

AMERICAN UNIVERSITY OF BEIRUT

A Hybrid Game Theory and Reinforcement
Learning Approach for Cyber-Physical Systems
Security

by
Joseph C. Khoury

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Department of Computer Science
of the Faculty of Arts and Sciences
at the American University of Beirut

Beirut, Lebanon
December 2019

AMERICAN UNIVERSITY OF BEIRUT

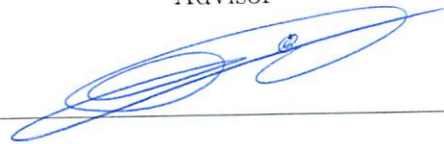
A Hybrid Game Theory and Reinforcement Learning Approach for Cyber-Physical Systems Security

by
Joseph C. Khoury

Approved by:

Dr. Mohamed Nassar, Assistant Professor
Computer Science

Advisor




Dr. Wassim El-Hajj, Associate Professor
Computer Science

Member of Committee



Dr. Haidar Safa, Professor
Computer Science

Member of Committee

Haidar Safa


Date of thesis defense: December 19, 2019

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Khairy Joseph Chawki
Last First Middle

Master's Thesis Master's Project Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One ___ year from the date of submission of my thesis, dissertation or project.**
Two ___ years from the date of submission of my thesis, dissertation or project.
Three ___ years from the date of submission of my thesis, dissertation or project.

Joseph January 31, 2020
Signature Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

Acknowledgements

To my small family..

An Abstract of the Thesis of

Joseph C. Khoury for Master of Science

Major: Computer Science

Title: A Hybrid Game Theory and Reinforcement Learning Approach for Cyber-Physical Systems Security

Cyber-Physical Systems (CPS) like smart cities and industries 4.0 including nuclear power plants, oil and gas pipelines, electric power grids, railways, and other Critical Infrastructures (CI) are monitored and controlled by Supervisory Control and Data Acquisition (SCADA) systems that use advanced computing, sensors, control systems, and communication network. At first, CPSs were protected and secured by isolation. However, with recent industrial technology advances, the increased connectivity of CPSs and SCADA systems to enterprise networks has uncovered them to new cybersecurity threats and made them a primary target for cyber-attacks with the potential of causing catastrophic economic, social, and environmental damage. This thesis work proposes two complementary cybersecurity risk assessment approaches to evaluate and assess cybersecurity for CPS networks extensively. First, we propose a game-theoretical model for cybersecurity in Industrial Control System (ICS) using Monte Carlo simulations

to evaluate the payoffs, given different variants of randomness, selected strategies, budget spending, and look-ahead. Second, we propose a refined approach to frame CPS security in two different levels, strategic and battlefield, by meeting ideas from both game theory and Multi-Agent Reinforcement Learning (MARL). The strategic level is modeled as imperfect information, extensive form game. Here, the human administrator and the virus author decide on the strategies of defense and attack, respectively. At the battlefield level, strategies are implemented by machine learning agents that derive optimal policies for run-time decisions. The outcomes of these policies manifest as the utility at a higher level, where the aim is to reach a Nash Equilibrium (NE) in favor of the defender. A framework is implemented to simulate the scenario of a virus spreading in a realistic CPS network. Promising results show that the defender can learn optimal policies to counter viruses that could be equipped with Artificial Intelligence (AI) components.

Contents

Acknowledgements	v
Abstract	vi
1 Introduction	1
1.1 Problem Definition	2
1.2 Goals	4
1.3 Document Structure	5
2 Background Information	7
2.1 SCADA in CPS	7
2.1.1 Overview	7
2.1.2 Threats	8
2.1.3 Attacks	10
2.2 Risk Assessment	11
2.2.1 What Is Risk Assessment?	11
2.2.2 Monte Carlo Method	12
2.2.3 Preexisting Risk Assessment Frameworks	13
2.3 Attack Trees	15
2.4 Game Theory	17
2.4.1 What Is Game Theory?	17
2.4.2 Zero-sum Game vs. Non-zero-sum Game	21
2.4.3 Perfect Information Game vs. Imperfect Information Game	22
2.4.4 Deterministic Game vs. Stochastic Game	22
2.4.5 Example About A Game Theory Approach for Cybersecurity	23
2.5 Reinforcement Learning	23
2.5.1 What Is Reinforcement Learning?	24
2.5.2 Exploration vs. Exploitation	25
2.5.3 On-Policy/ Off-Policy	25
2.5.4 Reinforcement Learning Algorithms	25
2.5.5 Multi-Agent Reinforcement Learning (MARL)	27

3	Literature Review	28
3.1	Adversarial Reinforcement Learning in a Cyber Security Simulation [1]	28
3.2	Evaluation of Reinforcement Learning-Based False Data Injection Attack to Automatic Voltage Control [2]	31
3.3	A Game-Theoretic Approach to Cross-Layer Security Decision-Making in Industrial Cyber-Physical Systems [3]	33
3.4	A Multistage Game in Smart Grid Security: A Reinforcement Learning Solution [4]	36
4	Methodology	38
4.1	A Game Theoretical Model using Monte Carlo Simulations	39
4.1.1	Model Advantages	40
4.1.2	Network Layer	40
4.1.3	Security Layer	41
4.1.4	Game Layer	43
4.1.5	Implementation Setup	46
4.1.6	Simulation Results	47
4.1.7	Closing Notes	54
4.2	A Hybrid Game Theory and Reinforcement Learning Approach for Cyber-Physical Systems Security	55
4.2.1	Introduction	57
4.2.2	Proposed Model	58
4.2.3	Strategic level	58
4.2.4	Battlefield level	59
4.2.5	Implementation Setup	67
4.2.6	Hyperparameters Tuning	70
5	Discussions and Results	72
5.1	Discussions	72
5.1.1	Strategy Vs. Policy	72
5.1.2	Game Strategies	73
5.2	Results	74
5.2.1	MARL Policies	74
5.2.2	Testing Strategies With Learned Policies	75
6	Conclusion	78
A	Abbreviations	81

List of Figures

1.1	Overall problem definition.	3
2.1	SCADA system main components.	8
2.2	Attack tree example.	17
2.3	Prisoner’s dilemma payoff matrix.	18
2.4	Nash Equilibrium for the prisoner’s dilemma.	19
2.5	Kodak Vs. Polaroid game tree.	20
2.6	Backward induction equilibrium.	21
2.7	Reinforcement learning environment, agent, input, and outputs.	25
2.8	Multi-Agent reinforcement learning (MARL) environment, agent, inputs, and outputs.	27
3.1	The game from the Attacker and Defender perspective.	31
3.2	State transition of the POMDP during FDI attack.	33
3.3	Attacker game state transition graph.	35
4.1	Example of a functional network layer	41
4.2	Example of an attack tree in the security layer	43
4.3	An example of a cybersecurity strategic response model	43
4.4	Modbus slave attack response module	44
4.5	Example of a game in the game theory layer	46
4.6	Random strategy for defender vs. Random strategy for attacker	49
4.7	Random strategy for defender vs. Look-ahead = 2 for attacker	51
4.8	Greedy budget spending for defender	53
4.9	Conservative budget spending for defender	53
4.10	Uniform budget spending for defender	54
4.11	Our approach in a nutshell: cyber-physical systems, human agents and machine agents, proposed modeling solution.	56
4.12	Proposed CPS network topology using MiniCps.	59
4.13	Proposed game model for virus spreading in a multi-subnet and cross-layer CPS network.	64
4.14	Proposed framework architecture using OpenAI Gym toolkit, and MiniCps network simulator.	68

5.1	Strategy Vs. Policy.	73
5.2	Win-factor for learned attacker vs. random defender.	75
5.3	Win-factor for learned attacker vs. learned defender.	76
5.4	Testing only attacker strategies with derived policies.	77
5.5	Testing both attacker and defender strategies with derived policies.	77

List of Tables

4.1	Sampled vulnerability analysis gathered from CVE details [5]. . .	61
4.2	Hyperparameters tuning.	70

Chapter 1

Introduction

Supervisory Control and Data Acquisition (SCADA) deployed in Cyber-Physical Systems (CPS) such as smart cities and industry 4.0 like nuclear power plants, oil and gas pipelines, electric power grids, railways, and other Critical Infrastructures (CI) use advanced computing, sensors, control systems, and communication networks to monitor and control industrial processes and distributed assets. These systems, due to the extreme impact level on human lives, are ranked first among the bank of targets for malicious attackers or even for hostile countries. They are also considered as a Weapon of Mass Destruction (WMD) because they can bring notable harm to a high number of humans or cause considerable damage to human-made structures, natural structures, or the biosphere. In 2010, the cyber world witnessed the arrival of the most technologically sophisticated and malicious program developed for a targeted attack. It was the arrival of the first cyber warfare weapon ever; known as Stuxnet[6]. It turns out after a lot of investigation and code analyzing that Stuxnet, was only targeting Iran in its Natanz uranium enrichment plant. The attack was not against SCADA software; it was aimed at industrial controllers that might or might not be connected to a SCADA

system [6]. Stuxnet's dropper loaded rogue code to PLCs made by Siemens, it went through a complex process of fingerprinting to find the target, and it utilized at least four zero-day Windows vulnerabilities to gain access to vulnerable computers and search for Siemens PLC software [7]. Since then, similar attacks proliferated, such as Havex and Industroyer¹.

1.1 Problem Definition

Back in the days, CPSs were protected and secured by isolation from the Internet. These systems are distributed in nature and have control loops receiving measurements from sensors and transmitting control signals to actuators. However, with the technological industrial advances, CPS and SCADA systems are being exposed more and more to the Internet. They are growingly using standard computing and networking technologies, such as TCP/IP, Modbus, remote access, and web services, to transport sensor data and control signals, to monitor and control industrial processes. Additionally, the interconnection between these systems and organizational enterprise networks is hugely increasing. These facts are only making these systems vulnerable and prone to cyber-attacks. By just visiting the Shodan Search Engine [8] and searching for connected ICS devices, 55000 ICS devices distributed around the globe are scanned and displayed, with different protocols connected to the Internet [9]. Protecting these systems with available security solutions adapted in traditional networks is very revealed and not sufficient. Security in conventional networks is primarily concerned with confidentiality and data integrity, unlike the security in SCADA networks that are concerned with availability, reliability, and safety.

Traditional network security tools such as Intrusion Detection Systems (IDS),

¹<https://www.welivesecurity.com/2017/06/16/seven-years-stuxnet-industrial-systems-security-spotlight/>



Figure 1.1: Overall problem definition.

firewalls, and the option of isolating these systems are no more an option or a solution. Attacks on these systems can be very sophisticated and targeted, such as Stuxnet, or inadvertent, such as Slammer; thus proving the urgent need for risk assessment approaches, and optimized defense policies in the will of protecting these systems against future attacks. We illustrate in Fig. 1.1 the scenario of possible targeted attacks in a red arrow initiated from malicious hackers or rogue viruses on the controlling systems of the ICS represented by several layers and subnets. And on the other hand, possible defense strategies in a green arrow initiated by the network administrator to block targeted cyber-attacks. How to model our problem? To model the problem, we tend to use methodologies from recent researches that focus on the use of game theory to formalize the problem as a game, and the use of Monte Carlo method to simulate most or all possible risks in cybersecurity areas. Alternatively, the use of Reinforcement Learning (RL) to add a learning component for our game players to derive optimal defense policies. Having solutions from both the Monte Carlo method and RL, we question the

order of use of both solutions. For that, we use the Monte Carlo method first to discover all possible risks that an ICS environment face, and second we use RL to add learning properties for game players to derive optimal attack sequences and defense policies.

1.2 Goals

Recent research focuses on new methodologies for risk modeling and assessment. An increasing interest in game theory, control theory, and network optimization are witnessed amongst other methodologies. The goal of this thesis effort is to propose two complementary approaches that can be used to assess cybersecurity solutions for CPS and ICS, and also to derive adequate defense strategies with optimized policies to mitigate cyber-attacks.

The first approach proposes a game-theoretical model for cybersecurity risk assessment in ICS using Monte Carlo simulations. This approach proposes a risk assessment model which is composed of three layers: (1) a functional network layer that represents a typical industrial company network, (2) a security layer that represent the different security states of the network, (3) and a game theory layer where the environment, action spaces, utilities, and strategies of a cyber battle is modeled between an attacker and a defender. A tool is developed in Python to represent and configure the different layers and probabilistic parameters and to run the simulations [10].

We refine upon the first approach by introducing Artificial Intelligence (AI) Components on both the defender and the attacker sides. This approach is based on a hybrid human-machine model. The proposed model is based on game theory and RL, where human adversaries decide on strategies. For instance, how to deal with security situations with different intensity levels. From the defender's point

of view, which defense components are deployed, how firewalls are configured, how system patching is managed, etc. From the attacker’s point of view, which infiltration techniques are used, how the malware is packaged, etc. Based on the formulated game and decided strategies, RL agents decide on the best actions (proactive or reactive) to take over the battlefield. Based on the knowledge of the network and the alerts of the defense systems, the defender chooses actions and navigates its own state space. The attacker navigates a parallel state space based on an attack tree. Therefore, the Multi-Agent Reinforcement Learning (MARL) agents learn the best policy for an automated response at run-time. These best policies for both the attacker and the defender manifest themselves as the utilities at the strategic game level. Therefore, we can compute a Nash Equilibrium (NE) at the strategic level that represents the best possible strategy in the presence of the adversary. A NE is a point that no party would like to deviate from without losing utility. We focus in particular on a virus spreading scenario, with realistic assumptions such as discovering and using zero-day vulnerabilities. We run simulations using the MiniCPS simulator, and we use OpenAI Gym for the MARL implementation [10].

1.3 Document Structure

Chapter II presents background information about CPS in general and SCADA Systems in particular. Detailed information is also given about the threats in CPS and SCADA systems and the attack vectors. A thorough discussion about risk assessment, and the Monte Carlo method. We also present existing tools, and attack trees. Also, a detailed discussion about game theory is given with an example that shows the use of this formal tool in the cybersecurity field. In the end, an overview of RL and MARL is given, while focusing on its central concepts.

Chapter III discusses literature review and work related to game theory that uses RL algorithms as a solution to find optimal defense strategies. Limitations of the presented related work are also discussed.

Chapter IV presents two approaches. First, a game-theoretical model using Monte Carlo simulations. Second, an extended model including RL agents.

Chapter V shows related discussions and results.

Chapter VI concludes the thesis work.

Chapter 2

Background Information

Chapter II presents background information about SCADA systems in CPS. A discussion is held about risk assessment in CPS, in addition to tools for risk assessment in CPS, related work and background information about attack trees, game theory, Monte Carlo simulations, and MARL with RL algorithms.

2.1 SCADA in CPS

This subsection presents information on the different components of SCADA systems in CPS and their architecture. SCADA field devices and all the components found in the SCADA control centers are introduced. Next, threats that target SCADA systems, CPS, and control units in addition to highlights on the methods that help gain access to SCADA systems. Finally, information on inadvertent and targeted attacks on SCADA systems in CPS are discussed.

2.1.1 Overview

SCADA systems are highly distributed systems used to supervise and control geographically scattered assets where centralized data acquisition and control are crucial to system operation. Popular SCADA networks include water treatment

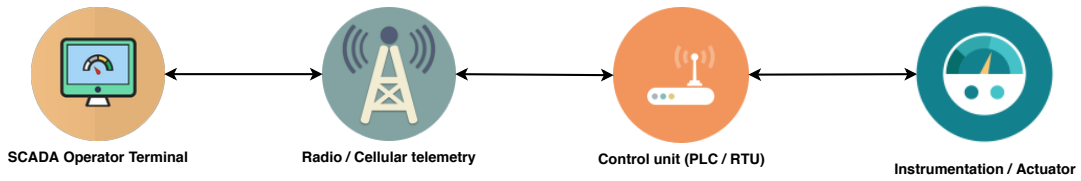


Figure 2.1: SCADA system main components.

facilities, oil and gas pipelines, railways, electric power grids, and more. [11]. Common devices to SCADA networks include mechanical and computerized ones, thus the need for safety, availability, and reliability requirements. SCADA networks include control centers that perform centralized monitoring and control for field and edge devices over long-distance communication networks. SCADA control centers monitor and process communications such as alarms and status data transmitted by SCADA field devices[11]. Fig. 2.1 shows the SCADA system main components: First, we have SCADA operator terminal represented by a Master Terminal Unit (MTU), Data Historian, and a Human Machine Interface (HMI). Second, we have a radio or cellular telemetry serving as a transmitting medium. Third, we have the on-field control unit represented by Remote Terminal Unit (RTU), Programmable Logic Controller (PLC), or Intelligent Electronic Device (IED). And finally, instrumentation to read important data and actuators to apply forces when required. The SCADA operator receives instrumental readings; received readings are stored on the HMI to be later analyzed, based on these readings control messages are sent through the radio/cellular telemetry passing by the controlling unit to be finally applied by the actuator on the industrial process.

2.1.2 Threats

In [11], the authors provided a list of possible incidents that can target SCADA systems in CPS.

- Blocked or delayed flow of information, which could disrupt CPS operations.

- Unauthorized changes to commands, or alarm threshold which could damage and disable equipment and subsequently create environmental impact, and endanger human life.
- Tampered information sent to system operators, to cause the operator to initiate undesired actions, which will result in negative consequences.
- Infected CPS software or settings with malware or rogue scripts that can have adverse effects.

The authors in [11], also highlight the potential goals of an adversary once access to a SCADA system is gained. In the past, network isolation was the primary feature of SCADA systems. It was isolated from all other networks and the Internet. An adversary needed to gain physical access to the SCADA system to perform rogue acts. Due to the technological industrial advances, businesses were obliged to connect their segregated SCADA network to enterprise networks to cut costs and improve connectivity, and this indirectly connects SCADA systems to the Internet. This allows adversaries to infiltrate the business network, find the SCADA network, and begin issuing rogue commands to the SCADA network [12]. Even if SCADA network is not connected Leverett [13], in his work, mention that some field devices are directly connected to the Internet, negating the need to penetrate business networks. SCADA systems also allow some remote connections to authorized individuals, and these connections provide an additional attack space for an adversary. In case, an adversary manages to have access to SCADA systems; they can issue rogue commands that lead to catastrophic results on the industrial process, nature, or human life. Intentional cyber-attacks can occur from both insiders and outsiders. Although both can issue attacks on these systems; insider attacks are more dangerous than outsider attacks mainly because

an insider has better information about internal-procedures and potential weak spots in the systems [14].

2.1.3 Attacks

Several attacks in the past have targeted CPS in general and SCADA systems in particular; all these were proof that these systems are vulnerable and prone to adversarial cyber-attacks. Slammer reported by Dacey [15] is a Microsoft SQL Server worm that infected a private computer network at Davis-Besse nuclear power plant in Ohio. This worm exploited an SQL vulnerability usually found in conventional Information Technology (IT) networks, that affects the power plant indirectly. Slammer is an example of an inadvertent attack that causes the plant's process computer to malfunction and generates so much traffic that degraded SCADA network communications. Slammer also affected the network traffic of other physical systems like airline reservation systems [7]. Ralph Langner[6], consider that Stuxnet is the most technologically sophisticated malicious program developed for a targeted attack. Stuxnet impacted the centrifuges of the Natanz uranium enrichment facility. The virus targeted control systems running a Siemens PLC, an utilized at least four different Windows zero-day vulnerabilities to gain access to a computer and search for the Siemens PLC software [7]. Stuxnet was able to affect the system through the network and threw thumb drives [16]. Attacks demonstrate the immense threats to SCADA networks and thus prove the need to implement additional security solutions. Traditional security solutions may sound feasible for SCADA systems. However, in the case where SCADA networks are interconnected to organizational networks, traditional IT security solutions must be tested first, because they might not be easily extended to SCADA networks [17]. Failing to test such solutions may result in shutting down sensitive networks, production and revenue losses, or harm to human life

or the environment.

2.2 Risk Assessment

2.2.1 What Is Risk Assessment?

Risk assessment [18] is a way to abstract out the technical security details and come up with a qualitative or quantitative description of the security situation. It is a systematic process of evaluating potential risks. This process starts with risk identification, passes by a risk impact assessment, and comes up with final response and mitigation steps to avoid the risks or reduce their impact [19]. Many ICS environments deploy monitoring tools and Security Information and Event Management (SIEM) solutions to support compliance and qualitative risk assessment methodologies such as the NIST cybersecurity framework [20] and ES-C2M2 [21]. Risk measurement is all-determining the probability of attacks and their potential impact. Suppose a system security officer for a drinking water utility servicing a small city of about 30,000 residents and assuming that the risk assessment has identified that with a very low probability, a cyberattack could shut down the water supply to the city for a week or longer. Despite the very low probability, this kind of impact is considered critical, and countermeasures for risk aversion must be taken [22]. Selecting and prioritizing the countermeasures are based on an elaborate cost-benefit analysis.

Risk Analysis and Management for Critical Asset Protection (RAMCAP) [23] is one of the earliest quantitative models that defines risk as to the product of threat (T), vulnerability (V) and consequences (C):

$$R = T \times V \times C$$

The Return on Investment (ROI) of a prevention objective is the RAMCAP (R) divided by the implementation cost ($\$$):

$$ROI = R/\$$$

Therefore RAMCAP can be used to perform a cost-benefit analysis and prioritize the available countermeasures based on their ROI . The problems with most risk assessment standards and methodologies are ([19]):

- the employed definitions are sometimes inconsistent, too general, or lack rigor.
- the applicability of risk assessment models is, most of the time, limited to the analyst knowledge, business context, and subjective interpretations.
- databases of risk vulnerabilities and responses are most of the time, not exhaustive and not up-to-date, which largely affects quantitative models.
- many standards have emerged without substantial differences.

Sometimes, even the basic security requirements for the simplest devices are not fulfilled. Experts argue that current standards are not the ultimate solution for risk assessment ([24, 25]). In this thesis work, an alternative solution to ICS risk assessment based on simulating realistic pen-testing games against the ICS network model is proposed in one of the approaches.

2.2.2 Monte Carlo Method

Monte Carlo simulation is a method used to learn the impact of risk and ambiguity in financial, project management, and other forecasting models. A Monte Carlo simulator helps one conceive most or all of the possible results to have a better idea about the risk of a decision; it uses the process of repeated

random sampling to estimate unknown variables. Many risk assessment tools were built and implemented using the Monte Carlo method to simulate all the possible outcomes and afterward derive best practices to avoid the risks. A popular example of the use of the Monte Carlo method is the estimation of π . Assume that we have a circle of radius 0.5 enclosed in a 1×1 square, the area of this circle is $\pi r^2 = \frac{\pi}{4}$. We generate a number of random points n on the enclosed square, and we keep track of the number of points N generated inside the circle. By calculating $\frac{N}{n}$ we approximate the area of the circle, which is $\frac{\pi}{4}$.

$$\frac{\pi}{4} \approx \frac{N}{n}$$

$$\pi \approx 4 \times \frac{N}{n}$$

With the increase of n ; the approximation of π will become more accurate.

2.2.3 Preexisting Risk Assessment Frameworks

In this section, background information will be given about risk assessment frameworks that have used the Monte Carlo method to simulate most or all possible risks in cybersecurity areas.

CySeMoL

The CyberSecurity Modeling Language (CySeMoL) [26] was proposed to support ICS and SCADA security managers. CySeMoL is a modeling language based on logical relations, experimental research in the security domain, and experts judgment. It covers a variety of attacks, including software exploits, flooding attacks, privilege abuse, and social-engineering attacks. The language is coupled with a probabilistic inference engine. The probabilities used by the engine are a compilation of research results in many security domains. The probabilities are validated on a component level and a system level. Case studies show that CySeMoL assessments compare to the assessments of a security professional in

terms of correctness.

Haruspex

Haruspex [27] is a software tool that applies a Monte Carlo method to simulate intelligent and adaptive threat agents. The agents aim to reach predefined goals through plans with several attack sequences. Haruspex design is strongly model-oriented and focused on cyber-physical systems such as gas distribution, thermo-electric, and hydro-electric generation. Several modules are designed to model the target system, the attack agents, the attack simulations, and the output analysis. One of the metrics proposed by Haruspex is security stress, which is a measure of resistance against attacks. Formally the security stress is the probability that an attacker reaches its assigned goal within a time frame t . The security stress curves show better how the network resists to the attackers. The security model of Haruspex is a directed acyclic graph that describes all the attack plans a threat agent can implement to achieve one of its goals. Each node in this graph represents a set of rights. An arc between two nodes implies that the rights of the source node are in the pre-condition of the attack step, and those of the target node is in the post-condition of the attack step. A right is defined as a pair composed of one component and one attribute. A sequence of simple attack steps represents the acquisition of rights through a plan and is modeled by a path in the graph. The attack is successful if any final node is reached through a successful path execution. Monte Carlo simulations are required by this approach since many parts of the model are non-deterministic, for instance:

- the probability of discovering a system vulnerability
- the success probability of an attack step
- the choice among alternative attack steps in a plan

The framework allows vulnerabilities to be discovered or patched during the simulation by adding or removing arcs in the graph. The attacker’s strategy for selecting the next steps can be random or using a look-ahead with a given number of steps. Countermeasures are integrated by removing arcs in the graph. The simulator returns a set of countermeasure recommendations taking into account the cost of each countermeasure and its benefit. The benefit of a countermeasure is the number of attack plans sharing the corresponding arc. Haruspex allows for dynamic or conditioned countermeasures. For example, an intrusion detection system may raise the alarm to switch some connections off and prevent a plausible exploit

2.3 Attack Trees

A survey back to 2007 discusses the cybersecurity risk assessment methods of ICS networks and enumerates the organizations disseminating standards and best practices [28]. Fault Tree Analysis (FTA) [29] is a Probabilistic Risk Assessment (PRA) method. It is a deductive approach that starts with a failure or an undesired event. Although FTA is not originally designed in the context of information security or cybersecurity, extending it to build attack trees as proposed in [30] is straightforward. Basic event probabilities can replace the failure rates of events. In this thesis context, the failure can be a compromise of the controller, providing wrong sensor measurements, or unavailability of the system. FTA applies systematic backward reasoning to deduce the causes. An attack tree represents the events and their dependencies. AND and OR gates represent relationships. The AND gate outputs an event if all input events occur. The OR gate outputs an event if at least one input event occurs. A minimal cut set is the smallest number of basic events resulting in the failure or a security breach.

Hence the probability of a breach can be quantified based on the probabilities of the basic events. This representation is also useful to quantify the impact of a basic countermeasure (i.e., by setting the probability of the corresponding basic events to zero in the model).

In contrast to FTA and attack trees, inductive methods work in the opposite direction. They start with the initiating events and trace forward to the consequences. FTA and inductive methods are used to conduct Monte Carlo discrete-event simulations [31] or to calculate fuzzy logic probabilities in a way that couples personal expertise with the probabilistic tree model [32]. The tree representation helps in evaluating the possible security countermeasures even when an actual probability value is not computed.

Very similar to the idea of attack trees is the idea of vulnerability trees. In a vulnerability tree, a path is modeled and afterward can be followed by a threat agent to exploit a targeted vulnerability. Patel et al. [33] has merged the two ideas (attack trees and vulnerability trees) in one augmented vulnerability tree approach and proposed two metrics to quantify the risk: the *threat-impact* index and the *cyber-vulnerability* metric. The *threat-impact* index measures the economic impact of a successful attack while the *cyber-vulnerability* metric measures the vulnerability level.

In [34], attack trees have been applied to SCADA systems with eleven identified attacker goals. The outcome of their analysis is translated into best practices and suggested improvements to the MODBUS protocol. In [35], pivotal leaves in the tree are identified by solving an optimization problem. The results are used to suggest security improvements. Below we show an attack tree example from [36].

In this example, the root node depicts the main attacker's goal, which is to

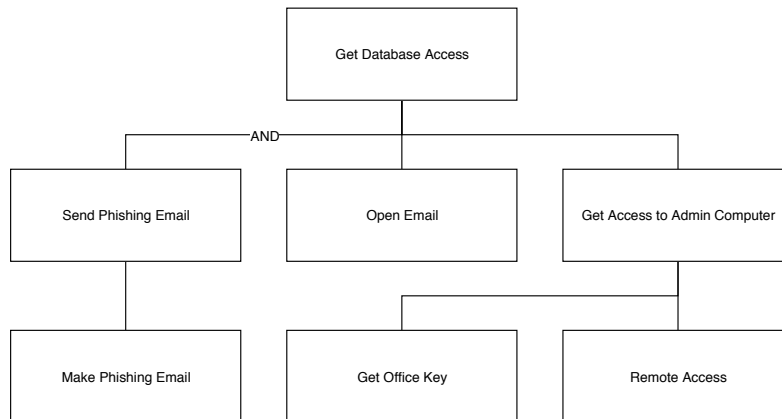


Figure 2.2: Attack tree example.

compromise the database. To achieve the desired goal, the attacker may follow two paths. An attack tree allows decomposing the primary attack goal into sub-goals, making the leaf nodes the starting points to achieve the ultimate goal.

2.4 Game Theory

2.4.1 What Is Game Theory?

Game theory is about agents that are self-interested and interacting through a shared environment. Each agent has its description of the states of the world and which states it likes most. Each agent acts based on this description and has a utility function. This function quantifies the degree of preference across alternative actions and models the impact of uncertainty. The agents have decision-theoretic rationality in the sense that each one of them acts to maximize its expected utility in the long run. The game representation takes one of two forms:

- Normal form (usually represented as a table).
- Extensive form, where the timing of the moves and the knowledge about the previous actions of other players can be modeled.

<div style="display: flex; justify-content: space-between;"> A B </div>	Confess	Deny
Confess	(5, 5)	(0, 10)
Deny	(10, 0)	(1, 1)

Figure 2.3: Prisoner's dilemma payoff matrix.

Normal-Form Game

A two players normal-form game is defined as follows:

- Action set for player i : A_i
- Utility function or payoff for player i : $u_i: A_i \rightarrow \mathbb{R}$

The normal-form game is usually represented by a matrix showing the players, strategies, and payoffs. More commonly, it can be represented by any function that associates a payoff to every possible combination of actions. $a = (a_1, a_2) \in A_1 \times A_2$ is an action profile and, $u = (u_1, u_2)$ is a utility profile.

Below, we discuss an example of a normal form game, the famous prisoner's dilemma to present the concept of Nash Equilibrium (NE):

In this dilemma, we have two prisoners in a separate room; the police offer each a deal. Confess, and accuse the other prisoner and earn a zero-day in prison if the other player denies. If both players deny, they each end up in prison for one year. If both players confess, they each go to prison for five years. Fig 2.3, is the payoff matrix of the prisoner's dilemma normal form game:

A NE is reached if no prisoner can change its strategy and improve its payoff. If both prisoners decide on denying, then both are sentenced to one year of

	B		
A	Confess	Deny	
Confess	(5, 5) Nash Equilibrium	(0, 10)	←
Deny	(10, 0)	(1, 1) Global Optimal	←

Figure 2.4: Nash Equilibrium for the prisoner's dilemma.

prison. One player can deviate decreasing its sentence to zero while the other to ten years. This means that (Deny, Deny) is not a NE. Now consider for prisoner A and B, (Deny, Confess) strategy, respectively. If prisoner A deviates to a confess strategy, his prison sentence decreases to five while increasing B prison sentence from zero to five. This means (Deny, Confess) is not a NE and by symmetry (Confess, Deny) is also not a NE. Finally, consider a (Confess, Confess) strategy where no prisoner is left with a choice to deviate for a lesser prison sentence. Thus, (Confess, Confess) is the NE of the prisoner's dilemma. Fig. 2.4 shows the NE of the prisoner's dilemma.

Extensive-Form Game

An extensive-form game is defined as follows:

- Set of players: N

And for each player i :

- The set of strategies: S_i
- The payoff function: $u_i: S_i \rightarrow \mathbb{R}$

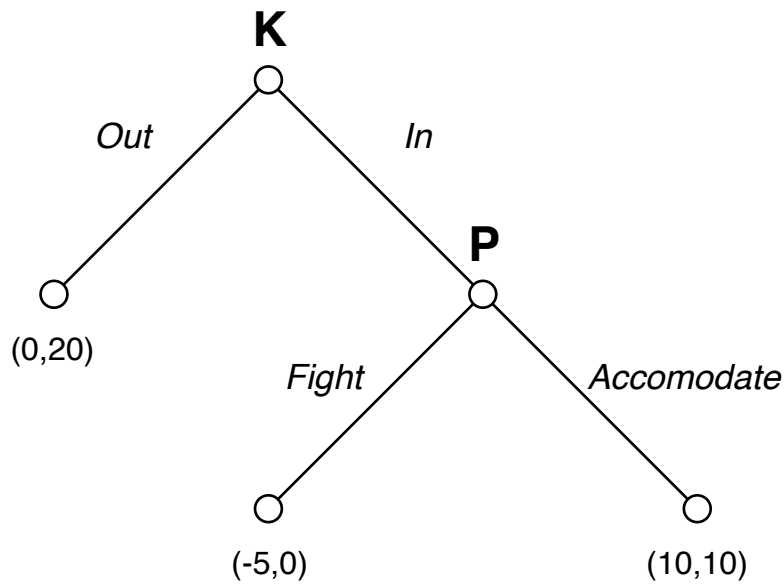


Figure 2.5: Kodak Vs. Polaroid game tree.

Extensive-form games formalize a time sequencing of moves based on trees. Each vertex in the game tree represents a point of choice for a player. The ramifications of the vertex represent possible actions for that player. Payoffs are defined at the bottom of the tree. Backward induction is commonly used to evaluate the game.

Below, we discuss an example of a perfect information extensive-form game: It is a competition between Kodak trying to enter the instant photography market, and Polaroid that can either accommodate this market entry or fight it [37]. Fig. 2.5 shows the tree of the game.

We can induce a normal form game, by converting the extensive form game into a normal form game.

- $N = \{K, P\}$
- $S_K = \{Out, In\}, S_P = \{F, A\}$
- Payoff matrix.

K \ P	<i>F</i>	<i>A</i>
<i>Out</i>	(0, 20)	(0, 20)
<i>In</i>	(-5, 0)	(10, 10)

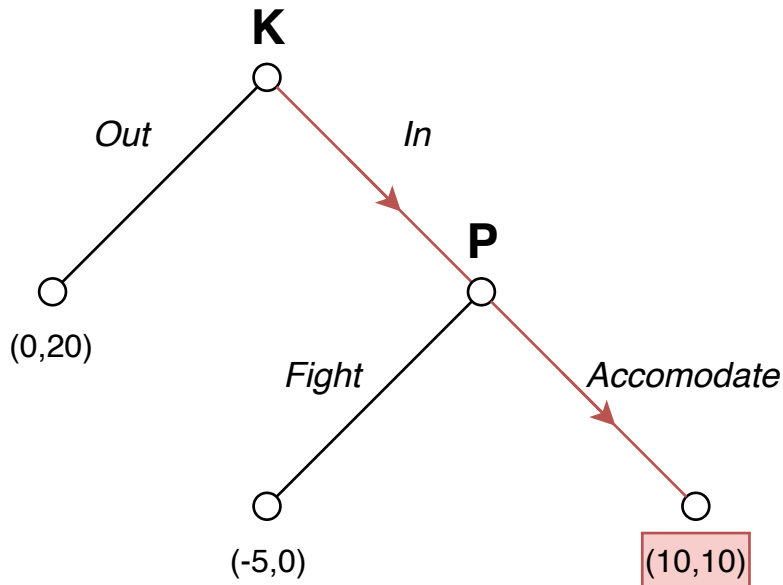


Figure 2.6: Backward induction equilibrium.

This example is a backward induction equilibrium; at a backward induction equilibrium, each player plays optimally at each decision node of the game tree. So, in this case, the unique backward induction equilibrium is (In, A) . Knowing that backward induction equilibrium eliminates equilibria based on incredible threats. Also, we can notice that (Out, F) is sustained by an incredible threat from Polaroid. Finally, the set of Nash equilibria is: $(In, A), (Out, F)$. Fig. 2.6 shows the backward induction equilibrium of the Kodak vs. Polaroid game.

2.4.2 Zero-sum Game vs. Non-zero-sum Game

In game theory, we have two concepts of games: a zero-sum game, which is considered as complete competitive games where what is considered suitable

for one is terrible for the other. Furthermore, in a zero-sum game, an optimal solution can always be found. The non-zero-sum game differs from a zero-sum game in that there is no single optimal strategy. Also, the non-zero-sum game is not strictly competitive because such games generally have both competitive and cooperative elements.

2.4.3 Perfect Information Game vs. Imperfect Information Game

Two types of games are derived from the zero-sum game: perfect and imperfect games. In a perfect information game, every player knows the results of all previous moves; such games include chess and tic-tac-toe. In this type of game, there is at least one best strategy; this strategy does not guaranty winning the game but reducing the losses. On the other hand, in imperfect information games, the players do not know all the previous moves; this is due because the players often play simultaneously.

2.4.4 Deterministic Game vs. Stochastic Game

In a deterministic game, we specify a choice of action for each player at every stage of the game. In games where players take turns, the policy is considered deterministic. In this case, it is optimal for the move to be a function of the state. An example of a deterministic game is chess because if we use the same moves at each game, we will obtain the same outcomes. In a stochastic game, we have a probability of transition at each stage of the game. In games where the players move simultaneously, the policy is considered non-deterministic. In this case, an optimal player will randomly choose an action at each stage of the game; this means that the probability is a function of the state. Stochastic games have applications in economics, computer networks, and others.

2.4.5 Example About A Game Theory Approach for Cybersecurity

Fielder et al. [38] suggest a game-theoretic model that optimally allocates cybersecurity resources such as defender's time across different tasks. In their work, non-zero-sum games represent the environment. Eventually, a mixed NE between the attacker and the defender (e.g., administrators of the system) is reached, because the actions taken are not deterministic but rather regulated by probabilistic rules. The defender Nash strategy is the solution to a minimax problem. An efficient minimax solver is proposed using Singular Value Decomposition (SVD). Enhancing the different model components requires expert feedback from system administrators.

The first approach in this thesis adopts a similar game theory approach. It proposes a new model of risk assessment based on simulating a stochastic extensive-form game. The game represents attacker versus defender interactions across a typical industrial cyber-physical system network. The environment is more realistic and, therefore, complicated. Monte Carlo simulation methods are used to estimate the utilities of attacker and defender. The second approach is also modeled as an imperfect information extensive-form game to formalize the interaction between agents. Both approaches are detailed in chapter IV.

2.5 Reinforcement Learning

In this section, an overview is given on Reinforcement Learning (RL). It focuses on central topics in RL, in addition to a discussion about Multi-Agent Reinforcement Learning(MARL).

2.5.1 What Is Reinforcement Learning?

RL dates back to the early days of cybernetics and works in statistics, psychology, neuroscience, and computer science [39]. It gained much attention in the last few years in the area of machine learning and Artificial Intelligence (AI). RL defines how an agent learns through trials and errors in a dynamic environment to maximize utility.

RL is defined by: (1) set A , represent all the possible actions that an agent can take. (2) set S , represents all the possible states in an environment. (3) R is an immediate reward to evaluate the agent's action $(S, A) \rightarrow R$. (4) policy π that the agent employs to determine next action based on the current state $S \rightarrow A$. (5) Discounted expected long term reward V , it is the expected long term reward under a policy π ; $V = \sum \sigma^i R_i$. (6) Action value, is the long term return of the current state s , taking action a with policy π . In RL, the agent interacts with the environment via observation and allowed action space. At each step of interaction, the agent will apply a specific action a from the set A on a specific state s of the environment. This action will return output for the agent represented as a new state s' from the set S , and a reward R . This process keeps on running until the environment returns a terminal state, we call this an episode. The agent should choose actions that tend to increase the overall expected reward. This behavior can be learned by systematic trial and error, using a wide variety of available algorithms. By other means, the agent job is to find a policy that maps a state to action, in order to maximize long term rewards. Some important application for RL is related to search, human decisions, lately, network optimization and cybersecurity. In Fig. 2.7 a general glimpse is given about RL (environment, agent, action, state, and reward).

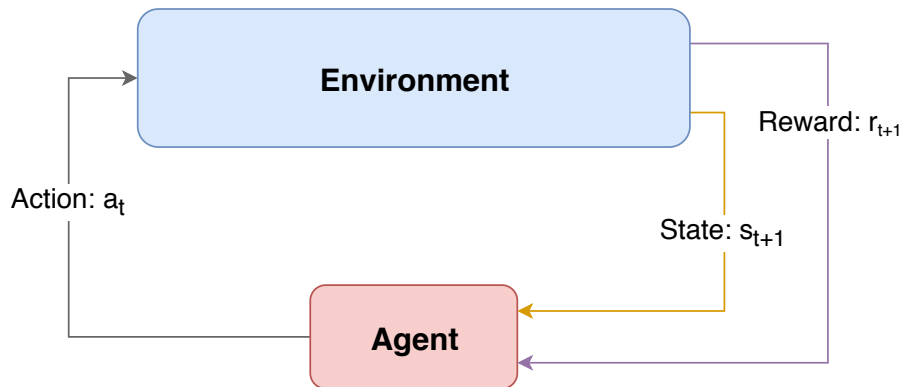


Figure 2.7: Reinforcement learning environment, agent, input, and outputs.

2.5.2 Exploration vs. Exploitation

RL is an online learning process. There are no data sets given to the model. Data is gathered as we go. Actions taken affect the type of received data. For that reason, it is good to take different actions to receive different results. This means exploration is used to discover the environment first, and gather more information that could lead to better results. However, exploitation is used to exploit and use the best decisions, given the information gathered about the explored environment so far.

2.5.3 On-Policy/ Off-Policy

A policy is a function that maps a state to an action. There are two types of policies: a deterministic policy, that means a specific state leads always to the same action, and a stochastic policy where a specific state leads to different actions based on a probability distribution.

2.5.4 Reinforcement Learning Algorithms

There is a wide range of algorithms that can be applied to RL, below is a comparison among the most popular and used algorithms.

In the case of an on-policy algorithm, we present an on-policy algorithm called

State-Action-Reward-State-Action (SARSA); when getting action A' from S' : $S' \rightarrow A'$ we use the same policy π used to brought action A from State S : $S \rightarrow A$

In the case of an off-policy algorithm, we need an algorithm that allows us to learn a policy that regulates the interaction with an environment under certain conditions to maximize rewards. An instance of an off-policy algorithm is the Q-learning algorithm. Q-learning is a traditional RL algorithm is about selecting an action given a state to maximize the overall reward. It is a tabular representation formed of actions as rows and states as columns and filled with values called the Q-values. The highest Q-value can map the best action to take considering a particular state. Below, we explain the Bellman equation that helps in filling what we call the Q-table with the Q-values.

$$NewQ(S, A) = Q(S, A) + \alpha[R(S, A) + \gamma max_{A'} Q'(S', A') - Q(S, A)]$$

Before learning begins, the Q-table is initialized with arbitrary or zero Q-values, then the Q-values for each state S and action A are updated with the above equation to update the Q-value for state S and action A . The equation starts by using $Q(S, A)$ an exiting Q-value for action A and state S , adding to it the learned value multiplied by the learning rate α . The learned value is formed by the reward $R : S \rightarrow A$ plus the discount factor of the estimated optimal future value, and from the learned value, we deduct the current Q-value of action A and state S . The Q-value can also be estimated by using a neural network, and the algorithm responsible for such estimation is called deep Q-learning. Deep Q-learning is much better than Q-learning if the number of states is large.

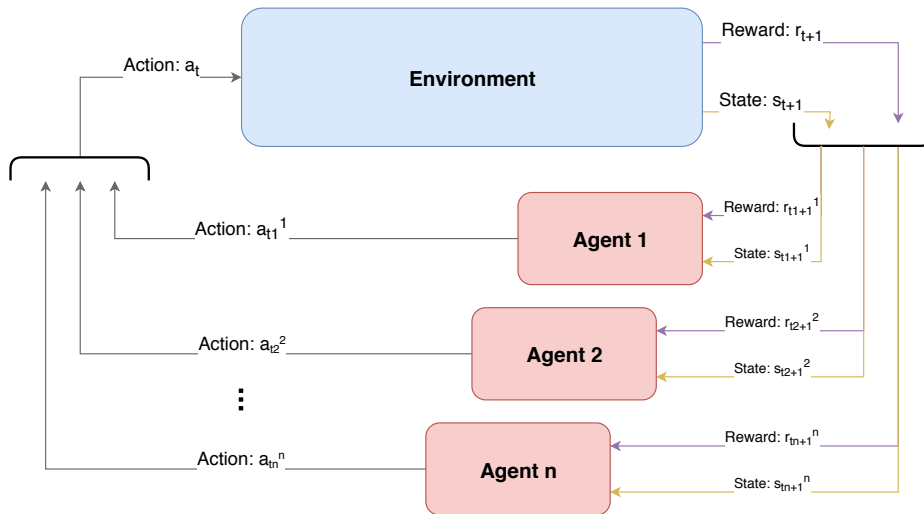


Figure 2.8: Multi-Agent reinforcement learning (MARL) environment, agent, inputs, and outputs.

2.5.5 Multi-Agent Reinforcement Learning (MARL)

Multi-Agent Reinforcement Learning (MARL), focuses on models that include two or more agents that learn by dynamically interacting on a shared environment. Different scenarios are created based on the action of each agent applied on the environment. The complexity of MARL is tightly related to the increase of agents number. In MARL, we have three types of behaviors; cooperative, adversarial, and neutral. In Fig. 2.8 a MARL context is presented with the interaction of multiple agent on a shared environment.

Chapter 3

Literature Review

Chapter III provides a literature review, and related work that has employed game theory in addition to RL algorithms to propose security decision-making approaches for ICS and CPS networks. In addition to deriving optimal defense strategies to reduce the loss caused by cyber attacks.

3.1 Adversarial Reinforcement Learning in a Cyber Security Simulation [1]

This paper [1] focuses on cybersecurity simulations in networks modeled as a Markov game with incomplete and stochastic elements. It is an adversarial sequential decision-making problem played with two agents, an attacker, and a defender. The two will use RL technique (Neural Networks, Monte Carlo Learning, and Q-learning) to examine their effectiveness against learning opponents. In [1], they addressed two problems:

- The uncertainty because of an adversary's inability to conduct detailed surveillance.
- Defender's uncertainty about attackers payoffs

In the simulation proposed, there are two agents, defender, and attacker. Each agent has limited access to the network, just like in the real world, and it has only influence on its side of the values in the nodes. Both agents have the goal of winning as many games as possible. The attacker goal is to attack the network and get to the asset. He only has an observation of the current attacked node. The attacker wins the game when he gets the asset. The defender's goal is to detect the attacker before reaching the asset. The defender has access to each node on the network.

At each time step, both agents choose an action from their set of possible actions. When an attack value is incremented, this indicates that the attack is successful; if the attack value outcomes the defense value, this means that the attacker has won the game. In this case, the attacker move to the next node. When the attack value of the attack type is lower than or equal to the defense value, the attack is blocked. In this case, the attack is detected with a probability always between 0 and 1. If the attack is detected, the game ends, and the defender wins. If the attack was not detected, another game step is played. At the end of each game, the winner gets a reward of +100 and the loser a reward of -100. It is a zero-sum game.

The agents in this simulation need to optimize their behavior, such that they win as many games as possible. Agents will base their decision based on the current state of the network. The entire state of the network is partially observable for both agents. Moreover, they have access to different kinds of information about the network. The agents also have different actions to choose from.

In their work [1], different RL algorithms (Monte Carlo learning, Q-learning, Neural Network, and Linear Network) are used with different exploration algorithms (ϵ greedy, Softmax, Upper Confidence Bound UCB-1, Discounted UCB) to find

the best strategies for both agents. Notable results show that for the defender agent, both algorithms based on neural networks perform worse than all algorithms based on tabular representations. This reflects the problem that neural networks have with adversarial learning. The network is slower in adapting to a continually changing environment than other algorithms, therefore being consistently beaten by attackers.

Another remarkable result is that Q-learning is among the best algorithms for the defender agent, while it is among the worst for the attacker agent. This is probably the case because the action-chains or optimal policy is less clear for the attacker. Another possible explanation for the Q-learning attacker underperforming the Monte Carlo ϵ -greedy algorithm could be that Q-learning takes slightly longer to find an optimal policy.

Below, in Fig. 3.1 we present the network simulated in [1], with its key information:

- **Attack starts** from node 'START'.
- **A_n**: Attack of type n .
- **V**: defense values.
- **D**: detection chance if attack fails.
- **N**: network node.
- **If** attack value overpowers defense value, attack is successful.
- **Else** attack is detected with a probability **D**.

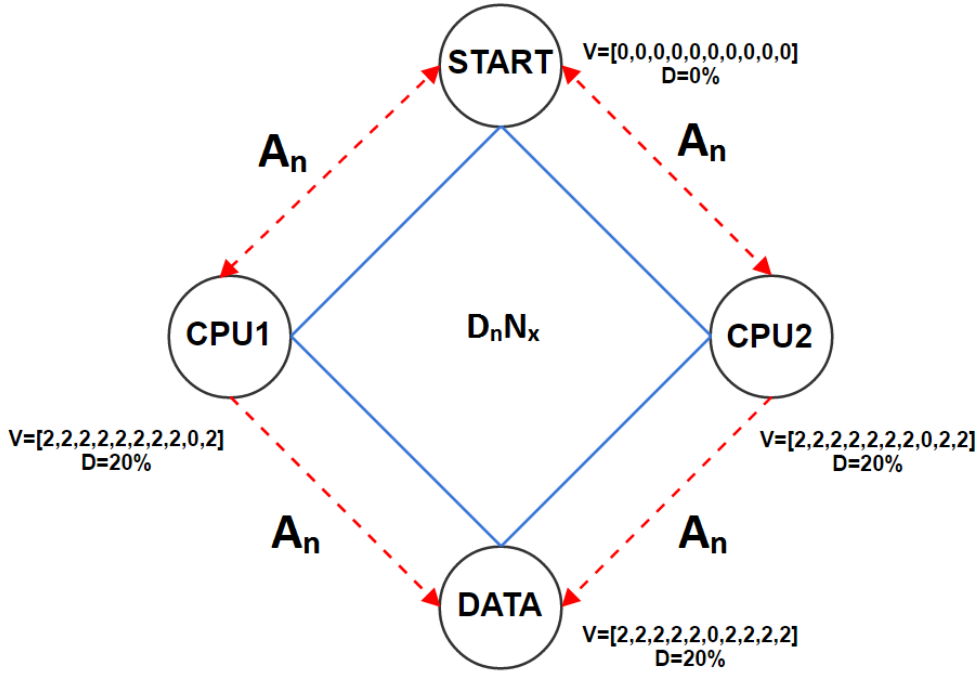


Figure 3.1: The game from the Attacker and Defender perspective.

3.2 Evaluation of Reinforcement Learning-Based False Data Injection Attack to Automatic Voltage Control [2]

False Data Injection (FDI) attacks intend to threaten the security of power systems. FDI attacks have been recognized as significant threats to smart grids. In [2], a novel strategy of FDI attacks is proposed, which aims to distort the regular operation of a power system regulated by Automatic Voltage Controls (AVCs).

In [2], FDI attacks to power systems with automatic voltage controls are studied, where the attacker is an online learner conducting attacks with data-driven strategies. The attack strategy is modeled as a Partial Observable Markov Decision Process (POMDP) problem. Attack actions are determined based on believed

state inferred from recorded observations. A Q-learning algorithm with Nearest Sequence Memories (NSM) is adopted because this problem is impossible to be solved analytically; this will help the attacker to reach optimal strategy by using RL algorithms. As a mitigation to the proposed FDI Attack, bad data detection and correction method is developed, which is based on Kernel Density Estimation (KDE). In [2], the authors assumed that the attacker is a malicious program inserted into the monitor and control unit of a substation. The attacker has no prior knowledge; it has to learn an effective attack strategy by trials and errors. The attacker should also be smart enough to attack and learn stealthily. Q-learning has been applied to solve POMDP models in many applications. K-nearest sequence memory was used to improve the efficiency and effectiveness of the Q-learning process. Results have shown that false data injection can create voltage collapse with minimal knowledge and information about the whole environment. Online learning helped in choosing attack moments to make the attack more stealthy. In this literature, the attacker's action was evaluated in the absence of a defender, and the information about vulnerable elements was not identified. To mitigate disruptive attacks, the authors proposed a bad data detection and correction by replacing susceptible data with its Maximum Likelihood Estimation (MLE) value. Test cases validated the feasibility of the proposed attack and the effectiveness of the mitigation approach. The authors in [2] offered in their future work an idea which is; the feasibility of FDI attacks based on RL and virus spreading.

Below, in Fig. 3.2 we show the proposed state transition in [2], with its key information:

- $\mathbf{a}(t)$: Attack action.

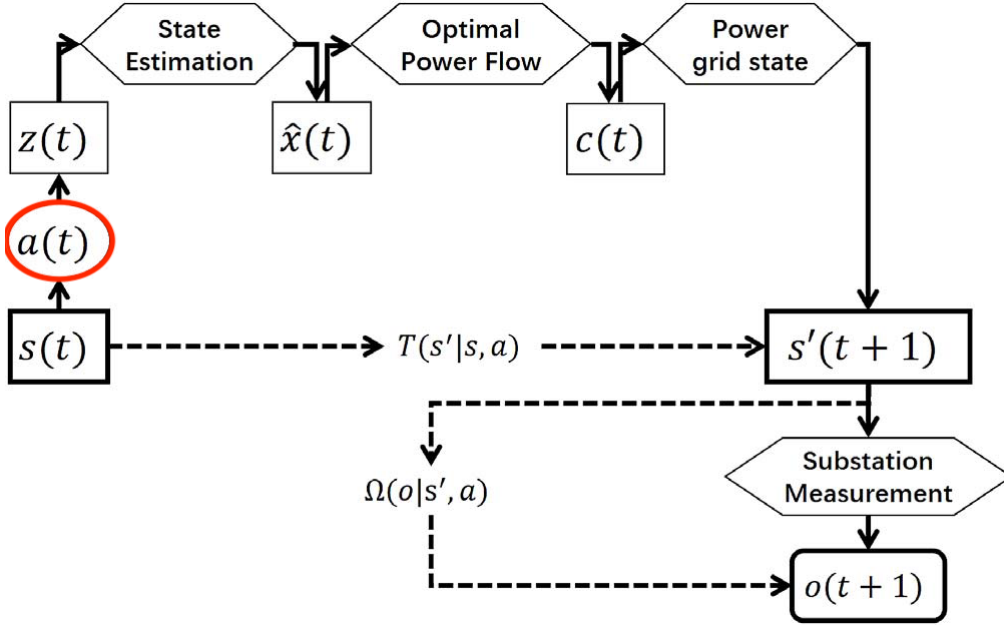


Figure 3.2: State transition of the POMDP during FDI attack.

- $\mathbf{z}(t)$: Tampered state of the power system, sent to dispatch center.
- $\mathbf{x}(t)$: Estimated system state; input for optimal power flow module.
- $\mathbf{c}(t)$: Regulation command.
- After $c(t)$ is executed the system will change to $s'(t+1)$.
- $\mathbf{o}(t+1)$: Attacker observation of the next state.

3.3 A Game-Theoretic Approach to Cross-Layer Security Decision-Making in Industrial Cyber-Physical Systems [3]

In [3], a security decision-making game model is proposed with stochastic elements to characterize the interaction between attackers and defenders in Industrial Cyber-Physical Systems (ICPS) to generate optimal defense strategies

to minimize system losses. Their approach was tested using case studies on hardware in the loop simulation test-bed. The distinction of their approach is that it was based on quantitative vulnerability analysis and time-based unified payoff quantification. In their work, they modeled a stochastic security game. The defender is equipped with an IDS which can detect attacks in real-time, and the attacker can probe the system to identify the defender's action.

The attacker can adopt all the possible atomic attacks, and also can choose a non-operation action. On the other hand, the defender will take all the known security countermeasures, in addition to the non-operation action. For the payoffs, the payoff of the attacker equals the benefit minus the cost. Also, the opposite of this is applied to the payoff of the defender, knowing that the game is considered a zero-sum game. For the strategy profiles, since it is a stochastic game, this means it is a mixed-strategy game. However, in a game problem, each player will try to maximize the total payoff by finding the optimal strategy, and this is called the Nash Equilibrium; where no player will deviate from a specific strategy profile. To reach this optimality, they have used an algorithm from other literature to optimize the strategy profile of both players. However, this algorithm is only applicable in case all the model parameters are known. For that reason, they referred to the Q-learning algorithm to approximate the optimal strategies by continually updating a Q-function despite not knowing the model parameters. They used Common Vulnerabilities & Exposures (CVE) database, to retrieve exploitability of real vulnerabilities and use them in the game model, also they introduced a transition graph that shows the possible transitions from a start point until reaching field devices and manipulates them.

To prove their approaches; they conducted two experiment one using an algorithm by assuming that all the model parameters are known in advance, and the

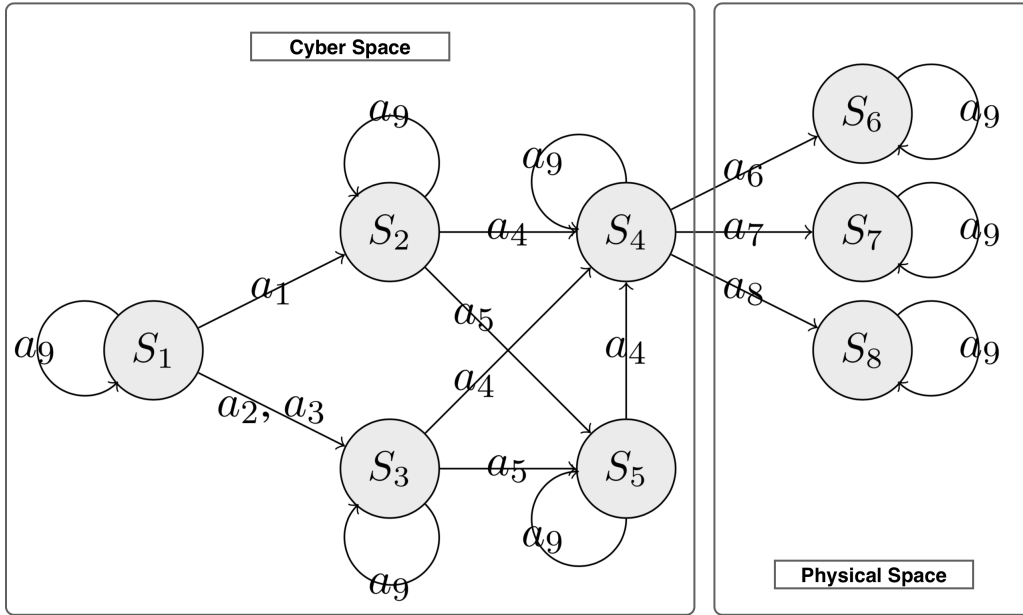


Figure 3.3: Attacker game state transition graph.

second one using the Q-learning algorithm to acquire the optimal attack/defense strategy profile. What is attractive to the work in this thesis is related mostly to the second experiment. They managed using the proposed Q-learning algorithm to obtain the optimal strategy profile. In their future work, they proposed; the introduction of unknown vulnerabilities into the system in addition to incomplete information of other player's actions.

Below, in Fig. 3.3 we present the attacker game state transition graph in [2], with its key information:

- **Attacker**, can exploit vulnerabilities or a apply a no-operation action.
- **Defender**, can patch vulnerabilities and apply a no-operation action.
- **Vulnerabilities**, are gathered from public databases.
- **Attacker game state**, take user/root privileges in cyber space and manipulate equipment in physical space.

3.4 A Multistage Game in Smart Grid Security: A Reinforcement Learning Solution [4]

Most smart grid research investigates different attack techniques while negating defense strategies. Game theory methods are applied for attacker-defender games in the smart grid. However, existing works only adopt a one-shot game and do not consider the dynamic process of the electric power grid. In this work [4], they propose a multistage game between the attacker and the defender using RL to identify the optimal attack sequences. The proposed reinforcement learning solution can identify optimal attack sequences that help the defender to enhance the security of the system. A comparison will be given between a one-shot game and a multistage game, to show the significance of the multistage attack. The main contribution in the mentioned work is: (1) they consider the defender's action and finds the attacker's optimal action sequences, (2) the learned optimal attacks actions will then be used as protection set for the defender, (3) the proposed multistage attack shows better performance compared to the one-shot attack in term of transmission lines outage. The proposed reinforcement solution can take advantage of generation loss and line outages. The game was modeled as a Markov decision zero-sum game, with two players an attacker and a defender that makes the actions, in turn, each will have a specific policy to follow and rewards to gain at each game step. In this game, both of the players have complete information about each other's payoff function. The two players repeatedly play the games according to their knowledge until the end. The objective of the attacker is to maximize the discounted sum of the future rewards, while the objective of the defender is to minimize it. In a multistage game, NE can be obtained by the repetitive decision-making process. Q-learning was used

as the best solution to find the optimal player's policies (NE). They presented the two players' zero-sum game between the attacker and the defender on the W&W 6-bus system and the IEEE 39-bus system. The obtained results showed the critical lines that are prone to attacks, and that also causes severe cascading failures. From this learned information, the critical lines can be included in the defender's protection set. The more critical lines included in the protection set, the fewer optimal policies will be available for the attacker to reach with the minimum number of actions. These results gained from simulated attacks provide useful information for power system operators to manage system planning and help the defender to prepare defense strategies better. As future work, they proposed the use of deep learning to identify vulnerable branches better.

Chapter 4

Methodology

In this chapter, two approaches will be introduced and discussed: the first is based on human decisions (pen-testing), and the second is AI equipped (automatic virus spreading).

First, inspired by Haruspex [27] an approach will be proposed below based on game theory and a model of three layers that simulates a cyber battle between a pen-tester and a network administrator on an CPS network. This approach is tested and justified with a developed tool [10] and detailed results.

Second, inspired by the infamous Stuxnet attack and from related works [1, 2, 3, 4] discussed previously, a more comprehensive and specific approach will be proposed with a learning element. This approach is based on ideas from both game theory and MARL; it is based on two levels: a strategic level modeled as an extensive-form game with stochastic elements to simulates adversaries behaviors and a battlefield level to conduct simulation for malware spreading in both the cyberspace and the physical space of a CPS. With the Q-learning algorithm applied on the battlefield level, optimal attack steps are derived. On the other hand, optimized policies for selected strategies are derived to impede the spreading of

a virus in the CPS network.

4.1 A Game Theoretical Model using Monte Carlo Simulations

This approach is modeled using game theory, a formal tool used here for strategic behavior analysis between two adversarial battling on an CPS network. One to penetrate systems, access and compromise industrial assets, and disrupt the normal function of industrial control units. And the other to monitor, control, and defend CPS assets and networks. This approach is composed of three layers: the functional network layer, the security layer, and the game theory layer. The network layer models a typical industrial company network that represents network devices, including a demilitarized zone (DMZ), enterprise network, internal network, and a SCADA network that is controlling PLCs and RTUs in the network field. The security layer represents the different security states of the network. This Security Layer acts as a platform for a cyber battle between the network administrator (defender) and the pen-tester (attacker). The actions, utilities, and strategies of this battle are modeled in the top game layer. Many games were simulated using a Monte Carlo method. A Python tool is developed to represent and configure the different layers and probabilistic parameters and to run the simulations [10]. Note that we use interchangeably the terms “pen-tester” and “attacker”. In reality, they may imply different settings: A pen-tester is usually hired to perform tests against the system and report findings. In this sense, it can be more closely monitored, and can be granted some entry access points. In contrast, much less information is assumed about a real attacker who can be an insider or outsider.

4.1.1 Model Advantages

This approach has the following advantages:

- It builds on top of previous research on the topic and after a detailed and thorough study of related work. Lessons learned from previous risk assessment methodologies, and models such as attack and vulnerability trees, Monte Carlo simulations, control theory, and game theory are incorporated in this model and segregated into three layers: network, security, and game.
- It bypasses computational difficulties by using Monte Carlo simulations since the space of possible strategies is infinite. Previous works were limited to simple and abstract models due to these difficulties. However, attacks against CPS are getting much more sophisticated, which requires more extensive and randomized modeling.
- It ensures flexibility for the security administrator to customize the system parameters.
- It integrates both proactive and reactive security mechanisms, and budget spending modeling to put both the defender and the attacker under realistic constraints.

4.1.2 Network Layer

An example of the network layer is depicted in Figure 4.1. The network has an Internet Firewall and a DMZ firewall. The network is segmented into four areas or sub-networks:

- The internet DMZ exposes the organization's external services, such as web services.

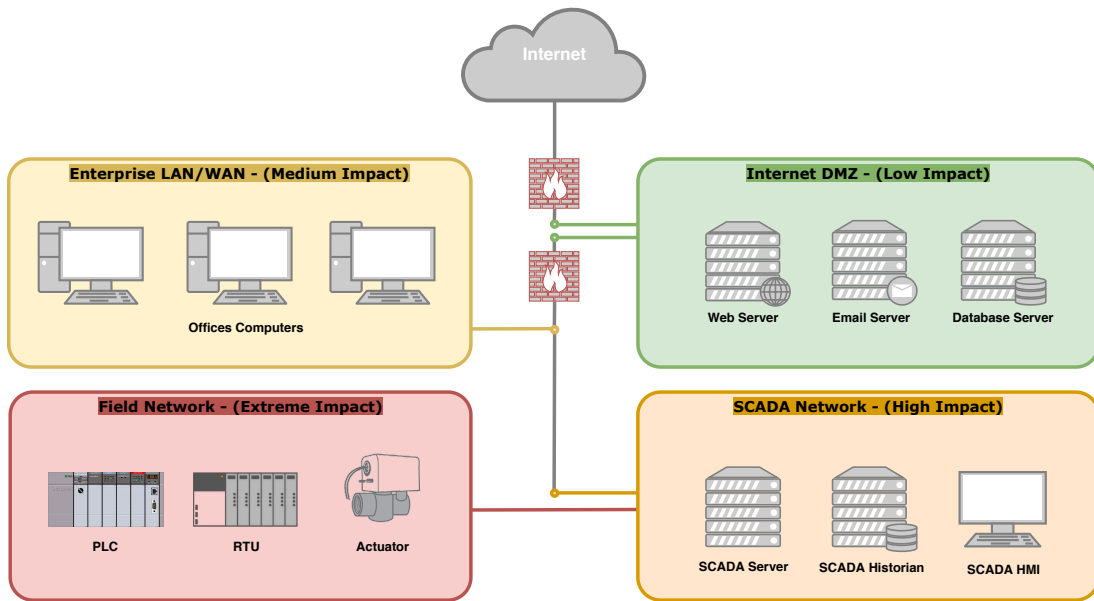


Figure 4.1: Example of a functional network layer

- The enterprise local and wide area networks (LAN/WAN) hosting users end-devices and internally shared services such as printers.
- The SCADA network is hosting the SCADA server, HMI, and historian. The SCADA server is responsible for the data acquisition and the management of controllers. The SCADA HMI provides the interface to monitor and control the PLCs and RTUs. The Historian database is used to collect and store sensors data for analysis and monitoring.
- The multi-drop network is hosting RTUs, PLCs, and actuators.

4.1.3 Security Layer

The security layer is composed of the attack-vulnerability tree, which is available for the pen-tester and the strategic response model, which is available for the defender.

Attack-Vulnerability Tree

In Figure 4.2 each node in the attacker tree is a security state representing the current security status as viewed by the attacker. The security state node has a utility value quantifying the reward of reaching this state. Reward points are widely used in security certification exams and pentesting competitions. Some examples of reward points in our scenario are:

- Obtaining scanning information = 1 point
- Firewall Bypassing = 4 points
- Denial of Service, hijacking a TCP connection = 20 points
- Compromising a host, SQL injection, replay attack to bypass authentication = 30 points
- Root privilege escalation, exploit against the field network = 40 points

Edges between the states indicate the possibility of transferring from one state to another by performing a successful action. The attacker may traverse different paths to reach a target state with a high reward. Examples of target states are the ability to control RTUs and PLCs, the ability to provide wrong sensory measurements, or the ability to deny the availability of monitoring and logging the events in the field network. Vulnerabilities are collected from open source databases such as CVE Details [5] and using open source tools like MulVAL [40].

Response Strategic Model

The response strategic model is a three-phase process as shown in Figure 4.3. For more details we refer the reader to [41] and [42]. Of particular interest is the response phase that can be automated to a large extent using table-driven rules and flowcharts. The rule-based response system forwards unresolved alerts to the

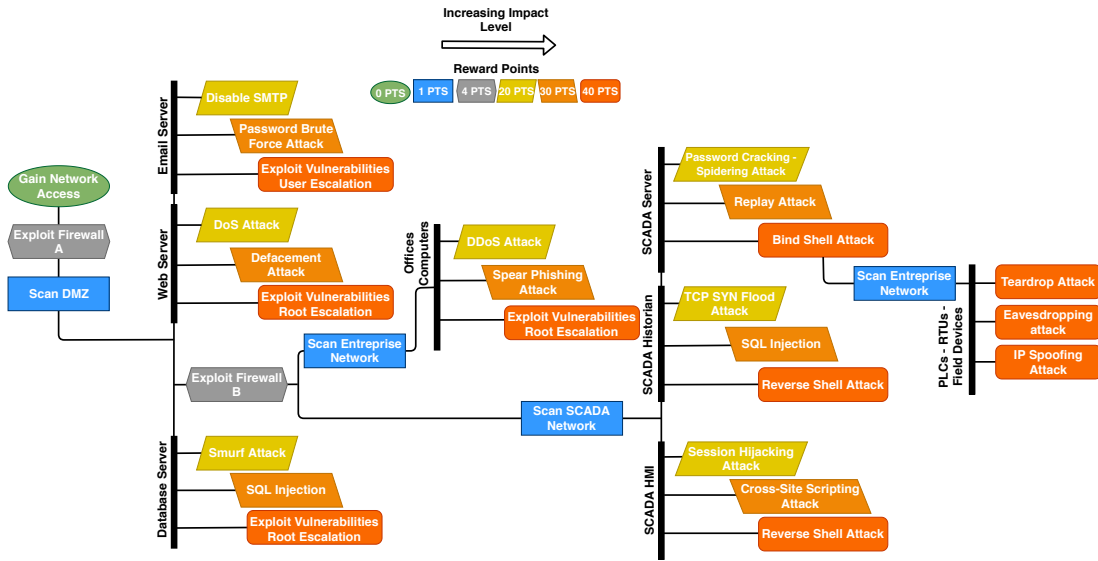


Figure 4.2: Example of an attack tree in the security layer

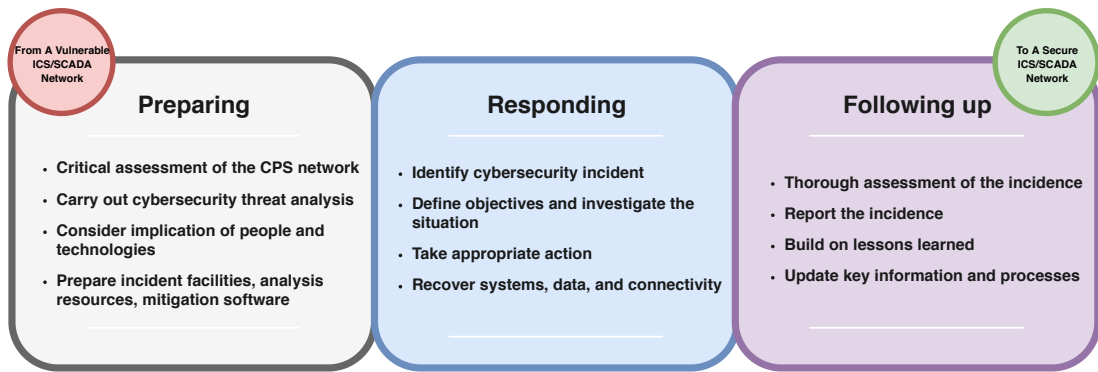


Figure 4.3: An example of a cybersecurity strategic response model

human analyst. For example, the response plan of a Modbus slave attack on an RTU is shown in Figure 4.4.

4.1.4 Game Layer

In the game layer, the defender and attacker player are modeled, rewards, and actions, as shown in Figure 4.5. The defender is equipped with an intrusion and anomaly detection system (IDS) at the security layer for monitoring the security state of the network layer. The IDS is responsible for sending security alerts to the defender. Each alert w has a confidence value $C(w)$, a severity

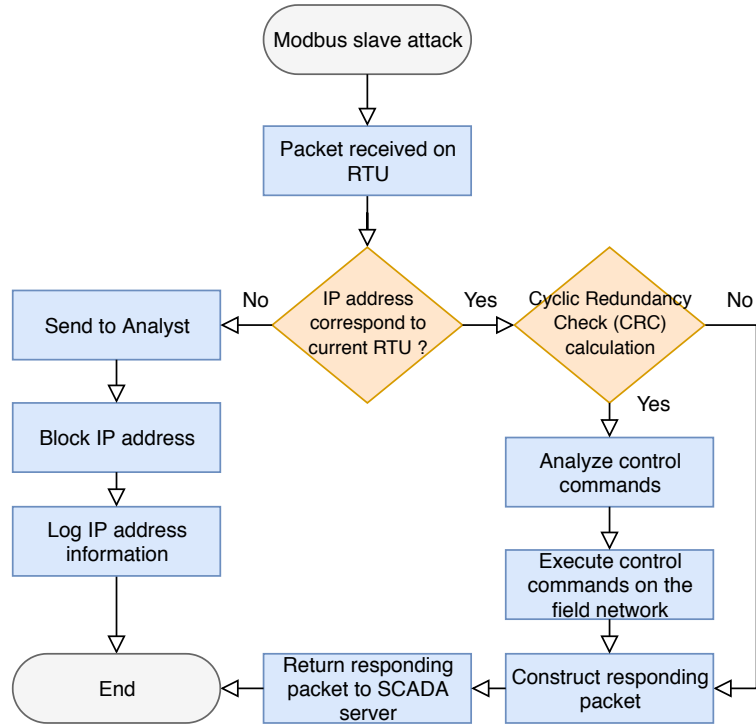


Figure 4.4: Modbus slave attack response module

$S(w)$ and costs $B(w)$. The defender pays the cost of the alert only in case the alert is considered and analyzed. Both the defender and the attacker have predefined limited budgets. The budget represents the resources of each party, such as money, time, and expertise. Upon considering an alert, the strategic response model is used to predict the current security state and the next step of the attacker. The defender has to take one of two actions:

Reactive

Act immediately to restore the previous security state. The taken response succeeds with a probability $P(d)$. The total success probability is: $C(w) * P(d)$. The total budget spent is $B(w) + B(d)$.

Proactive

Predict the next step of the attacker and act to prevent it. The taken

response succeeds with a probability $P(d') > P(d)$. The chance is higher since the defender is being proactive rather than reactive as in the first choice. However, the prediction is correct with a chance $P(\text{pred})$. The total success probability is: $C(w) * P(\text{pred}) * P(d')$. The total budget spent is $B(w) + B(d')$.

The attacker is either an outsider or an insider who is given a node in the DMZ. The attacker uses the attack-vulnerability tree from the security layer to choose a suitable path to compromise a target node. The attacker has predefined the total budget and look-ahead. The attacker strategy is based on the ratio of exploration versus exploitation. In exploration mode, the pen-tester chooses an attack step at random with a predefined probability of success. In exploitation mode, the attacker chooses the action maximizing the reward given the current state and the available look-ahead. Attackers and defenders can choose different budget spending strategies ranging from very conservative to very eager. The simulation performed runs an experiment with different pure and mixed strategies. Concretely the possible actions of the defender maybe by adding a firewall rule, throttling an external IP address or blocking it, shutting down a service or a node, or sometimes allowing some attack steps in order to collect more information about the attacker, such as in the case of a honeypot. Among the pen-tester action steps, enumerate reconnaissance steps, scanning and probing, exploiting a service, initiating remote shells or reverse shells, privilege escalation, password/hash cracking, flooding, or denial of service steps, social engineering, poisoned data injection, initiating malicious control commands. Each node in the security layer is assigned a reward or utility value and is concretely linked to the compromise of a functional node in the network layer. Target nodes have much higher rewards than intermediate nodes. This game is considered a zero-sum

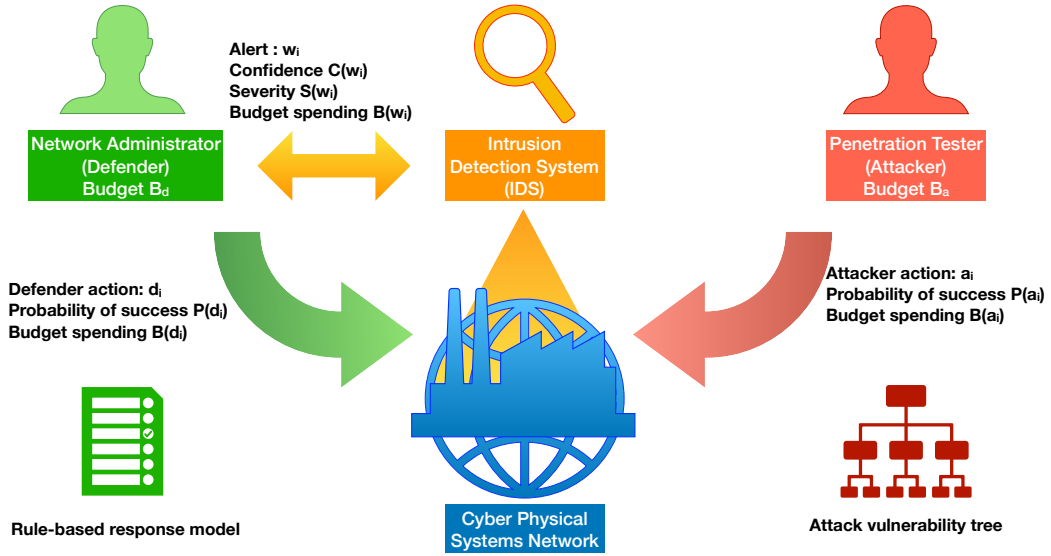


Figure 4.5: Example of a game in the game theory layer

game. The utility lost by the defender is awarded to the attacker and vice versa. The risk value is the utility of the game, and it is the final reward obtained by the attacker after that both parties have spent the total amount of their budgets. Both players try to maximize their profit by selecting the best strategies. However, no learning component is assumed in this approach.

4.1.5 Implementation Setup

A tool has been designed and developed using Python 3.7 to run game simulations [10]. It was developed using an Intel Core i7-7700 CPU running at 3.60 GHz and 32.0 GB of RAM. The tool takes for input three configuration files: (1) the network file contains the network topology and interconnections, (2) the security file contains the attack/vulnerability tree and the counterpart strategic response mode, and (3) the game file contains the player profiles and chosen strategies, the nodes rewards, the total budgets and success probability of actions. First, an input file was used to describe the architecture of the proposed network layer

with all the possible transitions between the network nodes. The data provided was stored in a two-dimensional python list to be accessed later during the simulation runs. Second, an input file was used to describe the attack/vulnerability tree in the security layer in addition to the severity level of each security node. The severity level was stored in a python list that will be used later in the simulation runs. Third, for each simulation scenario, a different input file was fed into the simulation tool describing the strategy profile of each agent, the number of game steps, reward points for each network node depending on its severity level. To implement Monte Carlo simulation, the random python module was used to describe stochastic elements in the proposed game model such as the probability of attack success, probability of detecting attacks using a simulated IDS, probability of defense success, probability of choosing between being active or proactive in the defense strategies. The randomness was not always uniform; it was also biased depending on the strategy of budget spending adopted by the agents. Finally, a vast number of simulation runs were conducted, and each game with its corresponding rewards was recorded into a two-dimensional python list to be later sampled and plotted to two different graphs that give a comprehensive overview of each simulated scenario. The first plot shows the frequency of compromising each subnet in the proposed network layer, and the second plot displays a curve showing how the network was resilient to the pen-tester attacks in terms of reward points.

4.1.6 Simulation Results

Below, the presentation of the five scenarios conducted using the python tool developed. Each simulation will be discussed apart with its corresponding settings configuration, such as policy profile and budget spending, obtained results that can be discussed on two plotted graphs for each scenario, in addition to a

conclusion describing the overall game scenario with its winner. The results are the output of 10^6 simulation runs of each scenario.

First scenario: Random strategies

The first scenario is a baseline to compare with other scenarios. In this scenario, both the attacker and the defender have random strategies in choosing attack and defense actions. The prior probability of the success of an attack step a_i is $P_{\text{success}}(a_i)$. The confidence value of the alerts is random and uniformly distributed. The posterior probability of success of an attack step if the defender took a reactive response is:

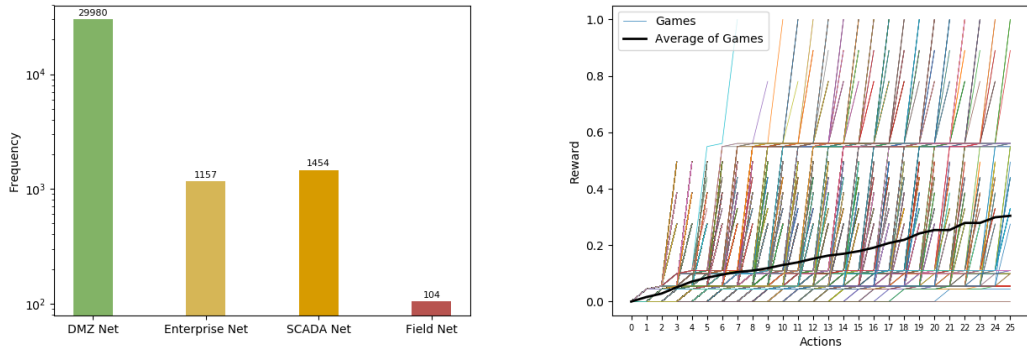
$$P_{\text{success}}(a_i|\text{reactive}) = P_{\text{success}}(a_i) * (1 - C(w_i) * P(d_i))$$

The posterior probability of success of an attack step if the defender took a proactive response:

$$P_{\text{success}}(a_i|\text{proactive}) = P_{\text{success}}(a_i) * (1 - C(w_i)P(\text{pred})P(d'_i))$$

Let: $P(d'_i) = 70\%$ and $P(d_i) = 50\%$. The choice proactive/reactive is random (50%). We take $P(\text{pred}) = 0.8$ which means that being proactive gives us a better chance to defend ($0.7 * 0.8 = 0.56$) than being reactive (0.5). Based on the targeted nature and the attack impact, the attacker reward can be 10, 20, 30, or 40 points. The simulation stops once a target node (i.e., a leaf node) is compromised. Future rewards are less important than present rewards. we use a discount factor γ to take this fact into account. The number of all possible points is used to normalize the reward:

$$R_{\text{attacker}} = \frac{\sum_i \gamma^i * R(a_i) * P_{\text{success}}(a_i|\text{defender})}{\sum_i \gamma^i R(a_i)}$$



(a) Frequency of compromising critical targets in subnets with different impact levels (b) Cumulative reward of the attacker per time step (simulations and average)

Figure 4.6: Random strategy for defender vs. Random strategy for attacker

where:

$$P_{\text{success}}(a_i|\text{defender}) = P(\text{reactive})P_{\text{success}}(a_i|\text{reactive}) \\ + P(\text{proactive})P_{\text{success}}(a_i|\text{proactive})$$

The defender reward is $R_{\text{defender}} = 1 - R_{\text{attacker}}$ since it is a zero-sum game. Figure 4.6a shows the frequency of compromising nodes in subnets with different impact levels. Results show that compromising internal subnets is more challenging than targeting the DMZ nodes. Figure 4.6b shows the cumulative reward curves for 10,000 sampled simulations as well as their average. Discrete-time steps shown at the x-axis with the assumption that the attacker is taking a new action at each time step. The average curve is an indication of the resilience of the network to a random attack. This curve shows a small slope and ends by having less than 40% of the total game rewards. The defender has won this game. Even though the defender has a random strategy, the risk is low in this scenario.

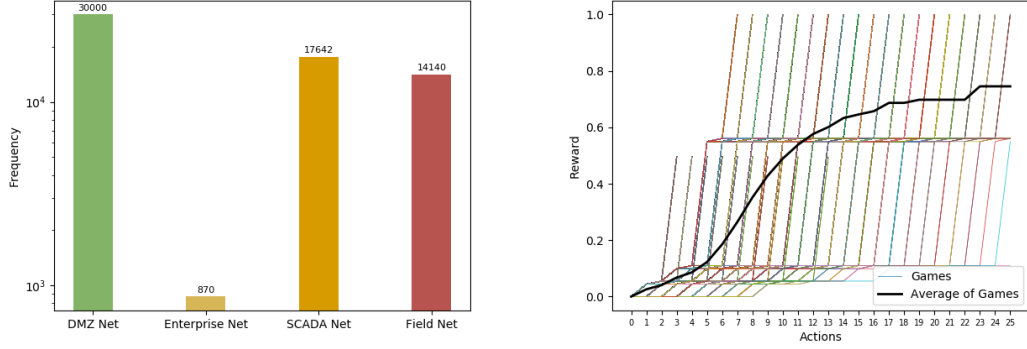
Second scenario: Exploration attack

In this scenario, the attacker is given a look-ahead of 2 and an exploitation-exploration strategy. The attacker will compute the reward of the first two steps for each path. The exploitation probability is set to 15%. In the case of exploitation, the attacker randomly chooses one of the paths with max rewards. The exploration probability is set to 85%. In the case of exploration, the attacker randomly chooses between the paths with less than max rewards. Interestingly, the exploration leads to long paths with higher accumulated rewards since it will lead to critical control targets.

The success percentage of an attack step is 70%, with the introduction of budget spending $B(a_i)$ for the attacker. The simulation stops when the attacker runs out of budget or reaches a leaf node. The confidence of the alerts and the probability of successful prevention are uniformly distributed. Hence, the attacker has more edge by using the look-ahead with exploration tendency.

Figure 4.7a shows the frequency of compromising nodes in subnets with different impact levels. Note that the attacker is almost ignoring the enterprise subnet since it tends to explore paths with less apparent rewards based on the look-ahead. Since the defender strategy is random, the attacker succeeds in reaching the critical subnets.

Figure 4.7b shows the cumulative reward curves for 10,000 sampled simulations as well as their average. The x-axis has discrete time steps, and the attacker is taking a new action at each time step. The average curve is an indication of the resilience of the network to a random attack. This curve shows a relatively large slope and ends by having more than 70% of the total game rewards. Here the attacker has won this game; the risk is high in this scenario.



(a) Frequency of compromising critical targets in subnets with different impact levels (b) Cumulative reward of the attacker per time step (simulations and average)

Figure 4.7: Random strategy for defender vs. Look-ahead = 2 for attacker

Third scenario: Greedy defender

In this scenario, both defender and attacker are allowed to increase the chances of success $P_{success}(a_i)$ and $P_{success}(d_i)$ by paying more budget. The following formulas govern this process:

$$\begin{aligned}
 P(a_i|b_i) &= P(a_i) + (1 - P(a_i)) * \frac{b_i}{K(B)} \\
 P(d_i|b_i) &= P(d_i) + (1 - P(d_i)) * \frac{b_i}{K(B)} \\
 C(w_i|b_i) &= C(w_i) + (1 - C(w_i)) * \frac{b_i}{K(B)} \\
 P(\text{pred}_i|b_i) &= P(\text{pred}_i) + (1 - P(\text{pred}_i)) * \frac{b_i}{K(B)}
 \end{aligned}$$

Where $K(B)$ is the maximum allowed budget to spend per action and is a function of the total budget of B . Therefore a party can reach 100% success for action in case it spends the amount of $K(B)$. A total budget of $B = \alpha * K(B)$ allows α actions to succeed with certainty. In this simulation, let $\alpha = 4$. For the attacker,

the budget can be the time and effort for gathering information about the target before unleashing the action, the time to discover a new vulnerability or adapt an existing one, or the cost to buy a suitable vulnerability from the dark web. The defender can spend the budget to increase the chance of a successful prevention step, or to enhance the quality of the alerts of the IDS. In practice, this is the cost or the time of an expert (human or not) to analyze the alert before handing it to the defender, or the effort to analyze and trace the security incidents.

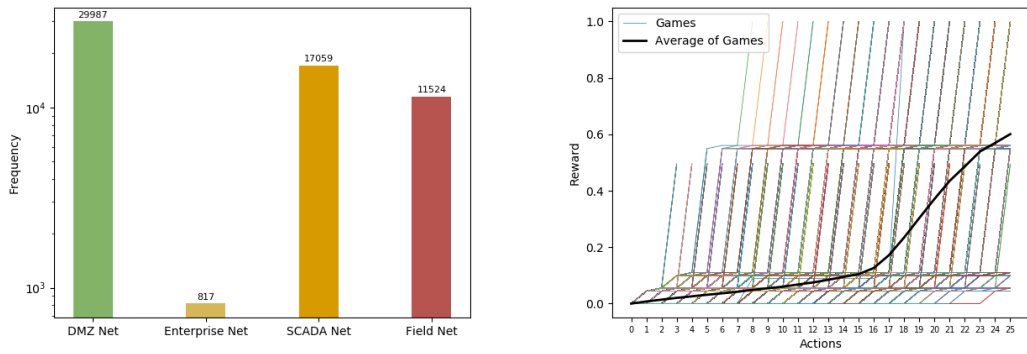
In this scenario, the defender is proactive and greedy. The defender is proactive 80% of the time and reactive only 20% of the time. The defender spends the maximum allowed budget per action, alert confidence, and next step prediction. The attacker strategy is the same as in the second scenario. No extra budget is used to raise the success probability of the attacker's actions.

Figure 4.8a shows the frequency of compromising nodes in subnets with different impact levels. Figure 4.8b shows the cumulative reward curves for 10,000 sampled simulations as well as their average. The curve escalates slowly in the beginning, due to the high quality of the defender reaction. Suddenly, the defender runs out of budget, and the attacker takes over the game to finish with 60% of the cumulative reward. The risk is high in this scenario.

Fourth scenario: Conservative defender

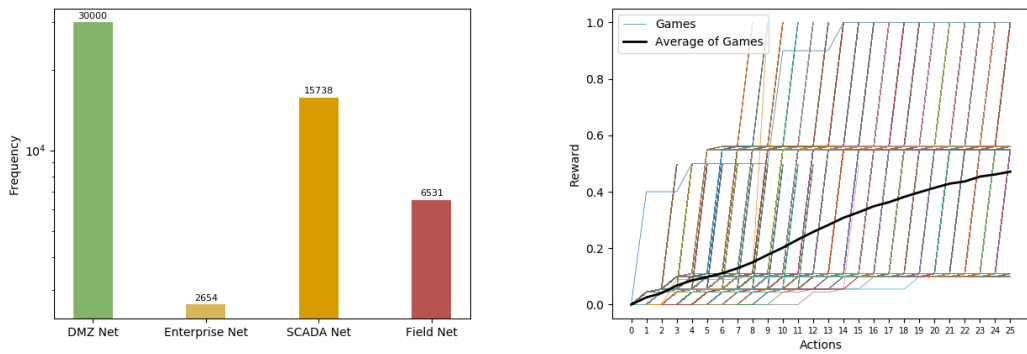
The fourth scenario has the same settings as for the third scenario. However, the defender uses a strategy of conservative budget spending. In the early stages of the game, the attacker has more potential in conducting successful attacks. The defender starts investing in the quality of alerts when the attacker reaches targets of impact level 3 or 4.

Figure 4.9a shows the frequency of compromising nodes in subnets with different impact levels. A relative drop is noticed in the frequency of successful attacks in



(a) Frequency of compromising critical targets in subnets with different impact levels (b) Cumulative reward of the attacker per time step (simulations and average)

Figure 4.8: Greedy budget spending for defender



(a) Frequency of compromising critical targets in subnets with different impact levels (b) Cumulative reward of the attacker per time step (simulations and average)

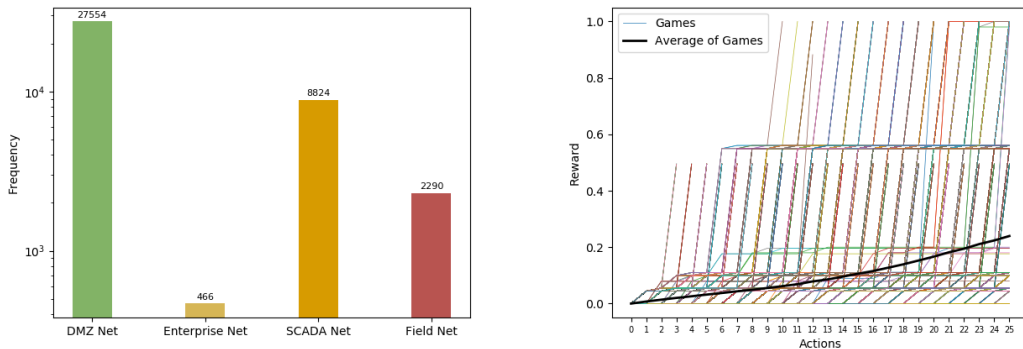
Figure 4.9: Conservative budget spending for defender

the field Network (Impact level 4).

Figure 4.9b shows the cumulative reward curves for 10,000 sampled simulations and their average. The average curve reaches 40% of the total game rewards.

Fifth scenario: Uniform defender

In the fifth scenario, the defender's strategy is to spend the budget along the attack time uniformly. The defender starts from the early beginning of the game in investing small amounts in the confidence of the IDS alerts, the probability



(a) Frequency of compromising critical targets in subnets with different impact levels (b) Cumulative reward of the attacker per time step (simulations and average)

Figure 4.10: Uniform budget spending for defender

of success for defensive actions, and prediction accuracy while using proactive defense.

Figure 4.10a shows the frequency of compromising nodes in subnets with different impact levels. As notice an even smaller probability of targeting nodes at levels three and four.

Figure 4.10b shows the cumulative reward curves for 10,000 sampled simulations and their average. An impressive performance is observed since the attacker rewards curve barely exceeds the 20% of the total game rewards. This scenario shows that a balanced defense is even better than a conservative one. No attacks on the surface nodes should be tolerated while focusing all the efforts on protecting the last line of nodes.

4.1.7 Closing Notes

This approach proposed new modeling of a typical industrial network using three layers: network, security, and game. Risk is evaluated in terms of the Nash equilibrium of a zero-sum game in between a pen-tester equipped with an attack tree and a defender equipped with an IDS and a strategic response graph model.

The concept of Nash Equilibrium is that no player would like to deviate from the current state. Each agent has the best possible response given the portfolio of played strategies. This situation is ideal from a security point of view since attacks can never be stopped. However, best responses need to be known given limited time and budget. A python Monte Carlo simulation tool [10] has been developed to evaluate the payoffs given different variants of randomness, selected strategies, budget spending, and look-ahead. Simulation runs have experimented with a combination of pure and mixed strategies for each player. Preliminary results show the efficiency of this approach method in selecting the best strategy for mitigating attacks against CPS. One of the interesting results is that the uniform budget spending strategy is more efficient for the defender than both conservative and aggressive budget spending ones. Therefore the defender must not rush to block the early steps in the pen-testing and leave the internal nodes with no protection or be tolerant in under-protecting the surface nodes. In this approach, a very small subspace of the infinite space of all possible strategies is explored. In future work we want to line the simulator to a game theory engine to decide on the parameters to reach a NE.

4.2 A Hybrid Game Theory and Reinforcement Learning Approach for Cyber-Physical Systems Security

This approach proposes to frame CPS security in two different levels, strategic and battlefield, by meeting ideas from game theory and Multi-Agent Reinforcement Learning (MARL). The strategic level is modeled as an imperfect information, extensive form game. Here, the human administrator and the virus author

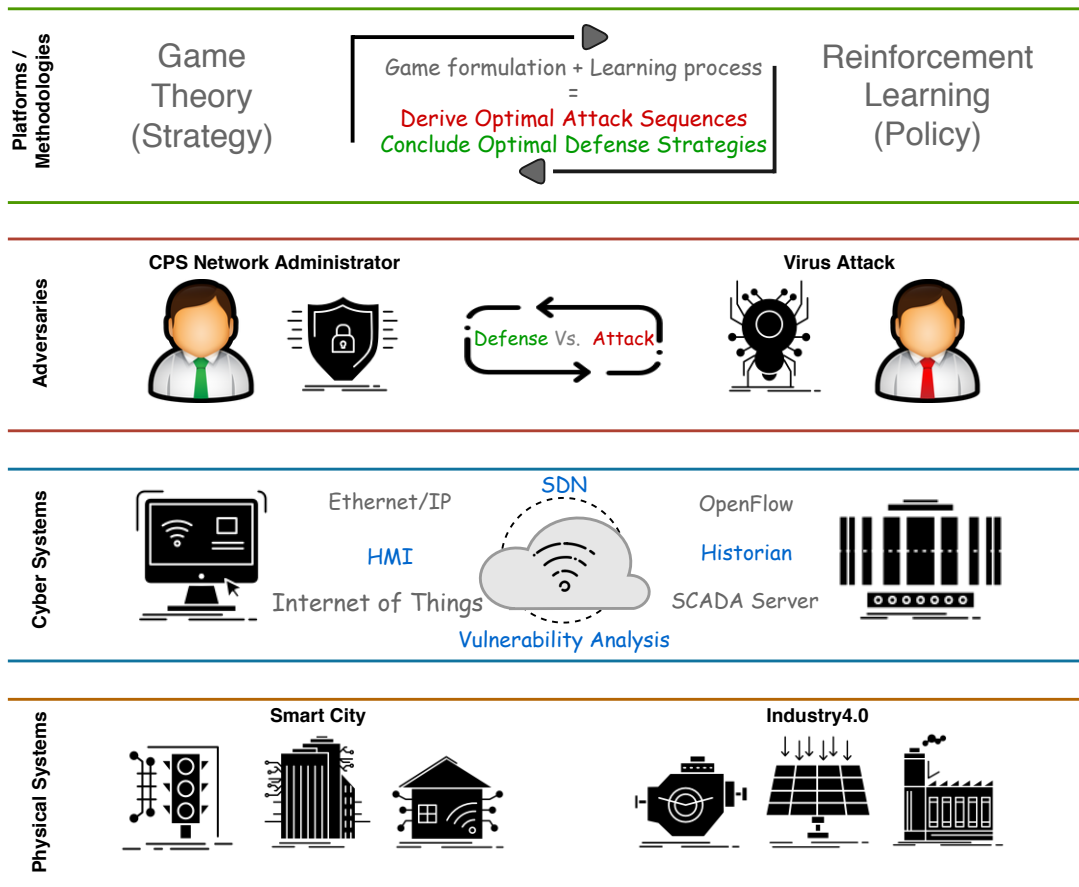


Figure 4.11: Our approach in a nutshell: cyber-physical systems, human agents and machine agents, proposed modeling solution.

decide on the strategies of defense and attack, respectively. At the battlefield level, strategies are implemented by machine learning agents that derive optimal policies for run time decisions. The outcomes of these policies manifest as the utility at the higher level, where we aim to reach a Nash Equilibrium (NE) in favor of the defender. We simulate the scenario of a virus spreading in the context of a CPS network. We present experiments using the MiniCPS simulator and the OpenAI Gym toolkit, and discuss the results in the next chapter.

4.2.1 Introduction

We propose in this approach a hybrid human-machine approach to model the security game in between adversaries in the CPS ecosystem, as depicted in Fig. 4.11. At the bottom, the physical systems layer gives a general idea about the application of CPS and SCADA systems, including smart cities, industry 4.0, and Critical Infrastructures (CI). Second, the cyber systems layer handles the operations related to monitoring and controlling the physical systems. The third layer is for adversaries: the CPS network administrator trying to defend the cyber systems and reduce potential threats, and the rogue attackers trying to breach the cyber systems to cause physical damage. On top comes the methodology layer. We propose a hybrid approach based on game theory as a tool to formalize the game interactions between the human adversaries who decide on strategies. For instance, how to deal with security situations with different intensity levels. From defender's point of view, which defense components are deployed, how firewalls are configured, how system patching is managed, etc. From attacker's point of view, which infiltration techniques are used, how the malware is packaged, etc. This game has an extended form since the security status may change from low danger estimation to more serious states, and most attacks are multi-stage. It is also a game with imperfect information, since most of the time, both the defender and the attacker may not know the exact security state at the current time. Based on the formulated game and decided strategies, MARL-powered machine learning agents may decide on the best actions (proactive or reactive) to take in the battlefield. The states in the RL model are different since they are run-time and perfect information states. Based on the knowledge of the network and the alerts of the defense systems, the defender chooses actions, and navigates its own state space. The attacker navigates a parallel state space based on an

attack tree. Therefore, the MARL agents learn the best policy for an automated response at run-time. These best policies for both the attacker and the defender manifest themselves as the utilities at the strategic game level. Therefore, we can compute a Nash Equilibrium (NE) at the strategic level that represents the best possible strategy in the presence of the adversary. A NE is a point that no party would like to deviate from without losing utility. We focus in particular on a virus spreading scenario, with realistic assumptions such as discovering and using zero-day vulnerabilities. We run simulations using the MiniCPS simulator, and we use OpenAI Gym for the MARL implementation.

Additional novelties of the proposed model are as follows:

- Simulate a virus attack in a modern CPS network composed of four subnets.
- The use of predefined known and discovered vulnerabilities, in addition to the chance of discovering zero-day vulnerabilities.
- Derive optimal attack sequences and defense policies using MARL and Q-learning.
- Create a simulation framework [10] composed of a network simulator and an RL toolkit.

4.2.2 Proposed Model

The proposed hybrid approach based on game theory and MARL; is divided into two levels first a higher game strategic level and second a lower battlefield level.

4.2.3 Strategic level

We model the strategic level using an imperfect information extensive form game. The states of this game are the overall security status: Low, medium, high

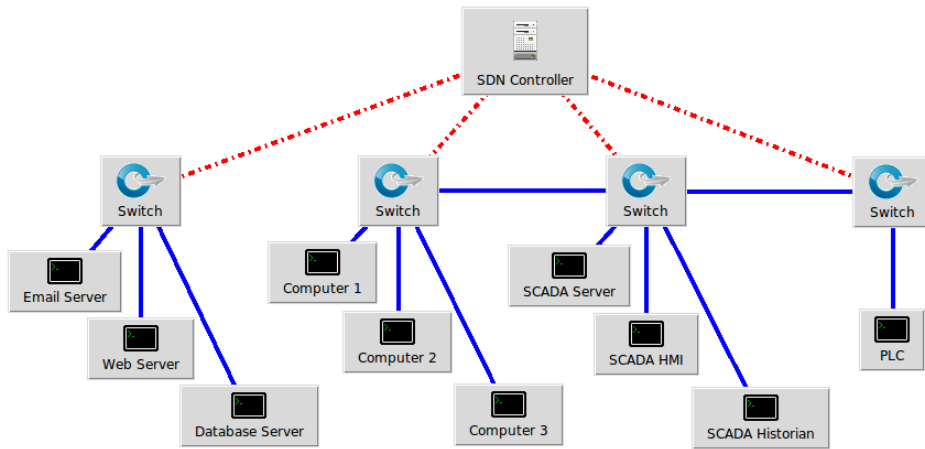


Figure 4.12: Proposed CPS network topology using MiniCps.

and critical danger. Obviously, the defender does not know for sure what is the actual state (e.g. the attacker has discovered a zero-day vulnerability). The attacker does not know what is the state since he does not have the full information about the target network or about all the defender countermeasures. Therefore, we consider an imperfect information model. This game has an extended form since the defender will try to recover from bad states to the original (low) state. The attacker will try to reach the critical state. Therefore the game have multiple stages. The defender chooses the strategy in terms of countermeasures to impede the progress of the attacker. The strategy is translated as a set of actions to choose from at the Battlefield level. The attacker chooses the strategy in terms of attack methodologies. This strategy is translated as as set of tools, vulnerabilities and penetration actions at the Battlefield level.

4.2.4 Battlefield level

This level is modeled using MARL and represents a multistage stochastic game with a learning component; each agent has a game turn with a random transition

probability and creates a new scenario for the other agent to act accordingly. The state of the game is composed of the state of the defender based on the alerts, and the state of the attacker based on its attack tree. The nature of this game is stochastic, since some attacks can fail with some probability, and some countermeasures can fail with another probability.

Network Architecture

The game model is based on a modern and realistic CPS network. As shown in Fig. 4.12, the system is segregated into four subnets: First, a cloud-based network hosting the web server, email server, and the database server of the CPS organization. Second, an enterprise network that hosts all the office's computers for the CPS enterprise. Third, a SCADA network that host the SCADA Server responsible for data acquisition, the Human Machine Interface (HMI) used by operators to monitor sensor values and take commands accordingly, and the SCADA historian where acquired data are stored. Finally, The field network contains the goal of the attacker which is the control units represented by a Programmable Logic Controller (PLC) that execute commands sent from the SCADA network to be applied on the physical processes and also retrieve sensor values from sensors deployed all over the industrial process line to be afterward sent to the SCADA network for processing and storing. Each host deployed on the different subnets of the CPS network is explicitly assigned a unique private directory acting as its storage unit. These directories will be used as a ground for virus spreading among the hosts, the four subnets, and the two layers of the CPS network. The four subnets are connected using OpenFlow switches, that are in their turn connected to a Software Defined Network (SDN) controller on the cloud. SDN has been applied to different kinds of networks, with its capability to separate the control plane from the data plane SDN can also be used to support the network com-

CVE ID	Type of Vulnerability	Vulnerability Access	Exploitability Score	CVSS Score
CVE-2019-11133	DoS	Local	2 (Low)	4.6
CVE-2011-5163	Overflow	Local	1 (Low)	4.6
CVE-2018-1168	Code Execution	Local	3 (Low)	7.2
CVE-2016-2281	Gain Privileges	Local	7 (High)	6.0
CVE-2016-9360	Gain Information	Local	6 (Medium)	4.4
CVE-2015-3940	Directory Traversal	Local	4 (Medium)	6.9
CVE-2013-0654	DoS	Remote	4 (Medium)	9.3
CVE-2011-1918	Overflow	Remote	2 (Low)	10.0
CVE-2015-7908	Code Execution	Remote	5 (Medium)	9.3
CVE-2019-6564	Gain Privileges	Remote	4 (Medium)	9.3
CVE-2018-11517	Gain Information	Remote	2 (Low)	5.0
CVE-2014-0751	Directory Traversal	Remote	2 (Low)	7.5

Table 4.1: Sampled vulnerability analysis gathered from CVE details [5].

munication in CPS networks [43]. Also, because CPS networks rely heavily on communication networks for control, SDN can be used to manage these networks.

The benefits of using SDN in a CPS network are many; it can dynamically reroute paths for CPS control commands, provide fast failure detection between the layers of the CPS network, improve the overall security of the network, and finally evolve the network to support new technologies, services, and needs [43]. To correctly mimic a modern CPS network with its two different layers and its subnets, MiniCps [44] is used as a network simulator to target network communication, control logic, and physical layer interaction. The network topology of the CPS network is presented in Fig. 4.12. Using MiniCps features, a terminal can be launched at each host to execute a specific network command. In this thesis work, we implement the virus spreading among the hosts, Netcat command was used to read and write files using a TCP connection, which allows the virus to pass from a directory to another until reaching its final destination.

Vulnerability Analysis

The CPS network that is designed and implemented in this thesis work is a vulnerable CPS network segregated into four subnets. The cloud-based subnet is hosted on the cloud by an outside organization such as Amazon Web Services (AWS) [45], which is a modern practice adopted by all big firms these days. Also,

the rest of the subnets are hosted locally in the CPS organization. The base metric provided by CVSS [46] is adopted to quantify vulnerabilities on network nodes. More specifically, the exploitability score describes best the complexity of exploiting a vulnerability. Since the first subnet is cloud-based, it is characterized by a high level of security with a secure IDS integration. However, the rest of the subnets hosted locally in the CPS organization are vulnerable with an uncertain IDS integration.

Table 4.1, shows a small sample of the used vulnerabilities related to the different CPS subnets, they are collected from CVE details [5] an online database that gathers vulnerabilities with their corresponding information such as attack types, access types, exploitability scores, and CVSS scores. We gathered in total 65 vulnerabilities for the network devices. Each network device assigned with a set of five vulnerabilities. These vulnerabilities, depending on their exploitability scores, determine the transition probabilities for the attacker to bypass the network defenses, and reach the end goal. The defender can invest in patching these vulnerabilities to strengthen the system, with rewards proportional to the exploitability scores. The proposed game also gives the chance to discover zero-day vulnerabilities that are unknown to the CPS network administrator until discovered and exploited by the virus attack. This intended feature provides the model with a sense of real-life virus targeted attacks. The defender must be aware that undiscovered vulnerabilities may exist.

Game Description

The setting of this game is a vulnerable CPS network based on a dynamic environment with two adversaries and a probability of transition. The attacker represented as a malicious virus spreading into the system and targeting the control units hosted on the field network, and the defender described as the CPS

network administrator defending the CPS network. The game is played in a multistage manner, where each agent applies an action on the environment and receive a reward, and at the same time creates a new situation for the environment to be faced by the adverse, and this makes the environment dynamic. Fig. 4.13 shows the proposed game model for virus spreading in a multi-subnet and cross-layer CPS network. The CPS network is formed of two main layers. The first layer that represents the cyberspace layer contains three subnets: the cloud-based system, which has a low impact on the overall CPS security. The enterprise network, with a medium effect on the security of the CPS network. Finally, the SCADA network in this layer is considered of high-security impact due to its nature of monitoring and controlling control units and edge devices hosted on the field network [47].

The second layer represents the physical space layer containing both the field network that is stamped with an extreme security impact on the overall CPS operations; due to the presence of the attacker goal, in addition to the physical industrial process that depends on the functionality nature of the CPS organization. The purpose of the attacker resides only in the physical layer and more precisely on the field network. Also Fig. 4.13 shows all the possible transition paths of the virus on the CPS network. The spread of the virus is done by exploiting vulnerabilities on network devices and switches having firewalls capabilities until reaching the end goal. Each node of the graph is associated with a set of vulnerabilities, a vector of defense values containing the exploitability scores of the corresponding vulnerabilities. The defender is assumed to have complete information about the architecture of the CPS network.

On the other hand, the attacker has no information about the CPS network; information is collected in the process of attacks only. At each game step, the

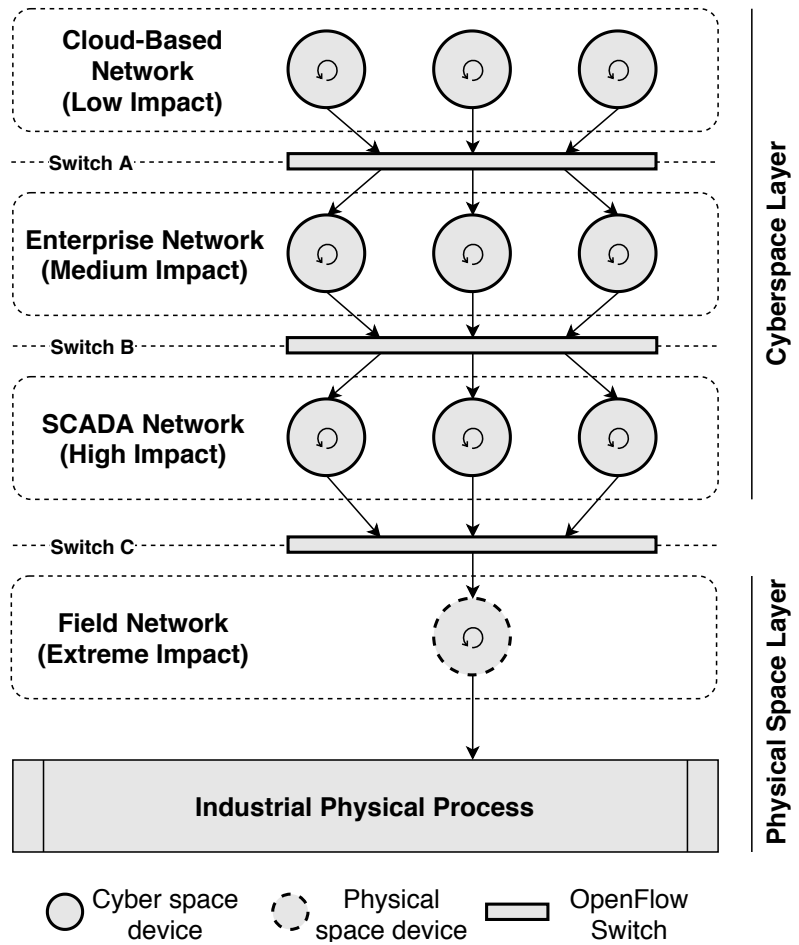


Figure 4.13: Proposed game model for virus spreading in a multi-subnet and cross-layer CPS network.

attacker tries to find and exploit a specific vulnerability on a particular node while the defender counters the attack by increasing the defense values to protect the system. Whenever the success rate of an attack exceeds the exploitability score of the selected vulnerability, the offense is considered auspicious, and the virus advances to the exploited node. However, the defender at each game step tries to make the system harder to breach by leveraging the complexity of exploiting vulnerabilities. First, RL is applied using Q-learning to find the optimal sequence attack path to reach the goal node. Moreover, MARL handling Q-learning is applied for both agents to reach the NE, and find the best defense strategies.

Game Formulation

Next is a six-tuples defining the essential parameters of the proposed game model: the adversaries agents tuple, a set of states representing the 14x6 game model matrix (14 nodes, 5 vulnerabilities, level of protection), two sets for attacker and defender action samples, a set of reward corresponding to each state, and finally a set of transition probabilities that is represented by the exploitability score of the vulnerabilities.

$$\left\{ \begin{array}{ll} \text{Agents:} & X = \{X_a, X_d\} \\ \text{States:} & S = \{\{S_1^1, \dots, S_1^6\}, \dots, \{S_N^1, \dots, S_N^6\}\} \\ \text{Att. actions:} & A = \{\{A_{S_1^1}, \dots, A_{S_1^5}\}, \dots, \{A_{S_N^1}, \dots, A_{S_N^5}\}\} \\ \text{Def. actions:} & D = \{\{D_{S_1^1}, \dots, D_{S_1^6}\}, \dots, \{D_{S_N^1}, \dots, D_{S_N^6}\}\} \\ \text{Rewards:} & R = \{\{R_{S_1^1}, \dots, R_{S_1^6}\}, \dots, \{R_{S_N^1}, \dots, R_{S_N^6}\}\} \\ \text{Trans. prob:} & P = \{\{P_{S_1^1}, \dots, P_{S_1^5}\}, \dots, \{P_{S_N^1}, \dots, P_{S_N^5}\}\} \end{array} \right.$$

The defender uses extra parameters during the simulation, which helps determine a win-game for the defender; a value W that returns zero if the virus is not detected and one if identified by the IDS, and another value T that indicates if the game is in a terminated state. In addition to value Y that delimits the steps of an attacker, and a success rate V for the attacker actions. N is used to indicate the index of the last node on the system:

$$\left\{ \begin{array}{ll} \text{IDS:} & W \in \{0 : \text{non-detected}, 1 : \text{detected}\} \\ \text{Terminal state:} & T \in \{0 : \text{non-terminal}, 1 : \text{terminal}\} \\ \text{Maximum steps:} & Y \in [0; 99] \\ \text{Success rate:} & V \in [0; 99] \\ \text{Last node index:} & N = 14 \end{array} \right.$$

Each agent of the game model has its goal state for winning a game:

- The goal state of the attacker is to reach the control units of the CPS network. It is represented by the end node of the proposed game model in Fig. 4.13 where the attack success rate should exceed the exploitability score; formally described in equation (4.1):

$$\begin{aligned} X_a^{goal} = \exists S_N^j \in S, A_{S_N^j} \in A \text{ and } V_{A_{S_N^j}} \in \mathbb{R} \mid \\ \forall j \in [1; 5], T = 1 \text{ and } V_{A_{S_N^j}} > P_{S_N^j} \end{aligned} \quad (4.1)$$

- For the defender goal state, a game is considered auspicious in two cases: First whenever the attacker actions success does not exceed the transition probability on the last node, and the IDS detects the attack then blocks it; formally depicted in equation (4.2). Second when the maximum attack steps are exceeded; presented in equation (4.3):

$$\begin{aligned}
X_d^{goal} &= \exists S_N^j \in S, A_{S_N^j} \in A, V_{A_{S_N^j}} \in \mathbb{R} \text{ and } Y \in \mathbb{R} \\
&| \forall j \in [1; 5], T = 1, W = 1, V_{A_{S_N^j}} \leq P_{S_N^j} \text{ and } Y \in [0; 99]
\end{aligned} \tag{4.2}$$

Or,

$$\begin{aligned}
X_d^{goal} &= \exists S_i^j \in S, A_{S_i^j} \in A, V_{A_{S_i^j}} \in \mathbb{R} \text{ and } Y \in \mathbb{R} | \\
&\forall j \in [1; 5], \forall i, T = 0, \forall V_{A_{S_i^j}} \text{ and } Y > 99
\end{aligned} \tag{4.3}$$

4.2.5 Implementation Setup

The goal in this approach is to design and implement a CPS network that is the closest in architecture and features to modern CPS networks and at the same time simple to be used for simulation and learning purposes. Fig. 4.14 shows the proposed framework architecture that is created with the combination of MiniCps network simulator, and OpenAI Gym toolkit to achieve the requirements of the proposed hybrid approach [10]. First, the CPS network is created using MiniEdit a helper tool in MiniCps [44] that allows the creation of a network with a drag and drop functionality and at the end generates a Python file describing the designed network with all its hosts, switches, and links. Each network device from Fig. 4.13 was implemented as a host, and the switches implemented as OpenFlow switches. Upon generation, the file is edited to support explicitly (1) private directories for each host on the CPS network, (2) IDS implementation to detect attacks and return proper observation for learning purposes, and (3) integration of vulnerability analysis on the hosts and switches of the CPS network. Using OpenAI Gym, the environment of the proposed game model is created; it is a

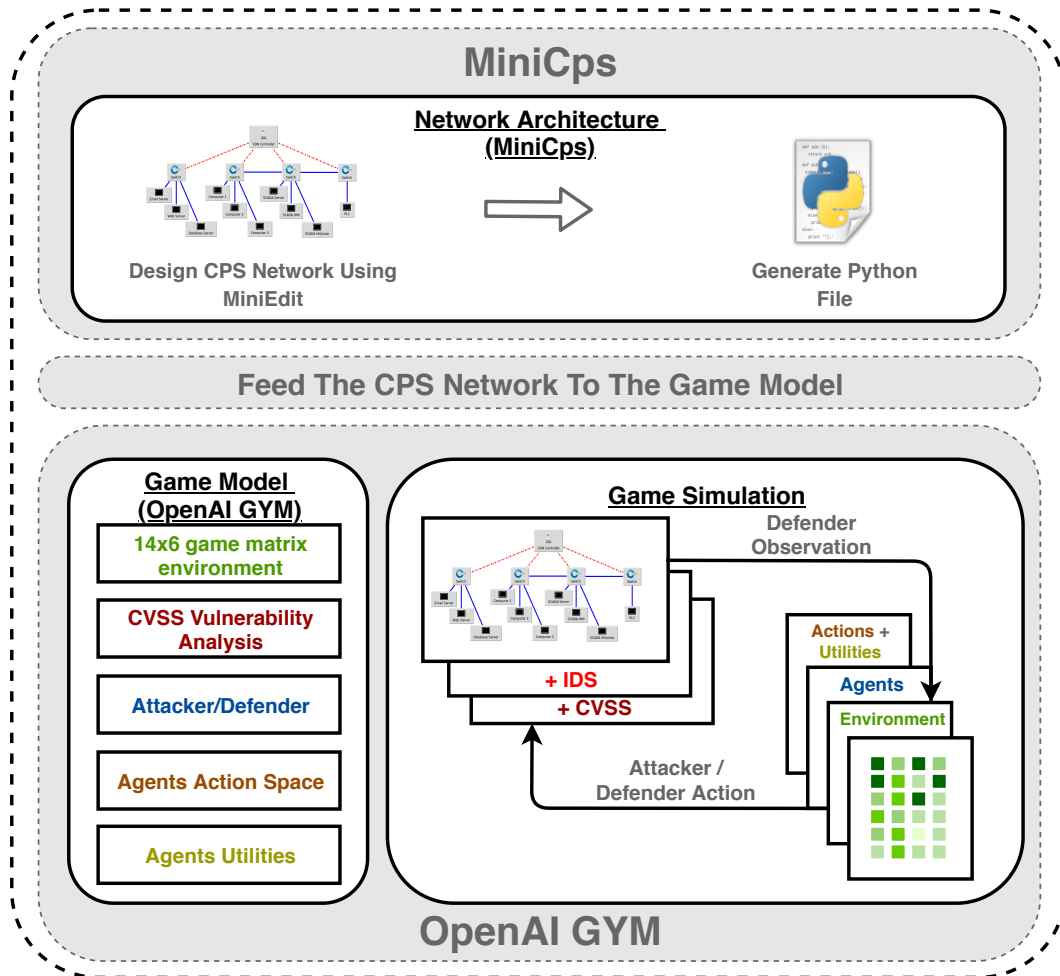


Figure 4.14: Proposed framework architecture using OpenAI Gym toolkit, and MiniCps network simulator.

14x6 matrix representing the 13 nodes of the network plus a starting node as rows and the set of five vulnerabilities plus the IDS strength for each host on the network as columns, at each node the first five cells of the matrix contains the exploitability score of a vulnerability on a specific node and the last cell contains the strength of the IDS on each node.

These exploitability scores represent the defense values of the defender and how the network is immune against attacks. Also using Gym library the action space of both agents are determined, the rewards for each node of the environ-

ment is set, and helper functions are created to apply the chosen action on the designed CPS network that is fed into the game model. By merging the CPS network created using MiniCps and its add-ons (Directory creation, IDS integration, and vulnerability analysis) with the created game model using OpenAI Gym, a robust framework is designed to conduct simulation, execute different kind of strategies, and apply learning algorithms to derive optimal defense policies to mitigate possible attacks such as viruses attacks and reduce network losses on a CPS network. In this study, Q-learning was used to achieve the learning element, the environment designed have a limited number of states with a limited number of actions allowed for agents this argue the use of an algorithm based on tabular representations instead of an algorithm based on neural networks. In addition to the problem that neural networks have with adversarial learning, where the network is slow in adapting to the continually changing environment [1]. The proposed framework is used along MARL implementation to conduct different game strategies, derive optimal attack and defense policies using Q-learning, and evaluate the proposed adversarial multistage game-theoretical model. In the beginning, two Q-tables are initialized with zeros for both agents. The rows are equal to the number of states, and columns corresponding to the number of actions for the agents. At each game stage, both agents take action and change the state of the environment while receiving a reward evaluating their actions, respectively. A trade-off between exploration and exploitation is determined using an epsilon value that starts with a high number indicating more exploration and decreases with the increase of iterations number meaning less exploration and more exploitation of the learned actions to states. The Q-tables are also updated at each game step, and at the end of the simulation runs Q-tables are concluded showing the best action to choose for each state of the environment. The com-

Hyperparameters	Hyperparameters Sets	Best Hyperparameter
Learning Rate	[0.3, 0.4, 0.5, 0.6]	0.5/0.6
Discount Factor	[0.7, 0.8, 0.9, 0.95, 0.99]	0.8/0.6
Decay Rate	[0.005, 0.006, 0.007, 0.008, 0.009]	0.008/0.005

Table 4.2: Hyperparameters tuning.

pleted Q-tables for the attacker helps discover potential vulnerabilities in the network to ease the attack and reduce the attack steps to reach the control units. Moreover, the completed Q-table for the defender helps discover optimal defense policies to counter the attacker’s action by patching the network vulnerabilities and improving the IDS strength in detecting attacks.

4.2.6 Hyperparameters Tuning

Below, we discuss the different hyperparameters used during training and their effect on the performance of learning. Table 4.2 shows all the hyperparameters used to tune the learning tuning, along the best hyperparameters found for attacker learning only, and for both attacker and defedner learning.

Learning Rate

The learning rate ranges between zero and one; it describes the degree of accepting a new value compared to previous values. Learning can occur quickly if the learning rate is close to one, and no learning develops when the rate is close to zero. For validation, we trained agents with different learning rates 0.3, 0.4, 0.5, 0.6. The best performance observed when deriving only attacker policies is when using a learning rate equal to 0.5. When deriving policies for both attacker and defender a learning rate equal to 0.6 performed better.

Discount Factor

The discount factor Gamma (γ) affects how much weight is considered for future rewards. A discount factor that is close to zero presents an immediate

reward. However, a discount factor close to one represents the discounted future reward. A set of discount factors is used to test the performance of learning agents. A discount factor equal to 0.8 performed best when deriving policies for the attacker only. On the other hand, for both agents, 0.6 as a discount factor performed better.

Decay Rate

Epsilon (ϵ) is the probability that selects the trade-off between exploration and exploitation. The decay rate is the value by which (ϵ) is decreased after an episode. A (ϵ) value that is close to one represents an exploration, and on the other hand, a (ϵ) value that is closed to zero shows exploitation. A set of decay rate is used to test the performance of the agents. 0.008 shows the best result for only attacker learning, and 0.005 shows the best outcome for both agents.

Chapter 5

Discussions and Results

To conclude an optimized sequence of attacks, and determine defense strategies with optimal policies, two different simulation scenarios are used to show first the potential of dedicated viruses attacks on a CPS network with no defense strategies, and second the urge of having optimized defense strategies to counter these kinds of attacks and reduce system losses.

5.1 Discussions

5.1.1 Strategy Vs. Policy

Fig. 5.1 gives an idea about the use of game theory and MARL; it correlates the notion of strategies in game theory with the concept of policies in RL. The problem faced in CPS networks is first modeled and formalized based on game theory principles. Each player has defined strategies and action space to act accordingly. Also, with the use of MARL, the formalization of the game model is implemented into an environment to apply the learning process using Q-learning algorithm. And afterward, derive optimal policies for the selected strategies.

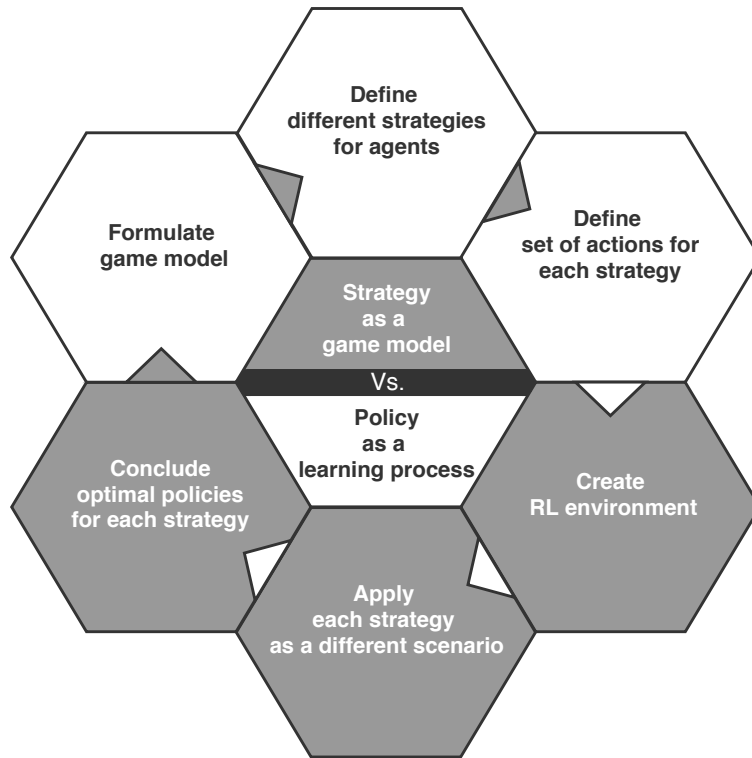


Figure 5.1: Strategy Vs. Policy.

5.1.2 Game Strategies

Each agent has three strategies; a common strategy is adopted by both agents, which is the random strategy that acts as a baseline. In this strategy, both agents will choose and apply actions to the environment based on arbitrary decisions. The attacker has a stealthy strategy that relies on reducing the number of steps and staying low key to avoid being detected. And the second strategy depends on budget spending to increase the chance of successful attacks. The defender in addition to the randomized strategy also has two strategies first one relies only on human capabilities and patch management, and the second relies on machine capabilities and IDS enhancement.

A simulation of 10^6 games is conducted with the application of MARL using the Q-learning algorithm with the proposed framework Fig. 4.14. The frame-

work requires several inputs such as the network architecture in addition to some configuration parameters required for the game model design. To depict the simulation results provided by the framework, a win-factor plot is used, which is a running average of the gained games for both agents over the entire simulation runs. A scale of $[1, -1]$ is used: (1) means the defender is the winner, (-1) means the attacker is the winner and presented with a green and red horizontal lines respectively.

5.2 Results

5.2.1 MARL Policies

First, attacker strategies are applied to learn their optimal policies and map actions to states. Fig. 5.2 shows three curves; In diamond, the random strategy acting as a baseline, and showing no clue for the virus attack in reaching the end goal and maximizing its rewards. In triangle, a stealthy strategy adopted by the attacker that shows how the attacker is learning with the increase of iterations runs. Finally, in circle, a budget spending strategy that performs better than the previous strategy due to the increase of attack success rate. In Fig. 5.2 optimal policies were derived for stealthy and budget spending strategies, that showed excellent performance compared to a randomized strategy.

Second, defender strategies are applied to learn their optimal policies and map actions to states. Fig. 5.3 shows three curves; In diamond, the same as previously described acting as a baseline. In triangle, a strategy that relies on human capabilities only; this strategy showed noticeable performance in preventing the attacker from winning more games. Finally, in circle, a strategy that relies only on machine capabilities showed also remarkable performance in limiting attacker success rate. In Fig. 5.3 optimal policies were derived for human and machine

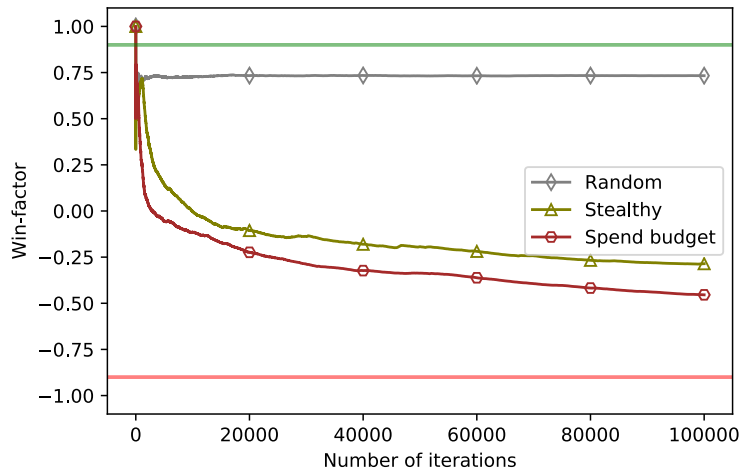


Figure 5.2: Win-factor for learned attacker vs. random defender.

strategies; that showed close results and prominent performance compared to only attacker strategies conducted in Fig. 5.2.

5.2.2 Testing Strategies With Learned Policies

In this section, testing is conducted on the learned policies for both attacker and defender strategies. The results show the percentage of successful attacks in a 10^6 game simulation runs for a virus attack starting from all four subnets independently and also shows the success rate by layer noting that the work incorporates both cyber and physical layer.

First, in Fig. 5.4, testing for attacker learned policies is done. Results show that virus attacks starting from all subnets the attack success rate was way above the average for both strategies. And it is summarized by layers; where a virus attack on average managed to succeed by 77% on the cyber layer and by 99% on the physical layer.

Second, in Fig. 5.5, testing for defender learned policies is done with the presence of attacker learned policies. Results show noticeable performance in

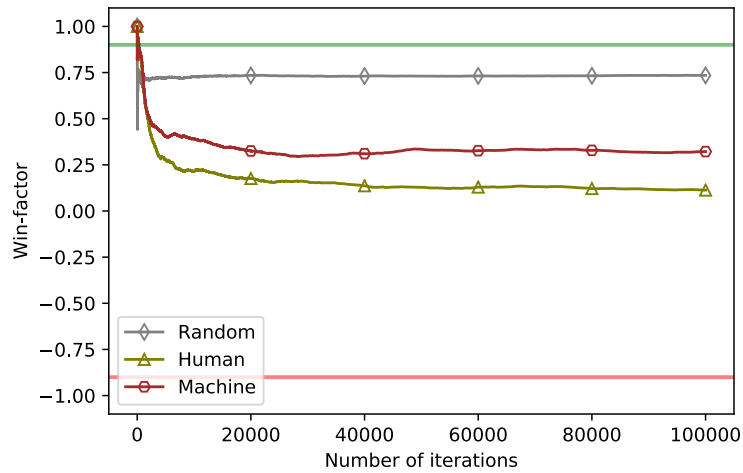


Figure 5.3: Win-factor for learned attacker vs. learned defender.

limiting the success rate of virus attacks. Optimized policies for the selected defender strategies showed that the defender is able to reduce the success rate of attacks, especially in the first three subnets that represent the cyber layer. However, these strategies barely changed the results and managed to limit viruses attacks on the last subnet which is hosted on the physical layer; due to the presence of the attacker goal where the defender has little to no chance in blocking attacks. As a summary, a virus attack on average managed to succeed by 25% on the cyber layer and by 94% on the physical layer. In conclusion, both defender strategies managed to limit virus attacks success rate.

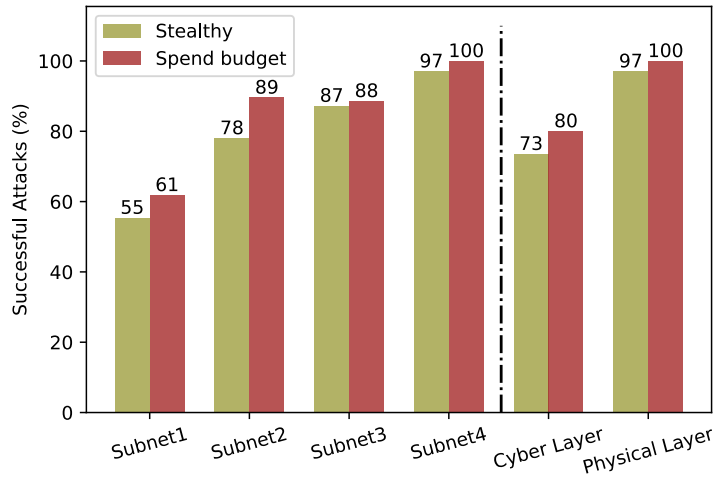


Figure 5.4: Testing only attacker strategies with derived policies.

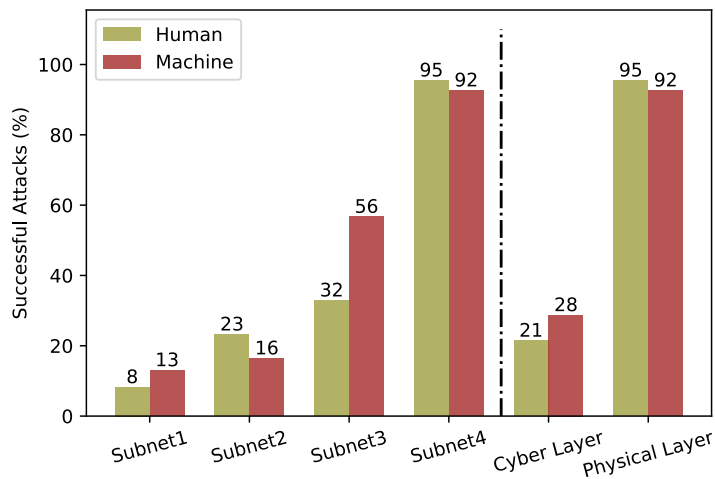


Figure 5.5: Testing both attacker and defender strategies with derived policies.

Chapter 6

Conclusion

In this thesis work, we reviewed and presented background information on game theory, attack trees, threats faced in SCADA systems, RL, and MARL. Lately used methodologies such as game theory and RL proved their applicability in addressing security issues in CPS. For the reason, as mentioned earlier, we discussed the current use of game theory and RL in cited related work solving security issues and finding adequate countermeasures for potential attacks in ICS and CPS. In our point of view, we find a promising future in addressing security problems in CPS using game theory as a formal tool to model security problems, especially between two adversaries, and RL as an AI technology to address human problems that need the intervention of machine intelligence.

We first evaluated the risk in terms of the NE of a zero-sum game in between a pen-tester equipped with an attack tree and a defender equipped with an IDS and a strategic response graph model. This approach is composed of three layers: the functional network layer, the security layer, and the game theory layer. Each agent has the best possible response given the portfolio of played strategies. A python tool has been developed to evaluate the payoffs given different variants of

randomness, and selected strategies for each agent. Results show the efficiency of this approach method in selecting the best strategy for mitigating attacks against CPS. We learned in this approach, the behavior that the defender should adopt in the case of facing a pen-tester. The defender must not rush to block the early steps in the pen-testing and leave the internal nodes with no protection or be tolerant in under-protecting the surface nodes.

Then we studied the case of a virus spreading in the different layers and subnets of a CPS. Future viruses will undoubtedly be equipped with automated learning components. We took this assumption and propose a framework based on a hybrid approach using game theory and RL to model an adversarial game for cross-layer virus spreading in CPS networks. The formulation includes a network architecture with different layers and subnets that mimics real and modern CPS network. The utilities of the game are the cumulative rewards that can be achieved by the RL agents after the learning phase. MARL was applied using Q-learning for both agents to learn optimal policies.

The obtained results show the ability of the defender to derive optimal defense policies based on a human-selected strategy to reduce losses and prevent viruses from spreading into the CPS networks. A mixed defense strategy can lead the game to a NE, where the attacker would not like to change behavior since it can be countered more easily. A particular focus in this approach was put on the RL level in terms of design and simulations. This approach allowed us to study and learn the behavior of a virus accommodated with AI capabilities to spread into CPS. On the other, we also discovered the potential of the defender in containing such targeted attacks while using the same capabilities. In future work, we will focus more on the strategic level, by dissecting the extended form security game with imperfect information.

As we all know, cybersecurity is a process and not a product; however, shortly, this process will translate from only human effort to a hybrid human-machine effort. In this thesis, we are installing the first building stone to a cybersecurity process that relies on human and machine intelligence combined to serve the security of CPS.

Appendix A

Abbreviations

AVC	Automatic Voltage Control
AI	Artificial Intelligence
CI	Critical Infrastructures
CPS	Cyber-Physical Systems
CPU	Central Processing Unit
CySeMoL	CyberSecurity Modeling Language
CVE	Common Vulnerabilities & Exposures
DMZ	Demilitarized Zone
FDI	False Data Injection
FTA	Fault Tree Analysis
HMI	Human Machine Interface
ICS	Industrial Control Systems
ICPS	Industrial Cyber-Physical Systems
IDS	Intrusion Detection Systems
IED	Intelligent Electronic Device
IEEE	Institute of Electrical and Electronics Engineers
IT	Information Technology
KDE	Kernel Density Estimation
LAN	Local Area Network
MLE	Maximum Likelihood Estimation
MTU	Master Terminal Unit
NE	Nash Equilibrium
NSM	Nearest Sequence Memories
PLC	Programmable Logic Controller
POMDP	Partial Observable Markov Decision Process
PRA	Probabilistic Risk Assessment
RA	Risk Assessment
RAM	Random Access Memory
RAMCAP	Risk Analysis and Management for Critical Asset Protection
RL	Reinforcement Learning

ROI	Return on Investment
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
SDN	Software Defined Network
SIEM	Security Information and Event Management
SVD	Singular Value Decomposition
WAN	Wide Area Network
WMD	Weapon of Mass Destruction

Bibliography

- [1] R. Elderman, L. J. Pater, A. S. Thie, M. M. Drugan, and M. Wiering, “Adversarial reinforcement learning in a cyber security simulation.,” in *ICAART (2)*, pp. 559–566, 2017.
- [2] Y. Chen, S. Huang, F. Liu, Z. Wang, and X. Sun, “Evaluation of reinforcement learning-based false data injection attack to automatic voltage control,” *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 2158–2169, 2019.
- [3] K. Huang, C. Zhou, Y. Qin, and W. Tu, “A game-theoretic approach to cross-layer security decision-making in industrial cyber-physical systems,” *IEEE Transactions on Industrial Electronics*, 2019.
- [4] Z. Ni and S. Paul, “A multistage game in smart grid security: A reinforcement learning solution,” *IEEE transactions on neural networks and learning systems*, 2019.
- [5] “Common vulnerabilities and exposures (cve).” <https://www.cvedetails.com>. Accessed: 2019-08-05.
- [6] R. Langner, “Stuxnet: Dissecting a cyberwarfare weapon,” *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.
- [7] T. M. Chen, “Stuxnet, the real start of cyber warfare?[editor’s note],” *IEEE Network*, vol. 24, no. 6, pp. 2–3, 2010.
- [8] “Shodan ics radar.” <https://www.shodan.io>. Accessed: 2019-10-2.
- [9] “Shodan ics radar.” <https://ics-radar.shodan.io/>. Accessed: 2019-10-2.
- [10] “projectcps.” <https://github.com/josephKhoury95/projectCPS>. Accessed: 2019-11-19.
- [11] K. Stouffer, J. Falco, and K. Scarfone, “Guide to industrial control systems (ics) security,” *NIST special publication*, vol. 800, no. 82, pp. 16–16, 2011.

- [12] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, “Using model-based intrusion detection for scada networks,” in *Proceedings of the SCADA security scientific symposium*, vol. 46, pp. 1–12, Citeseer, 2007.
- [13] E. P. Leverett, “Quantitatively assessing and visualising industrial system attack surfaces,” *University of Cambridge, Darwin College*, vol. 7, 2011.
- [14] T. T. Tesfay, J.-P. Hubaux, J.-Y. Le Boudec, and P. Oechslin, “Cyber-secure communication architecture for active power distribution networks,” in *Proceedings of the 29th annual acm symposium on applied computing*, pp. 545–552, ACM, 2014.
- [15] R. F. Dacey, “Critical infrastructure protection: Challenges and efforts to secure control systems (testimony before the subcommittee on technology information policy, intergovernmental relations and the census, house committee on government reform),” 2004.
- [16] A. Matrosov, E. Rodionov, D. Harley, and J. Malcho, “Stuxnet under the microscope,” *ESET LLC (September 2010)*, 2010.
- [17] A. Ginter and W. Sikora, “Cybersecurity for chemical engineers,” *Chemical Engineering*, vol. 118, no. 6, p. 49, 2011.
- [18] “Information technology — Security techniques — Information security risk management,” standard, International Organization for Standardization, July 2018.
- [19] M. Rausand, *Risk assessment: theory, methods, and applications*, vol. 115. John Wiley & Sons, 2013.
- [20] “Nist cybersecurity framework.” <https://www.nist.gov/cyberframework>. Accessed: 2019-05-25.
- [21] “Electricity subsector cybersecurity capability maturity model (es-c2m2).” <https://www.energy.gov/ceser/activities/cybersecurity-critical-energy-infrastructure/energy-sector-cybersecurity-0-1>. Accessed: 2019-05-25.
- [22] R. White, “Cybersecurity policy for water and electricity infrastructures.” <https://www.coursera.org/learn/cybersecurity-policy-water-electricity>. Accessed: 2019-05-25.
- [23] R. Nickell, J. Jones, and K. Balkey, “Ramcap: Risk analysis and management for critical asset protection, overview of methodology,” in *AMSE International Mechanical Engineering Congress and RD7D Expo, Anaheim, CA*, pp. 13–19, 2004.

- [24] S. Lüders, “Why control system cyber-security sucks.” <https://www.blackhat.com/us-14/archives.htmlwhy-control-system-cyber-security-sucks->, 2014. CERN.
- [25] J. Weiss, “Cyber security issues with level 0 through 1 devices.” <https://media.defcon.org/DEF2018>.
- [26] T. Sommestad, M. Ekstedt, and H. Holm, “The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures,” *IEEE Systems Journal*, vol. 7, no. 3, pp. 363–373, 2013.
- [27] F. Baiardi and D. Sgandurra, “Assessing ict risk through a monte carlo method,” *Environment Systems and Decisions*, vol. 33, no. 4, pp. 486–499, 2013.
- [28] P. A. Ralston, J. H. Graham, and J. L. Hieb, “Cyber security risk assessment for scada and dcs networks,” *ISA transactions*, vol. 46, no. 4, pp. 583–594, 2007.
- [29] C. A. Ericson, “Fault tree analysis,” *Hazard analysis techniques for system safety*, pp. 183–221, 2005.
- [30] B. Schneier, “Attack trees,” *Dr. Dobb’s journal*, vol. 24, no. 12, pp. 21–29, 1999.
- [31] G. D. Wyss, F. A. Durán, and V. J. Dandini, “An object-oriented approach to risk and reliability analysis: methodology and aviation safety applications,” *Simulation*, vol. 80, no. 1, pp. 33–43, 2004.
- [32] A. Pillay and J. Wang, “Risk assessment of fishing vessels using fuzzy set approach,” *International Journal of Reliability, Quality and Safety Engineering*, vol. 9, no. 02, pp. 163–181, 2002.
- [33] S. C. Patel, J. H. Graham, and P. A. Ralston, “Security enhancement for scada communication protocols using augmented vulnerability trees.,” in *CAINE*, pp. 244–251, 2006.
- [34] E. J. Byres, M. Franz, and D. Miller, “The use of attack trees in assessing vulnerabilities in scada systems,” in *Proceedings of the international infrastructure survivability workshop*, 2004.
- [35] C.-W. Ten, C.-C. Liu, and M. Govindarasu, “Vulnerability assessment of cybersecurity for scada systems using attack trees,” in *Power Engineering Society General Meeting, 2007. IEEE*, pp. 1–8, IEEE, 2007.

- [36] N. Alhebaishi, L. Wang, S. Jajodia, and A. Singhal, “Threat modeling for cloud data center infrastructures,” in *International Symposium on Foundations and Practice of Security*, pp. 302–319, Springer, 2016.
- [37] “Kodakvs.polaroid.” <http://home.ku.edu.tr/lkockesen/teaching/econ333/slides/7-Extensive-Form-Games-Handout.pdf>. Accessed: 2019-10-2.
- [38] A. Fielder, E. Panaousis, P. Malacaria, C. Hankin, and F. Smeraldi, “Game theory meets information security management,” in *IFIP International Information Security Conference*, pp. 15–29, Springer, 2014.
- [39] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.
- [40] X. Ou, W. F. Boyer, and M. A. McQueen, “A scalable approach to attack graph generation,” in *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 336–345, ACM, 2006.
- [41] P. Cichonski, T. Millar, T. Grance, and K. Scarfone, “Computer security incident handling guide,” *NIST Special Publication*, vol. 800, no. 61, pp. 1–147, 2012.
- [42] J. Creasey, *Cyber security incident response guide*. CREST, 2013.
- [43] M. H. Rehmani, A. Davy, B. Jennings, and C. Assi, “Software defined networks based smart grid communication: A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, 2019.
- [44] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” 2016.
- [45] “Amazon web services (aws).” <https://aws.amazon.com>. Accessed: 2019-08-05.
- [46] “Common vulnerability scoring system.” <https://www.first.org/cvss/>. Accessed: 2019-08-02.
- [47] H. Herry, E. Band, C. Perkins, and J. Singer, “Peer-to-peer secure updates for heterogeneous edge devices,” in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–5, IEEE, 2018.