophistication of cyberattacks. [33–35]. In anticipation of these burgeoning threats, researchers have a role to play in staying abreast of attacker capabilities and proposing innovative defensive countermeasures.

1.1.2 Attacker Motivations

A number of motivations exist for a party to execute a cyberattack against CI, such as the forwarding of political agendas, criminal financial incentives, or individuals seeking notoriety [36, 37]. Understanding the socio-political motivations for potential attacks can help provide insight into anticipating the goals of attackers and potential attack vectors.

The majority of the attacks presented in Section 1.1.1 have been attributed to state-sponsored groups engaging in cyberwarfare tactics. Rival nations provide a legitimate threat to the reliable operation of CI, whether to disrupt the operation of a society by compromising its infrastructure or as a demonstration of superiority. State-sponsored groups are often well funded, well trainted, and have the freedom to perform prolonged campaigns. Such an organization is often refered to as an Advanced Persistent Threat (APT) [38]. APTs may be part of the military of a nation, receive funding from governments, or simply be given the authority to operate within a political jurisdiction. Government supported groups are considered as the greatest cyber-threat to CI since they have the time and resources to execute protracted campaigns without fear of being pursued criminally or requiring immediate financial returns.

As technologies evolve, threats continue emerge from groups that may not have traditionally had the capabilities or motivations to perform cyberattacks against CI and cannot be overlooked. Terrorists may target CI to serve their ideological agenda, since there is the opportunity to inflict widespread damage. Hacktivists, hackers who employ cyber-based techniques to promote an agenda for societal change, may aim to disrupt the CI which conflicts with their philosophical beliefs. For example, groups advocating for decarbonization may target infrastructure related to the oil industry. In competitive industrial markets, such as much of the electrical grid in the United States, companies may resort to malicious cyber campaigns to gain an advantage over their competitors. It is conceivable for companies to engage in corporate espionage and potentially even to go as far as corrupting the infrastructure of competitors. Criminal financial incentives exist for well-organized cybergangs or individuals. A group may employ ransomware to hold portions of CI hostage in return for financial compensation. Such campaigns occurred in October 2020, in Montreal, Canada, disrupting healthcare and public transportation services [39, 40]. A less disruptive attack to operations, but which could have a large economic impact, may be performed by individuals wishing to modify sensor readings to reduce their utility bill, such as electricity or water usage. Also, individuals may perform a cyberattack to CI to exhibit civil disobedience, gain notoriety, or simply to gain satisfaction from overcoming the challenge involved in compromising a system.

A significant threat actor to all CI are individuals who are employed at a CI site, since they have intimate knowledge and access to mission critical equipment. These are considered

as *insider threats* and their motivations for performing an attack can vary. Attacks can be carried out by disgruntled employees and employees can also potentially be coerced (i.e. blackmailed) to perform some malicious action. There is even the possibility for nefarious agents to gain employment at a facility they wish to target.

1.1.3 Future Trends

The commitment from APT groups to target CI has been demonstrated and it is likely this will continue into the foreseeable future [4, 41]. There will likely be a continuation of APTs gaining unauthorized access to industrial systems to perform malicious control, but more worrisome, is a trend towards the development of general-purpose malware that can be reused across multiple sites [42, 43]. This capability migration and extensibility of developed malware allows for attackers from varying levels of skill to pose as serious threats. Additionally, the emergence of black markets, enabled through anonymous crypto-currency transactions, facilitates the dissemination of tools and exploits, opening the door for groups not associated with any government to try their hand at attacking CI [28]. This could lead to a rise in attacks from traditionally less sophisticated groups, such as cybercriminals employing ransomware, terrorists, hacktivists, and even hobbyists.

The Industrial Internet of Things (IIoT) is rapidly changing how CI is monitored and controlled, as devices are increasingly connected over the Internet Protocol (IP) and in some cases even exposed to the Internet. This provides a heightened flexibility for operators to remotely observe systems, however, adopting IIoT principles introduces substantial cybersecurity risks [44, 45]. Devices connected to the Internet can be identified relatively easily, such as through the website Shodan[1], and managing the risk associated with the IIoT will be an ongoing challenge for defenders.

As advancements in Artificial Intelligence (AI) and Machine Learning (ML) continue at an ever-increasing pace, there is cause for concern that attackers will attempt to use these methods in augmenting the capabilities of cyberattacks against CI [46]. At this point it is speculative as to how AI-based attacks will be conducted, but there is the possibility for malicious AI-based agents to de deployed in a CI-CPS setting, where they could monitor process values in the environment, learn an optimal strategy for disrupting physical processes, and ultimately execute an attack to compromise the system [47].

[1]Shodan: `https://www.shodan.io/`

1.2 Testbed-based Cybersecurity Experimentation

Addressing the growing threat against CI from an academic viewpoint, requires a methodology to perform repeatable scientific experiments. It is not safe to perform experiments using operational CI-CPS environments, nor is it financially feasible to completely replicate these real-world systems. For example, NPPs are comprised of a reactor core, numerous control systems, and a large number of staff. Also, electrical power systems, such as MGs, comprise the generation, transmission, and distribution of electricity along with sophisticated control algorithms. Lastly, ATC systems are used to coordinate aircraft by providing voice-based commands to pilots, aided by information provided from radar installations. A reasonable approach to facilitate cybersecurity experimentation in such environments is through the use of testbeds, where a subset real systems are replicated via physical devices, emulators, software simulations, or some combination of these [48–50].

CI-CPS settings are unique in the sense that there is some physical process that is being controlled and monitored by ICT infrastructure. Consequently, a CI-CPS testbed will generally be comprised of a representation of the physical process integrated with ICT infrastructure, thus simulating to some degree the operations CI-CPS environment. This setup enables the ability to witness how attacks performed against the ICT infrastructure impact the operation of physical processes. The goal for researchers is to appropriately represent some subset of the reality of the system where there is an open problem and attempt to resolve the problem. The findings can be then be used to propose solutions for increasing the overall cybersecurity posture of the system.

Testbeds have been accepted by the scientific research community as invaluable tools in performing cybersecurity research, enabling the ability to experiment with defensive tools, witness the impact of attacks, perform vulnerability testing, and can also be used for training purposes. A main challenge for researchers in developing a testbed, is representing a real-system to the appropriate fidelity to meet the intended research goals. Creating a testbed can require a great deal of domain-specific knowledge which may not be available to researchers, therefore there is difficulty in ensuring that a testbed accurately portrays its real-world counterpart. On the other hand, a testbed will never fully represent its real-world counterpart and there is no need to replicate portions of the real-world system which falls outside the of scope of the intended experiments. Hence, there is a need for researchers to understand to what level of realism a CI-CPS environment should be represented with a testbed, considering time and resource constraints, in order to develop meaningful experiments.

1.3 Problem Definition

The operation of CI is undergoing a period of dramatic change as digital ICT systems are increasingly used in daily operations to yield greater situational awareness, optimize activities, and reduce waste. The benefits from this digital transformation are so well accepted, that it occurs despite the growing risks it inherently incurs to cyberthreats. Critical physical processes now have digital-based controls and sensors, which can be exploited by malicious actors who have an ever-expanding arsenal of capabilities at their disposal. In an increasingly connected world with competing ideologies, foreign policies, and economies, motivations exist for malicious adversaries to attack CI through cyber-based means, while the frequency of such events is likely to increase. The result is an ever-increasing attack surface available for threat actors to target and there is a demand for researchers to continually assess the defensive posture of CI against the looming cyberthreats of the future.

Developing CI cybersecurity testbeds for investigating offensive and defensive capabilities is an established activity for assessing CI cybersecurity issues, however, there is a shortcoming of proven methodologies for proficiently developing testbeds to meet the needs of intended cybersecurity experiments. In attempting to solve open CI cybersecurity questions, researchers can be tasked with developing testbeds without a firm grasp of the requirements needed to conduct impactful experiments. This can result in the haphazard construction of testbeds and execution of experiments. Although, this approach may be able to produce some results, there is a demand to streamline this process in order to optimize the allocation of time and resources. Establishing testbed development and experiment guidelines can help researchers gain an edge over the emerging capabilities of adversaries. To arrive at this outcome, this work demonstrates three testbed case studies which investigate open cybersecurity problems in the domains of NPPs, MGs, and ATC systems. These case studies demonstrate testbed implementation details and conducted experiments that examine current cybersecurity challenges facing these respective domains. This work proposes the following problem definitions for the case studies:

- **NPP Case Study:** NPPs are large industrial installations which employ a complex array of IT/OT networks to support the reliable operation of physical processes. Due to the criticality of these systems, it is imperative to understand what types of actions attackers may perform to disrupt these processes and what impact these actions may have on physical processes. Through replicating portions of the digital and physical infrastructure of an NPP, we can create an environment to perform attacks in order

to collect datasets which demonstrate the actions attackers may execute to disrupt the operation of a physical process.

- **MG Case Study:** The economical operation of MGs requires a controller algorithm to receive accurate information describing the state of its environment and the reliable execution of the control commands it determines to input into the MG. However, as the threat of cyberattacks againts MG controllers continues to rise, it cannot be guaranteed that the sensor values passed to and control commands issued from an MG controller have not been tampered. By exploring how an MG reacts when its controller is subjugated to malicious input, we can gain an understanding of its behaviour in the face of attacks, as a precursor to determining appropriate defensive countermeasures.
- **ATC Case Study:** ATC systems are undergoing a period of digital modernization to more precisely identify the location of aircraft through the use of the Automatic Dependent Surveillance-Broadcast (ADS-B) protocol. However, there is the possibility for attackers to send falsified ADS-B messages to position reporting equipment causing the ATC displays of human personnel to show incorrect information, thus ushering a demand for defensive systems which can detect these falsified ADS-B messages. Ontological expert systems are a proposed solution for detecting these anomalous ADS-B messages and by implementing detection rules based on semantic reasoning we can gain insights into the viability of this approach.

While the findings from these experiments form part of the contributions of this work, the insights gained from these case studies are used as the basis to analyze testbed design traits and methodologies. This dissertation demonstrates the results of this analysis by proposing a taxonomy for classifying testbed characteristics and for generating CI-CPS cybersecurity testbed design specifications.

This dissertation thus formulates the following problem statement:

> *Assessing the defensive posture of CI against emerging cyberthreats requires rigorous experimentation of attacker capabilities and defender countermeasures, enabled through adroitly-defined testbed environments. However, there is an absence of established frameworks for mapping cybersecurity experiment requirements to testbed design characteristics. Through analyzing case studies of experiments conducted with CI cybersecurity testbeds, we can strengthen our understanding of how cybersecurity experiment requirements drive the development of testbeds and propose a methodology for capturing this interplay, to ultimately advance the potency of testbed-based experimentation addressing open CI cybersecurity challenges.*

1.4 Thesis Organization

This dissertation presents a summary of our efforts in advancing the comprehension of what testbed features are befitting particular CI-CPS cybersecurity research questions. To arrive at this understanding, this work presents three testbed case studies, each as their own respective chapters. In a subsequent chapter, the insights gained from these case studies are used to formulate a taxonomy for classifying the characteristics of CI-CPS cybersecurity testbeds, to be used to generate testbed design specifications from experiment requirements. The chapters are outlined as follows:

- *Chapter 2 provides an overview of the relevant literature to this dissertation.* This chapter provides the necessary background concepts of CI-CPS cybersecurity required for interpreting this work and a literature review of the current state of the art in CI-CPS testbed-based cybersecurity experimentation relevant to this dissertation.

- *Chapter 3 describes how a cyberattack conducted over an OT network can affect physical processes in an NPP.* An NPP houses a number of interdependent physical processes which are operated through a myriad of control and monitoring systems supported by ICT infrastructure, organized into different zones. Before considering detection approaches, it is necessary to understand what types of actions an attacker will take and how physical effects can be disrupted from a cyberattack. This chapter demonstrates an attack which disrupts a physical process and collects the data generated from the attack for future research.

- *Chapter 4 demonstrates a framework for analyzing the effects of attacks against MG control algorithm.* An MG is an electrical power system which can generate and store its own electricity, to help meet its demand. In many cases, MGs are connected to the the main electrical grid and can purchase electricity when there is a shortfall in locally produced electricity. A control algorithm is used within an MG to determine how to allocate generated electricity amongst storage devices and its load demand, as well as when to purchase electricity from the main grid. However, such a control algorithm will act unexpectedly when its input and output data is tampered. Therefore, it is in the interest of an MG operator to understand what effects malicious data will have on their control algorithm. This chapter outlines a framework for analyzing the effects of attacks against MG control algorithms and for establishing an understanding how control algorithms are vulnerable to malicious input.

- *Chapter 5 details a defensive solution for detecting falsified aircraft position reports in ATC systems.* As aircraft travel through a controlled airspace, their position is determined through a variety of analog and digital based technologies through the aid of radar and antennas. There is the possibility for attackers to falsify digital-based position reports, causing falsified aircraft to appear on the visual displays of ATC personnel, and compromising the ability for ATC personnel to safely guide aircraft. This chapter demonstrates a rule-based defensive application, which leverages the physical properties of aircraft, to detect falsified position report messages.

- *Chapter 6 illustrates a taxonomy for classifying CI-CPS testbeds and for generating testbed design criteria from experiment requirements.* Testbeds are developed using a variety of techniques and systems to serve their research goal, however, there are no specific procedures for proficiently undertaking this task. It is beneficial to have a method for providing a high level overview of testbeds, in order to compare its components and capabilities to other other testbeds. Additionally, it is desirable to understand what testbed design elements would best serve particular experiments. Through analyzing the insights gained from the testbeds developed in the three highlighted domains and in reviewing the relevant literature, this chapter presents a taxonomy that can be used to satisfy the aforementioned needs.

- *Chapter 7 concludes this dissertation.* This chapter provides a summary of the completed work, a description of the limitations of this research, and outlines avenues for future work.

CHAPTER 2 LITERATURE REVIEW AND BACKGROUND INFORMATION

As the cyberthreats facing CI continue to evolve, there is need for researchers to understand the capabilities of attackers and, in response, develop innovative defensive practices. This dissertation proposes to work towards this goal by modelling CI systems with testbeds to enable the execution of novel experiments and ultimately gain insights into combating these emerging threats. The process of testbed-based cybsecurity experimentation for CI has been endorsed within the scientific community and this chapter serves to identify where there is space to offer contributions. This chapter provides an overview of CI cybersecurity concepts, to equip the reader with the necessary background knowledge for the rest of this work, and a review of previous works, to orientate this work within the ecosystem of prior scholarship.

This chapter is organized as follows; Section 2.1 introduces CI cybersecurity concepts, Section 2.2 presents a review of testbed design characteristics, Section 2.3 reports on other testbeds relevant to those presented in this work, Section 2.4 provides an overview of works which model adversarial objectives in cybersecurity, and Section 2.5 is a conclusion to the chapter.

2.1 Critical Infrastructure Cybersecurity Concepts

This section explains a number of cybersecurity concepts and how they relate to CI, particularly CPS environments, which this work refers to as a CI-CPS environment. While there is some overlap between traditional cybersecurity (referred to as IT cybersecurity) and that of CI-CPS cybersecurity, there are also some fundamental differences. In traditional IT cybersecurity, threats interfere with the control-loop of business process which occur over relatively longer intervals (i.e. days, weeks, months). Contrarily, CI-CPS cybersecurity threats interfere with the control loop of physical processes, operating over relatively shorter intervals (i.e. seconds, minutes, hours), which are generally economically costly to shut-down.

Another distinction must be made about different types of CPS systems. A class of systems within the CPS umbrella term, are Industrial Control Systems (ICSs). An ICS is a system which controls some industrial process, such as energy production, manufacturing, and water treatment. For this work, NPPs and MGs, are examples of an ICS, while ATC is not an example of an ICS. Nonetheless, the cybersecurity concepts presented in this section generally apply to all three domains.

2.1.1 Layered Architecture

Purdue Enterprise Reference Architecture

A central theme to this work is that devices within ICSs, and CI-CPSs in general, are organized into a hierarchy of layers based on their function. A popular model for this, originating in manufacturing, is the Purdue Enterprise Reference Architecture (PERA) model, which is utilized in multiple industry standards (e.g. ISA-99) [51, 52].

PERA has gone through several iterations; in its current format systems are organized into six different levels spread across 3 zones [53]:

- Enterprise Zone
  - Level 5: Enterprise Network
  - Level 4: Site Business and Logistics
- Industrial Demilitarized Zone
- Manufacturing (Industrial) Zone
  - Level 3: Site Operations
  - Level 2: Area Supervisory Control
  - Level 1: Basic Control
  - Level 0: The Process

This framework serves as a blueprint for an organization to separate its business network from its ICS network. The Enterprise Zone comprises IT systems which support the logistics planning of the organization. The Industrial Demilitarized Zone provides a boundary between the Enterprise and Manufacturing Zone. The Manufacturing Zone (also referred to as the Industrial Zone) is where the physical processes reside, along with their supporting OT and IT infrastructure. To provide more insight into the functions provided by the different levels, some example systems and their timing cycles are provided in Table 2.1 (reproduced from [52]).

The systems at the lowest level are comprised of completely OT components, as the levels increase there is a transition from more OT-based systems to IT-based systems, and the highest level is completely comprised of IT components. This overview of the PERA model is meant to highlight the various degrees of control within a CI-CPS and provide the intuition that the different layers within a CI-CPS have their own unique cybersecurity challenges.

Table 2.1 Purdue Enterprise Reference Architecture (PERA): Levels, Systems, and Timing (reproduced from [52])

Level	Example Systems	Timing
5) Enterprise Network	Archives, File Servers	Days
4) Site Business and Logistics	Enterprise Resource Planning (ERP), Finance, Messaging	Hours
3) Site Operations	Operations Management, Historians	Minutes to hours
2) Area Supervisory Control	Supervisory Controls	Seconds to minutes
1) Basic Control	Safety Instrumented Systems	Milliseconds to seconds
0) The Process	Input and Output (I/O) from Sensors	Continuous

IT/OT Convergence

The OT systems which control CI-CPSs were historically considered to be secure from cyberattacks, since they operated on isolated networks with no outside connectivity. This concept is referred to as the *isolation assumption* and allowed OT systems to focus on performance, while disregarding security. This assumption may have been apocryphal in the pre-Internet age and should not be considered as an adequate defense in modern CI-CPS environments. Advancements in digital technologies have prompted CI-CPS operators to increasingly incorporate IT based technologies and more open network archtitectures into their OT infrastructure, particularly in the case of IIoT. This trend is often referred to as *IT/OT Convergence* [8, 54]. This convergence of technologies provides operators a heightened sense of awareness and control into their processes, however, it also increases the attack surface available to an attacker. This trend towards increased digitization and network connectivity amongst CI presents cybersecurity researchers and practitioners a growing challenge wherein to find innovative defensive solutions.

2.1.2 Key Concepts

CIA Triad

A core concept to cybersecurity is the CIA triad, which classifies threats which undermine three main classes [55]:

- **Confidentiality:** Unauthorized access of information or systems
- **Integrity:** Unauthorized modification of information or systems
- **Availability:** Unauthorized rendering of information or systems to be inaccessible

Although there is debate whether the CIA triad properly addresses all of the threats facing organizations, it does serve defenders in helping identify for what purpose they are defending particular assets and identify potential threats.

Prevention, Detection, Response

A simplified view of cybersecurity sees all defensive actions falling within three broad sets of activities [56, 57]:

- **Prevention:** Activities to prevent some sort of cyberincident
- **Detection:** Activities to detect an ongoing cyberincident
- **Response/Remediation:** Activities to perform during or after a cyberincident has been identified

Although other models exist for classifying activities more granularly, there is merit in this simplified view in aiding defenders assess their goals and align their objectives.

Zero-Days

Throughout the lifecycle of software and hardware systems, vulnerabilities are discovered and patches are made by the vendor. Due to the economics of producing software and hardware, it is virtually impossible for vendors to ensure there are no flaws in every extreme edge use-case of all of their components, despite their best efforts. Ideally, vulnerabilities are discovered by well-intentioned parties and are brought to the attention of the vendor.

However, in reality, a discovered vulnerability, which is not known to the general public, has important value to cyberattackers and is known as a *zero-day* vulnerability [58]. A zero-day attack exploits this unknown vulnerability, where there is no defense. For example, the Stuxnet attack exploited four zero-days in a Windows operating system.

The significance of zero-days has prompted organizations to create bounty programs to encourage the discovery of vulnerabilities by providing financial compensation to the parties discovering the vulnerability relative to its complexity, severity, and market-impact [59, 60].

Conversely, a black-market for zero-days has also emerged, offering more lucrative financial compensation in certain cases, allowing hackers to sell vulnerabilities to the highest bidder.

Intrusion Detection Systems

An Intrusion Detection System (IDS) is used to automate the procedure of auditing the events within a computer system or network to detect security issues and anomalies [61]. This task involves collecting and analyzing suspected data. An IDS can be classified as *Host-based* if the detection occurs in a single node of a system or it can be considered *Network-based* if the detection occurs over network traffic (physical or wireless). The National Institute of Standards and Techonology (NIST) has compiled a practical guide to IDS systems for ICSs in Special Publication 800-82 [62].

Two main paradigms exist for IDS detection techniques:

1. **Signature/Knowledge-based:** The detection system searches for pre-determined attack patterns in the data. These generally possess high detection rates and low false positives, however, they are unable to detect unforeseen (zero-day) attacks.

2. **Statistical/Behaviour-based:** The detection system searches for patterns that are out of the ordinary of established baselines. These potentially have the ability to detect attacks that have never been seen, however, they are susceptible to having a high number of false positives.

2.1.3 Defensive Frameworks

Cyber Kill Chain

The Cyber Kill Chain proposed by scientists at Lockheed-Martin[1] outlines that attacks, particularly those conducted by an APT group, will occur over multiple phases of escalating steps [63]. A defending organization should strive to understand how these phases would play out in their environment and use the kill-chain framework to iteratively develop defences. The Cyber Kill Chain phases are [64]:

1. **Reconnaissance:** Identification of targets and goals

2. **Weaponization:** Preparation of the operation

3. **Delivery:** Launching of the operation

[1]Lockheed Martin: `https://www.lockheedmartin.com/`

4. **Exploitation:** Gaining access to the victim
5. **Installation:** Establishing a beachhead at the victim
6. **Command & Control (C2):** Controlling of the implants remotely
7. **Actions on Objectives:** Executing the actions to achieve the goals of the mission

MITRE ATT&CK

The MITRE Corporation[2] provides the ATT&CK taxonomy, outlining real-world tactics and techniques used by attackers against typical IT and OT infrastructure [65–67]. Additionally, MITRE maintains the Common Vulnerabilities and Exposures (CVE) list which is a catalogue of known cybersecurity vulnerabilities for devices and services [68]. Defending organizations can utilize ATT&CK and the CVE list to identify vulnerable attack vectors and iteratively build upon their defenses. While the MITRE ATT&CK taxonomy and CVE are invaluable resources, they have no way of anticipating zero-day attacks.

2.2 Characteristics of Critical Infrastructure Cybersecurity Testbeds

It is generally infeasible to replicate entire CI-CPS environments in order to perform cybersecurity research. As a result, researchers must determine which parts of a system they wish to represent and how they will implement these choices. This section explains some common techniques and reviews the literature which considers testbed design characteristics.

2.2.1 Co-Simulation and Co-Emulation

CI-CPS environments are characterized by a large-scale physical process that is being monitored and controlled by an assortment of digital infrastructure. It is generally too costly or hazardous to physically represent all of these processes, so testbeds are constructed with the help of simulators.

Co-simulation refers to the techniques of coupling simulator software to compose representations of real-world systems [69]. Individual simulators can be used as black-boxes which provide the inputs and outputs similar to the real-world system they represent. In a testbed, the physical process is often simulated by one or several mathematical models (i.e. simulation software). Co-simulation allows the ability to build testbed systems to the scale of their

[2]MITRE Corporation: `https://www.mitre.org/`

real-world CI-CPS counterparts, however, some realism is sacrificed since actual devices are not actually used.

The term co-simulation is generally applied when all of the systems, including the digital infrastructure, are simulated. When digital systems are emulated and virtualized, the term *co-emulation* can be applied.

2.2.2 Hardware-in-the-Loop

Hardware-in-the-Loop (HIL) proposes to reflect the operation of some specific portion of a system with actual hardware and integrating this with a mathematical representation (i.e. simulation software) of the larger system [7]. It is generally too expensive, time consuming, or dangerous to replicate entire systems, so HIL offers a compromise by enabling the ability to investigate cybersecurity issues of particular components of a real-world CI-CPS environment [70].

The advantages gained from HIL do come with some tradeoffs. There is still a considerable cost in acquiring real-world components and the scale of experiments is limited to the number of HIL devices.

2.2.3 Design Considerations

Vaugh and Morris identify ten major cybersecurity concerns that occur accross CI and ICS environments, motivating the necessity for constructing cybersecurity testbeds to address these concerns [71]. In addressing these concerns with testbed-based experimentation, the authors classify ICS testbeds into four categories:

- **Implementation-based:** Use of real ICS controllers/actuators and sensors, to control the entirety of an actual (generally scaled-down) physical process. Cybersecurity issues related to the actual devices can be investigated, however, designing such a testbed is expensive, requires very domain-specific expertise, and does not easily scale.
- **Emulation and Implementation-based:** Often characterized as HIL, where there is a combination of simulation and real-systems. Some realism is sacrificed to simulate the physical process, however, the behaviour of real-devices can be investigated. There is still a significant cost and a reduced scale.
- **Single Simulation:** The entirety of the testbed is simulated within a single simulation. These can represent much larger systems and are less expensive, however, the overall fidelity of cyberattacks is reduced.

- **Federated Simulation:** Use of a combination of high-fidelity software simulations. Each individual component of an ICS is simulated with its own specific simulation which accurately reflect its operation. There is a high degree of realism and scale without using real equipment, however, this can require considerable resources to either purchase simulators or develop them.

Frank et al. highlight that cybersecurity testbeds are generally developed for a particular scenario which is not readily understood by the general public, and there is a need to create testbeds which can readily be repurposed to educate the general population [72]. The authors present a design lifecycle for testbeds which is used to not only iteratively identify features to include into a testbed, but also the cybersecurity challenges which are meant to be conveyed to the testbed users for education or training purposes.

Cintuglu et al. survey existing Smart Grid testbeds and classify them by a number of different dimensions depending on their research focus [73]. Testbeds are classified by their targeted research area, covered smart grid domain (i.e. generation, transmission, distribution), platform type (i.e. simulator, hardware, real-time simulator, and hybrid), and communications infrastructure (i.e. protocols and network type). The testbeds specifically designed for cybersecurity research are further classified by the types of attacks that are investigated (i.e. man-in-the-middle, precision insider, rogue software, denial-of-service, ARP spoofing, eavesdropping, malformed packet, and database attack).

Kline and Schwab draw on experiences while working with two large scale testbeds in collaboration with DARPA[3], to propose four principles to guide the development of CI-CPS testbeds [74]. The outlined principles are:

- Reduce the cognitive burden on experimenters when designing and operating experiments
- Allow experimenters to encode their goals and constraints
- Provide flexibility in experimental design
- Provide multifaceted guidance to help experimenters produce high-quality experiments

Green et al. aptly refer to the process of developing CI-CPS testbeds as *ICS Testbed Tetris* and provide practical implementation guidelines for creating a testbed [6]. The work outlines the set of characteristics one should consider when defining testbed objectives, architecture, and evaluation processes (shown in Table 2.2, reproduced from [6]). The authors also propose

[3]Defense Advanced Research Projects Agency: `https://www.darpa.mil/`

that testbeds should be comprised of a series of layers, analogous to PERA, and outline how their model can be applied. The layers are organized as follows; Management Layer, User Layer, Infrastructure Bridge, Experimental Layer, and Remote Access Layer.

Table 2.2 Testbed Characteristics (TBO: Testbed Objective, TBA: Testbed Architecture, TBE: Testbed Evaluation) (reproduced from [6])

Characteristic	**TBO**	**TBA**	**TBE**
Fidelity		✓	
Modularity	✓	✓	
Diversity		✓	
Interoperability		✓	
Monitoring and Logging	✓	✓	
Openness	✓	✓	
Scalability/Extensibility		✓	
Flexibility/Adaptability	✓	✓	
Repeatability/Reproducibility	✓	✓	✓
Measurability and Measurement Accuracy		✓	✓
Cost-effectiveness	✓	✓	✓
Isolation/Safe Execution	✓	✓	
Usability	✓	✓	
Complexity		✓	

Yamin et al. survey articles describing previous work on cybersecurity testbeds across a wide range of fields, including CI-CPS environments, and demonstrate statistics for a number of trends [49]. They conclude that the Scenario and Learning Objective are key components to a testbed. The authors use the results of the survey to provide a taxonomy for classifying cybersecurity testbeds. The main classes from the taxonomy include; Scenario, Monitoring, Learning, Management, Teaming, and Environment.

2.3 Critical Infrastructure Cybersecurity Testbeds

This work presents cybersecurity experiments which have been conducted using NPP, MG, and ATC cybersecurity testbeds. This section demonstrates a representative overview of previous work conducted using testbeds in these respective disciplines. Note that this section focuses solely on a review of testbeds, a more in-depth review of the cybersecurity challenges facing each domain reside within their respective chapter.

2.3.1 Nuclear Power Plant Testbeds

An et al. have developed a testbed for investigating cybersecurity issues related to instrument and control systems in NPPs [75, 76]. The testbed features a reactor simulator built with LabVIEW, which is integrated with a commercially available triple modular redundant PLC. Attack scenarios inject falsified signals into sensors to assess the reliability, tolerance, and reliability of the control systems when faced with the malicious input. A fault analyzer has been developed to provide operators with information regarding the attacks.

Kim et al. propose a design for a Digital Plant Protection System and Plant Monitoring Annunciator System to be incorporated into an NPP testbed [77]. These proposed systems leverage NPP-specific operational practices and data gathered from PLCs to prevent malware-based cyberattacks.

Lee et al. outline a testbed, which is under development at the time of writing, utilizing an NPP simulation developed with MATLAB/Simulink, known as *Asherah*, which has been developed in collaboration with the International Atomic Energy Agency (IAEA) [78, 79]. The software simulation will be integrated with PLCs, Human-Machine Interfaces (HMIs), and engineering workstations, typical to an NPP. The testbed will be used to validate the Asherah simulation and explore the effects which various cyberattack scenarios have against NPP digital infrastructure.

Zhang et al. experiment with data-driven cyberattack detection methods for ICSs, which has been developed using an NPP testbed [80, 81]. The testbed consists of an NPP simulation built in LabVIEW of a two-loop nuclear thermal-hydraulic system which is integrated with an industry grade SCADA software over a Local Area Network (LAN). Datasets are generated by conducting Man-in-the-Middle (MITM), Denial of Service (DoS), data exfiltration, data tampering, and False Data Injection (FDI) attacks over the network. Network-based attacks are detected using ML-based supervised learning methods, which include K-Nearest Neighbour, Decision Trees, Bagging, and Random Forests. An unsupervised learning ML method, known as Auto-Associative Kernel Regression, is used detect process values which fall outside of expected bounds. The same research group has expanded their testbed to use the Asherah MATLAB/Simulink model, with physical PLCs, in a HIL architecture, to perform similar data-driven cyberattack detection experiements [82].

2.3.2 Microgrid Testbeds

Nelson et al. present a cyber-physical testbed which simulates the operation of the IEEE 13-node reference system MG architecture [83]. The MG was developed in MATLAB/Simulink,

then ported to OPAL-RT to perform HIL simulations. The physical components are integrated with a simulated communications network developed using OMNeT++ software. Experiments analyze how simulated cyberattacks which increase network delay affect the MG's ability to maintain a target grid import power band.

Poudell et al. developed a cyber-physical testbed which integrates an OPAL-RT simulation of the WSCC 9 bus reference system with SEL 351S relay protection systems to simulate the Remote Terminal Units in substations [84]. The authors analyze the effect of attacks which render two transmission lines inoperable has on Optimal Power Flow calculations.

The FUSE testbed is used to experiment with cyberattacks carried out against interconnected MGs which coordinate resources in a decentralized manner [85]. The testbed proposes to use readings from Phasor Measurement Units (PMUs) and Phasor Data Concentrators (PDCs) to detect the presence of cyberattacks.

Cintuglu et al. describe a cyber-physical testbed which is compliant to the IEC 61850 standard [86]. The authors use the testbed to validate a proposed multiagent hierarchical MG control approach. Future work with the testbed intends to investigate how the multiagent control scheme is affected by cyberattacks.

Ashok et al. developed a multi-level testbed which simulates a campus-level MG (simulated with GridLAB-D) connected to a transmission-level power system (simulated in OPAL-RT) [87]. The campus simulation includes a virtualized computer network and an attack scenario is presented where an attacker follows steps in the Cyber Kill Chain to pivot from the enterprise network onto the OT network to impact the larger power system.

Sarker et al. investigate the effect of cyberattacks against the IEEE 2030.5 protocol used within MGs [88]. The physical architecture of a MG is simulated using OpenDSS software and its communications network is simulated using Mininet software.

2.3.3 Air Traffic Control Testbeds

The ADS-B protocol is a digital-based protocol which is being adopted by the aviation industry to provide more accurate position reports to ATC services and is more cost effective than traditional radar-based technologies. The underlying principal of ADS-B is that aircraft broadcast ADS-B messages to ADS-B antennas by means of a device called a transponder. However, there is no encryption or authentication in ADS-B messages and they have been shown to be prone to being spoofed by a capable attacker [89–91]. In response to this, researchers have been investigating ways to defend ATC systems by means of updating the ADS-B protocol and detecting falsified ADS-B messages. By developing ATC testbeds which

simulate air traffic and the transmission of ADS-B messages, researchers are able to visualize attacks on ATC displays and develop defensive countermeasures.

Barreto et al. have developed a testbed which simulates the air traffic, ATC systems, and the transmission of ADS-B messages for the Campos Basin region of Brazil in a 3D environment [92]. This region accounts of 80% of Brazil's oil production and its security is of strategic importance. Conducting simulated ADS-B attacks with this testbed is used to develop an impact assessment framework to aid in the development of civilian and military response missions.

Amin et al. model the impact which spoofed ADS-B messages can have on the airspace of the United States over the Gulf of Mexico [93]. The authors developed a Signal Simulation, Airspace Throughput Capacity Simulation, and Collision Simulation to numerically demonstrate the potential aircraft collision rate incurred from spoofed ADS-B messages. Within their Signal Simulation they implement a design alternative to the current ADS-B implementation to reduce the collision rate.

Monteiro et al. have created a testbed which integrates the Kinetic ATC simulation software, the CORE network emulation platform, and a custom ADS-B emulation system [94]. Spoofed ADS-B messages are detected by verifying the location claims of received ADS-B messages through wide area multilateration. The detection approach is shown to be capable, however, it requires at least 5 ADS-B receivers to perform the multilateration process.

Schmitt et al. use an ATC simulator software called TrafficSim, developed by DLR, to witness the impact of cyberattack scenarios which compromise the availability of flight plans [95]. Although, not directly an ADS-B spoofing attack, flight plan information is a part of the information exchanged over ADS-B communications. The authors demonstrate the time delays and excess usage of fuel which can be accrued when flight plans are maliciously modified or removed altogether for scenarios located in Munich, Germany.

2.4 Modeling Adversarial Objectives in Cybersecurity

The actions taken by rational attacking and defending actors in a cybersecurity setting are taken to maximize the outcome of their respective goals. Also, each actor will likely take actions to minimize the effectiveness of their opponent. A realistic assumption is that this conflict can occur over an undetermined number of iterations where each actor will adjust their strategies to gain an advantage. This adversarial setting has led to some researchers to model this type of scenario with principles from bilevel optimization, multi-objective optimization, and game theory.

2.4.1 Bilevel Optimization

Colson et al. present the general formulation of a bilevel optimization problem, or Bilevel Programming Problem (BPP) [96]. The general pessimistic BPP is presented in Equation (2.1):

$$\min_{x \in X, y} \quad F(x, y) \tag{2.1a}$$

$$\text{s.t.} \quad G(x, y) \leq 0 \tag{2.1b}$$

$$y \in \arg\min_{\hat{y}} \quad f(x, \hat{y}) \tag{2.1c}$$

$$\text{s.t.} \quad g(x, \hat{y}) \leq 0. \tag{2.1d}$$

There are two sets of variables; the upper-level variables $x \in \mathbb{R}^{n_1}$ and the lower-level variables $y \in \mathbb{R}^{n_2}$. The set of functions $F : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$ are the upper-level objective functions, while $f : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$ are the lower level functions. Lastly, the upper-level constraint functions are $G : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{m_1}$ and the lower-level constraint functions are $g : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{m_2}$. Note that when the variables exist in both the upper-level and the lower-level the problem becomes significantly harder to solve, however, it can be used to model some more interesting phenomena.

BPPs are hard to solve to optimality since they are generally non-convex, non-differentiable, and have been shown to be at least NP-hard [96, 97]. Due to the difficulty of solving these problems with conventional methods, the notion of *embedded algorithms* has been developed [96]. The idea is, after a model has been developed to represent the problem scenario, to re-formulate the lower-level program in such a way that can generate lower bounded (or in some cases, upper bounded [98]) solutions. This bound is improved upon over several iterations until an optimum solution is found.

Kalashnikov et al. [99] provide an overview of bilevel programming theory, its history, and show how it can be applied to certain application domains. The article explains that a BPP can be considered as a *leader-follower* problem, where two adversarial actors take actions in a turn-based manner in such a way that actions taken by the *leader* restrict the actions which the *follower* can take. Therefore, solving a BPP from the viewpoint of the *leader*, involves selecting an optimal action which anticipates the follower's optimal action, given the *leader*'s action. This type of problem can be applied to cybersecurity problems in the electrical grid, for example, a defender (*leader*) may want to select which vulnerable components should be given additional security measures, assuming certain attacker (*follower*) capabilities. An example of a BPP formulated as a security problem is provided in Appendix A.

Shahidehpour et al. present a BPP formulation for coordinated cyber-physical attacks on the electrical grid, where the attacker uses a cyberattack to hide the effect of physically disabling transmission lines [100]. Liu et al. considers a similar cyber-physical attack against transmission lines and formulates it as a trilevel optimization problem [101]. Both works propose the use of load redistribution attacks to mislead the state estimation process, which has received formal mathematical justifications by Yuan et al. [102].

2.4.2 Multi-objective Optimization (Co-optimization)

To effectively model particular scenarios it can be necessary to consider multiple objectives, which may even be conflicting in nature. Finding co-optimized solutions to such situations must consider the inherent trade-offs in the conflicting objectives. Co-optimization can occur in situations such as goods production, where the operations of a factory aim maximize product quality while minimizing cost, or in a computer network which wants to maximize security will minimizing the time of servicing requests [103]. It can also be used to model adversarial settings such as defender-attacker games where the defender is trying to minimize its operating costs and an attacker is trying to maximize the defender's costs. The general formulation to the multi-objective optimization problem (from [104]) is provided in Equation (2.2):

$$\min \quad \{f_1(x), f_2(x), ..., f_k(x)\} \tag{2.2a}$$

$$\text{s.t.} \quad x \in S. \tag{2.2b}$$

Where there are $k \geq 2$ objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$. The vector of objective functions is denoted $f(x) = (f_1(x), f_2(x), ..., f_k(x))^T$. The decision variables are held in the vectors $x = (x_1, x_2, ..., x_n)^T$ and reside within the feasible region $S \in \mathbb{R}^n$. Here the constraint functions are formulated generally within the space S. If an objective function, f_i, is to be maximized, it is equivalent to minimizing $-f_i$.

For nontrivial problems there will be no solution that simultaneously optimizes all objective functions, thus the objectives are conflicting and there will be multiple Pareto optimal solutions. Due to the intractability of such problems, solutions are often found using heuristics and evolutionary approaches. Multi-objective optimization has been used in cybersecurity research to evaluate tradeoffs in the investment of additional security measures [105–107] as well as to model defender-attacker behaviors [108, 109].

2.4.3 Game Theory

When defending against cyberattacks there is a tendency for an arms-race scenario to unfold. An attacker may devise a method to circumvent some system defenses, the defender then introduces new countermeasures to prevent the attack, and the cycle continues. Game-theoretic analysis has been applied to cybersecurity research to gain insights into this dilemma, in order to understand what actions security personnel should take.

Hamilton et al. provide an overview of challenges faced when applying game theory to information warfare and cybersecurity: (1) there are limited databases of examples to work with, (2) players can perform multiple and simultaneous actions, (3) attackers have no time constraint when performing moves, (4) attackers likely have different goals than defenders, (5) the set of legal moves for each player will likely change over time, (6) the resources and goals of the players will likely change over time, and (7) it is hard to define the timing for player moves and system states [110].

Lye et al. model the intrusion of an IT network as a stochastic game with four nodes (web server, file server, work station, and the Internet) and two players (defender and attacker) [111]. The transition between network states is provided given potential defensive and offensive actions conducted by each player. The authors calculate a Nash Equilibria, (a broadly accepted solution to a game where no player will receive benefit for adjusting their strategy unilaterally) for three different types of network attacks (denial-of-service, defacing a website, and stealing confidential data). The formulation of this game is however not realistic as it considers the game to have complete information (i.e. the attacker and defender know all potential actions the opponent can take, all state transition probabilities are fixed, and the payoff functions for each player is known).

This highlights a common issue of game theory for cybersecurity research, that it is hard to model realistic and tractable attacker-defender games. A survey conducted by Roy et al. states that in reality a network defender faces a dynamic game where there is incomplete and imperfect information about attacker threats [112]. The authors explain that they haven't found any work that models this problem in a realistic way for IT network cyberdefense.

To limit the explosion of possible state-space representation of cybersecurity problems, Roy et al. suggest modeling problems as a partially observable stochastic game and propose the use of an Attack Countermeasure Tree [113,114]. The idea is to map all possible attack scenarios to countermeasure strategies reduced to an optimization problem. This approach is however limited as it only considers static attack scenarios with some predefined countermeasures.

The use of game-theoretic approaches has also been studied for cybersecurity of electric grids. Backhaus et al. model the behavior of an intruder and operator on a reduced electrical grid (one substation, one load, and one generator), where the operator takes an action in response to the probabilistic occurrence of an intruder, given the system's state estimation calculation [115]. Li et al. formulate a game scenario where an attacker must choose when to perform a jamming (DoS) attack against a sensor and when the sensor must choose to send information for system-wide state estimation. They show that the optimal strategies for both sides comprise a Nash equilibrium in this zero-sum game.

2.5 Conclusion

This chapter has outlined the field of CI cybersecurity, reviewed previous work which has investigated the use testbed-based CI-CPS experimentation in this field, and provided a review of methods for modelling adversarial objectives. The chapter serves to align the works presented in Chapters 3-6 within the field of CI-CPS cybersecurity and to initiate the discussion of where there is room to make novel contributions.

The NPP testbeds presented in Section 2.3.1 use a limited amount of equipment to represent their ICS environment, there is thus room to advance the design of NPP testbeds by utilizing a larger assortment of NPP OT and IT systems. The NPP testbed presented in Chapter 3 is larger in detail and complexity, than those presented, while also utilizing the Asherah simulation model. At the time of writing, the Asherah model has not been made available to the entire scientific community, however, it is actively being used by groups which collaborate with the IAEA and serves as a legitimate simulation of the complexities of the physical processes in an NPP.

The MG testbeds presented in Section 2.3.2 demonstrate some common design characteristics, namely, a software simulation is used to represent the physical dynamics of the MG and is connected to either actual or simulated control devices and network communications infrastructure. The use of a performant (and expensive) physical simulator, such as OPAL-RT, is often used to represent the dynamics of an electrical power system to a high level of fidelity, and, using actual control equipment, protocols, or communications networks, allows the ability to witness how those components are affected by a cyberattack. On the other hand, the MG testbed in Chapter 4 is purely a simulation and does not attempt to reflect an MG to a high degree of realism, where there is already a substantial amount of previous work. Instead, Chapter 4 presents a simplified MG developed with MATLAB/Simulink, where there is a focus on formalizing attacks in mathematical notation and demonstrating a framework for finding vulnerabilities in MG control algorithms.

The ATC testbeds shown in Section 2.3.3 are built specifically to answer particular cybersecurity research scenarios and are not readily available for reuse. Nonetheless, the ATC testbed presented in Chapter 5 reuses some of the themes of the presented testbeds, particularly, integrating the physical dynamics of an ATC simulator software application with a digital communications network. The ATC testbed presented in Chapter 5 has been custom built in order to build a defensive ADS-B anomaly detection expert system which utilizes knowledge described in ontologies to infer the presence of attacks.

The testbed classifications and taxonomies presented in Section 2.2.3 demonstrate a number of valid criteria for designing testbeds, however, they lack the ability to capture the interplay between IT and OT in a layered architecture. The taxonomy for classifying CI cybersecurity testbeds presented in Chapter 6 has been designed to demonstrate the unique properties of representations of IT and OT devices, when used to control a physical process in a testbed. Additionally, the previous work provide descriptions of testbed characteristics, however, there is no formalized method for generating requirements for testbeds based off of the intended experiments, as in the taxonomy presented in Chapter 6.

CHAPTER 3 SIMULATING ATTACKS AGAINST A BOILER LEVEL CONTROL SYSTEM IN A NUCLEAR POWER PLANT

Digital technologies are increasingly being used within NPP facilities in the replacement of original analog systems, as older plants are upgraded, and as part of largely all-digital designs in new builds. This brings tremendous advantages to operating organizations, as it increases the availability of data for decision making purposes and can decrease operating costs through reducing inefficiencies. However, the increasing level of digitization also increases the attack surface that can be exploited by cyber-capable adversaries. In the preparation against NPP cyberattacks, testbeds can be used to replicate portions of the control infrastructure of an NPP to understand attack vectors available to adversaries as well as to experiment with the development of defensive tools and architectures. This chapter presents an NPP testbed and demonstrates an attack which compromises the operation of a physical process, namely the Boiler Level Control (BLC) system. The presented testbed is part of an ongoing project and the contributions made by this research are highlighted in the chapter.

This chapter is organized as follows; Section 3.1 presents the cybersecurity challenges facing NPPs, Section 3.2 demonstrates a HIL testbed which simulates the physical process of a BLC system in a Pressurized Water Reactor (PWR) NPP, Section 3.3 shows a cyberattack scenario which has been conducted using the testbed, and Section 3.4 concludes this chapter by summarizing the completed work and describing the insights gained from this case study.

This chapter is adapted from work presented at the 2020 IAEA International Conference on Nuclear Security (ICONS 2020) [116] and conducted at the Canadian Nuclear Laboratories (CNL) National Innovation Centre for Cyber Security (NICCS).

3.1 Nuclear Power Plant Processes and Cybersecurity Concepts

NPP organizations increasingly depend upon digital-based technologies for the reliable operation of their facilities through a myriad of IT and OT systems. This enables the automation of industrial processes and heightens the exchange of information, while also providing the means for business units to conduct their day-to-day tasks. This has numerous advantages; however, it also increases the attack surface which can be potentially be exploited [8].

Computer networks, whether physically connected or wireless, are the core of all digital communications and processes within an NPP. These networks relay sensor values and control commands amongst digital devices as well provide a means to provide information to

human operators. To protect these computer networks, the IAEA recommends a Defensive Computer Security Architecture (DCSA) known as the Defence-in-Depth Architecture [117]. In this paradigm, computer networks and control systems are strategically deployed through layers of securely defined levels and zones. The intention is to shield mission critical NPP processes from a cyber-intrusion by restricting the communications between the different levels and zones. This architecture provides the backbone for NPP network security, however, as attacker capabilities continue to evolve, there is ongoing demand to find new innovations in protecting NPP computer network infrastructure. A successful cyberattack against an operating NPP could result in a loss of intellectual property as well as significant operational losses as the facility owners would need to demonstrate to their national regulator that the incident has been fully resolved and that the plant is safe to restart and begin normal operations. In addition, a successful cyberattack on an NPP could potentially undermine the public's confidence in these facilities.

3.1.1 Process Control in a Pressurized Water Reactor Nuclear Power Plant

A PWR NPP is an example of an ICS, where multiple physical processes are controlled and monitored [118]. This has historically been conducted using analog devices, however, the trend is towards digitization. To provide context, an overview of these processes is provided in Figure 3.1 (inspired by [119]).

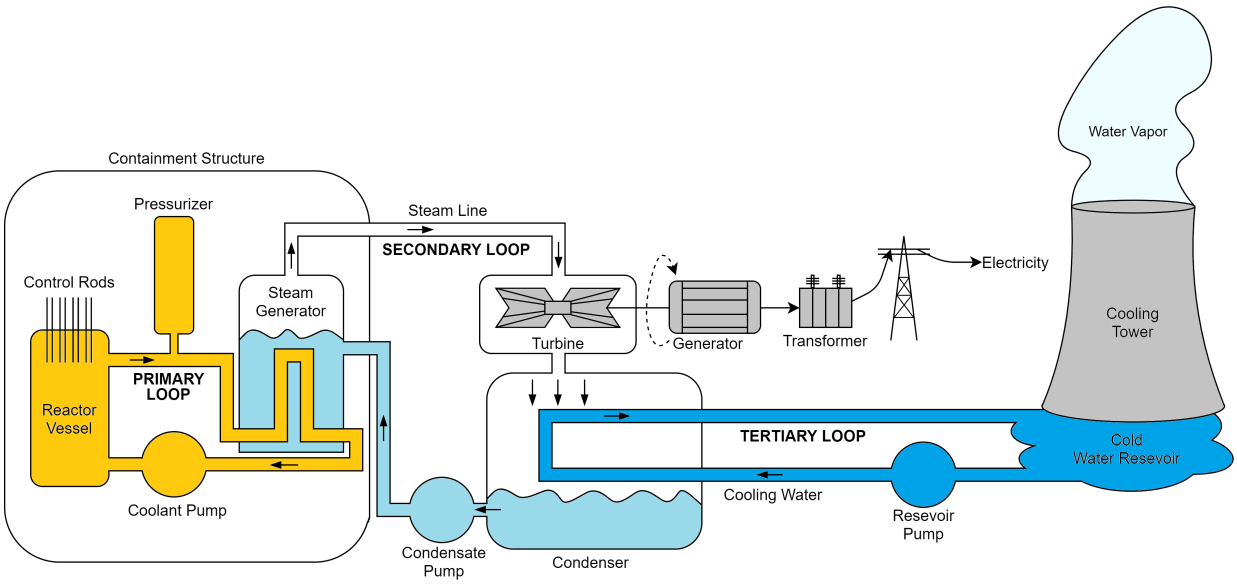

Figure 3.1 Overview of a Pressurized Water Reactor (inspired by [119])

Primary Loop: The process of fission occurs in the reactor vessel through the splitting of uranium atoms and is used to heat up water. Control rods are composed of chemical elements that prevent the rate of fission and are dropped into the reactor vessel in shutdown situations. The pressurizer keeps the heated water under pressure to prevent the water from boiling. The heated water in the primary loop is ultimately piped to the steam generator boiler to be used as a source of heat. The water is pumped back into the reactor vessel to be reused in the process.

Secondary Loop: The steam generator consists of a large boiler that is filled with water. The water in the steam generator evaporates due to the contact from the heated water from the primary loop. The evaporated steam travels through the steam line and is used to drive a turbine. The driving of the turbine is used to create electricity at the generator. The generated electricity's voltage is regulated at a transformer and is sent to the electrical grid. The steam that drives the turbine is cooled into water in the condenser and is pumped to the steam generator to be reused.

Tertiary Loop: Water from a large reservoir is pumped through the condenser to cool the steam passing through the secondary loop. Water vapor is released through a cooling tower.

These processes are increasingly automated using a variety of digital OT systems to invoke control over physical devices, such as switches and valves. OT refers to the hardware, software, and communications protocols used for the monitoring as well as controlling of industrial processes. In general, *sensors* report process values and *actuators* invoke actions to drive the physical processes towards some ideal setpoints. OT systems comprise components such as PLC devices, SCADA systems, and communications protocols such as Modbus or DNP3.

Traditionally, OT systems operated on isolated computer networks and cybersecurity was not a concern. With this perceived air gap, OT systems were not designed with security mechanisms (e.g. authentication, cryptography) and the focus was on performance. However, with advancements in computational capabilities coupled with the increased complexity of ICS processes, OT systems have been increasingly integrated with traditional IT systems to support operations. There is now increased concern for protecting mission critical NPP processes from the possibility of cyberthreats [120, 121].

3.1.2 Defence-in-Depth Architecture

The Defence-in-Depth architecture is an approach recommended by the IAEA to partition an NPP organization's computational infrastructure in levels of criticality amongst logical divisions with varying levels of access [117]. Although not identical to PERA (presented in Section 2.1.1), the Defence-in-Depth methodology comprises an akin philosophy of layering networks based on their function. The levels range from 1 to 5, based on the severity of compromise to the overall operation of an NPP. Level 1 is the most critical with the most restrictive access, while Level 5 is the most open. An overview of these levels is provided in Table 3.1.

Table 3.1 Overview of Defence-in-Depth Levels

Level	Function	Connectivity	Access
1) Reactor Protection Systems	Reactor control and safety systems	No network connectivity	No remote access
2) Operational Control Systems	Process control and safety systems	Limited one-way connectivity from Level 2 to 3	No remote access
3) Real Time Supervision Systems	Non-safety related controllers, process reporting, and engineering workstations	Limited two-way connectivity between Level 3 and 4	Remote access on a case-by-case basis
4) Technical Data Management Systems	Work management and configuration management systems	Limited two-way connectivity between Level 4 and 5	Remote access for predefined workflows
5) Business Supporting Systems	Business management systems, e-mail servers, and public website	Connectivity to the Internet	Remote access and third-party access

To ensure these logical boundaries are adhered to, technical controls such as boundary protection devices (e.g. firewalls, data diodes) are used to control the flow of communications. Technical controls are augmented with operational controls such as restricting physical access to systems based on work function and authorization, enforcing a two-person rule when performing work in high security levels, and utilizing portable media and device management controls.

Security levels can further be segmented into zones in order to distinguish between different process functions. For example, in Level 2, the steam generation process is in a separate zone than the pressurizer and there is limited connectivity between the two processes. This section

is by no means an exhaustive description of the Defence-in-Depth architecture paradigm, rather it is meant to provide the reader with the intuition of the need for such an approach and some of the safeguards that should be in place.

3.1.3 Threat Model

For an NPP organization, the viable attack vectors will vary depending upon the overall defensive posture of the computer systems and the training of the staff. Threat actors will vary depending on socio-political factors and there are different measures of what would be considered a successful offensive cyber-operation. The most severe cyberattack would be one in which OT components in one of the inner layers of the DCSA were compromised, causing a plant shut down by one of the multiple and independent safety shut down systems. This could potentially impact the stability of the electrical grid and cause significant economic losses. Other attacks may have the goal of performing reconnaissance on an NPP to exfiltrate intellectual property, lay the grounds for a future attack, or to project some form of power.

Note that the multiple and independent Reactor Protection Systems at Level 1 are not considered in this work as a viable cyberattack target. Due to the criticality of these systems, there is no network connectivity and these systems cannot be reprogrammed or modified. Additionally, any physical access is performed in groups and is heavily monitored.

This chapter instead focuses on defending attacks aimed at compromising OT components residing in the Operational Control Systems in Level 2 of the Defence-in-Depth architecture. Some of the controlled components at this level include the steam generator, pressurizer, coolant pumps, and turbines. An attacker does not have a direct connection to Level 2 from the outside, but still has a variety of entry points at their disposal from within an organization.

An attacker can attempt to steal authentication credentials from personnel through targeted phishing emails or other social engineering strategies. Should this be successful, the attacker could log into systems and progressively gain access to security critical controls. This type of attack campaign would require a relatively weak defensive posture, where firewalls are misconfigured, passwords are reused, devices do not have recent patches, and other poor cybersecurity practices. A less direct approach would be to try and introduce malware into control system devices as they receive regular incremental upgrades by attacking the supply chain. This could involve compromising source-code developed by third-party vendors or corrupting media used to install software onto devices. The most dangerous attacker would be an insider threat, as they have access and knowledge of the systems at potentially critical

network segment levels. This type of threat actor may be driven by some personal motive, or they could be coerced, to perform some malicious actions.

Should an attacker gain access to Level 2 control systems, they may try and render devices inoperable (i.e. bricked), causing control systems to potentially fail. An attack may also try to send false information to controlling devices so that unintended situations may occur. For example, an attacker may alter the water level reported in a steam generator system to be higher than what it is in reality. This could cause regulating valves to restrict water flow and put the nuclear reactor in risk of not having a capable heat sink. An attacker would ideally mask any effects they are introducing into the system by changing the reported sensor values and present normal looking situations on display monitors[1]. The attacker's aim would be to push the system into an undesirable state, such that one of the multiple and independent safety systems force a reactor shutdown, while presenting operations staff a system operating under normal conditions.

3.2 Nuclear Power Plant Testbed

This work considers a HIL testbed which simulates the BLC process in a PWR NPP. This section outlines the basics of how BLC is used in steam generation in an PWR NPP, describes the implemented BLC HIL of the presented testbed, and provides an overview of how the BLC HIL fits into the architecture of the overall testbed.

The described testbed is located at the CNL NICCS and is part of an HIL Cyber Range that mimics the segmented computer networks typically found in an NPP facility.

3.2.1 Overview of Steam Generation and Boiler Level Control

The steam generation process in a PWR is at the intersection of the primary and secondary control loops [122]. An overview of these processes for a typical steam generator boiler in a PWR is provided in Figure 3.2 (reproduced from [123]).

Primary Loop: Heated water enters the steam generator through the primary inlet (hot leg). The water is piped through tubes which are attached to the tubesheet. As the water flows through the tubes, heat is transferred to the secondary water system surrounding the tubes. The water traverses the tubes and ultimately exits through the primary outlet (cold leg). The exiting water is returned to the reactor vessel to be reheated and complete the primary loop.

[1] This was the case with the Stuxnet attack [20].

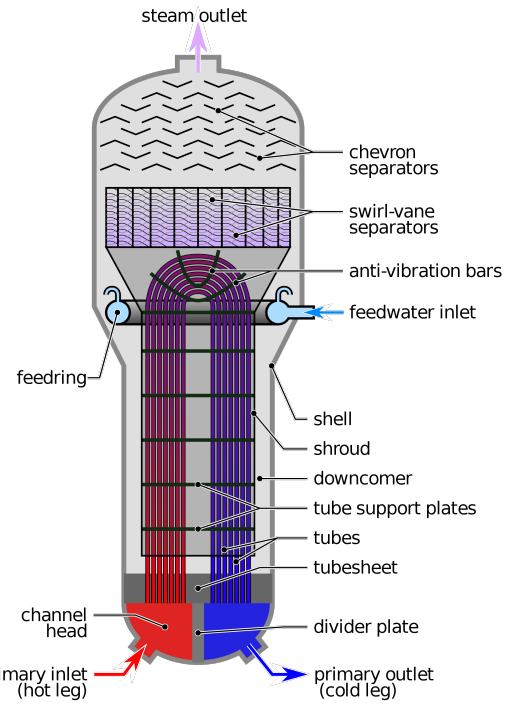

Figure 3.2 Generic Nuclear PWR Steam Generator (reproduced from [123])

Secondary Loop: The secondary water, which is used to generate steam, enters the steam generator through the feedwater inlet. This feedwater enters the outer shell encasing of the boiler through the feedring. The feedring is torus in shape and has inverted-J shaped nozzles spaced around its circumference which feed the water into the boiler. The secondary water is heated by the contact with the primary loop tubes, becomes a steam-water mixture, and rises. The steam-water mixture passes through the swirl-vane separators where centrifugal action separates the steam from the water. The steam then passes through the chevron separators to remove more water and extract high-quality steam. The high-quality steam exits the steam outlet and will ultimately turn the turbine to generate electricity. The steam will be condensed to water and will be reused as feedwater to complete the secondary loop.

Control: This work primarily considers the steam generator water control system, also known as the BLC system. The water level control system has the goal of controlling the rate of feedwater flow to ensure the desired levels in the boiler are maintained. The control system which manages this process is provided the inputs; steam flow, feedwater flow, actual water level, and setpoint water level.

3.2.2 Boiler Level Control Hardware-in-the-Loop Setup

Since it is infeasible to replicate the entirety of the OT and IT processes of an NPP to perform cybersecurity research, this work utilizes a simplified HIL implementation of a BLC. The HIL principle proposes to reflect a specific portion of a system and integrate this with a mathematical representation (i.e. simulation software) of the larger system. The effect is that the equipment that is connected with the simulation will behave as if it were operating in an actual environment.

This work utilizes a simulation of a two-loop 2,772 Megawatt (MWt) PWR which includes primary, secondary, and tertiary loops, along with their control systems. The software simulation, known as Asherah, was developed as part of a separate research project led by the IAEA and has been made available to this presented work [79, 124, 125]. The simulation software is developed in MATLAB/Simulink and determines the physical dynamics, transients, and process values of a simplified PWR. The simulation provides outputs which can be used by devices, such as PLCs, and can receive values calculated by external devices to influence the operation of the simulated NPP.

The BLC system is simulated using a reduced-scale industry standard training system which provides pressure, flow, level, and temperature controls [126]. The system prominently features two cylinders which can be filled with water and are used to replicate two steam steam generators. Note that a typical PWR NPP will have an even number of steam generators (i.e 2, 4, 6, 8, etc.), with the number depending on the plant's electrical output. Feedwater to the cylinders are provided through pumps controlled by Variable Frequency Drives (VFDs) and regulating valves. The testbed system does not generate the necessary heat to produce steam, instead outlet water leaves the cylinders by means of VFD controlled pumps and regulating valves. The feedwater and outlet water for each of the cylinders is controlled by a PLC. The PLCs utilize information from the system from sensors which provide the water level, feedwater flow rate, feedwater pump speed, feedwater valve position, outlet flow rate, outlet pump speed, and outlet valve position in order to effectuate control over the water level. This simulated BLC process for a single PLC and water cylinder is outlined in Figure 3.3.

The PLCs receive inputs such as the power being produced by the NPP and the desired boiler water level. The PLCs control the pump and valves to ensure that the water entering the cylinders are in equilibrium with the water leaving due to steam generation calculated by the simulation. Since actual steam is not generated, water is pumped out of the boiler in relation to how much steam is being produced as calculated by the simulation.

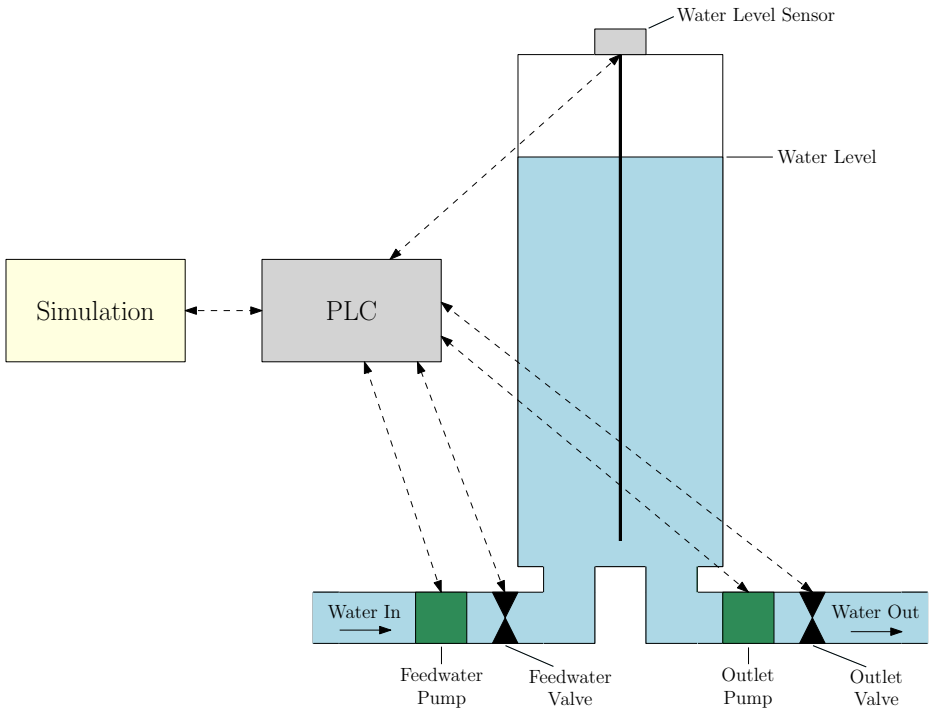

Figure 3.3 HIL setup of the scaled-down BLC system. The dynamics of the NPP are calculated in the Simulation and are passed to the PLC controlling the BLC system. The PLC receives the water level in the cylinder from the Water Level Sensor to be used in the control process and to be passed to the Simulation. The PLC issues commands to the Feedwater Pump, Feedwater Valve, Outlet Pump, and the Outlet Valve to control the water level in the cylinder. The water entering and exiting the system is housed underneath the setup in a large reservoir (not shown in the diagram).

The simulation considers the steam generation boilers to be 15 meters (m) in height and 5 m in diameter. The BLC system must maintain an acceptable level in the boiler at all times. Severe damage can be caused to the steam generator should the water level go below 10 m as the heating tubes become exposed to air, or, if the water level goes above 14 m as the chevron separators are submerged in water. In either case, the reactor core must be tripped to prevent damage to the steam generator and potentially the entire plant. Restarting the reactor core in such an event requires several days and would would result in severe economic impacts as well as damaging the public's trust in the plant. To safeguard against this scenario, a 1 m buffer is used at the bottom and top of these extremes, resulting in an operating capacity between 11 m (considered as 0% water capacity) and 13 m (considered as 100% water capacity). In the event these limits are reached, the steam generator is tripped, however, the reactor core does not need to be stopped. To further prevent an unwanted stoppage to the steam

generator, the BLC is programmed to only operate within the range of 11.5 m (25% water capacity) and 12.5 m (75% water capacity).

The water cylinders in the testbed are approximately 1 m in height. The values from the simulation are scaled so that when the cylinders are 25% in capacity, the steam generator is considered to be at 11.5 m, and, when the cylinders are 75% in capacity, the steam generator is considered to be at 12.5 m. A successful cyberattack would compromise the PLCs to adjust these water levels in an undesired manner.

3.2.3 Testbed Architecture

The HIL BLC setup described in Section 3.2.2 is a component which fits into the overall testbed which replicates Levels 2 and 3 of a PWR NPP. Note that Level 2 comprises the Operational Control Systems and houses other control processes (i.e. pressurizer, turbine, generator, etc.) in a similar fashion to the BLC system. Each process would reside in their own specific zone within the level. Data is gathered from each zone's control process and is passed to Level 3 which comprises the Real Time Supervision Systems. A representation of the logical elements which exist in each of Levels 2 and 3 of the testbed is provided in Figure 3.4.

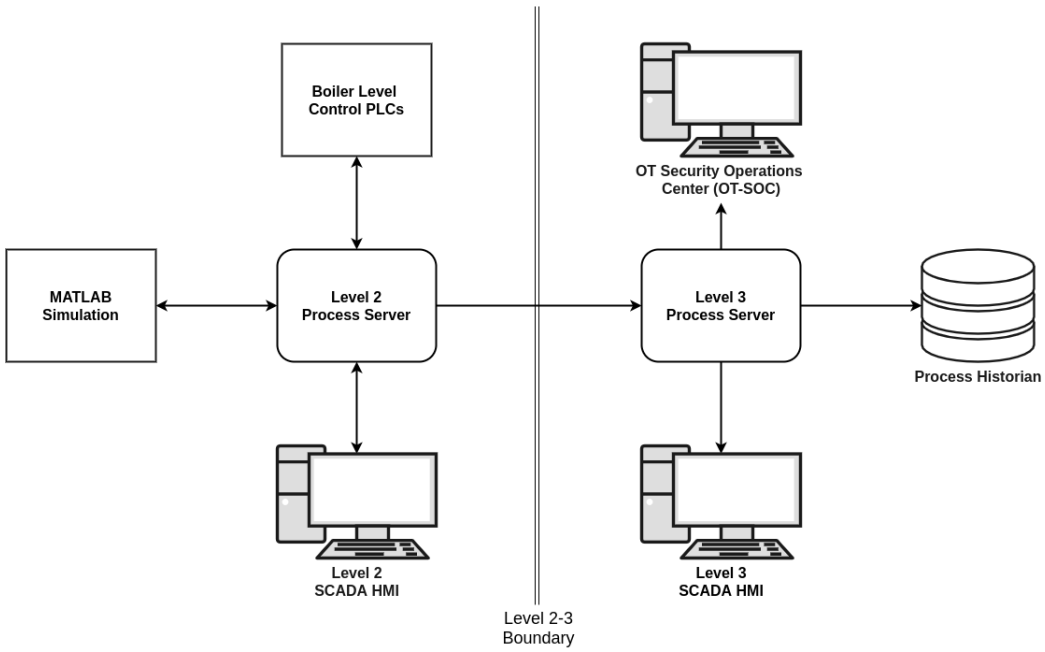

Figure 3.4 Logical Architecture of NPP Testbed

The MATLAB Simulation calculates the dynamics of the NPP and interfaces with the Level 2 Process Server. The Level 2 Process Server acts as an intermediary which passes process

values amongst the connected devices in Level 2. Additionally, the Level 2 Process Server passes the process data generated by the Level 2 devices to the Level 3 Process Server which aggregates this process data for reporting functions. The Level 2 Process Server relays data between the MATLAB Simulation to the different Level 2 zones, such as the Boiler Level Control PLCs. Additional zones could be configured to also interact with the Level 2 Process Server to increase the realism of the testbed. The presented logical architecture assumes that all the other zones are simulated in the MATLAB Simulation. The Level 2 SCADA HMI is a workstation for a human operator to monitor the processes and provide control commands to the processes. The Level 3 SCADA HMI is a workstation for a human operator to monitor the control processes from Level 2. The Process Historian is a database which records the data from the Level 2 processes. The Operational Technology Security Operation Center (OT-SOC) receives network data and utilizes security systems to monitor for unaccounted network traffic.

Within Levels 2 and 3 are a number of networks and devices which replicate the infrastructure of a PWR NPP. An overview of the testbed's physical infrastructure is provided in Figure 3.5. Note that in order to preserve the anonymity of the real-world testbed, the architecture shown in Figure 3.5 is not an exact description of the testbed components, rather it provides an overview of its logical elements.

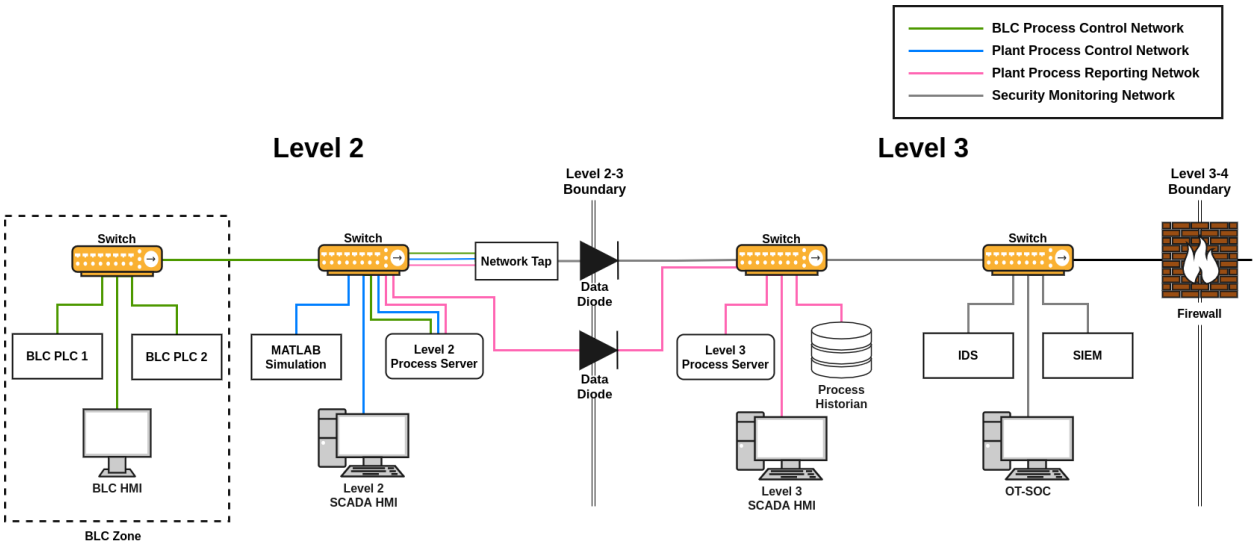

Figure 3.5 Physical Architecture of NPP Testbed

BLC Process Control Network: This network comprises the BLC zone and relays the BLC data to the Level 2 Process Server. BLC PLC 1 and 2 are the devices which control

the water level of the cylinders (described in Section 3.2.2) as part of the HIL system which simulates steam generation. The BLC HMI is a terminal where a human operator can monitor and control the processes specific to the BLC zone.

Plant Process Control Network: This network contains the MATLAB simulation, which acts as the remaining process zones, and relays the process values generated at this level to the Level 2 Process Server as well as the Level 2 SCADA HMI.

Plant Process Reporting Network: This network relays the process values from Level 2 to Level 3 for monitoring and reporting purposes. The data from in the Level 2 Process Server is sent across the Level 2-3 boundary to the Level 3 Process Server by means of a Data Diode[2]. The process data received by the Level 3 Process Server is forwarded to the Process Historian and Level 3 SCADA HMI.

Security Monitoring Network: This network captures all OT related network data from Level 2 and uses security tools to monitor for unaccounted network traffic. The Network Tap in Level 2 captures all the network traffic on the Level 2 OT networks. The captured network data is passed across the Level 2-3 Boundary with a Data Diode. In Level 3, the network traffic is analyzed by security tools (i.e IDS and Security Information and Event Management (SIEM) software) to search for anomalous events which may indicate the presence of a cyberattack. A human operator can monitor and investigate the network traffic and alerts generated by the security tools at the OT-SOC workstation.

3.3 Boiler Level Control Attack Simulation

This section demonstrates a MITM carried out against the testbed's BLC system. Performing such an exercise generates the type of network traffic which may occur during an actual attack and induces the physical effects which can occur from a successful attack. This can be used to verify the effectiveness of security tools (i.e. IDS and SIEM software) and better understand vulnerabilities in OT equipment. The ultimate goal is to equip an NPP organization with the knowledge to prevent such an attack in a real-world setting.

[2]Data-diodes are network communication devices which transfer data securely in a single direction. Unlike firewalls which can be reconfigured, data-diodes are physically constructed to only enable one-way communication [127].

3.3.1 Attack Overview

The attacker's goal is to reduce the water level in the boiler and could trigger several malicious consequences. For example, the amount of steam generated would be reduced causing less electricity to be produced than what is expected from the plant, the primary loop would risk overheating as not enough heat is dissipated through the steam, and the turbines would risk being damaged from not receiving sufficient steam.

The BLC PLC has been configured to use a value known as the *Reactor Power Setpoint* from the simulation to determine at what level the water should be in the boiler. The *Reactor Power Setpoint* value indicates at what percentage capacity the NPP intends to be operating. Generally, this value is 100%, indicating that the steam generators should be producing as much steam as possible and the BLC will maintain a water level at its programmed upper limit of 12.5m. The attacker aims to inject a falsified *Reactor Power Setpoint* value of 0%, indicating to the PLC that the NPP aims to have no electrical output and there should be as little of steam as possible produced. The result will be that the BLC PLC will reduce its water level to its lower programmed limit of 11.5m, causing a reduction in steam production.

The attacker is an insider threat (either a disgruntled employee, coerced employee, or nefarious operative who has gained employment at the facility) who has access to the BLC systems and the knowledge of its operation. The attacker connects a Kali Linux machine to the Plant Process Control Network to execute the attack. The attacker performs an Address Resolution Protocol (ARP) Spoof poisoning attack to redirect traffic between the MATLAB Simulation and the BLC PLC 1. The attacker then sends falsified values to the PLC, presenting the PLC with an incorrect version of the reality of the NPP's operations, causing the PLC to perform malicious control of the BLC system.

3.3.2 Assumptions

This work assumes that the effect of the attacker connecting the Kali Linux machine to the network is not acted upon by the defensive tools. This connection is in fact detected by the deployed IDS software and if defensive measures are properly implemented, the IDS could trigger mechanisms to prevent further actions from the attacking machine. The assumption is that no defensive actions are triggered when the Kali Linux machine connects to the network. This is a reasonable assumption, since the occurrence of unaccounted for machines connecting to the network is rare and it is difficult for defenders to put in defences which anticipate every attack, such as this one. Additionally, if IDS tools are not properly configured they are susceptible to generating a number of false positives. In this case, defensive personnel may

consider this an alert from this connection as a false alarm. Lastly, it is also possible that the attacker has performed some previous actions which prevent the IDS tool from triggering defensive actions.

Another assumption is that the BLC PLC solely uses the *Reactor Power Setpoint* value as an indicator to determine at what level the water should be in the steam generator. In an operational NPP there will likely several process values which affect the water level and the control systems should be implemented in a way so that a single compromised process value does not have such a drastic effect on the water level. Additionally, the control systems should have some checks in place to verify that the received values appear logical. It is unlikely that in an operational NPP that the *Reactor Power Setpoint* will suddenly appear to be 0%. However, for the purpose of this research of witnessing the physical effects on the testbed's BLC system, this simplification has been deemed acceptable.

A final assumption is that there is a direct connection between the MATLAB Simulation and the BLC PLC. The architecture from Figures 3.4 and 3.5 shows that all values from the MATLAB Simulation must pass through the Level 2 Process Server. However, the Level 2 Process Server uses an industrial protocol that, at the time of writing, has yet to be broken and there is no way to send falsified process values through an MITM attack. Instead, for the purpose of this work, a direct connection is made between the MATLAB Simulation and the PLC using the Modbus protocol, which is susceptible to MITM attacks. The Matlab simulation has a Modbus server which communicates directly the PLC as illustrated in Figure 3.6.

Figure 3.6 A Modbus server in the MATLAB simulation communicates process values directly with the BLC PLC

3.3.3 Attack Actions

The devices considered in this attack are the BLC PLC, the MATLAB Simulation, and the Kali Linux attacker machine. Note that the information presented in this section regarding these devices and their communication are anonymized to protect the integrity of the presented testbed. The anonymized ID, Media Access Control (MAC) Address, and IP Address for each device is shown in Table 3.2.

Table 3.2 Devices Involved in Attack Scenario

Device	ID	MAC Address	IP Address
PLC	`plc_12:34:56`	`12:34:56:12:34:56`	`10.9.99.01`
Simulation	`matlab_ab:cd:e0`	`ab:cd:e0:ab:cd:e0`	`10.9.99.02`
Kali Linux	`kali_88:88:88:88`	`88:88:88:88:88:88`	`10.9.99.88`

The attacker connects the Kali machine to the network and performs an ARP spoof attack to redirect traffic between the simulation's Modbus server and the PLC. The attacker then sends its own malicious Modbus packets to the PLC and returns its packets from the transactions, as if the PLC and the simulation were communicating directly. This MITM attack is outlined in Figure 3.7.

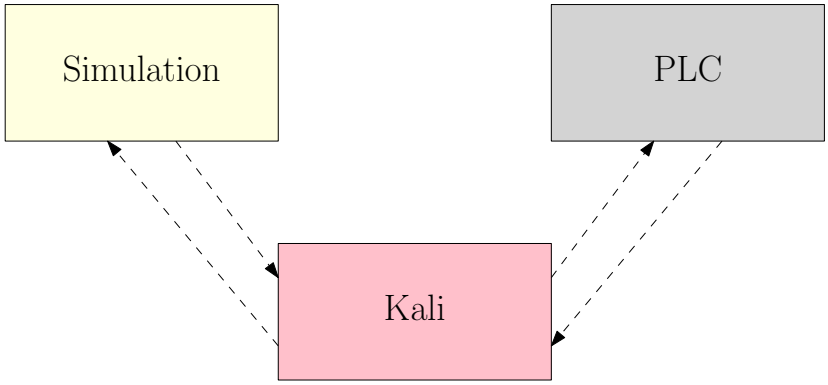

Figure 3.7 Attacker's Kali machine relays and alters communications between the Simulation and the PLC

A Packet Capture (PCAP) file has been generated during the simulated attack, using Wireshark [128]. A subset of the traffic has been anonymized and is presented in Table 3.3 to demonstrate the actions taken by the attacker to effectuate the attack. The packets are explained as follows:

Packet 1: Simulation sends value for *Reactor Power Setpoint* to PLC. This is a normal query transaction prior to the attack.

Packet 2: PLC returns a confirmation response to the Simulation. This is a normal response transaction prior to the attack.

Table 3.3 BLC Attack Packet Capture Information

No.	Source	Destination	Prot.	Info
1	10.9.99.02	10.9.99.01	Modbus/ TCP	Query: Trans: 101; Unit 1, Func: 16: Write Multiple Registers
2	10.9.99.01	10.9.99.02	Modbus/ TCP	Response: Trans: 101; Unit: 1, Func: 16: Write Multiple Registers
3	kali_88:88:88:88	Broadcast	ARP	Who has 10.9.99.01? Tell 10.9.99.88
4	kali_88:88:88:88	Broadcast	ARP	Who has 10.9.99.02? Tell 10.9.99.88
5	plc_12:34:56	kali_88:88:88:88	ARP	10.9.99.01 is at 12:34:56:12:34:56
6	matlab_ab:cd:e0	kali_88:88:88:88	ARP	10.9.99.02 is at ab:cd:e0:ab:cd:e0
7	kali_88:88:88:88	matlab_ab:cd:e0	ARP	10.9.99.01 is at 88:88:88:88:88:88
8	kali_88:88:88:88	plc_12:34:56	ARP	10.9.99.02 is at 88:88:88:88:88:88
9	10.9.99.88	10.9.99.01	ICMP	Redirect (Redirect for host)
10	10.9.99.88	10.9.99.02	ICMP	Redirect (Redirect for host)
11	10.9.99.02	10.9.99.01	Modbus/ TCP	Query: Trans: 102; Unit 1, Func: 16: Write Multiple Registers
12	10.9.99.02	10.9.99.01	TCP	[TCP Retransmission] 40000 → 200 [PSH, ACK] Seq=2296 Ack=1621 Win 64084
13	10.9.99.01	10.9.99.02	Modbus/ TCP	Response: Trans: 102; Unit 1, Func: 16 Write Multiple Registers
14	10.9.99.01	10.9.99.02	TCP	[TCP Retransmission] 200 → 40000 [PSH, ACK] Seq=1621 Ack=2313 Win 8192

Packet 3: Kali broadcasts a query to the network asking which MAC address is at the PLC's IP address.

Packet 4: Kali broadcasts a query to the network asking which MAC address is at the Simulation's IP address.

Packet 5: PLC informs Kali the MAC address it is using at its IP address.

Packet 6: Simulation informs Kali the MAC address it is using at its IP address.

Packet 7: Kali updates the Simulation's MAC address to use Kali's MAC address.

Packet 8: Kali updates the PLC's MAC address to use Kali's MAC address.

Packet 9: Kali sends a redirect to notify the PLC that the route to the destination is via Kali.

Packet 10: Kali sends a redirect to notify the Simulation that the route to the destination is via Kali.

Packet 11: Simulation sends value for *Reactor Power Setpoint* to PLC.

Packet 12: The transaction is redirected through Kali and the malicious value for *Reactor Power Setpoint* is sent from Kali to the PLC.

Packet 13: PLC returns a confirmation response to the Simulation.

Packet 14: The transaction is redirected through Kali and a falsified response to the transaction is sent from Kali to the Simulation.

3.3.4 Attack Effects

The physical effects of the attack is demonstrated in Figure 3.8. The presented process values are values recorded within the Process Historian database shown in Figures 3.4 and 3.5. The falsified value for *Reactor Power Setpoint* begins being sent from the Kali machine to the BLC PLC at the time of roughly 15:57 and triggers physical effects in the steam generation process. The effects on the testbed from the attack is explained as follows:

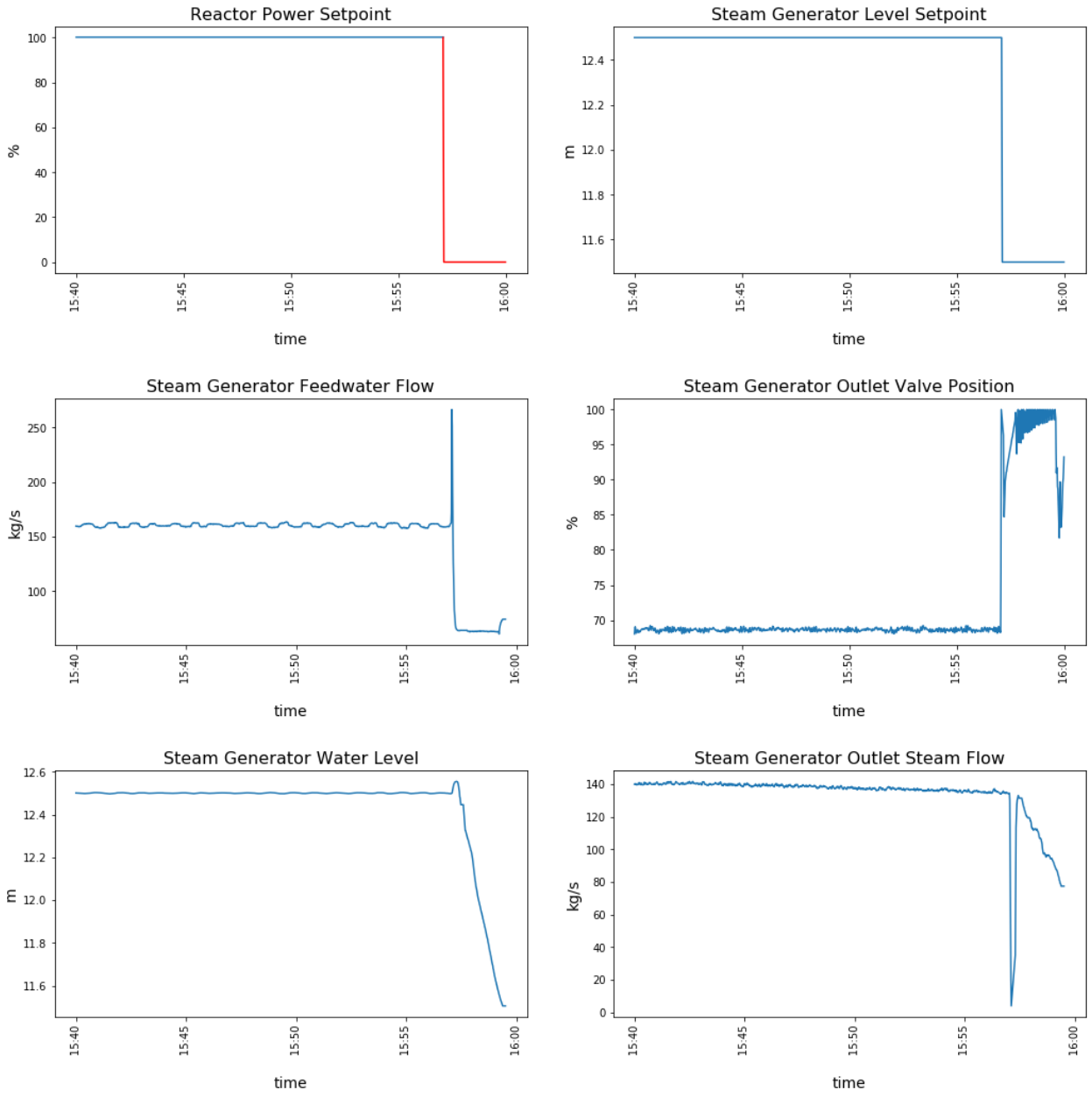

Figure 3.8 Results of the BLC MITM Attack. *Top-left:* Reactor Power Setpoint value passed received by the PLC. The falsified value of 0 is shown in red to emphasize the attacked value. *Top-right:* The water level setpoint determined by the PLC. *Middle-left:* The amount of feedwater measured entering the steam generator. *Middle-right:* The position of the outlete valve. *Bottom-left:* The water level in the steam generator. *Bottom-right:* The amount of steam generated by the steam generator.

Reactor Power Setpoint: The PLC is receiving a legitimate value for the *Reactor Power Setpoint* of 100% from the Simulation until the attack begins at 15:57. At this point the Kali

machine is sending a malicious value of 0% to the PLC. Receiving this falsified value triggers the PLC to perform physical actions and modify the normal operations of the BLC system.

Steam Generator Level Setpoint: Upon receiving the falsified value for the *Reactor Power Setpoint*, the PLC adjusts the setpoint value for the desired water level from 12.5 m to 11.5 m.

Steam Generator Feedwater Flow: The amount of feedwater into the steam generator is stable until the point where the falsified value for *Steam Generator Level Setpoint* is received. At the time of attack, there is a transient spike in feedwater, followed by a significant decrease in feedwater flow into the steam generator so that the water level is not replenished.

Steam Generator Outlet Valve Position: This value indicates at what percentage the outlet valve should be opened. Prior to the attack the valve is just below 70% open and is restricting the rate at which the water is leaving the testbed's water cylinder. After the attack, the valve opens to release more water out of the cylinder to reduce the water level. Once enough water has left the cylinder, the valve again restricts the rate of water flow out of the cylinder.

Steam Generator Water Level: Prior to the attack the BLC system is maintaining a water level of 12.5 m. At the time of attack, the water level briefly increases due to transient effects, then sharply drops towards a value of 11.5 m.

Steam Generator Outlet Steam Flow: This value is a calculated process value determined by the PLC, since the testbed does not actually produce steam. The generated steam is relatively stable until the attack occurs. At the time of attack, transient effects cause the steam generator to record nearly no steam generated. Following the transient event, the amount of steam generated reduces at a rate relative the dropping water level in the steam generator.

3.4 Conclusion

This chapter presents a HIL testbed which integrates a scaled-down version of a PWR BLC system with a simulated NPP model. To motivate this research, a description of the cyber-security concerns related to the processes in a PWR NPP are described. Safeguards to these processes are put in place using the Defence-in-Depth network architecture, however, due

to the increasing complexity of the digital control systems being used, there is still need to explore defenses to potential threats. This work describes a simulated MITM attack against the PLCs controlling the water level in the steam generation process. This work has laid the groundwork for future experimentation activities to be carried out with the described testbed.

Overcame Challenges

The infrastructure of the testbed has been developed by a number of individuals over several years and the activities being performed with this testbed is ongoing [116]. The work conducted in this research has nonetheless contributed significantly to the development of the testbed and has overcame a number of implementation challenges to realize the state the of the testbed presented in this chapter.

The components implemented in this work include; the deployment of the Level 2 and 3 Process Servers, the integration of the MATLAB/Simulink model with the Level 2 Process Server, the deployment and integration of the Level 2 and 3 SCADA HMIs, and the deployment of software to transfer process data across the Data Diode to the Level 3 Process Server. This work has also served a supporting role in deploying the Process Historian, the configuration of the BLC PLC code, and the development of the MITM cyberattack.

Additionally, this work has synthesized the capabilities and architecture of the testbed in a coherent and relatable manner. The description of the simulated attack has been made using data queried from the Process Historian along with network traffic from a PCAP file, and is presented uniquely in this work. The following artefacts are original to this work; Figure 3.3, Figure 3.4, Figure 3.5, Figure 3.6, Table 3.2, Figure 3.7, Table 3.3, and Figure 3.8.

Limitations

The MATLAB/Simulink model used in the HIL process has a very limited number of inputs and outputs available, limiting the realism of the HIL effect. The NPP simulation used is an early version and as newer versions become available the overall realism of the HIL effect will be increased.

The overall scale of the physical BLC system implemented in the testbed is greatly reduced compared to its real-world counterpart. As a result, the physical effects of the changes to water level due to a successful cyberattack do not accurately reflect a real-world BLC system, but are considered as suitable for many research activities and for conducting realistic hands-on incident response training exercises. Additionally, the MITM attack from this work was

carried out over the Modbus protocol, while there is already an understanding of Modbus vulnerabilities. The Modbus protocol is still used in some cases in some ICS environments, however, it is generally being phased out. Nonetheless, the implementation does allow for the ability to witness some simulated physical effects of a cyberattack and generates its associated network traffic.

Future Work

The overall realism of the behaviour of the testbed can be increased by deploying newer versions of the MATLAB/Simulink model as they are released. While there are some limitations to the realism of the testbed, it has utility in supporting a number of research vectors.

The testbed can be used to explore the development of defensive IDS tools. These could include rule-based detection capabilities which look for anomalous network traffic, as well as, data-driven detection techniques which attribute changes in physical processes to potential cyberincidents. There is much work previously done in data-driven analysis of ICS processes which analyze changes in physical processes to identify the degradation of equipment for predictive maintenance, however, there is still much work to be done to attribute changes to physical processes to cyberattacks.

This testbed can also be used to find vulnerabilities in devices which are deployed in real-world NPPs. For example, by using particular makes and models of PLCs within the testbed, focused investigations can be done to find out under what circumstances they can be compromised.

Lessons Learned

This chapter presents an MITM which was carried out over the Modbus protocol, however, the intention was to perform the attack against a different industrial protocol that has yet to be cracked by researchers or practitioners, at the time of writing (this other protocol is not named to preserve the anonymity of the testbed's features). Much effort was spent trying to perform an MITM attack against this protocol, but ultimately to no avail. This is reassuring from an operations perspective, but was frustrating from a research perspective since it was a precursor to effectuating an attack agains the BLC PLCs. Fortunately, the architecture was built to also effectuate communications between the simulation and the PLCs over Modbus, allowing us to perform the MITM attack. It is important to understand the goals of an experiment and one should not expect to be the first to break a system or protocol, unless that is the intended goal of the research.

The Level 2-3 architecture of the testbed was described in great lengths, however, ultimately only the architecture in Level 2 was needed for the demonstrated attack. Nonetheless, the efforts involved in developing this architecture has not gone to waste, since the Level 2-3 architecture is used in ongoing research activities and the described implementation details are deemed important in the context of this dissertation as a whole. In this case it was acceptable to incorporate these supplementary features, but researchers should be mindful of when effort is ill-spent on extraneous features.

An interesting facet of the results of the attack is the appearance of the transient effects in a number of the processes as the water level dropped. This type of behaviour would not be witnessed if the BLC system was represented with a software simulation and highlights the interdependencies between tightly coupled cyber-physical systems. Although these transients effects are unique to this experimental setup, this illustrates that successful cyberattacks can cause unexpected propagating effects across physical processes.

CHAPTER 4 ANALYZING THE RESILIENCY OF MICROGRID CONTROL ALGORITHMS AGAINST OPTIMIZED ATTACKS

In response to environmental concerns and increased strain on legacy electrical grids, there is growing interest in meeting some of our electricity needs using Renewable Energy Sources (RESs). MGs are a promising approach for this, since they incorporate Distributed Energy Resources (DERs) (e.g. photovoltaic panels, wind turbines) and Energy Storage Systems (ESSs) (e.g. batteries) to power small-scale power systems within clearly defined regions. This has several advantages, however, the adoption of MGs introduces a number of cybersecurity concerns, since the operation of an MG is generally performed by digital devices over wireless communication networks. Additionally, the coordination of MG devices is performed by a centralized controller which can act in an unanticipated manner when faced with falsified input caused by a cyberattack. Most MG cybersecurity research focuses on identifying vulnerabilities in the equipment and communication architecture within MG installations, manually designing cyberattacks to exploit these vulnerabilities, and offering detection or mitigation strategies to counter the threats. This chapter instead proposes a framework for uncovering vulnerabilities in MG control algorithms by describing a simulated MG testbed which is managed by a custom control algorithm and demonstrating how simulated attacks can be conducted to find vulnerabilities in the control algorithm.

This chapter is organized as follows; Section 4.1 introduces MG cybersecurity concepts, Section 4.2 presents a simulated MG testbed along with its control algorithm, Section 4.3 shows the effects of various attacks against the MG control algorithm, and Section 4.4 concludes this chapter by summarizing the completed work and describes the insights gained from this case study.

4.1 Microgrids and Cybersecurity Concepts

MGs are increasingly being adopted to decrease the dependency of a region or campus to the traditional electical grid. This has potential cost-saving and environmental benefits, however, it also introduces a number of cybersecurity challenges. This section outlines these issues by describing how MGs fit into the traditional electrical grid and provides an overview of MG cybersecurity concepts.

4.1.1 The Electrical Grid, Smart Grid, and Microgrids

An electrical power system is a real-time energy system constituted of a network of components which generate, transport, and supply electric power [129]. The *grid* is an example of an electrical power system which provides power over a large area.

The electrical grid is a vast system comprised of three key components; generation, transmission, and distribution (summarized in Figure 4.1, reproduced from [130]) [129, 131]. Electricity is produced at generating stations, or power plants, at voltages typically between 10 and 25 kilovolts (kV) [131]. Voltage is increased at step up transformers, then transported towards population centers over high voltage transmission lines. Large scale industrial customers (e.g. mining operations) may be fed directly from transmission lines. Electricity is reduced in voltage at substations, or step-down transformers, then rerouted. Substations which deliver electricity to end-customers are known as *distribution substations*, while substations which redirect electricity within the transmission network are known as *transmission substations*.

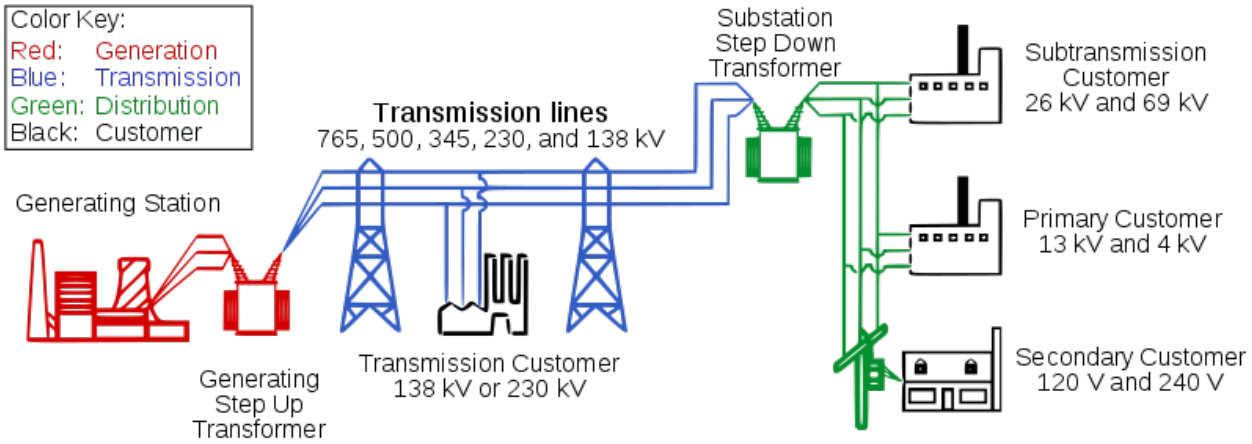

Figure 4.1 Simple Diagram of Electricity Grids in North America (reproduced from [130])

The electrical grid system can be considered as a legacy system that has been receiving incremental upgrades for over a century. It is a vertically oriented structure where a large industrial-scale producer generates and distributes electricity to consumers. The Smart Grid can be considered as the next generation of the electrical grid where there is a two-way exchange of information and electricity between providers and consumers [132]. The goal is to respond more dynamically to real-time energy demand, ultimately increasing efficiency and reducing waste. The Smart Grid will include devices such as Advanced Metering Infrastruc-

ture (AMI) (e.g. Smart Meters), RESs, ESSs, smart appliances, and synchophaser systems (e.g. PMUs, PDCs).

MGs are small-scale electrical power systems which interconnect DERs and loads within clearly defined regions. These regions are often regional distribution networks which can generate, store, and supply electricity to service local consumers through the use of RESs (e.g. photovoltaic solar panels, wind turbines) and ESSs (e.g. batteries) [133, 134]. MGs may act autonomously in island-mode or they can operate in grid-connected mode where they can exchange electricity with the main electrical grid, with the capacity to realize a transition between the two modes. MGs may operate under the Smart Grid philosophy and utilize smart technologies, but it is not always the case.

MGs are generally controlled through a hierarchy of three levels each with their own goals and security concerns [135]. Primary control acts locally on the DERs in response to transient dynamics to stabilize voltage and frequency (range of milliseconds to minutes). Secondary control is performed to optimally dispatch DER production as well as to synchronize the voltage and frequency within the microgrid (range of minutes to hours). Tertiary control coordinates the economics of the microgrid, to determine how much electricity should be produced locally at the microgrid and how much electricity should be purchased from the main grid (range of hours to days).

4.1.2 Microgrid Cybersecurity Landscape

The adoption of MGs introduces a number of cybersecurity concerns, since the operation of an MG is generally performed by digital devices over wireless networks [135]. The digital infrastructure used in an MG to relay sensory information and perform control commands can potentially be compromised due to a cyberattack from a capable adversary.

The base defense against MG cyberattacks is to follow industry norms and best practices. The National Institute of Standards and Technology (NIST) has identified the following cybersecurity vulnerability classes in the Smart Grid and MGs; 1) People, Policy, and Procedures, 2) Platform Software/Firmware Vulnerabilities, 3) Platform Vulnerabilities, and 4) Networks [136]. The Sandia National Laboratories outline several MG threat models and proposes isolating key digital infrastructure into segmented computational enclaves through a defence-in-depth approach to mitigate the effect of cyberattacks [137]. The MITRE Corporation has expanded its ATT&CK framework to consider typical attack tactics and techniques which can be used in ICSs, such as MGs [67, 138].

Despite an organization's best efforts, various attack vectors are likely to exist through vulnerable components in MG installations. Over the course of the lifetime of an MG installation there will be the opportunity for cyber-capable adversaries to compromise devices and processes. Vulnerable devices include PLCs, Smart Meters, PMUs, and PDCs [137, 139]. These components are susceptible to malware delivery, software misconfiguration, DoS, and eavesdropping over Transmission Control Protocol and Internet Protocol (TCP/IP) networks. Compromised reporting devices can be subjugated to FDI attacks, which involves sending falsified values amongst MG devices with malicious intent. This has motivated researchers to study how FDI attacks can be carried out against MGs [140–142], along with potential defensive countermeasures [143–145]. Although precautions should be taken in securing reporting devices in an MG, it is wise to consider the scenario where devices may be compromised and report falsified data to control algorithms. Experimentation with MG attacks and defenses using testbeds can be used to this end without causing any negative real-world effects [146, 147].

4.2 Microgrid Testbed

This work considers the effect of FDI attacks against a small MG through the use of a simulated testbed to demonstrate a framework for uncovering vulnerabilities in MG control algorithms. This section describes the composition of the MG that the testbed represents, the control logic used within the MG, the threat model considered against the MG, and the architecture of the testbed implementation.

4.2.1 Microgrid Architecture

This work considers a small MG consisting of a photovoltaic system (i.e. solar panel), an ESS (i.e. battery), and three households acting as fluctuating loads over time. This reference system represents a small neighborhood distribution network and is outlined in Figure 4.2. The solar panel generates electricity relative to the sun's solar irradiance throughout the day and is used to supply electricity to the households. Excess electricity generated by the solar panel can be stored in the battery to be used at a later time. Note that the MG is connected to the main electrical grid via a grid-connected transformer. Should the MG not be able to satisfy the load demand, electricity can be bought from the grid operator. Conversely, excess generated electricity can also be sold to the grid operator. The Microgrid Central Controller (MGCC) receives sensory information about the given state of the MG and determines control actions to satisfy the electrical load while minimizing total cost.

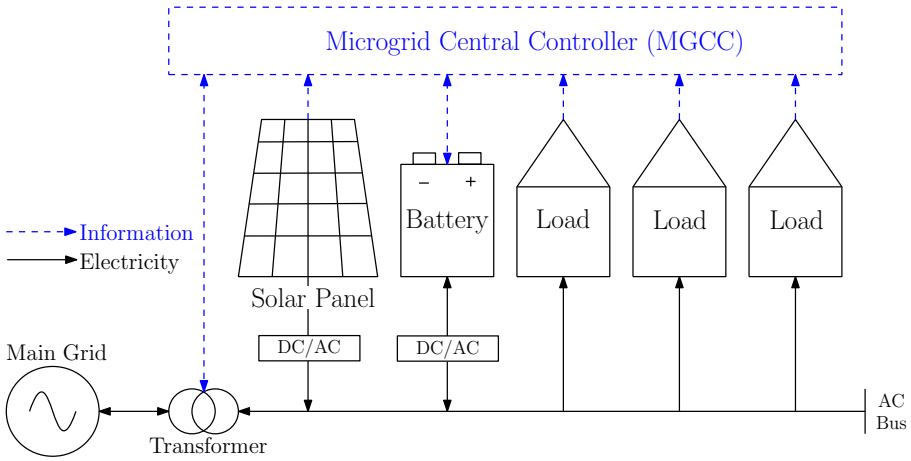

Figure 4.2 Microgrid Architecture

Controlling this MG involves switching the battery into an `ON` or `OFF` state. There are four control scenarios which are dictated by how much power is being generated by the solar panel outlined in Table 4.1.

Table 4.1 Battery control scenarios

	Battery `ON`	**Battery `OFF`**
Solar Power $\geq$ Load	Charge battery	Sell to grid
Solar Power $<$ Load	Discharge battery	Buy from grid

By default, any generated solar power is used to satisfy the load. Any excess solar power can be utilized to charge the battery or can be sold to the main grid. When there is a deficit between the solar power and the load, this missing quantity can be met from the battery reserve or can be bought from the grid. The control logic which operates in the simulation is described in Section 4.2.2.

Note that in reality other battery control scenarios would exist. For example, a combination of grid and battery power could be used to satisfy the load, or, any excess solar power could be split to have a portion charge the battery and have the other portion sold to the grid. These scenarios are not considered to favour a more tractable simulation setting.

4.2.2 Optimization-Based Microgrid Controller

The controller is tasked with satisfying the load demand while minimizing the total cost. Purchasing power from the grid increases the total cost and selling excess power reduces the

total cost. No preexisting controller software could be readily used in this MG simulation, hence a controller based on mathematical optimization has been developed using CPLEX and the OPL language [148].

At each hour, k, of the total hours in the simulation, K, the controller is provided an input set describing the state, $\mathbb{X}_k$, and determines the control set, $\mathbb{Y}_k$, to drive the MG in a cost-effective manner. This process is then performed at the next hour $k+1$, and so on until the end of the simulation. This is outlined in Figure 4.3 and System (4.1), where the controller minimizes its cost function f, subject to satisfying the set of operating constraints g.

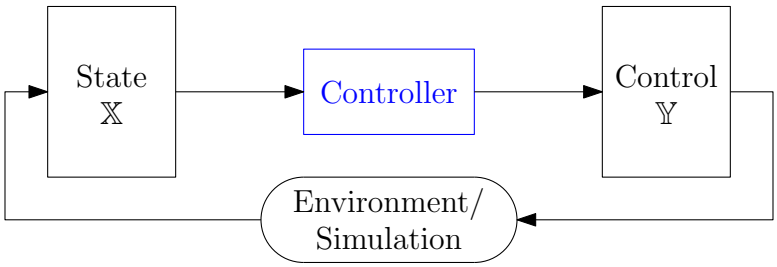

Figure 4.3 Control Loop Overview

$$\min \text{cost}_k = \min_{\mathbb{Y}_k} f(\mathbb{X}_k, \mathbb{Y}_k) \tag{4.1a}$$

$$\text{s.t.} \quad g(\mathbb{X}_k, \mathbb{Y}_k) \leq 0. \tag{4.1b}$$

The performance of such a controller can be improved if it considers what future states and decisions are likely to be. For that reason, the controller in this chapter uses forecast values for a time window of a fixed length T. The controller determines an ideal set of commands over the entire forecast window and the command determined for the current timestep will be used in the system/simulation. More precisely, at each hourly timestep, $k = 1, ..., K$, the controller is provided the forecast state values over a time horizon of $t = 0, ..., T$ hours. Note that the indexing for t starts at 0 and indicates the offset of the forecast from the current hourly timestep k. For each given timestep, k, the controller determines the optimal control actions over the entire time horizon, $k+0, k+1, ..., k+t, ..., k+T$, and then sends the determined control signal to the battery for the current timestep. This process is repeated at the next timestep, $k+1$, and so on.

For example, consider a simulation of $K = 24$ hours, with a forecast horizon of $T = 3$ hours. At time $k = 1$, the controller will determine the optimal commands over the forecast horizon for the hours $\{1, 2, 3, 4\}$. Say the commands are determined to be `ON` at hour 1, `OFF` at hour 2,

OFF at hour 3, and ON at hour 4. The command for the current timestep, $k = 1$, is determined to be ON and the battery is put into an ON state for the duration of the hour $k = 1$. At the next hourly timestep, $k + 1 = 2$, the process is repeated with the new forecast horizon for the hours $\{2, 3, 4, 5\}$.

For the control problem of this MG, the state is defined as $\mathbb{X}_k = \{P_k^L, P_k^P, \Lambda_k^B, \Lambda_k^S, b_k\}$. The set $P_k^L = \{p_{k,0}^L, p_{k,1}^L, ..., p_{k,t}^L, ..., p_{k,T}^L\}$ is the expected average cumulative load power demand values from the households over the horizon window, where $p_{k,0}^L$ is the current load power and $p_{k,1}^L, ..., p_{k,T}^L$ are the forecast loads. The set $P_k^P = \{p_{k,0}^P, p_{k,1}^P, ..., p_{k,t}^P, ..., p_{k,T}^P\}$ is the average expected generated photovoltaic solar power values over the control horizon, where $p_{k,0}^P$ is the current solar power and $p_{k,1}^P, ..., p_{k,T}^P$ are the forecast values. The buying and selling prices over the window are $\Lambda_k^B = \{\lambda_{k,0}^B, \lambda_{k,1}^B, ..., \lambda_{k,t}^S, ..., \lambda_{k,T}^B\}$ and $\Lambda_k^S = \{\lambda_{k,0}^S, \lambda_{k,1}^S, ..., \lambda_{k,t}^S, ..., \lambda_{k,T}^S\}$, respectively. The buying prices, Λ_k^B, are comprised of positive values, while the selling prices, Λ_k^B, are comprised of negative values and are used in the total cost calculation. Negative cost implies that there is a profit. The initial battery State of Charge (SOC) is b_k.

For the control problem of this MG, the control set is defined as $\mathbb{Y}_k = \{Y_k\}$. The set $Y_k = \{y_{k,0}, y_{k,1}, ..., y_{k,t}, ..., y_{k,T}\}$, is the expected set of control commands over the forecast window. This set is determined by the controller to minimize the total cost over the horizon, where $y_{k,t} = 1$ implies that the battery is expected to be in an ON state for the hour $k + t$, and, $y_{k,t} = 0$ implies that the battery is expected to be in an OFF state for the hour $k + t$. Once the control problem is solved for time k, the command for $y_{k,0}$ is given to the system. The simulation runs until the next decision is made at the next hour, $k + 1$, and so on.

To further demonstrate the control process, consider the following example, where; the current hour is $k = 2$, the forecast window is $T = 3$, and the expected set of control commands is determined to be $Y_2 = \{y_{2,0} = 0, y_{2,1} = 1, y_{2,2} = 1, y_{2,3} = 0\}$. This implies the battery:

- will be OFF for hour $k = 2 + 0 = 2$,
- is forecasted to be ON for hour $k + 1 = 2 + 1 = 3$,
- is forecasted to be ON for hour $k + 2 = 2 + 2 = 4$, and
- is forecasted to be OFF for hour $k + 3 = 2 + 3 = 5$.

The variables described up to this point are considered as the basic formulation of the MG control problem and are summarized in Table 4.2. To simplify the control problem, the formulation of the state is updated, in an intermediary step, by using P_k^L, P_k^P, Λ_k^B, and Λ_k^S to calculate two new sets; Costs C_k and Differences D_k. These sets are calculated using Algorithm 1.

Table 4.2 Formulation of MG Control Problem

State $\mathbb{X}_k$	
Load Power Forecast	$P_k^L = \{p_{k,0}^L, p_{k,1}^L, ..., p_{k,t}^L, ..., p_{k,T}^L\}$
Photovoltaic Power Forecast	$P_k^P = \{p_{0,1}^P, p_{k,1}^P, ..., p_{k,t}^P, ..., p_{k,T}^P\}$
Electricity Buy Price	$\Lambda_k^B = \{\lambda_{k,0}^B, \lambda_{k,1}^B, ..., \lambda_{k,t}^B, ..., \lambda_{k,T}^B\}$
Electricity Sell Price	$\Lambda_k^S = \{\lambda_{k,0}^S, \lambda_{k,1}^S, ..., \lambda_{k,t}^S, ..., \lambda_{k,T}^S\}$
Initial Battery SOC	b_k
Control $\mathbb{Y}_k$	
Predicted Battery Control Command	$Y_k = \{y_{k,0}, y_{k,1}, ..., y_{k,t}, ..., y_{k,T}\}$

In Algorithm 1, the value $e_{k,t}$ is a temporary variable to hold the difference between the solar power and load demand at time $k+t$. The value $c_{k,t}$ indicates how the overall cost of the controller is affected at time $k+t$. If $c_{k,t} < 0$, there is a surplus of solar power that can be sold to the grid, and, if $c_{k,t} > 0$, there is a shortfall of solar power which can be bought from the grid. The value $d_{k,t}$ indicates how the battery's SOC is affected at time $k+t$. The constant ω represents the maximum power quantity of the battery and is used to convert the battery power to a percentage (i.e. SOC). If $d_{k,t} < 0$, there is shortfall of solar power which can be retrieved from the battery (if there are sufficient reserves), and, if $d_{k,t} > 0$, there is an excess of solar power which can be supplied to the battery. The optimization problem will determine at each time $k+t$ to either, update the controller's total cost by taking quantity $c_{k,t}$, or, update the battery's SOC by taking quantity $d_{k,t}$. The updated version of the state is $\dot{\mathbb{X}}_k = \{C_k, D_k, b_k\}$.

Consider the example sets $P_k^P = \{160, 140, 100, 0\}$, $P_k^L = \{100, 120, 170, 250\}$, $\Lambda_k^B = \{3, 3, 2, 2\}$, and $\Lambda_k^S = \{-1, -1, -1, -1\}$, with $\omega = 10$. Algorithm 1 will return $C_k = \{-60, -20, 140, 500\}$ and $D_k = \{6, 2, -7, -25\}$.

Some other control variables are also needed to perform more effective control of this MG. Since the proposed controller formulation is a Mixed Integer Linear Programming (MILP) problem[1], another control variable is used to represent the opposite of the battery control set Y_k. This set can be considered as the grid control command and is defined as $\bar{Y_k} = \{\bar{y}_{k,0}, \bar{y}_{k,1}, ..., \bar{y}_{k,t}, ..., \bar{y}_{k,T}\}$. If $y_{k,t} = 1$, then $\bar{y}_{k,t} = 0$, and vice versa. Also, there is a need to determine the battery's predicted SOC over the forecast horizon, to ensure that the battery

[1]This thesis does not provide an overview of MILP principles, for a detailed description of the subject the reader is referred to Wolsey and Nemhauser [149]

Algorithm 1: Construct Costs and Differences sets at time k

$C_k := \emptyset$
$D_k := \emptyset$
for $t = 1$ *to* T **do**
  $e_{k,t} := p^P_{k,t} - p^L_{k,t}$
  if $e_{k,t} \geq 0$ **then**
    $c_{k,t} := \lambda^S_{k,t} e_{k,t}$
  else
    $c_{k,t} := -\lambda^B_{k,t} e_{k,t}$
  end
  $d_{k,t} := e_{k,t}/\omega$
  $C_k := C_k \cup c_{k,t}$
  $D_k := D_k \cup d_{k,t}$
end
return C, D

constraints are satisfied (i.e. the SOC does not go outside its minimum and maximum operating limits). The predicted battery SOC is defined as $B_k = \{b_{k,-1}, b_{k,0}, b_{k,1}, ..., b_{k,t}, ..., b_{k,T}\}$, where $b_{k,-1}$ is the initial battery SOC from the state and $b_{k,0}, b_{k,1}, ..., b_{k,T}$ are the predicted values to be calculated by the optimization model.

The updated version of the control is $\dot{\mathbb{Y}}_k = \{Y_k, \bar{Y}_k, B_k\}$. The updated state and control formulation is in Table 4.3.

Table 4.3 Updated Formulation of MG Control Problem

State $\dot{\mathbb{X}}_k$	
Costs	$C_k = \{c_{k,0}, c_{k,1}, ..., c_{k,t}, ..., c_{k,T}\}$
Differences	$D_k = \{d_{k,0}, d_{k,1}, ..., d_{k,t}, ..., d_{k,T}\}$
Initial Battery Charge	b_k
Control $\dot{\mathbb{Y}}_k$	
Predicted Battery Control Command	$Y_k = \{y_{k,0}, y_{k,1}, ..., y_{k,t}, ..., y_{k,T}\}$
Predicted Grid Control Command	$\bar{Y}_k = \{\bar{y}_{k,0}, \bar{y}_{k,1}, ..., \bar{y}_{k,t}, ..., \bar{y}_{k,T}\}$
Predicted Battery SOC	$B_k = \{b_{k,-1}, b_{k,0}, b_{k,1}, ..., b_{k,t}, ..., b_{k,T}\}$

The fully expanded optimization control problem is modeled by System (4.2):

$$\min_{Y_k, \bar{Y}_k \in \{0,1\}^T, B_k \in \mathbb{R}_+^T} \quad \sum_{t=0}^{T} \bar{y}_{k,t} c_{k,t} \tag{4.2a}$$

$$\text{s.t.} \quad b_{k,-1} = b_k \tag{4.2b}$$

$$\text{for } t = 0, 1, ..., T$$

$$b_{k,t} = b_{k,t-1} + y_{k,t} d_{k,t} \Delta^{\tau} \tag{4.2c}$$

$$y_{k,t} + \bar{y}_{k,t} = 1 \tag{4.2d}$$

$$b^{\min} \leq b_{k,t} \leq b^{\max}. \tag{4.2e}$$

The controller determines $Y_k \in \{0,1\}^T$, $\bar{Y_k} \in \{0,1\}^T$, and $B_k \in \mathbb{R}_+^T$, to minimize Objective Function (4.2a), subject to satisfying the constraints in Equations (4.2b)-(4.2e).

Objective Function (4.2a) states that the total controller's cost is the sum of the values $c_{k,t}$ when the battery is `OFF` (i.e. $\bar{y}_{k,t} = 1$ and $y_{k,t} = 0$). Equation (4.2b) sets the initial battery SOC. Equation (4.2c) states that the battery SOC at time $k + t - 1$ is equal to the previous SOC plus the value $d_{k,t}$ if the battery is `ON` (i.e. $y_{k,t} = 1$ and $\bar{y}_{k,t} = 0$), times the length of the step time Δ^{τ}. Equation (4.2d) is used to ensure that the battery is either in an `ON` or `OFF` state. Lastly, Equation (4.2e) keeps the battery SOC within its allowed operating range, where $b^{\max}$ is the battery's maximum SOC and $b^{\min}$ is the battery's minimum SOC.

For a simplified control example scenario and the CPLEX code to solve System (4.2), the reader is referred to Appendix B. For validating examples of the proposed MG control algorithm, the reader is referred to Appendix C.

4.2.3 Threat Model

System (4.2), presented in the previous section, serves as the controller that is attacked by a capable adversary. This work assumes the attacker to be a skilled adversary which has the capability to undermine the MG's underlying digital-based communications and control infrastructure to position themselves in a setting where they can modify the controller's input, $\mathbb{X}_k$, and output, $\mathbb{Y}_k$. This is illustrated in Figure 4.4.

The attacker has the ability to insert some set of input $\mathbb{A}_k^{\mathbb{X}}$ into the state values received by the controller as follows:

$$\mathbb{X}_k^{\mathbb{A}} = \mathbb{X}_k + \mathbb{A}_k^{\mathbb{X}}. \tag{4.3}$$

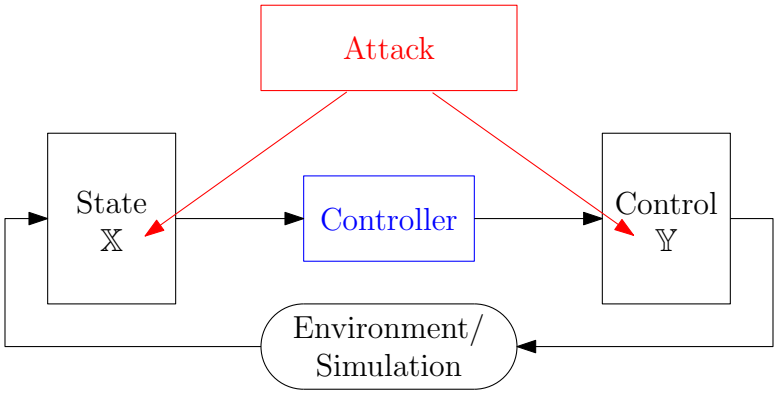

Figure 4.4 Threat model. An attacker has the ability to modify the input to the controller, $\mathbb{X}$, and the output of the controller, $\mathbb{Y}$.

The attacked state set, $\mathbb{X}_k^{\mathbb{A}} = \{P_k^{L,A}, P_k^{P,A}, \Lambda_k^{B,A}, \Lambda_k^{S,A}, b_k^A\}$, will be used as the input to the controller. Note that, if $\mathbb{A}_k^{\mathbb{X}} = \emptyset$, then $\mathbb{X}_k^{\mathbb{A}} = \mathbb{X}_k$.

The attacker can also insert some set of input $\mathbb{A}_k^{\mathbb{Y}}$ into the control values outputted by the controller as follows:

$$\mathbb{Y}_k^{\mathbb{A}} = \mathbb{Y}_k + \mathbb{A}_k^{\mathbb{Y}}. \tag{4.4}$$

The attacked control set, $\mathbb{Y}_k^{\mathbb{A}} = \{Y_k^A\}$, will be used as the control invoked into the simulation. The attacked battery control command, $y_{k,0}^A \in Y_k^A$, is used as the control command at time k. Again note that, if $\mathbb{A}_k^{\mathbb{Y}} = \emptyset$, then $\mathbb{Y}_k^{\mathbb{A}} = \mathbb{Y}_k$.

The goal of the attacker is to increase the total cost of the controller by constructing $\mathbb{A}_k^{\mathbb{X}}$ and/or $\mathbb{A}_k^{\mathbb{Y}}$ in such a manner that the controller invokes sub-optimal commands.

4.2.4 Testbed Implementation Details

This work uses an MG model which comes by default with MATLAB/Simulink as the simulation environment (see Figure 4.5) [150]. This model has been considerably modified to enable the utilization of custom controller logic and the ability to insert falsified values from simulated attacks.

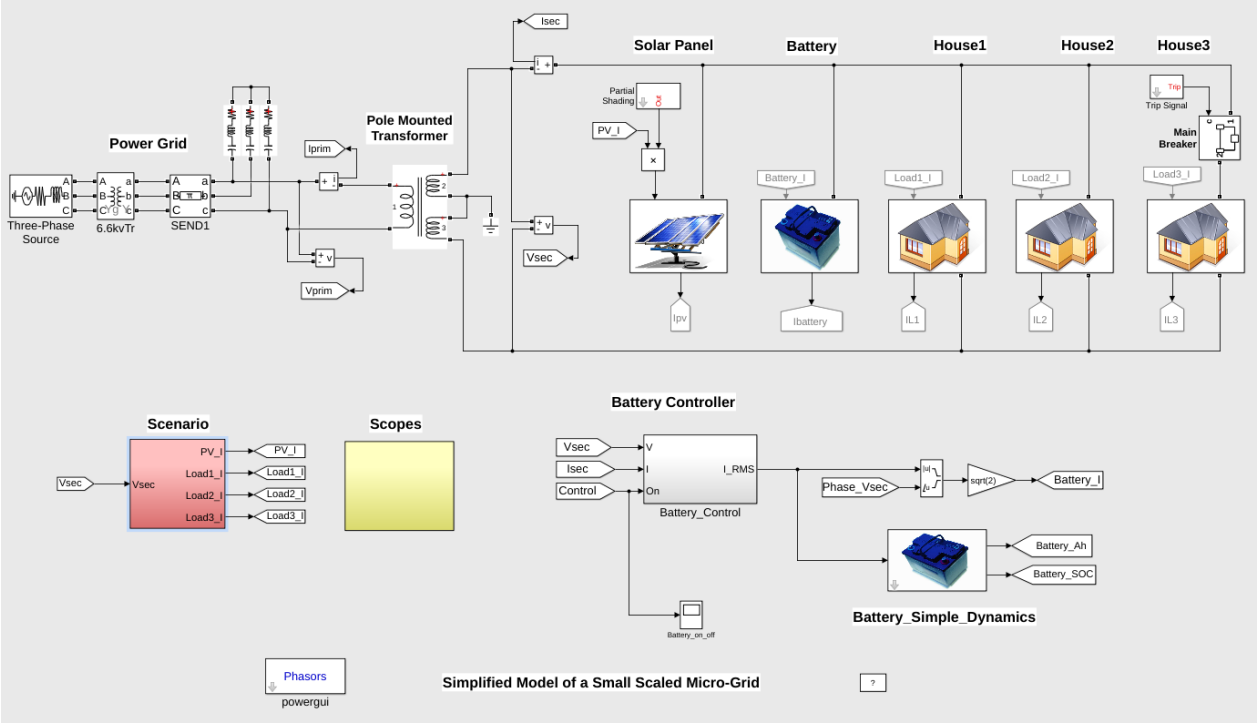

Figure 4.5 Screenshot of MATLAB/Simulink Microgrid Model

This MATLAB/Simulink model has been integrated into the developed MG testbed environment, which is illustrated at a high level in Figure 4.6. The figure demonstrates how the developed software components are integrated to each other by demonstrating their inputs and outputs. More precisely, in Figure 4.6 we have the following modules:

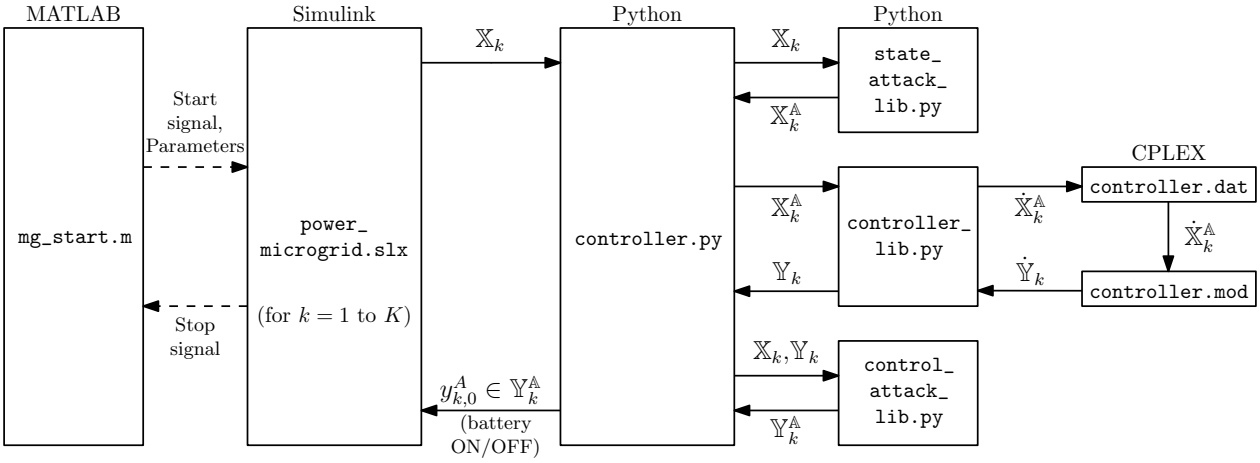

Figure 4.6 Overview of Microgrid Testbed Implementation

`mg_start.m` This component is a Matlab script which loads the simulation parameters and initiates the MG simulation. The simulation parameters include; the length of simulation in hours (i.e. K), the controller logic, the attack logic, the electrical load data, the solar irradiance data, the electricity buying price data, and the electricity selling price data. The data is loaded from Comma-Separated-Values (CSV) files with entry values existing for every minute. The parameters are held in memory to be used by `power_microgrid.slx`. After loading all the necessary parameters, the script initiates `power_microgrid.slx`. Once the simulation terminates, this module calls other Matlab scripts to calculate the simulation score and generate graphs (not shown in Figure 4.6).

`power_microgrid.slx` This component is a Simulink model (from Figure 4.5) which performs the MG simulation scenario. The module uses the electrical load and solar irradiance data to perform the voltage regulation activities in accordance with the battery control commands. Within this component is the `Battery_Control` module which has been modified to use custom battery control software developed for this research. At every hour in the simulation, k, this module passes the state values, $\mathbb{X}_k$, to the `controller.py` script and receives the control command as a response from the `controller.py` script. If the control command is $y^A_{k,0} = 1$, then the battery will be in an `ON` state for the duration of the timestep k. If the control command is $y^A_{k,0} = 0$, then the battery will be in an `OFF` state for the duration of the timestep k. This process is continued for all timesteps $k = 1, ..., K$. Once all electrical load

and solar irradiance values have been evaluated, the simulation is complete, and the control is reverted back to `mg_start.m`.

`controller.py` This component is a Python script which receives the state of the simulation, $\mathbb{X}_k$, from `power_microgrid.slx`, and uses other Python libraries to determine the control to invoke in the simulation as well as attacks to perform. Since the attacks in this research are simulated and occur immediately before and after the controller logic, it has been deemed acceptable to place the attacks at this point in the testbed. To simulate state attacks, this script passes the state, $\mathbb{X}_k$, to `state_attack_lib.py` and receives the attacked state set $\mathbb{X}_k^{\mathbb{A}}$ as response. This script then passes the attacked state, $\mathbb{X}_k^{\mathbb{A}}$, to `controller_lib.py` and receives the determined control set, $\mathbb{Y}_k$, as a response. To simulate control attacks, this script passes the state set, $\mathbb{X}_k$, and the control set, $\mathbb{Y}_k$, to `control_attack_lib.py`, then receives the attacked control set, $\mathbb{Y}_k^{\mathbb{A}}$, as a response. The value for $y_{k,0}^{A}$ (either 1 or 0) from the attacked control set ,$\mathbb{Y}_k^{\mathbb{A}}$, is returned to `power_microgrid.slx` as the control command for the hour.

`state_attack_lib.py` This component is a Python script which receives the state of the simulation, $\mathbb{X}_k$, and determines the attacked state set, $\mathbb{X}_k^{\mathbb{A}}$, to maliciously modify the state reported to the controller. This script comprises several attacks which can be selected by a parameter in `mg_start.m`.

`control_attack_lib.py` This component is a Python script which receives the state set, $\mathbb{X}_k$, as well as the calculated control to be invoked in the simulation, $\mathbb{Y}_k$, and determines the control attack set, $\mathbb{Y}_k^{\mathbb{A}}$, to maliciously modify the control to be used in the simulation. This script comprises several attacks which can be selected by a parameter in `mg_start.m`.

`controller_lib.py` This component is a Python script which updates the received state set, using Algorithm 1, to create the updated state set, $\dot{\mathbb{X}}_k^{\mathbb{A}}$, writes this information into the file `controller.dat`, and triggers the execution of solving of the optimization-based control problem by `controller.mod`. After the optimization problem is solved, the updated control set, $\dot{\mathbb{Y}}$, is parsed from a text file created by `controller.mod`, and the control set $\mathbb{Y}$ is passed to `controller.py`.

`controller.dat` This component is a CPLEX data file which contains the given variables used in solving the optimization-based control problem of `controller.mod`.

`controller.mod` This component is a CPLEX module which is comprised of the codified representation of System (4.2). This module solves the optimization problem and writes the solved values for $\dot{\mathbb{Y}}$ to a text file which is parsed by `controller_lib.py`. The code for this CPLEX module is available in Appendix B.

4.3 Penetration Testing of the Microgrid Control Algorithm

Penetration Testing (PT) is the process of performing an authorized attack on a system in order to uncover its vulnerabilities or to witness how the system will act in the presence of a cyberattack [151]. This is an accepted activity which is practiced by specialized professionals, studied by academia, and is commonly used in industry for assessing the defensive posture of a system. An MG operator is interested in knowing the inherent vulnerabilities within their system and should regularly perform PT to understand their defensive posture against potential cyberattacks. PT generally involves looking for defensive coverage blinspots in software and hardware infrastructure. However, this work proposes that the logic in control algorithms should also be considered by PT activities. This section demonstrates the effects of simulated FDI attacks against the proposed MGCC environment from Section 4.2 to gain an understanding of the vulnerabilities of the control logic.

4.3.1 Experimental Setup

The MG implementation uses the parameters, $\omega = 125000$ kilowatts (kW) in Algorithm 1 and $\Delta^{\tau} = 60$ minutes in Equation (4.2c).

The battery has an operating range between 75% and 100% SOC, thus $b^{\min} = 75$ and $b^{\max} = 100$. This was chosen to prevent the controller from discharging all of its available power, causing the MG to not have any reserve power in the event of an emergency.

The MG simulation reads a CSV file with solar irradiance and household electrical demands for each minute of the simulation period. The length of the CSV file is $K * 60$, where K is the total hours of the simulation and varies depending on the length of the experiment. The solar and load forecast values are the average value for the quantity taken from the CSV over the hour. For example, say $k = 1$ implies the time 00:00, the value for $p^{S}_{1,0} :=$ (CSV solar value at 00:00 + CSV solar value at 00:01 + ... + CSV solar value at 00:59)/60. This forecast method is used as a simplification and is deemed satisfactory for this research since the goal of this work is not to assess the performance of the control algorithm. The length of the forecast window is fixed to $T = 23$ (a total length of 24 hours).

The buying prices fluctuate throughout the day such that peak consumption times are more expensive. The pricing scheme used in all experiments is the same as for Ontario Hydro residential customers where the price varies throughout the day at fixed intervals [152]:

- **Off-peak:** 6.5 cents/kWh (19:00-07:00)
- **Mid-peak:** 9.4 cents/kWh (07:00-12:00 and 17:00-19:00)
- **On-peak:** 13.4 cents/kWh (12:00-17:00)

The selling price is fixed:

- **Constant:** -5.0 cents/kWh

The score of the controller over a simulation is the summation of the cost incurred at each minute. The cost value will be negative when electricity is sold to the main grid and positive when electricity is bought from the main grid.

Note that, for all simulations the battery is in an `ON` for the first hour due to a limitation in the Simulink model `power_microgrid.slx`

4.3.2 Manual Attacks on Default Data Scenario

This section is adapted from work presented at the 2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2020) [153].

This section considers attacks performed against the controller over a period of $K = 168$ hours (i.e. 7 days), where the solar irradiance and electrical load data are default values which come with the MATLAB/Simulink model. The values are identical for each day of the week-long period. This is considered as the 'clean' version of the control and attack problem. The five following attacks are considered:

- **(State Attack) Initial SOC +5%:** The attacker modifies the battery's inital SOC to be 5% higher than what it is in reality. That is, $b_k^A := b_k + 5$.
- **(State Attack) Initial SOC -5%:** The attacker modifies the battery's inital SOC to be 5% lower than what it is in reality. That is, $b_k^A := b_k - 5$.
- **(Control Attack) Switch control command every hour:** The attacker modifies the battery control value to be the opposite of what the controller decides. That is, $y_{k,0}^A := \begin{cases} 1, & \text{if } y_{k,0} = 0 \\ 0, & \text{if } y_{k,0} = 1. \end{cases}$

- **(Control Attack) Switch control command every 5 hours:** The attacker modifies the battery control value to be the opposite of what the controller decides at every fifth hour. That is, $y^A_{k,0} := \begin{cases} 1, & \text{if modulo}(k,4) = 0 \text{ and } y_{k,0} = 0 \\ 0, & \text{if modulo}(k,4) = 0 \text{ and } y_{k,0} = 1 \\ y_{k,0}, & \text{if modulo}(k,4) > 0. \end{cases}$
- **(Control Attack) Maximize cost control:** The attacker solves System (4.2) as a maximization problem to determine the attacked control command, regardless of what the controller chooses. That is, $y^A_{k,0} := y_{k,0} \in \arg \max_{\mathbb{Y}_k} f(\mathbb{X}_k, \mathbb{Y}_k),$ s.t. $g(\mathbb{X}_k, \mathbb{Y}_k) \leq 0.$

As an illustrative example, Figure 4.7 shows the control of the MG simulation with *no attack* and an initial battery SOC of 80%. Here, the controller charges the battery whenever there is excess solar power and performs *peak shaving* to offset power demand during periods of expensive cost.

Interpreting the Figure: There is a lot of information being displayed in Figure 4.7 and this style of figure is reused throughout this document. An explanation of the axes are provided to ensure the figure is properly interpreted. The axes of the figure are interpreted as follows:

- **Load Power Sources**: Description of the allocation of the power sources in satisfying the demand and charging the battery, measured in kW. The quantities are as follows:
  - **PV-to-Load (dark blue)**: quantity of PV power used to satisfy the load
  - **Batt-to-Load (purple)**: quantity of battery power used to satisfy the load
  - **PV-to-Batt (light blue)**: quantity of PV power used to charge the battery
  - **Grid-to-Load (red)**: quantity of grid power to satisfy the load
  - **PV-to-Grid (green)**: quantity of solar power sold to the grid
  - **Load (yellow line)**: the fluctuating load over time
- **Battery Control**: Binary value of either 0 for `OFF` or 1 for `ON`.
- **Battery Charge**: The battery SOC shown as a percent. 0% is empty and 100% is fully charged.
- **Prices**: Prices for buying (red) and selling (green) throughout the week, measured in \$/kWh. Buying power increases the total cost and selling power decreases the total cost.

- **Cost per minute**: The cost incurred at the given minute, measured in $\$\times 10^{-3}$. Red indicates electricity is bought from the grid at a positive cost and green indicates electricity is sold to the grid at a negative cost.
- **Total Cost**: The total cost incurred over the course of the simulation, measured in \$. Red indicates a positive cost and green indicates a negative cost.

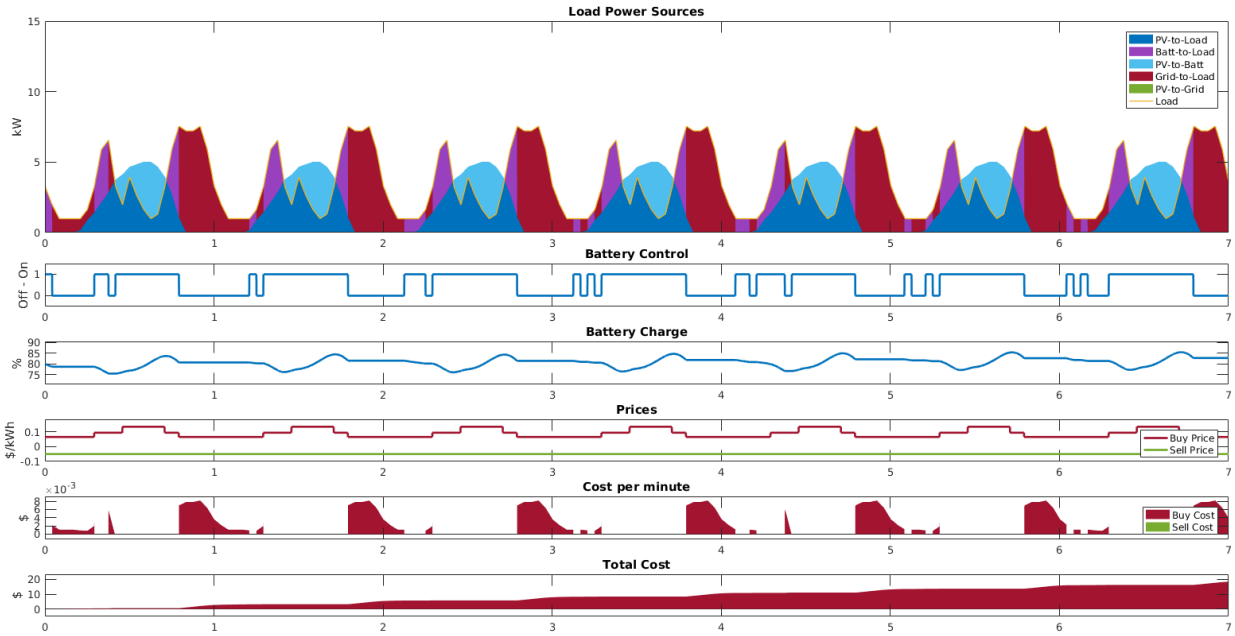

Figure 4.7 Scenario: default data, init. batt. SOC 80%, no attack (Cost: \$18.40)

Figure 4.8 shows the control of the MG simulation with the *maximize cost control attack* and an initial battery SOC of 80%. Here, the attacker aims to purchase as much power as possible from the main grid and not use the battery reserves to offset the load demand. The attack charges the battery as close as possible to its maximum, to initially not sell any excess solar power, then chooses to sell excess solar power to the main grid since the selling price is less than the purchasing price.

The remaining attacks for the default data scenario, with an initial SOC of 80%, are shown in Appendix D, in Figures D.1- D.4.

Table 4.4 shows the results of the simulated attacks against the MGCC, over the one week default-data simulation period with varying initial battery SOC levels. When the initial battery SOC is reported to being 5% higher than in reality, the controller will utilize the battery reserves outside of peak demand periods, as it believes there is battery power to

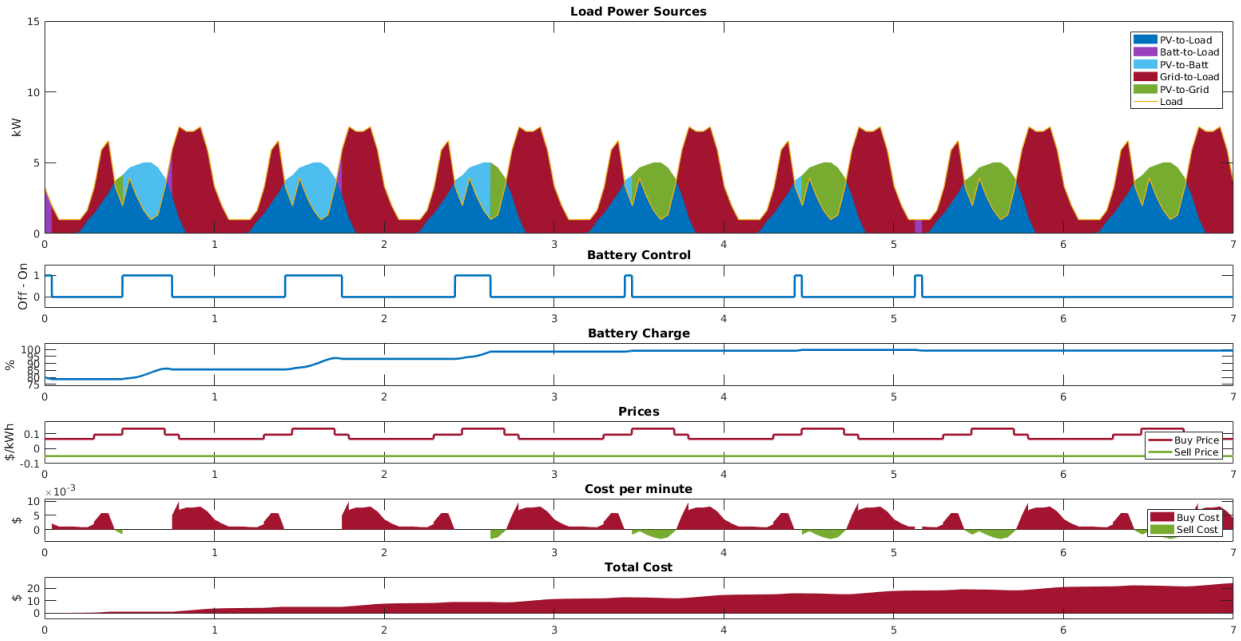

Figure 4.8 Scenario: default data, init. batt. SOC 80%, maximize cost control attack (Cost: $24.32)

spare, and will be less effective at performing *peak shaving*. When the initial battery SOC is reported to being 5% lower than in reality, the controller believes there is less battery power available and is less effective at performing *peak shaving*. When the attacker switches the control command, excess solar power is sold to the main grid, when it should be used to charge the battery, and the battery reserves are not used to offset the load demand. The switching effect is most pronounced when the control is switched every hour, but there is still a noticeable impact when the switch occurs at every fifth hour, which may be a more realistic and elusive attack to detect.

The most notable effect generally comes from the attack which chooses an attacked control value by solving System (4.2) as a maximization problem[2]. This attack chooses a control value which charges the battery to maximum capacity and purchases electricity from the main grid whenever possible. Once the battery is fully charged it has no choice but to sell some electricity to the main grid at a relatively small profit. Despite this profit, this attacking strategy is seen to be the most effective of the compared attacks.

[2]This attack is referred to as *Max Cost* since it solves a maximization problem to determine the attacked control value. However, this does not necessarily mean that it generates the maximum cost possible for every particular scenario, since some work can be done to further improve the maximization problem. For instance, in Table 4.4, for the scenario with the initial battery SOC of 100%, the *Switch (1)* attack generates a slightly higher cost.

Table 4.4 Simulation results on Default Data for 168-hour (7-day) period with varying initial battery SOC values and attack scenarios

	Init. Batt. SOC: 75%			**Init. Batt. SOC: 80%**		
Attack	Cost ($)	Cost Incr.	Avg. SOC	Cost ($)	Cost Incr.	Avg. SOC
None	19.20	-	80.31%	18.40	-	80.69%
SOC +5%	19.25	+0.26%	78.00%	18.51	+0.60%	78.24%
SOC -5%	20.01	+4.22%	84.81%	19.53	+6.14%	84.96%
Switch (1)	21.89	+14.01%	75.02%	21.34	+15.98%	75.12%
Switch (5)	19.87	+3.49%	77.28%	19.15	+4.08%	77.55%
Max Cost	24.71	+28.70%	93.10%	24.32	+32.17%	94.71%
	Init. Batt. SOC: 85%			**Init. Batt. SOC: 90%**		
Attack	Cost ($)	Cost Incr.	Avg. SOC	Cost ($)	Cost Incr.	Avg. SOC
None	17.77	-	81.76%	17.06	-	82.31%
SOC +5%	17.82	+0.28%	79.18%	17.07	+0.06%	79.13%
SOC -5%	18.35	+3.26%	85.41%	17.77	+4.16%	86.76%
Switch (1)	20.74	+16.71%	75.80%	20.09	+17.76%	76.64%
Switch (5)	18.40	+3.55%	78.21%	17.35	+1.70%	80.67%
Max Cost	23.43	+31.85%	96.58%	22.94	+34.47%	98.00%
	Init. Batt. SOC: 95%			**Init. Batt. SOC: 100%**		
Attack	Cost ($)	Cost Incr.	Avg. SOC	Cost ($)	Cost Incr.	Avg. SOC
None	16.19	-	82.01%	15.85	-	83.78%
SOC +5%	16.50	+0.28%	79.57%	16.85	+6.31%	80.80%
SOC -5%	17.06	+3.26%	87.31%	16.70	+5.36%	87.44%
Switch (1)	19.31	+16.71%	77.64%	22.56	+42.33%	100.00%
Switch (5)	16.86	+3.55%	79.60%	16.13	+1.77%	81.05%
Max Cost	22.92	+31.85%	98.31%	22.04	+39.05%	99.11%

4.3.3 Manual Attacks on Real-World Data Scenario

This section considers attacks performed against the controller over a period of $K = 168$ hours (i.e. 7 days), where the solar irradiance and electrical load data are taken from real-world data. The household loads come from a freely accessible dataset of one minute samples of household power consumption from a household in Sceaux, France over a four year period [154]. The three MG households are represented by the days of October 6-13, 2008, October 12-19, 2009, and October 11-18, 2010. Solar irradiance data could not be found from Sceaux,

France, so solar data was taken from a week with mostly sunny days in Ottawa, Canada for the days of July 07-17, 2018 [155]. The solar irradiance measurements from the data source occur at every hour, so a linear interpolation is used to create entries for each minute in the period. The simplification to the solar irradiance values and the use of data sources from differing locations are deemed acceptable for the purpose of this research, as the constructed scenario is presumed to be a fair approximation of a real-world setting. The five attacks from Section 4.3.2 are reused in this section.

As an illustrative example, Figure 4.9 shows the control of the MG simulation with *no attack* and an initial battery SOC of 90%. Here, the controller generally charges the battery whenever there is excess solar power and performs *peak shaving* to offset power demand during periods of expensive cost. Additionally, on the first day the controller sells a portion of the excess solar power when the battery is near its upper capacity limit and also sells other small excesses on the third, fifth, sixth, and seventh day.

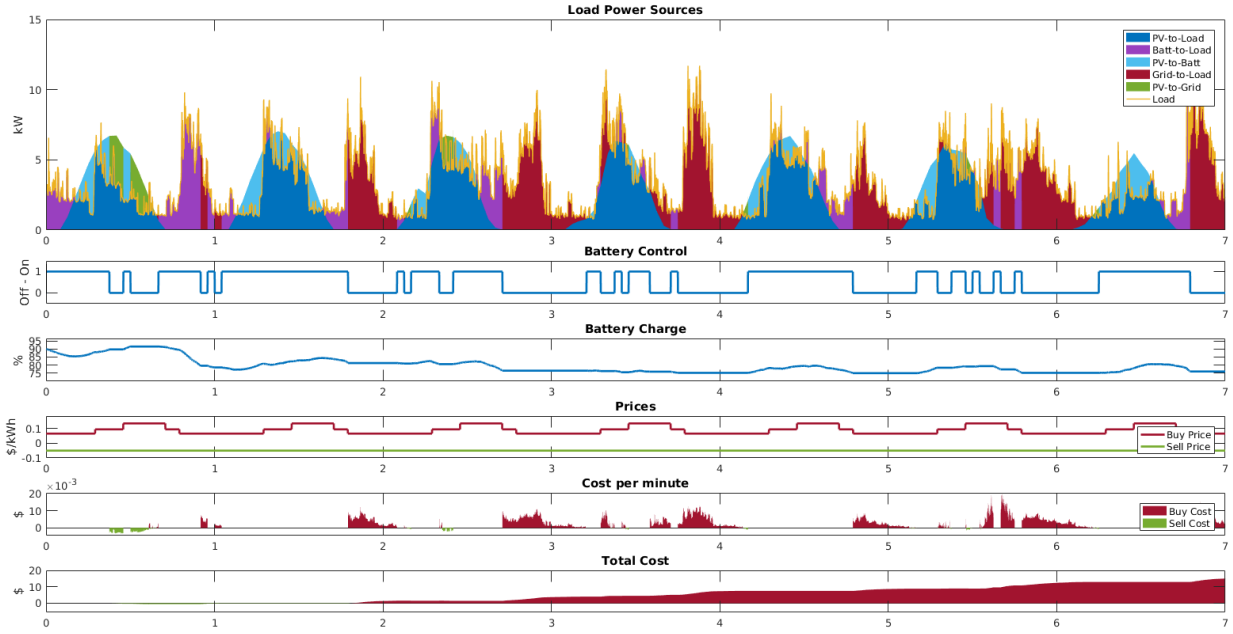

Figure 4.9 Scenario: real-world data, init. batt. SOC 90%, no attack (Cost: $15.14)

Figure 4.10 shows the control of the MG simulation with the *maximize cost control attack* and an initial battery SOC of 90%. Here, the attacker aims to purchase as much power as possible from the main grid and not use the battery reserves to offset the load demand. The attack charges the battery as close as possible to its maximum, to initially not sell any excess

solar power, then chooses to sell excess solar power to the main grid since the selling price is less than the purchasing price.

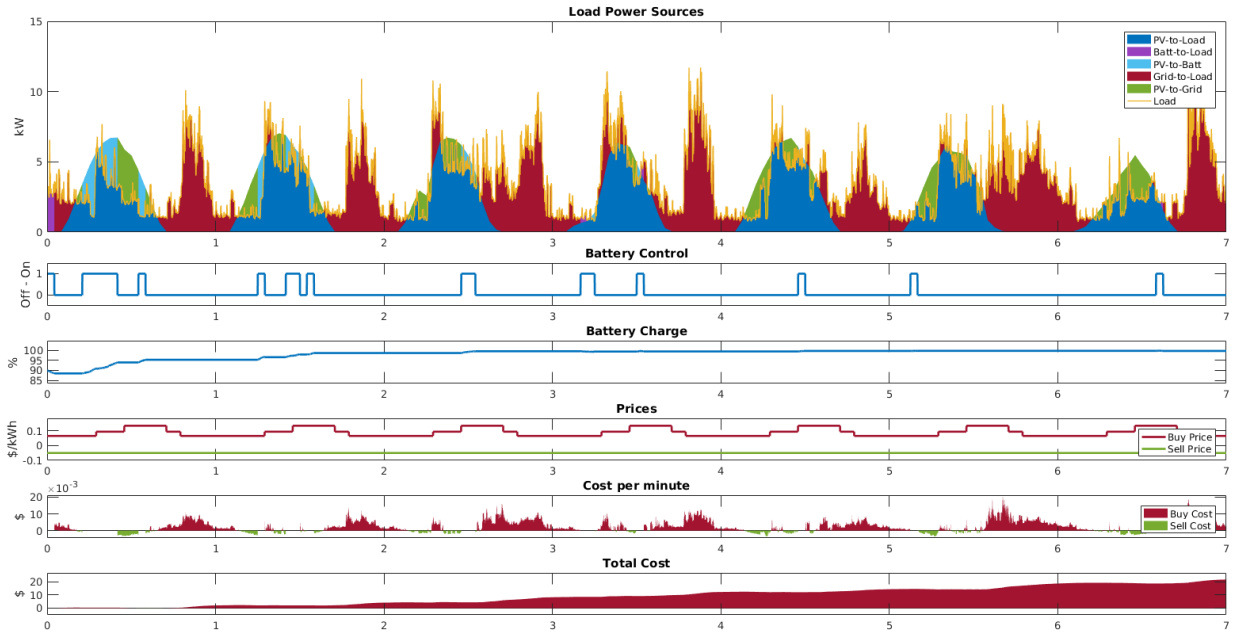

Figure 4.10 Scenario: real-world data, init. batt. SOC 90%, maximize cost control attack (Cost: $21.77)

The remaining attacks for the real-world data scenario, with an initial SOC of 90%, are shown in Appendix E, in Figures E.1- E.4.

Table 4.5 shows the results of the simulated attacks against the MGCC, over the one week real-data simulation period with varying initial battery SOC levels. The description of Table 4.4 from Section 4.3.2 generally applies to Table 4.5. The exception being for the attacks which report a battery SOC of 5% higher than in reality for the scenarios where the initial battery SOC is 80%, 85%, and 90%. Here the attacks actually induce a lower operating cost compared to when there is no attack present. This is due the particularities of the data and the perceived higher battery SOC causing the controller to more aggressively operate near the battery's lower SOC limit.

4.3.4 Reinforcement Learning Based Attacks on Real-World Data Scenario

This section is adapted from work presented at the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC 2021) [156].

Table 4.5 Simulation results on Real-world Data for 168-hour (7-day) period with varying initial battery SOC values and attack scenarios

	Init. Batt. SOC: 75%			**Init. Batt. SOC: 80%**		
Attack	Cost ($)	Cost Incr.	Avg. SOC	Cost ($)	Cost Incr.	Avg. SOC
None	16.52	-	77.94%	16.41	-	78.66%
SOC +5%	16.71	+1.15%	77.35%	16.02	-2.38%	77.48%
SOC -5%	17.02	+3.03%	82.07%	16.65	+1.46%	80.48%
Switch (1)	20.39	+23.43%	75.00%	19.84	+20.90%	75.10%
Switch (5)	18.20	+10.17%	76.31%	17.57	+7.07%	76.91%
Max Cost	22.95	+38.92%	94.47%	21.77	+32.66%	96.62%
	Init. Batt. SOC: 85%			**Init. Batt. SOC: 90%**		
Attack	Cost ($)	Cost Incr.	Avg. SOC	Cost ($)	Cost Incr.	Avg. SOC
None	15.56	-	78.80%	15.14	-	79.32%
SOC +5%	15.37	-1.22%	77.75%	14.96	-1.19%	77.99%
SOC -5%	16.20	+4.11%	81.97%	15.83	+4.56%	84.04%
Switch (1)	19.29	+23.97%	77.17%	22.00	+45.31%	100.00%
Switch (5)	16.22	+4.24%	78.03%	16.09	+6.27%	78.40%
Max Cost	22.26	+43.06%	97.71%	21.77	+43.79%	98.20%
	Init. Batt. SOC: 95%			**Init. Batt. SOC: 100%**		
Attack	Cost ($)	Cost Incr.	Avg. SOC	Cost ($)	Cost Incr.	Avg. SOC
None	14.53	-	79.81%	14.25	-	80.26%
SOC +5%	14.66	+0.89%	78.55%	14.38	+0.91%	80.68%
SOC -5%	14.92	+2.68%	82.15%	14.47	+1.54%	81.99%
Switch (1)	21.05	+44.87%	99.91%	20.59	+44.49%	100.00%
Switch (5)	16.45	+13.21%	78.99%	14.73	+3.37%	80.86%
Max Cost	20.45	+40.74%	98.34%	20.20	+41.75%	99.69%

This section proposes the use of a form of AI known as Reinforcement Learning (RL) as a methodology to automate the generation of PT attacks and evaluates this approach using attacks on the simulated MG. Through trial-and-error episodic interactions with the simulated MG, an RL agent is trained to craft near-optimal malicious values for the initial battety SOC used by the MGCC in solving System (4.2).

Finding Optimal Malicious Input

The attacker's goal in this setting is to find the ideal malicious input to provide to the controller, so that the controller performs the worst possible action. This type of problem can be represented as a BPP, where the attacker constructs a set of input to be provided to the controller.

The general BPP for FDI attacks, where the attacker can maliciously insert falsified state or control values is provided in System (4.5):

$$\max_{\mathbb{A}_k^{\mathbb{X}}, \mathbb{A}_k^{\mathbb{Y}}} \quad F(\mathbb{X}_k, \mathbb{Y}_k + \mathbb{A}_k^{\mathbb{Y}}) \tag{4.5a}$$

$$\text{s.t.} \quad G(\mathbb{A}_k^{\mathbb{X}}, \mathbb{A}_k^{\mathbb{Y}}) \leq 0 \tag{4.5b}$$

$$\min_{\mathbb{Y}_k} \quad f(\mathbb{X}_k + \mathbb{A}_k^{\mathbb{X}}, \mathbb{Y}_k) \tag{4.5c}$$

$$\text{s.t.} \quad g(\mathbb{X}_k + \mathbb{A}_k^{\mathbb{X}}, \mathbb{Y}_k + \mathbb{A}_k^{\mathbb{Y}}) \leq 0. \tag{4.5d}$$

The attacker's actions are represented in the upper part of the BPP, where the attacker is trying to maximize its goal F by choosing $\mathbb{A}_k^{\mathbb{X}}$ and $\mathbb{A}_k^{\mathbb{Y}}$, subject to its constraints G. The attacker must take into account that the controller chooses a control $\mathbb{Y}_k$ which minimizes its total cost function, f, subject to its constraints, g, given the information it is provided. In the case where the attacker is trying to subvert the goal of the controller, F will be the same as f.

This work specifically considers the case where the attacker modifies the initial battery SOC that is reported to the controller. This is shown in System (4.6), where at timestep k, the attacker is given the state, $\dot{\mathbb{X}}_k$, and determines some malicious value, a_k^b, to add to the initial battery state of charge, b_k, in order to create an attacked initial battery SOC, b_k^A, which will be used by the controller in its control problem.

$$\max_{a_k^b \in \mathbb{R}, b_k^A \in \mathbb{R}_+} \quad \sum_{t=0}^{T} \bar{y}_{k,t} c_{k,t} \tag{4.6a}$$

$$\text{s.t.} \quad b_k^A = b_k + a_k^b \tag{4.6b}$$

$$a^{b^{\min}} \leq a_k^b \leq a^{b^{\max}} \tag{4.6c}$$

$$b^{\min} \leq b_k^A \leq b^{\max} \tag{4.6d}$$

$$\min_{Y_k, \bar{Y}_k \in \{0,1\}^T, B_k \in \mathbb{R}_+^T} \quad \sum_{t=0}^{T} \bar{y}_{k,t} c_{k,t} \tag{4.6e}$$

$$\text{s.t.} \quad b_{k,-1} = b_k^A \tag{4.6f}$$

$$\text{for } t = 0, 1, ..., T$$

$$b_{k,t} = b_{k,t-1} + y_{k,t} d_{k,t} \Delta^\tau \tag{4.6g}$$

$$y_{k,t} + \bar{y}_{k,t} = 1 \tag{4.6h}$$

$$b^{\min} \leq b_{k,t} \leq b^{\max}. \tag{4.6i}$$

BPPs are hard to solve to optimality since they are generally non-convex, non-differentiable, and have been shown to be at least NP-hard [97]. However, it is in the interest of an MG operator to understand which malicious values could be inserted into their system and have it operate in a sub-optimal manner. Due to the difficulty of solving BPPs, this work proposes the use of RL as a metaheuristic to uncover approximate optimal solutions to this MG attack problem.

Reinforcement Learning for Penetration Testing

This work considers using an RL agent to learn an effective attack strategy in an MG testbed environment. Successes in RL-based game-playing agents for Go [157] and StarCraft II [158] have shown the ability of RL agents to uncover strategy-spaces not considered by the best human players, which also may be the case for PT activities.

RL can be characterized as a form of statistical learning where an agent learns to perform actions in an environment with the goal of maximizing some notion of long term reward [159]. The agent's goal is to learn a near-optimal policy π, through repeated trial and error episodic interactions with an environment. There is a balance to be played of exploring unseen decision-spaces while exploiting known 'good' decisions-spaces. RL problems are often modelled as some form of a Markov Decision Process (MDP) where there exists a set of states S, a set of actions A, the probability $P_a(s, s')$ to transition from state s to state s' given action

a, and the reward $R_a(s, s')$ for transitioning from state s to s' using action a. Formulating PT activities as an RL problem involves defining the testbed environment's state, the set of actions an attacking agent can perform, the reward which the agent receives for taking actions, and a method for updating the agent's policy [160–162].

Training Setup

The training occurs over a period of $K = 48$ hours, where the solar irradiance values are taken from Ottawa, Canada, for the days of July 20-22, 2015 [155] and the household load values come from the dataset from a household in Sceaux, France, for the days of October 6-8, 2008, October 12-14, 2009, and October 11-13, 2010, summed together [154].

The attacker has the ability to provide a malicious input, a_k^b, to be added to the initial battery SOC, b_k, which is passed to the controller algorithm as the value $b_k^A := b_k + a_k^b$. The RL learning agent is tasked in determining what is a 'good' value to provide to the controller as input, causing it to make sub-optimal decisions and increase the overall cost incurred. This work utilizes an implementation of the Temporal Difference Advantage Actor-Critic algorithm [163], since the RL agent's action involves choosing a continuous value [164, 165]. The details of the implementation are in Appendix F.

At each timestep, k, the RL agent receives the state, $s_k = \{b_k, c_{k,0}, d_{k,0}, p_{k,0}^L, p_{k,0}^S\}$, where b_k is the current battery SOC, $c_{k,0}$ is the cost value over the current hour, $d_{k,0}$ is the difference value over the current hour, $p_{k,0}^L$ is the current load power, and $p_{k,0}^S$ is the current solar power. The algorithm samples an action from the current policy to determine the value a_k^b to construct b_k^A. The input is provided to the controller and the simulation runs until the next hourly timestep. The total cost of the controller over the hour is summed (and multiplied by 1000), then returned as the reward r_k along with the new state of the system $s'_k = s_{k+1}$. The actor-critic networks are updated to minimize the actor and critic total loss, with the goal of finding optimal weight parameters for maximizing total reward. An episode consists of continuing this process for a 48-hour simulation in the testbed. The cumulative reward for an episode is $\sum_{k=1}^{K} r_k$.

Training Results

Figure 4.11 shows the cumulative reward of the trained RL agent over 1000 episodes. This agent is permitted to choose any value for a_k^b, while respecting the requirement that $75 \leq b_k + a_k^b \leq 100$. The cumulative reward trends upward as the agent learns an effective strategy

for this scenario, converging near a value of 4600. The long training time is typical for the Temporal Difference Advantage Actor-Critic algorithm.

For comparison, an agent is trained which only has the ability to modify the reported battery charge by plus or minus 5 percent of the actual battery charge. That is, it can choose a value for a_k^b, such that $-5 \leq a_k^b \leq 5$ and $75 \leq b_k + a_k^b \leq 100$. The training is is shown in Figure 4.12. The capabilities of this agent in increasing the total cost is more limited and there is a more gradual convergence.

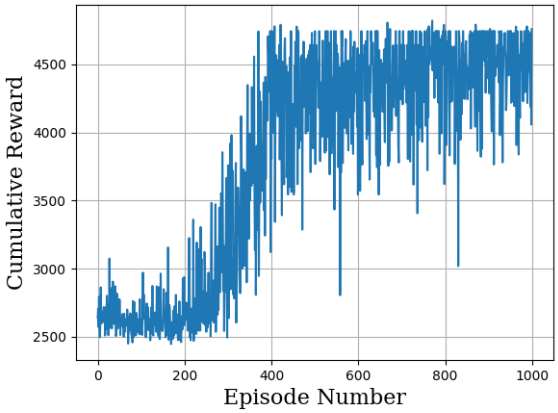

Figure 4.11 Training of RL agent which can choose any value for a_k^b, such that $75 \leq b_k + a_k^b \leq 100$.

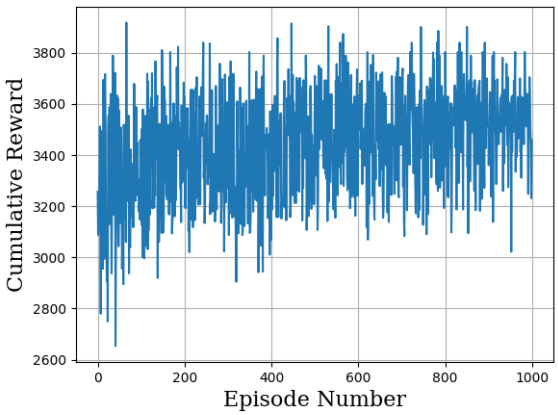

Figure 4.12 Training of RL agent which can choose $-5 \leq a_k^b \leq 5$, such that $75 \leq b_k + a_k^b \leq 100$.

Attack Results

As an illustrative example, Figure 4.13 shows the control of the MG simulation with *no attack* and an initial battery SOC of 80%. Here the controller received the correctly reported value for the battery SOC and is capable to perform effective *peak shaving*.

Figure 4.14 shows the control of the MG simulation where the attacking agent has the ability to choose any value for the battery SOC that is passed to the controller. The attacker generally reports that the battery charge is near the allowed lower bound of 75%, causing the controller to believe there is little reserves to be used to satisfy the load. The effect is that controller is less effective at performing *peak shaving*, causing the battery to not offset the demand, and the charge of the battery to rise. The overall score is substantially higher than the same scenario in Figure 4.13 where there is no attack.

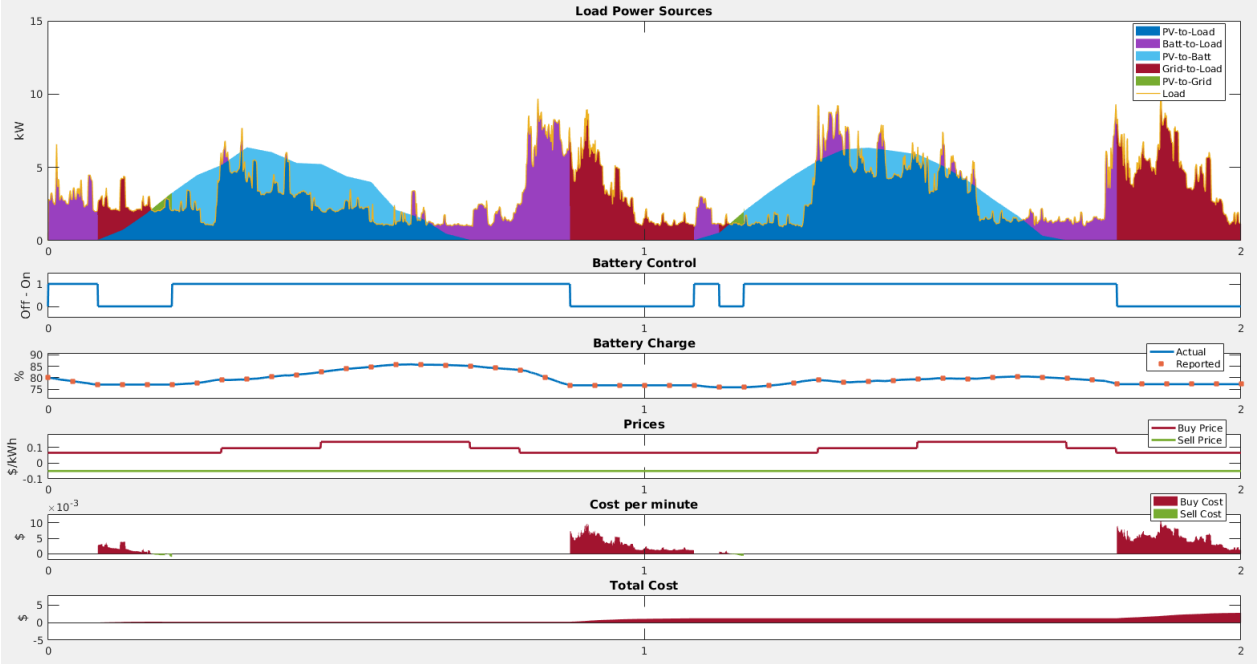

Figure 4.13 Scenario: 2-day real-world data, init. batt. SOC 80%, no attack (Cost: $2.74)

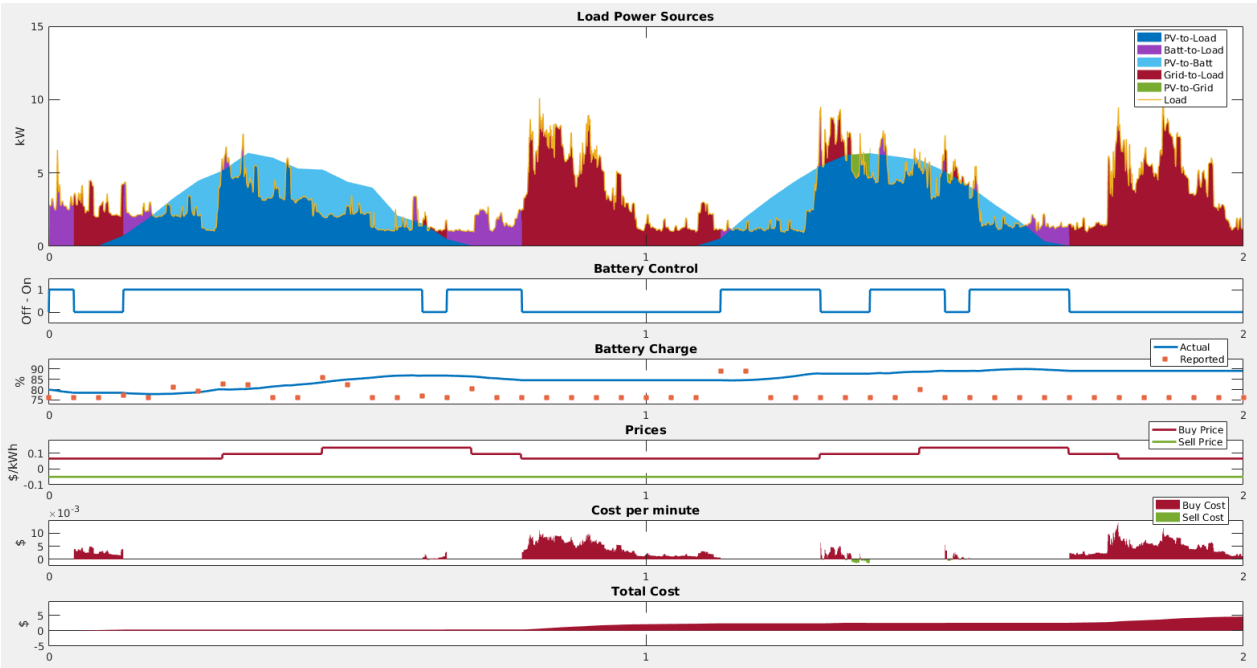

Figure 4.14 Scenario: 2-day real-world data, init. batt. SOC 80%, RL-attack agent with any value for the SOC (Cost: $4.55)

Figure 4.15 shows the same scenario, however, this time with the RL agent that can modify the battery charge by plus or minus 5%. The agent generally reports a battery charge as low as possible, causing the controller to believe that the reserve battery power is reduced. There is the same effect as in the previous attack of a rising battery charge and total cost, however, it is less drastic.

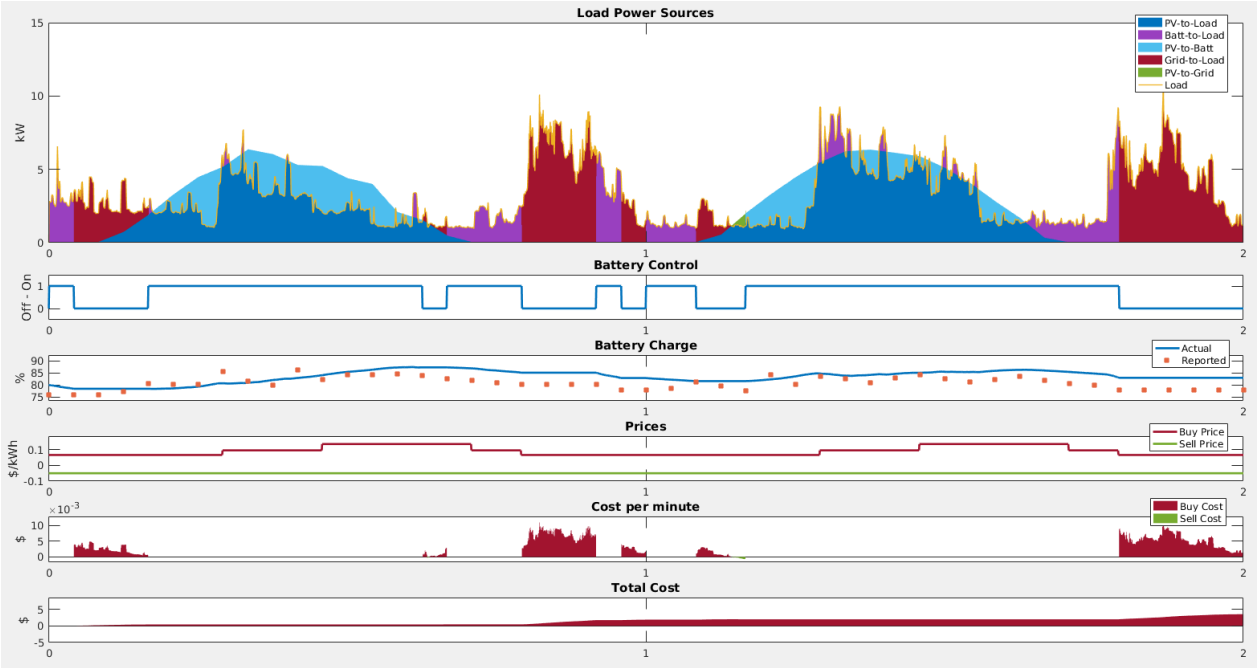

Figure 4.15 Scenario: 2-day real-world data, init. batt. SOC 80%, RL-attack agent with SOC modified by $+/-5$ (Cost: \$3.52)

A more detailed analysis of the behavior of the attacking agents is provided in Table 4.6. Attacks are carried out in the simulation over the 48-hour period with different initial starting battery SOC levels. The following attacks are compared: the RL attack agent which can modify the reported SOC by plus or minus 5 (referred to as the *SOC +/- 5* attack), the RL attack agent which can modify the reported SOC by any value (referred to as the *SOC +/- any* attack), and the *Max Cost* attack (from Section 4.3.2). For each simulation scenario the following values are provided; the cost, the cost increase from the performing the attack (as a percentage), the average battery SOC, and the average reported battery SOC to the controller. Each scenario is run 10 times and the values are averaged.

In general the RL attacking agents, *SOC +/- 5* and *SOC +/- any*, report as low of battery charge as possible. This causes the controller to not use its reserve battery power to supply the load and the overall battery charge increases. The *SOC +/- any* agent generally does

not cause as much as an impact as the *Max Cost* attack. This implies that attacks which modify the control invoked by this controller can have a more negative impact than attacks which modify the state values passed to the controller. The exception being for the cases where the initial battery SOC are 95% and 100%, which are probably less likely scenarios. In these cases the *Max Cost* attack cannot bring the battery SOC all the way to 100%, whereas the *SOC +/- any* attack is able to do so and ultimately is able to have a larger impact.

Table 4.6 Simulation results for 48-hour (2-day) period with varying starting battery SOC values and RL attacks

	Init. Batt. SOC: 75%				**Init. Batt. SOC: 80%**			
Attack	Cost ($)	Cost Incr.	Avg SOC	Avg SOC Rep.	Cost ($)	Cost Incr.	Avg SOC	Avg SOC Rep.
None	3.62	-	79.64%	79.62%	2.74	-	79.30%	79.30%
SOC +/- 5	3.95	+9.12%	80.80%	79.44%	3.41	+24.45%	82.21%	80.18%
SOC +/- any	4.13	+14.08%	80.47%	78.99%	3.83	+39.78%	83.08%	78.62%
Max Cost	5.18	+43.09%	82.85%	82.86%	5.18	+89.05%	86.98%	86.98%
	Init. Batt. SOC: 85%				**Init. Batt. SOC: 90%**			
Attack	Cost ($)	Cost Incr.	Avg SOC	Avg SOC Rep.	Cost ($)	Cost Incr.	Avg SOC	Avg SOC Rep.
None	2.25	-	82.58%	82.58%	1.87	-	83.94%	83.99%
SOC +/- 5	2.97	+32.00%	86.36%	82.22%	2.47	+32.09%	88.24%	83.79%
SOC +/- any	4.41	+96.00%	89.80%	77.20%	4.60	+145.99%	95.46%	76.39%
Max Cost	5.15	+128.89%	91.87%	91.87%	4.89	+161.50%	95.87%	95.86%
	Init. Batt. SOC: 95%				**Init. Batt. SOC: 100%**			
Attack	Cost ($)	Cost Incr.	Avg SOC	Avg SOC Rep.	Cost ($)	Cost Incr.	Avg SOC	Avg SOC Rep.
None	1.58	-	86.07%	86.14%	1.06	-	87.30%	87.39%
SOC +/- 5	1.74	+10.13%	88.29%	84.28%	1.45	+36.79%	90.08%	85.64%
SOC +/- any	4.65	+194.30%	100.00%	76.10%	4.75	+348.11%	100.00%	76.03%
Max Cost	4.28	+170.89%	98.13%	98.14%	4.04	+281.13%	98.88%	98.92%

4.4 Conclusion

This chapter provides an overview of MG concepts and cybersecurity concerns to motivate the need for research which uncovers vulnerabilities in not only MG devices, but MG control algorithms themselves. A custom MILP control algorithm was developed to perform control of a simulated MG in MATLAB/Simulink. Simulated FDI attacks which modify the battery's SOC reported to the controller and the control actions conducted by the control were used in the simulation to witness the behaviour of the control algorithm when faced with malicious input. It was found that state attacks which report a lower battery SOC and control attacks which increase the SOC of the battery have the most profound effect on the performance of the controller. Although the insights gained from these conducted experiments are only applicable to this particular MG control algorithm, the process presented in this chapter can be readily applied to more realistic MG controllers (or other ICS environments) to uncover

vulnerabilities in control algorithms. This type of attack analysis activity can be conducted in order to equip MG operators with the knowledge to prioritize the allocation of resources to deploy necessary defensive countermeasures.

Overcame Challenges

An instrumental part to this research was the development of the MILP control algorithm. This novel control algorithm required developing a unique mathematical formulation of the MG control problem and integrating diverse software technologies (i.e. MATLAB/Simulink, Python, CPLEX). The end result is an effective MG controller which is highly configurable using CPLEX code. While the controller is a simplification of what would exist in the real-world, the architecture of the testbed allows for the possibility to use more sophisticated MILP systems which can be more finely-tuned to this control problem.

The development of the simulated FDI attacks were also vital to this work and provided many challenges. The logic behind the custom attacks is relatively straightforward, however, implementing the attacks involved heavily modifying the MATLAB/Simulink model to communicate with custom Python scripts. The RL-based attacks provided an even larger hurdle as it required using Python scripts to act as a server on the local machine, to hold calculations in memory that are cleared between the simulated timesteps and episodes in the MATLAB/Simulink model (discussed in Appendix F).

Limitations

The primary limitation of this work is that the overall architecture of the simulated MG is quite small in terms of scale and realism, limiting the overall applicability of any gained insights. A more sophisticated MG would include a greater number of DERs, ESSs, and loads configured in a radial lateral configuration. Additionally, although the implemented control algorithm is novel and effective, it does not exactly replicate real-world control scenarios. One notable limitation of the control algorithm is that it does not split the allocation of quantities to more optimally reduce cost. Such as, when there is a shortfall of solar production and not enough battery reserves to meet the demand, it would make sense to use a portion of the battery power to meet the demand and purchase the remaining missing portion from the grid. The presented algorithm will instead satisfy the entire shortfall by purchasing power from the main grid.

Ideally, the analysis done in this chapter should be conducted using real MGCC software that is operating a more complex MG. However, there are some advantages gained from using

the simplified MG presented in this chapter, such as, the ease of interpretability of attack results and a more relaxed setting to develop attacks, particularly the RL-based attacks. The testbed presented in this chapter serves its goal of demonstrating the utility of performing attacks against an MGCC to uncover vulnerabilities in the underlying control algorithm.

Future Work

A primary theme to the future work would be to increase the realism of the testbed and the control algorithm. The size of the simulated microgrid can be increased (as discussed earlier), a simulated communications network can be incorporated, and real-world hardware devices (either physical or emulated) can be present to create a HIL effect. The attack analysis can also consider the effect of impact on the grid and go beyond only the considered economic impact. The MILP controller can also be improved so that it can split up the power quantities and also make decisions more frequently than every hour.

More work can be done to develop RL-based attacking agents which have more subtle (and less obvious) attacks. Currently, the RL-based attacking agents perform an attack at every hour in the simulation. However, the manually generated control attack which switches the control every 5 hours (presented in Section 4.3.2) demonstrates that the timing of attacks can have an impact as well. It would be interesting to train an agent which is limited in the number of attacks it can perform and determine if the agent can find opportune moments to execute an attacking action. This would help an MG operator understand at what times attacks would have the most impact and allow the operator to take the necessary defensive measures.

This chapter did not consider any defensive actions in response to the attacks. A stream of work could look at creating agents which could automatically learn defensive rules when faced with various attacks. To go even further, experiments could be conducted to explore how an attacking agent can learn new attacking strategies to circumvent defensive actions. This type of AI vs. AI experimentation could be used to potentially uncover the most elusive attacks and even better equip an MG operator with an understanding of its control vulnerabilities.

Experiments can also be conducted using dynamic electricity buying and selling prices which fluctuate according to market factors, as is the case with parts of Europe [166]. In this setting, the control actions taken by the MGCC would be less obvious and there is potentially more space for attacking agents to manoeuvre, while remaining undetected.

Lessons Learned

A significant hurdle to the work presented in this chapter was determining an appropriate testbed environment and research question. The organization funding this work laid out the requirement that there is a need to investigate cybersecurity challenges in electrical power systems, but without any other clear guidelines. Much effort was spent in attempting to gather resources to develop a testbed and there were many pivots as we investigated possibilities across a number of electrical power system domains, such as, transmission networks, smart grids, substations, and MGs. Ultimately, by identifying the research question of finding vulnerabilities in control algorithms, this work was able to establish the requirements for the testbed and the presented small-scale residential MG testbed was deemed appropriate. The useful development effort ultimately lied in constructing the presented MILP controller and not integrating physical/emulated equipment or communications networks.

The previous point leads into the next insight, that it can be acceptable for a testbed to simplify its representation of its real-world counterpart in order to generate meaningful experimental results. It would have been a bonus to replicate an MG to the point where experiments can simulate the actions taken by an attacker to infiltrate a network and compromise equipment to lay the groundwork for an attack against the control logic, but in the end it was not necessary.

When working with RL agents that require many episodes to learn an effective control strategy, it is advisable to work with simulated environments that can perform episodes very quickly. With the presented testbed, a single 7-day simulations takes about 3 minutes to execute and proved to be an obstacle in training the RL agents. Instead, the 2-day simulations were used, taking about 1 minute to execute, and drastically increased the speed of training the RL agents. The training of the presented RL-agents, in Figures 4.11 and 4.12, ultimately took about 16 hours to complete the 1000 episodes. There is indeed some effort that could be spent in tuning the agents to learn at a quicker rate, but execution of faster episodes undoubtedly aids in the training process. Perhaps there could even have been more simplifications to the simulated environment, by not using MATLAB/Simulink for instance, to further increase the execution of episodic simulations and provide more time to experiment with iterating over the development of attacking agents.

CHAPTER 5 ONTOLOGY-DRIVEN ANOMALY DETECTION OF MALICIOUS ADS-B MESSAGES IN AIR TRAFFIC CONTROL SYSTEMS

The ADS-B protocol is increasingly being adopted by the aviation industry as a method for aircraft to relay their position to ATC monitoring systems. ADS-B is a digital based communication protocol which provides greater precision compared to traditional radar-based technologies, however, it was designed without any encryption or authentication mechanisms and has been shown to be susceptible to spoofing attacks. A capable attacker can transmit falsified ADS-B messages using Software Defined Radio (SDR) devices causing falsified aircraft to be shown on ATC displays and potentially threatening the safety of air traffic. Updating the ADS-B protocol will be a lengthy process, therefore there is a need for systems to detect anomalous ADS-B communications. This chapter presents an ATC testbed which can simulate aircraft traffic in a controlled airspace, as well as the injection of falsified aircraft on ATC displays from ADS-B spoofing attacks. The testbed is used to develop ontology-based anomaly detection rules which detect falsified ADS-B messages. The presented testbed is part of an ongoing project and the contributions made by this research are highlighted in the chapter.

This chapter is organized as follows; Section 5.1 provides an overview of ATC reporting technologies and the security concerns related to the ADS-B protocol, Section 5.2 presents the architecture of a simulated ATC testbed and the details of the ontology-based anomaly detection approach, Section 5.3 demonstrates a computational performance analysis of the ontology-based detection approach, and Section 5.4 concludes this chapter by summarizing the completed work and describing the insights gained from this case study.

5.1 Air Traffic Control Systems and Cybersecurity Concepts

ATC is a service provided to ensure the flow of air traffic in a controlled airspace occurs in a safe and efficient manner [167]. An air traffic controller (or simply *controller*) has the goal of ensuring separation between aircraft operating in an airspace. The controller provides instructions to pilots over radio communications to do things such as change an aircraft's speed, altitude, and direction.

As aircraft navigate through a controlled airspace, several types of surveillance technologies are used to identify individual aircraft and generate position reports. This information is used to populate the displays used by controllers to see the position of aircraft in real-time.

Traditionally this has been done using Primary Surveillance Radar (PSR) and Secondary Surveillance Radar (SSR) which broadcast signals over a geographic area and receive a response from aircraft within their range. This is gradually being upgraded with ADS-B systems which can more precisely identify aircraft and their properties. In the ADS-B paradigm, an aircraft determines its position using a Global Positioning System (GPS) and broadcasts its position, speed, altitude, and flight number using digital ADS-B communication packets to ADS-B antenna receivers. ADS-B is envisioned to be a central component of air traffic surveillance modernization projects across the globe and is mandatory in the United States as of 2020 as part of the Next Generation Air Transportation System [168]. ADS-B is also currently being used in regions which traditionally had no radar coverage, such as the Canadian Arctic [169].

However, the ADS-B protocol was developed with the focus on performance and not on security. By design, there are no mechanisms for authentication or encryption. This has not been a problem until the early 2000s with the availability of low-cost and performant tools, such as SDR devices, which can be used to broadcast falsified signals to ADS-B antennas [89]. An SDR device can perform complex signal processing tasks using a general-purpose computer processor, instead of requiring specialized hardware components not historically available to the general public. The feasibility of attacks which target the ADS-B communication protocol has been shown to be theoretically attainable by both academic [90] and hacking communities [91].

ADS-B is envisioned to aid in the management of increasingly crowded air spaces. In 2018 there were a total of 46.1 million flights servicing 4.8 billion people, with industry forecasts expecting this number to double by 2030 [89,170]. As air traffic continues to grow, the strain on air traffic controllers is becoming a growing concern. An individual controller can only track a certain number of flights at a given time, if there is not enough ATC capacity for a given flight plan, an airline must choose to delay its flight or choose a different route through less congested airspace. In 2018, the Eurocontrol flight authority reported that over 60% of all en-route air-traffic delays were attributed to lack of ATC capacity, 25% was related to inclement weather, and 14% were due to controller personnel on strike [171]. An air traffic controller has a demanding workload in a complex environment, where erroneous actions can have devastating consequences. It is crucial that the information they receive to guide aircraft is reliable and it is therefore important to analyze how to protect this information from being attacked.

5.1.1 Overview of Air Traffic Control Systems

Controllers

Controllers are located at ground stations and communicate with pilots during all phases of a scheduled flight. This includes the taxi-out, take-off, landing, and taxi-in, as well as when an aircraft enters, traverses, or leaves an airspace. A controller can identify aircraft by their squawk code (a four digit number send by an aircraft's transponder), the call sign, and the flight number. When an aircraft leaves an airspace, a controller hands the aircraft off to a controller in the adjacent airspace. For example, on a typical flight between London and Frankfurt, approximately a dozen controllers will manage a single aircraft [172]. The airspace sectors follow political boundaries and there is generally a distinction between military airspace sectors as well as civilian airspace sectors. All aircraft have an assigned type and can only operate in classes of controlled airspace in which they are authorized.

Flight Plan

Prior to a flight, a pilot must submit a flight plan to all the agencies which control an airspace wherein the aircraft will travel. A flight plan contains information such as the origin airport, destination airport, a planned route, alternate routes, aircraft information, aircraft type, etc. A flight plan is generally required by aviation authorities for commercial flights. Additionally, when an aircraft enters a controlled airspace, a flight strip is created and assigned to a controller to track the flight.

Position Reporters

As aircraft travel through an airspace, timestamped position reports are generated by a position reporter which identify the latitude, longitude, angle, and potentially other information about the aircraft. The list of reported positions for an individual aircraft constitute the flight's track. These are generally generated by ground-based stations with a limited coverage area. This chapter focuses on PSR, SSR, and ADS-B position reporters.

Primary Surveillance Radar (PSR)

PSR is a radar technology with its history tracing back to World Ware II. PSR works by echoing signals off of physical devices and does not require any specialized equipment. These types of signals cannot be falsified since they work on physical properties, however, this technology has several key limitations. PSR can identify the longitude and latitude of aircraft,

however, it cannot determine aircraft altitude. Additionally, PSR is susceptible to noise as any airborne entity can return a PSR signal and there is no way to uniquely identify an entity within range of a PSR station. PSR is still used in ATC systems today, however, in a complementary nature to more sophisticated technologies as explained below.

Secondary Surveillance Radar (SSR)

To overcome the shortcomings of PSR, SSR was developed to uniquely identify aircraft through the use of a *transponder*. Transponders are electronic devices which provide a coded signal in response to an interrogating device. Commercial airliners are equipped with a transponder device to aid in generating position reports and to provide other aviation information to controllers. A transponder transmits an aircraft's position, altitude, and identity. SSRs can be located at their own dedicated groundstation, however, they are often mounted on top of PSR installations (shown in Figure 5.1, reproduced from [173]). SSR signals have the potential to be spoofed by an attacker, since an attacker can send false commands to an aircraft's transponder or send falsified transponder response signals to an SSR groundstation. However, since SSR devices initiate the broadcast-response it is extremely difficult for an adversary to carry out a viable spoofing attack and is not the focus of this chapter. The difficulty lies in the requirement of continuously spoofing the handshake-like operation in accordance with the physics of the SSR groundstation capabilities and is not considered in this work.

Automatic Dependent Surveillance-Broadcast (ADS-B)

ADS-B is a more recent development and is gaining in adoption as it offers more accurate position reporting through the use of GPS. The idea is that an aircraft can determine its position via GPS and broadcast it to ground ADS-B antennas as well as other aircraft over SSR-type communications using an aircraft's transponder. ADS-B has several advantages as it provides more precise tracking of aircraft and it will eventually provide the ability for pilots to see the location of ADS-B enabled aircraft on a visual display similar to what an air traffic controller has access to. However, this technology was developed with a focus on performance and not security. There is the possibility for attacks which spoof ADS-B signals to ground-based antennas [90, 91].

The core concepts of PSR, SSR, and ADS-B technologies are summarized in Table 5.1 and visualized in Figure 5.2.

Figure 5.1 Radar antenna on the Deister river close to Hanover, Germany. The parabolic antenna is the primary radar and the rectangular one above it is the secondary radar. The secondary radar interrogates the transponder on the aircraft, which sends back a coded signal identifying the craft and giving its altitude (reproduced from [173]).

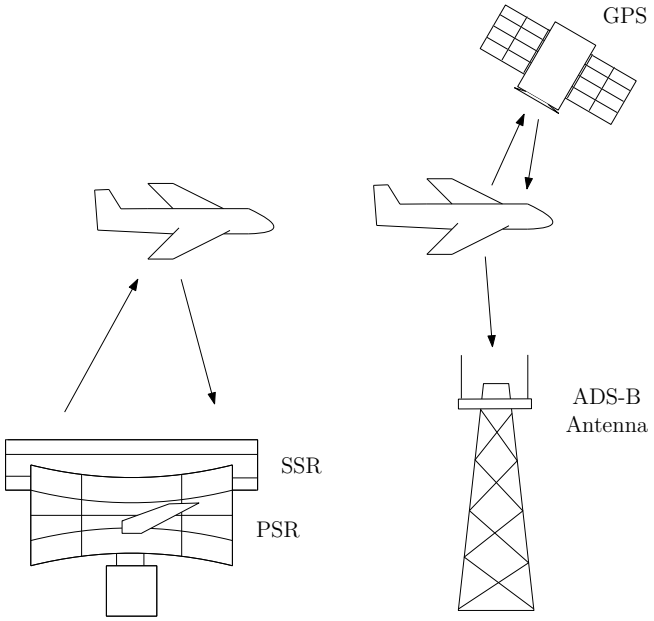

Figure 5.2 PSR, SSR, and ADS-B reporting principles

Data Distribution Service (DDS) Network

With data being generated by a number of different sources, there is the logistical task of ensuring this data is provided to the proper end-systems. One solution is the Data Distribution

Table 5.1 Overview of PSR, SSR, and ADS-B

ATC Reporting Technologies			
Tech.	**Reporting Principle**	**Information Captured**	**Security Concerns**
PSR	Groundstation broadcasts electromagnetic waves and judges reflection off of physical objects	Aircraft distance, angle, and radial velocity	Cannot be falsified, susceptible to physical noise
SSR	Groundstation broadcasts radio interrogations and receives encoded radio response from aircraft transponder	Aircraft position, altitude, and identity	Susceptible to eavesdropping and falsification (higher degree of difficulty)
ADS-B	Aircraft broadcasts GPS-based position from transponder to ADS-B antenna	Aircraft position, altitude (more frequently and accurately), identity, ground speed, and flight information.	Susceptible to eavesdropping and falsification

Service (DDS) architecutre, which is a middleware connectivity framework standard operating on the publish-subscribe pattern [174]. In this framework, data sources (publishers) broadcast information to systems interested in this data (subscribers), following the DDS standard. This is currently being used in a number of sectors including ATC, railways, autonomous vehicles, smart grids, nuclear reactors, mining operations, industrial automation, medical devices, aerospace, and the military [175,176]. The testbed presented in this chapter uses the DDS framework to aggregate all the PSR, SSR, and ADS-B data into a centralized network of standardized DDS packets. While it has been shown that a DDS network i susceptible to attack [177], this work does not consider this attack vector and assumes all DDS information to be accurate.

Security Response

The availability of SDR technologies allows a viable attack vector for injecting falsified ADS-B packets over ATC communications channels. This is possible because of the unsecured nature of the ADS-B protocol and the current lack of security certification criteria. Aircraft operate over large geographical areas and ATC systems utilize information from increasing complex sources. The result is a high exposure to potential threats. As the industry is slow-moving to address these issues, there is a demand for near-term defensive measures which do not require drastic alteration to current infrastructure and protocols. This chapter demonstrates

an anomaly detection framework based on ontologies which leverages the cyber-physical properties of aircraft to infer the presence of falsified input.

5.1.2 ADS-B Anomaly Detection

Prior to demonstrating the ontology-based ADS-B anomaly detection approach proposed in this work, an overview of previous work in this space is provided.

One stream of research for preventing the transmission of falsified ADS-B packets has looked at securing the protocols through the use of encryption and authentication methods [178–182]. These approaches either require modifying the current specification of the ADS-B protocol or require additional equipment to be deployed in ATC infrastructure to certify the authenticity of ADS-B messages. Coordinating such schemes across the aviation industry is a daunting task and is not a near-term solution.

Non-cryptographic methods for detecting anomalous ADS-B packets aim to verify that the physical properties of received ADS-B messages are in-line with that of actual aircraft [183–186]. These types of approaches verify that the Doppler shift measurements or the Received Signal Strength (RSS) of transmitted ADS-B packets match that of legitimate aircraft. These approaches do not require modifications to the ADS-B protocol, however, does assume that the attacker does not have the ability to send falsified ADS-B messages which completely resemble those of legitimate aircraft.

Advances in ML have prompted researchers to investigate how to detect anomalous ADS-B messages which fall outside of established statistical bounds. Deep Neural Networks (DNNs) have been proposed to detect falsified ADS-B messages and falsified aircraft by training on labelled datasets of legitimate and malicious messages [187]. The use of Long Short-Term Memory (LSTM) neural networks have been proposed to learn trajectories of typical flights and trigger an alarm when a given flight path deviates from its normal path [188]. These approaches show the feasibility of detecting falsified ADS-B messages through ML pattern matching, however, these approaches require that the captured samples cover enough statistical variety to prevent overfitting to particular patterns, which is difficult to collect in cybersecurity domains. Additionally, an open problem with ML is that there lacks explainability when a trained model makes a decision in a particular domain. If an ML system flags an ADS-B message as anomalous, there is no explanation as to why it was anomalous.

Ontologies on the other hand, have the knowledge of the domain built in and an anomaly is detected because an explainable rule is violated. For example, an anomalous ADS-B message will not have a corresponding PSR/SSR track. Using ontologies to detect anomalies in the

ATC domain has been previously explored using static datasets containing malicious ADS-B packets [189, 190].

5.1.3 Ontologies for Attack Detection

Ontologies have been used to model hierarchies of concepts in both academia and industry in a range of fields including cybersecurity [191, 192] and aviation [193]. A key characteristic of ontologies is that all data is stored as nodes in a graph, where directed edges between nodes capture their relationship (see Appendix G for more details about ontologies). Researchers have proposed to leverage this property for intrusion detection tasks [194–196]. These previous works demonstrate how attacks can be detected using ontologies in traditional IT environments (e.g. computer networks). The premise being that as low-level alerts are generated from different sensors, such as an IDS or a SIEM tool, they can be translated into ontological instances and be stored in an ontological database. The ontological database is then queried at a regular interval to look for violations to normal operating criteria. The queries can be seen as a mechanism for aggregating the meaning of low-level alarms into more semantically rich detection rules. These rules put the occurrence of low-level alarms into some established context and is known as alert correlation.

The most popular ontology language is the Web Ontology Language (OWL) [197], which utilizes data written in the Resource Description Framework (RDF) format [198]. RDF data is stored in an ontological (RDF) database and is queried using the SPARQL Protocol and RDF Query Language (SPARQL) language. Unlike relational databases, RDF databases are schema-less and allow a higher degree of flexibility when assigning properties to entities. In RDF, each statement is represented as a *triple* consisting of a subject, a predicate, and an object. Thus, RDF triples can be conceptualized as a node and arc labelled graph. Each element of a triple is identifiable by a Uniform Resource Identifier (URI). The schema defining the ontological classes and the instantiations of these classes appear in the same repository as RDF triples. There are different formats for representing RDF data such as RDF/XML, Turtle, JSON, and N-Triples. Consider the example RDF/XML in Listing 5.1, which contains two data packets captured over some communication protocol (e.g. TCP/IP). Its corresponding directed graph is shown in Figure 5.3.

For this example, suppose that there is a sensor reporting values to another system on a network. The only host ID connections should be that of the whitelisted sensor device of `10.10.100.9`, while all other connections would be anomalous and should be flagged. The SPARQL query in Listing 5.2 selects the packet ID of the packets where the host ID is not in the whitelist. The results of the query are in Table 5.2.

Listing 5.1 Example RDF/XML

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl/#"
     xmlns:ns0="http://www.example.com#">

<owl:NamedIndividual rdf:about="http://www.example.com#P111">
     <rdf:type rdf:resource="http://www.example.com#Packet"/>
     <ns0:hasBody rdf:resource="http://www.example.com#B111"/>
     <ns0:hasHostID>100.10.100.9</ns0:hasHostID>
     <ns0:hasID>111</ns0:hasID>
     <ns0:hasTime>1574969411</ns0:hasTime>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.example.com#P222">
     <rdf:type rdf:resource="http://www.example.com#Packet"/>
     <ns0:hasBody rdf:resource="http://www.example.com#B222"/>
     <ns0:hasHostID>100.12.100.9</ns0:hasHostID>
     <ns0:hasID>222</ns0:hasID>
     <ns0:hasTime>1547949461</ns0:hasTime>
</owl:NamedIndividual>
</rdf:RDF>
```

Listing 5.2 Select packets with an anomalous host ID

```
PREFIX ex:  <http://www.example.com#>
SELECT ?packetId ?packetTime WHERE {
  ?p  a ex:Packet ;
      ex:hasHostID  ?hostId;
      ex:hasID      ?packetId;
      ex:hasTime    ?packetTime.
  FILTER (?hostId !="100.10.100.9").
}
```

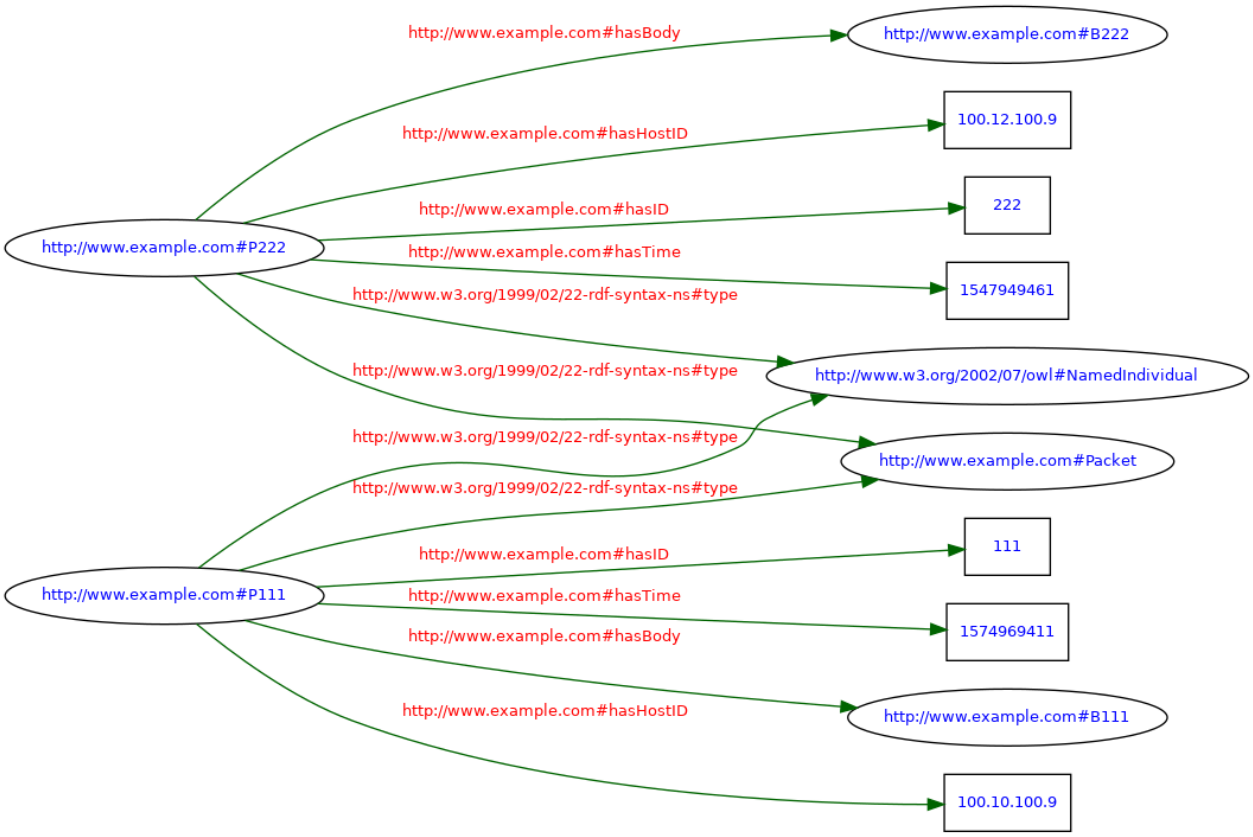

Figure 5.3 Directed graph of example RDF triples

Table 5.2 Example query results

packetId	packetTime
222	1547949461

The advantage gained from detecting attacks through an ontological-based alert correlation framework is that there is greater explainability as to why an anomaly was flagged, since the relationship between low-level alarms is captured. Describing knowledge domains with ontologies goes beyond the capability of taxonomies as it defines the relationships between entities, allowing all information within the ontology to form a graph structure and be reasoned upon [199].

In a security setting, an ontology-based anomaly detection approach may have several benefits over ML. In ML, an anomaly detection agent learns a decision boundary based on the distribution of samples it has trained upon and is susceptible to generating false positives

when it is provided unexpected samples to classify. However, by describing detection rules with ontologies, the attack space can potentially have more coverage with explainable rules.

It was stated that one of the benefits of ontologies is the graph-based nature of capturing data. This work has opted for ontologies based on the RDF standard, however, other graph data storage platforms exist, both open source and proprietary. The advantage of using RDF based ontologies is that they can easily be extended with other ontologies and that it is a W3C standardized data format.

5.1.4 Threat Model

Prior to the 2000s the general public had very limited abilities for interfering with avionics communications. Adversaries were limited to militaries or nation-states with electronic warfare capabilities. Attacks were limited to analog technologies, which are easier to detect. Also, the level of insider knowledge related to communications systems and aviation conduct was not readily available to the public [89].

However, as there is a shift towards digital technologies and automated processes, there is a much larger attack surface. The financial barrier to access SDR technologies is relatively low with competent transceivers starting around $150 USD. Additionally, access to open-source software which can process avionic communications can be readily downloaded. The availability of aviation knowledge is also easily accessed over the Internet. Protocol specifications can be accessed, plane-tracking websites exist, flight plans are made public, and individuals can even capture real-world ADS-B communications with home-made antennas.

This work considers an attacker which employs an SDR device with spoofing capabilities. The attacker has the capability to send spoofed ADS-B messages to a targeted ADS-B antenna receiver within the range of the SDR device. It is assumed the attacker has a complete understanding of the ADS-B protocol, knowledge of reporter groundstation locations, and access to real-world flight plans. The SDR based attack is outlined in Figure 5.4. The effect of the attack on the ATC display is provided in Figure 5.5.

5.2 Air Traffic Control Testbed

This section presents an ATC cybersecurity testbed and an ADS-B anomaly detection platform, known as ATC-Sense. This section provides a high-level depiction of the testbed's architecture, a description of the testbed's implementation details, and the logic behind the ontology-based detection platform.

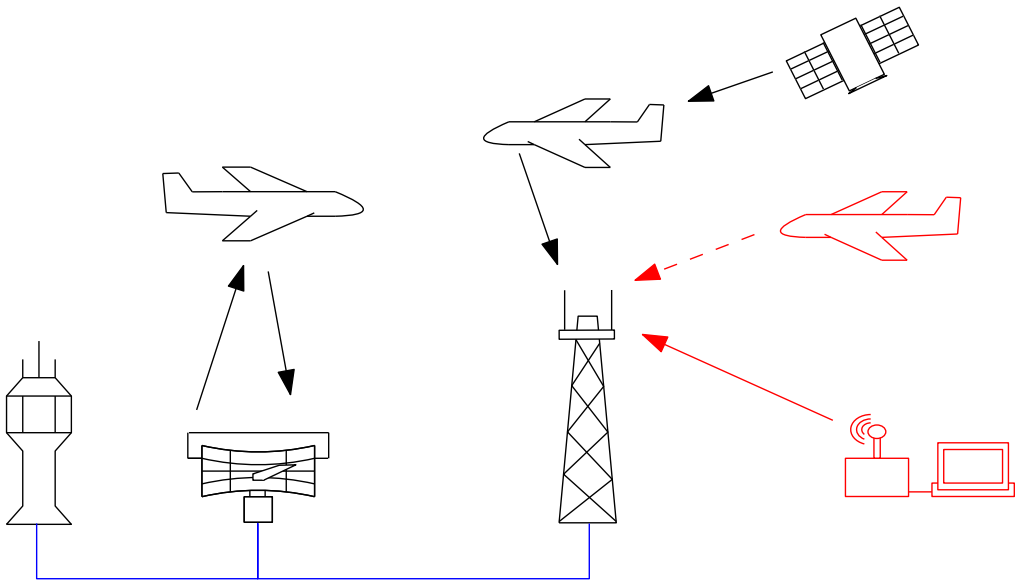

Figure 5.4 ATC Adversary Model. An attacker uses an SDR device to send falsified ADS-B messages to an ADS-B antenna. The reporters communicate with the ATC tower over a DDS network. Falsified aircraft appear to exist to ATC personnel.

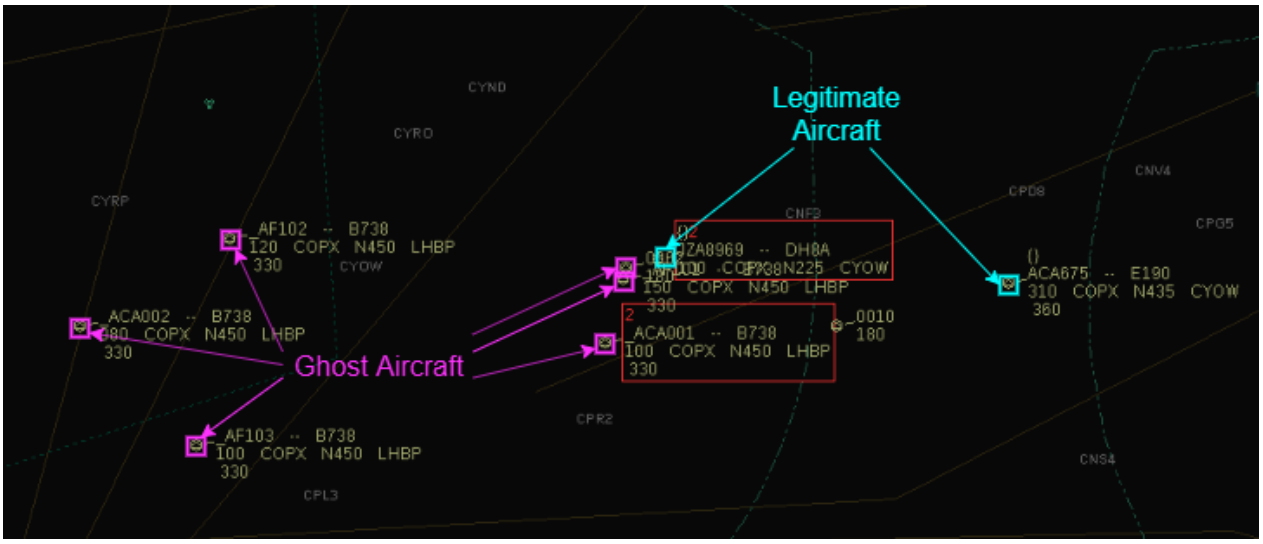

Figure 5.5 ATC display with multiple falsified (ghost) aircraft

This testbed and detection approach has been developed by a number of personnel over several years and the work conducted in this research has developed a subset of the overall testbed. The primary contribution of this section is the synthesis of the constructed architecture, which provides a backdrop for the rest of the chapter and for grading the testbed with the taxonomy proposed in Chapter 6.

5.2.1 Testbed Architecture

The testbed provides a simulated ATC environment where spoofed ADS-B messages can be injected and detected using ATC-Sense. The overall architecture of the testbed is summarized in Figure 5.6 and described below.

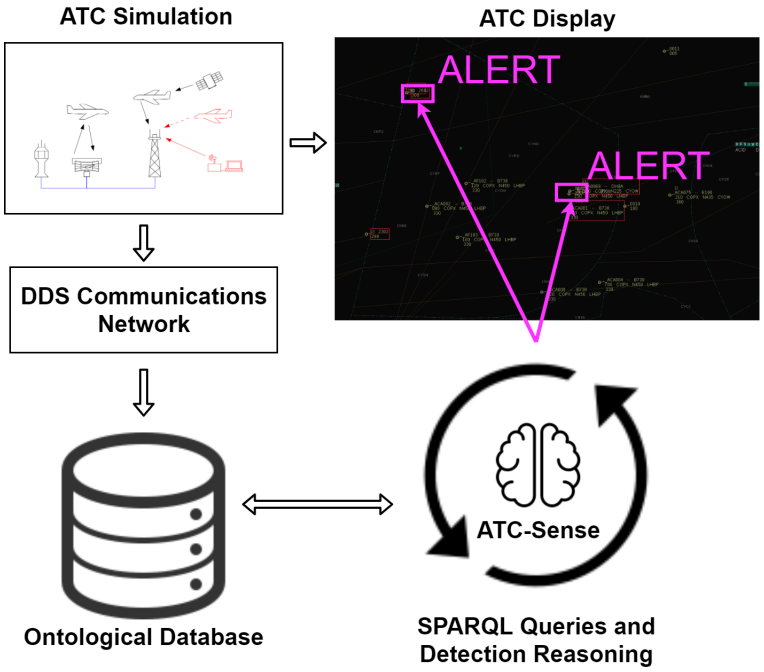

Figure 5.6 Overview of Testbed Architecture

At the core of the ATC testbed is a freely available ATC simulation software called Euroscope [200]. Prior to conducting an ATC simulation, data is loaded into Euroscope which includes airspace sector maps, flight plans, flight strips, aircraft physical properties, flight instruments, radar positions, airport classes, airport locations, and airport altitudes. This data is used as the parameters loaded into the simulation software to constitute a simulation scenario.

As Euroscope operates, it automatically coordinates the aircraft entities by generating and issuing ATC commands. As aircraft traverse the airspace, simulated radar and antenna groundstations receive responses from the aircraft within their pre-configured range. Attacks are generated by injecting falsified messages into the simulation, as if they were falsified ADS-B messages sent to simulated ADS-B antennas, causing falsified aircraft to appear in Euroscope.

All of the PSR, SSR, and ADS-B information generated in the simulation is converted into DDS messages and are transmitted over the DDS network. The DDS network messages are

converted to RDF format and are inserted into the Ontological Database. As raw RDF data enters the database, ontological instances are created from pre-defined entity concepts. ATC-Sense determines the presence of anomalous ADS-B messages by querying the database using SPARQL queries and performs reasoning on the retrieved entities. An alert is generated if an aviation constraint rule is violated. The alert is meant to provide a controller with an indication of which information being displayed cannot be trusted, or even to an ATC system administrator that there is an active attack.

5.2.2 Testbed Implementation Details

ATC-Sense is developed with the objective of detecting anomalous ADS-B messages in an operational ATC system. To achieve this, a data-pipeline has been built to convert simulated ATC communications into RDF triples. An overview of the implementation is provided in Figure 5.7.

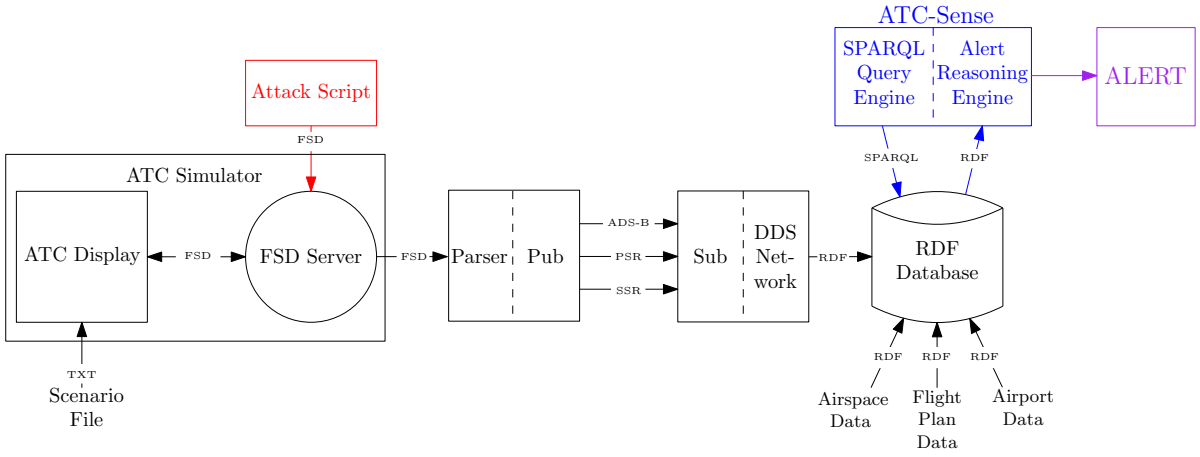

Figure 5.7 Block Diagram of ATC Testbed Components

The **ATC Simulator** is a freely available Windows application called *Euroscope*, which simulates the transmission of communications packets between aircraft and groundstations, issues automated controller commands to simulated aircraft, and provides a graphical display to observe the scenario from the viewpoint of a human controller. Euroscope is the most popular ATC client software used by the Virtual Air Traffic Simulation Network (VATSIM), an online community of over 79,000 active members of hobbyist virtual pilots and air traffic controllers which conduct real-time virtual flights between real-world airports [201].

Euroscope is comprised of two central components, the **ATC Display** and the **FSD Server**. Note that Euroscope does not actually utilize real PSR, SSR, and ADS-B messages (or other communications protocols for that matter), instead it coordinates entities using its

proprietary Flight Simulation Data (FSD) protocol. Euroscope relays FSD Packets between the FSD Server, performing the commands of the ATC simulation, and the ATC Display which displays the events occurring in the simulation (e.g. aircraft positions, flight paths, squawk codes, etc.).

A **Scenario File** is loaded into the ATC Display, prior to executing a simulation, to import airport features, physical barriers, and aircraft positions into the simulation. In Euroscope's *simulation mode*, the FSD server will automatically issue ATC commands to the aircraft and guide them to the airport based on an internal control algorithm. The **Attack Script** is a custom Python script which sends malicious packets to the FSD Server which cause falsified aircraft to appear on the ATC Display.

In real-world ATC systems FSD packets are not used between groundstations and the aircraft, instead data is transferred using radar signals (PSR, SSR) and ADS-B packets. The testbed has been configured so that the FSD Server sends all the FSD packets generated within Euroscope to a custom **Parser** program written in C, to convert the FSD packets into messages which contain the corresponding data of PSR, SSR, and ADS-B messages. Following the Pub-Sub pattern, all the generated PSR, SSR, and ADS-B messages are published by the **Pub** program to the **Sub** component, which then are made available on the **DDS Network**, all of which is done using Python scripts. This method allows for the generation of DDS Network traffic, as if it were generated by real-world PSR, SSR, and ADS-B sources.

As the DDS Network receives the simulated PSR, SSR, and ADS-B messages, a Python script is used convert the data into RDF triples and passes the RDF data to the **RDF Database** (i.e Ontological Database). The database tool used is a World Wide Web Consortium (W3C) compliant RDF triplesore called *GraphDB* [202], which has a base free-to-use license. GraphDB operates as a web server which can insert RDF triples and perform SPARQL queries through a Representational State Transfer (REST) Application Programming Interface (API). Prior to executing a simulation, the RDF Database is populated with some static *base information* which includes **Airspace**, **Flight Plan**, and **Airport Data**. This base information is used in conjunction with the real-time data coming from the DDS Network in the reasoning process to infer the presence of malicious ADS-B packets.

ATC-Sense is used to determine the presence of anomalous ADS-B packets and is comprised of the **SPARQL Query Engine** and the **Alert Reasoning Engine**. The SPARQL Query Engine is a series of Python scripts which performs SPARQL queries to the RDF Database at a regular interval, over a REST API, to retrieve ATC related entities which exist in the database. The Alert Reasoning Engine is a Python script which performs if-else type logic

on the retrieved RDF data to infer the presence of anomalous packets. An **Alert** is provided if an anomaly is detected. The alert indicates which packet is anomalous, along with its reported coordinates. This information could be used to provide a visual alert to a controller on a ATC display screen. This entire architecture has been implemented to execute on a single 64-bit Kali Linux machine running on an Intel Core i7-3770 processor with 4 cores of 3.40GHz.

5.2.3 ATC-Sense: An Ontology-Based Platform for Detecting Anomalous ADS-B Messages

Overview of Detection Constraints

ATC-Sense is designed to be a situation-aware defensive solution which infers the presence of malicious input when the received messages do not follow the rules of aviation constraints. During an ATC simulation all the received PSR, SSR, and ADS-B messages are converted into RDF format and are inserted into the ontological database. To support the reasoning process, additional base information is loaded into the database. This information includes airspace definitions, flight plans, radar positions, airport locations, and runway capacities. The ontological database is continually queried at a regular interval to analyze whether received messages are anomalous. At the time of writing the proposed ontological detection framework performs queries and reasoning to search for violations to three main streams of aviation constraints logic; 1) Track Constraints, 2) Radar Constraints, and 3) Flight Constrains. A series of constraints that have been identified to aid in the detection of anomalous ADS-B messages are explained below.

1. **Track Constraints:** The series of reported aircraft positions constitute a *track*. These constraints ensures that the list of reported positions is coherent between the origin and destination.

    (a) *Consistent Origin*: The first position of the track corresponds to the location of an airport (altitude, coordinates) or at the limit of the covered area.

    (b) *Consistent Destination*: The last position of the track corresponds to the location of an airport (altitude, coordinates) or at the limit of the covered area.

    (c) *Consistent Take-off*: When the first position corresponds to the location of an airport, the type of aircraft must be authorized for that airport.

    (d) *Consistent Landing*: When the last position corresponds to the location of an airport, the type of aircraft must be authorized for that airport.

2. **Radar Constraints:** Over the course of a flight multiple devices (PSR, SSR, ADS-B) are used to report the position of an aircraft. These constraints verify that the reported positions are consistent across all devices.

   (a) *Reporter Consistency*: The position of the last message received is close to that of the last report sent. For SSR and ADS-B, the longitude, latitude and altitude are considered. For PSR, longitude and latitude are considered.

   (b) *Temporal & Physical Consistency*: The position of the last received ADS-B message is reasonable since the last recorded position given the flight time and the type of aircraft used.

3. **Flight Constraints:** A commercial flight must provide a valid flight plan which provides the intended path the aircraft will take. These constraints ensure that a reported aircraft has a legitimate path and flight properties.

   (a) *Consistent Instrumentation*: The device uses the type of navigation instruments corresponding to that authorized by the air sector

   (b) *Existence of Flight Plan*: The aircraft has a flight plan.

   (c) *Consistency with Flight Plan*: The position of the aircraft is on the route defined by the flight plan.

From these identified constraints, a subset has been implemented using SPARQL queries and are explained in more detail in what follows. The SPARQL queries utilize a previously proposed ATC ontology [190].

Detection Logic of Track Constraint: (1.a) Consistent Origin

The track is the series of reported positions associated to an aircraft and is comprised of timestamped PSR, SSR, and ADS-B reports. Static ghost plane injection attacks are used to demonstrate the detection procedure. This attack involves sending fake ADS-B messages to an ADS-B antenna to cause one or several non-existent (ghost) aircraft to appear on the ATC display in a fixed position. The appearance of the ghost aircraft may go unnoticed to an overwhelmed controller and cause the controller to issue incorrect commands to pilots. Also, multiple ghost planes could completely flood the screen and severely harm the controller's abilities. An outline of the attack is provided in Figure 5.8.

Detection logic verifies if all newly created tracks have a logical starting point. The SPARQL query in Listing 5.3 is used to select the first ADS-B position report of each newly created track.

Listing 5.3 Select First ADS-B Position Report of the Track

```
PREFIX atc-adsb: </atc/dds-topics/adsb-broadcast#>
PREFIX atc-core: </atc/atc-core#>
PREFIX atc-data: </atc/atc-data#>
SELECT ?report ?lat ?long ?alt ?eID ?call WHERE {
{?report  a  atc-adsb:ADSBFlightPosition;
          atc-core:hasTrackRank   ?rank;
          atc-adsb:hasLatitude    ?lat;
          atc-adsb:hasCallsign    ?call;
          atc-adsb:hasLongitude   ?long;
          atc-adsb:hasAltitude    ?alt;
          atc-adsb:hasEquipmentID ?eID.}
FILTER(?rank=1)
}
```

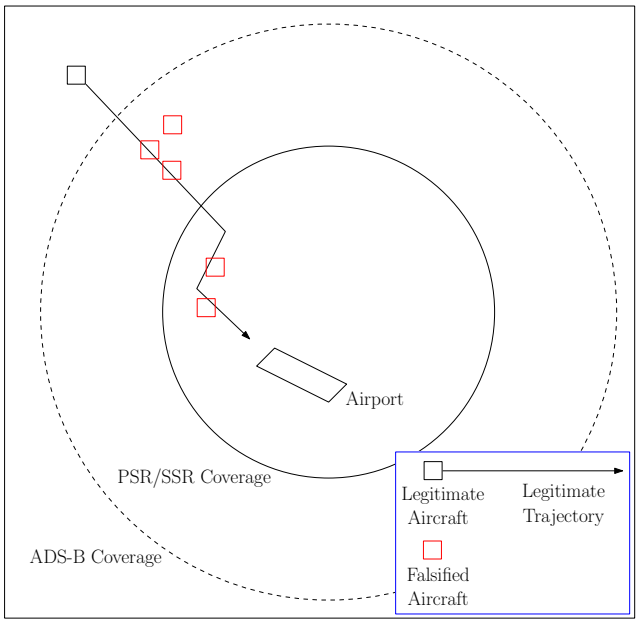

Figure 5.8 Ghost aircraft are displayed along the trajectory of an actual aircraft approaching an airport

This SPARQL query returns all of the ADS-B Flight Position Reports where the rank is equal to 1. The rank is used to determine the order of processed position reports for a given aircraft. The returned ADS-B Flight Position Reports have their associated latitude and longitude coordinates, thus constitute a point. The distance of this ADS-B point between nearby airports and the border of the ADS-B coverage area is calculated. If this report did not appear next to the border of an ADS-B coverage range or near an airport, then the

associated ADS-B packet can be flagged as anomalous along with its coordinate position. Note that, SPARQL is not optimized to perform certain mathematical operations, such as calculating distance between different points, therefore Python scripts are used to execute the SPARQL queries which retrieve the ADS-B position reports, then calculate the distances using Python math functions.

Detection Logic of Radar Constraint: (2.a) Reporter Consistency

This constraint detection logic uses SPARQL queries to determine if the reported ADS-B positions are associated with valid PSR radar positions. An attacker can falsify ADS-B messages, however, they cannot falsify PSR readings which operate according to physical properties. Consider an attack where a ghost aircraft enters the ADS-B coverage area from a legitimate location and then the ghost aircraft proceeds to travel towards PSR/SSR covered airspace to interfere with the trajectory of another aircraft. This attack is visualized in Figure 5.9.

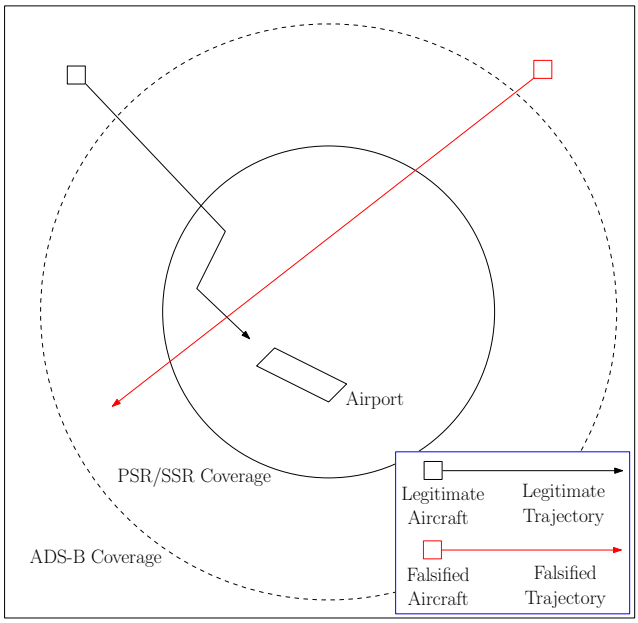

Figure 5.9 Ghost aircraft enters ADS-B coverage range and then enters PSR coverage range

ATC-Sense can detect anomalous ADS-B packets by verifying that all received ADS-B packets within the PSR or SSR coverage range have an associated PSR or SSR track. A linear interpolation between the trajectories of the received PSR and SSR tracks is done with the ADS-B tracks to match similar tracks. If an ADS-B track is not associated to a PSR or SSR track, then it is flagged as anomalous. The SPARQL query in Listing 5.4 selects all

the ADS-B Flight Position Reports which are not associated to any PSR or SSR track. The selected ADS-B Flight Position Reports are flagged as anomalous if they appear within range of a PSR or SSR radar.

Detection Logic of Flight Constraint: (3.b) Existence of Flight Plan

Suppose an attacker makes a moving ghost aircraft appear to be traveling solely in the ADS-B coverage area. Consider the attack in Figure 5.10 where the ghost aircraft appear from a valid position and is not within the PSR/SSR coverage range to verify if it has an associated PSR or SSR track.

One method to detect this attack is to verify that all ADS-B tracks are verified against actual flight plans that have been loaded into the ontological database. If an ADS-B track does not have the properties of a registered flight plan then its associated packets are flagged as anomalous. The SPARQL query in Listing 5.5 selects all ADS-B reports which do not have a call sign associated to a flight plan. These selected ADS-B Flight Position Reports are flagged as anomalous since they do not have a flight plan.

5.3 Computational Performance of Detection Approach

This section describes a simulation scenario which is used to verify the three ontology-based detection constraints previously explained in Section 5.2.3. The performance of the detection approach is measured by analyzing how the computational overhead of the ontology-based detection process is affected under an increased workload. This addresses one of the research goals in developing ATC-Sense, that being, to develop an understanding of the feasibility of using an ontology-based anomaly detection approach in the ATC domain and cybersecurity

Listing 5.4 Select ADS-B Position Reports with no associated PSR Track

```
PREFIX atc-core: </atc/atc-core#>
PREFIX atc-adsb: </atc/dds-topics/adsb-broadcast#>
SELECT ?track ?report ?lat ?long ?time WHERE {
{?report  a  atc-adsb:ADSBFlightPosition;
          atc-core:hasTrackRank           ?rank;
          atc-core:isAssociatedWithTrack  ?track;
          atc-adsb:hasLatitude            ?lat;
          atc-adsb:hasLongitude           ?long;
          atc-adsb:hasTimeStamp           ?time.}
MINUS {?track atc-core:hasSimilarTrack ?tk}
}ORDER BY ?track ASC(?time)
```

Listing 5.5 Select ADS-B Position Reports with invalid callsign

```
PREFIX atc-core: </atc/atc-core#>
PREFIX atc-adsb: </atc/dds-topics/adsb-broadcast#>
SELECT ?callsign ?report ?lat ?long ?time WHERE {
{?report  a  atc-adsb:ADSBFlightPosition;
          atc-adsb:hasCallsign  ?callsign;
          atc-adsb:hasLatitude  ?lat;
          atc-adsb:hasLongitude ?long;
          atc-adsb:hasTimeStamp ?time.}
MINUS {?fp atc-core:hasCallsign ?callsign}
```

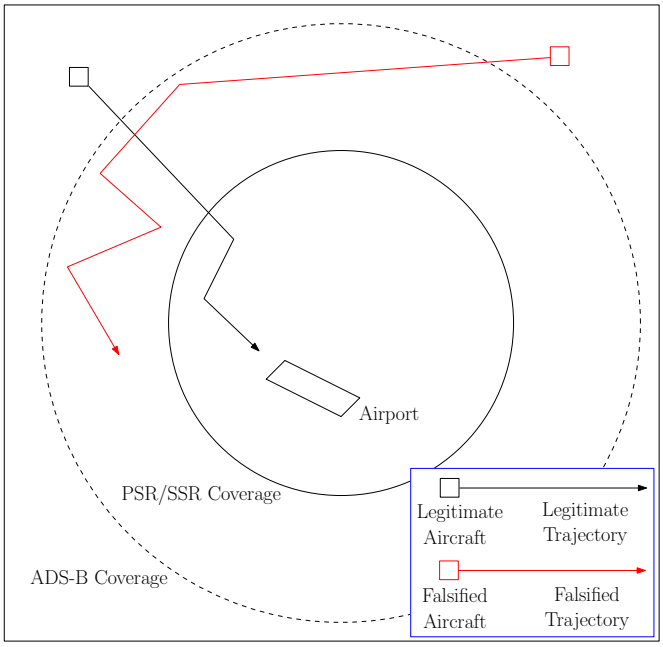

Figure 5.10 Ghost aircraft enters ADS-B coverage range and continues appearing solely in the ADS-B coverage range.

research in general. The utilized scenario is comprised of attacks which are limited to five malicious aircraft and the scenario has been designed so that all the ghost aircraft are detected. For this reason, detection statistics are not measured (e.g. precision, recall, F-score) in this work and is left for future work. This section concludes with a discussion of the overall performance of the detection approach.

Before continuing, it must be noted that the work presented in this section has been conducted in collaboration with a Masters student, as a result there is some overlap between the work presented in this section and the work conducted by the other student [203]. The choice to measure the computational complexity of the detection approach was led by this research,

while the actual execution of the experiments and the gathering of the results were carried out by the other student. The simulation scenario presented in Section 5.3.1 has been equally developed by both parties. The results and analysis in Sections 5.3.2 and 5.3.3 are unique to this work. The work presented by the other student uses a different simulation scenario to compare the detection performance of ATC-Sense with an LSTM-based approach.

5.3.1 Simulation Scenario

The simulations involve varying the number of ghost aircraft on a fixed scenario to measure how the computational overhead varies with the increased loads. The scenario has 10 legitimate aircraft, 4 PSRs, 4 SSRs, and 4 ADS-B antennas. The 4 PSR and 4 SSR locations correspond to actual radar placements around Montreal, Canada. Currently, there are no ADS-B antennas for ATC installed around Montreal, therefore 4 ADS-B antennas are placed in the simulation to test the features of ATC-Sense. The ADS-B antennas cover all of the PSR/SSR coverage area, however, there is a substantial area that is covered by ADS-B but not by PSR/SSR.

The simulation scenario runs for a total of 400 seconds (nearly 7 minutes). The first 3 minutes involve the initialization of the simulation. Within this initial period, the 10 legitimate aircraft operate within the simulation and there are no ghost aircraft. No packets are sent to GraphDB during this period and are instead stored in a buffer. Once 400 DDS packets are generated by the simulation, around the 3 minute mark, the packets are converted to RDF and inserted into GraphDB. At this point the simulation is initialized with the normal aviation entities. After this initialization, batches of 50 packets are inserted into GraphDB as they are generated by the simulation.

The ghost aircraft are injected into the simulation at the 3 minute mark. Simulations are run with attacks ranging from 0 to 5 ghost aircraft, all of which can be detected by the developed constraints logic. During the entire 400 seconds of the simulation one of the three detection constraints logic is operating and is actively looking for anomalous ADS-B packets. For each of the 3 detection constraints, 10 simulations are run with attacks ranging from 0 to 5 ghost aircraft. All results are averaged over the 10 runs. In total $3 * 6 * 10 = 180$ simulations were recorded, resulting in $(180 \text{ sims} * 400 \text{ secs})/3600 \text{ secs} = 20$ hours of total simulation time.

Figure 5.11 shows a simplified representation of the simulation scenario. This figure is meant to convey the behaviour of the injected ghost aircraft and does not reflect all of the 10 legitimate aircraft. The ghost aircraft are labelled from 1 to 5. The falsified aircraft labelled (1) and (2) are static ghost aircraft, (3) is a static ghost aircraft near an airport, (4) is a moving ghost aircraft which travels from the ADS-B coverage area into the PSR/SSR/ADS-B

coverage area, and (5) is a moving ghost aircraft which travels solely in the ADS-B coverage area.

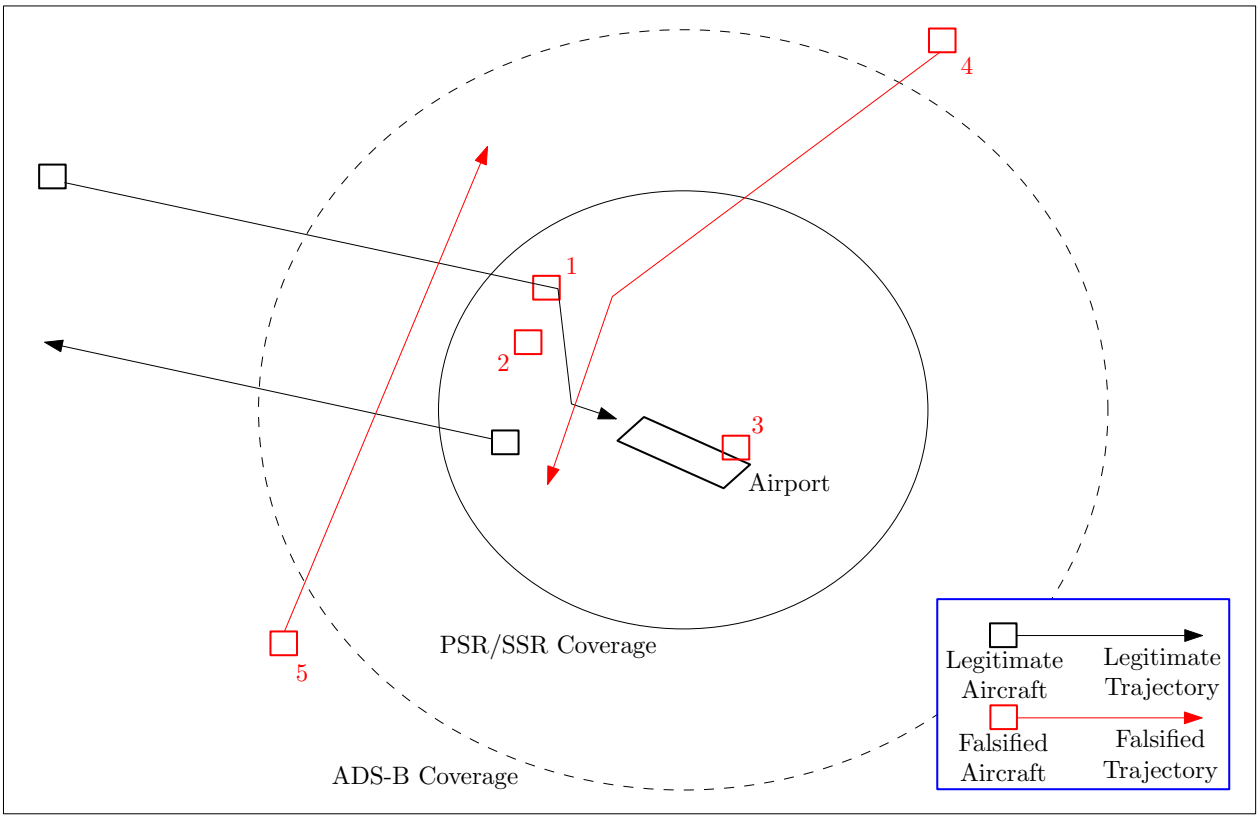

Figure 5.11 Simplified representation of the simulation scenario with 5 falsified aircraft being injected. (1) & (2) Static ghost aircraft, (3) Static ghost aircraft near an airport, (4) Moving ghost aircraft which travels from ADS-B coverage area into PSR/SSR/ADS-B coverage area, (5) Moving ghost aircraft which travels solely in ADS-B coverage area.

The Track Constraint for Consistent Origin detects the falsified aircraft labeled (1) and (2), since these aircraft appear at invalid locations. This is the fastest constraint to check and would be used first in a deployed system.

The Radar Constraint for Reporter Consistency detects the falsified aircraft labeled (1)-(4). The falsified aircraft labeled (1)-(3) are detected because they have a stationary track. The falsified aircraft labeled (4) is detected because there is no PSR/SSR track associated to the ADS-B track.

The Flight Constraint for Existence of Flight Plan detects all the falsified aircraft. In this simulation none of the falsified aircraft have an associated Flight Plan. This does not mean a

capable attacker could not inject a ghost aircraft with an associated Flight Plan, this scenario merely does not evaluate this case.

5.3.2 Performance Metrics

SPARQL Query Time

The SPARQL query time is the amount of time needed to execute the SPARQL queries. This quantity is the sum of the insert and select operating times. Figures 5.12, 5.13, and 5.14 show the SPARQL query times during the simulations for the implemented Track, Radar, and Flight Plan constraints logic, respectively, recorded every 5 seconds and averaged over 10 simulations.

The query logic is operating for the entire 400 seconds of the simulation and the first insert occurs at 180 seconds. The initial 180 seconds have very low times since the detection queries are being performed with no aviation entities in GraphDB. At 180 seconds there is a big increase in query time when the initial 400 packets are inserted. After 180 seconds there are noticeable spikes at some recurring interval. This corresponds to the batches of 50 packets being inserted into GraphDB.

The Track query logic is the most complex with the highest query time, the Radar query logic is slightly less complex with a reduced query time, and the Flight Plan logic is considerably less complex with a much quicker query time. The total query time generally rises as more ghost aircraft exist in the simulation, since there is more data entering the database and being reasoned upon.

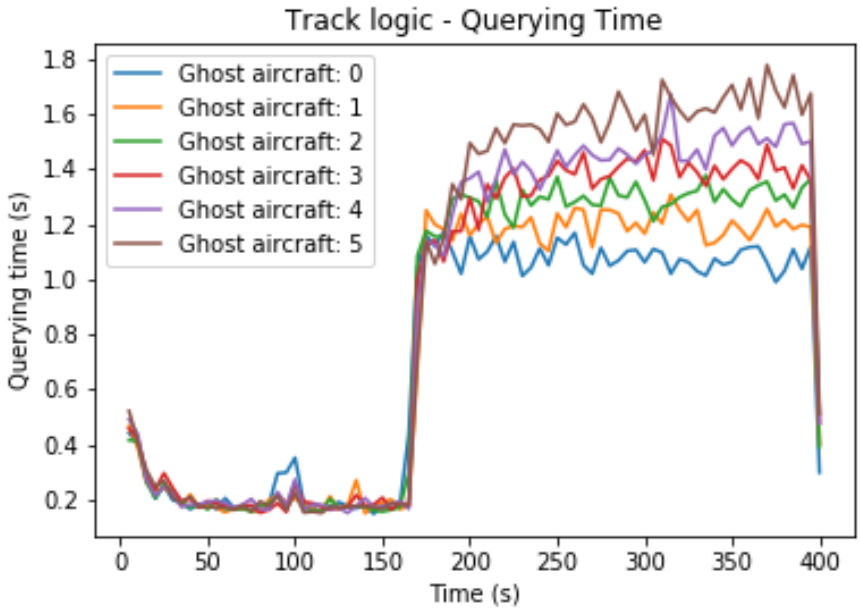

Figure 5.12 SPARQL Query Time: Track Constraint Logic (average of 10 simulations)

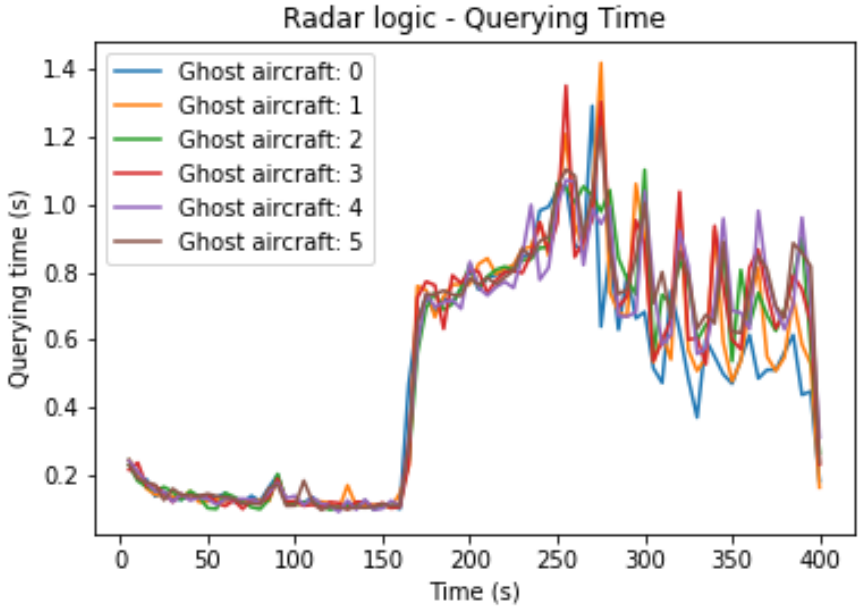

Figure 5.13 SPARQL Query Time: Radar Constraint Logic (average of 10 simulations)

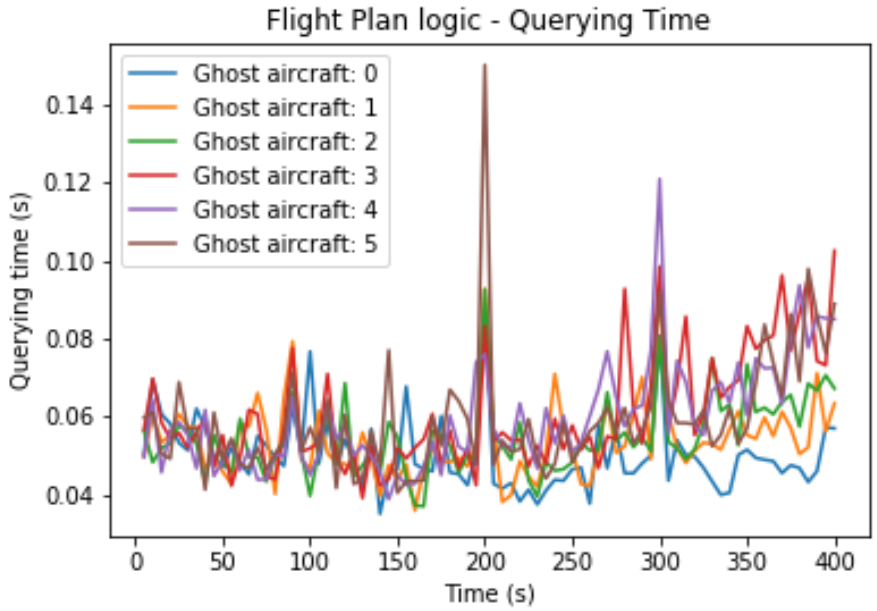

Figure 5.14 SPARQL Query Time: Flight Plan Constraint Logic (average of 10 simulations)

RDF Triples Downloaded

The RDF triples downloaded represents the quantity of RDF triples extracted from GraphDB by performing SPARQL select operations. Figures 5.15, 5.16, and 5.17 show the number of triples downloaded during the simulations for the implemented Track, Radar, and Flight Plan constraints logic, respectively, recorded every 5 seconds and averaged over 10 simulations.

In the initial 180 seconds there are no packets queried from GraphDB as part of the initialization period. For each of the Track, Radar, and Flight Plan constraints there is a noticeable proportional increase in the number of RDF triples downloaded relative to the number of ghost aircraft in the simulation. In the Flight Plan logic there is a large linear increase in triples downloaded over the course of the simulation, however, the total number of triples is relatively low and is an order of magnitude smaller than the other constraints.

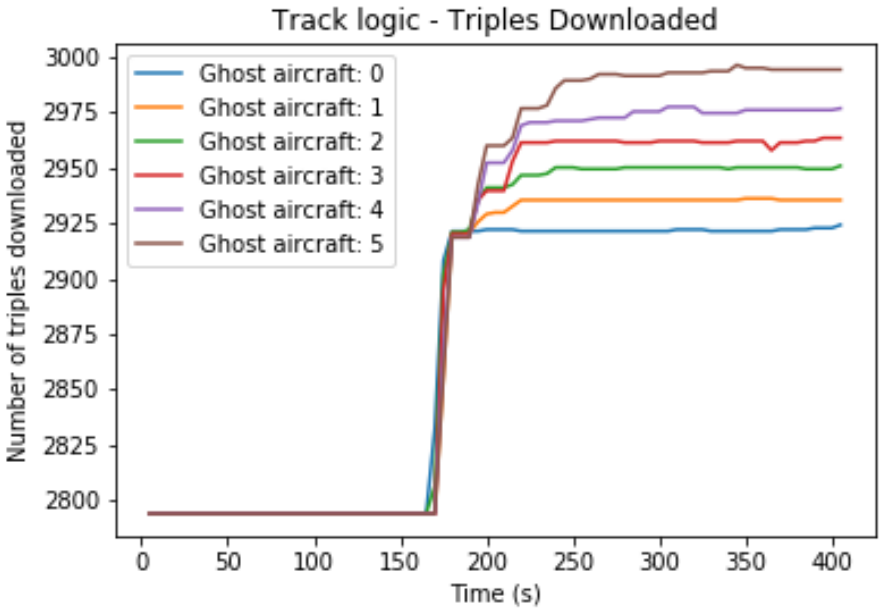

Figure 5.15 RDF Triples Downloaded: Track Constraint Logic (average of 10 simulations)

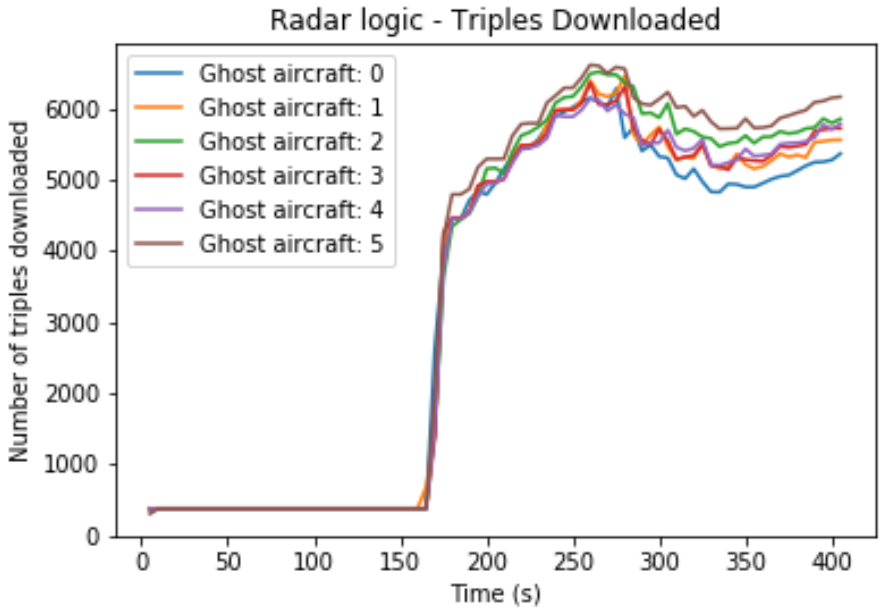

Figure 5.16 RDF Triples Downloaded: Radar Constraint Logic (average of 10 simulations)

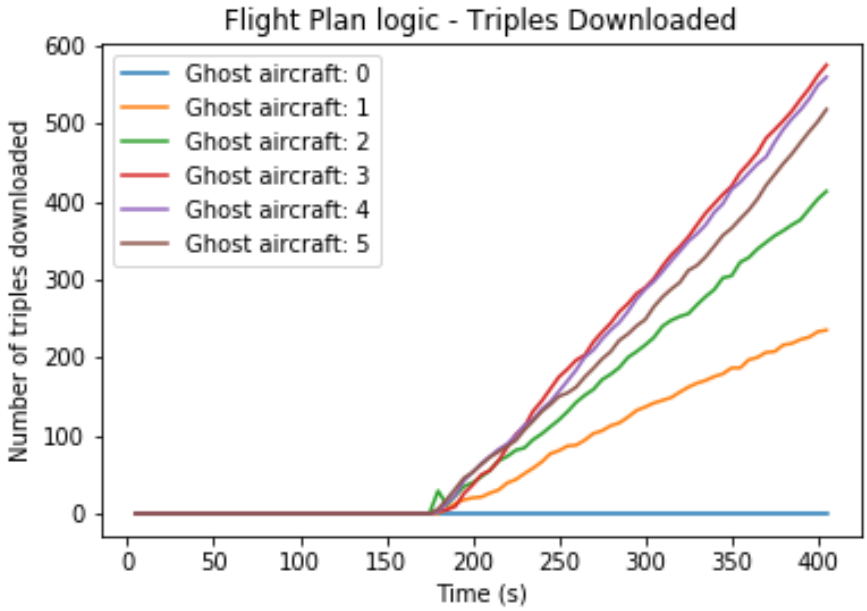

Figure 5.17 RDF Triples Downloaded: Flight Plan Constraint Logic (average of 10 simulations)

SPARQL Complexity

The SPARQL complexity is a quantity that represents the estimated number of iterations required to perform SPARQL queries. Figures 5.18, 5.19, and 5.20 shows the complexity of the SPARQL queries during the simulations for the implemented Track, Radar, and Flight Plan constraints logic, respectively, recorded every 5 seconds and averaged over 10 simulations.

In the initial 180 seconds the queries are less complex since the initialization data has not been inserted into GraphDB. There is a noticeable jump in complexity after the initial packets are inserted into GraphDB. In all three cases, the complexity of the queries is marginally affected by increasing the number of ghost aircraft. However, there is a dramatic linearly increasing total complexity as the simulation continues to execute. This occurs because as more triples are inserted into the database, the queries need to iterate over an increasingly large number of entities. This could be potentially remedied by implementing a process which removes old RDF data which is no longer needed.

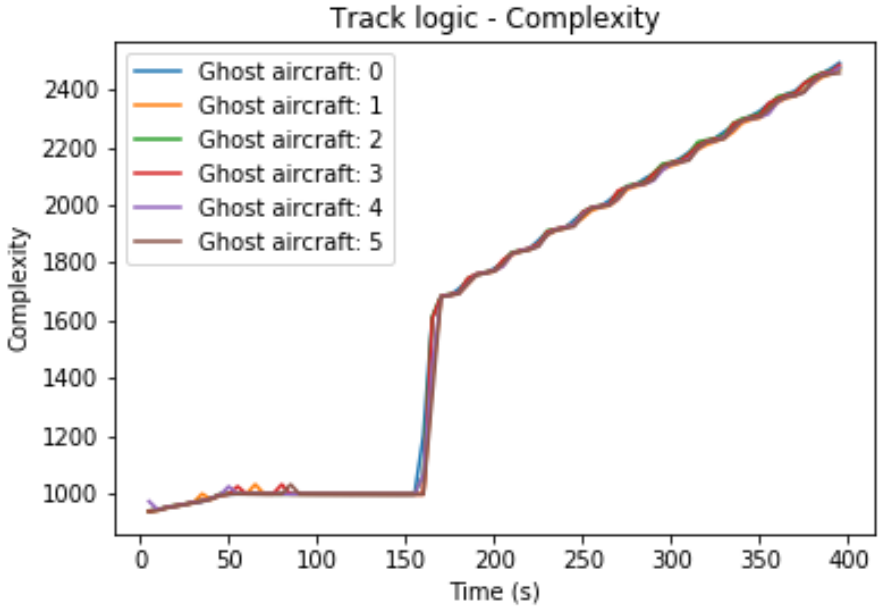

Figure 5.18 SPARQL Complexity: Track Constraint Logic (average of 10 simulations)

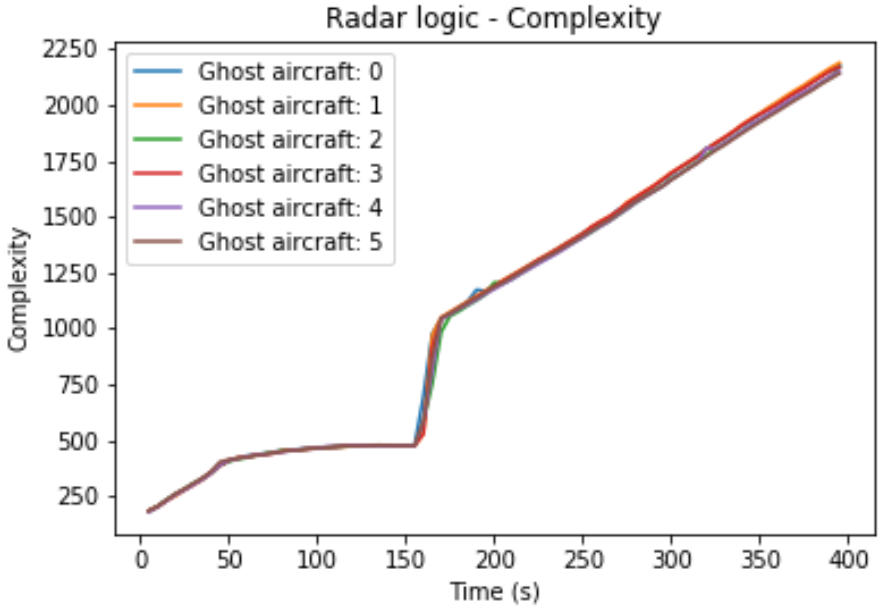

Figure 5.19 SPARQL Complexity: Radar Constraint Logic (average of 10 simulations)

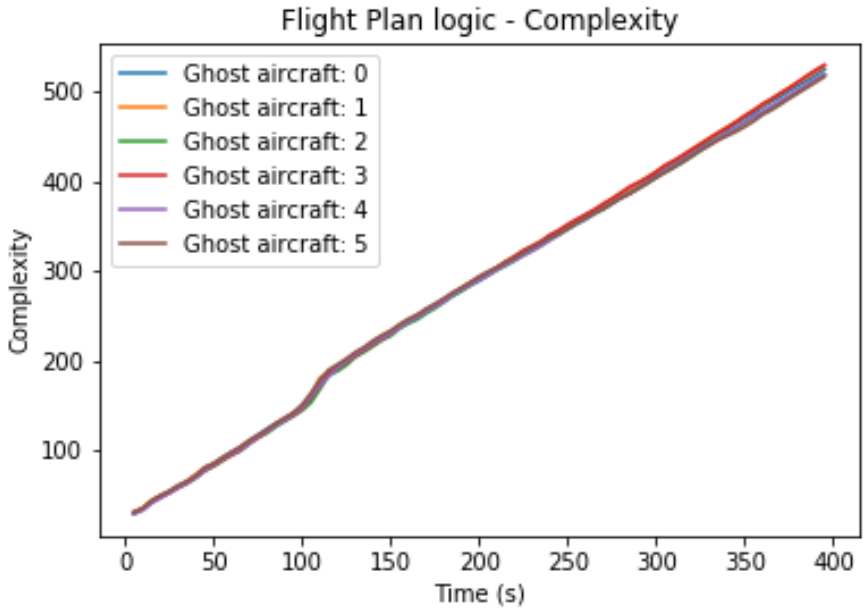

Figure 5.20 SPARQL Complexity: Flight Plan Constraint Logic (average of 10 simulations)

GraphDB Read/s and Write/s

GraphDB provides the average number of RDF reads and writes per second. Table 5.3 shows the reads/s and write/s when using each of the detection constraints.

Table 5.3 GraphDB Read/s and Write/s for the detection constraints (average of 10 simulations)

	Track Constraints		**Radar Constraints**		**Flight Constraints**	
# Ghosts	**Reads/s**	**Writes/s**	**Reads/s**	**Writes/s**	**Reads/s**	**Writes/s**
0	502.5751	5.3093	303.3250	5.3988	15.2259	4.9518
1	512.6787	5.3361	307.4927	5.3689	22.2010	5.5481
2	512.2853	5.2852	308.0748	5.2636	25.4083	5.3126
3	514.2179	5.3939	309.2526	5.2867	29.5177	5.3133
4	518.3481	5.2978	309.5447	5.3408	30.0592	5.5255
5	513.5878	5.4505	316.4494	5.4167	29.4796	5.4622

For each of the constraints, the reads/s tends to increase with the number of ghost aircraft in the simulation. This increase in number of read operations is to be expected, since there are more entities in the database to be extracted and analyzed when looking for anomalies.

The writes/s is not affected by the number of ghost planes nor by which detection constraints are used. This is likely because there are always batches of 50 packets being inserted into GraphDB at regular intervals for all the performed tests.

5.3.3 Analysis of Detection Approach

The experiments presented in this work are an initial study of the feasibility of using an ontology-based detection approach in detecting anomalies in not only ADS-B messages, but cybersecurity related artefacts in general. Previous cybersecurity-related research use ontology-based frameworks for detecting anomalies in static datasets, while this work is the first, to the author's knowledge, to do so in a real-time setting. The purpose of measuring the computational overhead of the detection approach is to gain an understanding of the issues that may arise when experiments are scaled to a more realistic set of air traffic.

The SPARQL query times generally increase as more entities are present in the simulation. This is particularly noticeable with the Track Logic in Figure 5.12 which shows that when there are no ghost aircraft the execution time is roughly 1.2 seconds and while there are 5 ghost aircraft the execution time jumps to around 1.6 seconds. In such a small setting this is not drastic, however, if there are to be potentially tens or hundreds of entities, the SPARQL execution times will likely be magnitudes higher and may become too slow for a time-critical detection approach. Additionally, the complexity of the SPARQL queries shown in Figures 5.18, 5.19, and 5.20 demonstrate a dramatic linear increase as the simulation executes. The RDF triples are entering the database at a relatively constant rate, forcing the SPARQL queries to iterate over a steadily-increasing set of data. This gives the indication that there is need to further analyze at what point this increased complexity becomes a bottleneck for detecting anomalies. A primary conclusion from this analysis is that work needs to be done to speed up the execution times of the queries and limit their complexity. A stream of future work could be to devise a process for removing existing RDF triples from an ontological database that are deemed to no longer be necessary in the detection process, in hopes of reducing the computational overhead of the SPARQL queries.

Interpreting the reads/s and writes/s also offers some valubale insights. There is likely some upper limit of reads/s and writes/s, for a particular architectural setup. When given an environment with a large number of entities generating position reports, it is conceivable that not enough RDF data can be inserted and reasoned upon quick enough to perform adequate threat detection reasoning. Future work should strive to formalize an understanding of how many entities can be reasoned upon in an adequate time, for a particular architectural setup.

It must also be mentioned that the defensive coverage of the implemented detection constraints, at the time of writing, is relatively low. While this chapter identified a number of potential detection constraints within three streams of logic, only a subset has been implemented. The set of implemented detection constraints is shown in Table 5.4. Iteratively building upon these constraints will result in a larger defensive coverage space.

Table 5.4 Implemented Track, Radar, and Flight Plan constraints

Constraint	Implemented
1. Track	
(a) Consistent Origin	✓
(b) Consistent Destination	✓
(c) Consistent Take-off	
(d) Consistent Landing	
2. Radar	
(a) Reporter Consistency	✓
(b) Temporal & Physical Consistency	
3. Flight Plan	
(a) Consistent Instrumentation	
(b) Existence of Flight Plan	✓
(c) Consistency with Flight Plan	

5.4 Conclusion

This chapter provides a description of ATC concepts and reporting technologies, emphasizing the need for defensive solutions that can detect falsified ADS-B constructed by capable attackers. This work suggests using an ontology-based approach to detect anomalous ADS-B messages, since this does not assume any restrictions to attacker capabilities and provides explainable decisions when anomalies are detected. An ATC testbed has been built which simulates air traffic in a controlled airspace, the position reports generated by PSR, SSR, and ADS-B devices, and allows the ability to inject ADS-B attacks. The data generated by the ATC simulation is used to populate an ontological database with RDF data which describes the entities in the simulation. An ontology-based detection platform, known as ATC-Sense, is used to detect anomalous ADS-B messages which violate a set of aviation-related constraints. The conducted experiments demonstrate the feasibility of using an ontology-based approach for detecting anomalous ADS-B messages and highlight the need for future work to increase the speed at which SPARQL queries are used to detect anomalies.

Overcame Challenges

This testbed and detection approach has been developed by a number of personnel over several years and the work conducted in this research has contributed to a subset of the components of the overall testbed. In terms of implementation, this research has helped in the integration of the Parser program into the testbed, the execution of SPARQL read and write commands over a REST API with GraphDB, and the creation of scenario files. This research has also shaped the experimental approach presented in this chapter and synthesized the capabilities of the testbed in a coherent manner. The artefacts presented in this chapter are unique to this document.

Since the detection logic uses SPARQL queries to analyze RDF data, all the ATC data is converted from FSD to DDS and then into RDF format. A custom parser program written in C has been utilized to parse the FSD packets into DDS data. The parser receives FSD packets generated by the simulation and captures the information that is present in the respective PSR, SSR, and ADS-B packets. This information is then published to the subscriber module on the DDS network.

Another challenge involved getting the information from the DDS network into GraphDB as RDF triples. The mapping of DDS to RDF is relatively straightforward, the larger challenge involved inserting the RDF triples into GraphDB. The GraphDB free version is limited to two concurrent SPARQL operations. One of the SPARQL operations is consumed by the SPARQL logic which queries the triples in the database for the indication of anomalous ADS-B messages. Therefore, the remaining SPARQL operation is used to insert the RDF triples into the database. Single DDS packets cannot be inserted into GraphDB as RDF in real-time as they arrive, since the insertion does not complete in time before another insertion request arrives and is dropped by GraphDB. Instead, the RDF data is inserted into GraphDB in batches. Once 50 DDS packets (of PSR, SSR, and ADS-B data) are collected, they are converted to RDF then inserted into GraphDB all at once. The insertion then has time to complete before the next batch arrives. The number of 50 has been fixed for the current implementation.

Limitations

ATC-Sense is an expert system utilizing a constraints-based detection approach requiring every type of potential attack need to be covered by human-generated detection rules. Ideally, the detection rules would be written generally enough to cover a number of attack scenarios, however, it will take considerable effort to ensure complete defensive coverage.

The adoption of a platform such as ATC-Sense would require the deployment of a number of new systems to integrate this solution with existing infrastructure. It is not clear whether any advantages gained from an ontology-based approach would outweigh this required architectural overhaul. Additionally, RDF data was originally conceived as a method to enable semantically rich queries of data stored on the Internet as part of the Semantic Web. SPARQL queries are not known for their speed and the adoption of ATC-Sense would require some technical breakthroughs to increase their speed.

Future Work

Particular scenarios can be constructed which would not be detected by the full set of the proposed constraints. Consider the scenario where a ghost aircraft enters the ADS-B coverage range at a time very close to that of an actual registered aircraft. The proposed Track, Radar, and Flight Constraints logic will not be able to distinguish between the legitimate aircraft and the ghost aircraft because the track reports will be coming from similar locations. An additional set of physics-based constraints must be developed to determine that the trajectories of reported ADS-B positions follow the laws of physics.

ATC-Sense has been developed as a proof of concept to demonstrate the applicability of semantic based querying to detect attacks in the ATC domain. This approach can detect attacks, however, more work needs to be done to further validate this approach. Additional optimization of the detection engine is required for this solution to scale to actual ATC settings in real-world airspace. An interesting avenue to test this approach would be to deploy the solution in a virtual ATC environment, such as the VATSIM community. Attacks could be injected into the virtual environment, where there are humans piloting virtual aircraft and performing ATC tasks, without any real world consequences. This would provide a venue to further verify the constraints-based detection in a more lifelike setting. Upon verifying this detection approach in a virtual setting, the next step would be to test the approach in an actual physical setting under experiments in small regional airports where the air traffic can be controlled and notified of potential false ADS-B messages.

Lessons Learned

A significant amount of work was spent developing features were never successfully implemented and were not used in any experimental capacity. Some of the initial intentions of the testbed were to automatically gather real-world flight plans and weather data to populate the Euroscope simulation with scenarios with a high resemblance to real-world air traffic. Additionally, since flight plans contain aircraft type, the ATC ontology was equipped with

properties that could be leveraged to aid in the anomaly detection process based of off aircraft capabilities. These intended features would indeed have value in performing more realistic experiments, however, the efforts spent on these may have been better applied initially in validating the ontology-based detection approach using a smaller experimental scope. It is likely more fitting to incrementally add features, conduct validating experiments, and publish results, in an iterative manner.

The theory, tools, and processes behind RDF databases are still relatively recent, especially in comparison to relational databases, thus there will still be some time before ontology-based anomaly detection reaches a mature state. With this in mind, an insight gained from this work is that instead of performing large SPARQL queries to search for anomalies, the detection process is more performant and manageable when several small SPARQL queries are used with some processing of results between queries.

Using Euroscope as a simulation platform proved to be invaluable, since it can automatically issue commands to coordinate air traffic, as well as other numerous advantages. This enabled the development process to advance the anomaly detection platform without much prior knowledge in aviation. This highlights the advantage gained from reusing existing tools whenever possible, allowing efforts to be focus on addressing the research questions at hand.

CHAPTER 6 A TAXONOMY FOR CRITICAL INFRASTRUCTURE CYBERSECURITY TESTBEDS

Cybersecurity testbeds are used to replicate some subset of the reality of an actual operational system in order to perform repeatable experiments. This is done to gain an understanding of open cybersecurity challenges facing CI without endangering any real-world systems. This chapter presents a taxonomy for classifying CI-CPS cybersecurity testbed characteristics to better understand what aspects of the real-world system are covered by the testbed and align the insights to be gained from performing cybersecurity experimentation with the testbed. The characteristics of the taxonomy are influenced by the case studies shown in Chapters 3-5 and aims to capture the fundamental components of the described testbeds in a concise manner. Additionally, this taxonomy is proposed as a method for generating testbed requirements derived from intended CI-CPS cybersecurity experiments arising out of research questions. The idea is to first design the testbed at a high level by filling in the taxonomy with the intended components to be constructed and then building the testbed to meet this blueprint.

A significant challenge when performing the research presented in Chapters 3-5 was getting stakeholders to agree upon what components should be represented in a testbed. There are many open cybersecurity questions in CI-CPS, making it difficult to identity which ones to investigate given the available resources and expertise. Upon completing the work presented in the previous chapters it became apparent that a methodology for streamlining the testbed design and construction phases would be beneficial to future researchers. A taxonomy that captures the hierarchies of control amongst the physical and digital components in a CI-CPS environment is a natural fit for solving this problem, since it allows researchers to identify which components they intend to represent in a testbed. Having this taxonomy at the outset of this research would have allowed us to rapidly iterate through testbed designs and determine more quickly what our finished systems would be comprised of. At the time of writing this proposed taxonomy and testbed development process has not been published since this came at the end of this research. However, it is the intention of this research group to publish it in the future and promote it in the scientific community.

This chapter is organized as follows; Section 6.1 describes the testbed taxonomy, Section 6.2 applies the taxonomy to the three testbeds presented earlier in this work (i.e. NPP, MG, ATC), Section 6.3 is the analysis of the taxonomy and demonstrates the proposed process

for generating testbed requirements from initial research questions, and Section 6.4 is a conclusion to this chapter.

6.1 POILA: A Critical Infrastructure Cybersecurity Testbed Taxonomy

This section presents the POILA taxonomy which describes the components of CI cybersecurity testbeds among five main classes. Note that this taxonomy specifically applies to CI-CPS cybersecurity testbeds and is most appropriate when there are physical processes being controlled by digital components. Many of POILA's characteristics are extensions of the work done by Lemay et al. regarding testbed design for cybersecurity experimentation [204–206]. This initial work demontrates how cyberattacks against SCADA systems can have serious impact on an electrical grid though the use of co-emulation testbed principles, while also laying the groundwork in testbed design for modeling the hierarchy of digital control in an industrial CPS .

The insights gained from the case studies in Chapters 3-5, as well as the reviewed relevant literature, have provided a backdrop for establishing the classes and properties for classifying CI-CPS testbeds proposed in POILA.

The name POILA is derived from the first letters of the five main classes of information captured in the taxonomy. The class are described as follows:

- **P**hysical Process Architecture: Classifies the physical process represented by the testbed.
- **O**T Architecture: Classifies the OT sensors and controllers represented by the testbed.
- **I**T Architecture: Classifies the IT communications network and data management systems represented by the testbed. This represents all IT-related systems which support the control of the physical process.
- **L**ogistics Architecture: Classifies the systems used for enterprise logistics planning represented by the testbed. These systems are IT-based systems, however, they are classified separately from the IT Architecture class since they are not directly involved in the control or reporting of the physical process.
- **A**ctivity Classification: Classifies the cybersecurity research activities which can be conducted with the testbed.

The OT Architecture class is further broken down into two sublclasses; Sensors and Controllers. The Sensors subclass is used to classify the OT components which are used to report

the physical processes. The Controllers subclass is used to classify the OT components which are used to invoke control over the physical processes.

The IT Architecture class is also further broken down into two subclasses; Communications Network and Data Management. The Communications Network subclass is used to classify the IT components which comprise the communications network within the testbed. The Data Management subclass is used to classify all software systems which utilize data generated in the testbed. The Data Management subclass is used to some extent as a catch-all for all software systems not captured by the other classes.

The classes and subclasses used in the POILA taxonomy provide clear definitions for classifying testbed components, providing a method to capture the essence of CPS-CI testbed capabilities. The POILA taxononomy is represented in Figure 6.1, which demonstrates the hierarchy of classes and their associated properties. The properties of the Physical Process Architecture class are explained in Table 6.1, the properties of the OT and IT Architecture subclasses are the same for all the subclasses and are explained in Table 6.2, the properties of the Logistics Architecture class are explained in Table 6.3, lastly, the properties of the Activity class are explained in Table 6.4.

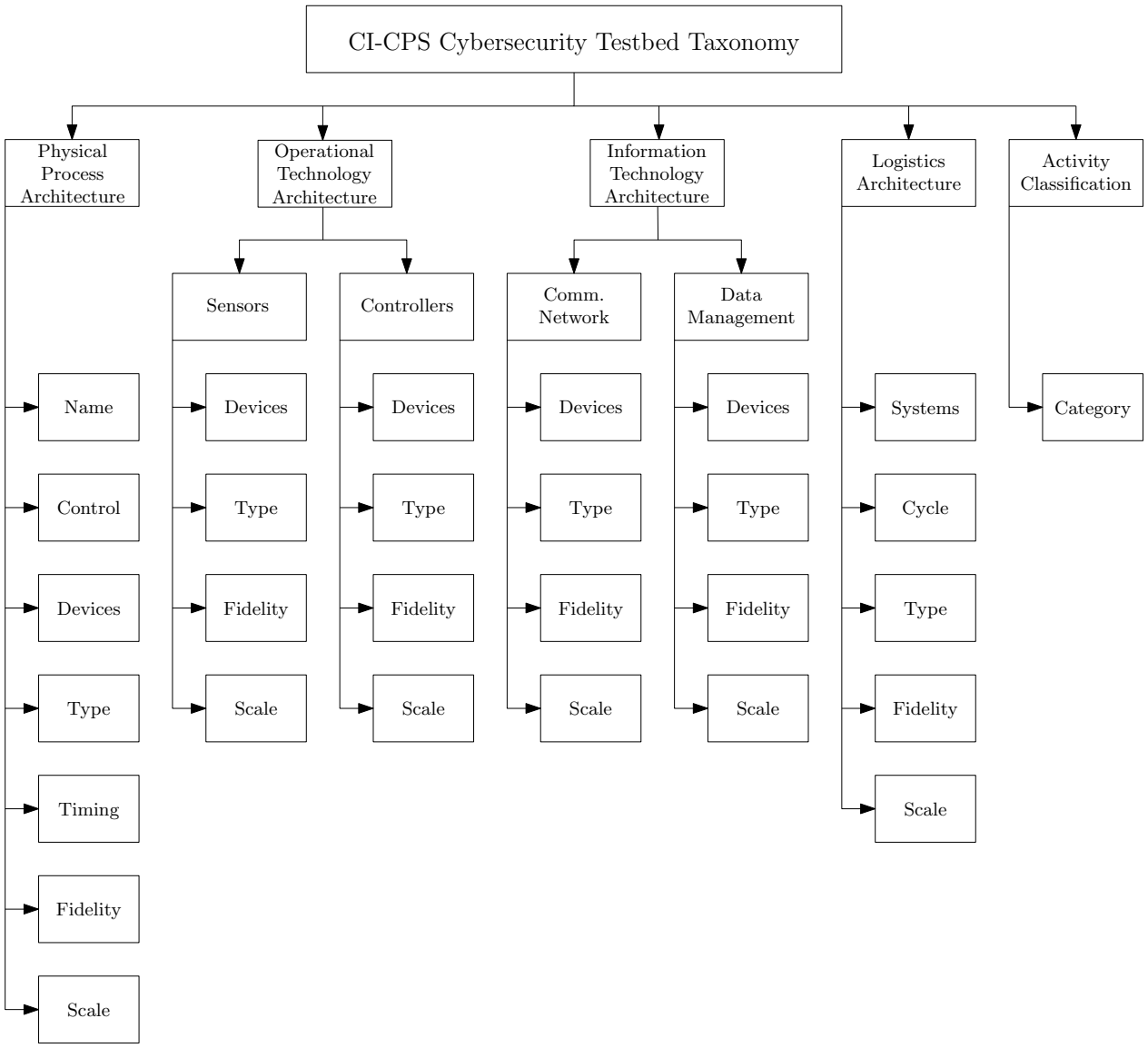

Figure 6.1 POILA Taxonomy for classyfing CI-CPS cybersecurity testbeds

Table 6.1 Description of Physical Process Architecture class properties

Class: Physical Process Architecture			
Property	**Description**	**Values**	**Interpretation**
Name	Name of the process	*Text*	Description of the process
Control	Control discipline	*Text*	Description of the control
Devices	Represented equipment	*Text list*	Description of the devices
Type	Representation of the physical process	Physical	Represented by physical components
		Semi-physical	Represented by some physical components, integrated with a simulation (HIL)
		Simulation	Components are simulated with software
Timing	Time in which the representation of the physical process executes	Real-time	Actual timing
		Semi-real-time	Some processes operate in real-time, others are simulated
		Simulated-time	Time is accelerated due to software simulation
Fidelity	Faithfulness of the representation to the real-world physical process	Full	Complete resemblance
		High	Close resemblance, few assumptions
		Medium	Reasonable resemblance, moderate assumptions
		Low	Small resemblance, many assumptions
Scale	Faithfulness of the representation to the size of the real-world physical process	Full	Completely resembles
		High	Close resemblance, slightly scaled-down
		Medium	Reasonable resemblance, moderately scaled-down
		Low	Slight resemblance, scaled-down to a large degree

Table 6.2 Description of OT and IT Architecture class properties

Class: OT Architecture, Subclass: Sensors **Class: OT Architecture, Subclass: Controllers** **Class: IT Architecture, Subclass: Comm. Network** **Class: IT Architecture, Subclass: Data Management**			
Property	**Description**	**Values**	**Interpretation**
Devices	Devices represented	*Text list*	Description of the devices
Type	Representation of the devices	Physical	Actual devices are used
		Virtualization	Devices are emulated
		Simulation	Software simulates the devices
		Logical	Devices are not represented, however, relevant data is generated by simulation software
Fidelity	Faithfulness of the behaviour of the represented devices to their real-world counterparts	Full	Physical devices, full resemblance
		High	Close resemblance
		Medium	Moderate resemblance
		Low	Small resemblance
		N/A	Devices are not represented
Scale	Faithfulness of the amount of represented devices to their real-world counterparts	Full	The same
		High	Slightly scaled-down
		Medium	Moderately scaled-down
		Low	Scaled-down to a large degree
		N/A	Devices are not represented

Table 6.3 Description Logistics Architecture class properties

Class: Logistics Architecture			
Property	**Description**	**Values**	**Interpretation**
Systems	Systems represented	*Text list*	Description of the systems
Cycle	Planning cycle of the logistics systems	Monthly	Monthly planning horizons
		Weekly	Weekly planning horizons
		Daily	Daily planning horizons
		Hourly	Hourly planning horizons
Type	Representation of the systems	Physical	Actual systems are used
		Virtualization	Systems are emulated
		Simulation	Software is used to simulate the systems
		Logical	Systems are not represented, however, relevant data is generated by simulation software
		None	Systems are not represented
Fidelity	Faithfulness of the behaviour represented systems to their real-world counterparts	Full	Actual systems are used
		High	Close resemblance
		Medium	Moderate resemblance
		Low	Small resemblance
		N/A	Systems are not represented
Scale	Faithfulness of the amount of represented systems to their real-world counterparts	Full	The same
		High	Slightly scaled-down
		Medium	Moderately scaled-down
		Low	Scaled-down to a large degree
		N/A	Systems are not represented

Table 6.4 Description of the Activity class properties

Class: Activity			
Property	**Description**	**Values**	**Interpretation**
Category	The cybersecurity experiment objectives of the testbed	Attack Impact Analysis (AIA)	Witness the effects of attacks against the systems represented by the testbed
		Infrastructure Vulnerability Analysis (IVA)	Identify inherent vulnerabilities in the systems represented by the testbed
		Defensive Tools Analysis (DTA)	Determine the coverage provided by defensive tools deployed in the testbed
		Education and Training (E&T)	Provide personnel with an environment to explore operating procedures during cyberincidents

6.2 Classification of Testbeds

This chapter presents the POILA taxonomy as a method for generating testbed requirements from initial research questions, however, the testbeds presented in this work had already been developed prior to POILA being conceived. In fact, experiencing the process of developing testbeds without a clear methodology is what inspired this work to develop POILA. This section instead demonstrates another use case for POILA, that being, a process for classifying existing testbeds as to gain a succinct overview of their characteristics. Hence, this section applies the POILA taxonomy, presented in Section 6.1, to the three cybersecurity testbeds described in Chapters 3-5. This exercise is intended to provide the reader with an understanding of POILA and naturally leads to the discussion of how it can be applied to developing testbed requirements, in the Section 6.3.

The overall classification of all the testbeds using the POILA taxonomy is shown in Table 6.5. The selected classification values of the individual properties for each testbed are subjective, thus the justification for the chosen values are explained in more detail; the classification of the NPP testbed described in Chapter 3 is explained in Section 6.2.1, the classification of the MG testbed described in Chapter 4 is explained in Section 6.2.2, and the classification of the ATC testbed described in Chapter 5 is explained in Section 6.2.3.

Table 6.5 POILA Taxonomy applied to the CI testbeds presented in this work

POILA Taxonomy			Testbed Classification		
Class		Property	NPP	MG	ATC
Physical Process		Name	Steam Generation	Solar Power, Energy Storage	Aircraft Traffic
		Control	BLC	Minimize Cost	Separation Control
		Devices	Steam Generator, Primary Loop, Secondary Loop	PV, ESS, Households	Aircraft
		Type	Semi-physical	Simulation	Simulation
		Timing	Semi-real-time	Simulated-time	Simulated-time
		Fidelity	Medium	Medium	Medium
		Scale	Medium	Low	High
OT	Sensors	Devices	PLCs, SCADA, Water Level Sensor	PLCs, Smart Meters	PSR, SSR, ADS-B
		Type	Physical	Logical	Simulation
		Fidelity	High	N/A	Low
		Scale	Medium	N/A	High
	Controllers	Devices	PLCs, VFDs	PLCs	Air Traffic Controller
		Type	Physical	Logical	Simulation
		Fidelity	High	N/A	Medium
		Scale	Medium	N/A	High
IT	Network	Devices	LAN, Data Diodes, Switches	WAN	DDS Network
		Type	Physical	Logical	Simulation
		Fidelity	High	N/A	Low
		Scale	Medium	N/A	Low
	Data Mgmt.	Devices	Historian, Process Servers, OT-SOC	Weather Forecaster, Load Forecaster	Parser, Publisher, Subscriber, RDF Databas
		Type	Physical	Logical	Simulation
		Fidelity	High	N/A	Medium
		Scale	Medium	N/A	Low
Logistics		Systems	Production Planning	MGCC	Flight Planning, Route Scheduling
		Cycle	Monthly, Weekly	Hourly	Daily
		Type	None	Simulation	None
		Fidelity	N/A	Medium	N/A
		Scale	N/A	Medium	N/A
Activity		Category	IVA, DTA, E&T	AIA	DTA

6.2.1 Nuclear Power Plant Testbed Classification

This section demonstrates and explains the POILA classification of the NPP testbed presented in Chapter 3.

Testbed: NPP, Class: Physical Process Architecture

- **Name:** Steam Generation
  - The primary process being controlled is steam generation.
- **Control:** Boiler Level Control (BLC)
  - The testbed is used to control the water level in the simulated boiler
- **Devices:** Steam Generator, Primary Loop, Secondary Loop
  - The steam generator is simulated with physical components. The physical BLC system implementation is influenced by values generated by the MATLAB/Simulink model of the PWR NPP primary and secondary loops.
- **Type:** Semi-physical
  - There is a HIL effect between the physical BLC system and the MATLAB/Simulink model.
- **Timing:** Semi-real-time
  - The BLC system operates in real time, while the MATLAB/Simulink model operates in an accelerated time.
- **Fidelity:** Medium
  - The MATLAB/Simulink model simulates the entirety of a PWR NPP, with simplifications of the physical processes. The physical BLC implementation controls actual water levels, however, does not produce actual steam.
- **Scale:** Medium
  - The MATLAB/Simulink model simulates the entirety of a scaled-down PWR NPP. The physical BLC implementation is substantially reduced compared to its real-world counterpart.

Testbed: NPP, Class: OT Architecture, Subclass: Sensors

- **Devices:** PLCs, SCADA, Water Level Sensor
  - Sensors connected to PLCs perform the majority of the automated sensing operations. SCADA systems are deployed for reporting processes to personnel.
- **Type:** Physical
  - Actual devices are used.
- **Fidelity:** High
  - Real sensor devices are used, with simplifications to their behaviour to handle the reduced complexity of the testbed.
- **Scale:** Medium
  - The amount of sensors is reduced compared to a real-world BLC system.

Testbed: NPP, Class: OT Architecture, Subclass: Controllers

- **Devices:** PLCs, VFDs
  - PLCs perform the majority of the automated control functions. VFDs control the pumps used for for the BLC feedwater and outlet water
- **Type:** Physical
  - Actual devices are used.
- **Fidelity:** High
  - Real control devices are used, with simplifications to their behaviour to handle the reduced complexity of the testbed.
- **Scale:** Medium
  - The amount of controllers is reduced compared to a real-world BLC system.

Testbed: NPP, Class: IT Architecture, Subclass: Communications Network

- **Devices:** LAN, Data Diodes, Switches
  - A LAN connects the devices, data diodes are used to transmit data across the level 2-3 boundary, and switches are used to relay data amongst the network.

- **Type:** Physical
  - Actual devices are used.
- **Fidelity:** High
  - Real network devices are used, with simplifications to handle the reduced complexity of the testbed.
- **Scale:** Medium
  - The size of the network is reduced compared to real-world control rooms.

Testbed: NPP, Class: IT Architecture, Subclass: Data Management

- **Devices:** Historian, Process Servers, OT-SOC
  - The historian captures all process values, the process servers aggregate all process related data, and the OT-SOC analyzes are security-related network traffic
- **Type:** Physical
  - Actual systems are used.
- **Fidelity:** High
  - Real data management systems are used, with simplifications to handle the reduced complexity of the testbed.
- **Scale:** Medium
  - The amount of data management systems is reduced compared to real-world control rooms.

Testbed: NPP, Class: Logistics Architecture

- **Systems:** Production Planning
  - Production planning systems determine the amount of electricity produced.
- **Cycle:** Monthly, Weekly

  - NPPs cannot quickly change the amount of electricity they generate, thus they generally produce a relatively constant amount of electricity. However, long-term planning horizons are needed to prepare for reducing the capacity of the NPP to perform maintenance. Experiments conducted in this work consider the simulated NPP to be operating at full capacity.

- **Type:** None
  - Production planning is not represented by this testbed. Instead, the experiments assume the NPP operates in a manner to produce a constant amount of electricity.

- **Fidelity:** N/A
  - Systems are not represented.

- **Scale:** N/A
  - Systems are not represented.

Testbed: NPP, Class: Activity

- **Category:** Infrastructure Vulnerability Analysis (IVA), Defensive Tools Analysis (DTA), Education and Training (E&T)
  - This work presented an IVA activity, which analyzed how a BLC system may be vulnerable to being affected by a cyberattack. However, the testbed is a relatively complex representation of the infrastructure used in the levels 2 and 3 of an NPP, utilizing many real-world systems, and has been constructed to also perform for DTA, and E&T activities not presented in this dissertation [116]

6.2.2 Microgrid Testbed Classification

This section demonstrates and explains the POILA classification of the MG testbed presented in Chapter 4.

Testbed: MG, Class: Physical Process Architecture

- **Name:** Solar Power, Energy Storage
  - The testbed simulates the generation of solar power and the storage of energy in batteries

- **Control:** Minimize Cost
  - The generated electricity is used to satisfy the loads of the households in the MG while minimizing the total cost of purchasing electricity from the main electrical grid.
- **Devices:** PV, ESS, Households
  - The testbed considers simulated PV, ESS, and households.
- **Type:** Simulation
  - All systems are simulated using MATLAB/Simulink software
- **Timing:** Simulated-time
  - The time is accelerated in the simulation
- **Fidelity:** Medium
  - The MATLAB/Simulink model reasonably simulates the operation of the devices and their voltage regulation activities. Real-world data for solar irradiance and household electrical demand can be loaded into the simulation. The battery control options are limited.
- **Scale:** Low
  - The simulated MG is a small implementation.

Testbed: MG, Class: OT Architecture, Subclass: Sensors

- **Devices:** PLCs, Smart Meters
  - PLCs are used to monitor the amount of generated solar power and the state of the battery. Smart Meters report household electrical demand.
- **Type:** Logical
  - The sensors are not implemented in the testbed, however, the data representing the state of the MG is generated by the MATLAB/Simulink software.
- **Fidelity:** N/A
  - The devices are not implemented in the simulation.

- **Scale:** N/A
  - The devices are not implemented in the simulation.

Testbed: MG, Class: OT Architecture, Subclass: Controllers

- **Devices:** PLCs
  - PLCs operate the battery control and the connection to the main electrical grid.
- **Type:** Logical
  - The control systems are not implemented in the testbed, however, simulated control is effectuated in the MATLAB/Simulink software.
- **Fidelity:** N/A
  - The devices are not implemented in the simulation.
- **Scale:** N/A
  - The devices are not implemented in the simulation.

Testbed: MG, Class: IT Architecture, Subclass: Communications Network

- **Devices:** WAN
  - A WAN sends sensory data and control commands amongst the MG devices.
- **Type:** Logical
  - The network communications is not implemented in the testbed, however, data is provided to the simulated processes in the MATLAB/Simulink software.
- **Fidelity:** N/A
  - The devices are not implemented in the simulation.
- **Scale:** N/A
  - The devices are not implemented in the simulation.

Testbed: MG, Class: IT Architecture, Subclass: Data Management

- **Devices:** Weather Forecaster, Load Forecaster

  - A Weather Forecaster system provides expected solar irradiance values to the MGCC. A Load Forecaster system provides expected household electrical demand to the MGCC.

- **Type:** Logical
  - The systems are not implemented, but simulated solar irradiance and electrical load forecast values are provided to the MGCC.
- **Fidelity:** N/A
  - The systems are not implemented in the simulation.
- **Scale:** N/A
  - The systems are not implemented in the simulation.

Testbed: MG, Class: Logistics Architecture

- **Systems:** MGCC
  - The MGCC receives sensory information describing the current state of the MG, solar irradiance forecasts, and electrical load forecasts. The MGCC uses this information to plan a battery control schedule.
- **Cycle:** Hourly
  - The battery control schedule is determined at each hour.
- **Type:** Simulation
  - Actual MGCC software is not used. Instead, the MGCC was implemented using CPLEX and integrated with the MATLAB/Simulink software.
- **Fidelity:** Medium
  - The MGCC is effective at controlling the battery to minimize the cost from purchasing electricity from the main electrical grid. A real-world MGCC would have the capacity to change the control more frequently than every hour and would have more control options than simply turning the battery `ON` or `OFF`.
- **Scale:** Medium

  - A real-world MGCC would receive more input data for basing its decisions and there would be more than one centralized module in the control software.

Testbed: MG, Class: Activity

- **Category:** Attack Impact Analysis (AIA)
  - The testbed has been used to demonstrate a procedure for analyzing how an MGCC will react when faced with attacks which falsify its inputs and outputs.

6.2.3 Air Traffic Control Testbed Classification

This section demonstrates and explains the POILA classification of the ATC testbed presented in Chapter 5.

Testbed: ATC, Class: Physical Process Architecture

- **Name:** Aircraft Traffic
  - The testbeds simulates air traffic in a controlled airspace.
- **Control:** Separation Control
  - Commands are issued to ensure there is safe separation between all the aircraft
- **Devices:** Aircraft
  - Aircraft are what is being controlled.
- **Type:** Simulation
  - The aircraft are simulated with the Euroscope software system.
- **Timing:** Simulated-time
  - Experiments are generally run in simulated time, however, Euroscope does have the ability to operate in real-time.
- **Fidelity:** Medium
  - The behaviour of the aircraft is relatively realistic and the aircraft respond to the commands issued from the simulated controller. Euroscope considers the physical capabilities of aircraft depending on their type. The aircraft could be more effective at not travelling near other aircraft.

- **Scale:** High
  - Euroscope can represent a large number of aircraft in an airspace.

Testbed: ATC, Class: OT Architecture, Subclass: Sensors

- **Devices:** PSR, SSR, ADS-B
  - PSR, SSR, and ADS-B installations receive positions of aircraft travelling within their coverage area.
- **Type:** Simulation
  - The behaviour of the devices is simulated with Python scripts.
- **Fidelity:** Low
  - Actual physical messages (waves, packets) are not sent between the sensors and the aircraft. When aircraft enter the coverage area of a sensor, FSD packets are generated with the data that would be reported by the sensor.
- **Scale:** High
  - The amount of PSR, SSR, and ADS-B devices can be configured to represent the reality of actual airspace. The experiments described in the work used a mix of real and created reporter installations around Montreal, Quebec.

Testbed: ATC, Class: OT Architecture, Subclass: Controllers

- **Devices:** Air Traffic Controller
  - A human Air Traffic Controller issues verbal commands to pilots. Although not technically a device, classifying a human controller as the OT controller emphasizes where the control commands originate in an ATC environment.
- **Type:** Simulation
  - The Air Traffic Controllers are simulated using the Euroscope software.
- **Fidelity:** Medium
  - The simulated Air Traffic Controllers can issue commands from a predefined set.
- **Scale:** High

  - The simulated Air Traffic Controllers can coordinate all of the air traffic in a simulation scenario.

Testbed: ATC, Class: IT Architecture, Subclass: Communications Network

- **Devices:** DDS Network
  - The DDS Network aggregates and forwards all generated PSR, SSR, and ADS-B positions reports to their necessary endpoints.
- **Type:** Simulation
  - The network is simulated using Python scripts.
- **Fidelity:** Low
  - Actual DDS packets are not sent across the network, instead Python dictionaries are used which represent the data that would be in a DDS packet.
- **Scale:** Low
  - The size of the DDS network is simplified by a large-degree.

Testbed: ATC, Class: IT Architecture, Subclass: Data Management

- **Devices:** Parser, Publisher, Subscriber, RDF Database
  - The Parser-Publisher-Subscriber pipeline is used to convert the reported flight positions (in FSD format) into a simplified DDS packet format. The RDF Database houses the DDS-to-RDF converted data which describes the aircraft traffic in a simulation scenario.
- **Type:** Simulation
  - The Parser-Publisher-Subscriber pipeline is simulated using custom built software using C and Python. The RDF database using the GraphDB software hosted on a local machine.
- **Fidelity:** Medium
  - The conversion of FDS data generated by Euroscope into RDF data reasonably captures the reality of the air traffic in the simulation.

- **Scale:** Low
  - The amount of RDF data generated is low compared to what would be needed in a real-world operational system.

Testbed: ATC, Class: Logistics Architecture

- **Systems:** Flight Planning, Route Scheduling
  - Planning systems determine things such as the arrival times, departure times, frequency, and routes of flights.
- **Cycle:** Daily
  - Planned flights are generally updated on a daily basis
- **Type:** None
  - These activities are not represented by this testbed. Instead, flight plans are loaded into Euroscope prior to performing a simulation scenario.
- **Fidelity:** N/A
  - Systems are not represented.
- **Scale:** N/A
  - Systems are not represented.

Testbed: ATC, Class: Activity

- **Category:** Defensive Tools Analysis (DTA)
  - The testbed has been used to create an ontology-based (i.e. rule-based) approach for detecting ghost aircraft attacks from falsified ADS-B messages.

6.3 Analysis and Usage

The classification of the three testbed case studies in the previous section (Section 6.2) demonstrates how the POILA taxonomy can capture the details of CI-CPS cybersecurity testbeds in a clear and concise manner. This use case can help efficiently capture the high level details of a testbed implementation and identify how it can be utilized.

The other intended use case for this taxonomy is that it can be used to generate testbed implementation requirement specifications. Through participating in the three testbed development projects outlined in Chapters 3-5, it became apparent that there is need for a clearer process to aid researchers develop CI-CPS testbeds. Much of the effort spent towards developing these testbeds was used to develop new features which had no impact on the outcome of the conducted experiments. There is a tendency for a desire to continually better reflect the operations of a real-world system by adding more capabilities to a testbed, without pausing to reflect upon what is the intended purpose that the testbed is meant to serve. This process instead proposes to first outline the intended research question and derive experiment requirements. Then use these experiment requirements to develop a blueprint of the testbed, using POILA, that is to be built.

The remainder of this section outlines the steps of the proposed process for developing a CI-CPS cybersecurity testbed using the POILA taxonomy.

1) Define reseach question. The process begins with identifying an open cybersecurity research challenge in a given CI-CPS domain where testbed-based experimentation can help provide an answer to the question. Broadly speaking, this chapter identifies four main activities that can be carried out with a testbed and the research question will likely be a variation of these activities. These identified activities are; Attack Impact Analysis (AIA), Infrastructure Vulnerability Analysis (IVA), Defensive Tools Analysis (DTA), and, Education and Training (E&T). Cybersecurity research questions generally fall into one of these categories, however, if some other testbed use case should emerge, the remaining steps can generally be applied without modifying the POILA taxonomy.

2) Define experiment requirements. Next, it is necessary to identify what are the intended experiments to be conducted in order to address the research questions and what are the requirements needed to enable the experiments. To provide the intuition behind this process, some issues to consider are provided as examples. For AIA research, some aspects to identify include; the normal operating criteria, the considered attacks, the equipment/processes being considered as targets, the required actions to carry out the attack, etc. For IVA research, some aspects to identify include; the equipment/protocols being considered as targets, the make/model/version of equipment/protocols being targeted, whether equipment can be emulated, etc. For DTA research, some aspects to identify include; the attacks that are to be considered, the attacks that are out of scope, the occurrence of attacks, the process for generating attacks, the actions that should be taken when an attack is detected, etc. For E&T research, some aspects to identify include; the intended users of the testbed, the

intended use cases for the users, the duration of intended exercises, the equipment/processes that the users will interact with, etc.

3) Create testbed blueprint with POILA. The next step, which is the main intended contribution of the proposed process, is to use the POILA taxonomy to develop a blueprint of what elements the intended testbed should be comprised of, to meet the experiment requirements. This step is crucial, as it generates a design artefact which reflects the intended specifications of the testbed. Each of the class and property elements of the POILA taxonomy are to be filled in with the desired components. Before continuing with developing the testbed, there is a need to reflect as to whether this design is attainable given the time, resources, and expertise available. This initial design may not be possible and there may be a need to identify areas where there can be simplifications to the implementation or there may even be a need to change the experiment requirements. For instance, if a particular experiment requires a high performance simulator or a specific piece of hardware that is too expensive to acquire, there is a need to consider if some sacrifice to the realism of such components can be tolerated. Ideally, the testbed's POILA blueprint can be formulated in a manner to enable the intended experiments given the capabilities available at hand. If there is no conceivable way to implement a particular testbed feature, there is a need to return to the previous step and modify the experiment requirements or even to potentially contemplate an entirely new research question.

4) Develop testbed and iterate. Upon determining a viable testbed blueprint, reflected with POILA, the next step is to begin the construction of the testbed. It is very likely that after spending some time developing the testbed, some of the specifications are not attainable. Perhaps this could be due to many factors, such as, budget restrictions, a gap in the expertise required to implement certain components, a lack of realism from a simulator, as well as countless other reasons. It is again necessary to reflect if there is a need to change the scope of the POILA blueprint or perhaps to adjust the experiment requirements. This proposed process is thus recommended to be carried out in an iterative manner and requires updating the POILA blueprint at regular time intervals.

5) Perform experiments and consider next phase. After constructing the testbed to meet the desired specifications it is time to execute the experiments, gather results, and analyze the findings. As with most scientific inquiry, it is likely that the conducted experiments do not completely resolve the challenges from the initial research question, and new questions emerge. Perhaps new experiments can be conducted with the existing testbed infrastructure,

however, it is likely that some upgrades to the testbeds are desired. The process can be repeated, ideally reusing as much of the testbed's infrastructure as possible and new testbed target features can be added to the POILA blueprint.

6.4 Conclusion

This chapter presents the POILA taxonomy, which is used to classify the characteristics of CI-CPS cybersecurity testbeds. The taxonomy organizes testbed characteristics along 5 main classes; Physical Process Architecture, OT Architecture, IT Architecture, Logistics Architecture, and Activity Classification. These classes provide distinct boundaries for depicting the systems represented in CI-CPS cybersecurity testbeds and their capabilities. This chapter applies the POILA taxonomy to the NPP, MG, and ATC testbeds described in Chapters 3-5 respectively, to more precisely illustrate the capabilities and limitations of the testbeds. Additionally, this chapter proposes a process for developing testbeds using the POILA taxonomy as a guiding framework. The process involves identifying the experiment requirements needed to investigate a particular research question, using POILA to develop a blueprint of the testbed to be developed which meets the needs of the experiments, and iteratively developing the testbed to meet the POILA design artefact.

Limitations

The POILA taxonomy utilizes a clear distinction between OT and IT systems, however, the line between these systems is increasingly blurred as OT networks are beginning to resemble IT networks. The POILA taxonomy in its current form is limited in capturing this convergence of IT and OT networks.

Future Work

The use case of using the POILA taxonamy for classifying testbeds can be applied to a larger set of CI-CPS cybersecurity testbeds, outside of the testbeds presented in this dissertation. This could lead to a broad-scale work which surveys testbeds developed by the scientific community across a number of CI-CPS disciplines to find trends in existing developed testbeds and identify where there is room to make novel testbeds.

The use case of using the POILA taxonamy as a framework for developing a new testbed, or adding new features to an existing testbed, from an original research question can be also explored. This type of exercise would help validate the proposed process and help identify

refinements to be applied to the taxonomy. It is likely that through using the proposed framework, some updates to POILA classes and properties will become evident.

CHAPTER 7 CONCLUSION

Through the development of the NPP, MG, ATC testbeds described in this work, we have been able to address open cybersecurity issues related to these respective disciplines and have created a taxonomy for classifying CI-CPS cybersecurity testbeds to be used as part of the testbed development process. This work has taken steps towards securing CI-CPS environments, however, as the technical and threat landscape of these systems continues to evolve, there is ongoing work to done by researchers and practitioners.

This closing chapter is organized as follows; Section 7.1 summarizes the contributions of this work, Section 7.2 assesses this work's limitations, Section 7.3 outlines directions for future study, and Section 7.4 concludes this dissertation.

7.1 Summary

Chapter 1 demonstrates the growing threats facing CI-CPS environments from motivated cyber-capable adversaries and introduced how testbeds can be utilized to perform experimentation to address open cybersecurity problems. CI-CPS environments increasingly incorporate digital based ICT to aid in the operation of physical processes, resulting in an ever-increasing attack surface available to attackers. Researchers have a role in securing CI systems by identifying cybersecurity concerns, constructing testbeds to perform experiments to address these concerns, and ultimately proposing solutions based on the findings. This work particularly focuses on developing testbeds for investigating issues related to NPPs, MGs, and ATC systems. While these are specific examples of CI-CPS environments, by demonstrating how these testbeds have been constructed contributes to the dissemination of testbed design techniques amongst the scientific community.

Chapter 2 reviews the relevant literature and provides background information on CI-CPS cybersecurity concepts. A CI-CPS environment is often organized in a layered architecture based on the function and timing of its various control-loops. This shields mission-critical control and sensing devices of physical processes to some degree, however, as there is an increased convergence between IT and OT systems, the reliable operation of physical processes face an increased exposure to cyberthreats. Testbeds can be used to better understand how cyberattacks against IT and OT architecture can affect physical processes by modelling the physical processes and digital infrastructure of CI-CPS environment through simulations, real-devices, emulated-devices, or some combination of these techniques. In testbed settings,

the actions of attackers can be modelled and executed without causing any real-world impact, allowing researchers to perform repeatable experiments.

A NPP testbed is presented in Chapter 3, which replicates Levels 2 and 3 of the Defense-in-Depth architecture recommended by the IAEA. The prominent feature of this testbed is a scaled down physical BLC system, controlled by physical PLCs, which has been integrated with the Asherah nuclear simulation software, developed in MATLAB/Simulink. A simulated MITM attack is shown which passes falsified process values to PLCs controlling the water level in the scaled-down representation of physical water boilers, causing the water level to drop. This work presents the effects of the attack and the generated data has been used in a research project in collaboration with the IAEA.

Chapter 4 presents a simulated MG testbed, developed in MATLAB/Simulink, which is controlled by a custom MILP control algorithm, implemented in CPLEX. This chapter proposes a framework for uncovering what the effect of attacking the input and output data of the control algorithm will have on the economic operations of the MG. Manual attacks which modify the state data provided to the controller and the control actions performed by the controller are conducted in the simulation. Also, RL agents are trained to find ideal falsified values to pass to the controller to have the most negative effect on the effectiveness of the controller. The most effective attacks prevent the controller from discharging electricity from the battery reserves, forcing any shortfalls in electricity production to be purchased from the main electrical grid. This framework can be reused to discover vulnerabilities in other ICS control algorithms.

An ATC testbed which simulates air traffic in a controlled airspace and the corresponding generated position reports from PSR, SSR, and ADS-B devices, is discussed in Chapter 5. Falsified ADS-B messages are injected into the testbed to simulate Ghost Aircraft attacks, which can be carried out by capable attackers using SDR devices, causing falsified aircraft to appear on ATC display terminals. The generated PSR, SSR, and ADS-B position reports in the simulation are converted into RDF data and are used to populate an ontological database. This data is reasoned upon by an ontology-based anomaly detection platform, known as ATC-Sense, by using SPARQL queries to detect malicious ADS-B messages which violate aviation-related constraints. This work demonstrates the feasibility of using an ontology-based approach in detecting cyberattacks in a near-real time setting. The computational performance of the detection approach is measured and this work concludes that future work will need to increase the speed at which ontology-based anomaly detection approaches using SPARQL queries occur.

Chapter 6 demonstrates a taxonomy, known as POILA, for classifying CI-CPS cybersecurity testbeds and applies this taxonomy to the three testbeds presented in Chapters 3-5. POILA categorizes testbeds along five main classes; Physical Process Architecture, OT Architecture, IT Infrastructure, Logistics Architecture, and Activity Classification. The POILA taxonomy is unique to other taxonomies proposed by the community in that it captures the role which the IT and OT systems play in controlling a physical process. Such a classification method allows researchers to quickly assess the components and capabilities of testbeds, helps determine the work conducted by others, and ultimately helps identify where there is room to make scientific contributions. Additionally, POILA is proposed as a tool to create a testbed design artefact to streamline the process of constructing testbeds. The proposed process suggests to formulate the requirements of intended experiments which address open cybersecurity challenges and use POILA to form a blueprint for the intended design of a testbed. This helps define the testbed requirements and focus the effort required in constructing a testbed. The contributions of this chapter are used to unify the work conducted in this dissertation, in addressing the problem statement laid out at the beginning of this work.

7.2 Limitations

The limitations of the research specific to the NPP, MG, and ATC testbeds, as well as the POILA taxonomy have been previously discussed in Sections 3.4, 4.4, 5.4, and 6.4, respectively. As a result, this section highlights the limitations of this work as a whole.

The ultimate goal of using testbeds is to tackle open problems within a particular domain, however, a researcher must first devote a considerable amount of their available time and resources to implementing features in a testbed. This challenge was persistent throughout this research and the time spent on performing the actual experiments was limited. Considering this, this work has taken effort to demonstrate the architecture of the developed testbeds in hopes that others can learn from the lessons of this work and increase their efficiency in developing future testbeds.

Additionally, this work covers a lot of breadth by investigating cybersecurity issues related to the domains of NPPs, MGs, and ATC systems. As a result, there is a sacrifice to the depth of the investigations into each of these disciplines and the presented findings only scratch the surface of what there is to be done in these CI-CPS settings. Nevertheless, the testbeds presented in this work are part of ongoing research projects and the implemented features will be reused by future researchers.

7.3 Future Research

The future work related to the NPP, MG, and ATC testbeds, as well as the POILA taxonomy has been previously discussed in Sections 3.4, 4.4, 5.4, and 6.4, respectively. This section introduces directions for future research which is applicable across all of the presented work.

A growing future trend discussed in the introduction to this work in Chapter 1, was the rise of general purpose malware being used to compromise ICS systems, such as NPPs or MGs. This has been considered as out of scope for this work, as the conducted attacks considered that an attacker has already performed the necessary steps to be in a position to execute attacks to compromise the systems. Future work for CI-CPS testbeds should investigate how malware may be able to infect ICT components and be used to lay the groundwork for attacks against physical processes. It would also be interesting to configure the ICS testbeds to be honey-pots, that appear to be real-systems to some extent, so that they may retrieve some samples of real-world attacks. It is unlikely that such a honey-pot would be targeted with sophisticated malware, like CrashOverride [23] or Triton [24], however, they may retrieve some examples of untargeted attacks to be analyzed.

This work has considered the physical effects which can occur from attacks (NPP and MG testbed in Chapters 3 and 4 respectively) and the development of defensive tools (ATC testbed in Chapter 5), however, in a real-world setting, the execution of attacks and the use of defensive countermeasures occur at the same time. These testbeds can be upgraded so that there are numerous offensive and defensive techniques than can be employed enabling the ability to perform Red vs. Blue team exercises. The idea will be to see how developed defensive tools perform against sophisticated attacks conducted by some offensive operator with decision making capabalities. To go even further, experiments can be done with AI-based offensive and defensive agents who can adapt their strategies in an automated fashion.

7.4 Conclusion

As the threat landscape facing CI continues to evolve, so must the efforts of cybersecurity researchers and practitioners. We are at the beginning of a new era, where groups can launch increasingly sophisticated cyber-based operations from across the globe with relative security and anonymity. CI are prime targets for nations to exert force over their rivals, cyber-criminals to extort and manipulate in search of financial gains, and ideological groups to broadcast their message. This work has aimed to convey the utility derived from using testbeds in performing cybersecurity experimentation to better understand emerging attacker capabilities and to aid in the development of innovative defensive procedures. The insights

gained from this work, through conducting experiments with purpose-built testbeds, have allowed us to propose a testbed development framework for enhancing the efficacy of the testbed development process and ultimately provides a resource to the community to help combat emerging cyberthreats in an efficient manner. Only in anticipating the most elusive, sophisticated, and potentially devastating attacks, are defenders able to adequately prepare for this new age of cyberconflict.

REFERENCES

[1] Public Safety Canada, "Canada's Critical Infrastructure," May 2020, [Online]. Available: https://www.publicsafety.gc.ca/cnt/ntnl-scrt/crtcl-nfrstrctr/cci-iec-en.aspx [Accessed: November 2020].

[2] G. Brown, M. Carlyle, J. Salmerón, and K. Wood, "Defending Critical Infrastructure," *INFORMS Journal on Applied Analytics*, vol. 36, no. 6, pp. 530–544, 2006. [Online]. Available: https://pubsonline.informs.org/doi/abs/10.1287/inte.1060.0252

[3] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly, "Identifying, understanding, and analyzing critical infrastructure interdependencies," *IEEE Control Systems Magazine*, vol. 21, no. 6, pp. 11–25, 2001.

[4] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," *Journal of Computer and System Sciences*, vol. 80, no. 5, pp. 973 – 993, 2014, special Issue on Dependable and Secure Computing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000014000178

[5] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution," in *Design Automation Conference*, 2010, pp. 731–736.

[6] B. Green, R. Derbyshire, W. Knowles, J. Boorman, P. Ciholas, D. Prince, and D. Hutchison, "ICS Testbed Tetris: Practical Building Blocks Towards a Cyber Security Resource," in *13th USENIX Workshop on Cyber Security Experimentation and Test (CSET)*, 2020.

[7] S. McLaughlin, C. Konstantinou, X. Wang, L. Davi, A. Sadeghi, M. Maniatakos, and R. Karri, "The Cybersecurity Landscape in Industrial Control Systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1039–1057, 2016.

[8] G. Murray, M. N. Johnstone, and C. Valli, "The convergence of IT and OT in critical infrastructure," in *15th Australian Information Security Management Conference*, 2017, pp. 149–155.

[9] W. Knowles, D. Prince, D. Hutchison, J. F. P. Disso, and K. Jones, "A survey of cyber security management in industrial control systems," *International Journal of Critical Infrastructure Protection*, vol. 9, pp. 52 – 80, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1874548215000207

[10] J. Slay and M. Miller, "Lessons Learned from the Maroochy Water Breach," in *Critical Infrastructure Protection*, E. Goetz and S. Shenoi, Eds. Boston, MA: Springer US, 2008, pp. 73–82.

[11] M. D. Adams and J. Weiss, "Malicious Control System Cyber Security Attack Case Study–Maroochy Water Services, Australia," MITRE, Tech. Rep., August 2008, [Online]. Available: https://www.mitre.org/publications/technical-papers/malicious-control-system-cyber-security-attack-case-study-maroochy-water-services-australia [Accessed: November 2020].

[12] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the Slammer worm," *IEEE Security Privacy*, vol. 1, no. 4, pp. 33–39, 2003.

[13] G. Richards, "Hackers vs slackers - [control security]," *Engineering & Technology*, vol. 3, no. 19, pp. 40–43, 2008.

[14] A. Bindra, "Securing the Power Grid: Protecting Smart Grids and Connected Power Systems from Cyberattacks," *IEEE Power Electronics Magazine*, vol. 4, no. 3, pp. 20–27, 2017.

[15] CNN, "Sources: Staged cyber attack reveals vulnerability in power grid," September 2007, [Online]. Available: http://www.cnn.com/2007/US/09/26/power.at.risk/ [Accessed: November 2020].

[16] F. Khorrami, P. Krishnamurthy, and R. Karri, "Cybersecurity for Control Systems: A Process-Aware Perspective," *IEEE Design & Test*, vol. 33, no. 5, pp. 75–83, 2016.

[17] D. Locaria and J. R. Wool, "Cyberattacks Threaten Critical Infrastructure," *Risk Management*, vol. 62, no. 4, pp. 10–11, 05 2015. [Online]. Available: https://search.proquest.com/docview/1689363559?accountid=40695

[18] R. M. Lee (SANS), "Closing the Case on the Reported 2008 Russian Cyber Attack on the BTC Pipeline," Jun. 2015, [Online]. Available: https://www.sans.org/blog/closing-the-case-on-the-reported-2008-russian-cyber-attack-on-the-btc-pipeline/ [Accessed: November 2020].

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

[20] T. M. Chen and S. Abu-Nimeh, "Lessons from Stuxnet," *Computer*, vol. 44, no. 4, pp. 91–93, 2011.

[21] Reuters, "Jets vanishing from Europe radar linked to war games," Jun. 2014, [Online]. Available: https://www.reuters.com/article/us-europe-airplanes-safety/jets-vanishing-from-europe-radar-linked-to-war-games-idUSKBN0EO1CW20140613 [Accessed: November 2020].

[22] SANS ICS, "Analysis of the Cyber Attack on the Ukrainian Power Grid," March 2016, [Online]. Available: https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf [Accessed: November 2020].

[23] Dragos, "Crashoverride: Analysis of the Threat to Electrical Grid Operations," March 2016, [Online]. Available: https://www.dragos.com/wp-content/uploads/CrashOverride-01.pdf [Accessed: November 2020].

[24] S. Mansfield-Devine, "Critical infrastructure: understanding the threat," *Computer Fraud & Security*, vol. 2018, no. 7, pp. 16 – 20, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1361372318300654

[25] A. Greenberg (Wired), "Unprecedented Malware Targets Industrial Safety Systems in the Middle East," December 2017, [Online]. Available: https://www.wired.com/story/triton-malware-targets-industrial-safety-systems-in-the-middle-east/ [Accessed: November 2020].

[26] M. Galiardi, A. Gonzales, J. Thorpe, E. Vugrin, R. Fasano, and C. Lamb, "Cyber Resilience Analysis of SCADA Systems in Nuclear Power Plants," in *International Conference on Nuclear Engineering*, August 2020. [Online]. Available: https://doi.org/10.1115/ICONE2020-16071

[27] C. Cimpanu (ZDNet), "Confirmed: North Korean malware found on Indian nuclear plant's network," October 2019, [Online]. Available: https://www.zdnet.com/article/confirmed-north-korean-malware-found-on-indian-nuclear-plants-network/ [Accessed: November 2020].

[28] C. C. Lamb, R. E. Fasano, and T. Ortiz, "Advanced Malware and Nuclear Power: Past, Present, and Future," in *International Conference on Nuclear Security (ICONS)*. International Atomic Energy Agency (IAEA), Feb. 2020.

[29] C. Bing (Reuters), "Suspected Russian hackers spied on U.S. Treasury emails - sources," December 2020, [Online]. Available: https://www.reuters.com/article/us-usa-cyber-treasury-exclsuive/suspected-russian-hackers-spied-on-u-s-treasury-emails-sources-idUKKBN28N0PG?edition-redirect=uk [Accessed: March 2021].

[30] S. Halpern (The New Yorker), "After the SolarWinds hack, we have no idea what cyber dangers we face," January 2021, [Online]. Available: https://www.newyorker.com/news/daily-comment/after-the-solarwinds-hack-we-have-no-idea-what-cyber-dangers-we-face [Accessed: March 2021].

[31] I. Jibilian and K. Canales (Business Insider), "Here's a simple explanation of how the massive SolarWinds hack happened and why it's such a big deal," February 2021, [Online]. Available: https://www.businessinsider.com/solarwinds-hack-explained-government-agencies-cyber-security-2020-12 [Accessed: March 2021].

[32] United States Cybersecurity & Infrastructure Security Agency, "Alert (AA20-352A) Advanced Persistent Threat Compromise of Government Agencies, Critical Infrastructure, and Private Sector Organizations," February 2021, [Online]. Available: https://us-cert.cisa.gov/ncas/alerts/aa20-352a [Accessed: March 2021].

[33] Allianz, "Cyber attacks on critical infrastructure," 2015, [Online]. Available: https://www.agcs.allianz.com/news-and-insights/expert-risk-articles/cyber-attacks-on-critical-infrastructure.html [Accessed: November 2020].

[34] D. Lohrmann (Government Technology), "Hacking Critical Infrastructure is Accelerating and More Destructive," April 2015, [Online]. Available: https://www.govtech.com/blogs/lohrmann-on-cybersecurity/Hacking-Critical-Infrastructure-is-Accelerating-and-More-Destructive.html [Accessed: November 2020].

[35] W. Ashford (Computer Weekly), "Critical infrastructure commonly hit by destructive cyber attacks, survey reveals," April 2015, [Online]. Available: https://www.computerweekly.com/news/4500243886/Critical-infrastructure-commonly-hit-by-destructive-cyber-attacks-survey-reveals [Accessed: November 2020].

[36] L. Tabansky, "Critical infrastructure protection against cyber threats," *Military and Strategic Affairs*, vol. 3, no. 2, 2011.

[37] A. Gendron and M. Rudner, "Assessing Cyber Threats to Canadian Infrastructure," Canadian Security Intelligence Service (CSIS), Tech. Rep., Mar. 2012, [Online]. Available: https://www.canada.ca/content/dam/csis-scrs/documents/publications/CyberTrheats_AO_Booklet_ENG.pdf [Accessed: November 2020].

[38] A. Lemay, J. Calvet, F. Menet, and J. M. Fernandez, "Survey of publicly available reports on advanced persistent threat actors," *Computers & Security*, vol. 72, pp. 26 –

59, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404817301608

[39] CTV News, "STM targeted by ransomware attack, online platforms down," Oct. 2020, [Online]. Available: https://montreal.ctvnews.ca/stm-targeted-by-ransomware-attack-online-platforms-down-1.5152323 [Accessed: November 2020].

[40] K. Thomas (CTV News), "A week after STM cyberattack, security breach discovered at Jewish General Hospital," Oct. 2020, [Online]. Available: https://montreal.ctvnews.ca/a-week-after-stm-cyberattack-security-breach-discovered-at-jewish-general-hospital-1.5166141 [Accessed: November 2020].

[41] J. P. Farwell and R. Rohozinski, "Stuxnet and the Future of Cyber War," *Survival*, vol. 53, no. 1, pp. 23–40, 2011. [Online]. Available: https://doi.org/10.1080/00396338.2011.555586

[42] R. A. Lika, D. Murugiah, S. N. Brohi, and D. Ramasamy, "NotPetya: Cyber Attack Prevention through Awareness via Gamification," in *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, 2018, pp. 1–6.

[43] Q. Chen and R. A. Bridges, "Automated Behavioral Analysis of Malware: A Case Study of WannaCry Ransomware," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2017, pp. 454–460.

[44] G. Falco, C. Caldera, and H. Shrobe, "IIoT Cybersecurity Risk Modeling for SCADA Systems," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4486–4495, 2018.

[45] M. Lezzi, M. Lazoi, and A. Corallo, "Cybersecurity for Industry 4.0 in the current literature: A reference framework," *Computers in Industry*, vol. 103, pp. 97 – 110, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166361518303658

[46] N. Kaloudi and J. Li, "The AI-Based Cyber Threat Landscape: A Survey," *ACM Comput. Surv.*, vol. 53, no. 1, Feb. 2020. [Online]. Available: https://doi.org/10.1145/3372823

[47] K. Chung, Z. T. Kalbarczyk, and R. K. Iyer, "Availability Attacks on Computing Systems through Alteration of Environmental Control: Smart Malware Approach," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, ser. ICCPS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–12. [Online]. Available: https://doi.org/10.1145/3302509.3311041

[48] C. Siaterlis and B. Genge, "Cyber-Physical Testbeds," *Commun. ACM*, vol. 57, no. 6, p. 64–73, Jun. 2014. [Online]. Available: https://doi.org/10.1145/2602575

[49] M. M. Yamin, B. Katt, and V. Gkioulos, "Cyber ranges and security testbeds: Scenarios, functions, tools and architecture," *Computers & Security*, vol. 88, p. 101636, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404819301804

[50] B. Van Leeuwen, V. Urias, J. Eldridge, C. Villamarin, and R. Olsberg, "Cyber security analysis testbed: Combining real, emulation, and simulation," in *44th Annual 2010 IEEE International Carnahan Conference on Security Technology*, 2010, pp. 121–126.

[51] T. J. Williams, "The purdue enterprise reference architecture," *Computers in Industry*, vol. 24, no. 2, pp. 141 – 158, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0166361594900175

[52] U. P. D. Ani, H. M. He, and A. Tiwari, "Review of cybersecurity issues in industrial critical infrastructure: manufacturing in perspective," *Journal of Cyber Security Technology*, vol. 1, no. 1, pp. 32–74, 2017. [Online]. Available: https://doi.org/10.1080/23742917.2016.1252211

[53] P. Ackerman, *Industrial Cybersecurity: Efficiently secure critical infrastructure systems*. Packt Publishing, 2017.

[54] P. K. Garimella, "IT-OT Integration Challenges in Utilities," in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, 2018, pp. 199–204.

[55] S. Samonas and D. Cross, "The CIA Strikes Back: Redefining Confidentiality, Integrity and Availability in Security," *Journal of Information System Security*, vol. 10, no. 3, pp. 141 – 158, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0166361594900175

[56] L. Williams, G. McGraw, and S. Migues, "Engineering Security Vulnerability Prevention, Detection, and Response," *IEEE Software*, vol. 35, no. 5, pp. 76–80, 2018.

[57] M. Riaz, S. Elder, and L. Williams, "Systematically Developing Prevention, Detection, and Response Patterns for Security Requirements," in *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, 2016, pp. 62–67.

[58] FireEye, "What is a Zero-Day Exploit?" [Online]. Available: https://www.fireeye.com/current-threats/what-is-a-zero-day-exploit.html [Accessed: June 2021].

[59] L. Bilge and T. Dumitraş, "Before We Knew It: An Empirical Study of Zero-Day Attacks in the Real World," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 833–844. [Online]. Available: https://doi.org/10.1145/2382196.2382284

[60] T. Armerding (Forbes), "Apple's $1 Million Bug Bounty Could Launch Arms Race For Zero Days," Aug. 2019, [Online]. Available: https://www.forbes.com/sites/taylorarmerding/2019/08/15/bug-bounties-go-big/?sh=77812d921eeb [Accessed: June 2021].

[61] C. Day, "Chapter 18 - Intrusion Prevention and Detection Systems," in *Computer and Information Security Handbook*, J. R. Vacca, Ed. Boston: Morgan Kaufmann, 2009, pp. 293 – 306. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780123743541000182

[62] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "NIST Special Publication 800-82 - Revision 2: 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)," NIST, Tech. Rep., May 2015, [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-82r2 [Accessed: October 2020].

[63] E. Hutchins, M. Cloppert, and R. Amin, "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains," in *6th International Conference on Information Warfare and Security*, 2011, pp. 113–125.

[64] Lockheed Martin, "Gaining the Advantage: Applying Cyber Kill Chain Methodology to Network Defense," [Online]. Available: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/Gaining_the_Advantage_Cyber_Kill_Chain.pdf [Accessed: November 2020].

[65] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK: Design and Philosophy," MITRE, Tech. Rep., July 2018, [Online]. Available: https://www.mitre.org/sites/default/files/publications/pr-18-0944-11-mitre-attack-design-and-philosophy.pdf [Accessed: October 2020].

[66] MITRE, "ATT&CK," Jun. 2020, [Online]. Available: https://attack.mitre.org/ [Accessed: November 2020].

[67] ——, "ATT&CK for Industrial Control Systems," Jun. 2020, [Online]. Available: https://collaborate.mitre.org/attackics/index.php/Main_Page [Accessed: November 2020].

[68] ——, "Common Vulnerabilities and Exposures (CVE)," Nov. 2020, [Online]. Available: https://cve.mitre.org/ [Accessed: November 2020].

[69] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-Simulation: A Survey," *ACM Comput. Surv.*, vol. 51, no. 3, May 2018. [Online]. Available: https://doi.org/10.1145/3179993

[70] K. Mets, J. A. Ojea, and C. Develder, "Combining Power and Communication Network Simulation for Cost-Effective Smart Grid Analysis," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1771–1796, 2014.

[71] R. B. Vaughn and T. Morris, "Addressing Critical Industrial Control System Cyber Security Concerns via High Fidelity Simulation," ser. CISRC '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2897795.2897819

[72] M. Frank, M. Leitner, and T. Pahi, "Design Considerations for Cyber Security Testbeds: A Case Study on a Cyber Security Testbed for Education," in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/-CyberSciTech)*, 2017, pp. 38–46.

[73] M. H. Cintuglu, O. A. Mohammed, K. Akkaya, and A. S. Uluagac, "A Survey on Smart Grid Cyber-Physical System Testbeds," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 446–464, 2017.

[74] S. Schwab and E. Kline, "Cybersecurity Experimentation at Program Scale: Guidelines and Principles for Future Testbeds," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2019, pp. 94–102.

[75] Y. An, C. Sollima, R. Uddin, D. Chen, Z. Kalbarczyk, T. Yardley, and W. Sanders, "A test bed for digital I&C and cyber security for NPPs," in *9th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (NPIC&HMIT)*, 2015, pp. 2496–2502.

[76] Y. An, C. Sollima, R. Uddin, D. Chen, and Z. Kalbarczyk, "Recent results from a testbed for the reliability of the digital instrumentation and control systems and cyber-security in nuclear power plants," in *10th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (NPIC&HMIT)*, 2017, pp. 2139–2147.

[77] Y. Kim, I. Moon, and S. Lee, "A design of cyber security test-bed for DPPS and PMAS in Korean operating nuclear power plant," in *2016 16th International Conference on Control, Automation and Systems (ICCAS)*, 2016, pp. 1480–1483.

[78] C. Lee, C. Lee, J. Choi, and P. Seong, "Development of a Demonstrable Nuclear Cyber Security Test-Bed and Application Plans," in *Transaction of the Korean Nuclear Society Spring Meeting*, 2019.

[79] R. A. B. E. Silva, K. Shirvan, P. J. R. C., and R. P. Marques, "Development of the Asherah Nuclear Power Plant Simulation for Cyber Security Assessment," in *International Conference on Nuclear Security (ICONS)*. International Atomic Energy Agency (IAEA), Feb. 2020.

[80] F. Zhang, H. A. D. E. Kodituwakku, J. W. Hines, and J. Coble, "Multilayer Data-Driven Cyber-Attack Detection System for Industrial Control Systems Based on Network, System, and Process Data," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4362–4369, 2019.

[81] F. Zhang, J. W. Hines, and J. B. Coble, "A Robust Cybersecurity Solution Platform Architecture for Digital Instrumentation and Control Systems in Nuclear Power Facilities," *Nuclear Technology*, vol. 206, no. 7, pp. 939–950, 2020. [Online]. Available: https://doi.org/10.1080/00295450.2019.1666599

[82] F. Zhang and J. B. Coble, "Robust localized cyber-attack detection for key equipment in nuclear power plants," *Progress in Nuclear Energy*, vol. 128, p. 103446, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0149197020301980

[83] A. Nelson, S. Chakraborty, Dexin Wang, P. Singh, Qiang Cui, Liuqing Yang, and S. Suryanarayanan, "Cyber-physical test platform for microgrids: Combining hardware, hardware-in-the-loop, and network-simulator-in-the-loop," in *2016 IEEE Power and Energy Society General Meeting (PESGM)*, 2016, pp. 1–5.

[84] S. Poudel, Z. Ni, and N. Malla, "Real-time cyber physical system testbed for power system security and control," *International Journal of Electrical*

Power & Energy Systems, vol. 90, pp. 124 – 133, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061516312911

[85] E. Xypolytou, J. Fabini, W. Gawlik, and T. Zseby, "The FUSE testbed: establishing a microgrid for smart grid security experiments," *Elektrotechnik und Informationstechnik*, vol. 134, no. 1, pp. 30–35, Feb. 2017.

[86] M. H. Cintuglu, T. Youssef, and O. A. Mohammed, "Development and Application of a Real-Time Testbed for Multiagent System Interoperability: A Case Study on Hierarchical Microgrid Control," *IEEE Transactions on Smart Grid*, vol. 9, no. 3, pp. 1759–1768, 2018.

[87] A. Ashok, S. Sridhar, T. Becejac, T. Rice, M. Engels, S. Harpool, M. Rice, and T. Edgar, "A Multi-level Fidelity Microgrid Testbed Model for Cybersecurity Experimentation," in *12th USENIX Workshop on Cyber Security Experimentation and Test (CSET 19)*. Santa Clara, CA: USENIX Association, Aug. 2019. [Online]. Available: https://www.usenix.org/conference/cset19/presentation/ashok

[88] P. S. Sarker, V. Venkataramanan, D. S. Cardenas, A. Srivastava, A. Hahn, and B. Miller, "Cyber-Physical Security and Resiliency Analysis Testbed for Critical Microgrids with IEEE 2030.5," in *2020 8th Workshop on Modeling and Simulation of Cyber-Physical Energy Systems*, 2020, pp. 1–6.

[89] M. Strohmeier, M. Schäfer, R. Pinheiro, V. Lenders, and I. Martinovic, "On Perception and Reality in Wireless Air Traffic Communication Security," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1338–1357, 2017.

[90] A. Costin and A. Francillion, "Ghost in the Air (Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices," *Proceedings of Black Hat USA*, no. 8, pp. 1–12, 2012.

[91] M. Schäfer, V. Lenders, and I. Martinovic, "Experimental Analysis of Attacks on Next Generation Air Traffic Communication," *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS)*, pp. 253–271, 2013.

[92] A. Barreto, M. Hieb, and E. Yano, "Developing a complex simulation environment for evaluating cyber attacks," in *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, 2012.

[93] S. Amin, T. Clark, R. Offutt, and K. Serenko, "Design of a cyber security framework for ADS-B based surveillance systems," in *2014 Systems and Information Engineering Design Symposium (SIEDS)*, 2014, pp. 304–309.

[94] M. Monteiro, T. Sarmento, A. Barreto, P. Costa, and M. Hieb, "An integrated mission and cyber simulation for Air Traffic Control," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 2687–2692.

[95] A. R. Schmitt, C. Edinger, T. Mayer, J. Niederl, and T. Kiesling, "Simulation-supported aviation cyber-security risk analysis: a case study," *Computer*, vol. 10, pp. 517–530, 2019.

[96] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals of Operation Research*, vol. 153, pp. 235–256, 2007.

[97] R. G. Jeroslow, "The polynomial hierarchy and a simple model for competitive analysis," *Mathematical Programming*, vol. 32, pp. 146–164, 1985.

[98] A. Caprara, M. Carvalho, A. Lodi, and G. Woeginger, "Bilevel Knapsack with Interdiction Constraints," *INFORMS Journal on Computer*, vol. 28, no. 2, pp. 1–15, 2016.

[99] V. Kalashnikov, S. Dempe, G. Pérez-Valdés, N. Kalashnykova, and J.-F. Camacho-Vallejo, "Bilevel Programming and Applications," *Mathematical Problems in Engineering*, 2015.

[100] Z. Li, M. Shahidehpour, A. Alabdulwahab, and A. Abusorrah, "Bilevel Model for Analyzing Coordinated Cyber-Physical Attacks on Power Systems," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2260–2272, 2016.

[101] X. Liu and Z. Li, "Trilevel Modeling of Cyber Attacks on Transmission Lines," *IEEE Transactions on Smart Grid*, vol. 8, no. 2, pp. 720–729, 2017.

[102] Y. Yuan, Z. Li, and K. Ren, "Quantitative Analysis of Load Redistribution Attacks in Power Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, pp. 1731–1738, 2012.

[103] W. Zeng and M. Chow, "Optimal Tradeoff Between Performance and Security in Networked Control Systems Based on Coevolutionary Algorithms," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 7, pp. 3016–3025, 2012.

[104] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer, 1998.

[105] J. R. Bezerra, G. C. Barroso, R. P. S. Leão, and R. F. Sampaio, "Multiobjective Optimization Algorithm for Switch Placement in Radial Power Distribution Networks," *IEEE Transactions on Power Delivery*, vol. 30, no. 2, pp. 545–552, 2015.

[106] H. Ali and F. A. Khan, "Attributed multi-objective comprehensive learning particle swarm optimization for optimal security of networks," *Applied Soft Computing*, vol. 13, no. 9, pp. 3903 – 3921, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494613001397

[107] V. Viduto, C. Maple, W. Huang, and A. Bochenkov, "A multi-objective genetic algorithm for minimising network security risk and cost," in *2012 International Conference on High Performance Computing & Simulation (HPCS)*, 2012, pp. 462–467.

[108] T. Okimoto, N. Ikegai, K. Inoue, H. Okada, T. Ribeiro, and H. Maruyama, "Cyber security problem based on Multi-Objective Distributed Constraint Optimization technique," in *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, 2013, pp. 1–7.

[109] E. Eisenstadt and A. Moshaiov, "Novel Solution Approach for Multi-Objective Attack-Defense Cyber Games With Unknown Utilities of the Opponent," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 1, pp. 16–26, 2017.

[110] S. N. Hamilton, W. L. Miller, A. Ott, and O. S. Saydjari, "Challenges in Applying Game Theory to the Domain of Information Warfare," in *4th Information survivability workshop*, 2002.

[111] K. Lye and J. Wing, "Game strategies in network security," *International Journal of Information Security*, vol. 4, no. 1, pp. 71–86, 2005.

[112] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu, "A Survey of Game Theory as Applied to Network Security," in *43rd Hawaii International Conference on System Sciences*, 2010, pp. 1–10.

[113] A. Roy, D. S. Kim, and K. S. Trivedi, "Cyber security analysis using attack countermeasure trees," in *6th AnnualWorkshop on Cyber Security and Information Intelligence Research*, 2010.

[114] ——, "Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees," in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, 2012, pp. 1–12.

[115] S. Backhaus, R. Bent, J. Bono, R. Lee, B. Tracey, D. Wolpert, D. Xie, and Y. Yildiz, "Cyber-Physical Security: A Game Theory Model of Humans Interacting Over Control Systems," *IEEE Transactions on Smart Grid*, vol. 4, no. 4, pp. 2320–2327, 2013.

[116] C. Neal, D. Trask, K. Thiyagarajan, R. Doucet, P. Adamson, M. Daley, and J. Fernandez, "Advancements in Hardening the Cybersecurity Posture of Nuclear Power Plant Defence-in-Depth Network Architecture," in *International Conference on Nuclear Security (ICONS)*. International Atomic Energy Agency (IAEA), Feb. 2020.

[117] International Atomic Energy Agency (IAEA), "Computer Security Techniques For Nuclear Facilities: Draft Technical Guidance," August 2017, [Online]. Available: https://www-ns.iaea.org/downloads/security/security-series-drafts/tech-guidance/nst047.pdf [Accessed: November 2020].

[118] J. Dalheimer and D. Testa, *The Westinghouse Pressurized Water Reactor Nuclear Power Plant*. Westinghouse Electric Corporation, Water Reactor Division, 1984.

[119] Steffen Kuntoff (WikiCommons), "Nuclear power plant-pressurized water reactor-PWR," Oct. 2005, [Online]. Available: https://commons.wikimedia.org/wiki/File:Nuclear_power_plant-pressurized_water_reactor-PWR.png [Accessed: November 2020].

[120] W. Knowles, D. Prince, D. Hutchison, J. F. P. Disso, and K. Jones, "A survey of cyber security management in industrial control systems," *International Journal of Critical Infrastructure Protection*, vol. 9, pp. 52 – 80, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1874548215000207

[121] S. McLaughlin, C. Konstantinou, X. Wang, L. Davi, A. Sadeghi, M. Maniatakos, and R. Karri, "The Cybersecurity Landscape in Industrial Control Systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1039–1057, 2016.

[122] United States Nuclear Regulatory Commision, "Westinghouse Technology Systems Manual, Section 11.1, Steam Generator Water Level Control System," 2014, [Online]. Available: https://www.nrc.gov/docs/ML1122/ML11223A293.pdf [Accessed: November 2020].

[123] Mliu92 (WikiCommons), "Generic Nuclear PWR Steam Generator," Nov. 2014, [Online]. Available: https://commons.wikimedia.org/wiki/File:Generic_Nuclear_PWR_Steam_Generator.svg [Accessed: September 2020].

[124] International Atomic Energy Agency (IAEA), "Collaborative Research Project J02008: Enhancing Computer Security Incident Analysis at Nuclear Facilities," 2015, [Online]. Available: https://www.iaea.org/projects/crp/j02008 [Accessed: December 2020].

[125] U.S. Department of Energy Office of Scientific and Technical Inforamtion (OSTI): M. Rowland, "IAEA CRP on Enhancing Incident Response at Nuclear Facilities," 2019, [Online]. Available: https://www.osti.gov/servlets/purl/1643619 [Accessed: December 2020].

[126] Festo, "LabVolt Pressure, Flow, Level, and Temperature Process Learning Systems," 2020, [Online]. Available: https://www.labvolt.com/solutions/7_process_control/98-3531-00_pressure_flow_level_and_temperature_process_learning_systems [Accessed: November 2020].

[127] OWL Cyber Defense, "What is a Data Diode & How Do Data Diodes Work?" Jun. 2018, [Online]. Available: https://owlcyberdefense.com/blog/what-is-data-diode-technology-how-does-it-work/ [Accessed: November 2020].

[128] Wireshark, "About Wireshark," 2020, [Online]. Available: https://www.wireshark.org/index.html#aboutWS [Accessed: November 2020].

[129] S. W. Blume, *Electrical Power System Basics for the Nonelectrical Professional*, 2nd ed. Wiley-IEEE Press, 2017.

[130] United States Department of Energy (WikiCommons), "Final Report on the August 14, 2003 Blackout in the United States and Canada," http://www.ferc.gov/industries/electric/indus-act/reliability/blackout/ch1-3.pdf, Dec. 2008, [Online]. Available: https://en.wikipedia.org/wiki/File:Electricity_grid_simple-_North_America.svg [Accessed: November 2020].

[131] "Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations," U.S.-Canada Power System Outage Task Force, Tech. Rep., Apr. 2004, [Online]. Available: https://eta-publications.lbl.gov/sites/default/files/2003-blackout-us-canada.pdf [Accessed: November 2020].

[132] G. Sorebo, M. Echols, and M. Assante, *Smart Grid Cybersecurity: An End-to-End View of Security in the New Electrical Grid*. Boca Raton: CRC Press, 2012.

[133] E. Dall'Anese, H. Zhu, and G. B. Giannakis, "Distributed Optimal Power Flow for Smart Microgrids," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1464–1475, 2013.

[134] D. E. Olivares, A. Mehrizi-Sani, A. H. Etemadi, C. A. Cañizares, R. Iravani, M. Kazerani, A. H. Hajimiragha, O. Gomis-Bellmunt, M. Saeedifard, R. Palma-Behnke, G. A. Jiménez-Estévez, and N. D. Hatziargyriou, "Trends in Microgrid Control," *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 1905–1919, 2014.

[135] S. Marzal, R. Salas, R. González-Medina, G. Garcerá, and E. Figueres, "Current challenges and future trends in the field of communication architectures for microgrids," *Renewable and Sustainable Energy Reviews*, vol. 82, pp. 3610 – 3622, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1364032117314703

[136] National Institute of Standards and Technology (NIST): U.S. Department of Commerce - The Smart Grid Interoperability Panel - Cyber Security Working Group, "NISTIR 7628 - Guidelines for Smart Grid Cyber Security: Vol. 3, Supportive Analyses and References," Aug. 2010, [Online]. Available: https://www.smartgrid.gov/document/nistir_7628_guidelines_smart_grid_cyber_security_vol_3_supportive_analyses_and_references [Accessed: April 2020].

[137] C. K. Veitch, J. M. Henry, B. T. Richardson, and D. H. Hard, *SANDIA REPORT: SAND2013-5472 - Microgrid Cyber Security Reference Architecture (Version 1.0)*. Sandia National Laboratories, 2013.

[138] MITRE, "Press Release: MITRE Releases Framework for Cyber Attacks on Industrial Control Systems," Jan. 2020, [Online]. Available: https://www.mitre.org/news/press-releases/mitre-releases-framework-for-cyber-attacks-on-industrial-control-systems [Accessed: November 2020].

[139] X. Zhong, L. Yu, R. Brooks, and G. K. Venayagamoorthy, "Cyber security in smart DC microgrid operations," in *2015 IEEE First International Conference on DC Microgrids (ICDCM)*, 2015, pp. 86–91.

[140] N. Nikmehr and S. Moradi Moghadam, "Game-theoretic cybersecurity analysis for false data injection attack on networked microgrids," *IET Cyber-Physical Systems: Theory Applications*, vol. 4, no. 4, pp. 365–373, 2019.

[141] A. Teixeira, K. Paridari, H. Sandberg, and K. H. Johansson, "Voltage control for interconnected microgrids under adversarial actions," in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, 2015, pp. 1–8.

[142] M. Chlela, G. Joos, and M. Kassouf, "Impact of Cyber-Attacks on Islanded Microgrid Operation," in *Proceedings of the Workshop on Communications,*

Computation and Control for Resilient Smart Energy Systems, ser. RSES '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2939940.2939943

[143] A. J. Gallo, M. S. Turan, P. Nahata, F. Boem, T. Parisini, and G. Ferrari-Trecate, "Distributed Cyber-Attack Detection in the Secondary Control of DC Microgrids," in *2018 European Control Conference (ECC)*, 2018, pp. 344–349.

[144] M. M. Rana, L. Li, and S. W. Su, "Cyber attack protection and control of microgrids," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 2, pp. 602–609, 2018.

[145] O. A. Beg, T. T. Johnson, and A. Davoudi, "Detection of False-Data Injection Attacks in Cyber-Physical DC Microgrids," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2693–2703, 2017.

[146] E. Hossain, E. Kabalci, R. Bayindir, and R. Perez, "Microgrid testbeds around the world: State of art," *Energy Conversion and Management*, vol. 86, pp. 132 – 153, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0196890414004233

[147] S. Glover, J. Neely, A. Lentine, J. Finn, F. White, P. Foster, O. Wasynczuk, S. Pekarek, and B. Loop, "Secure Scalable Microgrid Test Bed at Sandia National Laboratories," in *2012 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2012, pp. 23–27.

[148] IBM, "IBM ILOG CPLEX Optimization Studio," 2020, [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio [Accessed: November 2020].

[149] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization*, 1st ed. Wiley-Interscience, 1999.

[150] MATLAB: MathWorks Documentation, "Simplified Model of a Small Scale Micro-Grid," 2020, [Online]. Available: https://www.mathworks.com/help/physmod/sps/examples/simplified-model-of-a-small-scale-micro-grid.html [Accessed: November 2020].

[151] P. Engebretson, *The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy*, 2nd ed. Elsevier, 2013.

[152] Ontario Hydro, "Ontario Hydro Rates," 2019, [Online]. Available: http://www.ontario-hydro.com/current-rates [Accessed: November 2020].

[153] C. Neal, R. Al Mallah, J. Fernandez, and A. Lodi, "Analyzing the Resiliency of Microgrid Control Algorithms Against Malicious Input," in *2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2020, pp. 1–6.

[154] D. Dua and C. Graff, "UCI Machine Learning Repository: Individual household electric power consumption Data Set," 2017, [Online]. Available: https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption [Accessed: November 2020].

[155] Canada Weather Stats, "Data Download for Ottawa (Kanata - Orléans)," 2020, [Online]. Available: https://ottawa.weatherstats.ca/download.html [Accessed: November 2020].

[156] C. Neal, H. Dagdougui, A. Lodi, and J. M. Fernandez, "Reinforcement Learning Based Penetration Testing of a Microgrid Control Algorithm," in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, 2021, pp. 0038–0044.

[157] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. v. d. Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 6, pp. 354–359, 2017.

[158] Deepmind, "AlphaStar: Mastering the Real-Time Strategy Game StarCraft II," Jan. 2019, [Online]. Available: https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii [Accessed: November 2020].

[159] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. A Bradford Book, 2018.

[160] M. C. Ghanem and T. M. Chen, "Reinforcement Learning for Intelligent Penetration Testing," in *2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, 2018, pp. 185–192.

[161] C. Saurraute, O. Buffet, and J. Hoffmann, "POMDPs Make Better Hakers: Accounting for Uncertainty in Penetration Testing," in *26th AAAI Conference on Artificial Intelligence (AAAI12)*, 2012, pp. 1816–1824.

[162] A. Applebaum, D. Miller, B. Strom, C. Korban, and R. Wolf, "Intelligent, Automated Red Team Emulation," in *Proceedings of the 32nd Annual Conference*

on Computer Security Applications, ser. ACSAC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 363–373. [Online]. Available: https://doi.org/10.1145/2991079.2991111

[163] A. Steinbach, "Actor-critic using deep-RL: continuous mountain car in TensorFlow," Dec. 2018, [Online]. Available: https://medium.com/@asteinbach/actor-critic-using-deep-rl-continuous-mountain-car-in-tensorflow-4c1fb2110f7c [Accessed: November 2020].

[164] R. S. Sutton, D. McAllester, S. Sing, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," in *Advances in neural information processing systems (NIPS)*, vol. 12, 1999, pp. 1057–1063.

[165] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[166] G. De Clereq (Reuters), "Run your dishwasher when the sun shines: dynamic power pricing grows," Aug. 2018, [Online]. Available: https://www.reuters.com/article/us-europe-electricity-prices-insight/run-your-dishwasher-when-the-sun-shines-dynamic-power-pricing-grows-idUSKBN1KN0L7 [Accessed: November 2020].

[167] P. Belobaba, A. Odoni, and C. Barnhart, *The Global Airline Industry*, 2nd ed. Wiley, 2016.

[168] Federal Aviation Administration, "Automatic Dependent Surveillance-Broadcast (ADS-B)," Aug. 2020, [Online]. Available: https://www.faa.gov/nextgen/programs/adsb/ [Accessed: November 2020].

[169] Nav Canada, "Implementation of ADS-B By Nav Canada," Oct. 2017, [Online]. Available: https://www.icao.int/SAM/Documents/2017-ADSB/09%20NAVCANADA-ADS_B_CAN_20170929%20(Mail%20version).pdf [Accessed: November 2020].

[170] International Civil Aviation Organization (ICAO), "Presentation of 2018 Air Transport Statistical Results," 2018, [Online]. Available: https://www.icao.int/annual-report-2018/Documents/Annual.Report.2018_Air%20Transport%20Statistics.pdf" [Accessed: November 2020].

[171] The Economist, "Losing control: Air-traffic control is a mess," Jun. 2019, [Online]. Available: https://www.economist.com/international/2019/06/15/air-traffic-control-is-a-mess [Accesed: November 2020].

[172] Wendover Productions (YouTube), "How Air Traffic Control Works," Jun. 2019, [Online]. Available: https://www.youtube.com/watch?v=C1f2GwWLB3k&t=1s [Accessed: November 2020].

[173] Herr-K (WikiCommons), "Deister-radar," Jan. 2006, [Online]. Available: https://commons.wikimedia.org/wiki/File:Deister-radar.jpg [Accessed: November 2020].

[174] E. Mayer and J. Fröhlich, "A DDS-Based Middleware for Cooperation of Air Traffic Service Units," in *Communication Technologies for Vehicles*, A. Sikora, M. Berbineau, A. Vinel, M. Jonsson, A. Pirovano, and M. Aguado, Eds. Cham: Springer International Publishing, 2014, pp. 94–102.

[175] B. Chen, H. H. Cheng, and S. Member, "A Review of the Applications of Agent Technology in Traffic and Transportation Systems," *A Review of the Applications of Agent Technology in Traffic and Transportation Systems*, vol. 11, no. 2, pp. 485–497, 2010.

[176] DDS Foundation, "Who's Using DDS?" 2020, [Online]. Available https://www.dds-foundation.org/who-is-using-dds-2/ [Accessed: November 2020].

[177] M. J. Michaud, T. Dean, and S. P. Leblanc, "Attacking omg data distribution service (dds) based real-time mission critical distributed systems," in *2018 13th International Conference on Malicious and Unwanted Software (MALWARE)*, 2018, pp. 68–77.

[178] Z. Feng and Y. Yang, "Throughput Analysis of Secondary Networks in Dynamic Spectrum Access Networks," in *2010 INFOCOM IEEE Conference on Computer Communications Workshops*, March 2010, pp. 1–6.

[179] C. Finke, J. Butts, R. Mills, and M. Grimaila, "Enhancing the security of aircraft surveillance in the next generation air traffic control system," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 1, pp. 3 – 11, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1874548213000048

[180] Z. Wu, A. Guo, M. Yue, and L. Liu, "An ADS-B Message Authentication Method Based on Certificateless Short Signature," *IEEE Transactions on Aerospace and Electronic Systems*, vol. PP, pp. 1–1, 08 2019.

[181] M. Viggiano, E. Valovage, K. Samuelson, and D. Hall, "Secure ADS-B authentication system and method," 2001, uS-Patent-7,730,307, US-Patent-Appl-SN-11/401,017.

[182] Y. Kim, J.-Y. Jo, and S. Lee, "ADS-B vulnerabilities and a security solution with a timestamp," *IEEE Aerospace and Electronic Systems Magazine*, vol. 32, no. 11, pp. 52–61, November 2017.

[183] M. Schäfer, V. Lenders, and J. Schmitt, "Secure Track Verification," in *IEEE Symposium on Security and Privacy*, May 2015, pp. 199–213.

[184] N. Ghose and L. Lazos, "Verifying ADS-B navigation information through Doppler shift measurements," in *IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, Sep. 2015, pp. 4A2–1–4A2–11.

[185] M. Schäfer, P. Leu, V. Lenders, and J. Schmitt, "Secure Motion Verification Using the Doppler Effect," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, ser. WiSec '16, 2016, pp. 135–145. [Online]. Available: http://doi.acm.org/10.1145/2939918.2939920

[186] M. Strohmeier, V. Lenders, and I. Martinovic, "Intrusion Detection for Airborne Communication Using PHY-Layer Information," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, M. Almgren, V. Gulisano, and F. Maggi, Eds. Cham: Springer International Publishing, 2015, pp. 67–77.

[187] X. Ying, J. Mazer, G. Bernieri, M. Conti, L. Bushnell, and R. Poovendran, "Detecting ADS-B Spoofing Attacks Using Deep Neural Networks," in *2019 IEEE Conference on Communications and Network Security (CNS)*, 2019, pp. 187–195.

[188] E. Habler and A. Shabtai, "Using LSTM encoder-decoder algorithm for detecting anomalous ADS-B messages," *Computers and Security*, vol. 78, pp. 155–173, 2018. [Online]. Available: https://doi.org/10.1016/j.cose.2018.07.004

[189] A. Babar, "An Approach to Represent and Transform Application Specific Constraints for an Intrusion Detection System," Master's thesis, Queen's University, 2020. [Online]. Available: https://qspace.library.queensu.ca/handle/1974/27685

[190] L.-P. Morel, "Using Ontologies to Detect Anomalies in the Sky," Master's thesis, Polytechnique Montréal, 2017. [Online]. Available: https://publications.polymtl.ca/2818/

[191] A. Souag, C. Salinesi, and I. Comyn-Wattiau, "Ontologies for Security Requirements: A Literature Survey and Classification," in *Advanced Information Systems Engineering Workshops*, M. Bajec and J. Eder, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 61–69.

[192] K. Arbanas and M. Cubrilo, "Ontology in Information Security," *Journal of Information and Organizational Sciences*, vol. 35, no. 2, pp. 107 – 136, 2015.

[193] R. M. Keller, "Ontologies for Aviation Data Management," *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pp. 1–9.

[194] W. Li and S. Tian, "An ontology-based intrusion alerts correlation system," *Expert Systems with Applications*, vol. 37, no. 10, pp. 7138 – 7146, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095741741000271X

[195] A. Sadighian, J. M. Fernandez, A. Lemay, and S. T. Zargar, "ONTIDS: A Highly Flexible Context-Aware and Ontology-Based Alert Correlation Framework," in *Foundations and Practice of Security*, J. L. Danger, M. Debbabi, J.-Y. Marion, J. Garcia-Alfaro, and N. Zincir Heywood, Eds. Cham: Springer International Publishing, 2014, pp. 161–177.

[196] L. Coppolino, S. D'Antonio, I. A. Elia, and L. Romano, "From Intrusion Detection to Intrusion Detection and Diagnosis: An Ontology-Based Approach," in *Software Technologies for Embedded and Ubiquitous Systems*, S. Lee and P. Narasimhan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 192–202.

[197] W3C, "OWL 2 Web Ontology Language Document Overview (Second Edition)," Dec. 2012, [Online]. Available: https://www.w3.org/TR/owl2-overview/ [Accessed: November 2020].

[198] ——, "RDF 1.1 Concepts and Abstract Syntax," Feb. 2014, [Online]. Available: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/ [Accessed: November 2020].

[199] J. Undercoffer, A. Joshi, and J. Pinkston, "Modeling Computer Attacks: An Ontology for Intrusion Detection," in *Recent Advances in Intrusion Detection*, G. Vigna, C. Kruegel, and E. Jonsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 113–135.

[200] EuroScope, "Euroscope," 2020, [Online]. Available: https://www.euroscope.hu/wp/ [Accessed: November 2020].

[201] VATSIM, "About VATSIM," 2020, [Online]. Available: https://www.vatsim.net/about [Accessed: November 2020].

[202] Ontotext, "GraphDB," 2020, [Online]. Available: http://graphdb.ontotext.com/ [Accessed: November 2020].

[203] J.-Y. De Miceli, "Détection d'attaques informatiques sophistiquées contre les communications ADS-B en aviation," Master's thesis, Polytechnique Montréal, 2020.

[204] A. Lemay, J. Fernandez, and S. Knight, "Modelling physical impact of cyber attacks," in *2014 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, 2014, pp. 1–6.

[205] A. Lemay, J. M. Fernandez, and S. Knight, "A modbus command and control channel," in *2016 Annual IEEE Systems Conference (SysCon)*, 2016, pp. 1–6.

[206] A. Lemay and S. Knight, "A timing-based covert channel for scada networks," in *2017 International Conference on Cyber Conflict (CyCon U.S.)*, 2017, pp. 8–15.

[207] R. Studer, V. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," *Data & Knowledge Engineering*, vol. 25, no. 1, pp. 161 – 197, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169023X97000566

[208] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199 – 220, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1042814383710083

[209] W. N. Borst, "Construction of Engineering Ontologies," Ph.D. dissertation, University of Twente, Enschede, 1997.

[210] T. Heath and C. Bizer, *Linked data: Evolving the web into a global data space*, 1st ed., ser. Synthesis lectures on the semantic web: theory and technology. Morgan & Claypool, 2011.

APPENDIX A BILEVEL KNAPSACK WITH INTERDICTION CONSTRAINTS SECURITY GAME

In the traditional knapsack problem a player must select a subset of items, where each item has an associated profit and weight, to place into their knapsack. The player's solution is the set of items which maximizes profit, while respecting the weight capacity of the knapsack. The basic model is:

$$\max_{y \in B^n} \quad \sum_{i=1}^{n} p_i y_i \tag{A.1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i y_i \leq c. \tag{A.1b}$$

Where, $B^n = \{0,1\}^n$, $y \in B^n$ is the binary decision vector ($y_i = 1$ if item i is packed, $y_i = 0$ otherwise), p_i is the profit of item i, w_i is the weight of item i, and c is the weight capacity of the knapsack.

In the bilevel interdiction knapsack problem there is a common set of items which two adversarial players can select to be placed into their own respective knapsack. The leader minimizes the profit of the follower by choosing a subset of items to remove from the common set by placing them into their own knapsack. These selected items are interdicted and cannot be used by the follower who only chooses from the remaining items to maximize profit.

This can be rephrased as a security problem, where the leader is a defending player trying to protect some assets and the follower is an attacking player trying to inflect damage on the assets to incur additional costs to the defender. The leader selects which assets to defend, while respecting a budget. The follower then can select, from the remaining assets, which ones to attack, while respecting their own budget. The value of the objective function can be considered as the cost of the damages incurred by the attacker's chosen attack vector. The formulation of the bilevel interdiction knapsack problem is presented in [98] and is stated as a security problem as follows:

$$\min_{(x,y)\in B^n\times B^n} \quad \sum_{i=1}^{n} p_i y_i \tag{A.2a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} v_i x_i \leq c_u \tag{A.2b}$$

where $y_1, ..., y_n$ solves the follower's problem

$$\max_{y\in B^n} \quad \sum_{i=1}^{n} p_i y_i \tag{A.2c}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i y_i \leq c_l \tag{A.2d}$$

$$y_i \leq 1 - x_i, \text{for } i = 1, ..., n. \tag{A.2e}$$

Where, $B^n = \{0,1\}^n$, $x \in B^n$ is the leader's binary decision vector, $y \in B^n$ is the follower's binary decision vector, $N = \{1, 2, ..., n\}$ is the set of assets, p_i is the cost associated with the damage inflicted by attacking asset $i \in N$, v_i is the leader's cost for defending asset $i \in N$, w_i is the follower's cost for attacking asset $i \in N$, c_u is the leader's budget, and c_l is the follower's budget. Note that it is the constraints of type (A.2e) which interdicts the items from the follower once they are selected by the leader.

Example: Suppose we are given a leader budget of $c_u = 10$, a follower budget of $c_l = 5$, and a set of assets defined as:

$$\begin{aligned} N = \{&1 = \{p_1 = 2, v_1 = 4, w_1 = 2\}, \\ &2 = \{p_2 = 3, v_2 = 6, w_2 = 2\}, \\ &3 = \{p_3 = 4, v_3 = 8, w_3 = 3\}\}. \end{aligned}$$

The optimization problem becomes:

$$\begin{aligned}
\min_{(x,y)\in B^3\times B^3} \quad & 2y_1 + 3y_2 + 4y_3 \\
\text{s.t.} \quad & 4x_1 + 6x_2 + 8x_3 \leq 10 \\
& \max_{y\in B^3} \quad 2y_1 + 3y_2 + 4y_3 \\
& \quad \text{s.t.} \quad 2y_1 + 2y_2 + 2y_3 \leq 5 \\
& \qquad\quad y_1 \leq 1 - x_1 \\
& \qquad\quad y_2 \leq 1 - x_2 \\
& \qquad\quad y_3 \leq 1 - x_3.
\end{aligned}$$

For this small example the optimal decision vectors and objective value can be determined through enumeration. The optimal decision vectors are $x^* = (1, 1, 0)$ and $y^* = (0, 0, 1)$, with an optimal objective value of 4.

APPENDIX B MICROGRID CONTROL EXAMPLE AND CPLEX CODE

This appendix demonstrates a simplified control example of the optimization-based MG control problem, outlined in Section 4.2.2, and provides the CPLEX code used to solve the problem.

Consider the example scenario where the controller, at time k, must determine the battery control to invoke into the MG, given a forecast horizon of $t = 0, 1, 2, 3$ and state set $\mathbb{X}_k = \{P_k^L, P_k^P, \Lambda_k^B, \Lambda_k^S, b_k\}$, with the following values:

- **Load Power Forecast:** $P_k^L = \{10, 20, 30, 40\}$
- **Photovoltaic Power Forecast:** $P_k^P = \{20, 25, 22, 25\}$
- **Electricity Buying Price:** $\Lambda_k^B = \{2, 2, 2, 8\}$
- **Electricity Selling Price:** $\Lambda_k^S = \{-1, -1, -1, -1\}$
- **Initial Battery Charge:** $b_k = 80$

Table B.1 shows the results of constructing the updated state set $\dot{\mathbb{X}}$ using Algorithm 1 (with $\omega = 1$ for simplicity), and the resulting control set $\dot{\mathbb{Y}}_k$ from solving System (4.2) (with $\Delta^\tau = 1$ for simplicity). The battery's SOC, $b_{k,t}$, is constrained to the lower bound of $b^{\min} = 75$ and an upper bound of $b^{\max} = 100$.

Note that, the value for $c_{k,t}$ reflects how the cost of the controller is expected to being affected at time $k+t$ and the value for $d_{k,t}$ reflects how the battery's SOC is expected to being affected at time $k + t$. If $y_{k,t} = 1$ then the battery is turned `ON` and the controller takes the value for $d_{k,t}$. If $y_{k,t} = 0$ then the battery is turned `OFF` and the controller takes the value for $c_{k,t}$. The optimal battery control command sequence is determined to be $y_{k,t}^* = \{1, 0, 0, 1\}$.

The controller determines that when $t = 0$ the excess solar power should be used to increase the battery's SOC by 10, when $t = 1$ the excess solar power should be sold to the main grid for a cost of -5 (negative cost implies a profit), when $t = 2$ the shortfall of solar power production should be purchased from the main grid at a cost of 16, and when $t = 3$ the shortfall of solar power production should be met from the battery resulting in a SOC decrease of 15.

The value $y_{k,0} = 1$ is determined as the battery control at this timestep k. This means that the battery is turned `ON` over the course of the hour k, until this process is repeated at hour $k + 1$, and so.

Table B.1 Results of the control scenario. Total cost is $0 - 5 + 16 + 0 = 11$.

Example Control Scenario at Time k						
			$t = 0$	$t = 1$	$t = 2$	$t = 3$
$\mathbb{X}_k$	Load Forecast	$p^L_{k,t}$	10	20	30	40
	PV Forecast	$p^P_{k,t}$	20	25	22	25
	Buy Price	$\lambda^B_{k,t}$	2	2	2	8
	Sell Price	$\lambda^S_{k,t}$	-1	-1	-1	-1
$\dot{\mathbb{X}}_k$	Costs Forecast	$c_{k,t}$	-10	-5	16	120
	Differences Forecast	$d_{k,t}$	10	5	-8	-15
$\dot{\mathbb{Y}}_k$	Pred. Prev. Batt. SOC	$b_{k,t-1}$	80	90	90	90
	Pred. Batt. Control	$y_{k,t}$	1	0	0	1
	Pred. Grid Control	$\bar{y}_{k,t}$	0	1	1	0
	Pred. Batt. SOC	$b_{k,t}$	90	90	90	75
Pred. Cost		$\text{cost}_{k,t}$	0	-5	16	0

Listing B.1 shows the CPLEX code used to solve the controller's optimization problem from System (4.2). Listing B.2 shows the CPLEX data file used in the example scenario from this appendix. Listing B.3 shows a log file with the output from solving this example scenario.

Listing B.1 Microgrid Controller CPLEX Code: controller.mod

```
// Given Variables
int cur_hour = ...;
int nbHours = ...;
float BattChargeInit = ...;
range hours = 0..nbHours;
range lHours = 0..nbHours+1;
float Costs[hours] = ...;
float Diffs[hours] = ...;
float delta_time = ...;
float max_batt = ...;
float min_batt = ...;

// Decision Variables
dvar boolean y[hours];      // 1 => Battery ON, 0 => Battery OFF
dvar boolean y_bar[hours];   // 0 => Grid OFF , 1 => Grid ON
```

```
dvar float+ BattCharge[hours];
dvar float+ BattChargePrev[lHours];
dvar float Cost[hours];

minimize
  sum( h in hours )
       Cost[h];
subject to{
  BattChargePrev[0] == BattChargeInit;

  forall( h in hours){
    ctCalcCost:
      Cost[h] == y_bar[h] * Costs[h];

    ctBattOnOff:
      y[h] + y_bar[h] == 1;

    ctBattChargeLevel:
      BattCharge[h] == BattChargePrev[h] + y[h] * (Diffs[h]) * delta_time;

    ctBattChargeForNextIteration:
      BattChargePrev[h+1] == BattCharge[h];

    ctBattChargeMinMax:
      min_batt <= BattCharge[h] <= max_batt;
  }
}execute{
  // Create log file
  var logfile = new IloOplOutputFile("controller_log.txt");
  logfile.writeln("Pred. y      = ",thisOplModel.y);
  logfile.writeln("Pred. y_bar  = ",thisOplModel.y_bar);
  logfile.writeln("Pred. BattChargePrev = ",thisOplModel.BattChargePrev);
  logfile.writeln("Pred. BattCharge     = ",thisOplModel.BattCharge);
  logfile.writeln("Pred. Cost_t = ",thisOplModel.Cost);
  logfile.writeln("Objective (Pred. Total Cost) = ", cplex.getObjValue());
  logfile.close();
}
```

Listing B.2 Example CPLEX Data Input File: controller.dat

```
cur_hour = 0;
nbHours =  3; // t=0,1,2,3
BattChargeInit = 80;
max_batt = 100;
```

```
min_batt = 75;
Costs =  [-10.0, -5.0, 16.0, 120.0];
Diffs =  [10.0, 5.0, -8.0, -15.0];
delta_time = 1;
```

Listing B.3 Example CPLEX Log Output File: controller_log.txt

```
Pred. y     =  [1 0 0 1]
Pred. y_bar =  [0 1 1 0]
Pred. BattChargePrev =  [80 90 90 90 75]
Pred. BattCharge     =  [90 90 90 75]
Pred. Cost_t =  [0 -5 16 0]
Objective (Pred. Total Cost) = 11
```

APPENDIX C NOTES ON DEVELOPING THE MICROGRID CONTROL ALGORITHM

This appendix demonstrates the dynamics of the MG simulation, outlined in Section 4.2, through the use of some basic control logic, and validates the need for the optimization-based control algorithm presented in System (4.2).

Scenario Overview

The examples in this appendix utilize an identical load that is repeated for 7 days. The load is characterized by a peak in the morning hours and a larger peak in the evening. The solar irradiance is also repeated each day with peak solar power generation in the early afternoon. The battery SOC is initialized at 80%. Should the MG not produce enough electricity, the shortfall can be purchased from the main grid at the following prices:

- **Off-peak:** 6.5 cents/kWh (19:00-07:00)
- **Mid-peak:** 9.4 cents/kWh (07:00-12:00 and 17:00-19:00)
- **On-peak:** 13.4 cents/kWh (12:00-17:00)

Excess electricity generated by the MG can be sold a the following price:

- **Constant:** -5 cents/kWh

Basic Control Algorithms

Battery Always `OFF`: This is the most basic scenario where there is no control by the battery. Since the battery is set to `OFF`, all the load that cannot be met by the solar panel is purchased from the main grid. Also, any generated solar power is sold to the grid. The results from this simulation are shown in Figure C.1 and has a total cost for the week of $22.54 (a description of how the axes in the figure are interpreted is provided in Section 4.3.2).

Battery Always `ON`: With the battery turned `ON` at all times, the demand is met by the battery when there is power available. Also, the battery is charged when there is excess solar power being generated. When there is no power remaining the battery, demand is met from the grid. This is shown in Figure C.2 and has a total cost for the week of $8.24

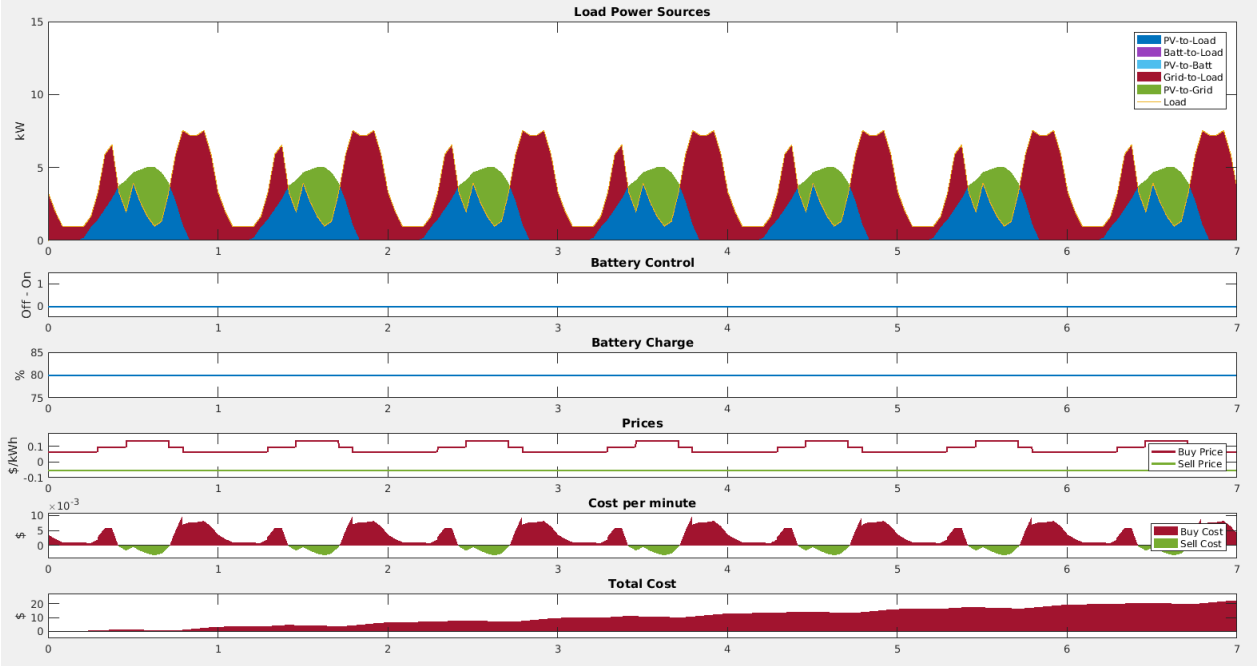

Figure C.1 Control Scenario: Battery `OFF` (Score: $22.54)

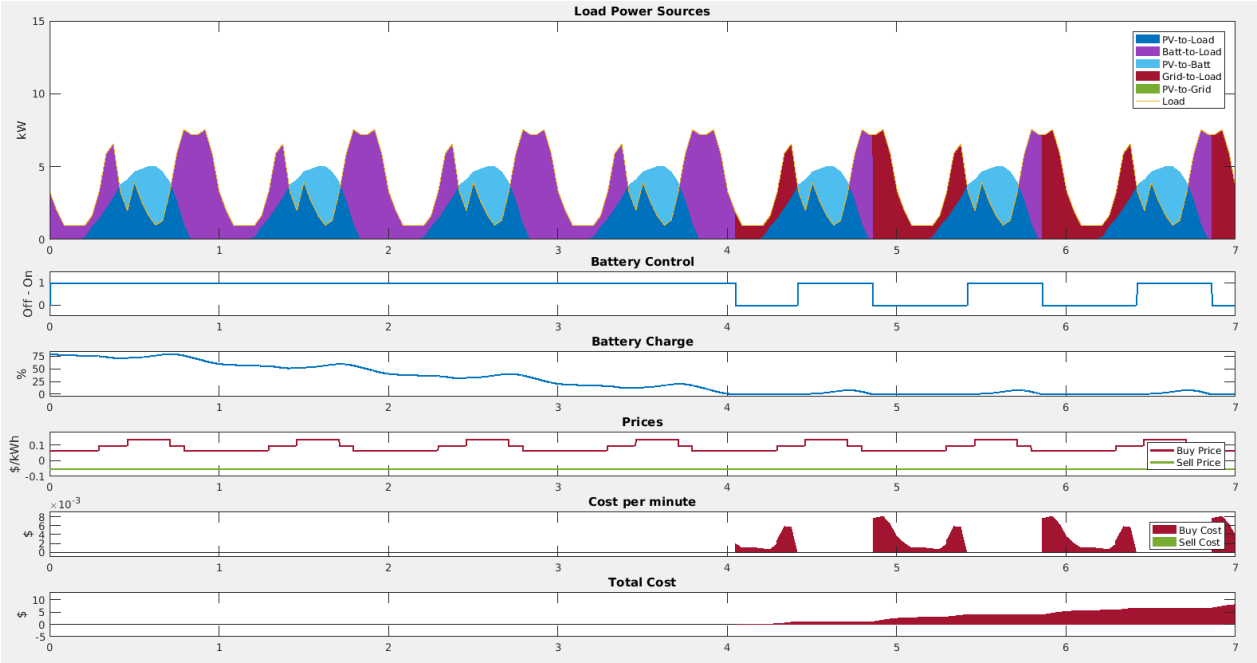

Figure C.2 Control Scenario: Battery `ON` (Score: $8.24)

Random `ON/OFF`: As a comparison, a simulation where the battery is randomly turned `ON` or `OFF` at each hour is shown in Figure C.3 and has a total cost for the week of $10.77.

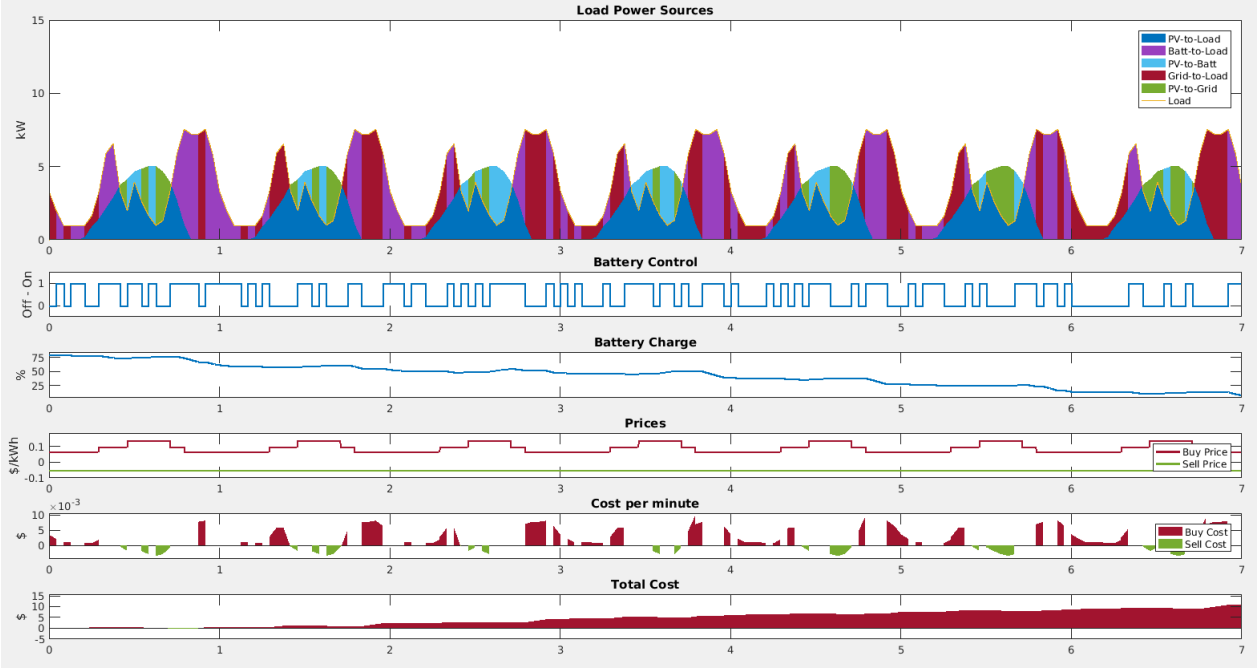

Figure C.3 Control Scenario: Random (Score: $10.77)

The results of these initial control algorithms are compared in Table C.1

Table C.1 Results of the basic control logic algorithms using default solar, load, and price values

Controller	Cost ($)
`OFF`	22.54
`ON`	8.24
Random	10.77

Using the control where the battery is always set to `ON` has shown the lowest cost so far. In this setting the battery completely exhausts its reserves by discharging all of its available power. However, once the battery charge is depleted by mid-week, a pattern emerges of small bursts of charging and discharging of the battery. This control algorithm would not be advantageous over the course of a larger timeframe, such as over the course of a year. There would be a one-time payoff of reduced cost in the first week, however the battery will be nearly empty for the entire year and could not be used as an emergency power supply.

Instead, it may be more advantageous to prevent the battery SOC from being lower than some threshold, such as 75%. The remaining control simulations consider this setting.

Basic Control Algorithms with SOC $\geq$ 75%

Battery `ON`, SOC $\geq$ 75%: In the simulation from Figure C.4, the battery is turned `ON` whenever possible as long as the battery SOC is over 75%. The same pattern of bursts of charging and discharging occur, however this occurs much sooner in the simulation. The total cost for the week is $18.98.

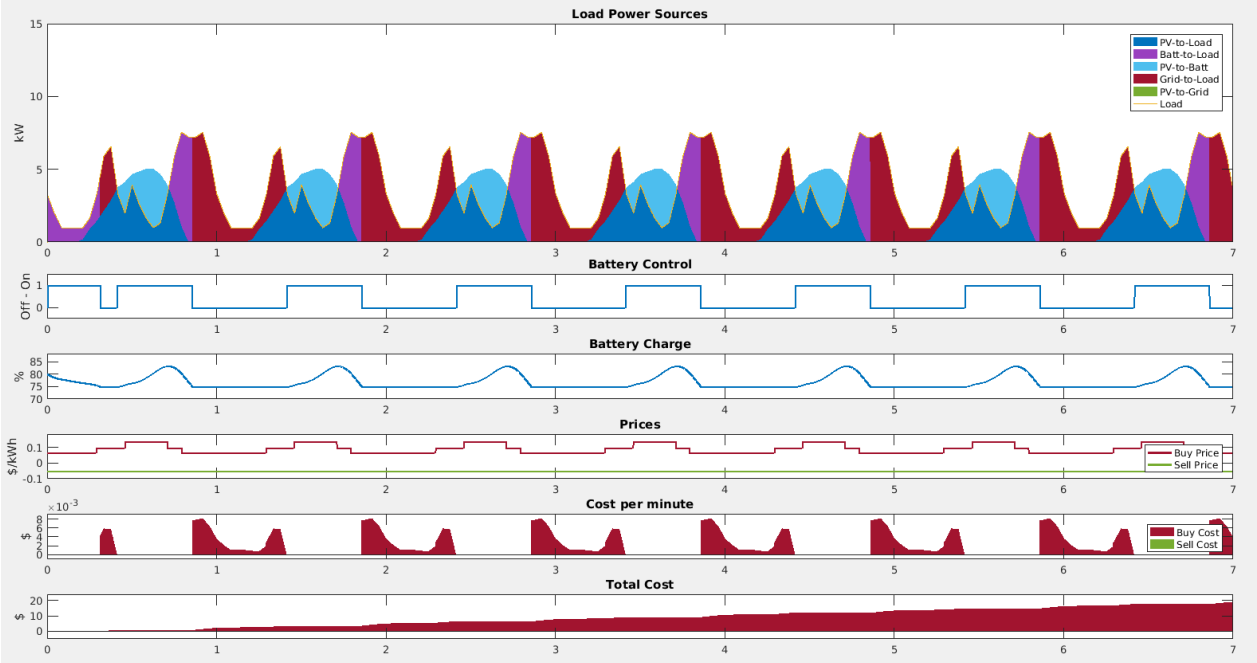

Figure C.4 Control Scenario: Battery `ON`, SOC$\geq$75% (Score: $18.98)

Battery Random `ON/OFF`, SOC $\geq$ 75%: As a comparison, operating the battery at random if the battery SOC is over 75% is shown in Figure C.5. The total cost for the week is $20.07.

Table C.2 shows the results of the initial control operations when the battery SOC is restricted to being over 75%.

Again, setting the battery to being in an `ON` state has shown to have the lowest cost so far. However, there is a better control strategy if the future values for electrical loads, solar irradiance, and prices are considered.

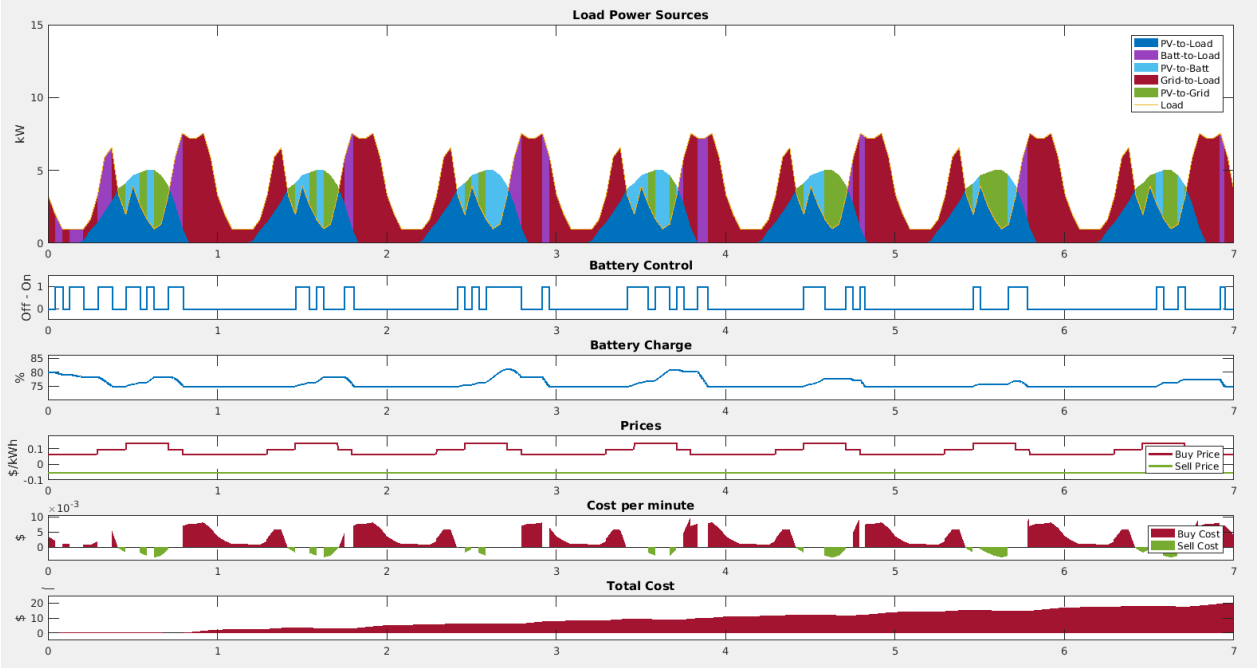

Figure C.5 Control Scenario: Random, SOC≥75% (Score: $20.07)

Table C.2 Results of the basic control logic algorithms with battery SOC≥75%, using default solar, load, and price values

Controller	Cost ($)
`OFF`	22.54
`ON`, SOC $\geq$ 75%	18.98
Random, SOC $\geq$ 75%	20.07

Optimization-Based Control with SOC $\geq$ 75%

CPLEX, SOC $\geq$ 75%: This is the controller presented in Section 4.2.2 and System (4.2). The control algorithm has been implemented using CPLEX and has been integrated to work within the simulation environment. The total cost for the week is $18.39 and is shown in Figure C.6.

The controller receives a score of 18.40 which outperforms the other seen control algorithms using the same scenario (see Table C.3). The control performs *peak shaving*, where the battery is used to meet peaks of demand when there is a higher price. In this scenario, this occurs at the periods immediately before and after peak solar irradiance.

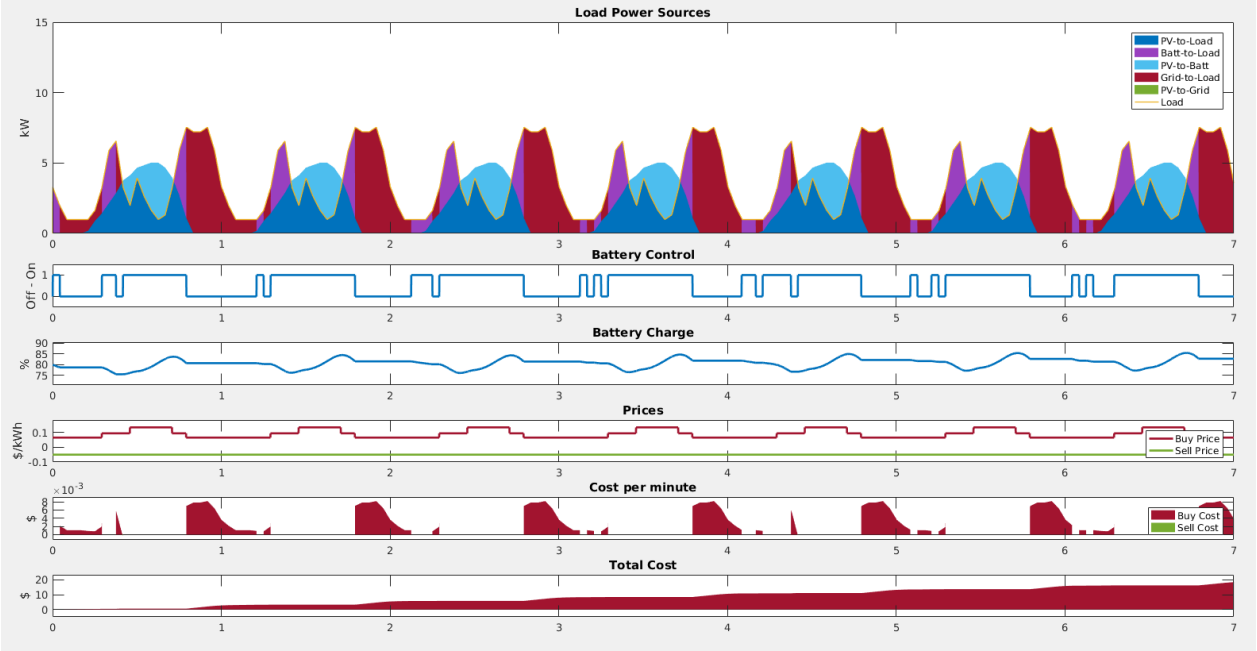

Figure C.6 Control Scenario: CPLEX, SOC≥75% (Score: $18.40)

Table C.3 Results of all demonstrated control algorithms with battery SOC≥75%, using default solar, load, and price values

Controller	Cost ($)
`OFF`	22.54
`ON`, SOC $\geq$ 75%	18.98
Random, SOC $\geq$ 75%	20.07
CPLEX, SOC $\geq$ 75%	18.40

To further illustrate the controller, the simulation is run with the same parameters except the selling price is set to an exceptionally high value of 100 cents/kWh (results are shown in Figure C.7). Under this setting, the controller is aware that it is more advantageous to sell all excess solar power to the grid at a high profit, as oppose to charge the battery. The controller then purchases nearly all of the power it is missing from the grid at a relatively low cost. The implemented control algorithm has shown to be robust at handling different control decisions and is used as the control algorithm in the MG testbed throughout Chapter 4.

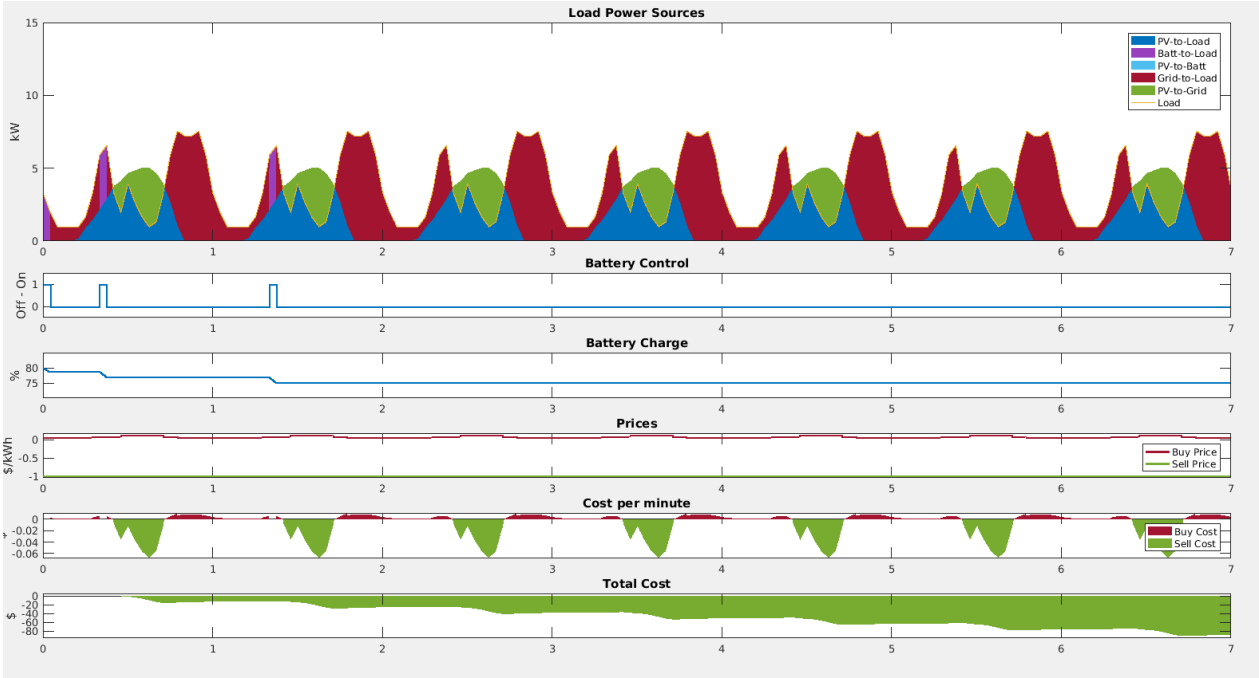

Figure C.7 Control Scenario: CPLEX, SOC≥75%, high selling price (Scenario Score: $-88.60)

APPENDIX D MICROGRID ATTACK SCENARIO: MANUAL ATTACKS ON 7 DAYS OF DEFAULT DATA

This appendix shows the effects of attacks against the MG simulation scenario outlined in Section 4.3.2. These simulations occur for the seven-day default data set with an initial battery SOC of 80%. Figure D.1 show the effects of the *initial SOC +5% attack*, Figure D.2 show the effects of the *initial SOC -5% attack*, Figure D.3 show the effects of the *switch control command every hour attack*, and Figure D.4 show the effects of the *switch control command every 5 hours attacks*.

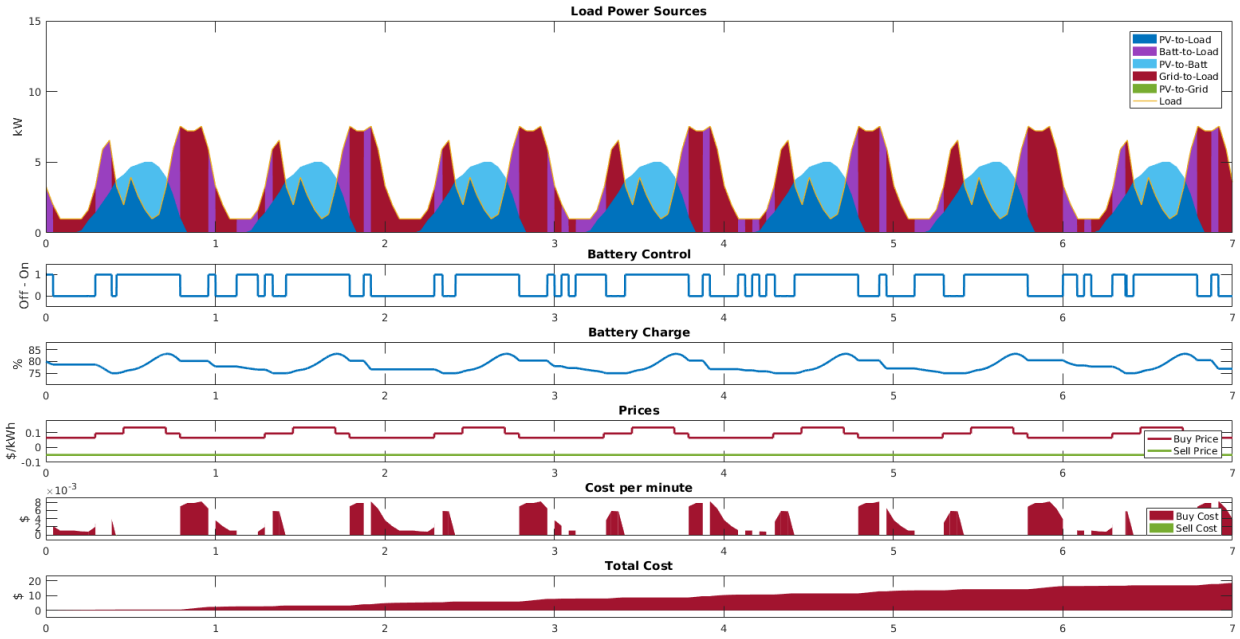

Figure D.1 Scenario: default data, init. batt. SOC 80%, initial SOC +5% attack (Cost: $18.51)

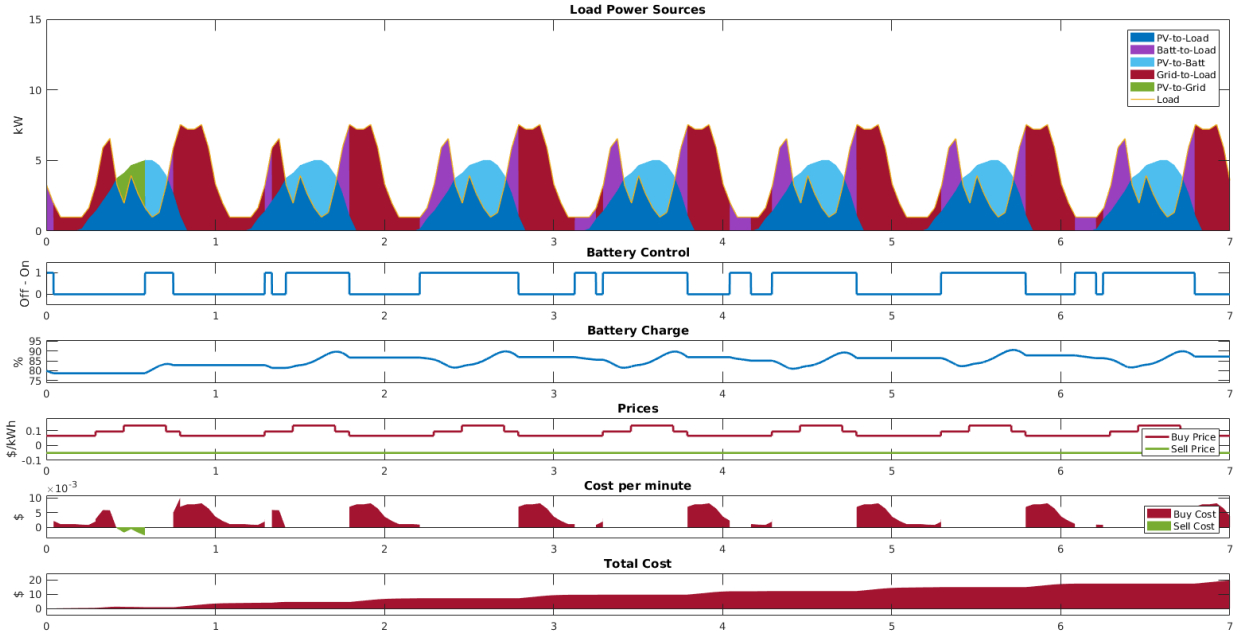

Figure D.2 Scenario: default data, init. batt. SOC 80%, initial SOC -5% attack (Cost: $19.53)

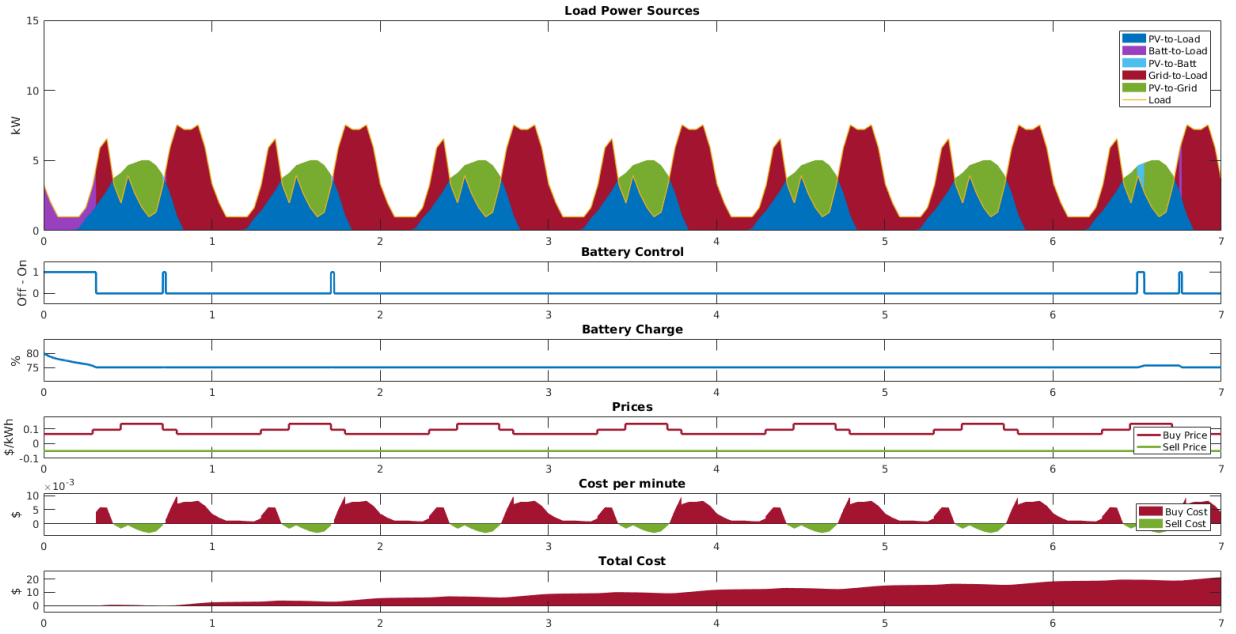

Figure D.3 Scenario: default data, init. batt. SOC 80%, switch control command every hour attack (Cost: $21.34)

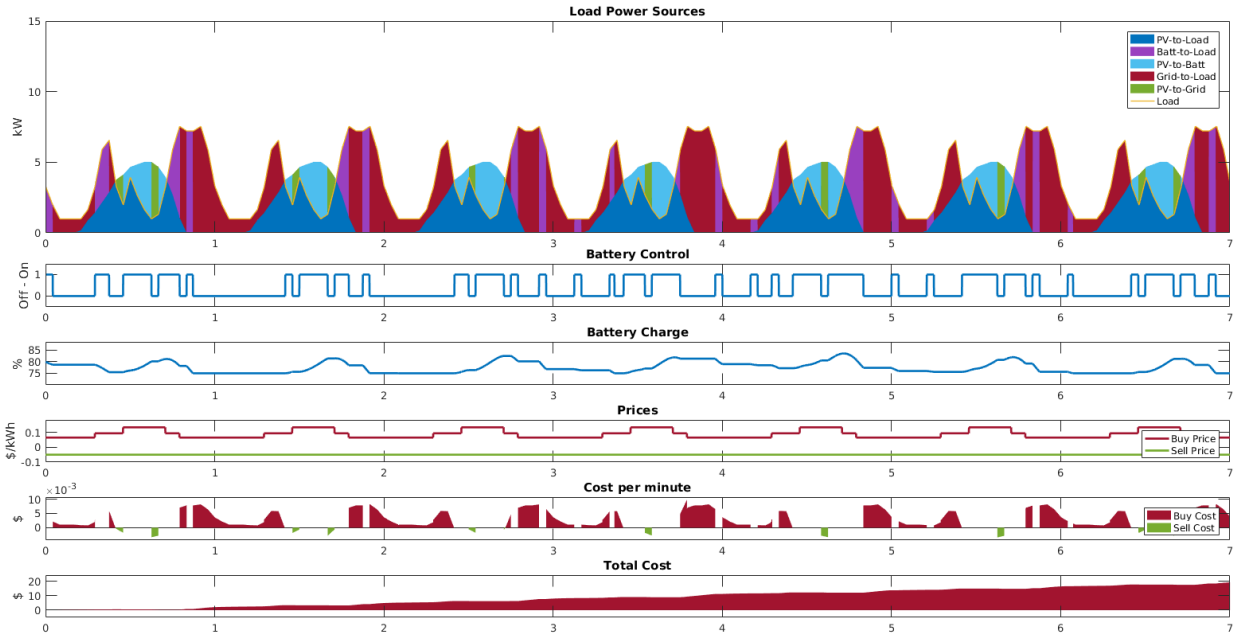

Figure D.4 Scenario: default data, init. batt. SOC 80%, switch control command every 5 hours (Cost: $19.15)

APPENDIX E MICROGRID ATTACK SCENARIO: MANUAL ATTACKS ON 7 DAYS OF REAL-WORLD DATA

This appendix shows the effects of attacks against the MG simulation scenario outlined in Section 4.3.3. These simulations occur for the seven-day real-world data set with an initial battery SOC of 90%. Figure E.1 show the effects of the *initial SOC +5% attack*, Figure E.2 show the effects of the *initial SOC -5% attack*, Figure E.3 show the effects of the *switch control command every hour attack*, and Figure E.4 show the effects of the *switch control command every 5 hours attacks*.

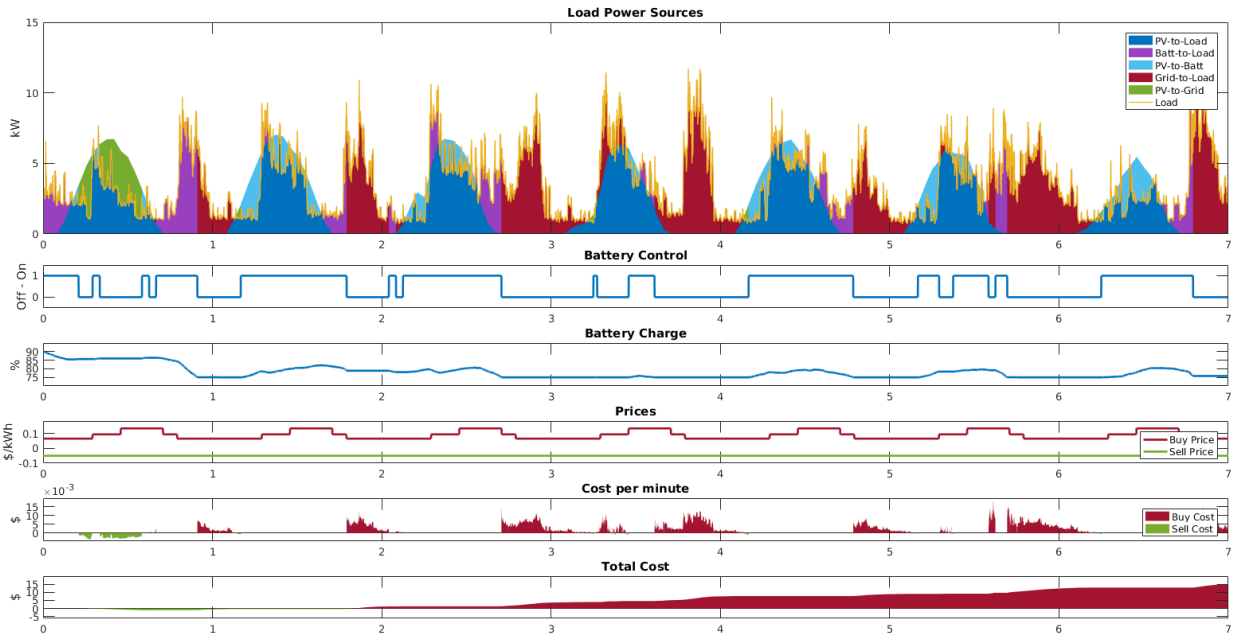

Figure E.1 Scenario: real-world data, init. batt. SOC 90%, initial SOC +5% attack (Cost: $14.96)

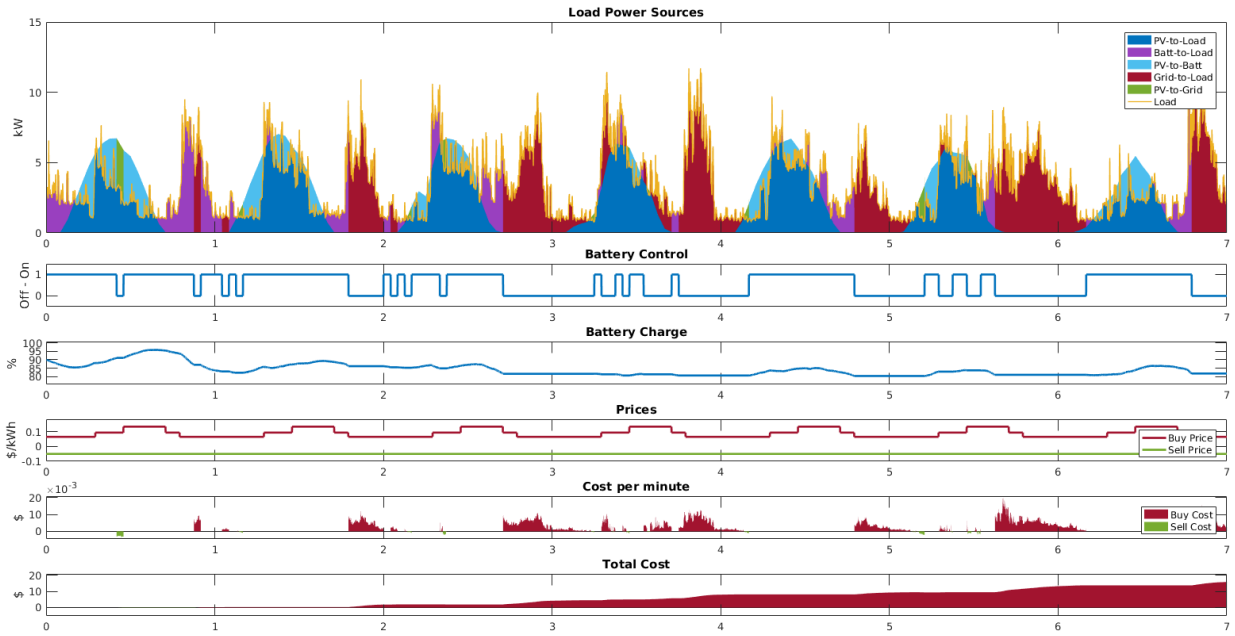

Figure E.2 Scenario: real-world data, init. batt. SOC 90%, initial SOC -5% attack (Cost: $15.83)

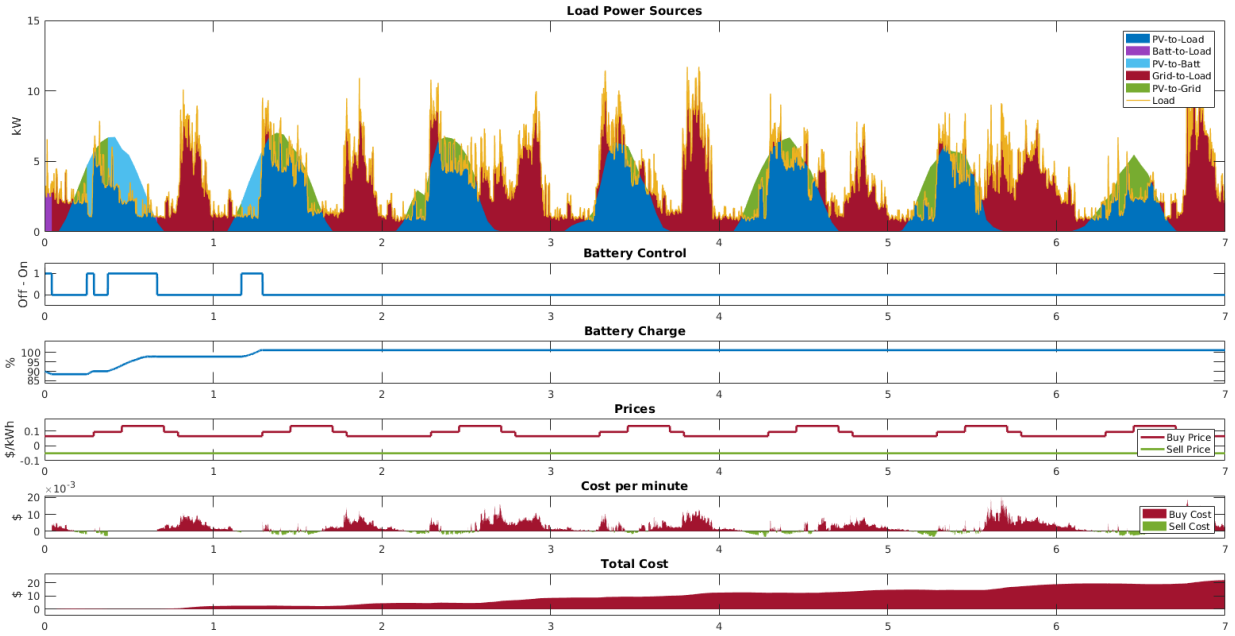

Figure E.3 Scenario: real-world data, init. batt. SOC 90%, switch control command every hour attack (Cost: $22.00)

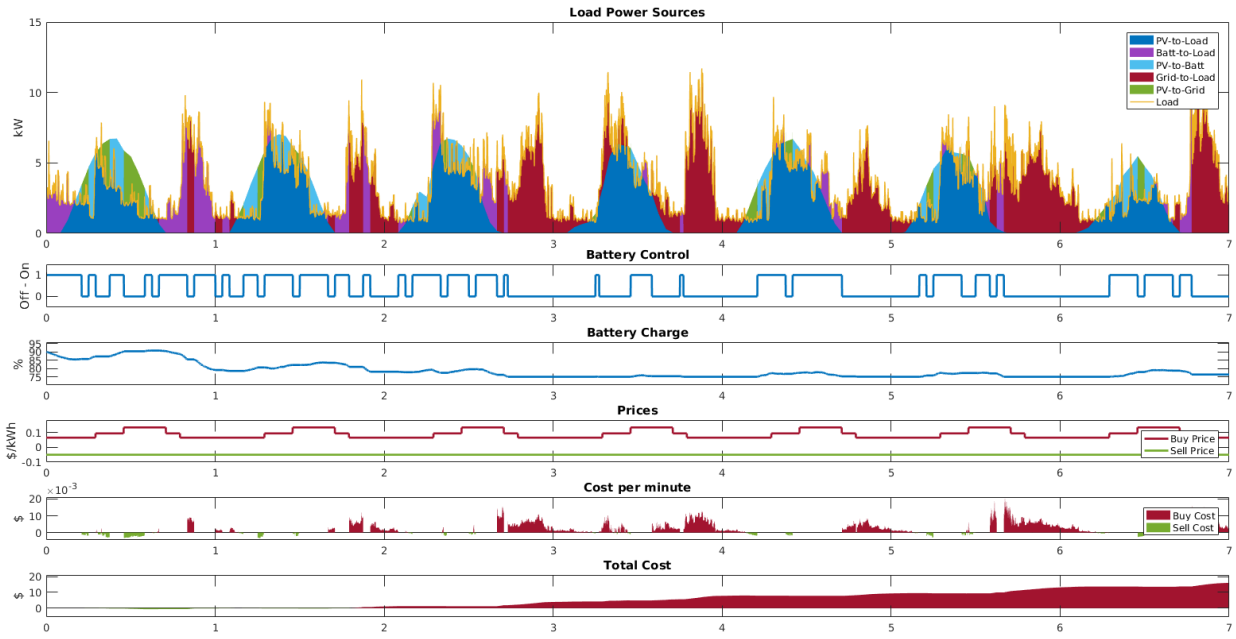

Figure E.4 Scenario: real-world data, init. batt. SOC 90%, switch control command every 5 hours (Cost: $16.09)

APPENDIX F IMPLEMENTATION DETAILS OF REINFORCEMENT LEARNING-BASED MICROGRID ATTACKING AGENT

This appendix describes the implementation of the RL-based MG attacking agents from Section 4.3.4. This work updates an implementation of the Temporal Difference Advantage Actor-Critic (TDAC) algorithm which was originally designed for the Mountain-Car control problem [163]. This appendix focuses on the implementation details and does not discuss the theory behind the TDAC algorithm. The TDAC algorithm was chosen since the control action can be a real number. Other RL algorithms could have been chosen for this reason, such as the Deep Deterministic Policy Gradient or Proximal Policy Optimization, however they were not evaluated since the results from the TDAC algorithm proved capable at satisfying the research questions of this work.

For each timestep, $k = 1...K$, in an episode, the agent is provided the current state of the MG, $s_k = \{b_k, c_{k,0}, d_{k,0}, p^L_{k,0}, p^S_{k,0}\}$ (defined in Section 4.3.4), and chooses an action a_k, based on its current policy, π. The agent then observes the effect of the action as a reward, r_k, and the next state, s_{k+1}. Based on these observations the agent updates its policy, π, with the goal of maximizing its longterm expected reward.

The TDAC algorithm defines two DNNs; the actor (policy) network, π^θ, which chooses the action to invoke and the critic (state-value) network, V^U_π, which evaluates the utility of the action. Learning involves updating the weight parameters U and θ with gradient descent.

The critic (state-value) network is outlined in Figure F.1. The input layer has a size of 5. There are two hidden layers, each with 400 nodes, activated with elu, and initialized with Xavier. The output layer is a scalar (real) number, with no activation function, and Xavier initialization. All layers are fully connected.

The actor (policy) network is outlined in Figure F.2. The input layer has a size of 5. There are two hidden layers, each with 40 nodes, activated with elu, and initialized with Xavier. The layers are fully connected. The output layer has two dimension and outputs two scalar functions; $\mu(s)$, the mean, and $\sigma(s)$, the standard deviation, of the Gaussian (normal) distribution. Actions are stochastically chosen by sampling from this distribution.

The TDAC algorithm implemented for this work is outlined in Algorithm 2. To begin, the actor and critic networks are initialized with random weights. The MG simulation acts as the environment, E, and is reinitialized for each episode $m = 1...M$. Each episode consists of timesteps $k = 1...K$. The state of the simulation is observed and the actor network is

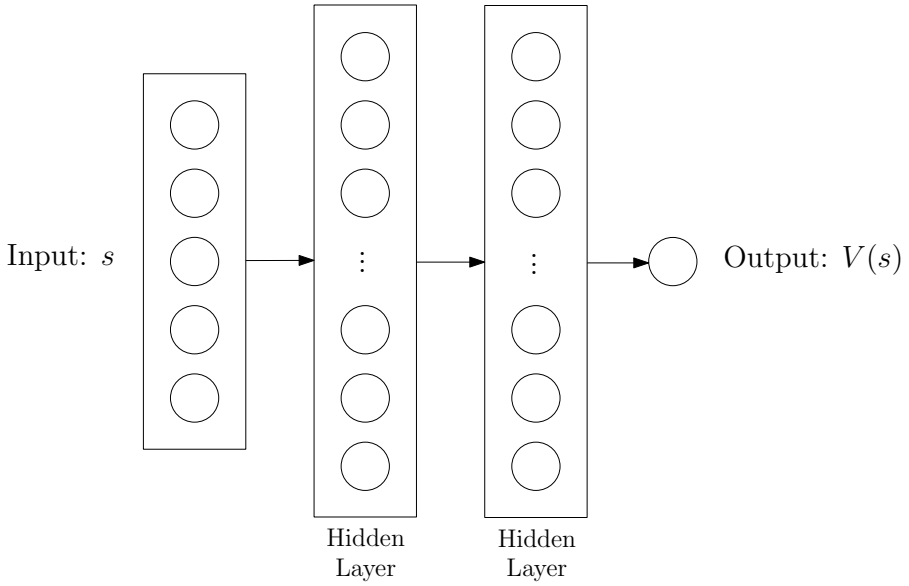

Figure F.1 Critic (State-Value) Network: $V_\pi^U(s)$

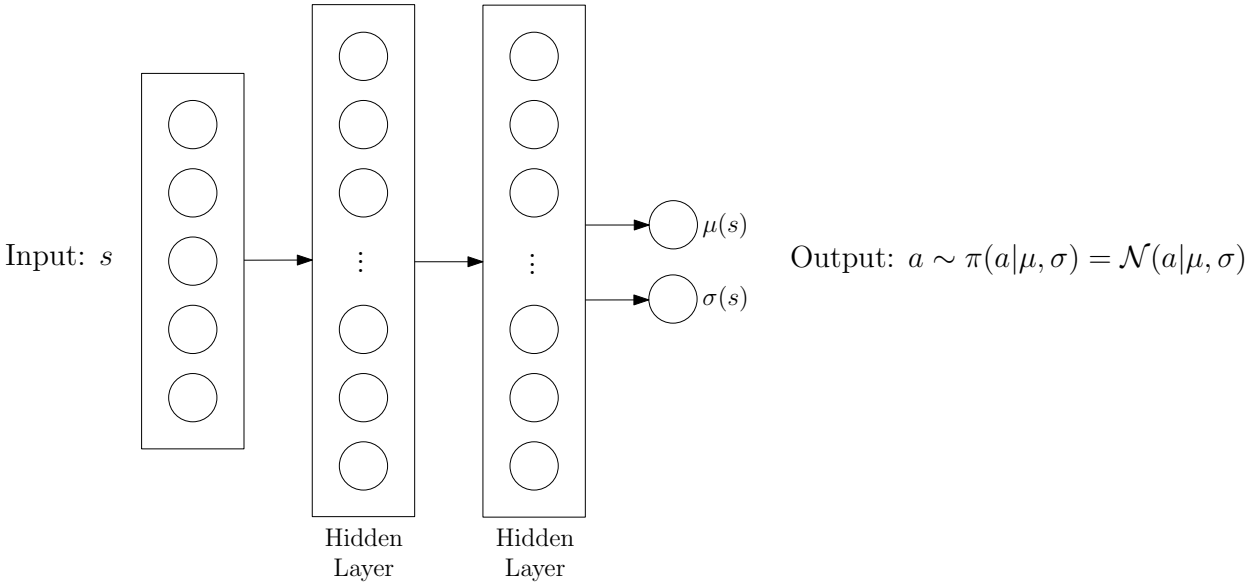

Figure F.2 Actor (Policy) Network: $\pi^\theta(s)$

evaluated to determine an action to insert into the environment. The reward and state are then observed from the environment. The TD target, w_k, is calculated to act as a training label for the critic network, with a discount factor of $\gamma = 0.99$. The TD error, δ_k, is calculated to act as training label for the actor network. The weights of the actor and critic networks are updated by using gradient descent to minimize loss.

Algorithm 2: TD Advantage Actor-Critic

Randomly initialize actor network network $\pi^{\theta}(s)$ with weights θ
Randomly initialize critic network $V_{\pi}^{U}(s)$ with weights U
for episode $m = 1$ *to* M **do**
  Initialize environment E
  for timestep $k = 1$ *to* K **do**
    Observe state s_k from E
    Sample action $a_k \sim \pi(a|\mu(s_k), \sigma(s_k)) = \mathcal{N}(a|\mu, \sigma)$ according to current policy
    Execute action a_k into E
    Observe reward r_k and next state s_{k+1} from E
    Get value of next state: $v_k := V_{\pi}^{U}(s_{k+1})$
    Set TD target: $w_k := r_k + \gamma \cdot v_k$
    Set TD error: $\delta_k := w_k - v_k$
    Update actor: π^{θ}.`minimize_loss(loss=`$-log(\mathcal{N}(a|\mu(s_k), \sigma(s_k))) \cdot \delta_k$ `, algorithm=Adam, step_size=0.0002)`
    Update critic: V_{π}^{U}.`minimize_loss(loss=`$(w_k - v_k)^2$ `, algorithm=Adam, step_size=0.001)`
  end
end

The original MG implementation architecture, from Figure 4.6, is updated in Figure F.3 to show the module which performs the learning. The script `rl_attack_server.py`, is a Python script which performs the logic from Algorithm 2. The script is a server which receives and sends IP requests on a local network on the machine. The script receives the current state, s_k, and the reward/cost from the previous action, r_{k-1}, and determines the action, a_k, to inject into the simulation.

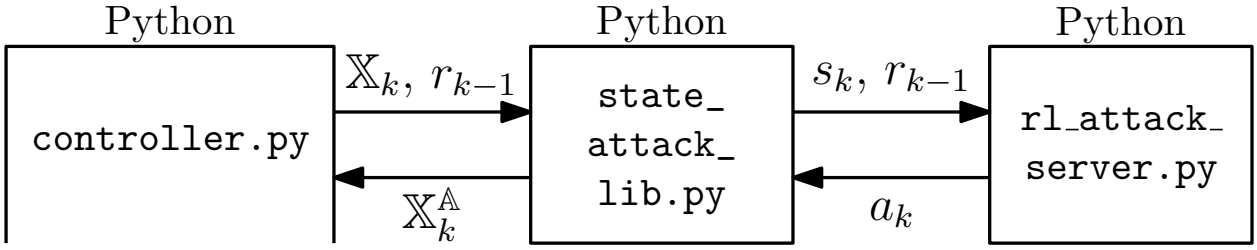

Figure F.3 Reinforcment Learning Attack Server

APPENDIX G ONTOLOGIES AND THE RESOURCE DESCRIPTION FRAMEWORK (RDF)

Ontologies

The term ontology originates from a branch of metaphysics within philosophy which deals with the nature of being, existence, reality, and how entities are related. This term has been borrowed by the field of information science to refer to the formal definition of entities and their relationships for a particular domain, in a machine-readable format. The often cited definition of an ontology for information systems, provided by Studer et al. [207] (based on definitions provided by Gruber [208] and Borst [209]), is as follows:

"*An ontology is a formal, explicit specification of a shared conceptualisation.*"

The components from the definition are further explained [207]:

- "A *conceptualisation* refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon."
- "*Explicit* means that the type of concepts used, and the constraints on their use are explicitly defined."
- "*Formal* refers to the fact that the ontology should be machine readable, which excludes natural language."
- "*Shared* reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group."

An ontology is described in a particular formal machine-readable language and is generally comprised of the following [192]:

- **Concepts (classes)**: High level abstract concepts or entities, often represented in taxonomies, which are the *types* of things being represented. For example, an ontology representing a business may have a concept of *Employee*, which could be sub-categorized as *Manager* or *Worker*.
- **Properties (attributes)**: The features or characteristics of the concept. For example, the *Employee* concept may have properties such as *Identification Number*, *Salary*, *Start Date*, *Position*, *Name*, etc.

- **Relationships**: How the entities are related. For example, a *Worker* entity may be *supervised by* a *Manager*.
- **Axioms**: Rules that govern the modeled domain. For example, an *Employee* must be a *Worker* or a *Manager*, and a *Manager* can supervise several *Workers*.
- **Instances**: The occurrences of the defined abstract concepts. For example, these are the actual employees of the business.

The Resource Description Framework (RDF)

Different languages for creating ontologies exist, however the most prevalent representation language for ontologies is based of the Resource Description Framework (RDF) and its related technologies. In RDF, each statement is represented as a *triple* consisting of three parts; a *subject*, *predicate*, and *object* [210]. The predicate describes the relationship between the subject and the object, thus an RDF triple can be conceptualized as a node and arc labeled graph (see Figure G.1).

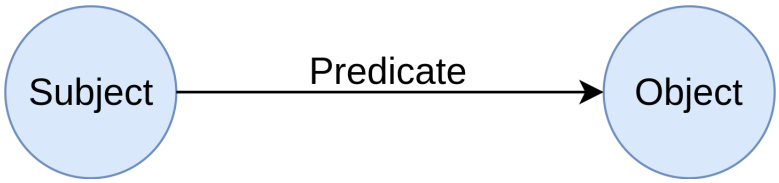

Figure G.1 RDF Model

An RDF triple can be represented as code:

```
<Subject> <Predicate> <Object>.
```

The subject is a resource (either a *concept* or *instance*) and is related to the object. The object can have a literal value (e.g. number, character string, etc.) or it can be another resource. When the object is another resource a link is formed between the resources. This is significant because it allows all the knowledge within the ontology to be represented as a directed graph and enables the possibility for automated reasoning. RDF *triples* are stored as rows in a database called a *triple store*, with the respective database columns being the subject, predicate, and object. There are two main types of entries in an ontological *triple store*; *t-box* triples and *a-box* triples. The *t-box* is comprised of the triples which describe the concepts and their relationships. The *a-box* is comprised of the triples which assert the

tangible instances existing in the data. A unique property of a *triple store* is that the schema is not static (unlike relational databases) and the schema can be updated by simply adding new *t-box triples* into the *triple store*. The primary language for querying RDF *triple stores* is called SPARQL (a recursive acronym for SPARQL Protocol and RDF Query Language). The process of developing an ontology for a particular domain is known as *ontology engineering* and should be performed in collaboration with domain experts.