QATAR UNIVERSITY

COLLEGE OF ENGINEERING

INTRUSION RESPONSE FOR CYBER-PHYSICAL SYSTEMS: A MODEL-FREE DEEP REINFORCEMENT LEARNING APPROACH

BY

MAY SAED MOHAMED BASHENDY

A Thesis Submitted to

the College of Engineering

in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computing

June 2022

COMMITTEE PAGE

The members of the Committee approve the Thesis of

May Bashendy defended on 11/05/2022.

Dr. Abdelkarim Erradi
Thesis/Dissertation Supervisor

Dr. Khaled Khan
Committee Member

Dr. Elias Bou-Harb
Committee Member

Dr. Ridha Hamila
Committee Member

Dr. Ahmed Massoud Abdou
Associate Dean of Research and Graduate Studies

Approved:

Khalifa Nasser Al-Khalifa, Dean, College of Engineering

ABSTRACT

BASHENDY, MAY, S., Masters: June : 2022,

Masters of Science in Computing

Title: Intrusion Response for Cyber-Physical Systems: A Model-Free Deep Reinforcement Learning Approach

Supervisor of Thesis: Abdelkarim, E, Erradi.

Cyberattacks on Cyber-Physical Systems (CPSs) are on the rise due to CPS increased networked connectivity, which may cause costly environmental hazards as well as human and financial loss. Although the connectivity of CPSs has significantly improved production, it introduced new vulnerabilities, which necessitate designing and implementing proper automatic cybersecurity defensive mechanisms to protect CPSs from cyberattacks. This thesis presents the design, implementation, and evaluation of a dynamic Intrusion Response System (IRS) to automatically respond to false data injection attacks against a model-based CPS testbed. The proposed IRS was designed using two approaches: an optimization approach with Genetic algorithm and a model-free Deep Reinforcement Learning-based (DRL) approach using Double Deep Q Networks (DDQN) algorithm. The proposed solutions are evaluated on an online Continuous Stirred Tank Reactor (CSTR) testbed that mimics a real-world CPS. Experimental results demonstrate the effectiveness of the proposed approaches in responding to false data injection attacks and minimize the impact on the system. Finally, the thesis highlights open research questions and sketches directions for future work.

DEDICATION

To mom and dad.

To my family for their prayers, love, and endless support.

To my friends for their encouragement and help.

ACKNOWLEDGMENTS

I wish to express my sincere gratitude and appreciation to my supervisor Dr. Abdelkarim Erradi for his guidance and support. My appreciation is also extended to Dr. Ashraf Tantawy for his continuous help, encouragement, and valuable feedback. This work would have never been possible without their assistance and supervision.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

CPS Cyber Physical System

ICT Information and Communication Technology

CSTR Continuous Stirred Tank Reactor

IPS Intrusion Prevention System

IDS Intrusion Detection System

IRS Intrusion Response System

AIRS Automatic Intrusion Response System

ML Machine Learning

RL Reinforcement Learning

ReLU Rectified Linear Unit

DNN Deep Neural Network

DQN Deep Q Network

DDQN Double Deep Q Network

GA Genetic Algorithms

NSGA Non-dominated Sorting Genetic Algorithms

VI Value Iteration

MITM Man in The Middle

DoS Denial of Service

MDP Markov Decision Process

HMI Human Machine Interface

BPCS Basic Process Control System

SIS Safety Instrumented System

PID Piping & Instrumentation Diagram

PDF Probability Density Function

CDF Cumulative Distribution Function (CDF)

POS Pareto Optimal Solution

CVSS Common Vulnerability Scoring System

SDG Service Dependency Graph

MLBN Multilevel Bayesian Network

MOOP Multi-objective optimization problem

ADT Attack Defense Tree

GPU Graphics Processing Unit

TTR Time to Recover

C Hazard Cost

Acost Availability Cost

P Profit per second

PP Production Profit

t Sampling time

M Scaling factor

N Scaling factor

Z Scaling factor

$\boldsymbol{R_{GA}}$ The reward function used for the GA-IRS approach

$\boldsymbol{R_{DRL}}$ The reward function used for the DRL-IRS approach

Exp1 Experiment 1

Exp2 Experiment 2

Exp3 Experiment 3

HL Hidden Layer

CHAPTER 1: INTRODUCTION

Cyber-Physical Systems (CPSs) integrate a physical plant with different embedded computing, communication networks, and human interfacing to monitor and control a process [1]. CPSs domains include manufacturing, energy, transportation, and health care. Originally, these systems were not considered vulnerable since they were designed to operate at physically isolated locations that run on proprietary hardware and software. However, with the recent evolution in information and communication technology (ICT), these systems have been networked to integrate them with corporate systems and enable remote access capabilities. In addition, to enable communication and interoperability between the different CPS components, open standard industrial protocols such as Modbus, DNP3, and CIP are used. Unfortunately, these protocols lack sufficient security mechanisms, such as encryption and authentication [2]. Consequently, such vulnerabilities have exposed CPSs to new security threats and made them an easy prime target for cyberattacks that aim to gain unauthorized access to the control network and cause disturbances/hazards in the physical process. The work in [3], [4], [5], and [6] studies and summarizes the different cyberattacks against CPSs.

According to [7], Figure 1 illustrates the general basic operations of CPSs. The controller utilizes sensing and actuation components to control the process. The sensors and actuators are connected to one or more controllers, where the control algorithms are implemented. The controller receives and interprets the sensors' signals. Then, based on the control logic and the target set points, the controller sends the suitable values to the actuators to adjust the state of the process. The operator can use the HMI to monitor and configure the process. Further details on CPSs components, architecture, protocols, vulnerabilities, and testbeds are available in [8], [9], and [10].

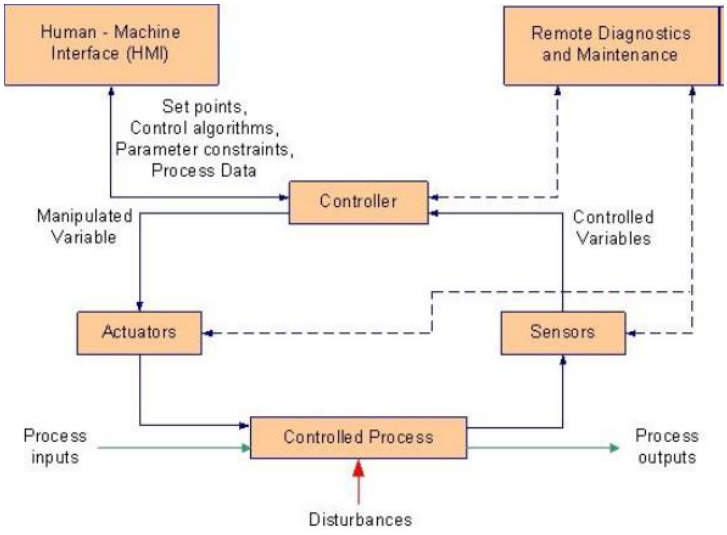

Figure 1. CPS operations [7]

Since CPSs are typically mission-critical systems, they are intolerant to errors or delays, which can cause catastrophic human, economic, and financial loss if no proper detection and response mitigation mechanisms are applied. Intrusion Detection Systems (IDSs) for a CPS have been extensively considered and studied in the literature [11], [12], [13], [14], [15], and [16]. However, they are not sufficient alone to protect CPSs. Accordingly, IDSs must be combined with IRSs to provide effective security protection for CPSs.

Intrusion Response Systems represent an essential protection layer for CPSs. They are responsible for automatically selecting and deploying optimal responses to mitigate the impact of the detected attack and keep the process in a safe state without any human intervention. Nevertheless, despite their necessity in securing CPSs, they have received less attention from the research community. This is because the design of IRSs is very complex and dangerous since poorly designed ones can result in higher damage than the damage of the intrusion itself. Besides, most of the studies that consider the design of IRS in a CPS focus on model-based approaches, which are challenging to design, time-consuming, and not accurate nor suitable for large systems. Also, most of these studies considered actions on only one level at a time, either the cyber level or the process level. Accordingly, designing an IRS for CPS that is model-free, which does not require a deep understanding of the process's

implementation details, generalizes to unseen situations, handles high-dimensional data, and combines different action levels is highly needed. In this thesis, a novel model-free deep reinforcement learning-based intrusion response agent for a CPS, that considers simultaneously cyber and physical actions, is designed using DDQN to mitigate the effect of false data injection attacks.

1.1 Problem Statement

The increased adoption of networked systems in critical infrastructures has made CPSs vulnerable to cyberattacks, which could cause catastrophic financial and human loss. Thus, CPSs should have the ability to respond at runtime to detected malicious attacks by performing appropriate defensive actions. To the best of our knowledge, there aren't much research works that focus on studying the application of reinforcement learning in intrusion response systems, especially for CPSs. For that reason, this thesis focuses on investigating the applicability of having a model-free deep reinforcement learning agent for building an intrusion response system using the DDQN algorithm for a CPS testbed. The designed intrusion response agent considers both cyber level and process level actions to mitigate the effect of false data injection attacks. More specifically, this thesis works towards answering the following two research questions:

- How can model-free deep reinforcement learning be adopted for building effective IRS for a given CPS testbed?
- Are model-free deep reinforcement learning intrusion response systems effective for cyber physical systems?

1.2 Thesis Objective and Contributions

The main objective of this thesis is to design an intrusion response system capable of automatically selecting optimal actions and responding to false data injection attacks against a CPS testbed. It is a dynamic IRS that reasons about an ongoing attack based on the observed alerts and determines an appropriate response to take. In this thesis, the problem of intrusion response in CPSs is tackled using two different approaches. In the first approach, we introduce the GA-IRS, an intrusion response system based on a conventional Genetic

algorithm. The second approach, which is the main focus of the thesis, is the DRL-IRS, an intrusion response system based on a model-free deep reinforcement learning DDQN algorithm. Accordingly, the main contributions of this thesis can be summarized as follows:

1. Present a taxonomy of IRSs design approaches based on different design parameters.
2. Review the state of the art on IRSs for CPSs and non-CPSs using both conventional optimization-based approaches and Reinforcement Learning (RL)-based approaches.
3. Model and design of a CPS testbed
4. Model and design of cyberattacks against the CPS testbed using an attack tree model.
5. Design an intrusion response system using Genetic algorithm
6. Setting up an RL environment and integrating it with the CPS testbed.
7. Design a novel model-free deep reinforcement learning intrusion response agent for a CPS testbed using DDQN algorithm.
8. Collect a representative dataset that includes the <state, action, next state, reward> tuples to be used for future offline approaches using RL.
9. Evaluate the performance of the proposed solution.
10. Discuss open challenges and possible directions for future research work in the area.

Note that Contributions 1 and 2 are compiled into a journal publication under review titled “Intrusion Response Systems for Cyber-Physical Systems: A Comprehensive Survey” and submitted to Journal of Computers and Security. Moreover, contributions 3 and 4 were compiled into a conference paper titled “Design and Implementation of Cyber-Physical Attacks on Modbus/TCP Protocol” which was published in the World Congress on Industrial Control Systems Security Conference (WCICSS-2020). Also, contributions 6 and 7 are compiled into a journal publication titled “Intrusion Response for Cyber-Physical Systems: A Model-Free Deep Reinforcement Learning Approach” and submitted to Journal of Network and Computer Applications.

1.3 Thesis Outline

The content of this thesis is divided into nine chapters to address the defined objectives and contributions. A schematic illustration of the thesis outline is shown in Figure

2. The figure starts with Chapter 1 which introduces CPSs, IRSs, the problem statement, thesis objectives, and thesis contributions. Chapter 2 provides background information on CPS cybersecurity, attacks taxonomy, and IRS taxonomy. Chapter 3 reviews and compare the different conventional and RL-based decision-making approaches. Chapter 4 surveys the state-of-the-art solutions and pinpoints the limitations in the literature. Chapter 5 models and describes the used CPS testbed with its architecture, components, communication, and designed cyberattacks. The proposed solution methodology of the intrusion response system problem is then solved in Chapter 6 and Chapter 7. Chapter 6 solves the IRS problem using a Genetic algorithm approach, while Chapter 7 solves it using a model-free deep reinforcement learning approach applying the DDQN algorithm. The experimental scenarios, training results, evaluation, challenges, and dataset collection are presented in Chapter 8. Finally, Chapter 9 concludes the thesis and highlights the future directions.

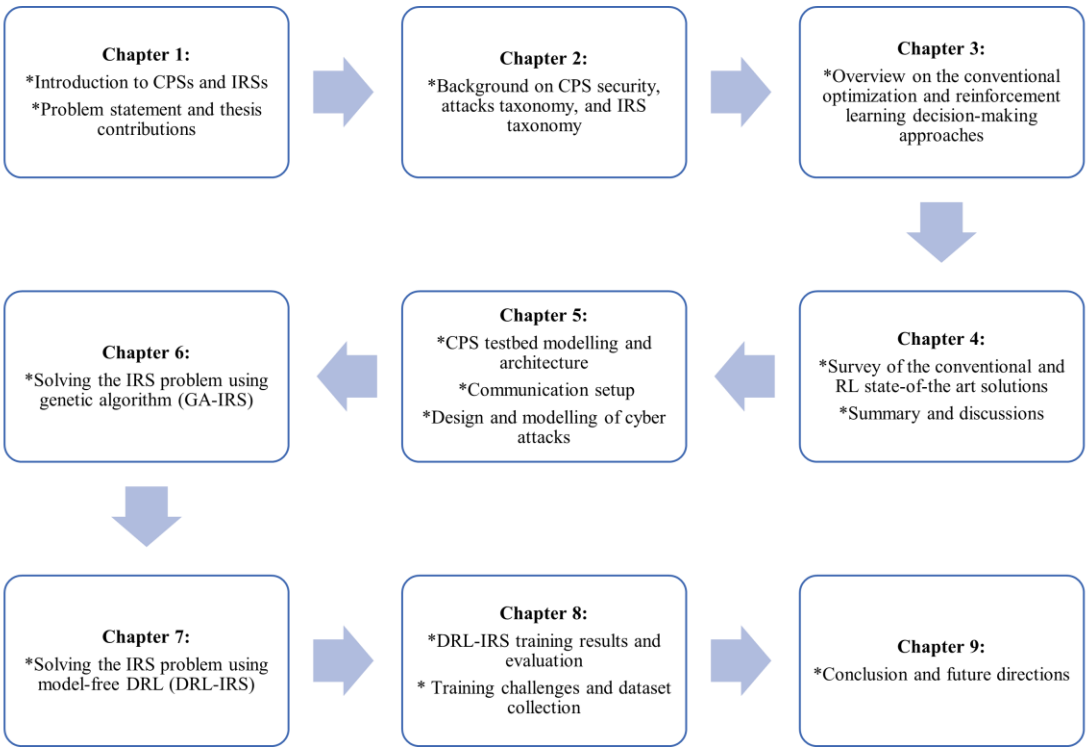

Figure 2. Thesis outline schematic

CHAPTER 2: BACKGROUND

This chapter starts with the motivation behind the necessity of designing and implementation proper automatic cybersecurity defensive mechanisms to protect CPSs from cyberattacks. It then provides a comprehensive attacks taxonomy and IRS systems taxonomy.

2.1 CPS Security Defense Mechanisms

CPS security is an ever-growing area of research for protecting CPSs from intruders who threaten the integrity and the availability of the physical process. It involves securing all the machines that communicate over the network, the physical hardware, and the data transfer between the different nodes. The work in [17], [18], [19], and [20] reviews and discusses the cybersecurity challenges and the different solutions to address the shift of control systems from stand-alone systems to unsecure networked CPSs.

Not all cybersecurity mechanisms used to protect Information Technology (IT) systems are suitable nor directly applicable to CPSs. This owes to the fact that CPSs integrate the cyber and the physical worlds, while IT systems only have cyber interactions. Accordingly, the connection with the physical world in a CPS has unique security requirements different from those found in traditional IT systems. For example, IT systems focus on maintaining the confidentiality security goal as the highest priority. However, in CPSs, the real-time availability and integrity of the physical devices have the highest priority. Also, the safety requirement of a CPS is much higher than that of an IT system. Besides, CPSs are more prone to security risks at different network levels and intolerable to errors or delays. Hence, mapping the security solutions of ITSs to CPSs is not ideal because they were not intended originally for them. This clearly illustrates why CPS security is more complex than IT security and hence, is currently an active research field. Interested readers can refer to [21] for more analysis on the vulnerabilities and threats facing critical CPSs.

According to [22], many countries have suffered from catastrophic cyberattacks against their CPSs. For example, in 2012, Qatar's RasGas oil company was attacked by a virus that brought down all the company's computers and forced a close until recovery [23]. The authors in [24] also mentioned the DDoS BlackEnergy Malware that targeted the

Ukrainian Power-grids for political motives in 2015. The Stuxnet worm and the Shamoon virus targeted Iran's nuclear infrastructure in 2010 and Saudi Arabia's oil production plant in 2012, respectively [25]. The United Arab Emirates (UAE) also suffered from Malware that targeted their energy sector in 2015. Furthermore, the authors in [26] highlighted more incidents against CPSs, such as the attack on the German steel factory in 2014, which caused massive damages because the plant was unable to perform the safety shutdown procedures.

In light of the previously mentioned threats, researchers emphasize the need to deploy well-designed and robust defensive security systems to secure critical cyber-physical processes. Figure 3 shows the recommended flow of defensive security mechanisms, which includes a prevention phase, detection phase, response phase, and forensics phase. In the prevention phase, the Intrusion Prevention Systems (IPSs) continuously monitor the process and generate proactive responses to prevent intrusions. In the detection phase, IDSs generate a passive alert response after identifying deviations in the traffic by using different detection techniques as summarized in Table 1. More details on the different IDS techniques and classifications are reported in [27], [28], and [29]. In the response phase, IRSs generate a reactive response in a timely manner to handle intrusions and mitigate their effect on the attacked system. The design of IRSs is the most complex and challenging part since poorly designed ones can result in higher damage than the damage of the intrusion itself. In the forensics phase, the security teams investigate the logged data from the previous phases to understand what had happened and how similar events can be avoided in the future.

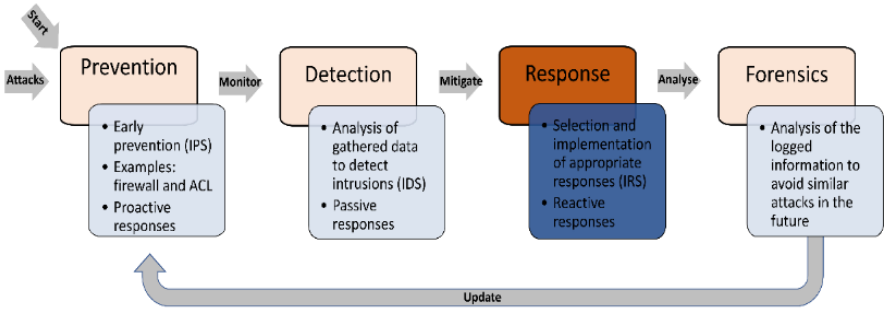

Figure 3. The flow of the defensive mechanisms

Table 1. Intrusion Detection Techniques

Detection Technique	Type	IDS Description
Knowledge based	Signature based	Looks for defined attack patterns in the traffic
	Protocol analysis	Identifies deviations of the protocol states/specification
Behavior based	Rule based	Detects an attack only if has a specific rule to detect it
	Process analysis	Uses the physical models to predict the expected output
	Statistical techniques	Uses statistical methods to determine deviations from the normal expected behavior
	Machine learning techniques	Learns from data and makes predictions based on them

In this thesis, the main focus is to investigate and design the response phase of the defensive mechanisms. Accordingly, it reviews existing conventional and reinforcement learning based decision-making solutions for IRSs. Also, it designs a novel model-free DRL-based agent to automatically respond to false data injection attacks against a CPS testbed.

2.2 Attacks Taxonomy

Figure 4 classifies attacks according to their targeted security objective into confidentiality, integrity, and availability attacks, which are summarized below:

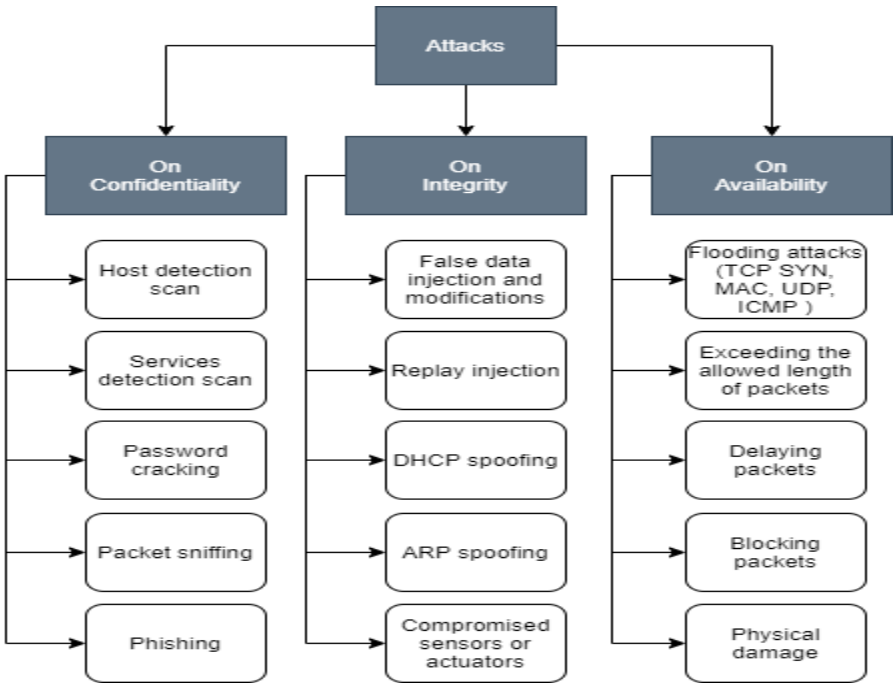

Figure 4. Attacks taxonomy against CPSs

2.2.1 Attacks on Confidentiality

Attacks on confidentiality, also referred to as reconnaissance attacks, are information gathering attacks. These attacks are the initial steps that attackers take to discover, study, analyze, and collect information about the targeted network before developing any sophisticated attacks [30]. During this phase, the attacker can know the network capabilities, network topologies, hosts with their IP and MAC addresses, used protocols, running services, open ports, etc. The work in [31] studies the different types of reconnaissance attacks and mentions some applicable detection and mitigation techniques.

2.2.2 Attacks on Integrity

Attacks on integrity involve unauthorized manipulation, injection, or modification of the data and the system resources. It occurs when the attacker compromises the communication channel between two components and intercepts exchanged messages. This enables several attacks, including Man in The Middle attack (MITM) that modifies packets payload and replay attack that plays back old stale recorded messages. These data tampering attacks are serious threats, particularly to CPSs. The work in [32] identifies data integrity attacks in terms of their criticality, strategy, detection approaches, and countermeasure recommendations.

2.2.3 Attacks on Availability

Attacks on availability, such as Denial of Service (DOS) attacks, involve crashing a target device or a communication connection by either flooding the target with enormous noise traffic or changing some of the fields of the transmitted packets. Accordingly, the target will not be able to perform its normal intended services and will eventually halt all the services running on it, which may cause catastrophic impacts. Several studies show the vulnerability of cyber-physical systems to availability attacks, such as DoS, including the work in [33] that models and assesses DoS attacks to sensors and actuators of a CPS.

2.3 IRS Taxonomy

Response systems are considered extremely important yet challenging to design due to the complexity of the CPS network and the complexity of attacks. IRSs are categorized based on their different design parameters according to [34], [35], [36], [37], and [38]. Initially, depending on their level of automation, they are divided into notification, manual, and automatic systems. In the notification systems [39], similar to an IDS, the IRS generates an alert to the operators, who are responsible for deciding on a suitable mitigation action to stop the attack. In the manual systems, the IRS generates a list of possible predefined countermeasures to mitigate the detected attack and sends it to the operators to choose from. Both notification and manual systems introduce delays between detection and response time. In the automatic intrusion response systems (AIRSs) [40], which is the focus of this taxonomy, the optimal response is selected and executed automatically without the need for any human intervention. This makes AIRSs suitable for the time and availability demands of CPSs. Figure 5 summarizes the evolution of the IRSs area, with its different cyber-response solution approaches. These classifications are briefly explained below:

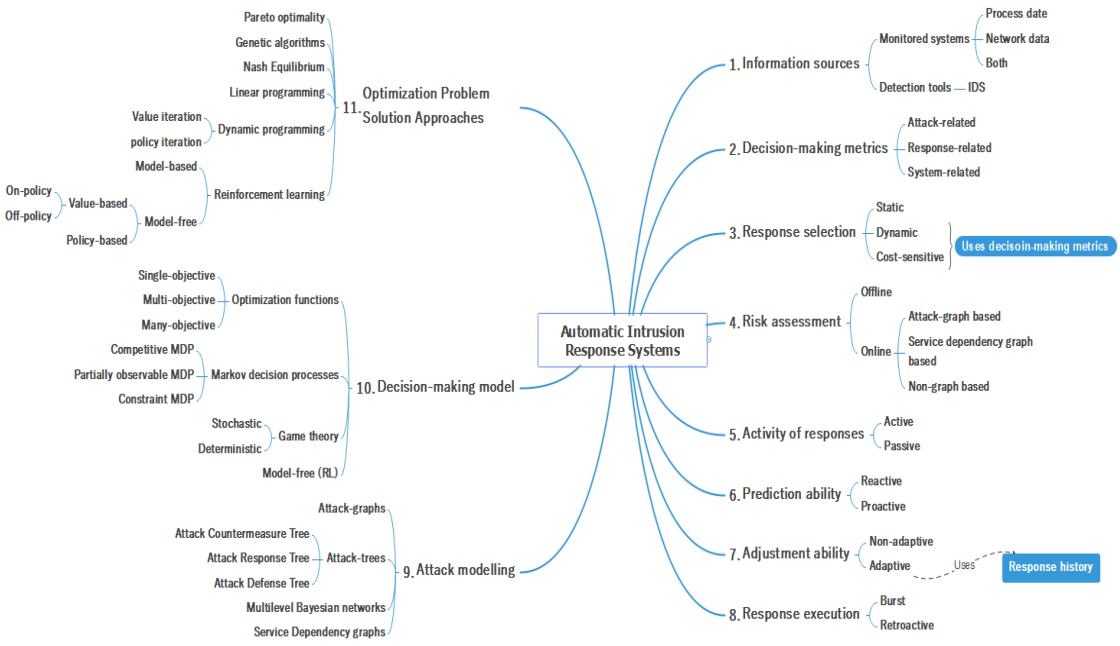

Figure 5. AIRS taxonomy

2.3.1 Information Sources

Intrusion response systems expect inputs from different information sources, including the monitored system and the different detection tools, such as IDS. From the monitored system, valuable information can be extracted, such as the system data, network topologies, and asset configurations. From an IDS, the IRS can collect intrusion alerts, intrusion confidence level, intrusion severity level, and so on. These collected information aids the IRS in selecting the optimal response.

2.3.2 Decision-making Metrics

Various metrics can be considered in the response selection process. They are classified into three different categories, which are attack-related, response-related, and system-related. Examples for each metric type are shown in Table 2. IRSs use either one-type metric or a combination of different metrics to build a more effective response system. For more detailed analysis of the different decision-making metrics, the reader can refer to [41], [42], and [43].

Table 2. Decision-Making Metrics

Type	**Metrics**
Attack-related	Frequency, type, confidence, time, and severity
Response-related	Cost, time, negative impact, positive effect, and goodness
System-related	Importance of assets

2.3.3 Response Selection

AIRSs have three response selection techniques to map the detected attacks with their suitable responses, which are static, dynamic, and cost-sensitive mappings. In static mapping, each attack is mapped to a specific predefined response through a mapping table. This strategy is implemented by SNORT [44]. In dynamic mapping, each attack is mapped to a predefined set of responses in which the optimal response is chosen based on the attack-related decision metrics. This shows that it is possible to have different countermeasures to the same attack type. The authors in [45] utilize the dynamic mapping

approach when designing their response strategy model. In cost-sensitive mapping, the response selection process takes into consideration the countermeasure cost and the intrusion damage cost, which are not considered in static and dynamic mapping. A response is then chosen only if its cost is less than the intrusion damage as implemented in [46] and [47].

2.3.4 Risk Assessment

Risk assessment is used by automatic cost-sensitive IRSs and is categorized into offline assessment and online assessment. In offline/static risk assessment as in [48], an IRS calculates the risks on the system in advance depending on static values that are assigned by experts on every asset in the network. On the other hand, online/dynamic risk assessment calculates the risk in attack time, taking into consideration the dependencies between the different assets and the current number of users using them as in [49]. Three approaches are used in the online assessment, which are attack graph-based [50], service dependency graph-based [51], and non-graph based [52]. The authors in [53] and [54] review and propose different risk assessment methods.

2.3.5 Activity of Responses

Responses can be either passive or active. Active responses change the state of the environment/system to effectively block the attack and mitigate its negative impact. The work in [55] is one of the examples that use active actions. On the other hand, passive ones do not change the state of the environment. They only notify operators and raise alerts. The work in [56] and [57] enumerate the different possible actions for each type.

2.3.6 Prediction Ability

AIRSs are classified into reactive and proactive systems in regard to the considered response time. In the reactive designs (e.g.,[58]), the response is generated only after receiving an alert from the detection system. However, in the proactive designs as in [59], the AIRS deploys a prediction mechanism to be able to respond and control any malicious activities before being detected.

2.3.7 Adjustment Ability

Depending on the adjustment ability, AIRSs are categorized into adaptive and non-adaptive systems. In the adaptive AIRSs [60], the effectiveness/goodness history of the responses is considered when choosing the optimal response. This helps the AIRS to develop an optimal ordered set of responses over time. On the other hand, non-adaptive AIRSs, such as the work in [61], do not have mechanisms for adjusting and learning from the success/failure history of the responses when being selected.

2.3.8 Response Execution

Depending on the response execution method, AIRSs are classified into burst and retroactive modes. In the burst mode [62], there is no mechanism to evaluate the effect of the response after being deployed on the system. This could be a huge risk because the IRS has no clue whether the deployed responses were enough to neutralize the attack or not, which could lead to deploying many limiting responses when a subset of them would have been enough to mitigate the attack. In contrast, the retroactive mode checks the effect of the deployed response and only executes further responses when the system needs them. As an example, the authors in [63] and [64] consider a retroactive approach.

2.3.9 Attack Modelling

Attack modelling approaches are visual diagrams that model the attacker's behavior in an IRS. They are divided into four main approaches, which are attack graphs, attack trees, multilevel Bayesian networks, and service dependency graphs. The work in [65], [66], and [67] exemplifies how these techniques are used, which is usually in the risk assessment phase to obtain/calculate the different decision-making metrics, such as the attack damage. Unfortunately, despite their popularity, they do not have a standardized visual representation. Accordingly, the representation of their components changes with the system description. Table 3 summarizes the analyzed modelling techniques by highlighting their advantages and disadvantages. For more detailed information on the different modelling formalisms, the reader is advised to refer to [68], [69], [70], and [71].

Table 3. Summary of the Attack Modelling Approaches

Modelling Approach	**Description**	**Advantages**	**Disadvantages**
Attack graphs (AGs)	Show the possible attack paths that an attacker can use to reach a target. Vertices represent states and edges represent transitions among the states	-Help in measuring the system's risk -Can represent multiple attacks -Some tools are available to generate up-to 1000-node graphs automatically	-Not suitable for big networks -Not suitable for zero-day attack modelling
Attack trees (ATs)	Show the system states and how they can be attacked. Intermediate nodes are connected with AND/OR gate to create multiple paths to reach the attacker's goal (root node)	-Simple to use -Can be combined with other models -Includes some descriptive attributes, such as the attack likelihood, the cost of exploitation, the time needed to exploit, and the required knowledge. -Have other extensions to enable more features	-Challenging to map attacks to defenses -Identify high level details, not specific attack-related ones -Require constant updating and maintenance -Limited as they can only represent a single attack -Very complex to be used for large systems
Multilevel Bayesian Networks (MLBNs)	Show propagation of cyberattacks with exploit probabilities on the edges	-Modeling uncertainties -It is used to assess the risk caused by attacks	-Same limitations of the attack graphs -Require experts to assign the probabilities
Service Dependency Graphs (SDGs)	Show the dependency relationship between the different services in a system	-Show the functional dependencies between the different services -Reveal how attacking a service can affect other dependent one in terms of the CIA aspects	-Very complex for large systems -Require full deep knowledge of the system and the available services -Need experts to define the dependencies

2.3.10 Decision-making Models

An intrusion response decision-making model is used to formulate the selection problem of the optimal countermeasure action. The selection problem is usually based on

the current system state, a set of actions, a reward function that finds a trade-off among different decision-making metrics. For an IRS, the environment can be modeled using single-objective or multi-objective optimization functions, Markov decision processes, and game theory models [72]. Besides these conventional methods, reinforcement learning-based approaches, especially model-free ones, have been recently explored to design IRSs for unknown or known environments. More descriptive details on each approach are given in Chapter 3.

2.3.11 Optimization Problem Solution Approaches

There are several techniques to solve the response selection problems based on the considered modelling approach. For the single/multi-objective optimization functions, Pareto optimality is considered one of the most popular solution methods. For game theoretic problem models, Nash equilibrium, Genetic algorithms, and linear programming are some of the solution approaches that are used [73]. Recently, the usage of reinforcement learning algorithms has been invading the IRSs field. It is different from the conventional solution methods since it generalizes much better, usually has higher accuracy, and most importantly, can work without a system model

CHAPTER 3: OVERVIEW ON CONVENTIONAL OPTIMIZATION AND REINFORCEMENT LEARNING DECISION-MAKING APPROACHES

In this chapter, we start by describing the IRS decision-making problem that we aim to solve. Then we provide a fundamental overview on the three conventional decision-making models utilized in the literature. Also, this chapter describes the RL framework, DRL basics, RL taxonomy, and ends with a comparison between conventional optimization methods and DRL approaches in solving decision-making problems. The aim of this overview is to help the reader to understand the basic theories behind the solutions reviewed in Chapter 4.

3.1 Decision-making Problem Description

At first, let's present and visualize the decision-making problem for the IRS that we are aiming to solve by considering different modelling approaches. Figure 6 shows the main parts of an IRS, which are the monitored CPS environment and the defensive agent. A cyber physical system contains the physical plant and the control system, which communicate together through the communication network layer using open-standard protocols such as Modbus. The plant sends sensor-measured states, while the control system sends back the appropriate actuator actions. The Attacker is an outsider or sometimes insider entity who gains rewards by threatening the security of the CPS by either exploiting vulnerabilities in the system design or well-known ones in the used communication protocols. The IDS comes into the picture when an attack happens. It receives all the observations coming from the CPS in vector X[n] to monitor the system's activity. Those observations are the states of the system, which include both cyber-level data and process-level data. Once a malicious activity is encountered by the IDS, it generates an alert that triggers the activation of the IRS.

The IRS agent receives a threat detection and alert generation vector A[n] from the IDS, including IDS-related data, such as the confidence level, attack severity, and attack type. The IRS optimizer block, which is the algorithm used to find the optimal solution, uses these values to guide the risk assessment procedure embedded in it. Simultaneously, the agent receives the states vector X[n] that represents the attacked CPS environment. The IRS policy, which is the focus of our thesis, is responsible for selecting optimal actions for each of the

received states. After deploying the action, the environment returns a reward that indicates the goodness of the deployed action. The state, chosen action, and received reward are sent to the IRS optimizer to guide its learning procedure. Accordingly, the IRS optimizer updates the policy by either encouraging the same action or discouraging it in future similar scenarios. Following that, the environment moves to a new state and the same procedures are repeated until the agent is well-trained.

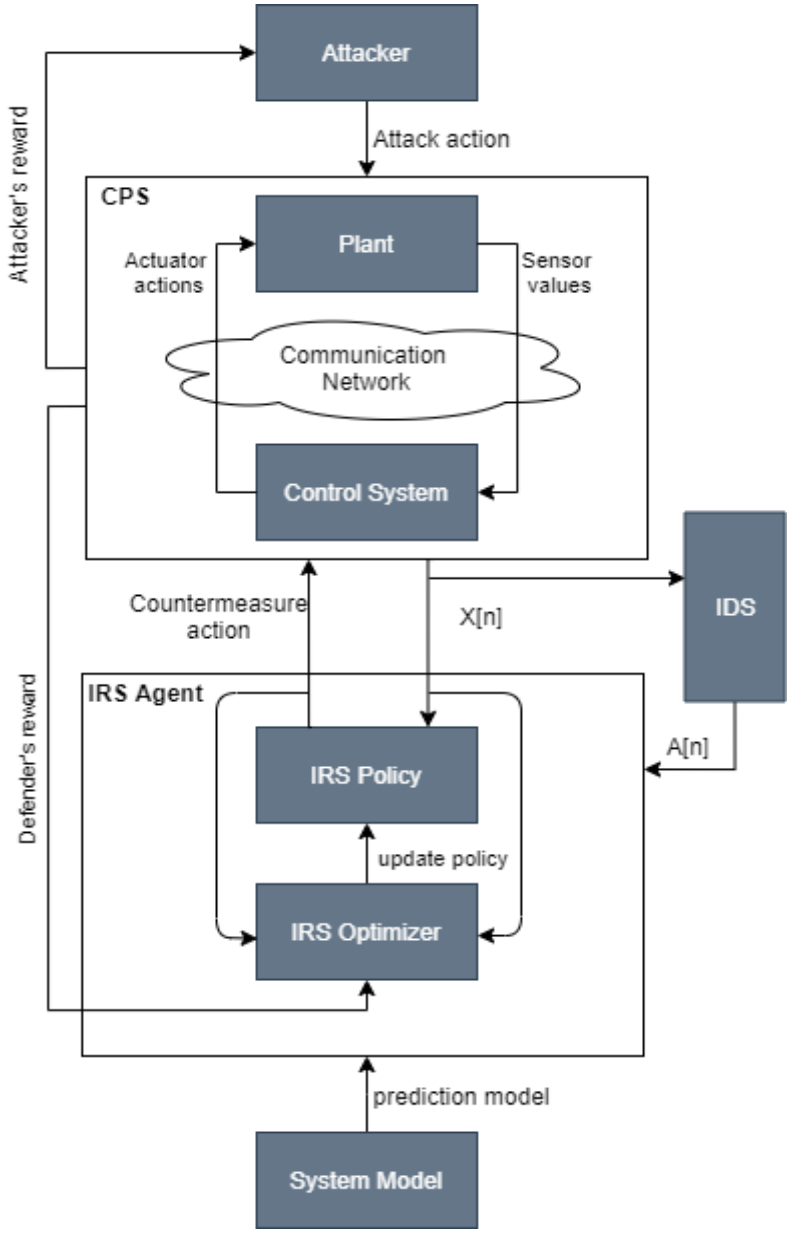

Figure 6. Decision-making problem architecture for intrusion response systems (X[n] includes plant and cyber data, A[n] includes IDS-related data)

The modelling part, which is the focus of this chapter, comes in the system model block at the bottom of Figure 6. When your CPS environment is known, a prediction system model helps the IRS optimizer to find the optimal policy for different scenarios. This prediction model gives the agent information about promising actions, future states, and rewards. The system model block could include the attacker model, the defender model, or both. Modelling the attacker behaviour can be done using the techniques mentioned earlier in Table 3. Attacker modelling helps in assessing the risk on the system and knowing the best positions for defense points. This is done by showing the propagation of the attack, the system vulnerabilities, and the possible attack paths with their likelihoods to reach different targets. For defender modelling, the multi-objective optimization functions, Markov decision process, and game theory, are the most widely used conventional decision-making methods. Moreover, Model-based reinforcement learning have been recently used in intrusion response system decision-making problems. They model the relationship between a defender agent and its environment by capturing the dynamics of the environment and defining the defender's objectives in terms of states, actions, rewards, and transition probabilities. In addition, game theory models the interaction between the attacker and the defender in a game format until it can reach Nash equilibrium.

Ultimately, solving this decision-making problem aims to present a more resilient CPS against advanced cyberattacks. The presented approaches are divided to conventional methods and reinforcement learning methods as summarized below:

3.2 Overview on Conventional Optimization Methods

3.2.1 Multi-objective Optimization Functions

Real-world optimization problems are naturally complex problems that require several conflicting objectives to be optimized at the same time. Therefore, single-objective functions are not applicable in most cases. Real-life problems are modeled as multi-objective optimization problems (MOOPs) if having up to 3 objective functions and many-objective optimization problems (MaOPs) if they are more than 3. An optimization problem is defined to get the minimum or maximum of functions with some specified

constraints. Accordingly, a MOOP can be defined as follows:

$$min/max\ F(x) = (f_1(x), \dots, f_m(x)) \quad subject\ to\ x \in X \qquad \text{Equation 1}$$

Where $F(x)$ is the vector of objective functions, m is the number of objective functions considered, and X is the set of feasible decision vectors.

To solve a MOOP with m objective functions, trade-off solutions are required to satisfy the different competing objectives. Pareto optimality is one of the widely used approaches to solve a MOOP. It provides Pareto Optimal Solutions (POSs), which are the solutions that are not dominated by any other solution in a solution set as they are all considered equally good. This happens when one objective function cannot be improved further without worsening another objective function.

The algorithms that are used to find the POSs are categorized into several groups, such as evolution and swarm intelligence, as presented in [74]. One of the widely used group of algorithms that proved their effectiveness in solving MOOPs are the Multi-Objective Evolutionary algorithms (MOEAs) [75]. For example, the Non-dominated Sorting Genetic algorithms (NSGA-II [76] and NSGA-III) have been exhaustively used in the literature because of their promising ability in generating a set of well-converged, diversified, and non-dominated POSs to the conflicting objectives in a single run. After finding the POSs, each algorithm considers different selection technique to choose the most optimal solution from the POSs. Based on the different selection techniques, algorithms are classified into Dominance-based algorithms, Decomposition-based algorithms, and Indicator-based algorithms. These techniques are extensively explained and compared in [77].

It is important to note that no one algorithm fits all optimization problems. The performance of the algorithms depends heavily on the considered objective functions. Consequently, the suitable choice of algorithms based on the problem at hand is very important since it significantly affects the required computational time and the number of iterations needed for convergence. Readers may refer to [78], [79], and [80] for more details on the usage of multi-objective optimization functions in decision-making.

3.2.2 Markov Decision Processes

A Markov Decision Process (MDP) is an optimization approach that models decision-making under uncertainty of an agent interacting with an environment. In comparison with Markov chains, MDPs extend them by adding actions and rewards. Also, MDPs satisfy the Markovian property, which ensures that taking an action only depends on the current state.

MDPs are defined with the following tuple $< S, A, P, R, \gamma >$, where S is the state space ($s \in S$), A is the action space ($a \in A$), P is the transition probability that represents the probability of changing the environment from state s to a new state s' based on the performed action a on state s $p(s, s') = p(s_{t+1} = s' \,|s_t = s, a_t = a)$, R is the immediate reward of choosing action a in state s, and γ is the discount factor that controls the importance of future rewards in a way that the agent cares more about present reward than future ones as γ increases ($\gamma = [0,1)$).

Solving a MDP means to find the optimal policy, which is the optimal mapping between states and actions, that maximizes the cumulative discounted reward function. Different approaches can be used to solve a MDP, such as dynamic programming, linear programming, and approximation methods. However, these approaches can only be used if the transition probabilities T are known. Unfortunately, in real world problems, the transition probabilities T and the rewards R are usually unknown. Accordingly, reinforcement learning algorithms, which are applicable for both model-based and model-free problems, have been recently proposed to solve the incomplete MDP and find optimal policies with inaccurate or no knowledge of the system model. For more information about the different solving algorithms, the reader is referred to the work in [81].

There are several extension variants of MDPs, such as the competitive Markov decision process (CMDP), the constrained MDP, and the partially observable Markov decision process (POMDP). More details on the variants are available in [82].

3.2.3 Game Theory

Game theory is a mathematical approach to model the interaction between rational decision-makers in different situations [83]. A game is described by four basic elements:

- Players: Individuals or entities who make decisions. In cyber security, players are typically the attacker and the IRS.
- Actions: Decisions made by each player based on the current state of the game. For cyber attackers, actions could represent attack steps such as password crack, or machine compromise. For the IRS, actions could include specific IP address blocking, applying a security patch, or shutting down a compromised machine.
- Payoff: The return that each player gets after taking the chosen sequence of actions. It could be positive or negative, depending on whether the return is a gain or a loss.
- Strategies: The guide to the actions that are taken by each player based on the available information about the game. The strategy could be pure, in which the player takes the same action for the same given information, or mixed, where the player picks an action randomly according to a pre-defined probability distribution over a finite set of actions.

Equilibrium is one of the key concepts in game theory. An equilibrium is a set of strategies, one strategy for each player, such that none of the players can improve their payoff by unilaterally deviating from the selected strategy. Figure 7 illustrates the different game classifications summarized below:

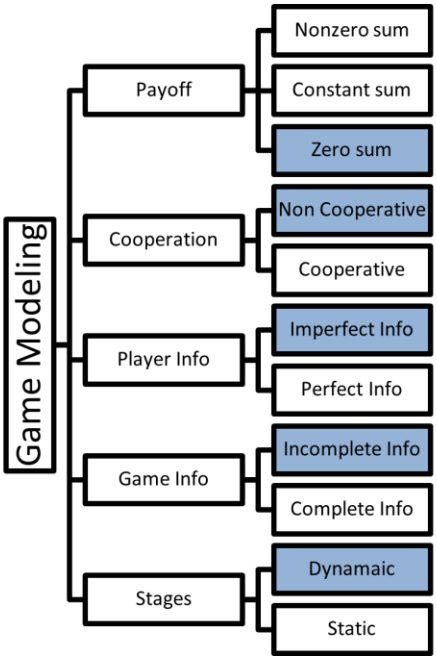

Figure 7. Classification of game theory modeling. Game theory modeling for cyber security is highlighted in blue color

- Number of Game Stages: A static/strategic game is a one-move game where players take their action at the same time. A dynamic game is a multi-step game where the number of steps could be finite or infinite.
- Game Structure Information: In a complete information game, all players know the rules of the game and the payoff for each player. Otherwise, the game is an incomplete information game.
- Players' Moves Information: In a perfect information game, each player knows all the previous actions of all other players. Otherwise, the game is considered an Imperfect information game.
- Cooperation: In Cooperative games, players may benefit from forming coalitions, but there could still be an underlying competition. Cooperative games typically have more than two players. Non cooperative games have no cooperation between players.
- Payoff: In a Zero-sum game, the sum of all outcomes adds up to zero. For a 2-player game, the gain of one player is the loss of the other player. In a Constant

sum game, the payoff outcomes add up to a constant value. In a Nonzero sum game, the outcomes add up to different amounts.

Game theory is used in cyber security in two key applications, defense strategy design and risk assessment. In defense strategy, the attack-defense interaction is modeled, and the optimal defense strategy is estimated. In risk assessment, the security state of the system is assessed using the predicted strategy for both the attacker and defender. Games in cyber security studies are typically classified as dynamic, incomplete information, imperfect information, non-cooperative, zero-sum games. More details on the different classifications of games can be found in [84], [85], and [86].

It should be highlighted that game theory assumes players are rational, i.e., each player takes the best decision that maximizes the benefit. This may not always be true in practice. Human behavior is not perfectly rational and may be influenced by many factors beyond the analyst's recognition. Albeit it represents a formal modeling approach to describe the interaction between the attacker and defense mechanism.

3.3 Overview on Reinforcement Learning Methods

Reinforcement Learning (RL) is a branch of machine learning, alongside supervised and unsupervised learning. Figure 8 shows a simple taxonomy of the three main machine learning algorithms. Supervised learning uses labelled data to learn a functional mapping between the inputs and their corresponding desired outputs. Unsupervised learning uses unlabeled data to learn similarities and discover patters from the input data. Reinforcement learning interacts with an environment to learn a series of optimal actions instead of relying on complex mathematical models. One of the core features of RL is its applicability in the decision-making field in order to reach a specific objective

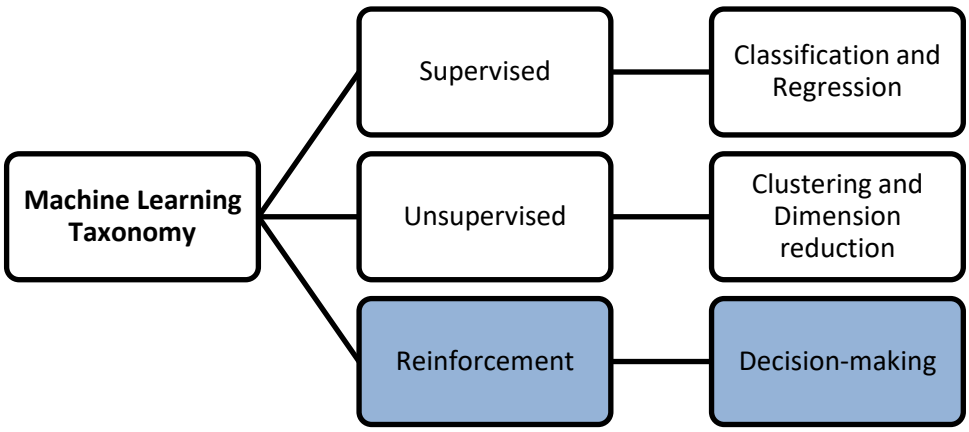

Figure 8. Machine learning taxonomy. The thesis's focus is highlighted in blue

3.3.1 Reinforcement Learning (RL) Framework

The standard framework of RL consists of an environment and an agent, as shown in Figure 9. The environment, which is everything outside the agent, is typically formulated as a MDP that is defined by a 5-tuple $< S, A, P, R, \gamma >$. The state S and action A spaces, which define the dynamics of the environment, can be discrete or continuous based on the considered problem. At each time step t, the agent receives the current states of the environment. These states should provide all the required information to describe the dynamics of the environment. Based on these states, the agent directly interacts with the environment by taking an action. As a result of this action, the environment moves to a new state at time $t + 1$, and the agent gets a scalar immediate reward/penalty value as feedback on performance. From the reward value, the agent can assess how good or bad the action was and consequently, how far is the agent from achieving the goal. The main objective of RL is to learn an optimal policy, which is the strategy that the agent follows to take an optimal action given the observed environment's state, that aims to maximize the cumulative rewards over time. It is worth mentioning that a complete interaction of having a state, action, and reward is called a step. While a series of decision steps are called an episode. At the end of each episode, or when reaching a terminal state, the environment resets to a random state to ensure that the agent explores the entire state space and does not overfit.

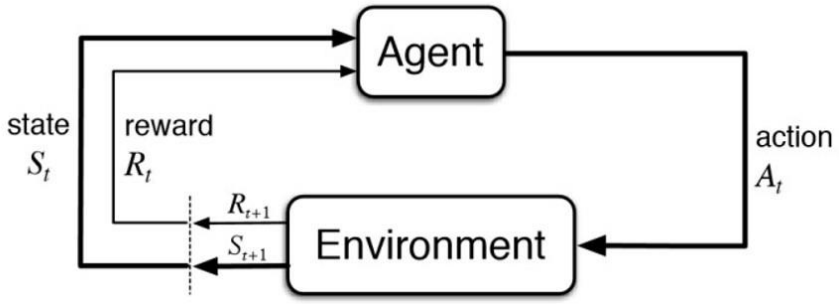

Figure 9. The standard framework of RL

3.3.2 Deep Reinforcement Learning (DRL)

DRL has the exact same framework as shown in Figure 9, but with integrating neural networks to work as the brain of the agent instead of using traditional tabular RL solvers, such as Q-Learning and Value Iteration (VI). A neural network works as a function approximator that learns how to map between the input states and the desired output values and hence improves the applicability of the algorithms. The usage of Deep Neural Networks (DNNs) overcomes several limitations that classical RL approaches suffer from in real-world implementations. For instance, DRL only stores the network parameters instead of all state-action values, so less memory demand is needed in comparison to tabular methods. Also, DRL handles the curse of dimensionality problem that tabular RL approaches suffer from, especially when dealing with large-scale systems that have large state and action spaces. Furthermore, it can efficiently and effectively handle complex tasks with no or less knowledge of the system. It also provides better generalization with unseen states and hence better performance. The ability of DRL to overcome these limitations and address challenging problems has made it popular in different fields including games [87][88], robotics [89][90], cybersecurity [91][92], and autonomous systems [93][94].

3.3.3 Reinforcement Learning Taxonomy

Figure 10 presents the main categorizations of reinforcement learning algorithms. Depending on whether the environment is known to the agent or not, RL algorithms are

classified into model-free algorithms and model-based algorithms [95]. Model-free algorithms do not require a model of the environment for solving a problem. They can learn an optimal policy and achieve near-optimal results using trial and error technique. Consequently, it is used when the state transition probabilities and the rewards are unknown. On the other hand, model-based algorithms use the transition probabilities and the known rewards, which represent a model of knowledge for the system, to derive the optimal best action policy. It is worth noting that having a model for the environment does not always mean that the agent is model-based. The agent is said to be model-based when the transition probabilities and the reward matrix of the state-action pairs are known or learned, otherwise, it is not.

Model-free algorithms are classified into value-based, policy-based, and Actor-critic algorithms. Value-based algorithms compute the value, which is also called Q-value, of an action given a state and do not learn an explicit policy since they map state-action pairs to values. These Q-values are the quality state-action value function, which indicate the goodness of a particular action in a state and whether this action should be reinforced in similar situations or not. On the other hand, policy-based algorithms do not need a value function, but an explicit policy with the best optimal parameters is constructed that maps states to actions. Actor-critic algorithms maintain both since they learn both a policy (actor) and an action-value function (critic) to evaluate the learned policy.

The model-free value-based algorithms are further classified into Off-policy algorithms and On-policy algorithms. They differ in the way they update their Q-values in which off-policy algorithms update them using the Q-value of the next state and the greedy best action. Also, off-policy approaches learn a policy π from using experiences sampled from another policy. while on-policy algorithms use the Q-value of the next state and the current policy's action, and they learn a policy π from experiences sampled from the same policy. Table 4 mentions the advantages and disadvantages of some of the widely used reinforcement learning algorithms from each category. For more information on Reinforcement learning framework and taxonomy, refer to [96], [97], and [98].

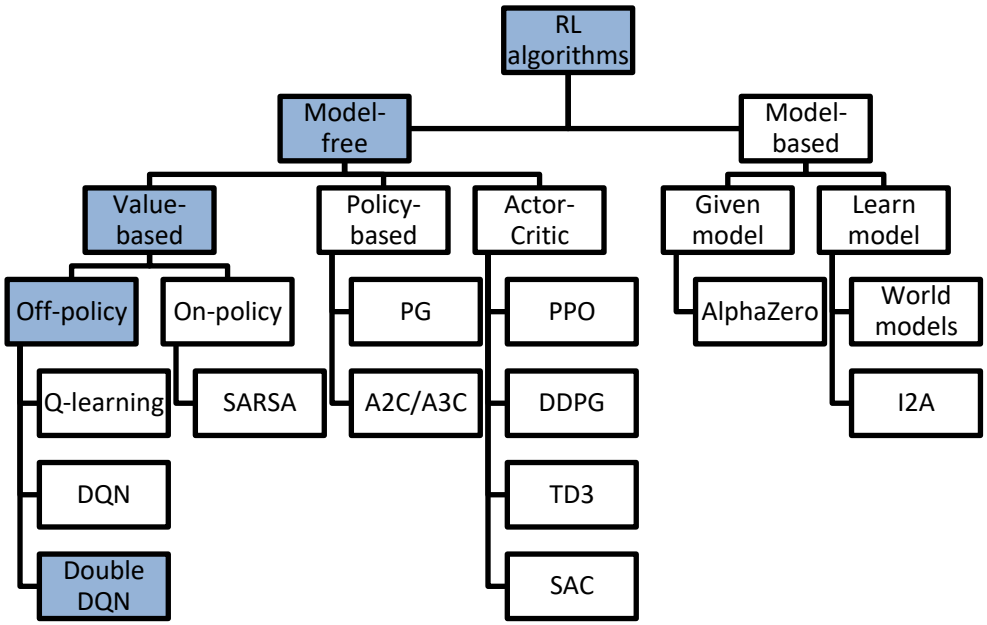

Figure 10. Reinforcement learning taxonomy. This thesis uses the algorithm path highlighted in blue

Table 4. Advantages and Disadvantages of Widely Used Model-free RL Algorithms

Algorithm	**Policy**	**Type**	**Advantages**	**Disadvantages**
Q-Learning	Off-policy	Value-based	-Simple -More freedom for exploration -Can select actions without MDP	-Not very stable -Not generalizable -Unable to deal with large spaces -Overestimation of the value function
Deep Q Network (DQN)	Off-policy	Value-based	-Good for large spaces -Generalizes to unseen states	-Unstable performance -Suffers from maximum bias issue -Takes long time to converge
Double Deep Q Network (DDQN)	Off-policy	Value-based	-Handles the issue of overestimation bias of Q-values in DQN -Uses two networks for decoupling between choosing and evaluating action -Faster and more stable than DQN	-Slow change in policy that can make the two networks too similar to make independent decisions

Algorithm	Policy	Type	Advantages	Disadvantages
Dueling Deep Q Network	Off-policy	Value-based	-Decomposes the Q-value as the sum of the state values and advantage function of taking that action	-Naively adding the two decomposed values can be problematic
SARSA	On-policy	Value-based	-Avoids high risk (conservative)	-Cannot use old data (no replay memory)
Policy Gradient (PG)	On-policy	Policy-based	-Effective in high dimensional spaces -Can learn stochastic policies	-Usually converges to a local optimum -Slow convergence
Deep Deterministic Policy Gradient (DDPG)	Off-policy	Actor-Critic	-Directly finds the policy with the most rewards -works well with continuous actions	-Sensitive to hyperparameters -Poor sample efficiency -Slow learning rate
Proximal Policy Optimization (PPO)	On-policy	Actor-Critic	-Improves samples efficiency	-Unstable during the training process
Twin Delayed Deep Deterministic Policy Gradient (TD3)	Off-policy	Actor-Critic	-Reduces the overestimation bias seen in previous algorithms	-The learning process is time consuming
Soft Actor-Critic (SAC)	Off-policy	Actor-Critic	-Enables stability -Robustness to noise	-Brittle to the hyperparameter that controls exploration

3.4 Conventional Optimization Vs RL Approaches

There are several differences between conventional optimization and RL decision-making approaches. In this section, we mention some of these variations and highlight the advantages and disadvantages of each approach as shown in Table 5.

To begin, conventional optimization solutions rely on explicit knowledge of the system that is expressed in complex mathematical formulations. These expressions are referred to as the objective functions of the optimization problem. Usually, abstracting these functions is not an easy task especially for complicated large-scale real-world systems.

Alternatively, RL approaches do not require an accurate mathematical model. It can even work with model-free problems while relying on a reward function to evaluate the decision-making performance of the agent based on its interactions with the environment.

Another point is that most conventional optimization techniques do not achieve the real-time decision-making requirement, which is crucial in several scenarios. Also, other traditional programming approaches such as dynamic and stochastic programming tend to have high computational costs, and their effectiveness is not assured for large complicated systems. On the contrary, reinforcement learning approaches are more robust, can handle higher-dimensional complex scenarios, and provide real-time decisions since they make decisions based on the current state of the system.

Although reinforcement learning approaches tend to be more promising solutions that can provide scalable, generalized, online, robust, and efficient performance, they also have some major challenges. To start with, they provide slower convergence in comparison to the conventional optimization techniques. Also, tuning the hyperparameters is a very time-consuming procedure that can highly affect the performance of the agent. Designing a representative reward function that can effectively describe the goal of the agent and deciding on a reasonable size and type for the state and action spaces are other factors that highly affects the computational time of the RL agent.

Table 5. Advantages and Disadvantages of Decision-making Modelling Approaches

Decision-making Modelling Approach	**Advantages**	**Disadvantages**
Multi-objective optimization functions	- Easy to formulate -Can find Pareto solutions in a single run	-Mostly suffer from high computational cost -Do not guarantee an optimal solution
Markov Decision Processes	-Help defenders to know what paths are more probable to be followed by an attacker	-Do not scale well -Need other attack modelling techniques to assign probabilities to the paths that an attacker can take
Game Theory	-Simple	-The definition of the payoff

Decision-making Modelling Approach	Advantages	Disadvantages
	-Can be deployed in a distributed manner	function is very challenging -Lacks scalability and robustness -It is a planning technique (game matrix should be already known)
Reinforcement Learning (RL)	-It is a learning not a planning problem -Can generalize to unseen states -Can handle continuous spaces -Can handle large environments -Can resist biased datasets when using online learning -Uses samples to optimize the performance -Can work with model-free systems	-Needs huge number of simulation trials for training -Designing good representative reward function is challenging -Requires tuning of numerous hyperparameters -Provides no performance guarantees at all

CHAPTER 4: SURVEY OF WORKS ON INTRUSION RESPONSE SYSTEMS

This chapter presents a survey of the current major works on intrusion response systems. Initially, we aimed to target only the work intended for CPSs. However, we found that the work in this area is still in its infancy, so we considered reviewing IRSs for both CPS and non-CPS domains to enrich our knowledge on how to approach the area of securing CPSs and bridge the gap between security in ITS and CPS domains. The surveyed papers are classified into two main categories based on whether the decision-making approach is done using conventional optimization methods, including multi-objective optimization functions and game theory, or is done using reinforcement learning approaches. It is worth mentioning that we prepared a comprehensive survey paper on intrusion response systems and submitted it to the Journal of Network and Computer Applications.

4.1 Conventional Approaches for IRSs

Conventional optimization approaches depend on having a mathematical model consisting of objective functions, constraints, and different decision variables that the agent has to optimize. This section includes research works using multi-objective optimization functions and game theory solutions to build IRSs. For easy reference, a summary of the main points of the reviewed works is available in Table 6.

4.1.1 Multi-objective Optimization Functions Solutions

Optimizing objective functions to find optimal policies is one of the conventional approaches that is used when designing intrusion response systems.

In [99], the authors design an intrusion response system that can dynamically select the optimal countermeasure based on a trade-off between the attack damage and the countermeasure cost. The problem is treated as a multi-objective optimization problem (MOOP) with four metrics to consider when choosing the optimal response, which are attack damage cost, countermeasure positive effect, countermeasure negative impact, and countermeasure deployment cost. The main aim is to choose the best countermeasure that can maximize its positive effect, minimize its negative impact, and minimize its deployment cost. The proposed framework uses Attack-Defense Trees (ADTs), which is

an attack modelling approach that shows the attacker's progress and possible paths to the target. Also, the framework uses the Service Dependency Graphs (SDGs), which shows the dependencies between the different services and measures the severity of the undergoing attack. A Pareto optimal solution set is initially created with the possible countermeasures that satisfy the optimization trade-off. Then, a simple additive weighting (SAW) method is used to select the optimal countermeasure, which holds the maximum final score value, from this set. The evaluation of the proposed solution is done on a real cloud environment that consists of 6 virtual machines connected by 4 switches and 15 vulnerabilities in total. The considered countermeasures are blocking IP addresses, blocking ports, blocking all traffic, restarting the service, closing connection, disabling features, patching, and shutdown. The authors conclude that this framework dynamically evaluates countermeasures, selects optimal ones, and deploys them in 449 milliseconds. This is considered very fast, which is one of the main advantages of this approach. One drawback of this approach is that it uses only one attack tree for a service, which is not the case in real-life scenarios where a forest of trees is usually used to protect system assets, so it lacks scalability. Finally, although the proposed solution is not intended for CPSs, it is applicable for their usage.

A dynamic decision-making approach for intrusion response for securing industrial control systems at run-time is proposed in [100]. The decision-making problem is formulated as a MOOP to maximize security benefit ($SecB$), system benefit ($SysB$), and state benefit ($StaB$). The architecture of the proposed model takes the attack evidence and abnormal evidence as inputs from the intrusion detection system. Then, these inputs are mapped into a multilevel Bayesian Network (BN) to extract all the possible attack paths to the target and hence study the attack propagation from the cyber domain to the physical domain. A security measure set is developed after considering both the defense and recovery measures. From this security measure set, a candidate security strategy space is built with 2^n strategies where n corresponds to the number of security measures (countermeasure actions). Using this security strategy space, several Pareto optimal

solutions are found by solving the MOOP using the Non-dominated Sorting Genetic Algorithm-II (NSGA-II). Finally, the optimal security strategy solution is selected based on a distance-based evaluation method. The proposed approach has experimented with a simplified Tennessee Eastman chemical process control system to assess its effectiveness. Two attack scenarios are simulated using MATLAB environment in which the first scenario assumes that the launched attacks do not compromise the physical control system, while the second scenario assumes that they do. The considered defense measures include shutdown servers, blocking traffics, limiting incorrect password attempts, closing connections, restarting, and encrypting messages. In comparison with other approaches, this proposed solution is of an advantage in terms of considering security measures covering both the cyber and the physical domains. Also, it is one of the very few approaches that considers not only defense measures, but also recovery ones. On the other hand, the scalability and the time complexity of this approach concern the authors as they highly depend on the scale of the constructed BN and the size of the strategy space.

The work in [101] proposes a general real-time control approach for designing IRS for industrial cyber-physical systems. Unlike most works that focus on the response policy selection decisions, this paper focuses on the security policy execution based on a given real-time security policy input. The main aim is to ensure that the system does not have massive losses while enforcing corrective mitigation responses. Initially, a MOOP is defined to maximize security protection time, minimize communication load, and minimize execution time of responses. Their solution approach is based on table-driven scheduling of responses using NSGA-II algorithm, which is used to solve the MOOP and find the Pareto solution set. Then, the responses in the Pareto solution set are ranked to select the one with the highest security protection time, which is the highest priority, as the optimal response. A directed acyclic graph (DAG), which models the dependency between the different tasks, is used to map the system and response tasks and schedule the tables on each node. Each system node has a scheduling table that is periodically updated with response tasks to be performed to protect against ongoing cyberattacks. The

proposed framework is evaluated on a simulated simplified TEP control system based on OPEN and MATLAB. The results show that the execution of the responses is done effectively without negatively affecting the system tasks with a fast execution time of less than 60 ms, which is about 10 generations. The advantage of this approach is that it addresses the response execution scheduling perspective that is neglected by many researchers when designing IRSs. However, they assumed some values without clear justifications, such as indicating that the communication load and the execution time of a task/response is statically defined in advance for all tasks.

The authors in [102] formulate the countermeasure decision-making as a single objective optimization problem. The attack damage, deployment cost, negative impact on the QoS, and security benefit are the four metrics considered in the optimal policy selection process. To solve the optimization function, they propose a Genetic algorithm with three-dimensional encoding (GATE). Unlike most works, this work not only selects the optimal countermeasure, but also decides on where it should be deployed, in what order, and for how long. Determining these fine-grained decisions influences the effectiveness of the selected responses. For evaluation, an experimental network of servers is used to validate the effectiveness of the proposed approach. The considered countermeasures are blocking traffic, blocking ports, and closing connections. The results show that the framework can effectively generate an appropriate reasonable response policy to the detected attacks. In addition, three different algorithms are compared with the proposed one, which are the traverse algorithm (TA), the random algorithm (RA), and the simulated annealing algorithm (SAA). The comparison reveals the superiority of the GATE algorithm in terms of the fitness value and computational time. The advantage of this framework is emphasizing the importance of considering the fine-grained details when designing an intrusion response system. On the other hand, this approach does not consider modeling the attacker. We believe that this approach is not very suitable for CPSs because they are usually more complex for a single objective representation. Also, some of the taken assumptions, such as not allowing multiple countermeasures on the

same defense point, are not suitable for a CPS, which usually requires a composite of actions to mitigate the attack effect that is usually seen at multiple layers.

4.1.2 Game Theory Solutions

Recent studies present clearly that game theory is increasingly used in building intrusion response systems.

In [103], the authors propose a cyber-physical response system (CPR) that can automatically mitigate cyber-originated attacks causing physical consequences against a power grid critical infrastructure. The proposed framework considers both cyber-side and physical-side response actions when mitigating the detected attacks. The interaction between the attacker and the CPR system is modeled using a sequential Stackelberg stochastic game approach with two players in which the CPR system is the leading player, and the attacker is the follower player. Both players aim to maximize their benefit with the help of a reward function. This game is represented as a competitive Markov decision process (CMDP). The states, which are obtained from both cyber intrusion detector (anti-virus) and power system sensors, represent the compromised cyber host and the physical consequences on the process. The CMDP is solved using the value iteration algorithm and the infinite-horizon discounted cost method to obtain the optimal response that maximizes the accumulative long-run reward. A case study is presented to evaluate the proposed framework in a publicly available power system that contains seven generator controllers. Four different attack scenarios are considered in which each time certain generator controllers are compromised. The considered countermeasures are disabling the compromised generator, dispatching other working generators, killing the malicious process on the controller from the cyber-side to restore its normal operations, and load shedding. The results show that in most scenarios, using both the cyber-side and physical-side responses is required to stop the attack since their capabilities complement each other. One of the advantages of this paper is that the solution approach is not too system specific as it can be applied to different network models. However, the usage of a very limited list of countermeasure actions is a common downside.

In [104], the authors propose a multi-layer defense architecture to protect the main layers of critical infrastructures from different cyberattacks. The considered layers to be protected are the management layer, supervision layer, real-time control layer, and physical layer. A multi-model anomaly-based intrusion detection approach is used to detect the existence of intrusion behaviors. An analytic hierarchy process (AHP) approach is used to assess the impact and the risk of ongoing attacks on the process, considering asset identification and classification, asset quantization, and dynamic assessment. A hierarchical, dynamic, two-player, non-cooperative, finite game is then proposed to select the optimal protection strategy in a two-step approach, which is building an attack defense tree (ADTrees) model and building a security performance game (SPG) model. Finally, the real-time intrusion response interprets the selected security mechanism and applies the changes on the required nodes. For evaluation, a networked water level control system is considered to test the effectiveness of the proposed framework. Different attack scenarios are simulated, such as setpoint change attack at the supervision layer, response injection at the real-time control layer, and command injection at the physical layer. The considered countermeasures are access control list, self-reconfiguration, self-updating, and activating safety guards to take over the compromised controller. The results show the effectiveness of the proposed defense approach in mitigating the impacts of these attacks. The advantage of this approach is its applicability to wide range of industrial CPSs. Nevertheless, one of the limitations of this approach is that the attacker's level of experience and the time taken for building the attacker model is not considered when evaluating the solution.

In [105], the authors designed a multi-step dynamic decision-making approach to obtain the optimal defense strategy for the detected attacks. There are two kinds of possible defense strategies considered, which are security strategies and recovery strategies. The security strategies reduce the risk of the attacks but invalidate some of the system functions in return. The recovery strategies are the ones that can recover failed system functions under an execution cost. The IDS inputs are the evidence list that

contains any attack occurrence or function invalidation and the current state of the system. The architecture of the proposed framework contains several modules, including a Multilayer Bayesian network (MLBN), process model, state controller, optimal defense strategy generator, and a strategy execution system. It uses a 2-player non-cooperative zero-sum game between the defender and the attacker. The game uses the Nash equilibrium theorem to find the solution and obtain the optimal defense strategy. This approach has been evaluated using a chemical reactor control system simulated in MATLAB. Different attacks are considered, such as network scanning, buffer overflow, vulnerability scanning, DoS attack, brute force attack, and reactor failure. The defensive actions are shutdown, disconnect, and encrypt. For the recovery actions, rebooting different PLCs are considered. The results show that the proposed framework effectively found the optimal defense strategy that minimizes the system loss within seconds. Moreover, the presence of a state controller model reduced the computational complexity by 87.5%. This paper has an advantage of showing a novel quantification approach for the risk assessment calculations to find the optimal attack-defense strategy using MLBN and system models. However, the computation time is still an open issue that requires further improvements to meet more strict real-time requirements of CPSs.

In [106], the authors propose an automated cost-sensitive response and recovery engine (RRE) to mitigate attacks in real-time. The response decision-making model uses a two-player, sequential, multi-step, non-zero sum, hierarchical, Stackelberg stochastic game strategy with attack-response trees (ARTs). The ARTs are designed offline by experts to describe the security state of each asset. The modeling of the game uses a partially observable competitive Markov decision process (POCMDP) that is automatically obtained from the ART. The solution of the game is the optimal network-level countermeasures, which have the minimum accumulative attack damage and the maximum accumulative long-run response reward. The architecture consists of two engines: the local engine resides in host computers, and the global engine, which is responsible for the security of the whole infrastructure, resides in the response and

recovery server. The local engines receive a set of assets that need to be protected, IDS alerts, and ART graphs as inputs. Then, it generates a finite set of security state spaces of all the safe states that the host computer can tolerate. The state space is sent to the decision-making unit that uses a game-theoretic approach with the value iteration method and maxmin strategy to find the optimal response. Then, the selected optimal responses are implemented by the RRE agent, which reports back whether the actions are accomplished successfully or not. The global engine comes into the picture to handle attacks if the local engines are compromised or unable to recover the system. The effectiveness of the proposed RRE engine is evaluated on several networks of different sizes. Results show that the RRE engine works very well in choosing the appropriate countermeasure action even for large-scale networks that have more than 500 nodes, which are still solvable in less than 40 seconds. The main contributed advantage of this approach is providing a scalable distributed solution that improves the performance of a response engine. However, it only uses cyber-level actions, includes some subjective static definitions of network properties, introduces trust issues between the different nodes, and lacks accurate system state insights because of the distributed nature.

Table 6. Summary of Reseach Works on IRSs Using Conventional Solutions

Decision-making Model	**Ref.**	**CPS**	**Attack Modelling Approach**	**Actions**	**Decision-making Metrics**	**Solving Approach**	**Evaluation**
Multi-objective optimization problems (MOOPs)	[99]	No	ADTs and SDG	Blocking IP addresses, blocking ports, blocking traffic, restarting services, closing connection, disabling features,	Attack damage cost, positive and negative impacts of actions, and deployment cost	SAW method to extract the optimal solution from the Pareto optimal set	Based on response selection and deployment time

Decision-making Model	Ref.	CPS	Attack Modelling Approach	Actions	Decision-making Metrics	Solving Approach	Evaluation
				patching, and shutdown			
	[100]	Yes	MLBN	Shutdown servers, blocking traffic, limiting incorrect password attempts, closing connection, restarting, and encrypting	Security benefit, system benefit, and state benefit	NSGA-II algorithm with a distance-based method	Based on execution time and response goodness
	[101]	Yes	NA	Not mentioned	Protection time, execution time, and communication load	NSGA-II algorithm with a crowding distance method	Based on execution time
	[102]	No	NA	Blocking traffic, blocking ports, and closing connection	Attack damage, negative impact on the QoS, security benefit, and deployment cost	Genetic algorithm with three-dimensional encoding (GATE)	Based on fitness values and computational time
Game theory	[103]	Yes	NA	Disabling compromised generators, re-dispatching, and killing malicious processes	The accumulative long-run reward of responses	The value iteration algorithm with the infinite-horizon discounted cost	Based on decision-making convergence iterations

Decision-making Model	Ref.	CPS	Attack Modelling Approach	Actions	Decision-making Metrics	Solving Approach	Evaluation
	[104]	Yes	Attack defense trees	Access control lists, activating safety guards, and self-reconfiguration	The risk (impact) of ongoing attacks	An analytic hierarchy process (AHP)	Based on checking the state of some process-related values
	[105]	Yes	MLBN	Shutdown, disconnect, encrypt, and PLCs rebooting	Attack damage, system degradation, system stability, and state reachability	Nash equilibrium theorem	Based on computation time (computational complexity)
	[106]	No	Attack response trees	Not mentioned	Attack damage and accumulative long-run response reward	The value iteration algorithm with the infinite-horizon discounted cost and maximin approach	Based on handling scalability, and time needed to respond

4.2 Reinforcement Learning Approaches for IRSs

The application of reinforcement learning to the intrusion response field is relatively new, especially for CPSs. We divided the work in this area into model-based solutions and model-free solution. Table 7 shows a summary for each of the examined works in this section.

4.2.1 Model-based Solutions

In [107], the authors propose an intrusion response system using deep reinforcement learning that is modeled with a Markov Decision Process (MDP). They state that they are the first to use deep reinforcement learning (DRL) in designing an IRS. Table 7 shows the states, actions, and reward function of the considered MDP model. For solving the MDP

model, the authors propose a Deep Q-Learning (DQL) technique that uses a Convolution Neural Network as a non-linear function approximator. The considered system for evaluation is a stationary microservice-based system. Both Q-Learning and DQL algorithms have been used to estimate the action-value function to solve the formulated MDP model. Results show that DQL converges to an optimal solution faster than the tabular Q-learning algorithm in terms of time and number of episodes. In addition, it achieved noticeably less memory utilization and maximum cumulative reward. In comparison with conventional standard methods, which require an accurate model of the system for designing an IRS, the use of DRL reduces the needed time to find defense strategies and handles large-scale systems more effectively. On the other hand, DQL approach require tuning many hyperparameters, which can highly influence the agent behavior, to achieve the desired optimal rewards.

In [108], the authors use the MDP framework to design an IRS. The proposed design captures both the defender and the attacker models and plans optimal long-term responses to protect the system. Initially, a Single Agent MDP is used to model the IRS defender model such that the agent has an objective of finding an optimal policy with a maximized reward. In the defender model, the responses are evaluated based on their response time, operational cost, and impact index on the system. Some considered response actions are firewall activation, blocking source IP address, and generating an alert. After developing the defender model, the attacker model is added using a competitive multiple agent MDP that is implemented as a stochastic game. Each attack in the attacker model has three characteristics, which are attack belief, which is the probability that a specific attack will be executed by the attacker in the future, attack action, which is the future action to be taken by the attacker based on the dependencies between attacks, and the intrusion damage of the attack on the system. The MDP model is solved using the Value Iteration (VI) and the Upper Confidence Trees (UCT) algorithms. Considering a system with 1000 system attributes and 1000 possible response actions, the VI outperforms the UCT in all the experiments. Since the MDP model grows exponentially when used to describe large

systems, the authors designed a dynamic attributes and actions selection engine, which instantiates the MDP problem with the minimum number of attributes and actions that are only directly related to the detected attack. Also, the usage of the parallel version of the VI algorithm was considered to handle the scalability. As a consequence, the proposed IRS was able to generate the optimal response policy in less than 2 seconds. Advantages of the solution include modeling both the defender and the attacker and handling scalability. One disadvantage is considering statically defined transition probabilities instead of using a dynamic feedback loop between the protected system and the IRS.

The authors in [109] use the Q-learning algorithm to select the optimal strategy by optimizing the game model. They propose a two-player zero-sum stochastic game model to generate optimal defense strategies to mitigate highly organized cyberattacks in Industrial Cyber-Physical Systems (ICPSs). The attacker in this game is assumed to infiltrate from the corporate network and propagate until he/she reaches the physical process. The authors use the base-group metrics provided by the common vulnerability scoring system (CVSS) to quantify the probability of success of attacks. The state of the game is represented by a binary vector where a value of 1 means the device is compromised and a value of 0 means not. The defender actions are security responses, such as installing patches and restarting. The reward function depends on time-based quantification in a way that $R(A_{t,i})$ is the time needed to recover the compromised device after an attack, $T(D_{t,j})$ is the time needed to perform the defender action, and $C(A_{t,i})$ is the time needed to perform the attacker action. The proposed game model is evaluated on a simulated simplified Tennessee-Eastman (STE) process control system that uses Modbus communication protocol. Q-learning algorithm shows its effectiveness in solving the game model, indicating a fast convergence rate with higher learning rates. The proposed approach highlights the possibility of performing self-learning to derive the optimal defense strategy without accurate knowledge of the model parameters. Also, this approach considers both the cyber and physical layers when designing an IRS for ICPSs.

However, the assumption that the players have complete information of the actions of other players is unrealistic in real-world problems.

In [110], The authors model the security problem between the defender and the attacker in a CPS as imperfect information stochastic game that is solved using a Multi-Agent Reinforcement Learning (MARL) approach. The Q-Learning algorithm is applied to achieve the learning element and decide on the best actions, which increase the overall expected reward, to be taken by each player at run-time. The work aims to reach a Nash Equilibrium (NE) in favor of the defender system. The considered system states are the security status, which is either low, medium, high, or critical. The architecture of the modeled CPS, which is assumed to be known by the defender, composes of four layers to mimic a real vulnerable CPS network. The different considered types of vulnerabilities are taken from the Common Vulnerability Scoring System (CVSS) with their corresponding information, including vulnerability access, exploitability score, and CVSS score. The experimental section focuses on a virus spreading attack scenario that exploits zero-day vulnerabilities to reflect realistic assumption. The simulations are done using the MiniCPS simulator and OpenAI Gym for implementing the reinforcement learning algorithm. The results show that the proposed hybrid approach cannot stop the attack but can limit its success rate. Accordingly, the attackers succeed by 25% on attacking the cyber layer while by 94% on the physical layer. This shows a huge disadvantage since the proposed method is not very effective in protecting the physical layer, which is originally the main target of any attacker against CPSs.

In [111], the authors also use the Q-learning algorithm with a game model to find the optimal defender actions in a simulated power system environment. The interactions between the adversary and defender are modeled as a two-person zero-sum repeated game. This game considers several parameterized factors when calculating the reward function, including the attack and defense costs, allocated budgets, and the players' strengths. The Minimax Q-learning algorithm is used to solve the game and find the optimal action in favor of each player. The system simulation is performed using

MATLAB. The conducted experiments reveal the postattack effects on the system in terms of voltage violation that reflects the saved and lost power of its elements. The results show that this solution can defend the transmission lines, which are the prime targets of attackers, effectively and efficiently. The advantage of using a repeated game is that players can generate actions independent of the actions' history, and hence the game is close to real-life scenarios. Disadvantages include poor evaluation in terms of comparison with different RL algorithms.

4.2.2 Model-free Solutions

In [112], the authors extend their work in [107] to find the optimal intrusion response in a non-stationary microservice-based system. They use a model-free DQN algorithm for building their IRS. The considered approach consists of the following four phases: designing the system model, building a software simulator for the system, using a RL agent to learn the simulated system, and detaching the RL agent from the simulator to attach it to a real system. The considered states are five Boolean variables representing the current status of each component, such as active, updated, new version available, corrupted, and vulnerable. Seven actions are considered with their pre-conditions, post-conditions, execution time, and cost, including restart component, start firewall, and fix a vulnerability. For evaluation, the authors compared the effectiveness of the tabular Q-Learning agent and the DQL/DQN agent on both stationary and non-stationary systems. The considered evaluation metrics are steps of convergence, cumulative reward, and execution time. The results show that for stationary systems, DQL converges faster in terms of the number of episodes. Also, it exhibits a constant behavior with the increasing number of system attributes and linear memory utilization. For non-stationary systems, the results show that Q-learning converges faster than DQL only if a structural change to the neural network is needed. Otherwise, DQL is a better choice because it is memory-efficient and provides better generalization capabilities. The proposed solution also proved effectiveness when compared with a standard planning technique (VI). All in all, this approach is scalable, which can work with large systems. Also, it is one of the very

first papers that addresses non-stationary systems. However, like any learning-based approach, many hyperparameters need tuning, which is not a simple task to do. Moreover, the proposed IRS is reactive; since it only models the defender system and ignores the attacker model.

In [113], the authors use model-free reinforcement learning with the off-policy tabular Q-learning approach to decide on the optimal response policy in network security. The proposed model consists of two network states $\{S_N, S_A\}$, which correspond to the state of the network under normal conditions and the state when it is under attack, respectively. Two actions $\{a_p, a_{dn}\}$, were considered in which a_p is for a protection action and a_{dn} is for a do-nothing action. The model also includes a reward matrix R, which contains the immediate reward the agent will obtain by performing action a in state s. Estimating the transition probability matrix T, which shows how the environment will change from one state to another under the selected actions, is done using the maximum likelihood estimation (MLE) on bootstrapped data sequences and the Laplace smoothing approach. This transition probability estimation is not needed when using Q-learning, because it is a model-free algorithm. However, it is used for evaluating the model with other techniques that require knowledge of the environment. The used dataset is the ISCX NSL-KDD, which contains 42 variables and 24 different attack scenarios. The results show that after 100,000 iterations, the action-value matrix Q (S, A) for each action and state combination is computed. Once the action-value function is determined, the optimal policy is created by choosing the action with the maximum value in each state. For evaluation, several other techniques are used with the help of the estimated transition probabilities, which are Linear Programming (LP), Policy Iteration (PI), and Value Iteration (VI). The results show that Q-learning was able to obtain the same optimal policy as the other approaches, but with the advantage of not requiring environment knowledge. This emphasizes the value of using model-free reinforcement techniques in network security. However, the approach provides poor evaluation as it did not mention

the cumulative reward and execution time metrics.

In [114], the authors propose a model-free reinforcement approach using the Q-Learning algorithm to detect and respond to attacks in non-stationary systems by solving its MDP. Non-stationary systems present the concept of having a dynamic environment where an action can be added, removed, or changed over time. This dynamicity makes model-based approaches challenging to apply, because non-stationary systems do not always behave as modeled. The represented environment is a three-tier web application with a state space that consists of several variables representing if each component server is vulnerable, started, up to date, has a CPU load, or under attack. The agent considers several actions, such as starting, stopping, updating, and patching the attacked server. Three scenarios are used in the experimental section, which are adding new actions, changing the reward parameters of the actions, and removing actions. After training the agent for 200,000 learning episodes on a simulation system, the results show that the agent converges to the near-optimal solution and can adapt to any changes in the different experimental scenarios by obtaining a high cumulative reward. Advantages show that the proposed approach succeeds in capturing the dynamics of the changing environment and automating the defense against advanced attacks in a non-stationary system, which does not depend on a static model of the system. The disadvantages of the proposed approach include requiring more time to converge to the near-optimal solution and poor evaluation of the performance.

The authors in [115] propose a DRL-based approach to mitigate a different range of DDoS attacks in real-time, including TCP SYN, UDP, and ICMP flooding. Their framework leverages Software-Defined Networks (SDN) with OpenFlow standards to have a centralized global view of the network that helps with collecting the network statistics. They use the Deep Deterministic Policy Gradient (DDPG) algorithm, which is an actor-critic-based algorithm. The state-space contains eight features from the collected traffic, which are port number, number of received packets, number of transmitted packets, number of received bytes, number of transmitted bytes, number of packets of

each flow, number of bytes of each flow, and time the switch has been alive in nanoseconds. For the actions, the agent outputs a vector of size N (N is the number of hosts) where each value is between [0.1, 1] and represents the maximum bandwidth that is allowed by the host. In this way, the attacker traffic is throttled, and most of the resources are available to serve legitimate traffic. The considered reward function is -1 if the traffic load on a server is greater than a defined upper boundary $U_s = 6\ Mbps$ and is $\lambda p_b + (1-\lambda)(1-p_a)$, otherwise. The hyper-parameter λ weights the two parts of the reward function, p_b represents the percentage of legitimate traffic reaching the server, and p_a represents the percentage of the malicious attack traffic reaching the server. For evaluation, the proposed approach is compared with two popular state-of-the-art throttling methods, which are the AIMD router throttling and the CTL, on a sample SDN topology of virtual hosts and OpenFlow switches that is created using Mininet. The evaluation is done in different attack dynamics, such as constant rate attack, increasing rate attack, pulse attack, and group attack. The results show that the proposed agent outperforms them and can effectively mitigate DDoS flooding attacks of different protocols. The advantage of the SDN approach is that it decouples control and data aspects in the network and generalizes well with different unseen scenarios. On the other hand, the disadvantages include not addressing the scalability requirements and assuming that the sending rate of the attacker is significantly higher than the legitimate user, which is not always the case.

In [116], the authors use a multiagent router throttling decentralized approach using SARSA RL algorithm and the Coordinated Team Learning (CTL) design to defend DDoS attacks in network intrusions. The purpose of the RL agents is to rate-limit the traffic directed towards a victim when a DDoS attack happens in a scalable system. The state-space of each agent consists of four features corresponding to the traffic rates of four considered routers. The action space consists of 10 actions that correspond to 0% to 90% traffic drop probabilities from traffic directed to the victim server. Teams of agents use

the task decomposition approach and work as independent teams that receive rewards at the team level. Experiments are done on a network emulator testbed using tree network topologies consisting of homogeneous teams of agents. The training is done offline to obtain a policy that can be used later in evaluation. Different attack dynamics were considered for evaluating the proposed approach against other popular throttling approaches, including constant-rate attack, increasing-rate attack, pulse attack, and group attack. Results show that the proposed throttling approach outperforms the baseline and the popular AIMD router throttling technique. The proposed method was also evaluated with online training and showed promising results, but with more time needed for convergence. The same multi-agent approach was used in [117] to design an intrusion response system using Deep Q-Networks, but with a model-based mindset. The advantage of the proposed solution is that it focuses on addressing the scalability challenge using offline learning. Also, it is more resilient than centralized approaches because it does not have a single point of failure. However, this work does not consider attackers who send traffic at a rate similar to legitimate users.

Table 7. Summary of Research Works on IRSs using RL Solutions

Algorithm	**Ref.**	**CPS**	**Model**	**States**	**Actions**	**Reward**	**Evaluation**
DQN	[107]	No	Model-based	Variables to indicate if each component is active, updated, has a new version, corrupted, or vulnerable	Starting component, restarting component, starting firewall, or updating component	$R(S_t, a, S_{t+1} = -w_t \frac{T(a)}{T_{max}} - w_c \frac{C(a)}{C_{max}}$, where the weights show the importance of the execution time $T(a)$ and cost $C(a)$	Algorithms are compared in terms of execution time, learning episode, and cumulative reward
	[112]	No	Model-free	Same as in [107], but with one extra action to start the component faster. Also, the evaluation is done on stationary and non-stationary systems			

Algorithm	Ref.	CPS	Model	States	Actions	Reward	Evaluation
VI-parallel	[108]	No	Model-based	Probability that the system is under attack, and the system status, such as if the firewall is active or no	18 actions are mentioned including blocking source IP, redirect to honeypot, and alert	$R(S_t, a, S_{t+1}) = -w_t \frac{T(a)}{T_{max}} - w_c \frac{C(a)}{C_{max}} - w_i I(x)$, where $I(x)$ shows the impact index of the action x on the system	The VI and the UCT algorithms are compared in terms of resolution time, cost, and impact
Q-learning	[109]	Yes	Model-based	Binary vector indicating compromised devices	Patching, restarting, and switching to another device	$U_t^D(A_{t,i}, D_{t,j}) = -\epsilon(A_{t,i}, D$ $*R(A_{t,i}) + T(D_{t,j}) - C(A_{t,i})$	Evaluated on a control process in terms of convergence time
	[113]	No	Model-free	S_A for under attack and S_A for normal	a_p for protection and a_{dn} for a do-nothing	A reward matrix is defined $R: \begin{matrix} s_N & s_A \end{matrix} \begin{bmatrix} 0 & 2 \\ 1 & -1 \end{bmatrix}^T \begin{matrix} a_p \\ a_{dn} \end{matrix}$	Effectiveness evaluation is done using root mean square error
	[114]	No	Model-free	8 variables including: isWebServerOn, isWebServerUnder Attack, and CPU load	14 actions such as: scaleup WS, start WS, patch httpd, update, and shutdown	$R(a) = -w_t \frac{T(a)}{T_{max}} - w_c \frac{C(a)}{C_{max}} - w_{conf} Con$ $-w_I I(a) - W_A$A(a)	Cumulative reward is used to test the approach
	[110]	Yes	Model-based	Security status level (low, medium, high, or critical)	Random strategy, human capabilities, and patch managing	Not mentioned	Based on the success rate of the attacker after applying the policies
	[111]	Yes	Model-based	A set of game repetitions	A set of 10 actions of different transmissi	$R_D(s, a, d) = -1, if\ U_A > U_D$ and 0 otherwise, where	Evaluation is based on the voltage violation of the system

Algorithm	Ref.	CPS	Model	States	Actions	Reward	Evaluation
					on lines	U_A and U_D are the payoffs	elements
DDPG	[115]	No	Model-free	Eight features from the network traffic, such as port number and the number of packets	A vector of values between 0.1 and 1 that shows the maximum bandwidth allowed by each host	-1 if the traffic load on the server is $>$ U_S and is $\lambda p_b +$ $(1 -$ $\lambda)(1 - p_a)$ when the load is $\leq$ U_S	Compared with AIMD and CTL approaches in different dynamics
SARSA	[116]	No	Model-free	Traffic rates (the traffic arrived at the router over the last T seconds	10 actions correspon ding to the traffic drop probabilit y (0% to 90%)	-1 if the load on the router is $>$ the upper boundary. Otherwise, the reward is between 0 and 1 $r =$ $\frac{legitimateLoa}{LegitimateLoa}$	Scalability, adaptability , and resiliently are tested against existing throttling approaches

4.3 Summary, Limitations, and Discussions

This section provides a full comparison of the numerous works analyzed, focusing on both the positives and the negatives of each work. According to our conducted survey, Table 8 presents some of the advantages, disadvantages, and future works of the different IRSs solutions. In here, we discuss and point-out several challenges and shortcomings that seem to be the most prominent in the development of Intrusion response systems. Also, we highlight the future directions analyzed from our conducted survey that require immediate attention from the research community.

Traditionally, intrusion response techniques had been a manual and time-consuming process. However, this approach is not suitable for real-time critical cyber-physical systems. Unfortunately, there are many challenges that researchers face when designing automatic intrusion response systems. To begin with, there is a very limited explored countermeasure

pool of actions, especially the ones that are applicable for CPSs and consider both the cyber and the physical damages. Also, it is noticeable that most of the used countermeasures in designing IRSs are cyber-level actions. This shows a deficiency of response actions usage in the physical-level, which is very important for securing CPSs. Moreover, only little is known about combining several atomic actions when responding to a detected attack. The execution requirements of the countermeasures have been also ignored by the researchers who focused mainly on the optimal selection of responses. Accordingly, it is noteworthy that the reviewed works use a very small set of countermeasures, which makes their performance questionable when the search space expands.

Concerning the conventional approaches, Table 6 reveals that most of the works require an attack modeling technique for the risk assessment phase, which is usually an overhead and not very easy to implement, especially for large networks. Also, the usage of MOOPs to solve the decision problem neglects the nature of having a constantly changing state space. Besides, there is no standard representation for attack modeling techniques, which makes it even more challenging. Moreover, considering a static attack model, as in [100], is not very realistic to the dynamicity of the attacker behavior. Some works don't use an attack model and assume that the risk parameters are statically obtained by security experts as in [101]. However, this is another simple and not very realistic assumption for critical industrial CPSs. Furthermore, the presence of experts' opinions when designing models, such as attack trees and service dependency graphs, makes the decision of optimal countermeasures somehow subjective to their own perspectives, which is not always desirable. These limitations encourage the exploration of more advanced, automated, and intelligent solutions, such as the current direction of using reinforcement learning in solving decision-making problems where the risk assessment part is embedded in the learning algorithm. There are also clear limitations in the works that use the game theory approach, such as assuming a finite state space, which is usually not the case in cybersecurity. Additionally, some authors falsely assume in game theory solutions that each player must know the cost function of other players. This assumption is difficult to be justified in real-life scenarios. Another limitation in

game theory approaches is that all of them build games with no more than two-players.

Another important limitation analyzed from our literature review is that most of the studied works don't deal with the problem of handling zero-day attack scenarios and scalability issues. We noticed that most works rely on small-scale simulated environments to evaluate their solution as in [104], while Only few, such as [106], assessed the feasibility of their IRS in large-scale environments. It is worth mentioning that the usage of real environments in the evaluation process was also neglected, which raises some concerns on the applicability of the solutions in real-life scenarios.

Continuing on the limitations, most of the solution approaches do not consider modelling both the attacker and the defender when designing an IRS as in [112]. The unavailability of a publicly available dataset, which stores all of the (state, action, next state, reward) tuples, for building response systems for CPSs using an offline approach is another deficiency that authors face. There is also a lack in IRSs designs for dynamic non-stationary environments that could change abruptly with unknown probabilities. Additionally, there is a lack of open-source tools that can be used for preventing, assessing, and responding to security breaches in CPSs.

Another fact stemming from Table 7 is that all the works analyzed that use RL for designing IRSs, except those presented in [109] and [111], do not consider a CPS environment. Scopus brought up only 16 search results, as shown in Figure 11, for ({Intrusion response} OR {Countermeasure} OR {Defense} OR {Incident response} OR {Mitigation}) AND {Reinforcement learning} AND ({Cyber-physical} OR {Industrial systems}). From this Scopus figure, we can see that most of these results are in 2020, which is very recent. This indicates that the utilization of RL in IRSs that are designed for CPSs is still in its very early stages. One can easily notice that many other challenges have been overlooked by the researchers, such as the real-time response issue, the alert parallelization problem, and handling false alarms since most works assume that the received alarms are 100% accurate.

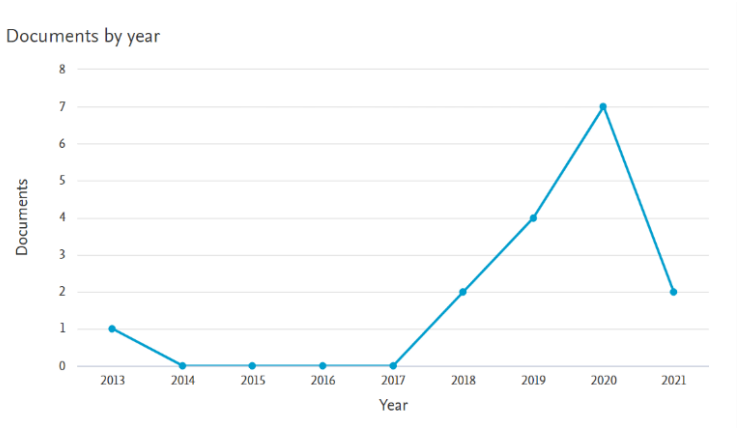

Figure 11. Scopus: publications on RL for IRSs in CPSs

All these complex challenges resulted in limited applicability of the research work in IRSs to real-life systems. Also, it delayed the development of commercial IRS tools and publicly available datasets. This emphasizes that the work on IRSs, especially for CPSs, is still in its very early stages and that there are still open research questions that require further investigations from the research community in the area of IRSs design.

All in all, the limitations discussed reflect how important, yet very difficult to design, is intrusion response systems for CPSs. In this thesis, our proposed methodology, which utilizes model-free deep reinforcement learning, addresses some of these limitations. Initially, our solution provides an enhanced countermeasure pool, which provides composite actions on both the cyber-level and the process-level, applicable for effectively securing CPSs. Also, the proposed method handles the generalization issue that conventional approaches suffer from. Moreover, using a model-free approach avoids the overhead of dealing with the modelling part, which is usually very hard and inaccurate for large complex CPSs because they involve hundreds of sensors and actuators. Accordingly, we investigate the usage of both conventional Genetic algorithm solution and model-free deep reinforcement learning for developing an IRS for a CPS, which is a solution approach that has not been explored to date. Our goal is to peruse an online RL approach, which considers both the cyber level and the process level data, for the optimal selection of countermeasures in a CPS testbed.

Table 8. Advantages, Disadvantages, and Future Works of the IRSs Approaches

Approach	Ref.	Advantages	Disadvantages	Future works
Multi-objective optimization functions	[99]	-Fast dynamic response selection -Adaptive approach -Models attacker behavior -Considers combining responses	-Unrealistic usage of 1 ADT to protect assets -Lacks in scalability -The difficulty of creating ADT/SDG is not considered	-Considering the applicability of using multiple attack defense trees
	[100]	-Considers defense and recovery measures on both cyber and physical domains -Prevents the expansion of the attack surface -Experiments with different attack scenarios	-Does not address the scalability and time complexity issues	-Not mentioned
	[101]	-Focuses on the execution of the countermeasures	-Does not discuss effects of ongoing attacks -Uses unjustified static values	-Considering heterogeneous nodes -Discussing security policy generation
	[102]	-Decides on the order and duration of selected actions	-Does not model the attacker behavior	-Considering the time interval between actions' deployment and execution
Game theory	[103]	-Considers cyber-side and physical-side responses -Considers sensor alert uncertainties -Not too system-specific	-Using a limited list of countermeasures	-Not mentioned
	[104]	-Provides a complete multi-layer defense architecture -Applicable to different CPSs	-Uses an abstract high-level explanation -Does not take the attackers' expertise and time to build attacker model into consideration	-Exploring different applications for CPS protection
	[105]	-Presents a novel risk assessment approach -Considers both defense and recovery actions	-Computation time and complexity are an issue	-Using a more efficient dynamic update algorithm for the MLBN

Approach	Ref.	Advantages	Disadvantages	Future works
	[106]	-Provides a scalable solution -Uses a distributed model that improves the performance	-The distributed model adds trust issues between the nodes -The used countermeasures are not presented	-Not mentioned
Reinforcement learning	[107]	-Does not need an accurate model of the system -Handles large-scale systems	-Requires several episodes to converge -Tuning hyperparameters is time consuming	-Using GPUs to increase processing speed -Considering multi-agent systems
	[112]	-Deals with a non-stationary system -Better generalization property -Scalable approach	-Ignores modelling the attacker behavior	-Using GPUs to increase processing speed -Considering multi-agent systems
	[108]	-Captures both the defender and attacker models -Suitable for large-scale systems -Provides a proactive approach	-No feedback loop between the system and the IRS agent	-Establishing a feedback loop between the controller and the system -Considering a non-deterministic MDP
	[113]	-Uses offline approach with a network dataset	-Poor evaluation metrics are used	-Exploring Q-learning algorithm in different domains -Considering combining Q-learning with other models to improve the performance
	[114]	-First to consider dynamic non-stationary systems -Different attack dynamic scenarios are considered	-More time needed to converge to a near-optimal solution -Does not model the attacker behavior	-Comparing with different algorithms -Including attacker behavior

Approach	Ref.	Advantages	Disadvantages	Future works
	[115]	-The usage of SDN decouples control from data -Different attack dynamics are considered	-Does not address the scalability issue	-Not mentioned
	[116]	-Scalable decentralized approach -Both offline and online training are considered	-Does not consider attackers with legitimate sending rates	-Focusing on online training -Exploring ways to improve the learning speed
	[109]	-The time needed to recover metric is considered	-It assumes complete players knowledge	-Considering unknown vulnerabilities -Considering players with incomplete information of other players
	[110]	-Models both the defender and attacker -Uses realistic assumptions	-Does not protect the physical layer	-Considering a game with imperfect information
	[111]	-Models both the defender and attacker	-No comparison with other RL algorithms	-Extending the approach for general CPSs
Proposed solution	NA	-Detailed comprehensive background and survey work -Modeling and design of a CPS testbed -Modelling and design of cyberattacks -Uses both a Genetic algorithm and a model-free DDQN algorithm to solve the intrusion response system problem -Builds an offline dataset for RL usage	-No comparison with other RL algorithms -Very long training time (scalability is an issue) -Limited state-space and action-space considered	-Consider multi-agent RL approaches -Consider more attack scenarios -Use the collected dataset in an offline RL approach -Compare with other RL algorithms

CHAPTER 5: MODELING AND DESIGN OF A CPS TESTBED

In this chapter, we present the modelling and design of a small-scale simulated exothermic Continuous Stirred Tank Reactor (CSTR) testbed using both MATLAB/Simulink and LabVIEW. The motivation behind building this testbed is to use it as our interactive environment, where we apply our conducted experimentations, in both the GA-based and RL-based intrusion response decision-making designs. This is due to the lack of publicly available security-relevant datasets that can be used in our cyber security research investigations. Also, we present the modelling and design of different attack scenarios considered in each proposed solution.

5.1 CPS Description

In this section, we give the overall picture of our considered CSTR physical process. Also, we mention the high-level architecture of our designed CPS testbed, including the used components, their roles, and the communication protocol used between them.

5.1.1 Process Description

The CSTR plays a vital role in the process industry, where cyber security is essential for the safety and reliability of its physical system operations. It is essential in any process plant that generates new products from raw inlet reactants. We chose the CSTR as the physical system because of many reasons. First, the process variables, which we aim to regulate and control, are closely coupled. Thus, any change in one process variable will impact other variables and manifest itself in the overall process behaviour. Second, the process has several safety hazard scenarios, which can be produced by a cyberattack. Finally, mitigation layers for a number of safety hazards rely mainly on the control and safety systems, which are cyber systems that could be compromised by a cyberattack. Accordingly, this type of process is suitable for experimenting with different cyberattack scenarios, implementing different mitigation techniques that consider simultaneously cyber and physical actions, and evaluating the effectiveness of the defensive mechanisms in a realistic operating process environment.

We consider an irreversible exothermic CSTR process, with a first order reaction in the reactant A with rate k and a heat of reaction λ.

$$A \xrightarrow{k} B$$

Figure 12 shows the Piping & Instrumentation Diagram (P&ID) for the reactor. The reactor vessel has an inlet stream, an outlet stream, and a coolant stream. The inlet stream is where the reactant is carried in, the output stream is where the product is carried out, and the cooling stream is where the cooling fluid is carried in to absorb the heat of the exothermic reaction. Reactant A enters the reactor with concentration C_{A_0}, temperature T_0, and volumetric flow rate F_0. A first-order reaction takes place where a mole percentage of reactant A is consumed to produce product B. The outlet stream contains both reactant A and product B, with reactant A concentration C_A, outlet temperature T, and flow F. The outlet temperature T is the same as the reactor temperature. The coolant fluid flows into the reactor jacket with temperature T_{J_0} and flow rate F_{J_0}, and leaves the jacket with temperature T_J. The total coolant volume in the jacket is designated by V_J. The detailed mathematical model of the non-linear reactor is out of the scope of this thesis, but readers can refer to [118] for modelling details.

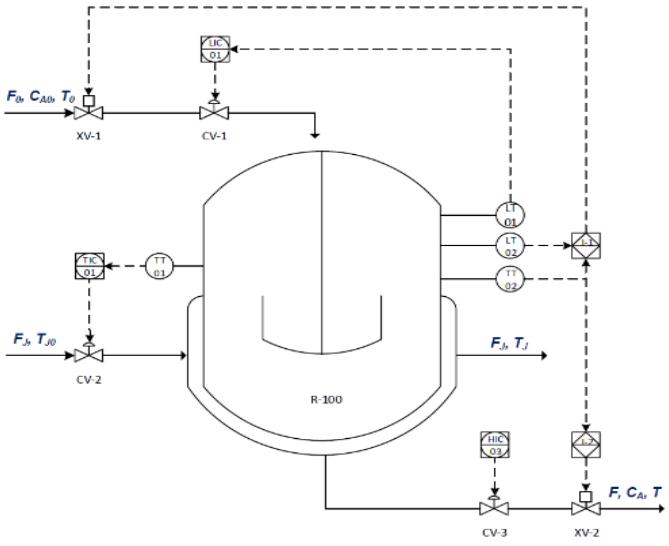

Figure 12. Reactor P&ID

5.1.2 Cyber System Description

Figure 13 shows the architecture of the CPS testbed, which is implemented as part of an NPRP project at Qatar University. This part describes the role of each component found in the architecture. Starting from the process simulator, it numerically solves the model differential equations. The Basic Process Control System (BPCS) executes the control logic responsible for regulating the different variables of the process. The Safety Instrumented System (SIS) executes the safety shutdown logic when BPCS fails. Following the IEC 61511 standard, the BPCS and SIS have to be completely independent, including fields sensors, logic solvers, and field actuators. The Human Machine Interface (HMI) is a graphical user interface for monitoring the physical process and allowing manual controlling by a human operator when needed. It should be highlighted that no operator action is allowed on the SIS. The firewall separates the control network from the cooperate network. The IDS detects abnormalities and sends related evidence to the RL agent. It is worth noting that the design of the IDS is out of the scope of this thesis, but we assume the presence of an active IDS with a 100% trust. Finally, the RL agent node is where we placed our IRS agent during the training and testing phases to receive system states and send applicable selected actions. All these components connect to the control network via an Ethernet interface.

Each node has its own communication path that is used to interconnect with the rest of the testbed nodes. The process simulator communicates with the BPCS controller physically using the I/O lines in the cRio module. The SIS communicates with the BPCS through a Modbus link. Also, the BPCS controller communicates and receives commands from the HMI through a Modbus link. Since Modbus TCP/IP communication protocol considers a Master/Slave architecture, it is worth noting that the BPCS is the master, and the SIS is the slave in the first communication link. However, in the second Modbus link, the HMI plays the master role and the BPCS takes the slave role. Also, it should be highlighted that no direct communication link is allowed on the SIS from the HMI. The RL agent node communicates with the rest of the components using the UDP/IP

communication protocol.

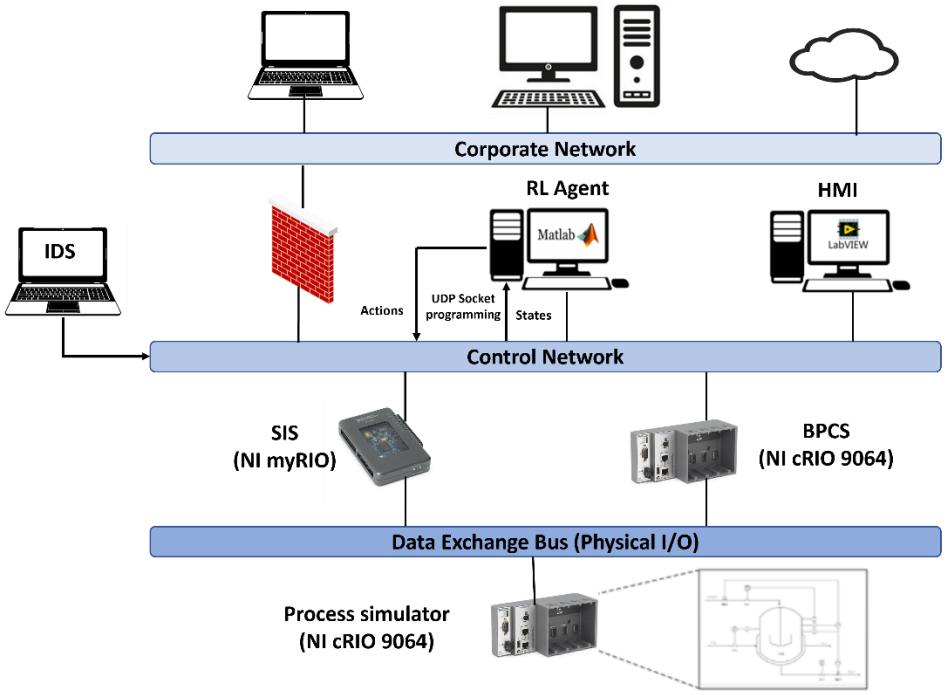

Figure 13. Testbed architecture

5.2 CPS Implementation

The process is simulated using two open-source platforms, which are MATLAB/Simulink and LabVIEW Real-Time (RT) module. It is worth mentioning that several other simulated testbeds were developed at various labs for researchers to experiment with cyberattacks and vulnerabilities and evaluate their detection and defensive mechanisms. Examples include the smart power grid testbeds presented in [119], the National SCADA Testbed (NSTB) [120], and the Idaho National Labs (INL) SCADA Testbed [121]. In our experiments, we use the Simulink simulation to solve the intrusion response problem using the Genetic Algorithm approach (GA-IRS). However, the LabVIEW simulation is used to solve the same problem using the reinforcement learning-based DDQN approach (DRL-IRS).

5.2.1 Process Simulation

The process simulation model was implemented in both Simulink and LabVIEW. Both platforms are graphical-based (block diagram) environments that are widely used to simulate industrial systems. Initially, we simulated the non-linear controller, which was

designed to stabilize the system, in Simulink as shown in Figure 14. The Reactor S-Function is where we define the mathematical model equations for the reactor and initialize all the required parameters. Then, we moved to LabVIEW, where we added hardware components and a communication network. Figure 15 shows the front panel of the designed process simulation in LabVIEW, while Figure 16 shows a snippet from its block diagram where the mathematical formulas of the model are implemented.

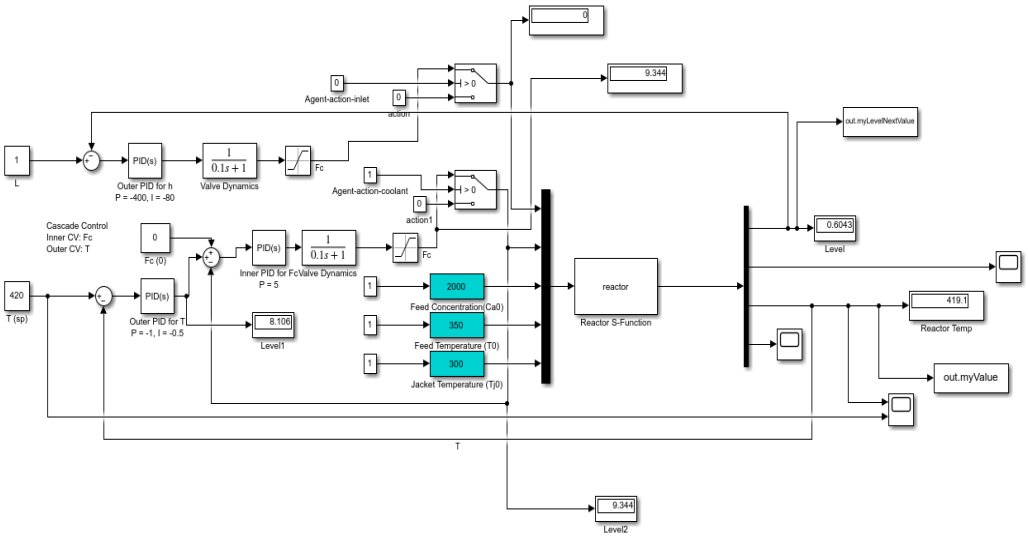

Figure 14. Process simulation model in MATLAB/Simulink

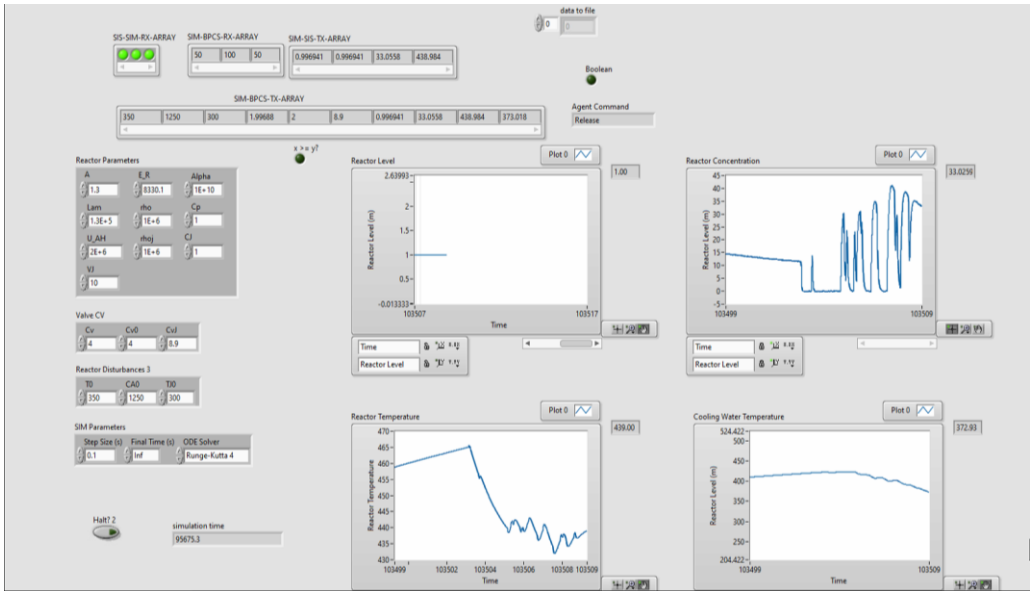

Figure 15. Process simulation for the reactor in LabVIEW (front panel)

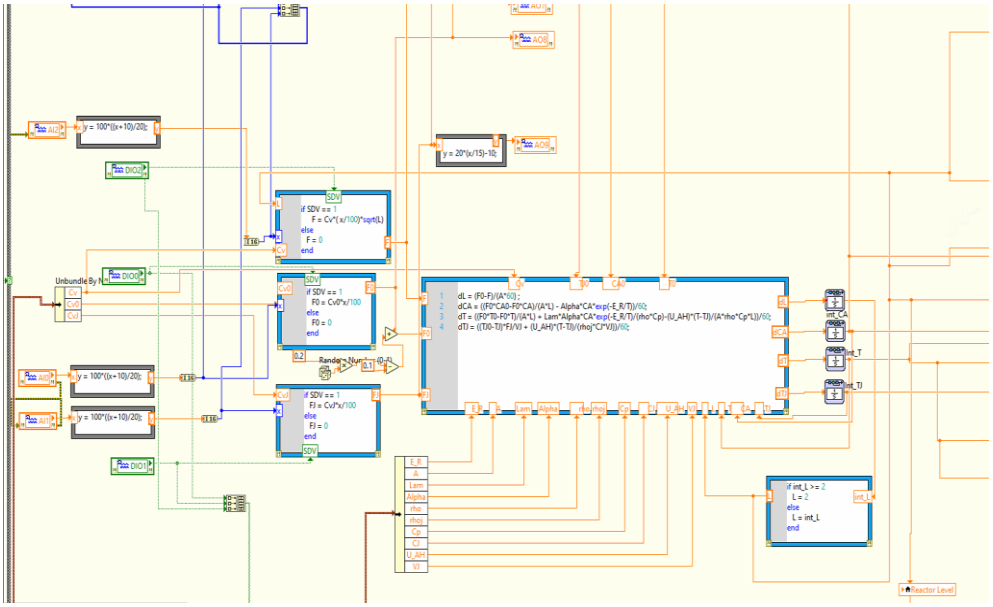

Figure 16. Process simulation for the reactor in LabVIEW (block diagram snippet)

5.2.2 Cyber System Implementation

In this section, we explain some implementation details of the testbed's components shown in Figure 13. The testbed uses open hardware and software components to allow control of the experimental environment. For example, the testbed uses industrial NI controllers, which allow low-level programming for all software tasks, including communication protocols. The process simulator, BPCS controller, and SIS controller run on Compact RIO (cRIO) 9064, Compact RIO (cRIO) 9064, and myRIO 1900, respectively.

Initially, the measurement and actuation signals were exchanged between the process simulation, controllers, and HMI using a high-speed UDP/IP communication over Ethernet. However, we wanted to simulate a more realistic environment for an industrial process. Accordingly, we changed the communication links to Modbus/TCP, which is a more reliable and widely used communication protocol for industrial processes. The Modbus Application Data Unit (ADU) is shown in Figure 17. We also represented each sensor and actuator as one I/O line in cRio modules. The I/O lines are connected physically to the controllers to mimic real sensors' and actuators' connections. The sensors

that are considered in the testbed measure the temperature, the concentration, the flow rate, and the reactor level. While there are three main actuators, which are the inlet valve, the outlet valve, and the coolant valve, that control the flow rate for the different pipes.

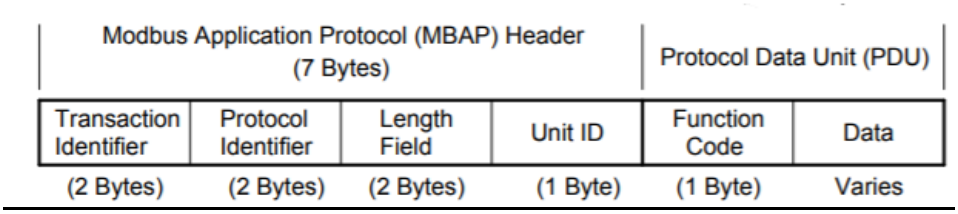

Figure 17. Modbus/TCP ADU

The process simulation model is implemented using LabVIEW Real-Time (RT) module. The Simulation uses fixed Ordinary Differential Equations (ODE) solver with a simulation step size of $0.1\,ms$ real-time resolution. It also sends the process data to the process controller every $10\,ms$, through the I/O lines, as responses to the received queries. The BPCS, which runs RT Linux OS, uses the PID control algorithm since it is the defacto standard in the process control industry. The PID is used to control the level, temperature, and concentration of the reactor. Commands are sent from the HMI to the BPCS using the 'write holding registers' and 'write holding coils' Modbus communication blocks in LabVIEW based on the type of the sent data. These values are used by the PID to perform the control logic. The controller also exchanges the data received from the HMI to the physical process visa periodic communication.

The SIS, which runs RT Linus OS, implements the safety shutdown logic when hazards are identified. Hazards happen mainly when the process variables exceed their safe operating limit. For example, in a reactor overflow hazard (level > 95%), the inlet stream has to be closed. While for a high-temperature hazard, both the inlet and the outlet stream valves should be closed. The SIS communicates two types of data to the BPCS, which are periodic field measurements and discrete events that take place during a shutdown for display purposes and further control actions. The HMI, which acts as an

operator interface to the physical process, is developed using LabVIEW graphical programming and runs on Windows 10 OS, as shown in Figure 18. The HMI periodically reads process data information from the BPCS. As can be seen from Figure 19, which shows a snippet from the front panel of the designed HMI, reading responses from the BPCS are done using the Modbus communication blocks, such as ‘read input registers’ and ‘read discrete inputs’. It should be highlighted that each type of these Modbus blocks uses a different Modbus function code. Another point to notice is that the HMI is not allowed to write directly to field devices nor to have direct communication with SIS for safety reasons. Concerning the firewall, it was implemented using iptables running on Ubuntu Linux. These tables were used to ensure that no direct traffic is allowed between the control and cooperate networks.

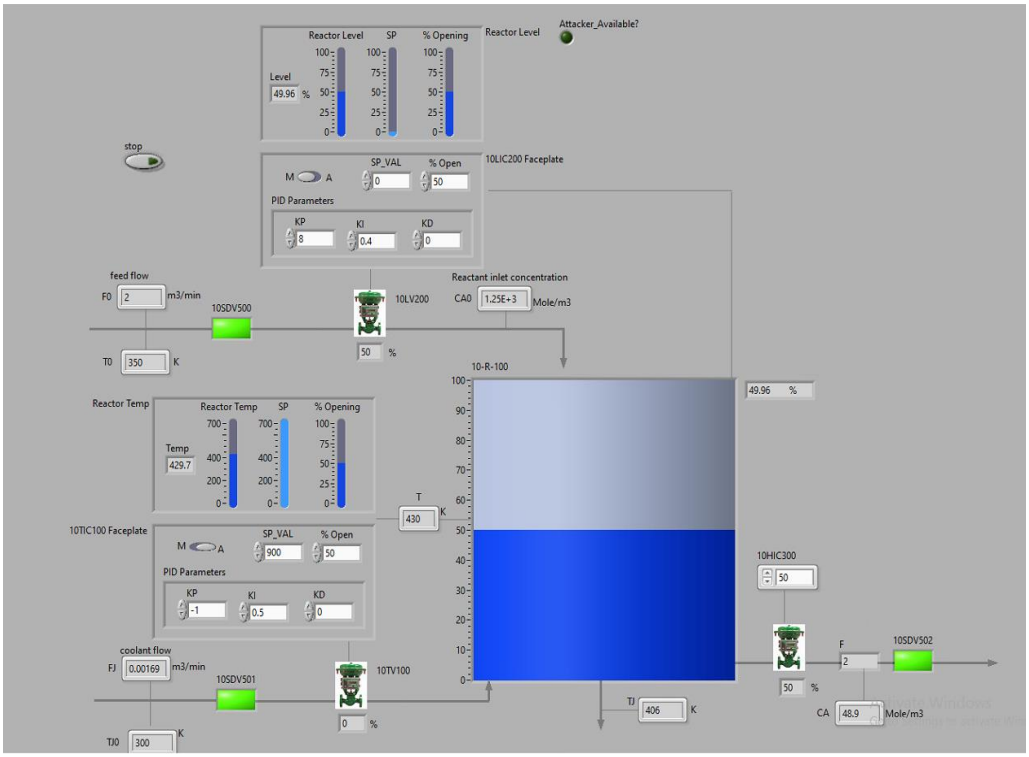

Figure 18. HMI for the reactor process (front panel)

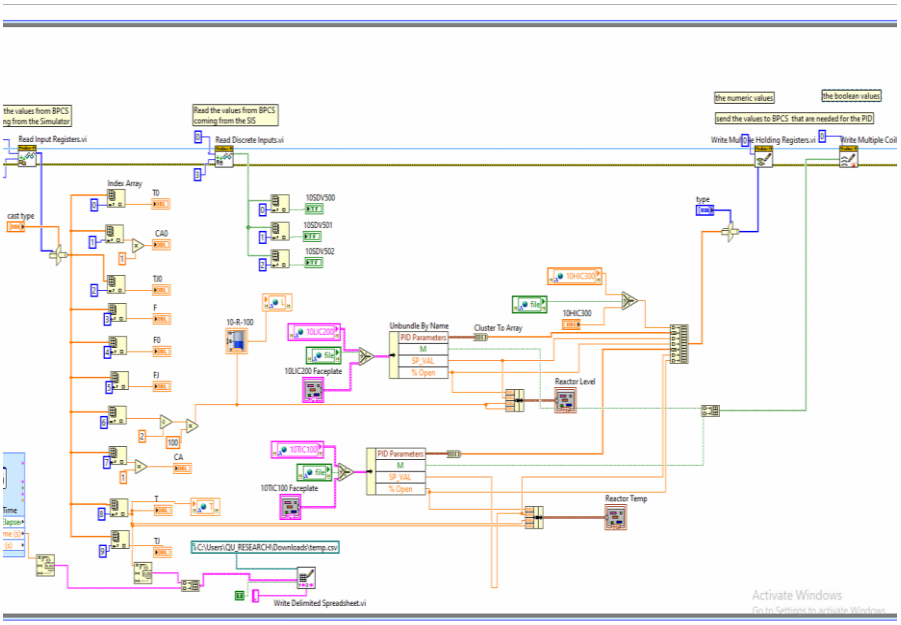

Figure 19. HMI for the reactor process (snippet block diagram)

The RL Agent node in the architecture is where we placed our reinforcement learning intrusion response agent during the training and testing phase. This node is responsible for receiving states and sending back appropriate defensive actions. For communication purposes, we opened a UDP/IP communication channel between our RL agent node (which uses MATLAB) and the BPCS controller, and the SIS controller in LabVIEW. The selection of UDP/IP is mainly because we looked for a fast, simple, and efficient communication protocol. In our experiments, we assume that all our required states are gathered and sent from the BPCS controller to the RL agent through this UDP/IP communication link. Also, this communication link is used by the RL agent to send the selected actions to their execution locations, which can be on the BPCS controller, SIS controller, or the control network. Note that the RL agent has direct access to all the components and can overwrite any logic. All in all, these modifications in the communication setups are to prepare the CSTR testbed to be used as an interactive environment for the training of the reinforcement learning IRS agent (DRL-IRS), which will be presented in Chapter 7.

5.3 Modelling and Design of Cyberattacks

This section presents a comprehensive and detailed attack tree for modelling different attack scenarios, which are Reconnaissance attack, MITM attack, Denial of Service (DoS) attack, and Replay attack. We also discuss the different attack scenarios considered in both the GA-IRS and DRL-IRS solution approaches. More details on the design and implementation of cyber physical attacks on Modbus/TCP protocol is available in our published research paper [122].

5.3.1 Attacks Model

According to [68], attack trees are visual diagrams that are very popular for modelling the sequence of steps needed to perform different cyber-attacks. Figure 20 shows our designed bottom-up attack tree that outlines how to perform reconnaissance, replay injection, command/response modification, and DoS attacks against our CPS testbed that uses Modbus/TCP communication protocol.

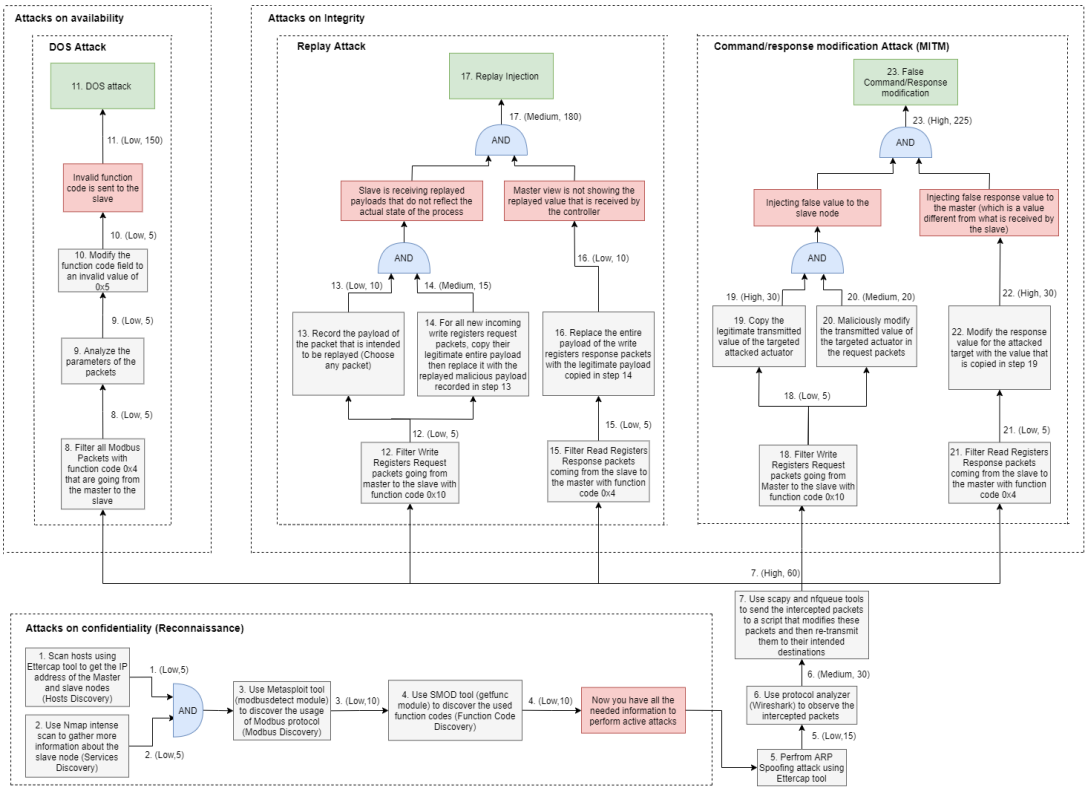

Figure 20. Attack tree model against CPS using Modbus/TCP protocol

Three attributes, which are summarized in Table 9, are associated with each attack step: Time to execute the attack step (t), resources/tools required (r), and knowledge needed (k). The tuple (k, t) is added to each tree edge while the resources (r) are embedded in each attack step. The time to execute each attack step could be modeled as a random variable with a Probability Density Function (PDF) that depends on the attacker's profile. In this work, we assume a Gaussian distribution for mathematical tractability, T ~ N (μ, σ), with a mean time to execute the attack (μ) and standard deviation (σ), when possessing the required knowledge and resources.

Table 9. Attack Tree Attributes

Attributes	**Symbol**	**Description**
Resources	r	Resources needed to perform each attack step, such as tools and manpower
Time	t	The time taken (in min) to successfully execute each attack step
Knowledge	k	The level of attacker's knowledge needed on a scale (Low/Medium/High)

Other distributions could be used as well. The correct approach to decide on a specific distribution is to learn from real attack data, which is still lacking in the research community. To capture the variation of the PDF with the attacker profile, several attacker's attributes could be defined, such as knowledge, access to tools, financial ability, and motivation [123]. In this work, we consider two attributes only, resources and knowledge, that are mapped to the attack attributes. To facilitate quantitative analysis, we assume r, k ∈ [0,1] are normalized values. This gives rise to the two-dimensional attack space depicted in Figure 21. The attack step is represented by the vector $[r_s, k_s]$ and the attacker profile is represented by the vector $[r_a, k_a]$. The distance between the attacker profile and attack vector changes the execution time PDF via a function mapping g(.). An example function that changes the Gaussian distribution parameter μ is

$$\mu = \mu_0 - \alpha_1(r_a - r_s) - \alpha_2(k_a - k_s) \qquad \text{Equation 2}$$

where μ_0 is the parameter value for (r_s, k_s) attribute values, and α_1 and α_2 are weight factors that reward or penalize the excess or shortage in required resources and

knowledge, respectively.

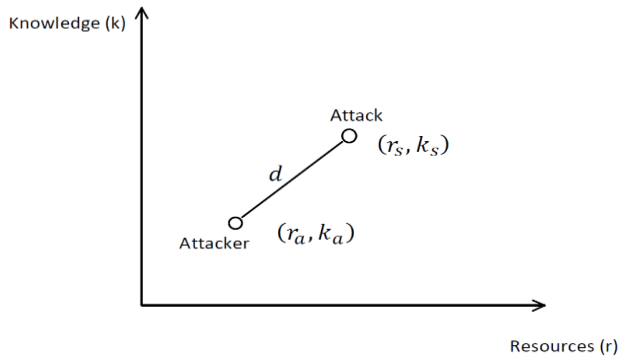

Figure 21. Attack attributes vs attacker profile. The attack attributes define the required resources and knowledge required for a successful attack

Equation 2 assumes that if the attacker has less knowledge and resources, the attack could still succeed but with a longer time. Other functions could model the scenario of the impossibility of launching an attack if the resources, such as tools or knowledge, cannot be met. This will have the impact of shifting the execution time PDF such that the mean time approaches infinity. Therefore, different attack steps will have different function definitions. Figure 22 shows a family of distributions for the attack execution time for different attacker profiles where the parameter μ is calculated according to (1) with $\mu_0 = 10,\ \sigma = 1$.

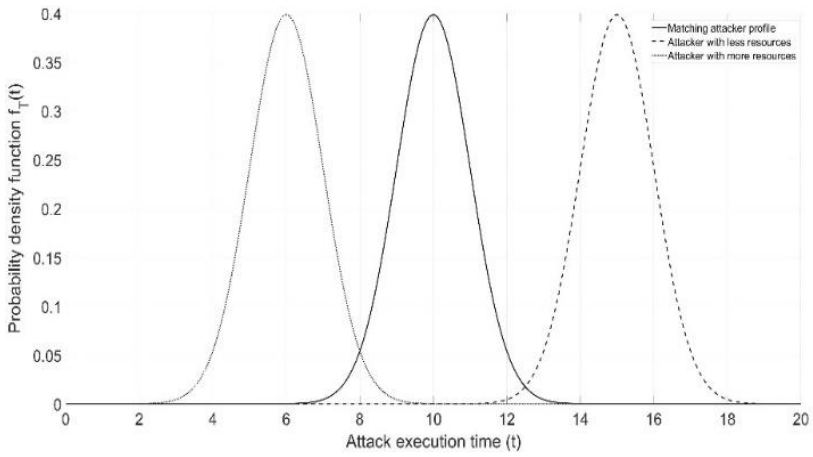

Figure 22. Execution time distribution for different attacker profiles

The attack tree, shown in Figure 20, identifies the different attack paths that can be used to compromise and threaten our CPS. Initially, it starts with performing reconnaissance attacks to affect the confidentiality of the system and gather enough information for performing more sophisticated active attacks. The next step intercepts the flow of traffic between the targeted parties using ARP spoofing, which poisons the ARP table of the two parties by linking their IP addresses with the attacker's MAC address. Following that, the packets are analyzed using Wireshark and thus are accessible by the attacker to be modified. Modifications include changing the function code field to an unsupported one, injecting a whole replayed payload, or changing any specific value in the payload. Finally, the packets are re-transmitted to their intended destination after being maliciously changed to cause either a DoS attack, a replay attack, or a false command/response modification attack. In this work, we assume an insider attacker profile with resources and knowledge (r, k) matching the requirements of each attack step.

5.3.2 Attack Scenarios

There are several vulnerabilities in the process that can be exploited by attackers to launch different attack scenarios. The two main hazards associated with the CSTR process are the reactor overflow (high level) and reactor runaway (high temperature). In the process industry, Hazard and Operability (HAZOP) study is the key risk assessment methodology used to identify hazards, their initiating events, and the consequences [124]. Table 10 is a partial HAZOP sheet showing the two hazards and their consequences.

Table 10. Partial HAZOP Sheet for the Reactor Process

Hazard	**Initiating Event (Cause)**	**Consequences**	**Safeguards (IPL)**
High Level (Reactor overflow)	Controller failure OR Outlet control valve fully closed OR Inlet valve stuck fully open	2 or more fatalities (safety), Product loss (financial), Environmental contamination (environment)	Reactor dike (Mitigation)

Hazard	Initiating Event (Cause)	Consequences	Safeguards (IPL)
High Temperature (Reactor Meltdown)	Coolant inlet control valve fully (partially) closed OR Inlet valve stuck fully open	10 or more fatalities (safety), Product loss (financial), Environmental contamination (environment)	None

In this thesis, both GA-IRS and DRL-IRS solutions, consider that our testbed environment is under the Command/Response modification attack against the setpoint, which is an example of the false data injection MITM attacks. This attack aims to tamper different defined setpoints to either increase the reactor's temperature or reactor's level beyond design limits and hence, cause a reactor meltdown attack or a reactor overflow attack, respectively. An important point to highlight is that the detection procedures of the attacks is out of the scope of this thesis, but there is an assumption that there is an active IDS that detects abnormalities with a 100% trust since dealing with IDS uncertainties is also beyond the scope of this thesis.

5.3.2.1 Attack Scenarios for the GA-IRS Approach

In the GA-IRS approach, we consider different scenarios for the false data injection attack. Each scenario manipulates the temperature and/or the level of the reactor by setting a fault setpoint to cause different hazardous consequences. It is worth mentioning that the considered attack scenarios are simulated, and that the attacker profile is assumed to be not persistent, which means that it is a one-shot attack since the attacker gets out of the network once succeeding in performing the attack that drives the process towards hazardous situations.

At first, we divided the range of the reactor's temperature and level into 4 regions, which are low, normal, high, and hazard as shown in Table 11. Under normal conditions, when the level is less than 30%, the water pump turns on and the inlet flow becomes $2\ m^3/sec$ to prevent an underflow. When the water level is greater than 70%, the water pump turns off and the inlet water flow becomes 0 to prevent a

possible overflow. For the temperature, the coolant valve opens to cool down the reactor's temperature when it is greater than $460\ k$. Accordingly, the level dangerous point is $< HL = 95\%,\ LL = 10\% >$, The level setpoint is $< H_L = 70\%, L_L = 30\% >$, the temperature dangerous point is $< HT = 480\ k, LT = 360\ k >$, and the temperature setpoint is $< H_T = 460\ k,\ L_T = 410\ k >$. It is important to know that the ideal reference temperature is $420\ k$ and level is $1\ m$.

Table 11. Categorical Classifications of the Reactor's Parameters'

Parameter	**Low**	**Normal**	**High**	**Hazard**
Temperature (T)	$360 < T < 410$	$410 \leq T \leq 460$	$460 < T < 480$	$T \geq 480$ $T \leq 360$
Level (L)	$0.2 < L < 0.6$	$0.6 \leq L \leq 1.4$	$1.4 < L < 1.9$	$L \geq 1.9$ $L \leq 0.2$

After knowing the normal behavior of our CSTR process with standard setpoints, we designed different attack scenarios for tampering the setpoint configuration data of the temperature and the level as shown in Table 12. The following scenarios represent how tampering the setpoint can lead to dangerous abnormalities if not detected and regulated by selecting the appropriate countermeasures.

Table 12. Attack Scenarios Description for the GA-IRS solution

Scenario	**Attack Type**	**Attack Description**
1	Setting a fault setpoint for $L_L\ and\ L_T$	$L_L\ and\ L_T$ are tampered from 30% and 410 to 5% and 370, respectively. So, when the level is less than 30% but greater than 5%, the water pump will not turn on because the setpoint was tampered to 5%. Thus, the level will continue to drop. The same goes to the temperature.
2	Setting a fault setpoint for H_L	H_L is tampered from 70% to 100%. So, when the level is more than 70% and less than 100%, the water pump will not turn off because of the fault setpoint. Thus, the level will continue to increase until it brims over the tank causing a reactor overflow attack.
3	Setting a fault setpoint	H_T is tampered from 460 to 500. So, when the

Scenario	Attack Type	Attack Description
	for H_T	temperature is more than 460 and less than 500, the coolant valve will still not open to cool down the process. Hence, the temperature will keep on increasing.

5.3.2.2 Attack Scenarios for the DRL-IRS Approach

In our experiments to train the DRL-IRS agent, we considered the fault setpoint data injection attack on the temperature. Figure 20 shows the detailed implementation steps of this attack, which aims to increase the reactor's temperature beyond design limits and hence, cause a reactor meltdown attack. In this attack, the attacker compromises the communication link between the HMI and the BPCS controller to send false commands that tamper the temperature setpoint from the standard $420\ k$ to $900\ k$. Accordingly, the controller adjusts to the new malicious setpoint and does not open the coolant valve when the temperature increases beyond $460\ k$ as being designed, and the temperature continues to increase. At the same time, A tampered response is sent back from the BPCS to the HMI showing a normal $420\ k$ temperature setpoint to deceive the operator, while the actual malicious temperature setpoint received and used by the controller is 900. Figure 23 shows how the attacker affects the data integrity of the response/command packets by modifying their payloads from the network level to perform the false data injection attack.

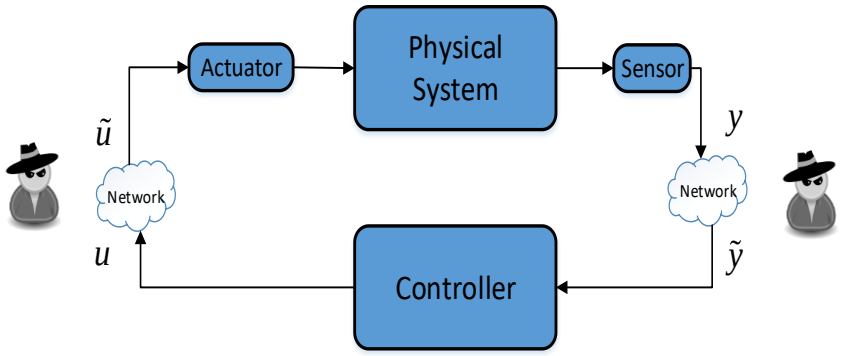

Figure 23. False data injection attack

From this fault setpoint data injection attack, we designed three different scenarios that the DRL-IRS agent has to deal with and find the optimal sequence of actions for each one of them in the different training experiments discussed in Section 8.2 Agent Training . Keeping in mind that the ultimate aim of the three scenarios is to cause a reactor meltdown attack. Table 13 describes the situation of the environment when each scenario is detected. Also, unlike the GA-IRS scenarios where we considered a non-persistent attacker profile, in here, we consider an active determined attacker profile that has all the needed resources and is persistent to reach the goal as long as no mitigations are performed to stop the attacker. Another assumption is that the attacker is an insider, so the steps of penetrating the CPS network is neglected. Finally, we assume that the DRL-IRS agent is activated after receiving an alert from the IDS to respond and select the optimal sequence of actions that can reduce the risk of the detected undergoing attack scenario.

Table 13. Attack Scenarios Description for the DRL-IRS Solution

Scenario	**Attack Scenario Description**
1	The false data injection on the temperature setpoint is performed successfully, the safety controller (SIS) is not compromised by the attacker, and the attacker source or IP address is known
2	The false data injection on the temperature setpoint is performed successfully, the safety controller (SIS) is not compromised by the attacker, but the attacker source/address is not known
3	The false data injection on the temperature setpoint is performed successfully, the safety controller (SIS) is compromised by the attacker, and the attacker source/address is also not known

CHAPTER 6: IRS DESIGN USING GENETIC ALGORITHM (GA-IRS)

In this chapter, we use a conventional optimization approach to solve the intrusion response decision-making problem. We formulate the decision-making problem into a weighted single-objective optimization function that aims to regulate the CSTR process. The Genetic Algorithm (GA) is used to solve the optimization problem to find the optimal response for the different considered case studies. Finally, the impact of the applied approach is evaluated based on its computational complexity and response effectiveness.

6.1 Single-objective Optimization Formulation

The selection of the optimal response action from the action space is formulated as an unconstraint weighted single-objective optimization problem. The goal is to find the optimal action vector that maximizes the final reward objective function. The state space of this chemical reactor control system is defined as $S = [T, L]$, where T is the reactor's temperature in kelvin (k) and L is the reactor's level in meter (m). The considered decision-making actions, which are used to defend against the cyberattacks, are compacted in the vector $A = [IV\ CV\ SP]$. IV and CV are Boolean values to either open ($IV = CV = 1$) or close ($IV = CV = 0$) the inlet valve and the coolant valve, respectively. SP is the temperature setpoint value, which ranges discretely between 300 and 500, and is used to regulate the reactor's temperature. The objective function, which we aim to maximize, is mainly composed of the benefit of the selected action and its cost as follows:

$$max_{x \in X}\ \ Benefit(x) - Cost(x) \qquad \text{Equation 3}$$

Where x is the action vector selected, which belongs to the action space X. The $Benefit(x)$ is a positive value given for action x for keeping the process away from hazards. The $Cost(x)$ is a value that evaluates how deviated is the process from the defined setpoints after performing the selected action vector x. The $Benefit(x)$ is calculated as follows:

$$Benefit(x) = t * P \qquad \text{Equation 4}$$

Where t is the sampling time in sec, and P is the profit per sec. This benefit is given to the selected action as long as the action did not drive the process to hazards but kept it working in

a safe operational window. The safe operational window for the temperature is being greater than $360\ k$ and smaller than $480\ k$. While for the level, the operational window is having the level greater than $0.2\ m$ and smaller than $1.9\ m$. Other than these operational windows, the process is said to be in a hazardous situation, and hence the benefit is zero.

The calculation of $Cost(x)$ needs more detailed discussion. The cost is represented as the Time to Recover (TTR) metric. The TTR is the time needed (in sec) by the CSTR system to put the compromised process near the defined setpoints for the temperature and level. As the TTR increases as a result of deploying action x, the action's cost increases as well. It is worth mentioning that the $Cost(x)$ for actions that drive the system to hazardous conditions when deployed at a specific state is set to be 3000, which is a large penalty (hazard cost (C)) to that action and hence, next time a better action will be explored in that state. The TTR metric, which is multiplied by the profit per sec value (P) to change it from sec to \$ unit, is calculated as shown below:

$$Cost(x) = TTR \qquad \text{Equation 5}$$

$$TTR = P * (w_T * TTR_T + w_L * TTR_L) \qquad \text{Equation 6}$$

Where TTR_T is the time needed to recover the temperature of the reactor and TTR_L is the time needed to recover the level of the reactor. w_T and w_L are weight values to indicate the importance of each term in the equation in which $w_T + w_L = 1$. For calculating TTR_T and TTR_L , the following equations are used:

$$TTR_T = e^{\varepsilon * dev_T(x)} - 1 \qquad \text{Equation 7}$$

$$TTR_L = \alpha * dev_L(x) \qquad \text{Equation 8}$$

Where $dev_T(x)$ and $dev_L(x)$ represent the deviations of the reactor's temperature and level from their defined setpoints (distance metric) after deploying action x, respectively. ε and α are hyperparameter values that are tuned to achieve the best performance. Equation 7 shows an exponential relationship between the deviation of the temperature and its TTR_T cost. While Equation 8 shows a linear relationship between the deviation of the reactor's level and its TTR_L cost. These assumptions are made for simplification purposes and to show the

significant danger of the reactor meltdown attack since it causes more disastrous consequences than the reactor overflow attack as shown in Table 10. To calculate the error deviations, we use the following equations:

$$dev_T(x) = \left|\frac{ref_T - New_T}{ref_T}\right| \quad \text{Equation 9}$$

$$dev_L(x) = \left|\frac{ref_L - New_L}{ref_L}\right| \quad \text{Equation 10}$$

Where ref_T and ref_L are the defined setpoint values for the temperature and level of the CSTR model, respectively. New_T and New_L are the new temperature and new level states, respectively, that the process transmitted to after performing action x. So, the TTR is:

$$TTR = P * ((w_T * e^{\varepsilon * \left|\frac{ref_T - New_T}{ref_T}\right|} - 1) + (w_L * \alpha * \left|\frac{ref_L - New_L}{ref_L}\right|)) \quad \text{Equation 11}$$

Finally, combining all the equations together gives the objective function (Reward function) that our optimization algorithm aims to maximize. This reward function, which is multiplied by a scaling factor M, assesses the quality of the produced product by using the TTR metric as follows:

$$R_{GA} = M * (t * P - P * ((w_T * e^{\varepsilon * \left|\frac{ref_T - New_T}{ref_T}\right|} - 1) + (w_L * \alpha * \left|\frac{ref_L - New_L}{ref_L}\right|))) \quad \text{Equation 12}$$

6.2 Genetic Algorithm Framework

According to [125], Genetic Algorithm (GA) is a very popular heuristic evolutionary technique for solving constrained or unconstrained optimization problems. GA was inspired by Darwin's theory of natural evolution and survival of the fittest. The choice of using GA to solve our intrusion response decision-making problem is mainly because it is one of the most well-established and widely used algorithms in the literature. Figure 24 is a flowchart showing the general scheme of how GA works.

Initially, GA starts with randomly initializing a population of solutions in which each solution is called a Chromosome and is represented as a set of genes. The genes are the decision parameters that describe each solution depending on the type of the problem. The fitness value, which is dependent on the problem we are trying to solve, is then computed in the evaluation step to show the goodness of each solution in the population. For the fitness

values, the higher the number, the better the solution.

Since GA uses the concept of a generational loop to improve their solutions, we need to figure out when to stop looping so that the process does not continue forever. This is done by checking if the termination conditions are satisfied or not. Some termination conditions include having a goal achieved, reaching a maximum number of generations, or noticing a performance stagnation of the fitness scores from generation to generation. Following that, the fittest individuals that we expect to have the most valuable genetic information are selected from the population to become parents and produce the next offspring generation. Several selection methods can be used, such as Roulette-wheel selection, binary tournament selection, and rank selection. More comparative details on each selection approach is available in [126]. Variation operators including crossover and mutation methods, which are used to generate a new generation from the previously chosen one, are then used to combine the two selected parents to create new individuals for the offspring population. In the crossover, genes from each parent are separated and exchanged to produce an offspring having several genes from each parent. There are several possible strategies for a crossover which are randomly selecting a single point for the crossover, multi-point crossover, or uniform crossover. In mutation, an arbitrary gene is randomly changed from some randomly chosen offspring individuals to increase the variety of the offspring population by introducing new genetic information into the population. Finally, the newly evolved population is then used as the next population and loops back for evaluation. This new population contains both the parent's population and the offspring population and it has the size of the originally initialized population. The algorithm repeatedly performs modifications on the population of solutions until the termination conditions are satisfied and the final population, which explored the solution space and evolved toward the optimal solutions across several generations, is returned.

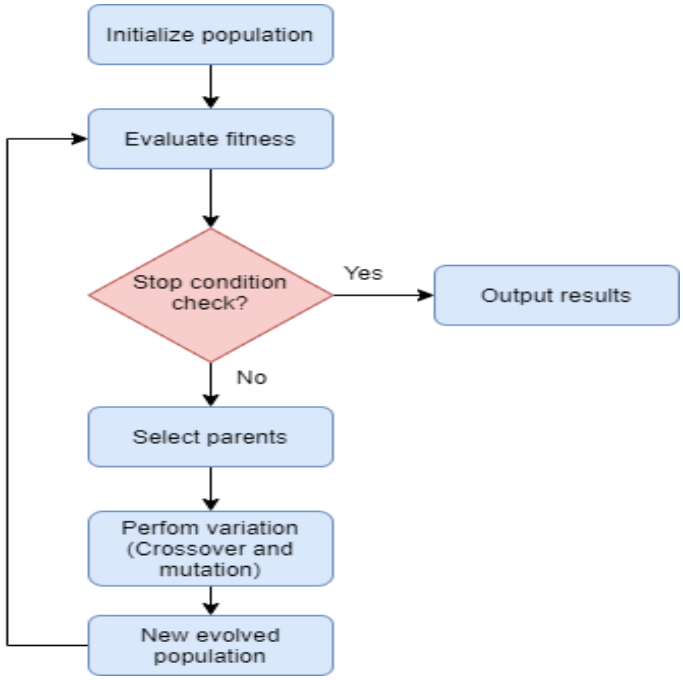

Figure 24. Flowchart of the Genetic algorithm (GA)

6.3 Experimental Settings

The CSTR process used in evaluating the proposed GA-IRS approach is executed in MATLAB R2021a/Simulink on a computer with Intel® Core™ i7-8565U CPU @ 1.8GHz and 16GB RAM memory. The optimization approach, which aims to maximize the reward function shown in Equation 12, is implemented using the inbuilt function for single-objective optimization (ga) from the optimization toolbox in MATLAB. However, since the ga function is only used for minimization problems, we set the problem to minimize the negative value of the reward function in order to achieve our desired objective. Table 14 displays the different settings of the parameters that are used in the objective function and the GA options. The choice of each parameter value is tuned to achieve the desired objective of protecting the CSTR system from going into hazardous states resulting from attackers exploiting the system's vulnerabilities. For example, since we only have 3 decision actions, which are controlling the opening and closing of each of the two considered valves and reconfiguring the setpoint, a suitable population size of 100 is set to avoid local minima solution. Also, it is worth mentioning that using a higher weight in the reward function for the temperature term $w_T = 0.8$ in comparison to the weight for the level term $w_L = 0.2$ indicates that our

objective function prioritizes temperature regulation over level since it is more costly to have a meltdown reactor attack.

Table 14. Parameter Settings for the Conventional GA Approach

Parameter	**Value**
M , t , P	3 , 2 (sec), 5 (\$)
w_T, w_L	0.8, 0.2
ε , α	10, 5
ref_T, ref_L	420 (k) , 1 (m)
HazardCost (C)	-3000
Population size, Max generations, Max Stall Generations	100, 100, 100
Function Tolerance	0.001
Actions lower bounds $[IV\ CV\ SP]$	$Lb = [0\ 0\ 300]$ (SP takes discrete values)
Actions upper bounds $[IV\ CV\ SP]$	$Ub = [1\ 1\ 500]$ (SP takes discrete values)
Termination conditions	Objective function value is less than the defined function tolerance or reaching the maximum defined number of generations

6.4 Case Studies

Different case studies are used to validate the performance of the proposed GA in solving the intrusion response decision-making problem under different attack scenarios. The considered attack scenarios are described earlier in Table 12. The objective of the optimization algorithm is to find the optimal action vector $A = [IV\ CV\ SP]$ that can defend against the cyberattack, maximize the reward function and bring the system closer to the setpoints and away from hazards. Table 15 describes the performed cases studies and the attack scenarios they dealt with. These case studies investigate how the GA selects the optimal action configurations to recover the process to its normal operations.

Table 15. Case Studies Description for the GA-IRS Appraoch

Case Study	**Attack Scenario**	**Description**
1 and 2	1	The current system state is S = [380,0.5]. The level is less than 30% but L_L is tampered in this attack scenario so the water pump doesn't turn on, and the level continues to drop. The same goes to the temperature.
3	2	The current system state is S = [436.8,1.88], the level is more than 70% but H_L is tampered in this attack scenario

Case Study	Attack Scenario	Description
		so the water pump does not turn off, and the level continues to increase until it brims over the tank
4	3	The current system state is S = [477,1.35], the temperature is more than 460 but H_T is tampered in this attack scenario so the coolant valve does not open, and the temperature keeps on increasing

6.4.1 Case Study 1

This is the base case study; it uses a random policy without optimization to show the random behavior of the CSTR model and compare it with the other optimized case studies to give some initial thoughts about the performance. For this case, the CSTR has a low level of 0.5 m and a low temperature of 380 k. Figure 25 shows the results of performing random actions for 100 runs. We can see that the system continuously selects random actions that lead to falling into hazards, showing a hazard cost (C) penalty of 3000. This behavior is not desired since we aim to protect the CSTR from falling into hazards and thus, optimize the performance.

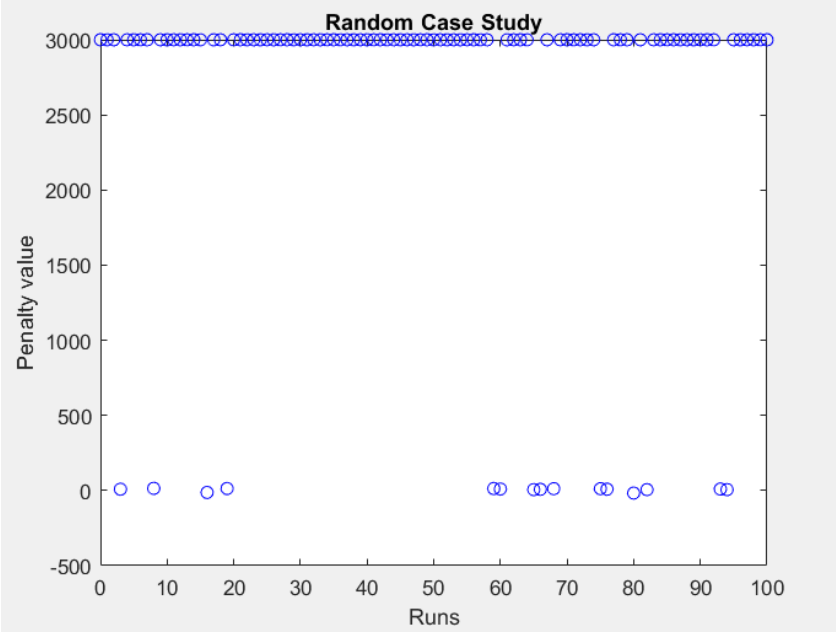

Figure 25. Case study 1: Random policy approach

6.4.2 Case Study 2

In the second case study, the CSTR environment was also set to simulate having a low level of 0.5 m and a low temperature of 380 k, as in the first study. Figure 26[1] shows the results of the GA optimization algorithm. The optimal action that was selected as the selected solution is $A = [1\ 1\ 422]$, which means to open the inlet valve, open the outlet valve, and adjust the temperature setpoint value to 422. As a result of deploying the selected action, the reactor's temperature settled at 420 k and the level at 0.6 m, which shows a normal behavior of both the temperature and level. In comparison to the base case study, we can see the effectiveness of the optimized solution in regulating the CSTR to its normal operations and keeping it away from hazards.

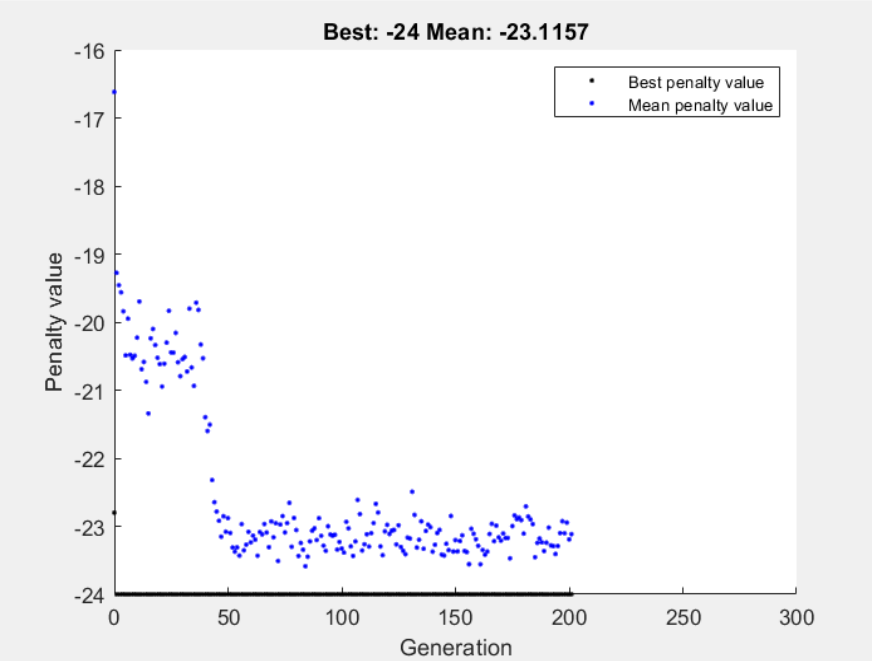

Figure 26. Case study 2: GA optimization for S = [380,0.5] scenario

6.4.3 Case Study 3

The CSTR environment simulates having a high level of 1.88 m, as a result of the fault setpoint attack previously described, and a normal temperature of 436.8 k. Figure 27

[1] This is the only case study that has a maximum generation of 200

shows the results of the optimization algorithm, which selected the best action as $A = [0\ 1\ 435]$. This action closes the inlet valve, which is a reasonable action since this scenario already suffers from a high level, so we aim to reduce the level by initially stopping the inlet water flow. Also, the action opens the coolant valve to regulate the temperature with a setting point of 435. After deploying the chosen action, the reactor's new temperature settled at 420 k and the level at 1 m, which are the exact desired defined setpoints for having optimal temperature and level values. This case study still shows a consistent superiority to the baseline case.

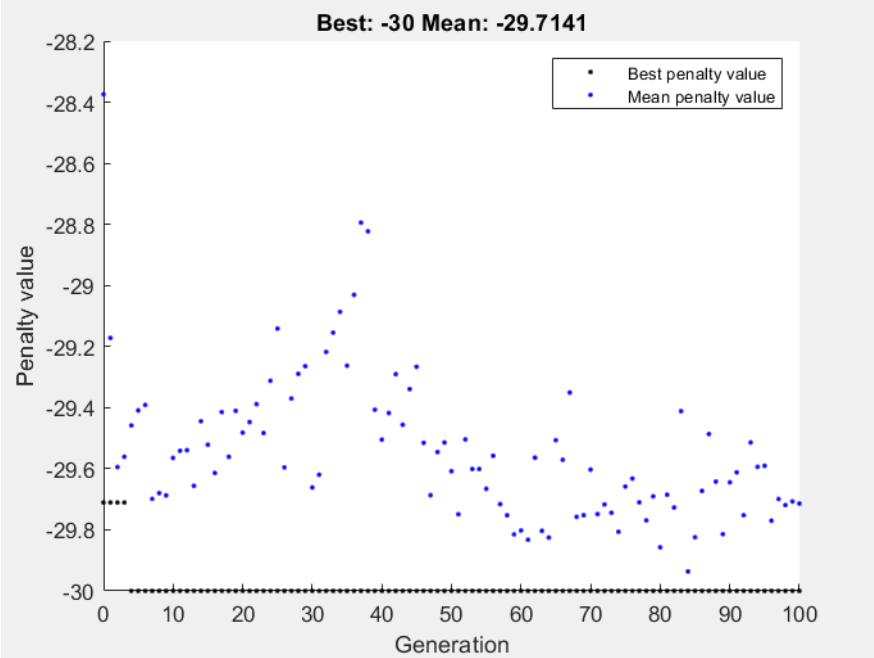

Figure 27. Case study 3: GA optimization for S = [436.8,1.88] scenario

6.4.4 Case Study 4

This case study simulated a CSTR environment with a high temperature of 477 k and a normal level of 1.35 m. Figure 28 shows the results of the optimization algorithm, which selected the best action that can be deployed as $A = [1\ 1\ 324]$. This action opens both the inlet and the coolant valves with a regulating setpoint value of 324. After deploying this action, the reactor's temperature settled at 434 k and the level at 0.9 m, which shows a normal behavior of both the temperature and level as desired. It is

noticeable that tuning the setpoint value plays a major role in regulating the CSTR and reducing the potential losses as seen from the presented case studies. Consequently, an attack targeting the tampering of these setpoints can cause catastrophic damages. Also, these case studies proved that their performance is effective in defending against attacks and obviously regulating the CSTR better than the random baseline case.

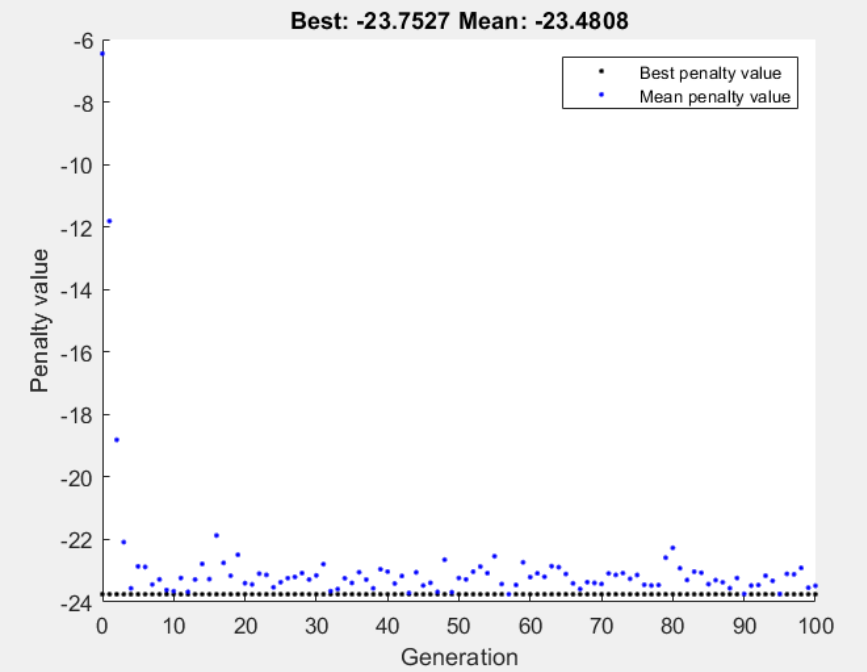

Figure 28. Case study 4: GA optimization for S=[477, 1.35] scenario

6.5 Evaluation

This section examines the impact of the proposed conventional GA decision-making algorithm in terms of computational time complexity and response action effectiveness in reducing the deviations of the reactor's temperature and level in the CSTR process.

Since the state and action spaces were small, it was not hard for the optimization algorithm to find the optimal decision for the different case studies in a reasonable period of time. Accordingly, the computational time complexity of this approach is not high since the case studies needed approximately 5 to 7 hours for the optimization to finish. This is obviously affected by the size of the state and action spaces. Thus, this metric is not comparable with other state-of-the-art approaches since each has its own simulated process

with its different considered state and actions spaces.

Concerning the GA-IRS approach effectiveness in reducing the temperature and level deviations, which is the main evaluation metric to be considered, Table 16 evaluates the temperature, level, and the percentage deviation for each case study before and after applying the selected response action. The deviation percentages are calculated using Equation 13 for the temperature, and Equation 14 for the level. From this table, we can state that the proposed approach successfully generated appropriate action policies that were able to reduce the deviations in all the different case studies and thus, maximize the reward function. In other words, the deviations were reduced by 9.5% and 10% for the second case study, 4% and 88% for the third case study, and 10.27% and 25% for the third case study, for the reactor's temperature and level, respectively. Also, we can also notice that the second, third, and fourth case studies outperform the random policy approach.

$$Temp\_dev = \left|\frac{temp - ref_T}{ref_T}\right| * 100 \qquad \text{Equation 13}$$

$$Level\_dev = \left|\frac{level - ref_L}{ref_L}\right| * 100 \qquad \text{Equation 14}$$

Table 16. Evaluating the Impact of GA Optimization in Reducing the Deviations

Case Study	**2**		**3**		**4**	
	Before	**After**	**Before**	**After**	**Before**	**After**
Temp (T)	380	420	436.8	420	477	434
T deviation	9.5%	0%	4%	0%	13.6%	3.33%
Level (L)	0.5	0.6	1.88	1	1.35	0.9
L deviation	50%	40%	88%	0%	35%	10%

Despite the successful performance of the GA solution approach in protecting the CSTR model from falling into hazardous states and minimizing the system's deviations, they are not suitable for real-world online applications. This is because this approach is a one-shot optimization that does not provide an adaptive solution. Usually, when real industrial systems are under attack, a sequence of non-identical actions are needed to bring the system to a stable state, not just a single optimal action execution. Also, this approach is not generalizable and

cannot handle changes in the system dynamics. Accordingly, there is a need for adaptive, generalized, and intelligent solution for solving the intrusion response decision-making problem, such as the model-free reinforcement learning-based approach discussed in the following chapter.

CHAPTER 7: IRS DESIGN USING DEEP REINFORCEMENT LEARNING (DRL-IRS)

In this chapter, the design methodology of the intrusion response agent using a model-free deep reinforcement learning approach is presented. We thoroughly discuss the proposed DRL-IRS architecture along with the details of its state space, action space, reward function, and utilized DDQN algorithm.

7.1 DRL-IRS Agent Architecture

Figure 29 depicts the high-level architecture of the proposed design of the intrusion response agent for a CPS using Double Deep Q Network (DDQN) algorithm. The architecture starts by getting the states information from the simulated CPS environment at every time step. These states are mapped to the input layer of the Deep Neural Network (DNN). The hidden layers of the Neural Network (NN), with the defined activation function, perform transformations of the inputs into something that the output layer can use. The output layer has N outputs, where N is the number of the available actions, such that each output corresponds to the Q-value of each potential action at the given state. The Q-value indicates how good is it to perform action a in state s. During testing, the agent acts by picking the action with the highest Q-value to be deployed. However, during training, the agent is encouraged to act randomly sometimes to explore the environment carefully and find the best possible sequence of actions to the goal. Accordingly, the action selection and the exploitation-exploration balance is handled by the decaying epsilon greedy policy, which will be discussed later. After deciding on the action, the DDQN agent directly interacts with the environment online to deploy the selected action. As a result, the environment moves to a new next state, and the agent gets a scalar reward/penalty value as feedback on performance to update the parameters of the deep network. From the reward value, the agent can assess how good or bad the deployed action was, so it can learn to take better actions in the future. Accordingly, the agent can learn depending on its own experience with the unknown environment.

The main objective of the proposed IRS agent is to learn an optimal policy, which is the strategy that the agent follows to take an action given the state, that maximizes the long-

term cumulative reward, and balances between exploring the environment and exploiting the agent's current knowledge. Now that we have seen the general broad picture of the proposed architecture, let's get into the details of each part.

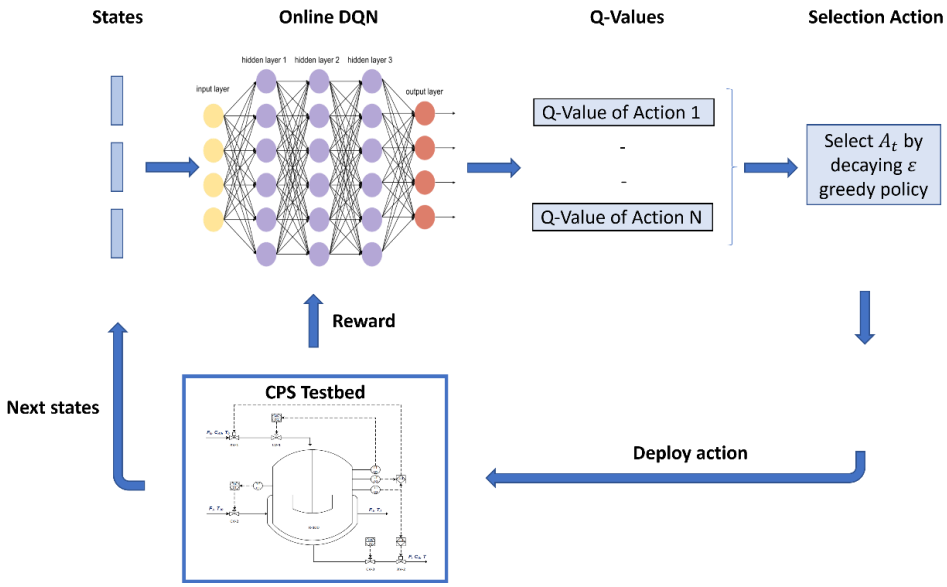

Figure 29: DRL-IRS Agent on a CPS testbed architecture

7.2 State Space

The states in reinforcement learning are the input information given to the agent to help it take decisions in different situations. A state-space can be either of discrete values or continuous values depending on the considered problem. In cyber-physical systems, the states are usually either process-based, cyber-based, or both. In our experiments, we integrated process-based and IDS-based states to enrich the agent visibility to the current state of the protected system. Also, integrating different categories of states exposes the agent to different training scenarios and thus, allows it to generalize and make accurate decisions in different situations.

The complete state space that we consider for our problem is defined to be $S = [T\ L\ OF\ AS]$. T and L are process-based states, which are the reactor temperature and the reactor level, respectively. OF is the outlet flow, which is a process-based state indicating the quantity of the produced output product in the CSTR process. Finally, AS is a state expected

from the IDS showing which attack scenario is being detected. To elaborate, we consider three different attack scenarios, which are described previously in Table 13, for the false data injection attack in which the IDS is assumed to be able to detect them successfully. Each attack scenario will have a different optimal sequence of actions that the agent has to learn from interacting with the testbed to reach convergence. Thus, at time step t, the state vector is $s_t = [t_t\ l_t\ of_t\ as_t]$, which encompasses only discrete variables for all the state parameters.

7.3 Action Space

The actions in reinforcement learning are the decisions that the agent takes as an output based on the given states. An action-space can also be of discrete or continuous values. In CPSs, the actions can be implemented only on the cyber-level, the process level, or both at the same time. In the literature, the combination of both cyber-level and process-level actions in one action vector is not addressed thoroughly. For this reason, we targeted to include this as one of the action combinations in our actions list.

The complete action space that we consider for our problem to mitigate the false data injection attack has several atomic actions, such as dropping attack packets, safety shutdown using the three safety valves, setpoint reconfiguration, and manual shutdown. Accordingly, our action vector is defined to be $A = [DP\ SI\ SC\ SO\ SP\ MS]$, where DP is a cyber-level action to drop attack packets from the network, if attack source is known. SI, SC, and SO are the safety shutdown actions for the three valves in our process, which are the inlet valve, the coolant valve, and the outlet valve, respectively. The SP corresponds to the setpoint regulating action, and MS decides whether the manual shutdown action is activated or not. Thus, at time step t, the action vector is $a_t = [dp_t\ si_t\ sc_t\ so_t\ sp_t\ ms_t]$, which encompasses only discrete variables since all the actions can either be 0 or 1, except sp_t which is set to be 420. For the three safety valves, 0 means closing the valves and 1 means opening them. Also, for dp_t and ms_t, 0 means not to drop attack packets and not to perform manual shutdown while 1 means to drop the packets and perform manual shutdown, respectively. It is worth noting that the choice of the actions contributing to the action space are process dependent.

Also, the atomic actions that the action vector A is made up from are used to build 6 different combinations of discrete actions. Each combination can include either one or more than one action at a time and are designed to suit the different training experiments conducted. This will be thoroughly discussed in detail training section.

7.4 Reward Function

The reward is a scalar value that assesses the performance of the agent during the training period. The ultimate aim of the agent is to maximize the long-term reward received from the environment. The reward function encapsulates and encodes the goals that the agent aims to achieve. There are three approaches to formulate the reward function, which are continuous rewards, discrete rewards, and mixed rewards [127]. The continuous reward is usually improving the convergence allowing for simpler network configurations, the discrete reward usually guides the agent to avoid specific state regions but with slower convergence, and the mixed reward combines the advantages of each type. In here, we discuss the three goals that derive the formulation of our reward function.

Initially, the first goal of the agent is to assess the quality of the produced product from the CSTR process. The product quality is assessed by two indirect measurements, which are the temperature and the level of the reactor. To elaborate, as long as the product is produced while the reactor maintains its temperature and level around the defined setpoints then, we can say that the produced product has good quality. Accordingly, the product quality degrades as the temperature and level deviate away from their defined setpoints. It is worth mentioning that using the concentration variable of the produced product would have been a more appropriate measure to assess its quality. However, we did not use it because it was unstable during training hence, using it would have not given reliable feedback. The second goal is to assess the quantity of the produced product. This is assessed by considering the outlet flow value, which represents the amount of the produced product, such that the quantity is directly proportional to the reward. The third and final goal is to prevent the CSTR process from falling into hazardous states. Thus, the temperature and the level should not exceed the upper and lower boundaries (hazard states) defined in Table 11. Otherwise, the agent receives a very high negative penalty of -3000 (C). Accordingly, the reward function that fulfills these goals is formulated as shown below (

Table 17 shows the symbols description)

$$r_1 = \left(N * OF * (PP * \mathrm{e}^{-TTR})\right) + P_{\mathrm{MS}} \quad \text{Equation 15}$$

$$PP = Z * P * t \quad \text{Equation 16}$$

$$P_{MS} = \begin{cases} -\mathrm{TTR} - \mathrm{Acost}, & \text{if MS} == 1 \\ 0, & \text{otherwise} \end{cases} \quad \text{Equation 17}$$

$$r_2 = -\mathrm{C}\,(T \leq 360 \mid T \geq 480 \mid L \leq 0.2 \mid L \geq 1.9) \quad \text{Equation 18}$$

$$R_{DRL} = r_1 + r_2 \quad \text{Equation 19}$$

Table 17. Reward Symbols Description for the DRL-IRS approach

Reward Symbol	**Description**
N, Z	Scaling factors
OF	Outlet flow state
PP	Maximum production profit the process can get at each time step
P	Profit per sec
t	Sampling time
TTR	Time to recover (calculated using Equation 11)
P_{MS}	Penalty the agent gets only when performing the manual shutdown action (MS),
Acost	Availability cost penalty
C	Hazard cost
T	Reactor's temperature
L	Reactor's level

Our reward function is of a mixed type that combines both continuous and discrete reward components. r_1, which is the continuous signal, is used to provide a higher reward when the quality and the quantity of the produced product are near target ideal values. The quality is shown by the TTR distance metric, in which better quality shows a lower TTR value and vice versa. The quantity, on the other hand, is captured by the OF parameter that gives a higher reward as it increases. Also, r_1 encounters the manual shutdown action penalty, which is a terminal state. r_2, which is the discrete signal, provides a large penalty to drive the system away from hazardous conditions. In case of falling into a hazardous state, which is also a terminal state in our experiments, r_2 would be much higher than r_1. This is because even if there is a produced quantity of product captured by r_1, this product is a waste and would not have been used by the process due to its possible toxicity. All in all, the total reward received at each time step is given by Equation 19.

7.5 Double Deep Q Network (DDQN) Algorithm

In practical large-scale real-life systems, such as our CSTR testbed, it is unfeasible to precisely model the complex dynamics of the process by knowing all the transition probabilities. Accordingly, this calls for efficient and intelligent approaches that do not require the transition probabilities to learn the optimal solution for optimization. Fortunately, RL supports model-free approaches where the agent can learn the optimal policy without requiring an accurate model of the environment. Precisely, Q-Learning algorithm is one of the most commonly used model-free algorithms [128]. However, the tabular approach of the Q-learning algorithm, which suffers from the curse of dimensionality with high dimensional state/action spaces, limits its applicability to a narrow range of applications. Subsequently, several variants of the Q-learning algorithm that utilizes the advantages of NNs have been developed to overcome these limitations, such as the DDQN algorithm.

The DDQN algorithm is an off-policy, model free, online, value-based RL algorithm. It utilizes deep NNs, which act as a function estimator, to improve generalization, allow for large state/action spaces, and reduce the complexity of training complex environments. DDQN algorithm uses the exact same procedures of the popular Deep Q Network (DQN) algorithm, but with an additional independent network for Q-value estimation, that is where the term 'Double' came from. The usage of the extra identical target network allows DDQN to avoid the maximum estimation bias issue found in DQN and ensures a faster, robust, and more stable learning than DQN.

Figure 30 shows the general framework of how DDQN works, which will be thoroughly explained in the Algorithm steps in the following paragraph. The first row of the framework is previously explained when discussing the DRL-IRS architecture in Figure 29. Hence, continuing on the DRL-IRS architecture, A replay buffer is used to store the experiences in the form of $(s_t, a_t, r_{t+1}, s_{t+1})$ tuples, which are collected by the agent from the interaction with the environment. It is worth noting that oldest tuples are deleted in case the buffer was full, so that they can be replaced by new experiences. For training the agent, a mini-batch is used with two NNs: the online critic DQN and the target network. Introducing a

second target network, which is an exact replica of the critic network, helps in stabilizing the learning by breaking the correlation of errors that happens when using only one NN. The role of the online DQN is to make all the decisions and choose which action the agent is going to take in every step. Whereas the target network evaluates the action and decides how valuable it is. More details on the algorithm steps are given below:

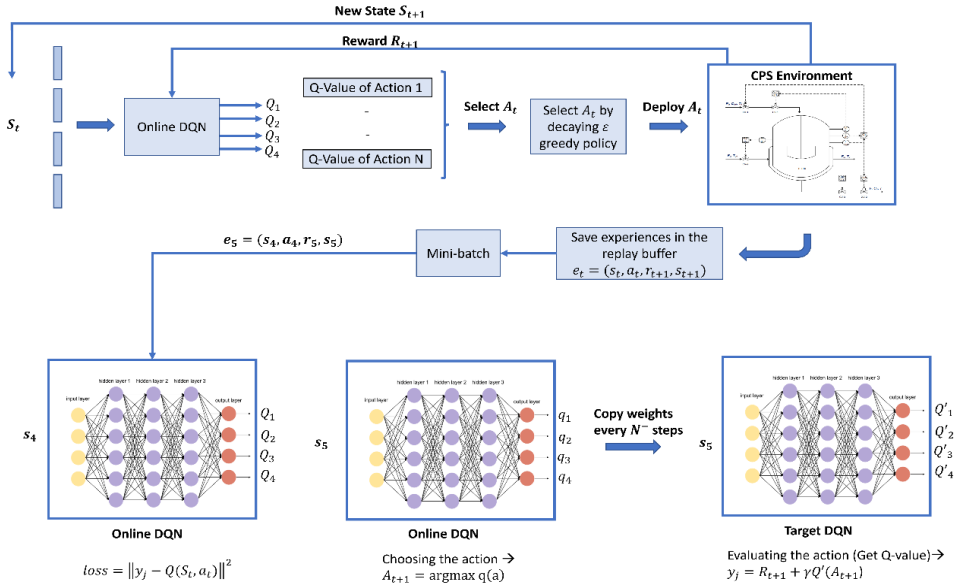

Figure 30. General framework of the DDQN algorithm

Algorithm 1 [129] illustrates the details of the DDQN algorithm that show how the agent learns. Initially, the agent initializes several input parameters, including a replay memory buffer D of maximum size N_r, the online critic network parameters θ, the target network parameters θ^-, a training batch size N_b, and the target network replacement frequency N^-. The two initialized networks: the online critic network and the target network, are given the exact same initial weight parameters. For each training step, the agent decides on whether to choose a random action or perform the action suggested by the online network that has the maximum Q-value based on the decaying epsilon greedy policy. After executing the action, the agent receives a reward and the next state from the environment. The transition experience tuple is also stored in the replay buffer. Following that, a random minibatch of size N_b is sampled from the replay buffer and fed to the online NN to optimize on its parameters.

The target network is used to compute the target Q-value for each of the N_b tuples by accounting for terminal states. To elaborate, if it was a terminal state, then we are sure that there is no future reward for us to look forward to, and so the second term in the target value function formula y_j is set to 0. If it is not a terminal state, the critic network estimator θ is used first to select the action by performing the $argmax$ operation then, the second target network estimator θ^- is used to evaluate that action by getting its Q-value. Afterwards, the Mean Square Error (MSE) is computed, which is the difference between the target Q-value y_j and the predicted Q-value that is computed using the critic network θ. Finally, the parameters of the critic network undergo gradient update to update the network's weights by minimizing the loss function. It is worth noting that the target network is not trained hence it never undergoes gradient updates since it is only used to guarantee that the predicted Q-values and the target Q-values would be computed via two separate independent networks to gain training stability. At the end, the weight parameters of the critic network are copied into the target network at regular intervals every N^- steps.

Algorithm 1: Double DQN Algorithm.

input : $\mathcal{D}$ – empty replay buffer; θ – initial network parameters, θ^- – copy of θ
input : N_r – replay buffer maximum size; N_b – training batch size; N^- – target network replacement freq.
for *episode* $e \in \{1, 2, \ldots, M\}$ **do**
  Initialize frame sequence $\mathbf{x} \leftarrow ()$
  for $t \in \{0, 1, \ldots\}$ **do**
    Set state $s \leftarrow \mathbf{x}$, sample action $a \sim \pi_{\mathcal{B}}$
    Sample next frame x^t from environment $\mathcal{E}$ given (s, a) and receive reward r, and append x^t to $\mathbf{x}$
    if $|\mathbf{x}| > N_f$ **then** delete oldest frame $x_{t_{min}}$ from $\mathbf{x}$ **end**
    Set $s' \leftarrow \mathbf{x}$, and add transition tuple (s, a, r, s') to $\mathcal{D}$,
      replacing the oldest tuple if $|\mathcal{D}| \geq N_r$
    Sample a minibatch of N_b tuples $(s, a, r, s') \sim \text{Unif}(\mathcal{D})$
    Construct target values, one for each of the N_b tuples:
    Define $a^{\max}(s'; \theta) = \arg\max_{a'} Q(s', a'; \theta)$
    $y_j = \begin{cases} r & \text{if } s' \text{ is terminal} \\ r + \gamma Q(s', a^{\max}(s'; \theta); \theta^-), & \text{otherwise.} \end{cases}$
    Do a gradient descent step with loss $\|y_j - Q(s, a; \theta)\|^2$
    Replace target parameters $\theta^- \leftarrow \theta$ every N^- steps
  end
end

CHAPTER 8: DRL-IRS TRAINING RESULTS AND EVALUATION

In this chapter, we discuss the experimental settings, including the NN architecture, network parameters, DDQN settings, and training episodes. Also, we present the details of the three different training experiments conducted (Exp1, Exp2, and Exp3). Then, the learning performance results of each training experiment is examined, discussed, and evaluated. Finally, the chapter ends with brief information about the collected dataset and the challenges of developing this thesis.

8.1 Experimental Setup

8.1.1 Deep Neural Network Architecture

The adopted Neural Network for the DDQN agent consists of one critic network, which has two fully connected hidden layers. The number of neurons in the first layer is 150 and the second layer is 100 for Exp1. However, in Exp2 and Exp3, the used neurons are 50 and 25 for the first and second hidden layers, respectively. For the activation function, the rectified linear (ReLU) activation unites are utilized. The size of the input layer is the same as the number of states, while the size of the output layer equals to the number of considered actions. Each output from the critic network represents the Q-value, which is a single linear unit, of the given state with a specific action. In order to optimize the network parameters with a learning rate of 0.001, Adam optimizer is employed. To avoid overfitting, the L2 regulation is used with a value of 0.002. Table 18 shows the architecture, which was fine-tuned with trial-and-error process to decide on its different parameters.

Table 18. Neural Network Architecture

Neural Network	**Number of Layers**	**Structure (number of neurons in each layer)**	**Activation Function**
Critic network (Target network has the same structure)	4 fully connected layers (input layer, 2 hidden layers, and the output layer)	Dependent on the different experiments conducted	ReLu

8.1.2 Replay Buffer

A replay buffer is used to store the agent's experiences in memory to be sampled in a minibatch that is used for the training to optimize the network parameters. The usage of a random minibatch approach in training is important snice it breaks the correlation between the samples, provides better sample efficiency, and allows the agent to see each sample more than once before being removed from the memory. Figure 31 shows the concept of using the replay buffer experiences in optimizing the network. Concerning the size of the replay buffer, it is set to be 1,000,000 in our experiments since large sizes are better to ensure that the samples are not thrown away quickly.

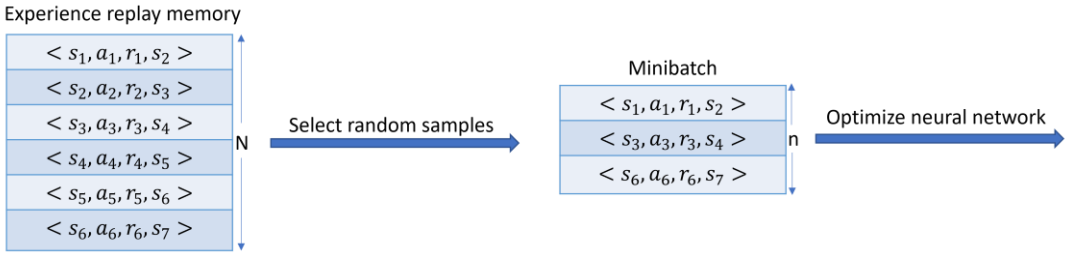

Figure 31. Mini-batch training

8.1.3 Training Episodes

An episode is the path that the agent follows from an initial state to a terminal state. There is a maximum number of episodes that has to be defined for the training session. In our experiments, an episode terminates when either the system reaches a hazardous state, which happens when the rector's temperature and level states enter their respective hazard zone, when manual shutdown happens, or when reaching the maximum defined number of steps per episode. The manual shutdown is considered a terminating state because the agent cannot progress anywhere from it. Accordingly, these cases define our termination criteria. The maximum number of steps per episode is another essential hyperparameter that needs to be set for effective training. It is important to be tuned in a way that allows the agent to explore the state space carefully. Thus, a maximum of 100 steps/episode were used in Exp1 and 300 steps/episode were used for Exp2 and Exp3.

8.1.4 Decaying Epsilon Greedy Approach

The action selection process at time t in RL is facing an exploration-exploitation dilemma, which is simplified as shown in Figure 32. Exploration encourages the agent to act randomly to explore all the different actions in different states. On the other hand, exploitation chooses the action with the highest known q-value at a given state. Always choosing one method over the other is not advised because only exploring will not drive the agent to the optimal solution and always exploiting can likely cause the agent to be stuck at a local optimum. Accordingly, the decaying epsilon greedy approach comes to balance the tradeoff between exploration and exploitation.

Decaying epsilon greedy approach chooses between exploration and exploitation based on the value of epsilon ε, which refers to the probability to explore. Initially, we want the agent to explore thoroughly the action space, so we set epsilon to be high. However, as time passes, we would want the agent to use its knowledge in choosing the action with the highest Q-value in each state. Thus, the ε keeps on decaying by a defines epsilon decay rate as time passes until it reaches the minimum defined epsilon value. For example, if $\varepsilon = 0.4$ then we are selecting random actions with a probability of 0.4 regardless of the established actual q value and exploiting the agent knowledge in choosing the best-known action with a probability of 0.6.

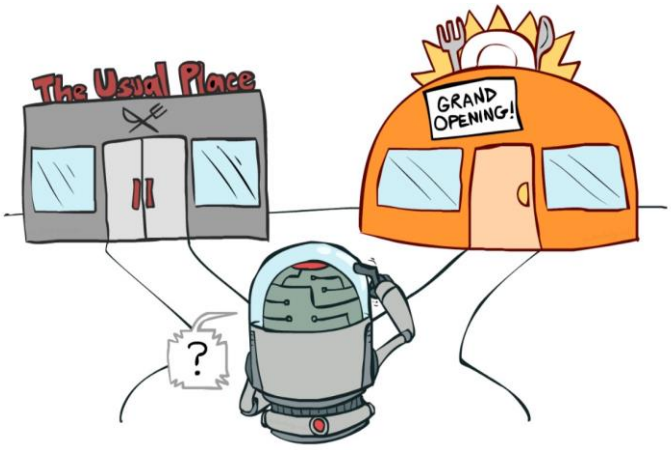

Figure 32. Exploration-Exploitation dilemma [130]

8.2 Agent Training Experiments

In this section, we present the different training experiments that were conducted on the testbed. Table 19 presents a summary of the performed experiments, including their considered attack scenarios, which are explained in Table 13, the state space, the action space, and the reward function. Notably, we start with the simple experimental setup then, add more attack scenarios, states, and actions as we move from one experiment to another. Table 20 summarises the used parameters in each experiment.

Table 19. DRL-IRS Agent Training Experiments Description

Experiments	**Attack Scenarios Considered**	**State Space**	**Action Space**	**Reward Function**
Exp1	Scenario 1	$S = [T\ L\ ]$	4 actions: $A_{exp1} = [\ SI\ SC\ SO]$	Equation 12
Exp2	Scenario 1 and 2	$S = [T\ L\ OF\ AS]$	5 actions: $A_{exp2} = [DP\ SI\ SC\ SO\ SP]$	Equation 19
Exp3	Scenario 1, 2, and 3	$S = [T\ L\ OF\ AS]$	6 actions: $A_{exp3} = [DP\ SI\ SC\ SO\ SP\ MS]$	Equation 19

8.2.1 Experiment 1 (Exp1): Training results

In the first experiment, we aimed to have a simple setup to examine how the agent will react with only one attack scenario. The state-space only consisted of the reactor's temperature and level values $S = [T\ L]$. The action-space only considered the three actions that manipulate the inlet, coolant, and outlet safety valves $A_{exp1} = [\ SI\ SC\ SO]$. Using these three Boolean actions, we could have made 8 different combinations. However, only 4 different combinations for opening (Boolean=1) and closing (Boolean=0) the valves were used. The 4 actions manipulating the safety valves are $a_1 = [0\ 0\ 0]$, $a_2 = [0\ 1\ 0]$, $a_3 = [1\ 0\ 1]$, and $a_4 = [1\ 1\ 1]$. These actions are chosen carefully after making sure that the process is maintaining its stability when implementing

any of them. On the other hand, the unchosen action combinations were throwing the process off-guard, and the process had to be restarted every time any of these actions are performed. Accordingly, we eliminated them from the action space to simplify the training procedure. For the reward function, Equation 12 was used for this experiment. This reward function only assessed the quality of the produced product by considering the TTR metric. Consequently, the agent aims to learn how and when to use these 4 actions, which manipulate the status of the safety valves, to prevent the process from falling into hazardous states and achieve the optimum product quality by keeping the process operating around the ideal defined setpoints as much as possible.

Figure 33 shows the learning performance of the DDQN algorithm in the first experiment. It captures the episodic reward and the average reward of 30 successive episodes of the learning process by the blue curve and the red curve, respectively. It is demonstrated from the figure that the agent was falling into hazards repeatedly while exploring the environment in the first 2400 episodes. This is shown by the fluctuated episodic reward behavior of the learning curve. The early fluctuation in the behavior is expected since the model-free agent was still exploring the environment, and the action policy was not optimized yet. Starting from episode 2400 until episode 2700, the agent was experiencing higher rewards most of the time. However, the behavior of the agent was not stable since it was still violating the hazard protection goal of the process. Approximately at episode 2700, the agent exhibited a stable performance, showing that the agent successfully converged to the optimal action policy. It is worth mentioning that our criteria for stopping the training phase for this experiment and declaring that the agent converged are: (1) having the average reward to be around the maximum known reward value (3000) by a maximum of 20% and (2) having a stable behavior for at least 500 consecutive episodes.

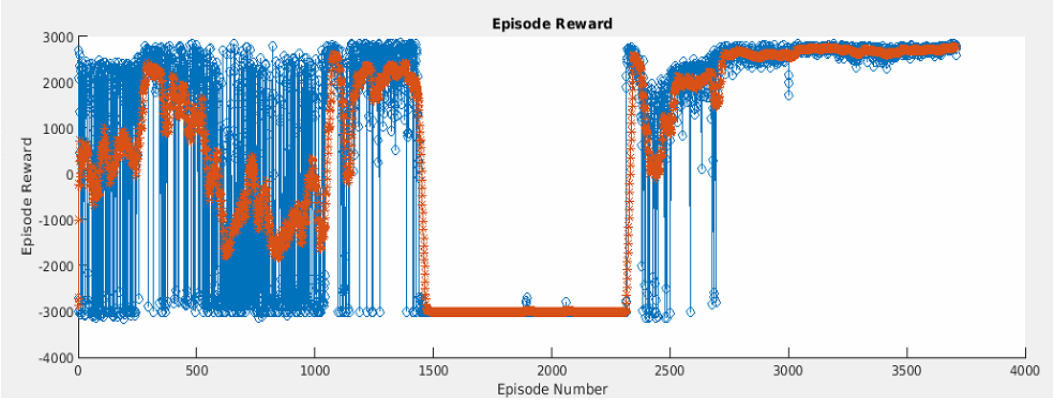

Figure 33. Experiemnt 1: Learning curve

8.2.2 Experiment 2 (Exp2): Training results

In the second experiment, we added one more attack scenario, two more states, and one more cyber-action to increase the function usability of the agent. The state-space consisted of the temperature, level, outlet flow, and an identifier (either 1 or 2) expected from the IDS to represent the type of the detected attack scenario $S = [T\ L\ OF\ AS]$. The action-space consisted of the previous 4 actions presented in Exp1 with an additional action defined as $A_{exp2} = [DP\ SI\ SC\ SO\ SP]$ = $a_5 =$ $[1\ 1\ 1\ 1\ 420]$. This action is implemented when the attacker source address is known so that it drops attacker packets, opens the three safety valves, and recovers the attacked setpoint value by reconfiguring it to 420 to return the process to its normal operations. It is important to know that the choice of the setpoint regulating value is process-dependent. Initially, we planned to have several actions with different setpoint values for the agent to decide which value to use. However, we wanted to reduce the action space as much as possible and thus, hopefully, reduce the training time. Also, to reduce the training time, we used a fewer number of neurons in the hidden layers, upgraded the reward function to give more representative values, and decreased the agent's sampling time from $5\ s$ to $2\ s$. For the reward function, Equation 19 was used for this experiment. This reward function assessed the quality and the quantity of the produced product by considering the TTR metric and the outlet flow value, respectively. Consequently, the agent aims to learn how and when to use these 5

actions, which either manipulate the status of the safety valves or select the cyber action to maximize the reward function and achieve the agent's objective in responding to cyberattacks in the most optimal way.

Figure 34 presents the learning performance of the DDQN algorithm in the second experiment. As previously mentioned, the agent usually performs badly in the first few episodes due to the exploration phase. At episode 200, the agent started exploiting its knowledge in selecting the actions that give better rewards depending on the considered state. Approximately at episode 400, the agent exhibited a stable performance showing that it converged to an action policy. However, it seems that the agent converged to a sub-optimal action policy. This is notable from the reward curve since the episodic rewards for attack Scenario 1 are approaching a value of 2000 when they should have been optimally approaching a value of 3000. It is worth mentioning that we stopped the training phase after having a stable behavior for 500 consecutive episodes.

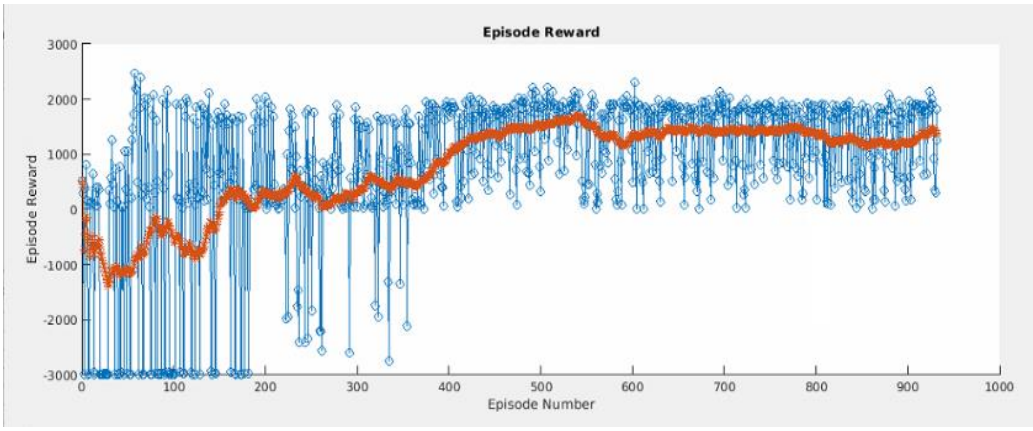

Figure 34. Experiment 2: Learning curve

8.2.3 Experiment 3 (Exp3): Training results

In the third experiment, we considered all the attack scenarios, the full state space $S = [T\ L\ OF\ AS]$, and the full action space $A_{exp3} = [DP\ SI\ SC\ SO\ SP\ MS]$ to improve the applicability of using the agent in different situations. The action space consisted of 6 actions; the five actions presented in the previous experiment with an additional action

$a_6 = [0\ 0\ 0\ 0\ 0\ 1]$, which disables/closes everything and decides to perform the manual shutdown action. Using the manual shutdown action indicates that all other automatic actions that the agent tried in this specific state failed to recover the system. Since the agent is not applicable to perform the manual shutdown itself, we assume that when it is selected as being optimal in a specific state, a human operator will be able to successfully perform the manual action and secure the system. Otherwise, the process will fall into hazards and suffer from costly environmental and financial losses. For the reward function, Equation 19 was used to assess the quality and quantity of the product.

Figure 35 shows the learning performance of the DDQN algorithm in the third experiment. The first 630 episodes show the initial exploring phase of the agent that results in fluctuated reward values. However, between episode 630 and episode 1400, the agent obtained higher rewards most of the time but with an unstable behavior. Approximately at episode 1400, the DDQN agent demonstrated a stable performance for more than 500 consecutive episodes indicating that it optimized the action policy successfully. In this experiment, we also used a smaller number of neurons in the DN hidden layers with a decreased agent's sampling time to reduce the training time. For this experiment, we cannot anticipate where the optimal average reward curve should be because this experiment includes three different scenarios each with different optimal responses and reward values. For example, the optimal action for the third attack scenario is taking the manual shutdown action, which has a negative reward value, unlike Scenarios 1 and 2. Accordingly, we stopped the training phase when we witnessed stable behaviour for 500 consecutive episodes and evaluated the effectiveness of the trained agent responses for each attack scenario in the testing phase.

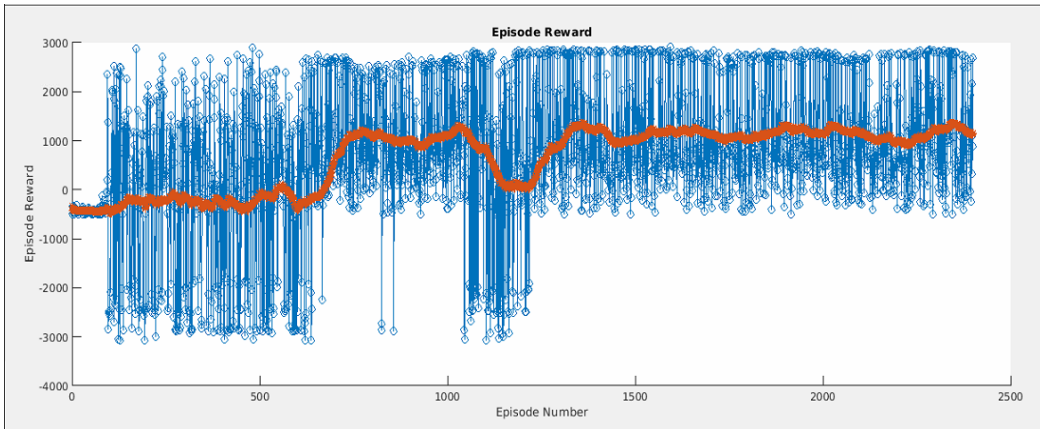

Figure 35. Experiemnt 3: Learning curve

Table 20. Parameters Settings Summary

Parameter	**Value**
Size of hidden layers (HL1, HL2)	$150, 100$ for Exp1 $50, 25$ for Exp2 and Exp3
Activation function of hidden layers	ReLu
Minibatch size	256
Experience buffer length	1,000,000
Target smooth factor	0.003
Learning rate	0.001
Discount factor	0.99
Maximum and minimum value of exploration	$1, 0.1$
Epsilon decay rate	0.005
Maximum episodes	10,000
Maximum steps per episode	100 for Exp1 300 for Exp2 and Exp3
Window length for averaging	30 for Exp1 50 for Exp2 100 for Exp3
Reward constants: $N, M, Z, P, t, C, Acost$	$NA, 3, NA, 5, 2, 3000, NA$ for Exp1 $\frac{1}{3}, NA, 2.5, 3, 2, 3000, NA$ for Exp2 $\frac{1}{3}, NA, 2.5, 3, 2, 3000, 500$ for Exp3

8.3 Agent Testing and Evaluation

In this section, we evaluate the trained agents of the different conducted experiments in terms of their response effectiveness and computational time. To evaluate the response effectiveness of each trained agent, we run the agent N times on our CSTR testbed to defend against different simulated attack scenarios. Each run starts at a random state in the environment and returns the total reward obtained. Following that, we calculate the average

return from the N runs to assess the effectiveness of the agent in responding to the different attack scenarios by choosing a sequence of actions that maximize the reward function.

8.3.1 Testing the trained agent of Experiment 1

In experiment 1, the agent has only one attack scenario to deal with (Scenario 1), which assumes a non-compromised safety controller and an unknown attack source. We simulated this attack scenario on our CSTR testbed and tested the performance of the agent in choosing suitable defensive response actions for $N = 10$ runs (each run has 100 steps). Table 21 shows the total reward obtained after each run with an average reward of 2739. We noted from these 10 runs that the agent selects a sequence of non-identical actions to keep the process operating around the ideal temperature value $(420\,k)$. Accordingly, action $a_2 = [0\ 1\ 0]$, which closes both the inlet and outlet valves, was repeatedly chosen when the reactor's temperature was increasing and approaching the hazardous region. As a result of performing action a_2, the temperature began to decrease. This is expected since it is a common practice to perform the safety shutdown in case of reactor high-temperature hazards in model-based solutions [53]. The agent kept on monitoring the state of the temperature, and if it decreased more than intended to a value far below our $420\,k$ reference, the agent selected another action $a_4 = [1\ 1\ 1]$ to reopen the valves. This pattern of actions was repeated throughout the 10 runs with a clear aim of controlling and regulating the process to work around the referenced operating points and thus, maximizing the reward function. The pattern of the selected actions and the obtained average total reward, which is very close to the ideal reward value $(3000/epsiode)$, proves the effectiveness of the trained agent.

The computational time of the DDQN algorithm in training experiment 1 was very high since it took approximately 5 days for the agent to reach its convergence state, which is a very long training period. This is a complex computational efficiency considering it is trained on a small-scale system.

Table 21. Experiment 1: Perfromance Evaluation

Run	Total Reward
1	2612
2	2780
3	2765
4	2790
5	2830
6	2842
7	2502
8	2684
9	2787
10	2800
Average Reward	**2739**

In addition to the above evaluations, we compared the performance of the DRL-IRS agent of Experiment 1 with the performance of the fourth case study from the GA-IRS solution. The fourth case study was simulated with the same attack scenario and was using the same reward function to solve the intrusion response decision-making problem. However, it was solved using a Genetic Algorithm approach. Both algorithms' performance is analyzed and compared in terms of response effectiveness in stabilizing the process away from hazards and closer to ideal setpoints.

As we observed from Table 16, the GA-IRS solution for case study number 4 stabilized the temperature at $434\ k$ and the level at $0.9\ m$. Using these values in Equation 12 obtained a reward of 23.75, which is shown in Figure 28 as being the best reward value achieved by the optimization algorithm. Assuming that we ran this experiment for 100 steps, just like the DRL-IRS experiment, then the total reward obtained from this approach is 2375. As can be seen, the adaptive solution of selecting non-identical sequence of actions, which is proposed by the DRL-IRS, shows better performance since it obtained a higher average reward of 2739. This indicates that the DRL-IRS solution was more effective in stabilizing the reactor's temperature around the ideal operational window. However, the GA-IRS was much faster in comparison to the DRL-IRS approach.

8.3.2 Testing the trained agent of Experiment 2

In experiment 2, the agent has two attack scenarios to deal with (Scenario 1 and 2), which are described in Table 13. Also, an additional cyber action to drop attack packets and reconfigure the setpoint is added to its action space. We simulated each attack scenario on the testbed and evaluated the trained agent for 10 runs (each run has 300 steps). Table 22 shows a summary of the different obtained runs. We observed that the agent mainly selects action $a_5 = [1\ 1\ 1\ 1\ 420]$, which drops attack packets and reconfigures the setpoint, in the sequence of chosen actions when the state indicates that the detected attack is Scenario 1. For attack Scenario 2, the agent repeatedly performs the safety shutdown action $a_2 = [0\ 1\ 0]$ when the temperature is approaching a high value. This action leads to a reduction in the temperature until it reaches a stable state. Then, the agent might consider to re-open the valves based on the current system state. If the agent decides to keep the valves closed for safety reasons, then the worst reward value that could be obtained for Scenario 2 is 0 because closing the valves means stopping the production line. However, this means that the testbed worth is gained since the agent's actions stopped it from falling into dangerous hazard situations that could have led to huge financial and human losses. This pattern of actions was repeated throughout the 10 runs with a clear aim to keep the process away from hazards, and close to defined setpoints. However, as we noticed from the learning curve in Figure 34, the agent might have converged to a suboptimal action policy. The agent clearly was able to achieve the third reward goal, which is protecting the system from hazards. However, the first and second goals of the reward function, which encourage the agent to select the optimal sequence of actions that gets the system as close as possible to the defined setpoints, are questionable. Even though reaching global optimal solutions are usually not guaranteed, and local optimal solutions are generally acceptable, we believe that this behavior might have been because the agent needed more training time to reach a better action policy.

The computational time of the DDQN algorithm in the training phase of experiment 2

took approximately 1.5 days to reach convergence, which is faster than the Exp1.

Table 22. Experiemnt 2: Performance Evaluation

Attack Scenario	**Run**	**Total Reward**
1	1	1904
	2	1852
	3	2111
	4	2033
	5	1863
	6	1633
	7	1812
	8	1849
	9	1612
	10	2107
	Average Reward	**1877**
2	1	0
	2	145
	3	1525
	4	445
	5	1033
	6	301
	7	280
	8	657
	9	481
	10	639
	Average Reward	**550**

8.3.3 Testing the trained agent of Experiment 3

In experiment 3, the agent has three attack scenarios to deal with, which are described in Table 13. To make sure that the trained agent can deal with each scenario optimally, we simulated each scenario 10 times and evaluated the performance of the agent in each run. Table 23 shows the total reward obtained after each run for the simulated attack scenarios. In attack Scenario 1, the agent was observed in the different runs selecting the cyber action to drop the attack packets and reconfigure the setpoint $a_5 = [1\ 1\ 1\ 1\ 420]$. In attack Scenario 2, since it assumes that the attacker is not known, the agent decided to manipulate the opening and closing of the three safety valves using actions $a_2 = [0\ 1\ 0]$ and $a_3 = [1\ 0\ 1]$ to stabilize the process around the ideal operational temperature value $(420\ k)$. In attack Scenario 3, which is the most powerful attack that compromises the safety controller and keeps its source hidden, the manual shutdown action was always

chosen by the agent as the optimal action to protect the system. The results obtained from this experiment indicate that the agent was able to successfully protect the system from falling into hazards in all the three considered attack scenarios.

The computational time of the DDQN algorithm in the training phase of Exp3 was still high since it took approximately 4 days to converge. However, it is considered an improvement in comparison to Exp1 since it has triple the size of its state space due to including more attack scenarios. This highlights the issue of scalability since the agent takes long time to train a relatively small-scale system.

Table 23. Experiment 3: Perfromance Evaluation

Attack Scenario	**Run**	**Total Reward**
1	1	2456
	2	2673
	3	2209
	4	2892
	5	2340
	6	2830
	7	2659
	8	2794
	9	2789
	10	2838
	Average Reward	**2648**
2	1	534
	2	439
	3	981
	4	602
	5	545
	6	819
	7	171
	8	238
	9	336
	10	145
	Average Reward	**481**
3	1	-444
	2	-362
	3	-494
	4	-301
	5	-384
	6	-236
	7	-108
	8	-500
	9	-81
	10	-124
	Average Reward	**-303**

It is worth mentioning that we initially aimed to use different RL algorithms with our intrusion response problem to perform an analytical comparison of their defensive performance. However, with the long training time challenge that we faced during the different experiments of the DDQN agent, using other RL algorithms was not achievable since we did not have the time for it. Also, it was not feasible to compare our solution with other state-of-the-art works because each work uses a different simulation environment. They could have the same objective, but with totally different state space, action space, and reward setup. Accordingly, comparing their computational time, performance, or their cumulative reward curve would have not been reasonable nor fair for a valuable comparison. Consequently, we mainly depended in our evaluation on different testing experiments conducted on the CPS-designed testbed to assess the effectiveness of each proposed solution in choosing the best defensive actions.

All in all, we strongly believe that the IRS field lacks a reference system for validation and comparison of the different intrusion response approaches. While intrusion detection systems can be validated and compared against some publicly available labeled datasets, unfortunately, intrusion response systems cannot. The evaluation of an intrusion response system is more complex because it needs to be performed on some testbed. Accordingly, given the lack of a reference testbed, it is almost impossible to rigorously compare different intrusion response approaches.

Finally, going back to our second research question, which is ‘Are model-free deep reinforcement learning intrusion response systems effective for cyber-physical systems?’, the answer is Yes. From our different experiments, we can evidently state that the agent protected the process from falling into hazards 100% of the time. Without the proposed automated defensive mechanisms approach, the alternative would have been human operators taking different actions. However, human actions in times of emergency are not guaranteed. Accordingly, in comparison to the human alternative solution, we believe that our automated model-free deep reinforcement learning intrusion response system solution is effective for cyber-physical systems.

8.4 Dataset Collection

There is a lack of available CPS security-related datasets, in the form of <state, action, next state, reward> tuples, that can be used with offline reinforcement learning approaches. Having a publicly available dataset will be a benefit for the research community so that they can validate and compare their algorithms and solutions against a reference dataset. Accordingly, we took the chance and collected a representative dataset of our CSTR CPS environment during online training using the DDQN algorithm.

The data were collected for 9 days with a total of 3 attack instances of the false data injection. It contains 406,047 samples, in which each sample in the dataset includes the state, action, next state, and the reward <state, action, next state, reward>. The state is defined as $S = [T\ L\ OF\ AS]$, where T and L are the reactor temperature and the reactor level, respectively. OF is the outlet flow and AS is IDS-related evidence showing the type of the detected attack scenario. The action space contains 6 discrete action vectors, which are precisely chosen by experts, to defend against the different attacks. The action vector is defined as $A = [DP\ SI\ SC\ SO\ SP\ MS]$, where DP is a cyber action to whether drop attack packets or not. $SI, SC, and\ SO$ are the actions that determine whether to open or close the inlet, coolant, and outlet safety valves, respectively. SP is an action to reconfigure the setpoint and MS decides whether the manual shutdown action is needed or not.

We collected this dataset to analyze and study its applicability in our future work for designing an offline reinforcement learning agent for intrusion response decision-making problems. We also aim to perform an analytical study to compare the performance of our pursued online DRL-IRS solution and the future offline RL solution.

8.5 Challenges

We faced many challenges during the preparation of this thesis. At first, the literature work on intrusion response systems for cyber-physical processes is still in its early stages. Accordingly, we spent a lot of time in the literature review phase to decide which approach is suitable and applicable to our CPS. Then, when deciding on using RL-based approaches, it was hard to fully understand the mathematical concepts of the bellman equation and the

different methodologies of the possible algorithms. Particularly, it was challenging because many things had to be understood and comprehended in a short time.

Setting up the testbed as the interactive online environment in the reinforcement learning framework was another difficult task. At first, we had to build the firewall Iptables; to separate the control network from the cooperate network. Then, we changed the communication protocol between the components from UDP/IP to Modbus/TCP. Following that, we established a new communication link between our RL agent, which runs on MATLAB, and the testbed components simulated on LabVIEW. Initially, we used the TCP/IP protocol. However, several errors were observed, such as a timeout error between the two communicated nodes, due to the reliability strictness of the TCP/IP protocol. For that reason, we reestablished the communication link between the RL agent in MATLAB and LabVIEW using UDP/IP protocol.

After setting up the testbed, we started designing and implementing different attack scenarios. We took some time to be able to perform the false data injection attack on our testbed. This step required knowledge of using the Wireshark protocol analyzer, understanding the different fields of the used Modbus/TCP communication protocol, and learning how to use different attacking tools, such as the Ettercap tool used for ARP spoofing. One of the outcomes of this phase was designing a complete attack script using python for the different attack scenarios modelled in Figure 20. This attack script, which utilizes scapy and nfqueue packages running on Ubuntu OS, contains all the needed attack steps to ease and automate the process of performing attacks to study the system vulnerabilities.

The next stage focused on building the main blocks of the reinforcement learning framework to start the training phase. We spent some effort looking for and experimenting with different mitigation actions that can be implemented automatically without human intervention. However, one of the hardest things that we faced is designing a representative reward function that shows our goals in a mathematical formula. Since there is no standard approach to designing the reward function, we had several iterations of modifying and experimenting with different types of reward functions before settling on the final one.

The training phase was the biggest challenge of the entire thesis. It was extremely time-consuming, and we had a lot of failed attempts, such as the attempt shown in Figure 36, that we did not have clear justifications for their failure. Not having a dataset to train our system offline contributed hugely to having a very long training time since online training is significantly slower. Moreover, having a huge number of hyperparameters to tune in the DRL framework was challenging and contributed to the high time complexity of the approach. Consequently, scalability is still an issue that we faced even for our small-scale system.

Evaluating our proposed solutions was also one of the thesis challenges. Due to the long training time issue that we discussed, we did not have the time to use different algorithms on our testbed and compare them. In addition, it was not reasonable to compare our solutions with the state-of-the-art works since each work uses a different simulation environment and a completely different setup. Consequently, we depended on our local experiments to evaluate the effectiveness of each trained agent.

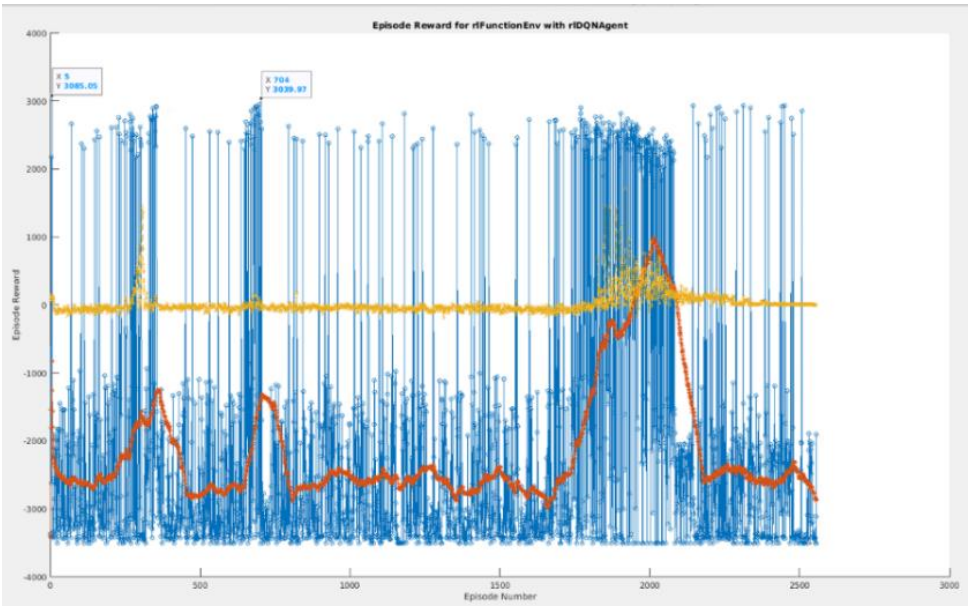

Figure 36. Agent training - failed attempt example (results after 5 training days)

CHAPTER 9: CONCLUSION AND FUTUREWORKS

CPSs are used in many systems, including industrial processes, transportation systems, and energy systems. The increased connectivity of CPSs made them more vulnerable to cyberattacks. Existing research effort on security defensive mechanisms has been focusing on intrusion detection to detect and classify cyberattacks. The work on intrusion response is still at its early stages and lacks applicability to real CPSs.

In this context, this thesis first provides a comprehensive review of the design taxonomy and decision-making solutions of intrusion response systems. Different works are surveyed, compared, and analyzed to highlight their main advantages, disadvantages, and future directions. Then the thesis presents a model-based design approach for a CPS security testbed. The testbed hardware and software architectural models were developed, and a systematic methodology to generate cyberattack experiments was explained. Additionally, we used two approaches to design an intrusion response system solution for defending the CPS testbed. The first approach uses an optimization technique with a Genetic Algorithm, and the second approach uses model-free deep reinforcement learning with a DDQN algorithm. Both approaches proved their effectiveness in selecting optimal actions for automatically responding to false data injection attacks. Furthermore, we collected a security-related dataset for designing and evaluating future offline reinforcement learning solution approaches.

We anticipate that this thesis will help interested readers to obtain a full view of IRS for CPS. Also, it presents to researchers and practitioners an effective IRS design for CPS as well as a CPS testbed and a dataset to ease further research and development. For future work, we will consider using a multi-agent reinforcement learning approach, utilizing the collected dataset for offline RL training approaches, dealing with variable action space, adding more attack scenarios, and comparing with other RL algorithms.

REFERENCES

[1] A. Humayed, J. Lin, F. Li, and B. Luo, “Cyber-Physical Systems Security — A Survey,” vol. 4, no. 6, pp. 1802–1831, 2017.

[2] Z. Drias, A. Serhrouchni, and O. Vogel, “Taxonomy of attacks on Industrial Control protocols,” *2015 Int. Conf. Protoc. Eng. Int. Conf. New Technol. Distrib. Syst.*, pp. 1–6, 2015.

[3] L. Cao, X. Jiang, Y. Zhao, S. Wang, D. You, and X. Xu, “A Survey of Network Attacks on Cyber-Physical Systems,” *IEEE Access*, vol. 8, pp. 44219–44227, 2020.

[4] A. Singh and A. Jain, “Study of Cyber Attacks on Cyber-Physical System,” *SSRN Electron. J.*, no. October 2019, 2018.

[5] F. Li, X. Yan, Y. Xie, Z. Sang, and X. Yuan, “A Review of Cyber-Attack Methods in Cyber-Physical Power System,” *APAP 2019 - 8th IEEE Int. Conf. Adv. Power Syst. Autom. Prot.*, pp. 1335–1339, 2019.

[6] B. Chen, N. Pattanaik, A. Goulart, K. L. Butler-Purry, and D. Kundur, “Implementing attacks for modbus/TCP protocol in a real-time cyber physical system test bed,” *Proc. - CQR 2015 2015 IEEE Int. Work. Tech. Comm. Commun. Qual. Reliab.*, pp. 1–6, 2015.

[7] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, “Guide to Industrial Control Systems (ICS) Security NIST Special Publication 800-82 Revision 2,” *NIST Spec. Publ. 800-82 rev 2*, pp. 1–157, 2015.

[8] H. Xu, W. E. I. Yu, D. Griffith, and N. Golmie, “A Survey on Industrial Internet of Things : A Cyber-Physical Systems Perspective,” *IEEE Access*, vol. 6, pp. 78238–78259, 2018.

[9] M. H. Cintuglu, O. A. Mohammed, K. Akkaya, S. Member, A. S. Uluagac, and S. Member, “A Survey on Smart Grid Cyber-Physical System Testbeds,” vol. 19, no. 1, pp. 446–464, 2017.

[10] G. Yadav and K. Paul, “Architecture and security of SCADA systems : A review,” *Int. J. Crit. Infrastruct. Prot.*, vol. 34, p. 100433, 2021.

[11] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, “Anomaly detection in cyber physical systems using recurrent neural networks,” *Proc. IEEE Int. Symp. High Assur. Syst. Eng.*, pp. 140–145, 2017.

[12] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun, “Anomaly detection for a water treatment system using unsupervised machine learning,” *IEEE Int. Conf. Data Min. Work. ICDMW*, vol. 2017-Novem, pp. 1058–1065, 2017.

[13] M. Kravchik and A. Shabtai, “Detecting cyber attacks in industrial control systems using convolutional neural networks,” *Proc. ACM Conf. Comput. Commun. Secur.*, no. October, pp. 72–83, 2018.

[14] Q. Lin, “TABOR : A Graphical Model-based Approach for Anomaly Detection in Industrial TABOR : A Graphical Model-based Approach for Anomaly Detection in Industrial Control Systems,” no. June, 2018.

[15] C. R. Nov, “Anomaly Detection for Industrial Control Systems Using Sequence-to-Sequence Neural Networks.”

[16] Y. Li, S. Member, L. Zhang, Z. Lv, and W. Wang, “Detecting Anomalies in Intelligent Vehicle Charging and Station Power Supply Systems With Multi-Head Attention Models,” vol. 22, no. 1, pp. 555–564, 2021.

[17] D. Bhamare, M. Zolanvari, A. Erbad, R. Jain, and K. Khan, “Cybersecurity for industrial control systems : A survey,” *Comput. Secur.*, vol. 89, 2020.

[18] H. Habibzadeh, B. H. Nussbaum, F. Anjomshoa, B. Kantarci, and T. Soyata, “A survey on cybersecurity , data privacy , and policy issues in cyber-physical system deployments in smart cities,” *Sustain. Cities Soc.*, vol. 50, p. 101660, 2019.

[19] A. A. Nazarenko and G. A. Safdar, “Survey on security and privacy issues in cyber physical systems,” *AIMS Electron. Electr. Eng.*, no. July, 2020.

[20] M. Rizwan, Q. Hu, and S. Zeadally, “Cybersecurity in industrial control systems : Issues , technologies , and challenges,” *Comput. Networks*, vol. 165, p. 106946, 2019.

[21] C. Alcaraz and S. Zeadally, “Critical infrastructure protection: Requirements and challenges for the 21st century,” *Int. J. Crit. Infrastruct. Prot.*, vol. 8, pp. 53–66, Jan.

2015.

[22] M. N. Al-mhiqani *et al.*, “Cyber-Security Incidents : A Review Cases in Cyber-Physical Systems,” vol. 9, no. 1, 2018.

[23] J. A. Yaacoub, O. Salman, H. N. Noura, N. Kaaniche, and A. Chehab, “Cyber-physical systems security : Limitations , issues and future trends,” *Microprocess. Microsyst.*, vol. 77, 2020.

[24] M. Iaiani, A. Tugnoli, S. Bonvicini, and V. Cozzani, “Analysis of Cybersecurity-related Incidents in the Process Industry,” *Reliab. Eng. Syst. Saf.*, vol. 209, p. 107485, 2021.

[25] S. Zhioua, “The middle east under malware attack dissecting cyber weapons,” *Proc. - Int. Conf. Distrib. Comput. Syst.*, pp. 11–16, 2013.

[26] M. M. Ahmadian, M. Shajari, and M. A. Shafiee, “Industrial control system security taxonomic framework with application to a comprehensive incidents survey,” *Int. J. Crit. Infrastruct. Prot.*, vol. 29, p. 100356, 2020.

[27] S. Han, M. Xie, H. H. Chen, and Y. Ling, “Intrusion detection in cyber-physical systems: Techniques and challenges,” *IEEE Syst. J.*, vol. 8, no. 4, pp. 1049–1059, 2014.

[28] T. Daniya, K. S. Kumar, B. S. Kumar, and C. Sekhar, “A survey on anomaly based intrusion detection system,” *Mater. Today Proc.*, no. xxxx, 2021.

[29] Kamaldeep, M. Dutta, and J. Granjal, “Towards a Secure Internet of Things: A Comprehensive Study of Second Line Defense Mechanisms,” *IEEE Access*, vol. 8, pp. 127272–127312, 2020.

[30] B. Y. W. Mazurczyk, L. Caviglione, and A. E. Day, “review articles Cyber Reconnaissance Techniques,” 2021.

[31] F. Salahdine, N. Kaabouch, and R. Gloria, “Social Engineering Attacks: A Reconnaissance Synthesis Analysis,” *IEEE Annu. Ubiquitous Comput. Electron. Mob. Commun. Conf.*, no. October, 2020.

[32] R. Chabukswar, Y. Mo, and B. Sinopoli, *Detecting Integrity Attacks on SCADA*

Systems, vol. 44, no. 1. IFAC, 2011.

[33] R. Lanotte, U. Insubria, and D. Informatica, "A Formal Approach to Cyber-Physical Attacks," *IEEE 30th Comput. Secur. Found. Symp.*, no. November 2018, 2017.

[34] Z. Inayat, A. Gani, N. B. Anuar, M. K. Khan, and S. Anwar, "Intrusion response systems: Foundations, design, and challenges," *J. Netw. Comput. Appl.*, vol. 62, pp. 53–74, 2016.

[35] A. Shameli-Sendi, N. Ezzati-Jivan, M. Jabbarifar, and M. Dagenais, "Intrusion response systems: survey and taxonomy," *Int. J. Comput. Sci. Netw. Secur*, vol. 12, no. 1, pp. 1–14, 2012.

[36] S. Anwar *et al.*, "From intrusion detection to an intrusion response system: Fundamentals, requirements, and future directions," *Algorithms*, vol. 10, no. 2, 2017.

[37] J. Wong, "A taxonomy of intrusion response systems," *Int. J. Inf. Comput. Secur.*, no. January 2007, 2014.

[38] A. Shameli-sendi, M. Cheriet, and A. Hamou-lhadj, "Taxonomy of intrusion risk assessment and response system," *Coputers Secur.*, vol. 5, pp. 1–16, 2014.

[39] E. Hodo *et al.*, "Threat analysis of IoT networks Using Artificial Neural Network Intrusion Detection System," pp. 4–8, 2020.

[40] H. A. Kholidy, "Autonomous mitigation of cyber risks in the Cyber – Physical Systems," *Futur. Gener. Comput. Syst.*, vol. 115, pp. 171–187, 2021.

[41] D. K. Singh and P. Kaushik, "Analysis of Decision Making factors for Automated Intrusion Response System (AIRS): A Review," vol. 14, no. 6, p. 5500, 2016.

[42] V. Mateos, V. A. Villagrá, F. Romero, and J. Berrocal, "Definition of response metrics for an ontology-based Automated Intrusion Response Systems q," vol. 38, pp. 1102–1114, 2012.

[43] C. Mu, B. Shuai, and H. Liu, "Analysis of Response Factors in Intrusion Response Decision-Making," *2010 Third Int. Jt. Conf. Comput. Sci. Optim.*, vol. 2, pp. 395–399, 2010.

[44] Snort Project Team, "SNORT Users Manual 2.9.16," 2020.

[45] Kamesh and N. Sakthi Priya, “A survey of cyber crimes Yanping,” *Secur. Commun. Networks*, vol. 5, no. June, pp. 422–437, 2012.

[46] A. Justina and A. Simon, “A Credible Cost-Sensitive Model For Intrusion Response Selection,” *2012 Fourth Int. Conf. Comput. Asp. Soc. Networks*, pp. 222–227, 2012.

[47] A. J. Ikuomola and J. O. Nehinbe, “A Framework for Collaborative , Adaptive and Cost Sensitive Intrusion Response System,” *2010 2nd Comput. Sci. Electron. Eng. Conf.*, pp. 1–4, 2010.

[48] J. S. Wong and C. Strasburg, “A Framework for Cost Sensitive Assessment of Intrusion Response Selection,” *2009 33rd Annu. IEEE Int. Comput. Softw. Appl. Conf.*, vol. 1, pp. 355–360, 2009.

[49] A. Shameli-sendi and M. Dagenais, “ORCEF : Online response cost evaluation framework for intrusion response system,” *J. Netw. Comput. Appl.*, vol. 55, pp. 89–107, 2015.

[50] M. Jahnke, C. Thul, and P. Martini, “Graph based Metrics for Intrusion Response Measures in Computer Networks 1,” 2007.

[51] N. Kheir and J. Viinikka, “Cost Evaluation for Intrusion Response Using Dependency Graphs,” no. c, pp. 1–6, 2009.

[52] C. Strasburg and J. S. Wong, “Intrusion response cost assessment methodology,” no. January, 2009.

[53] A. Tantawy, S. Abdelwahed, A. Erradi, and K. Shaban, “Model-Based Risk Assessment for Cyber Physical Systems Security Computers & Security Model-based risk assessment for cyber physical systems security,” *Comput. Secur.*, vol. 96, no. May, p. 101864, 2020.

[54] Y. Cherdantseva *et al.*, “A review of cyber security risk assessment methods for SCADA systems,” vol. 56, pp. 1–27, 2016.

[55] J. Greensmith, “Securing the Internet of Things with Responsive Artificial Immune Systems,” pp. 113–120, 2015.

[56] S. Anwar, J. M. Zain, M. F. Zolkipli, Z. Inayat, A. N. Jabir, and J. B. Odili, “Response

option for attacks detected by intrusion detection system," *2015 4th Int. Conf. Softw. Eng. Comput. Syst. ICSECS 2015 Virtuous Softw. Solut. Big Data*, pp. 195–200, 2015.

[57] N. B. Anuar, M. Papadaki, S. Furnell, and N. Clarke, "An investigation and survey of response options for Intrusion Response Systems (IRSs)," *2010 Inf. Secur. South Africa*, pp. 1–8, 2010.

[58] A. Nadeem and M. Howarth, "Adaptive intrusion detection & prevention of denial of service attacks in Adaptive Intrusion Detection & Prevention of Denial of Service attacks in MANETs," no. January, 2009.

[59] W. Kanoun, N. Cuppens-boulahia, S. Dubus, B. Laboratories, T. Bretagne, and I. Telecom, "Risk-aware Framework for Activating and Deactivating Policy-based Response," pp. 207–215, 2010.

[60] S. K. N. and P. Kabiri, "An Adaptive and Cost-Based Intrusion Response System," *Cybern. Syst.*, vol. 48, pp. 495–509, 2017.

[61] I. Balepin, S. Maltsev, J. Rowe, and K. Levitt, "Using specification-based intrusion detection for automated response," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2820, pp. 136–154, 2003.

[62] A. Nadeem and M. Howarth, "An Intrusion Detection & Adaptive Response Mechanism for MANETs An Intrusion Detection & Adaptive Response Mechanism for MANETs," no. January 2019, 2014.

[63] S. Berenjian, M. Shajari, N. Farshid, and M. Hatamian, "Intelligent Automated Intrusion Response System based on fuzzy decision making and risk assessment," *2016 IEEE 8th Int. Conf. Intell. Syst. IS 2016 - Proc.*, pp. 709–714, 2016.

[64] A. Shameli-Sendi, J. Desfossez, M. Dagenais, and M. Jabbarifar, "A retroactive-burst framework for automated intrusion response system," *J. Comput. Networks Commun.*, vol. 2013, 2013.

[65] B. Boˇ, "Game-Theoretic Algorithms for Optimal Network Security Hardening Using Attack Graphs Optimal Network Security Hardening Using Attack Graph Games," no. July, 2015.

[66] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley, “Optimal security hardening using multi-objective optimization on attack tree Optimal Security Hardening Using Multi-objective Optimization on Attack Tree Models of Networks,” no. January, 2007.

[67] L. Feng, W. Wang, L. Zhu, and Y. Zhang, “Predicting intrusion goal using dynamic Bayesian network with transfer probability estimation $,” *J. Netw. Comput. Appl.*, vol. 32, pp. 2008–2010, 2009.

[68] H. S. Lallie, K. Debattista, and J. Bal, “A review of attack graph and attack tree visual syntax in cyber security,” *Comput. Sci. Rev.*, vol. 35, p. 100219, 2020.

[69] P. Nespoli, D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, “Optimal Countermeasures Selection Against Cyber Attacks: A Comprehensive Survey on Reaction Frameworks,” *IEEE Commun. Surv. Tutorials*, vol. 20, no. 2, pp. 1361–1396, 2018.

[70] V. Shandilya, C. B. Simmons, and S. Shiva, “Use of attack graphs in security systems,” *J. Comput. Networks Commun.*, vol. 2014, no. April 2016, 2014.

[71] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer, “DAG-based attack and defense modeling: Don’t miss the forest for the attack trees,” *Comput. Sci. Rev.*, vol. 13–14, no. C, pp. 1–38, 2014.

[72] Y. Wang, Y. Wang, J. Liu, Z. Huang, and P. Xie, “A survey of game theoretic methods for cyber security,” *Proc. - 2016 IEEE 1st Int. Conf. Data Sci. Cyberspace, DSC 2016*, pp. 631–636, 2017.

[73] I. Musah, D. K. Boah, and B. Seidu, “A Comprehensive Review of Solution Methods and Techniques for Solving Games in Game Theory,” *J. Game Theory*, vol. 9, no. 2, pp. 25–31, 2020.

[74] H. Liu, Y. Li, Z. Duan, and C. Chen, “A review on multi-objective optimization framework in wind energy forecasting techniques and applications,” *Energy Convers. Manag.*, vol. 224, no. April, 2020.

[75] A. Konak, D. W. Coit, and A. E. Smith, “Multi-objective optimization using genetic algorithms : A tutorial,” vol. 91, pp. 992–1007, 2006.

[76] K. D. and S. A. and A. P. and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, vol. 6, pp. 182–197, 2002.

[77] K. Taha, “Methods that optimize multi-objective problems: A survey and experimental evaluation,” *IEEE Access*, vol. 8, no. 1, pp. 80855–80878, 2020.

[78] N. Gunantara, “A review of multi-objective optimization: Methods and its applications,” *Cogent Eng.*, vol. 5, no. 1, pp. 1–16, 2018.

[79] R. T. Marler and J. S. Arora, “Survey of multi-objective optimization methods for engineering,” *Struct. Multidiscip. Optim.*, vol. 26, no. 6, pp. 369–395, 2004.

[80] G. Chiandussi, M. Codegone, S. Ferrero, and F. E. Varesio, *Comparison of multi-objective optimization methodologies for engineering applications*, vol. 63, no. 5. Elsevier Ltd, 2012.

[81] Y. Rizk, M. Awad, and E. W. Tunstel, “Decision Making in Multiagent Systems : A Survey,” vol. 10, no. 3, pp. 514–529, 2020.

[82] M. A. Alsheikh *et al.*, “Markov Decision Processes With Applications in Wireless Sensor Networks : A Survey,” vol. 17, no. 3, pp. 1239–1267, 2015.

[83] C. Kiennert, Z. Ismail, H. Debar, and J. Leneutre, “A survey on game-theoretic approaches for intrusion detection and response optimization,” *ACM Comput. Surv.*, vol. 51, no. 5, 2018.

[84] A. P. Patil, S. Bharath, and N. M. Annigeri, “Applications of Game Theory for Cyber Security System : A Survey,” vol. 13, no. 17, pp. 12987–12990, 2018.

[85] C. T. Do *et al.*, “Game theory for cyber security and privacy,” *ACM Comput. Surv.*, vol. 50, no. 2, pp. 30–37, 2017.

[86] X. Liang and Y. Xiao, “Game theory for network security,” *IEEE Commun. Surv. Tutorials*, vol. 15, no. 1, pp. 472–486, 2013.

[87] Z. Zhou and H. Xu, “Deep Reinforcement Learning Based Intelligent Decision Making for Two-player Sequential Game with Uncertain Irrational Player,” *2019 IEEE Symp. Ser. Comput. Intell. SSCI 2019*, pp. 9–15, 2019.

[88] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*,

2015.

[89] Y. Long and H. He, “Robot path planning based on deep reinforcement learning,” *2020 IEEE Conf. Telecommun. Opt. Comput. Sci. TOCS 2020*, pp. 151–154, 2020.

[90] M. Saeed, M. Nagdi, B. Rosman, and H. H. S. M. Ali, “Deep Reinforcement Learning for Robotic Hand Manipulation,” *Proc. 2020 Int. Conf. Comput. Control. Electr. Electron. Eng. ICCCEEE 2020*, 2021.

[91] H. GÜLMEZ, “a Deep Reinforcement Learning Approach for Anomaly Network Intrusion Detection System,” no. September, pp. 5–10, 2019.

[92] M. Lopez-martin, B. Carro, and A. Sanchez-esguevillas, “Application of deep reinforcement learning to intrusion detection for supervised problems,” *Expert Syst. Appl.*, vol. 141, p. 112963, 2020.

[93] J. Liao, T. Liu, X. Tang, X. Mu, B. Huang, and D. Cao, “Decision-making strategy on highway for autonomous vehicles using deep reinforcement learning,” *IEEE Access*, vol. 8, pp. 177804–177814, 2020.

[94] Z. Zhou and H. Xu, “Switching Deep Reinforcement Learning based Intelligent Online Decision Making for Autonomous Systems under Uncertain Environment,” *Proc. 2018 IEEE Symp. Ser. Comput. Intell. SSCI 2018*, pp. 1453–1460, 2019.

[95] T. Yang, L. Zhao, W. Li, and A. Y. Zomaya, “Reinforcement learning in sustainable energy and electric systems: a survey,” *Annu. Rev. Control*, vol. 49, pp. 145–163, 2020.

[96] M. Alabadi and Z. Albayrak, “Q-Learning for Securing Cyber-Physical Systems : A survey,” no. June, 2020.

[97] A. Uprety, D. B. Rawat, and S. Member, “Reinforcement Learning for IoT Security : A Comprehensive Survey,” vol. 4662, no. c, pp. 1–14, 2020.

[98] X. Liu, H. Xu, W. Liao, and W. Yu, “Reinforcement learning for cyber-physical systems,” *Proc. - IEEE Int. Conf. Ind. Internet Cloud, ICII 2019*, no. Icii, pp. 318–327, 2019.

[99] A. Shameli-sendi, H. Louafi, and W. He, “Dynamic Optimal Countermeasure

Selection for Intrusion Response System," vol. XX, pp. 1–14, 2016.

[100] X. Li, C. Zhou, Y. Tian, and Y. Qin, "A Dynamic Decision-Making Approach for Intrusion Response in Industrial Control Systems," *IEEE Trans. Ind. Informatics*, vol. 15, no. 5, pp. 2544–2554, 2019.

[101] S. Huang, C. Zhou, N. Xiong, S. Member, S. Yang, and S. Member, "A General Real-Time Control Approach of Intrusion Response for Industrial Automation Systems," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 46, no. 8, pp. 1021–1035, 2016.

[102] Y. Guo *et al.*, "Decision-Making for Intrusion Response : Which , Where , in What Order , and How Long ?," 2020.

[103] S. Hossain, S. Etigowni, K. Davis, and S. Zonouz, "Towards Cyber-Physical Intrusion Tolerance," *2015 IEEE Int. Conf. Smart Grid Commun.*, pp. 139–144, 2015.

[104] S. Huang, C. Zhou, S. Yang, and Y. Qin, "Cyber-physical System Security for Networked Industrial Processes," vol. 12, no. December, pp. 567–578, 2015.

[105] Y. Qin, Q. Zhang, C. Zhou, and N. Xiong, "A Risk-Based Dynamic Decision-Making Approach for Cybersecurity Protection in Industrial Control Systems," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. PP, pp. 1–8, 2018.

[106] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley, "RRE : A Game-Theoretic Intrusion Response and Recovery Engine," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 395–406, 2014.

[107] S. Iannucci, O. D. Barba, V. Cardellini, and I. Banicescu, "A performance evaluation of deep reinforcement learning for model-based intrusion response," *Proc. - 2019 IEEE 4th Int. Work. Found. Appl. Self* Syst. FAS*W 2019*, pp. 158–163, 2019.

[108] S. Iannucci and S. Abdelwahed, "Model-Based Response Planning Strategies for Autonomic Model-based Response Planning Strategies for Autonomic Intrusion Protection," no. April, 2018.

[109] 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 Trans. Ind. Electron.*, vol. 67, no. 3, pp. 2371–2379, 2020.

[110] J. Khoury and M. Nassar, “A Hybrid Game Theory and Reinforcement Learning Approach for Cyber-Physical Systems Security,” *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. 2020 Manag. Age Softwarization Artif. Intell. NOMS 2020*, 2020.

[111] S. Paul, Z. Ni, and C. Mu, “A Learning-Based Solution for an Adversarial Repeated Game in Cyber-Physical Power Systems,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 11, pp. 4512–4523, 2020.

[112] S. Iannucci, V. Cardellini, O. Daniel, and I. Banicescu, “A hybrid model-free approach for the near-optimal intrusion response control of non-stationary systems,” *Futur. Gener. Comput. Syst.*, vol. 109, pp. 111–124, 2020.

[113] Z. S. Stefanova and K. M. Ramachandran, “Off-Policy Q-learning Technique for Intrusion Response in Network Security,” vol. 12, no. 4, pp. 266–272, 2018.

[114] S. Iannucci, A. Montemaggio, and B. Williams, “Towards Self-Defense of Non-Stationary Systems,” *2019 Int. Conf. Comput. Netw. Commun.*, pp. 250–254, 2021.

[115] Y. Liu, M. Dong, K. Ota, J. Li, and J. Wu, “Deep Reinforcement Learning based Smart Mitigation of DDoS Flooding in Software-Defined Networks,” *IEEE Int. Work. Comput. Aided Model. Des. Commun. Links Networks, CAMAD*, vol. 2018-Septe, pp. 1–6, 2018.

[116] K. Malialis and D. Kudenko, “Distributed response to network intrusions using multiagent reinforcement learning,” *Eng. Appl. Artif. Intell.*, vol. 41, pp. 270–284, 2015.

[117] V. Cardellini *et al.*, “An Intrusion Response System Utilizing Deep Q-Networks and System Partitions,” *SSRN Electron. J.*, 2022.

[118] A. T. Arash Golabi, Abdelkarim Erradi, “Towards Automated Hazard Analysis for CPS Security with Application to CSTR System,” *J. Process Control*, 2020.

[119] A. Hahn, A. Ashok, S. Sridhar, and M. Govindarasu, “Cyber-physical security testbeds: Architecture, application, and evaluation for smart grid,” *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 847–855, 2013.

[120] U.S. Department of Energy, “National SCADA Test Bed Enhancing control systems

security in the energy sector PROTECTING," 2009.

[121] T. Inl and Idaho National Laboratory, "National SCADA Test Bed Substation Automation Evaluation Report," 2009.

[122] M. Bashendy, S. Eltanbouly, A. Tantawy, and A. Erradi, "Design and Implementation of Cyber-Physical Attacks on Modbus / TCP Protocol," *World Congr. Ind. Control Syst. Secur.*, 2020.

[123] C. Deloglos, C. Elks, and A. Tantawy, "An Attacker Modeling Framework for the Assessment of Cyber-Physical Systems Security," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12234 LNCS, no. June, pp. 150–163, 2020.

[124] J. Dunjó, V. Fthenakis, J. A. Vílchez, and J. Arnaldos, "Hazard and operability (HAZOP) analysis. A literature review," *J. Hazard. Mater.*, vol. 173, no. 1–3, pp. 19–32, 2010.

[125] V. Punnathanam, C. Sivadurgaprasad, and P. Kotecha, "On the performance of MATLAB's inbuilt genetic algorithm on single and multi-objective unconstrained optimization problems," *Int. Conf. Electr. Electron. Optim. Tech. ICEEOT 2016*, pp. 3976–3981, 2016.

[126] Z. Jinghui, H. Xiaomin, G. Min, and Z. Jun, "Comparison of performance between different selection strategies on simple genetic algorithms," *Proc. - Int. Conf. Comput. Intell. Model. Control Autom. CIMCA 2005 Int. Conf. Intell. Agents, Web Technol. Internet*, vol. 2, no. January 2015, pp. 1115–1120, 2005.

[127] "Define Reward Signals," *MathWorks*. [Online]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/define-reward-signals.html#:~:text=To guide the learning process,of taking a particular action. [Accessed: 01-Mar-2022].

[128] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-Learning Algorithms: A Comprehensive Classification and Applications," *IEEE Access*, vol. 7, pp. 133653–133667, 2019.

[129] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-Learning,” *30th AAAI Conf. Artif. Intell. AAAI 2016*, no. September 2015, pp. 2094–2100, 2016.

[130] A. Mishra, “Tackling Exploration-Exploitation Dilemma in K-armed Bandits,” *Nerd For Tech*, 2021. [Online]. Available: https://medium.com/nerd-for-tech/tackling-exploration-exploitation-dilemma-in-k-armed-bandits-598c0329cf88.