5-12-2024

RECOVERY OF CYBER MANUFACTURING SYSTEMS FROM FALSE DATA INJECTION ATTACKS ON SENSORS BY REINFORCEMENT LEARNING

Romesh Prasad
Syracuse University

ABSTRACT

Cyber-manufacturing systems (CMS) represent the future design vision for manufacturing systems, aiming to enhance product quality, productivity, and reduce downtime. The third industrial revolution was driven by computer systems and automation, while the fourth industrial revolution was inspired by the internet, network, and connectivity. However, this integration also introduces vulnerabilities, particularly through false data injection attacks on sensors.

Sensors play a pivotal role in the automation framework of manufacturing systems spanning five levels: process, operational, supervisory, plant, and enterprise. Sensors are instrumental in communicating information from the process level to the enterprise level. It also enables controllers at the process and operational/supervisory level to implement precise control algorithms that enable the functioning of cyber manufacturing systems in a predetermined sequence to yield the desired end product. Moreover, they provide real-time state estimation of the manufacturing systems to the plant controller. With this information, the plant control algorithms work to minimize any discrepancies between the manufacturing system's output and the input required for optimal manufacturing operation. However, the integrity of the information from the sensors can be compromised

by the actions of cyber attackers who may exploit vulnerabilities in the manufacturing system's networks or control software. Hence jeopardizing the manufacturing system's availability and potentially leading it into an unsafe operational state.

The rising cyber threats on cyber manufacturing systems have motivated researchers to propose solutions aimed at enhancing resiliency. At a high level, these solutions draw inspiration from the cyber security domain and can be categorized into prevention and detection strategies. Prevention from cyber-attacks is achieved by enforcing constraints in the form of rules, and policies, or by incorporating modern methods of firewall, physical hash, and blockchain. While prevention strategies are crucial, they do not guarantee complete freedom from cyber threats and hence a detection strategy is advised. The objective of the detection strategy is to identify and flag anomalous behavior within the manufacturing systems. However, a notable gap within this literature is guidance *on how to respond effectively once a sensor attack is detected.* This critical aspect, *post-detection recovery*, remains underexplored and warrants further attention.

This research addresses this gap by proposing a reinforcement learning recovery agent and introducing a four-layer recovery architecture. The architecture encompasses the systems layer, attack identification layer, data auditing and detection layer, and recovery layer. The systems layer identifies and categorizes the manufacturing system components within the five-layer automation hierarchy. The attack identification layer performs risk analysis to identify vulnerabilities, while the data auditing and detection layer collects and trains data for attack detection. The recovery layer focuses on training reinforcement learning agents to respond effectively to detected attacks.

To validate this architecture, a testbed and manufacturing simulator are developed, featuring two robotic arms, a conveyor belt, and a drawing manufacturing process. Two distinct sensor attack scenarios are presented, and the proposed recovery agent's performance is compared against a PID controller using critical manufacturing metrics: downtime, throughput, and efficiency. The research aims to enhance the resilience and security of manufacturing systems by effectively responding to sensor attacks.

In conclusion, this research contributes to the evolving field of CMS security by proposing a novel recovery architecture and reinforcement learning-based recovery agent, filling a critical gap in post-detection response strategies. The validation through a manufacturing simulator demonstrates the potential of the proposed approach in minimizing the impact of sensor attacks and ensuring the continuous operation of CMS.

RECOVERY OF CYBER MANUFACTURING SYSTEMS FROM FALSE DATA INJECTION ATTACKS ON SENSORS BY REINFORCEMENT LEARNING

by

Romesh Satish Sarita Prasad

B.S., University of Mumbai, 2016

M.S., Syracuse University, 2019

Dissertation

Submitted in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy in

Mechanical and Aerospace Engineering

Syracuse University

May 2024

ACKNOWLEDGEMENTS

I extend my heartfelt gratitude to my family, the bedrock of my existence. My parents, Sarita and Satish, instilled in me values that shaped my character—humility, respect, and the importance of honest endeavors. My sisters, Roma and Roshni, taught me the significance of treating everyone with respect and enlightened me on the challenges women face in our society. Their influence has been pivotal in today's accomplishments.

I owe a debt of gratitude to my incredible friends, a source of constant support, laughter, and love. In no order Sanket Kulkarni, Gaurish Mestha, Snehav Sharma, Apoorv Bhatia, Shubham Jiwtode, Sai Kiran Bulusu, Varun Mulay, Somesh Singh, Nitish Kumar, Prashant Kamat, Kshitij Joshi, Amay Kalamkar, Nikita Dhulap, Priyanka Kalamkar, Ashish Shinde, Aishwariya Rajendraprasad, Shweta Shukla, Saloni Jain, Sanskriti Jain, Tanvi Tanwar, Swathi G S, Ashish Tribhuvan, Prathamesh Palav, Vijay Pandey, Soham Deshmukh, Abhijit Padyar, and Bhanukumar Trivedi—your unwavering support made my Ph.D. journey not only survivable but also memorable.

A special acknowledgment goes to Surekha Maruti Salunkhe, the most significant person in my life. Your sacrifices and unwavering love made my journey to the USA possible. Through our five years of a long-distance relationship, you’ve shown me the true essence of love, proving that it can withstand any distance.

I express my sincere thanks to the mentors who played crucial roles in my academic journey. Dr. Jianshun Zhang introduced me to the world of research, providing a foundation for my subsequent endeavors. Dr. Young Moon, my Ph.D. advisor, offered guidance and support through every challenge. Dr. Fanxin Kong provided valuable insights into my research, while Dr. Xiyanu Liu honed my skills as a teaching assistant. I would also like to express my sincere thanks to Dr. Victor Duenas and Dr. Yuzhe Tang for being part of my defense committee. To each of you, my heartfelt appreciation. This is for all of you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter serves as an introduction to cyber manufacturing systems (CMS), delving into security concerns that have gained attention from the researchers. It highlights the gaps in existing research and outlines the scope of the study, laying the foundation for a thorough investigation into recovery strategies tailored for cyber manufacturing systems.

1.1 Cyber manufacturing systems

Manufacturing systems are transforming to meet evolving customer demands (J. Lee, Bagheri, and Kao 2015). The integration of digital technologies, cloud-based systems, robotic arms, and automated material handling, as well as enhanced assembly lines, has aimed to augment standalone manufacturing systems, improving both operational efficiency and responsiveness to consumer needs (Mavrikios et al. 2013). However, despite these advancements, the Bureau of Labor Statistics reports only a 0.7 percent off annual growth in production from 2007–2018, compared to 3.6 percent from 1987–2006 (Labor Statistics 2019). Recognizing the need for innovation, the recent wave of manufacturing progress introduces Cyber-Manufacturing Systems (CMS) to potentially elevate stagnant productivity levels (Z. Song and Young B Moon 2016).

It's essential to distinguish between cyber manufacturing systems and manufacturing processes. While manufacturing processes are confined to a specific procedure or set of steps in the production of an item, manufacturing systems are a comprehensive and integrated framework that encompasses various processes, resources, and interactions, working in concert to produce the desired end product. According to Song, CMS integrates manufacturing systems with cyber components such as analytics, sensing, control, prediction, and monitoring, collectively enhancing manufacturing systems performance (Z. Song and Y. Moon 2017).

Cyber manufacturing systems, designed to adapt to achieve the intended objectives, involve individual manufacturing processes, their rela-

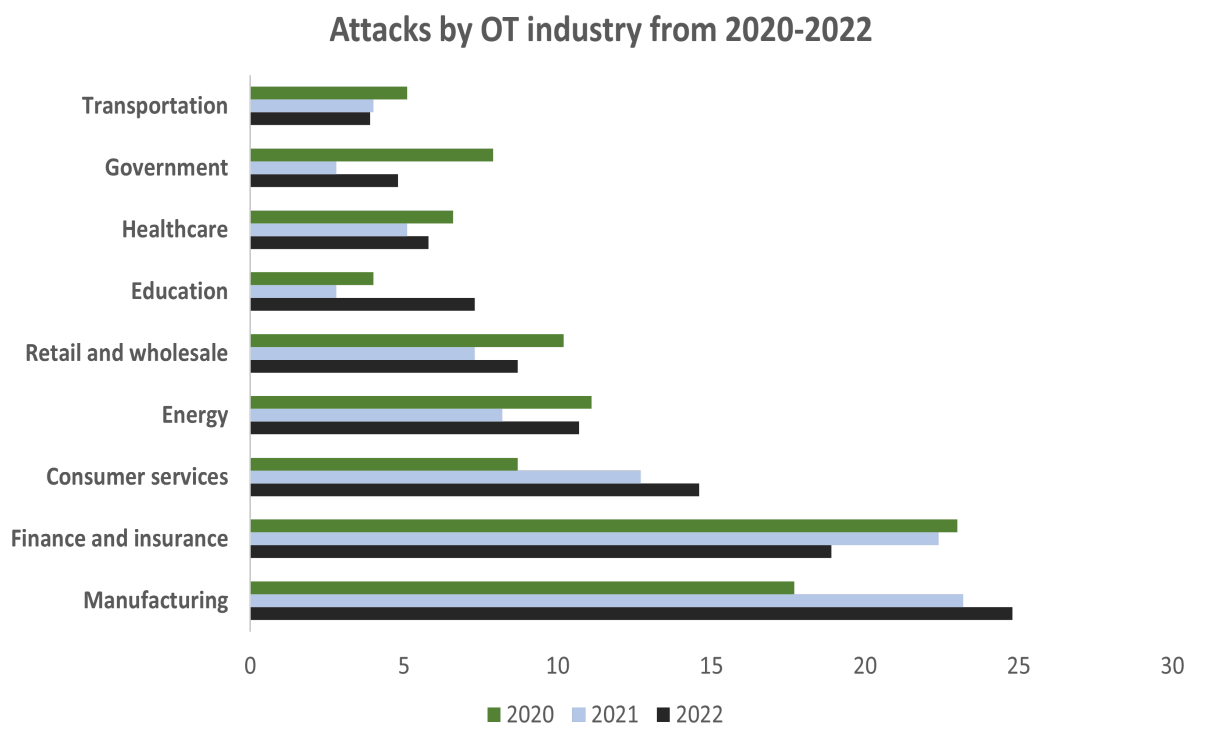


Figure 1.1: OT industry explored by cyber attackers from 2020-2022 (Worley et al. 2023).

tionship, and interactions with the cyber components and the operational environment (Z. Song and Y. Moon 2017). The interaction between and among cyber components and manufacturing systems is achieved through wired or wireless networks (Wu 2019; Yampolskiy et al. 2018). This integration, while providing benefits, also exposes these systems to potential threats from cyber attackers (Yampolskiy et al. 2018; Wu and Young B Moon 2017).

According to the report published by IBM X-Force Threat Intelligent Index in 2019, manufacturing ranked as the 8^{th} most attacked industry in 2019 and 2^{nd} in 2020 (Worley et al. 2023). It is evident from the report and Figure 1.1 that manufacturing is the most attractive operational technology industry for cyber attackers. The report mentions that *"man-*

ufacturing is more vulnerable to cyber-manufacturing attacks and future trigger events, or new attack tactics may lead to damage to manufacturing infrastructure and potentially human lives." While the motive behind most attacks is to leak intellectual properties and trade secrets, the report indicates that attackers may shift their focus toward controlling manufacturing processes due to their lower tolerance for downtime. Manufacturing companies often hesitate to disclose cyber-attacks, impacting customer trust, and potentially resulting in underreported breaches

A Deloitte article with Manufacturers Alliance for Productivity and Innovation's (MAPI) cyber security for smart factories states that around 40 percent of manufacturers surveyed reported that their operations were affected by security in the past (Wellener et al. 2019). Major causes of concern include ransomware, spoofing, unauthorized access, operation disruption, and intellectual property theft. The attack vectors used by the attackers reported in the survey are dominated by poor client-facing web applications and phishing attacks as shown in Figure 1.2. The average financial loss from each data breach is estimated to be approximately 7.5 million dollars. While around 90 percent of manufacturers surveyed claimed that they can detect attacks using information technology (IT) strategies, very few companies have extended their IT monitoring systems to operational technologies. Furthermore, less than half of the manufacturers surveyed have conducted security assessments in the last six months or so.

Hence, it is important to analyze and understand the cyber-attacks conducted on cyber manufacturing systems that are reported and demon-

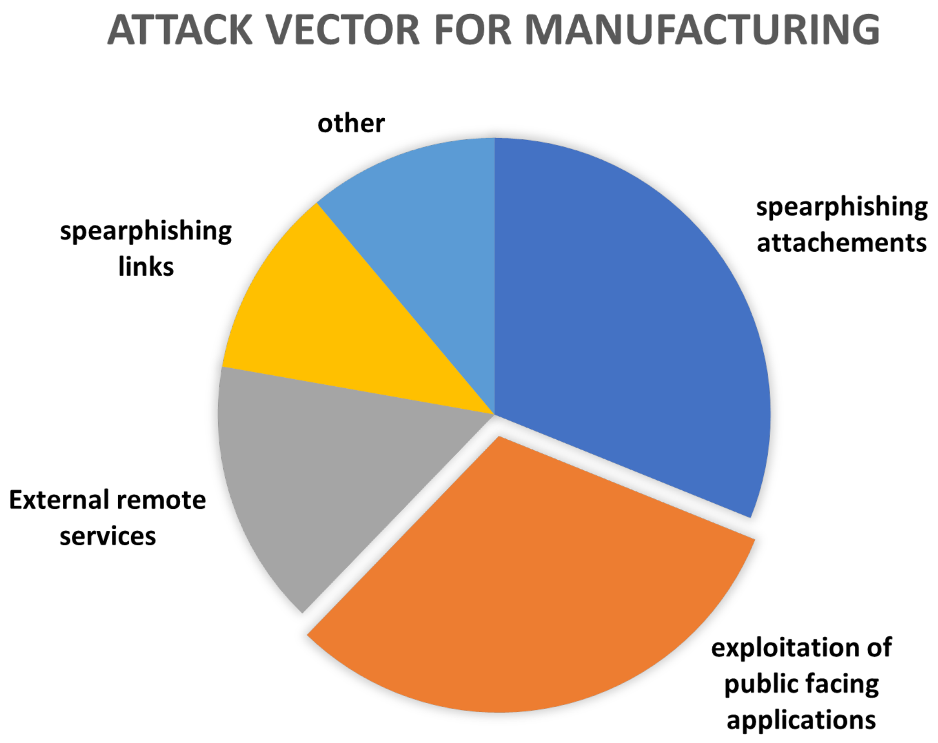


Figure 1.2: Attack vectors used by attackers in manufacturing (Wellener et al. 2019).

strated in research. The next section provides an in-depth insight into cyber attacks conducted on cyber manufacturing systems by both malicious actors and researchers.

1.2 Cyber manufacturing attacks reported in industry

The Stuxnet attack on the Iranian Uranium enrichment plant is the first publicly known cyber-physical attack. It was a complex and innovative attack that did not follow the conventional confidentiality, integrity, and availability model (Langner 2011). It infected PCs running on the Windows operating system, entering through USB sticks, spreading over local networks, and controlling particular Siemens PLC. It verified the model number, configuration details, and fingerprint to ensure it was on target. Then, it uploaded rogue codes on the controller monitored legitimate code, and occasionally took over and replayed prerecorded input to the legitimate code to avoid detection. It made the centrifuge spin faster and fail, resulting in a non-operational plant for a few days.

The German steel mill experienced the second documented manufacturing system attack, initiated through a phishing email that allowed unauthorized access to the control system, causing plant shutdowns, production delays, and physical damage (R. Lee, Assante, and Conway 2014). In another instance, a global WannaCry ransomware attack, as reported by Reuters (Staff 2017), disrupted Renault-Nissan production, encrypting files and demanding bitcoin payments. A similar incident at Norsk Hydro in 2019 (Greenberg 2019) led to password modifications, controller shutdowns, machine deactivation, and network disconnections, incurring a substantial cost of around 75 million dollars.

Some of these attacks are not discussed in this section but as seen from Figure 1.3, there is a significant portion of the reported attacks focused on shutting down the production. Given the lower tolerance manufacturers have towards downtime, the strategy to demand ransomware to remove these attacks from the system seems a lucrative strategy.

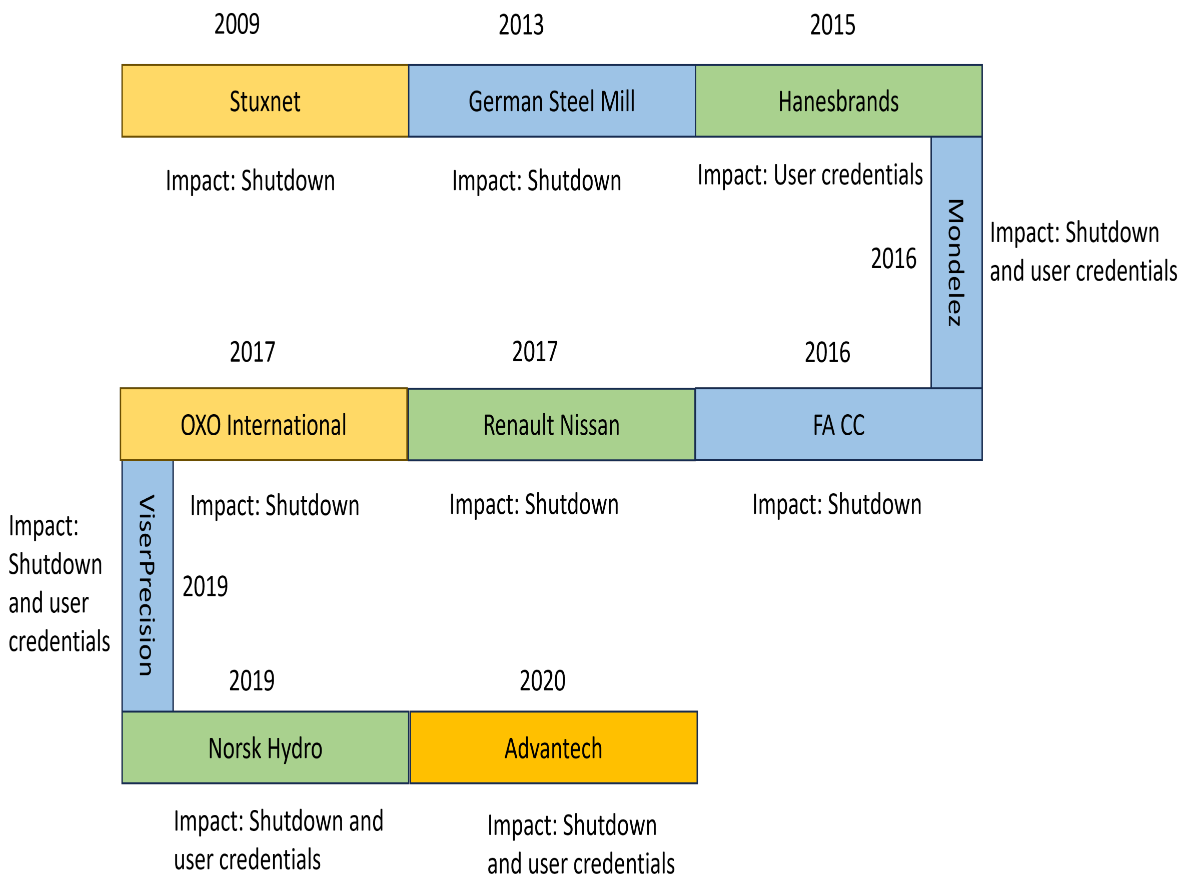


Figure 1.3: Timeline for reported cyber attacks on manufacturing facilities.

1.3 Cyber manufacturing attacks in research

Since the Stuxnet attack (Langner, 2011), the research community has intensively investigated vulnerabilities in cyber manufacturing systems. To develop a robust security strategy, a thorough analysis of attack vectors and types of attacks in CMS is important (Wu and Young B Moon 2017). Xiao (Xiao 2013) demonstrated the initial vulnerability of a manufacturing process by modifying the temperature and developing malware combining replay and false data injection attacks. In this instance, the malware modified the temperature twice the user input, while the control system received the replayed (true) data. Sturm (Sturm et al. 2014) developed malware that automatically inserted voids in submitted .stl files for the 3D printing process, resulting in tensile strength degradation of the material. Another work from the same research group created malware that introduced a tetrahedral void near complex geometry (Sturm et al. 2017).

Belikovetsky (Belikovetsky, Yampolskiy, et al. 2017) investigated the environment of the 3D printer and manipulated the blueprint file. An infected downloaded file searched for a .stl file, and upon finding it, it modified the .stl file despite of firewall and antivirus prevention strategies. The manufactured drone passed through a nondestructive quality check but failed during operation. Faruque (Al Faruque et al. 2016) used side channels to capture the 3D printer's motor noise, tracing it back using machine learning to recreate the motor movements with 78 percent accuracy. A similar attack (Hojjati et al. 2016) captured magnetic side channels.

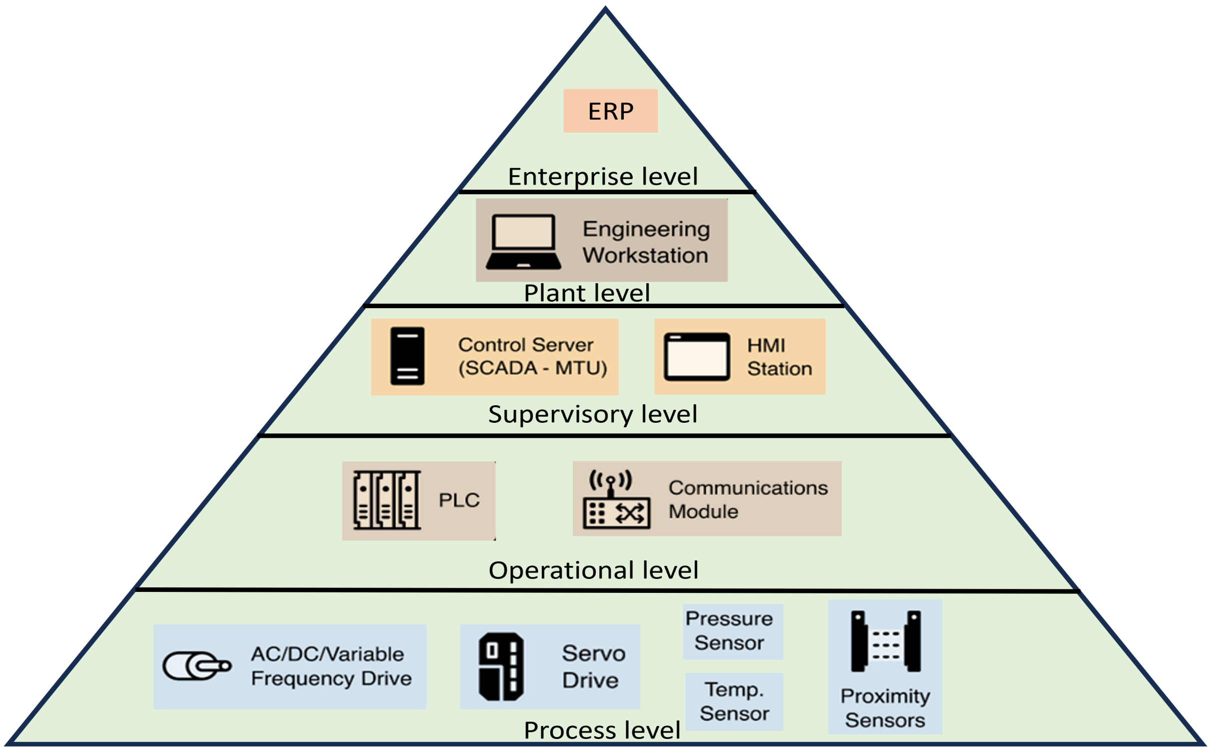


Figure 1.4: Manufacturing systems automation hierarchy.

Moore (Moore, Glisson, and Yampolskiy 2017) explored the vulnerability of using open-source software in desktop 3D printers to replace a print job with a different job. A SQL injection attack by (Wu, Z. Song, and Young B Moon 2019) on the database of CMS was demonstrated. The attacker logged in as a guest, explored the vulnerability of the website, and accessed the database. The attacker replaced the submitted file with a malicious file for additive and subtractive manufacturing operations.

Among different cyber attacks on cyber manufacturing systems demonstrated in the literature and in the real world, sensor attacks stand out as a threat with the potential to significantly disrupt manufacturing processes/systems. The foundation of automation in manufacturing is built upon the five-level industrial automation framework as depicted in Figure 1.4. Sensors are instrumental in the communication of information from process levels to enterprise levels within this framework (Salazar and Alvarado 2014). Sensors serve as the pivotal part of this framework, enabling controllers at the process level and operational/supervisory level to implement precise control algorithms that enable the functioning of cyber manufacturing systems in a predetermined sequence to yield the desired end product (Cruz Salazar et al. 2019). Moreover, they provide real-time state estimation of the manufacturing systems to the manufacturing plant level controller. With this information, the manufacturing plant-level control algorithms work to minimize any discrepancies between the manufacturing system's output and the input required for optimal manufacturing operation. However, the integrity of the information can be compromised by the actions of cyber attackers who may exploit vulnerabilities in the system's networks or software (Prasad and Y. Moon 2022b). Hence jeopardizing the manufacturing system's availability and potentially leading it into an unsafe operational state.

The following section discusses strategies discovered by researchers to counteract attacks on sensors of cyber manufacturing systems.

1.4 Security research in cyber-manufacturing systems

The threats on cyber manufacturing systems have motivated researchers to propose solutions aimed at increasing resiliency (Wu and Young B Moon 2017). At a high level, these solutions draw inspiration from the cyber security domain and can be categorized into prevention and detection strategies.

1.4.1 Prevention from cyber-manufacturing attacks on CMS

Prevention is the key to reducing cyber-attacks on manufacturing systems. It ensures the securing of digital and physical entities by blocking attacks. It can be obtained through encryption of data communicated among entities or by implementing firewalls to block dubious network traffic. Additionally, the antivirus can be added to the system to block attacks through the dictionary of attacks.

In CMS, there is an influx of data that is needed to be protected from cyber-attacks. Apart from encryption (Brandman et al. 2020; Krundyshev and Kalinin 2020; Prasad and Y. Moon 2022a), there is a new technology named blockchain that is used to gain trust among entities, and secure data, and transactions. Based on the blockchain trust ideology, (J. Song, Bandaru, et al. 2020) used the blockchain to store the UDP (User Datagram Protocol) based PLC (programmable logical controller) communication data. These data are input to control manufacturing processes via computer networks. This method was tested on a PLC and blockchain testbed. The proposed method was effective against direct modification—denial of service attacks and man-in-the-middle attacks. Shi (Shi et al. 2021) proposed a blockchain-based data storage and a Rivest-Shamir-Adleman-based asymmetry encryption technique to generate a unique hash value for each layer of G-code and store it in the blockchain architecture. The hash function was unique for different inputs. So, modification of the G-code resulted in a different hash value, thus preventing data modification.

1.4.2 Detection of cyber-manufacturing attacks on CMS

Attacks are evolving, making it impossible to prevent all attacks. Thus, it is important to include detection systems alongside prevention systems. Detection includes monitoring data and generating alerts as data deviates from the baseline model.

CMS emits side-channel noise in the form of vibrations, capturing information about the process. Researchers have leveraged this data to detect abnormal behavior. Bayens et al. proposed a verification and detection method for an insider attack on control logic and modification of PLC firmware (Bayens et al. 2017). They employed a microphone and a gyroscope to capture side channels during 3D printing and a CNC process. An algorithm verified authentication and detected the positioning of the printing nozzle relative to the base plate. The final layer confirmed the material using Raman Spectroscopy. Acoustic and spatial verifications confirmed the correct fill pattern and density in a print, while material verification determined whether the correct material was used.

Belikovetsky et al. (Belikovetsky, Solewicz, et al. 2017) presented an algorithm generating a master audio fingerprint and another algorithm to verify deviations due to tampering. A master audio print was created by receiving audio signals from four motors of a 3D printer. The audio recording was post-processed, and the output served as a master file for attack detection. The algorithm successfully detected deviations in a propeller.

Chhetri et al. (Chhetri, Canedo, and Al Faruque 2016) identified a zero-day kinetic cyber-attack on 3D printers. They implemented supervised learning to identify anomalous signals with 77.45 percent accuracy. Jacob et al. (Gatlin et al. 2019) created a master fingerprint for power tracing by capturing the current delivered to each 3D printer motor. They captured power from all the motors, converted it to a fast Fourier transformation, and verified the power signature against the master signature to detect deviations from the pattern.

Sophos (Rais, Li, and Ahmed 2021) detected nozzle-kinetic, filament-kinetic, and thermodynamic attacks on the fused deposition modeling (FDM)-based 3D printing process. It used spatiotemporal G-code modeling for attack detection and obtained a reliable state of the printing process. Sophos employed optical encoders for movement tracking, rotary encoders to detect x, y, and z axes movement, and thermocouples to measure nozzle temperature, successfully detecting attacks and benign counterparts.

Apart from side channels, hash values can be used to detect attacks. Hash values are unique for different inputs. A small change in input value will result in a different hash value. Brandman et al. (Brandman et al. 2020) used this idea to detect attacks on the 3D printer. They proposed an eight-step process creating a physical hash for each parameter and combining it with the QR code for secure transfer of part information from the machine to the monitoring station. The generated physical hash matched with the hash generated by the measurement systems, successfully verifying normal operation and detecting attacks in process parameters and toolpaths.

Furthermore, advances in computer vision and machine learning have been utilized for detection and monitoring. Wu et al. (Wu and Young B Moon 2019) used machine learning techniques, such as Naïve Bayes classifier and J48 decision trees, to detect malicious infill defects in the 3D printing process using images as input. Wu et al. (Wu and Young. B Moon 2020) presented detection strategies against man-in-the-middle attacks, constructing five types of malicious infills for the 3D printing process and two attack scenarios for subtractive manufacturing. Random forest and anomaly detection achieved good detection accuracy. Another work proposed by Wu and Moon (Wu and Young B Moon 2019) compared the proposed algorithm with SNORT and OSSEC—open-source network intrusion detection systems, demonstrating that detection through physical data was quicker and more accurate compared to SNORT and OSSEC alone.

Prakash et al. (Arul Prakash et al. 2020) introduced an image processing technique analyzing the amplitude and phase variations of the print head of a 3D printer platform arising from induced system manipulations. The method used an image sequence of the printing process to perform an offline spatial-temporal video decomposition to amplify changes attributable to a change in system parameters. A cyber-manufacturing testbed built by Jinwoo et al. (J. Song, Wang, et al. 2020) detected adversary insider manipulation (AIM) on a 3D printer from a supplier and a drawing machine from the manufacturer, using the proposed minimum mean absolute percentage error classification method against infill manipulation and image manipulation.

Another method to detect attacks was to correlate cyber and physical alerts. Wu and Moon (Wu and Young. B Moon 2020) utilized a similarity-based correlation between cyber and physical alerts to detect cyber-physical attacks, defining cyber alert correlation, physical alert correlation, and cyber-physical meta-alert correlation as the three attributes. A physical alert format captured manufacturing-specific information such as UID, Machine ID, sensor ID, and the manufacturing process. The proposed method was implemented on a 3D printer with improved accuracy against zero-day attacks. Romesh and Moon (Prasad and Y. Moon 2021) proposed an adaptive detection algorithm for network attacks based on reinforcement learning agents, detecting and generating alerts for anomalous network behavior inside a manufacturing plant and correlating the alert with the manufacturing process.

The above detection approaches can be classified within the automation hierarchy, as shown in Figure 1.5. It is noticeable that most research has focused on presenting strategies for the process and supervisory levels. Hence, it also emphasizes the importance of security in this level.

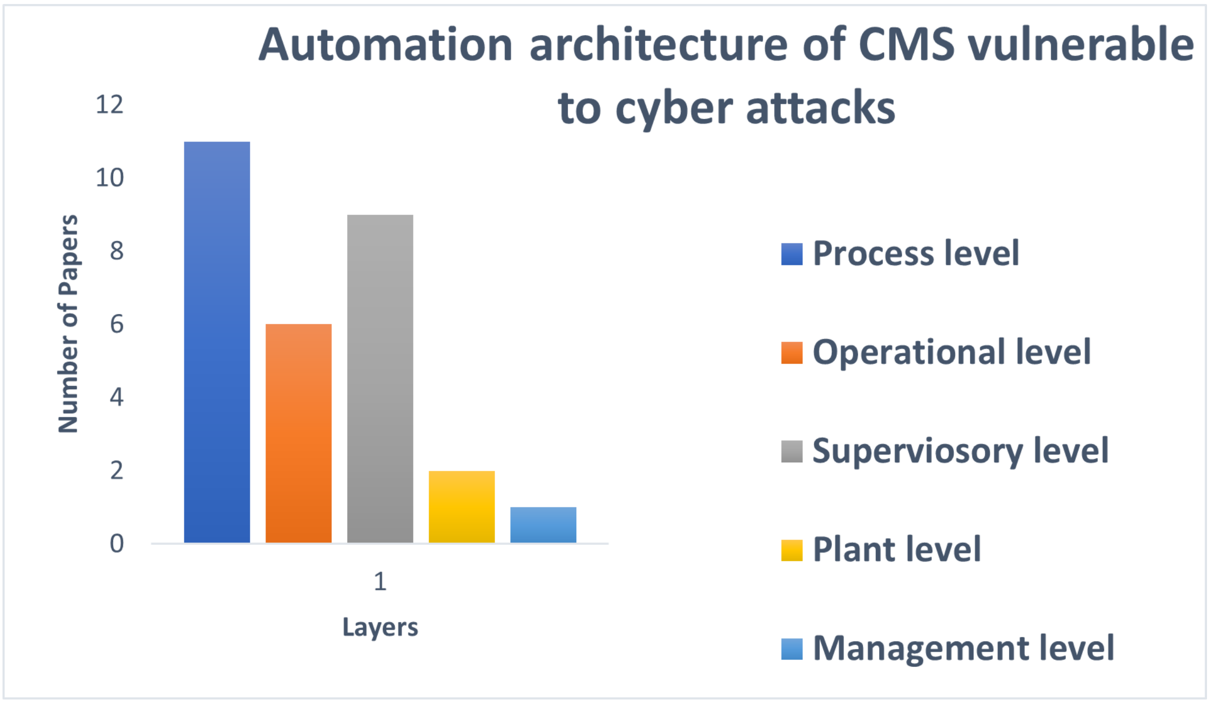


Figure 1.5: Detection and prevention strategies from literature classified among the automation hierarchy of cyber manufacturing systems.

1.5 Research gap

While the existing literature focuses on presenting detection and prevention strategies for attacks on sensors of cyber manufacturing systems, they do not answer *what to do after an attack is detected*. To ensure cyber manufacturing systems achieve their intended objectives, it is important to answer this question. Expanding our research beyond cyber manufacturing systems to any cyber-physical systems reveal that researchers have addressed these question through *recovery strategies for physical systems*. The existing literature asserts that complete resilience for any physical system can only be achieved when a system optimally recovers from a cyber attack (Standards and Technology 2018)

The Cybersecurity Strategy and Implementation Plan (CSIP) (Office and Budget 2015) defines recovery as "*the development and implementa-*

tion of plans, processes, and procedures for recovery and full restoration, in a timely manner, of any capabilities or services that are impaired due to a cyber event." NIST (Standards and Technology 2018) defines a cyber event as "*any observable occurrence in a system or network that violates security policies and best practices.*" Based on the definition above, recovery in information technology security can be summarized as a reactive measure. An example of its implementation in information technology would be reverting to the backup data in the event of a cyber attack that would lead to missing information. However, the recovery of manufacturing systems is more complex than practices used in information technology.

Thus this thesis aims to answer the question *how to efficiently recover the nominal behavior of cyber manufacturing systems without compromising the safety and performance.* Additionally, this work highlights the challenges associated with the development of a robust recovery strategy.

1.6 Thesis Outline

The thesis is structured into several chapters to systematically address the research gap. Chapter 2 discusses the literature on recovery strategies, highlights the challenges associated with developing or adapting presented recovery strategies for cyber manufacturing systems, and establishes the research objectives. Chapter 3 presents the recovery systems architecture developed to address the research objectives. Chapter 4 presents the experiment conducted to validate the functioning of recovery systems architecture. Chapter 5 focuses on engaging in a discussion of the findings in the context of recovery strategies for cyber manufacturing systems.

Chapter 2

Literature review and research objectives

While much research has focused on the prevention and detection of sensor attacks in cyber manufacturing systems, there is a noticeable gap in understanding *what to do after an attack is detected*. This chapter investigates challenges in the development of a robust recovery strategy for cyber manufacturing systems, investigates the contribution made by researchers not limited to the manufacturing domain to address this question, and the limitations of adapting the presented strategy for cyber manufacturing systems. This chapter ends with establishing research objectives.

2.1 Recovery strategy

Ideally, cyber manufacturing systems should encompass all the security measures to develop a resilient system (Espinoza-Zelaya and Young Bai Moon 2022). However, it is not possible to engineer a system to be completely secure and trustworthy (Ron, Mark, and M. Michael 2022). Thus any deviation from the normal trajectory of the manufacturing systems to an abnormal trajectory can hinder its ability to perform the desired tasks. Ensuring the availability and performance of cyber manufacturing systems during such time is possible through robust recovery strategies. In addition, the strategies when implemented should be capable of recovering from failure to either resume normal operation or degraded operation or move to an alternative process, all while ensuring that the strategies do not violate security norms.

According to NIST (B. Michael et al. 2016), different recovery strategies can be applied once the attack is detected as shown in Figure 2.1. These are reconfiguration, restart, and forward or backward recovery.

2.1.1 Restart

Ideally, upon detecting an attack, the optimal way to eliminate the threat from the system involves following the appropriate procedures to restart the manufacturing systems. This approach not only prevents further system failures resulting from the attack but also mitigates the potential spread of attacks to other systems. However, restarting cyber manufacturing systems is a non-trivial task. Machines within these systems

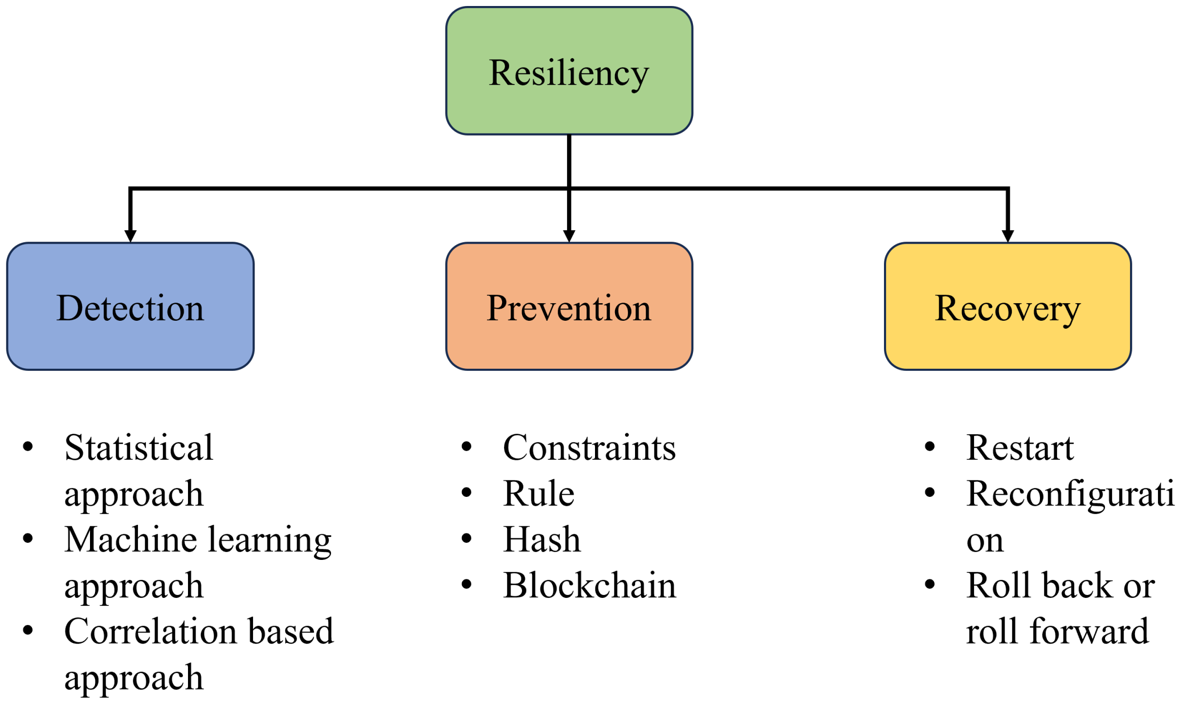


Figure 2.1: Resiliency (B. Michael et al. 2016).

are often in the midst of their operations, and due to attacks, the state of the controller within the process and operational/supervisory layer can be inconsistent with the physical state of the system. Peter (Loborg 1994) has outlined several approaches to restarting cyber manufacturing systems with minimal loss in time and material.

A straightforward restart method involves discarding the material of the infected machine and reinitializing it (Der Jeng 1997). However, discarding materials can be costly, especially in manufacturing facilities such as shipping, automotive, and aviation (Andersson, Lennartson, and Fabian 2006). Therefore, the concept of *restart points* was introduced (Loborg and Törne 1996; Tittus et al. 2000), where the machine restarts from points deemed safe for resuming normal operation. Anderson proposed a combination of automatic and manual operations, as illustrated in Figure 2.2,

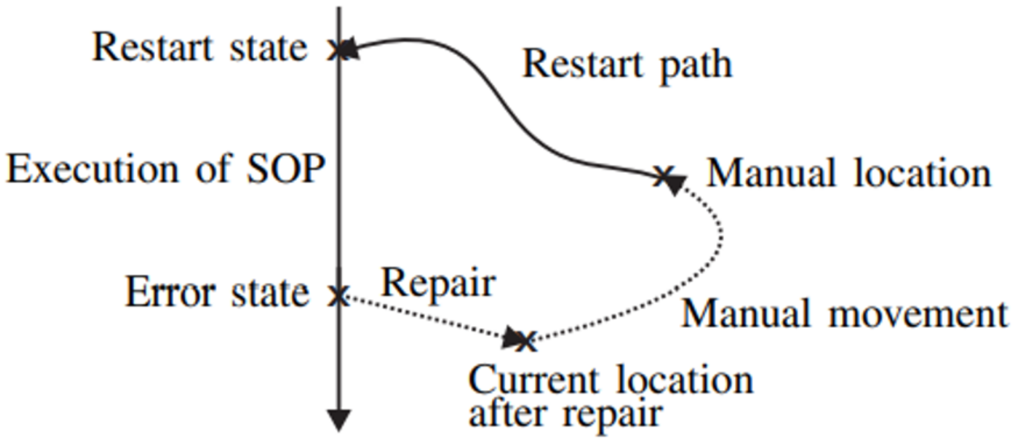
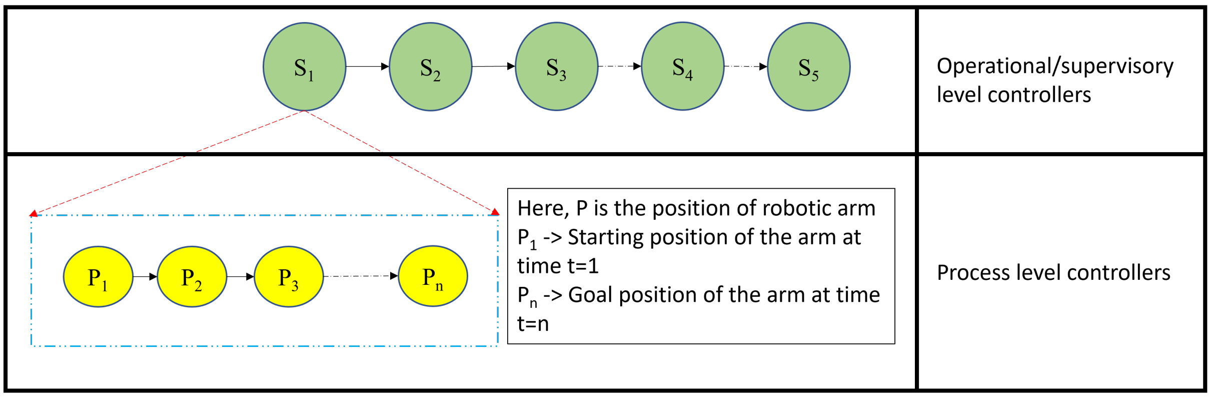


Figure 2.2: Restart a machine via restart state. The dotted line is a manual operation and the solid line is an automatic operation (Andersson, Lennartson, and Fabian 2006).

to delay the restart time. After detecting an error, the last normal state, known as the *error state*, is saved. An operator then manually moves the machines to a predefined location. Through the restart path, the machine is guided to the restart state, and the process between the error state and restart state is re-executed. This approach reduces the time required to restart the system and streamlines the restart path's dimensionality.

The limitations of these approaches include the need to add a restart path to the controller for each manufacturing process, which is challenging for large systems like automobiles and is often omitted from industrial controller programs. Moreover, the time needed to restart the machine introduces additional cycle time to complete the process. Additionally, existing literature primarily addresses restarting the system from a sensor error rather than a cyber attack. Furthermore, the proposed work only considers the operational/supervisory level of the CMS automation hierarchy, which is sufficient for controlling the sequence of operations.

Assembly operation	Tasks
S_1	Pick the object from inventory
S_2	Place the object on AGV
S_3	Move AGV
S_4	Pick the object from AGV
S_5	Place the object on conveyor



S_1
S_2
S_3
S_4
S_5
Operational/supervisory level controllers
P_1
P_2
P_3
P_n
Here, P is the position of robotic arm
P_1 -> Starting position of the arm at time t=1
P_n -> Goal position of the arm at time t=n
Process level controllers

Figure 2.3: Example: Cyber Manufacturing systems.

2.1.2 Reconfiguration

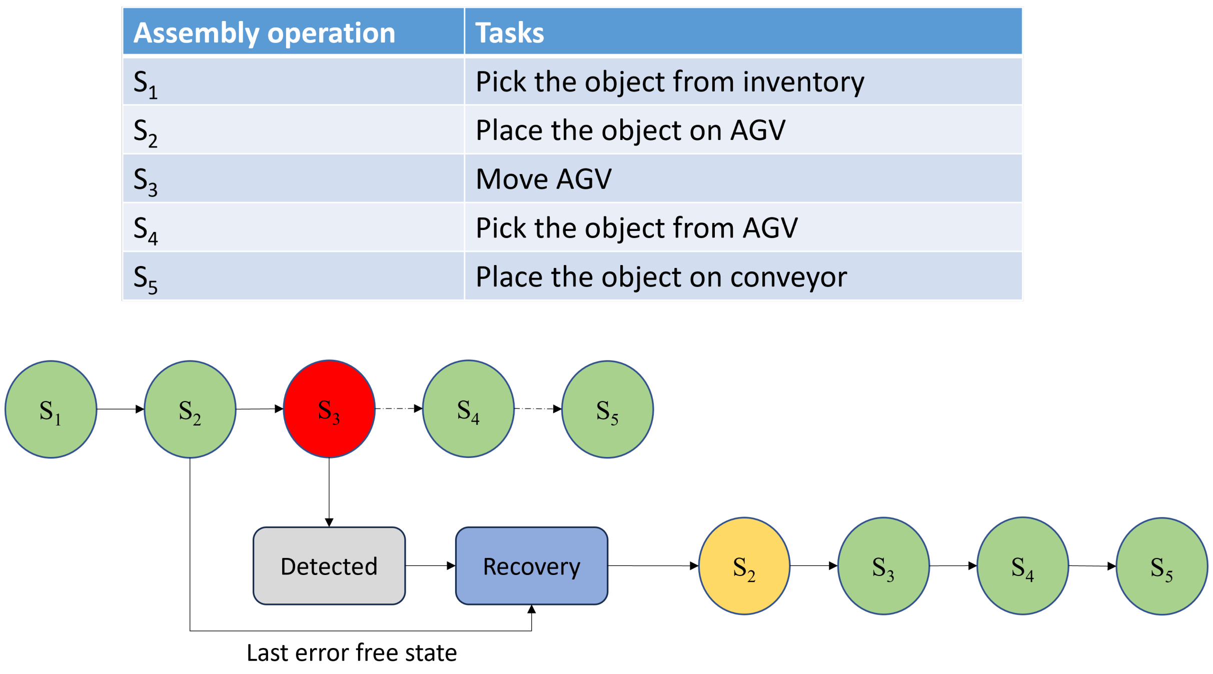
The second technique employed as a recovery strategy involves dynamically reconfiguring the control software of cyber manufacturing systems without affecting the regular process. Reconfiguration is accomplished by effectively devising an alternative manufacturing sequence of operation. These alternative approaches can be categorized as offline and online. The offline approach utilizes a human expert to direct the system to an alternative path, allowing the normal process to resume (Klein, Jonsson, and Bäckström 1999). In contrast, the online approach employs autonomous agents to plan an alternative route once the attack is detected (Lepuschitz et al. 2011; Kovalenko et al. 2023).

2.1.3 Forward or backward recovery

While the other approaches successfully recover however they have limitations such as an extensive search for alternate reconfiguration paths constrained by the availability of resources and planning for restart paths. Hence researchers have heavily invested in developing strategies that can either move the failed system forward or move the failed system back to the normal state. This approach has been explored from the *error recovery of manufacturing systems* perspective.

To illustrate this approach, consider the material handling scenario within a factory, as depicted in Figure 2.3. The operational level controller governs the sequence of operations denoted by S_1 to S_5 within the system. Each discrete state S is detailed in Figure 2.3, along with the process level controller for one of the discrete states. The positions of the motors (P_1 to P_n) represent the start position at time t_1 to the goal state S_n. In this example, the start process level state at t_1 is the idle position of the motor, and the S_n state is the position of the material in the inventory.

The literature of the *error recovery of manufacturing systems* focuses on the operational level controllers. The backward recovery approaches presented in literature (Afonso et al. 2008; Haerder and Reuter 1983; K. Kim and Welch 1989; Watanabe and Sakamura 1995) can be summarized into a strategy of recording the last error-free state and a recovery algorithm guides the impacted system to the recorded state. For example in the material movement example demonstrated in Figure 2.4 there are 5 sequences of operation represented by S_1 to S_5. The attack occurs on S_3

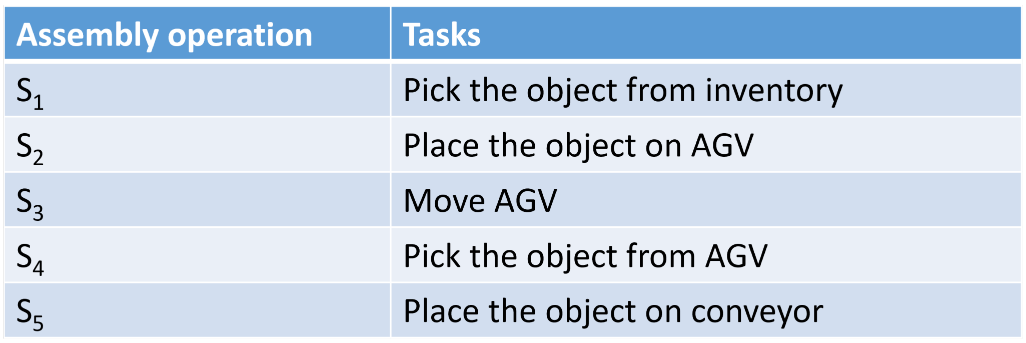


Assembly operation	Tasks
S_1	Pick the object from inventory
S_2	Place the object on AGV
S_3	Move AGV
S_4	Pick the object from AGV
S_5	Place the object on conveyor

Figure 2.4: Operation level backward recovery.

and hence the recovery algorithm reverts to the state of S_2, which is the last recorded safe state of the system. A different approach called forward recovery (Chenm and Trivedi 1991; Sousa and Santos 2007; Smara et al. 2022) in a similar environment is to move the state of the system to a predicted future state. In Figure 2.5 $S^{‘}$ represents the predicted future state that the system is recovered and then the system resumes normal operation thereafter.

The strategies proposed, despite their effectiveness in error recovery, are predominantly centered on the operational level controller. To encompass the process level controller, we broaden our literature review to include cyber-physical systems.



Assembly operation	Tasks
S_1	Pick the object from inventory
S_2	Place the object on AGV
S_3	Move AGV
S_4	Pick the object from AGV
S_5	Place the object on conveyor

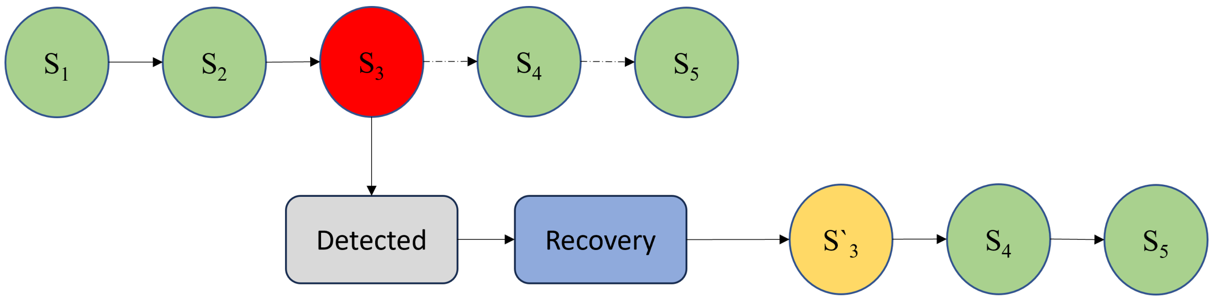


Figure 2.5: Operation level forward recovery.

2.2 Researchers addressing recovery in cyber-physical systems

Since the work of recovery related to the process level controller is not highlighted from the context of cyber manufacturing systems, this thesis explores recovery research conducted on cyber-physical systems. Recovery in cyber-physical systems deals with recovering the speed of the autonomous vehicle or recovering the voltages of the motors. Within the process level, we require mechanisms that can recover such sensors and hence drawing inspiration from cyber-physical systems is important. To ensure that the literature is relevant to the scope of the work, papers that highlight the recovery of the physical system, especially from attacks on sensors are focused during this study. The literature can be summarized into either rollback approaches or roll-forward approaches. These approaches are explained in depth in the following sections along with the limitations.

2.2.1 Rollback state

In this context, Kong highlights the necessity of a backup controller algorithm to facilitate the recovery of a physical process after sensor attacks occur (Kong et al. 2018). In the backward rolling, it tracks the last known safe state and rolls the system to that state. Figure 2.6 demonstrates this strategy with an example. This example looks only at the process level controllers represented from P_1 to P_n. Here P represents the position of the robotic arm that is tasked to pick up an object from inventory. The trajectory it takes to reach from the idle position to the goal position is shown in the bottom section of Figure 2.6. In this strategy, the last safe state of the system is saved and once the attack is detected and recovery is initiated the algorithm guides the system to the last safe state. Here it is represented as P_{12}.

However, Kong's discussion also brings to light certain drawbacks associated with implementing a recovery strategy that rolls the system back. This approach is burdened by overhead costs and is often rendered unfeasible due to the irreversible nature of many manufacturing processes.

2.2.2 Roll forward state

Consequently, the focus has predominantly been on forward recovery, which appears to be a more practical and viable strategy. In the context

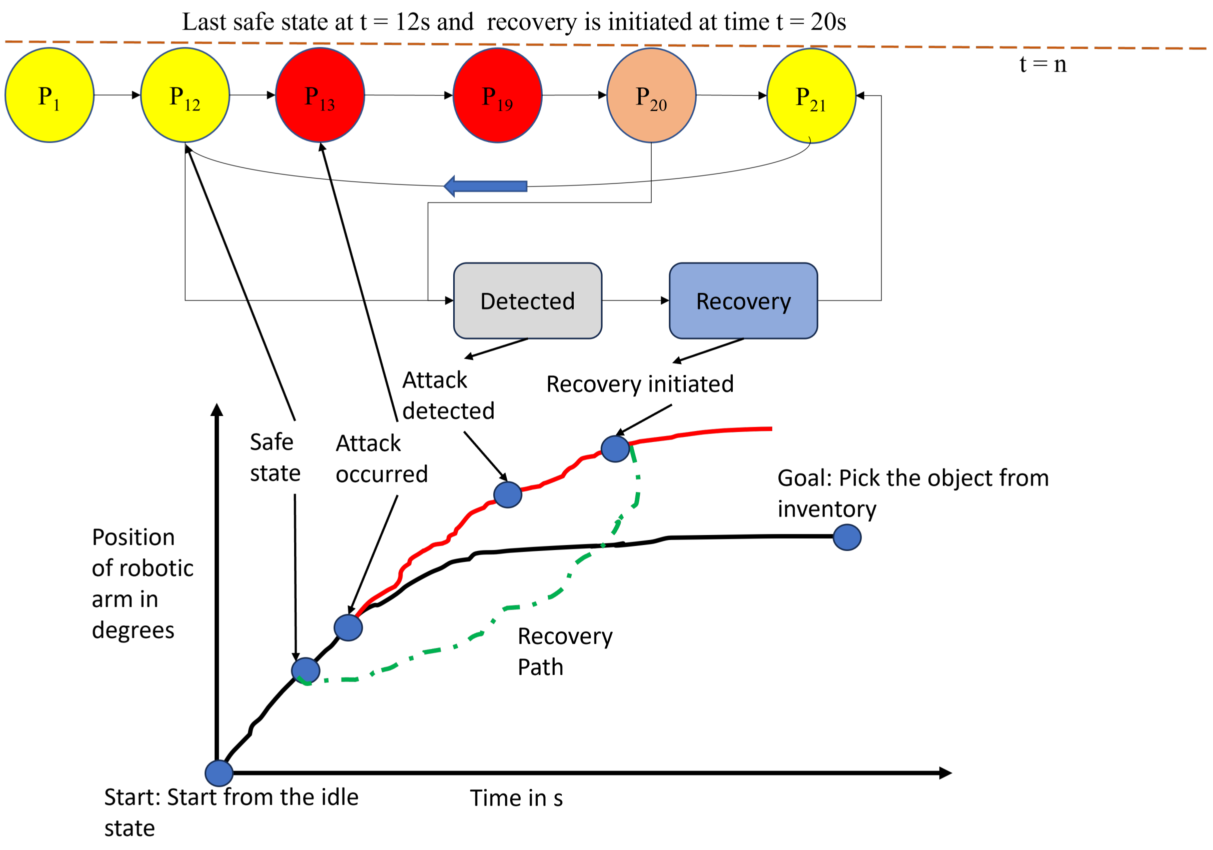

Last safe state at t = 12s and recovery is initiated at time t = 20s
t = n
P_1
P_{12}
P_{13}
P_{19}
P_{20}
P_{21}
Detected
Recovery
Attack detected
Recovery initiated
Safe state
Attack occurred
Goal: Pick the object from inventory
Position of robotic arm in degrees
Recovery Path
Start: Start from the idle state
Time in s

Figure 2.6: Process level backward state recovery.

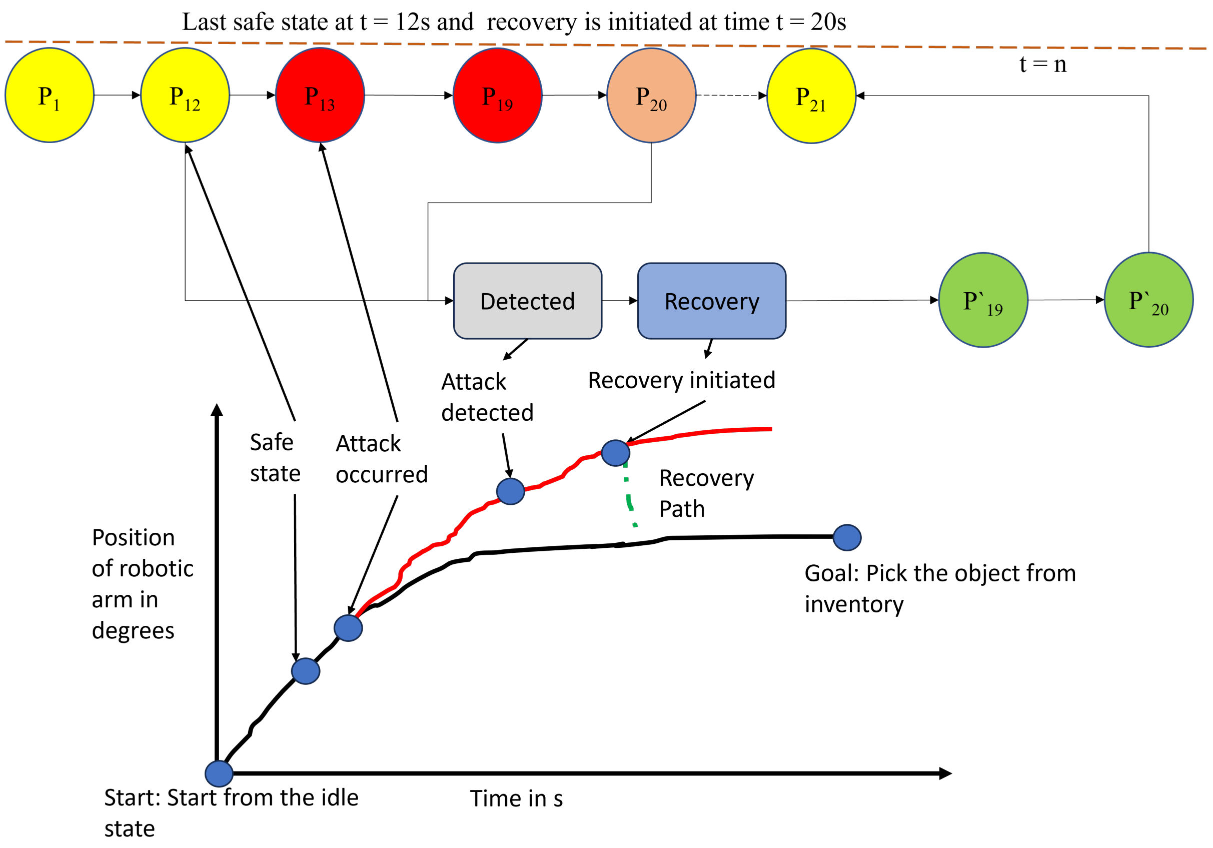

Last safe state at t = 12s and recovery is initiated at time t = 20s
t = n
P1
P12
P13
P19
P20
P21
Detected
Recovery
P`19
P`20
Attack detected
Recovery initiated
Safe state
Attack occurred
Recovery Path
Position of robotic arm in degrees
Goal: Pick the object from inventory
Start: Start from the idle state
Time in s

Figure 2.7: Process level forward state recovery.

of forward recovery, achieving a precise estimation of the system's future state is challenging (Kong et al. 2018). This estimation can be accomplished by modeling the systems as linear time-invariant and implementing various control policies such as linear quadratic regulators (Zhang, Lu, et al. 2021), linear approximations (Zhang, Lu, et al. 2021), and predictive control (Zhang, Sridhar, et al. 2023). These control strategies predict the future state and the controller rolls the system into these predicted states. An alternative approach to forecasting the future state of the system is to employ data-driven modeling techniques. This entails utilizing deep learning methods like long short-term memory (LSTM) to predict the system's future state based on historical data (Akowuah et al. 2021). Such data-driven models can enhance the accuracy of forward recovery strategies, making them more effective in mitigating the impacts of sensor attacks on manufacturing processes.

Figure 2.7 demonstrates the forward recovery strategy with an example. Similar to the example in the section 2.2.1 it looks only at the process level controllers represented from P_1 to P_n. The trajectory it takes to reach from the idle position to the goal position is shown in the bottom section of Figure 2.6. In this strategy after an attack is detected and recovery is initiated the system predicts the future state of the system $P`_{19}$ from the present state of the system $P_{20}$ and guides the system towards the predicted states. The new trajectory represented as $P`_{19}$ and $P`_{20}$ is the path taken by the forward recovery strategy to resume the normal operation of the system.

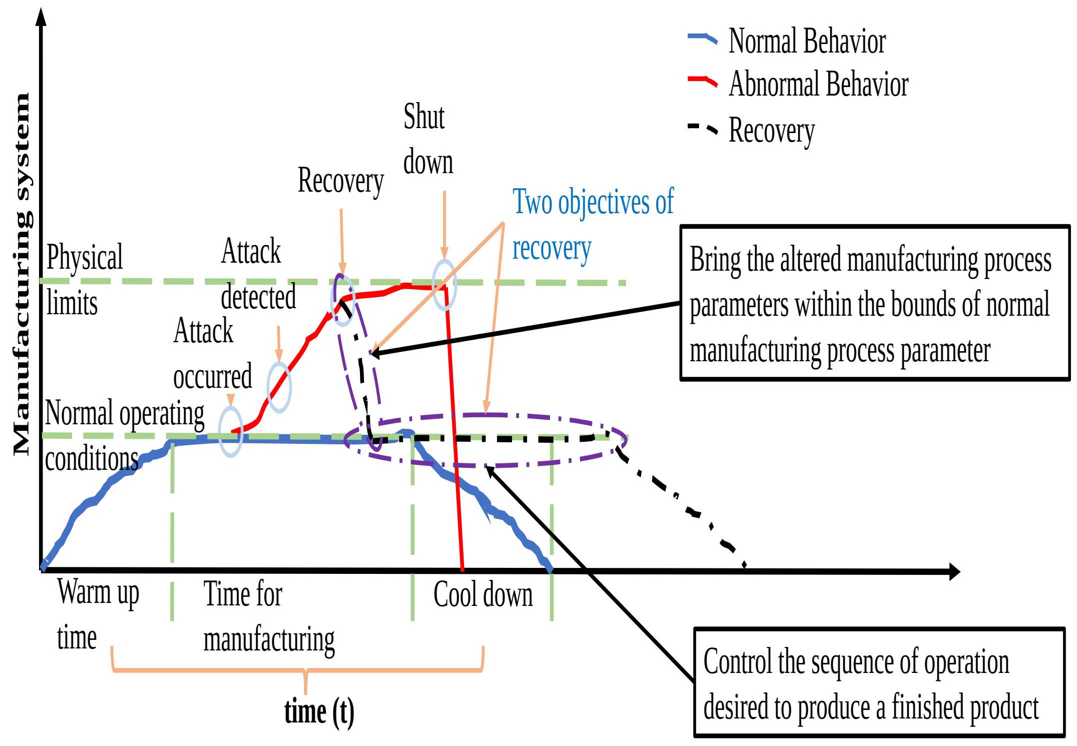
2.3 Challenges for developing a recovery strategy for CMS

Adapting the forward-rolling approach for cyber manufacturing systems presents a significant challenge due to the distinctive automation architectures in place. These manufacturing systems utilize a five-level automation hierarchy depicted in Figure 1.4. At the process level, controllers manage direct control of physical processes and equipment, including sensors, actuators, switches, and field devices that interact with manufacturing processes. These devices collect real-time data and provide feedback to the control system. Moving up to the operational level, primarily Programmable Logic Controllers (PLCs), handle basic control functions using processed sensor signals and control techniques to operate actuators. This level allows for seamless movement of the manufacturing systems. The supervisory control level oversees multiple operational levels, coordinating their activities through systems like Supervisory Control and Data Acquisition Software (SCADA). These systems allow operators to monitor processes, make adjustments, and respond to alarms or abnormal conditions. At the highest level, the Manufacturing Execution System (MES) and enterprise resource planning integrate production planning, scheduling, and resource allocation, bridging the gap between supervisory control and business planning.

Information from sensors is exchanged across and within these hierarchy levels, making it essential to consider all levels when implementing the forward-rolling recovery strategy. Only a process-level recovery is not sufficient to successfully recover manufacturing systems. Communication between all levels at the same time is desired. Hence a recovery strategy should consider multiple manufacturing process, their sequence, and production planning. In contrast, existing literature solutions typically concentrate on individual processes such as DC motor control, quad-copter navigation, or vehicle speed regulation. Furthermore, modeling cyber manufacturing systems for recovery proves to be a challenge, given their discrete, stochastic, and nonlinear nature, making it challenging to establish closed-form solutions or precise mathematical representations. This stark contrast in scale, complexity, and modeling challenges underscores the difficulties of adapting recovery strategies designed for cyber-physical systems to the intricate and multifaceted world of cyber manufacturing systems.

2.4 Recovery from CMS perspective

An ideal recovery strategy for the cyber manufacturing systems should go beyond restoring the impacted manufacturing systems state and extend to ensuring continuous, uninterrupted operation. A conceptual recovery model for the cyber manufacturing systems is depicted in Figure 2.8. The x-axis in Figure 2.8 represents different stages of manufacturing systems, such as warm-up time, cycle time, and cooling time. Under normal operating conditions, predefined processes are executed through controllers. However, in the event of a false data injection attack on sensors, these controllers can lead the system to undesirable states. One straightforward approach to achieve recovery in such a situation is to shut down



Normal Behavior
Abnormal Behavior
Recovery
Manufacturing system
Shut down
Recovery
Two objectives of recovery
Physical limits
Attack detected
Attack occurred
Normal operating conditions
Bring the altered manufacturing process parameters within the bounds of normal manufacturing process parameter
Warm up time
Time for manufacturing
Cool down
time (t)
Control the sequence of operation desired to produce a finished product

Figure 2.8: Conceptual: Recovery of manufacturing systems.

the process and restart it. However, as depicted in Figure 2.8, the warm-up time consumes a significant amount of production time, making this option less feasible. An alternative approach involves implementing the forward-rolling approach, where the impacted manufacturing systems are restored to their original operating conditions. This can be achieved by developing alternate control algorithms. However, merely returning it to a normal operating state is insufficient. In an automation hierarchy, any disruption in the discrete events can cause other subprocesses to start and stop abruptly. Therefore, the recovery strategy must ensure that the manufacturing systems not only return to normal operating conditions within the given constraints but also continue to operate seamlessly. This continuous operation is achieved through recovery strategies in operational and supervisory controllers. By implementing a comprehensive recovery strategy that encompasses all these elements, a cyber manufacturing system can recover from disruptions and maintain its operational efficiency while minimizing production losses and downtime.

2.5 Problem formulation

The problem under investigation in this study pertains to the recovery of cyber manufacturing systems, which poses distinct challenges compared to the recovery of both pure cyber systems and cyber-physical systems. Unlike typical recovery efforts in cyber systems, which primarily focus on computational or data recovery, and recovery in cyber-physical systems, which are geared towards physical systems recovery, the recovery challenge in cyber manufacturing systems is uniquely complex. The intricacies of cyber manufacturing systems make it impractical to directly apply recovery strategies proposed in the existing cyber-physical systems literature. Furthermore, given the significant cost of the assets within cyber manufacturing systems, the potential consequences of false data injection attacks leading the systems to unsafe states are catastrophic. To mitigate these risks and losses, an effective recovery strategy for a cyber manufacturing systems must encompass several key objectives:

- It should successfully recover the impacted manufacturing process, ensuring that it returns to a normal operational state.
- The recovery strategy must ensure the continuous functioning of the manufacturing systems, minimizing disruptions and production downtime.
- The strategy should avoid the need to shut down the impacted machines, as this can result in substantial production losses.
- Importantly, the recovery plan should be designed to function without requiring additional hardware in the form of sensors, streamlining the implementation process and minimizing resource requirements.

2.6 Assumptions

Recovering cyber manufacturing systems poses a formidable challenge, leading us to formulate specific assumptions to effectively address this issue.

- This paper primarily focuses on false data injection attacks capable of manipulating sensor data.

- In instances where the attacker compromises either the software or the network controlling the process and the operational level, this solution is unable to recover such systems. The limitation arises from the absence of alternative channels for transmitting information to the systems in this study.

- The scope of this work assumes that only one attack occurs at a time, with modification restricted to the data from a single sensor on the systems.

- Should the attack result in hardware damage, recovery cannot be initiated, as the implementation does not incorporate backup hardware for the recovery process.

- This work assumes there exists a detection strategy capable of detecting attacks and generating alerts that can trigger the recovery system.

In summary, the core problem addressed in this research revolves around developing a recovery strategy for cyber manufacturing systems that effectively addresses the unique challenges and requirements inherent to these systems, ultimately safeguarding their operational integrity and mitigating the potentially catastrophic consequences of false data injection attacks.

Chapter 3

Recovery architecture for cyber manufacturing systems

The main goal of proposing the recovery architecture for cyber manufacturing systems is to serve as a general recovery framework that can be applied to resume the functioning of any manufacturing systems that are impacted by cyber attacks on sensors. This architecture includes the systems layer, attack identification layer, data auditing and detection layer, and recovery layer. The functioning of each layer is explained in the following chapter.

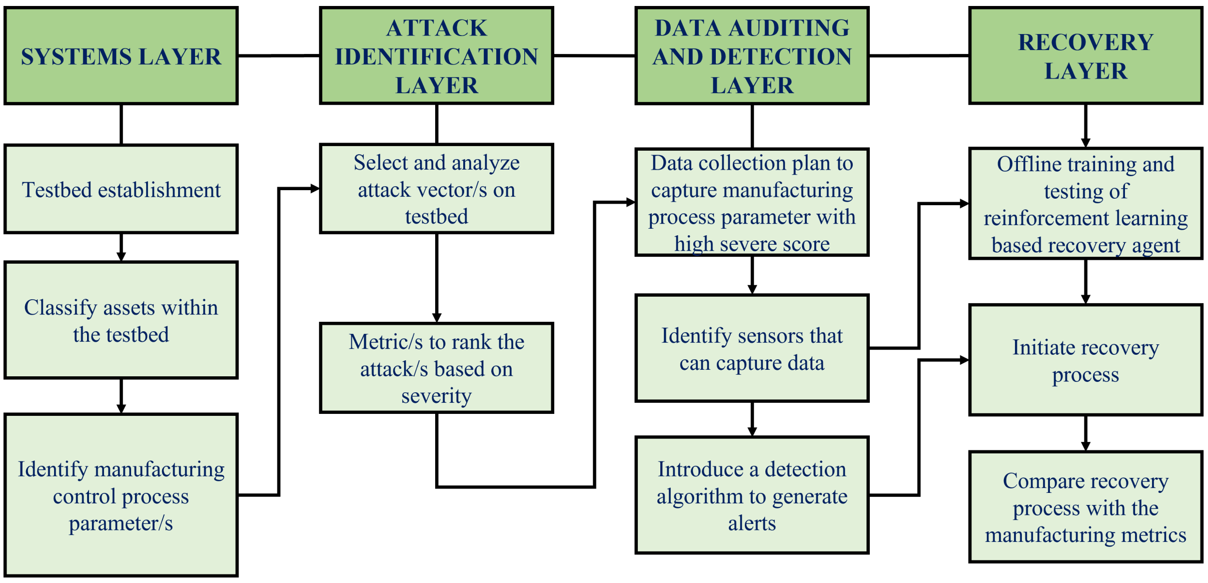

SYSTEMS LAYER
ATTACK IDENTIFICATION LAYER
DATA AUDITING AND DETECTION LAYER
RECOVERY LAYER
Testbed establishment
Classify assets within the testbed
Identify manufacturing control process parameter/s
Select and analyze attack vector/s on testbed
Metric/s to rank the attack/s based on severity
Data collection plan to capture manufacturing process parameter with high severe score
Identify sensors that can capture data
Introduce a detection algorithm to generate alerts
Offline training and testing of reinforcement learning based recovery agent
Initiate recovery process
Compare recovery process with the manufacturing metrics

Figure 3.1: Recovery systems architecture for cyber manufacturing systems.

The recovery architecture for cyber manufacturing systems is depicted in Figure 3.1 (Prasad, Mehr, and Y. Moon 2023). The layers within the architecture are specifically crafted to facilitate the recovery of cyber manufacturing systems in the event of sensor attacks. These layers include the systems layer, attack identification layer, data auditing and detection layer, and recovery layer. It is important to note that the attack identification layer is considered optional. However, during the architecture's development, this thesis necessitated an analysis of potential attacks that could harm the manufacturing process. Since not all attacks can be practically demonstrated on a testbed, a ranking matrix was devised to guide the selection of the attack. It is assumed in this work that a robust detection mechanism is in place, and therefore, detection is not explicitly addressed.

3.1 Systems layer

The functioning of the systems layer is to build cyber manufacturing systems based on the automation hierarchy. This process assists in the collection of relevant data from the cyber manufacturing systems. The objectives of this layer are as follows:

- replicating the cyber manufacturing systems through developing a small-scale version of the real factory,
- classifying the assets within the cyber manufacturing systems into process level, operational level, supervisory level, plant level, and management level
- identifying the discrete events at the process level and operational/supervisory level.

The objectives are explained in depth in the following subsections.

3.1.1 Testbed Establishment

The first objective is achieved by designing a cyber manufacturing systems testbed. The testbed is shown in Figure 3.2. To replicate the real cyber manufacturing systems, it is important to establish an objective for the testbed. The objective of the testbed is to demonstrate the movement of raw materials and finished goods within the factory.

The manufacturing testbed can be divided into two operations. The first process is moving and picking-placing the raw material/finished product from/to the inventory/warehouse. The second process is making a finished product from raw materials.

Picking and placing the raw material/ finished product from the inventory/warehouse simulates the material handling process within a manufacturing plant. Manufacturing process parameters that control this process include but are not limited to distance, positions, and availability status. This is achieved by using ultrasonic sensors to collect data about different positions (initial, final, rest, and current position), accelerometer sensors to collect speed, and current sensors to collect the current supplied data (assuming that we have a constant voltage supply).

For the second process, this testbed simulates the subtractive and additive manufacturing process. The testbed includes a CNC drawing process to simulate a subtractive manufacturing process and a 3D printer to simulate an additive manufacturing process. Manufacturing process control parameters that can be collected with sensors include but are not limited to the geometry of layer-by-layer extrusion, speed, depth, density, current, and temperature.

The security design of the testbed includes firewalls and intrusion detection systems. Apart from this, there are no other security systems placed, thus it is open to cyber-manufacturing attacks. The communication between cyber and physical entities is not encrypted. Therefore, a simple Wireshark scan can assist attackers in obtaining information. The firewall blocks traffic to the physical entities in the testbed. The firewall allows the main controller access to communicate with the internet. The main controller is registered under the IP address 192.168.60.01. The list

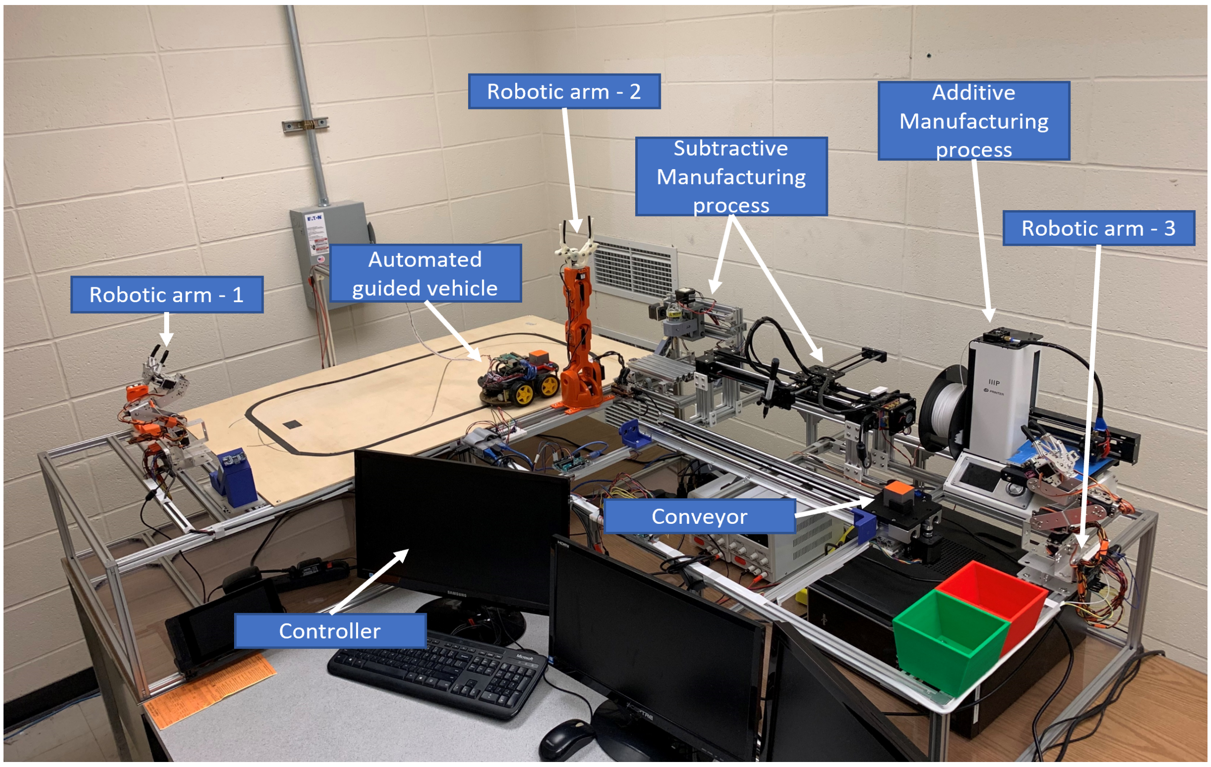


Figure 3.2: Cyber manufacturing systems testbed.

of all the registered IP addresses is presented in Table 3.1. The main controller is a computer that communicates with the webserver to send control commands to the Raspberry PI. It is also responsible for storing and retrieving information from the database. Hence, this setup allows us to investigate, analyze different attack scenarios, and identify attack vectors through which an attacker can penetrate the testbed.

Devices	**IP addresses**
Main controller	192.168.60.01
3D printer	192.168.60.10
Drilling Process	192.168.60.20
CNC process	192.168.60.05
Raspberry pi 1	192.168.60.30
Raspberry pi 2	192.168.60.40
Testbed server	192.168.60.80

Table 3.1: Devices and their IP addresses inside the testbed.

3.1.1.1 Classification of assets within the testbed

- Process level: Robotic arms, conveyor platform, automated guided vehicle, manufacturing drawing process, 3D printer, Arduino, Raspberry Pi, sensors

- Operational level/supervisory level: In the established testbed, the operational and supervisory level is combined through OpenPLC software that controls the Raspberry Pi. and Raspberry Pi controls the Arduino IDE

- Plant/Management level: The user submits the order through a web-based application. It implements a first-in first-out algorithm for the orders submitted and the status of each process and order is visualized in the plant controller.

3.1.1.2 Identifying discrete events for process level and operational level

The discrete events occurring at the process and operational level controller are given in Table 3.2 and also shown in Figure 3.3. The green solid circle in Figure 3.3 represents the operational level controller, and the PLC controls the events given by column 2 of Table 3.2. The yellow circle represents the discrete process level events given in column 3 of Table 3.2 and is controlled by their respective controller. For example, the process P_{11} represents an idle event at the process level controller for the function of the S_1 at the operational level.

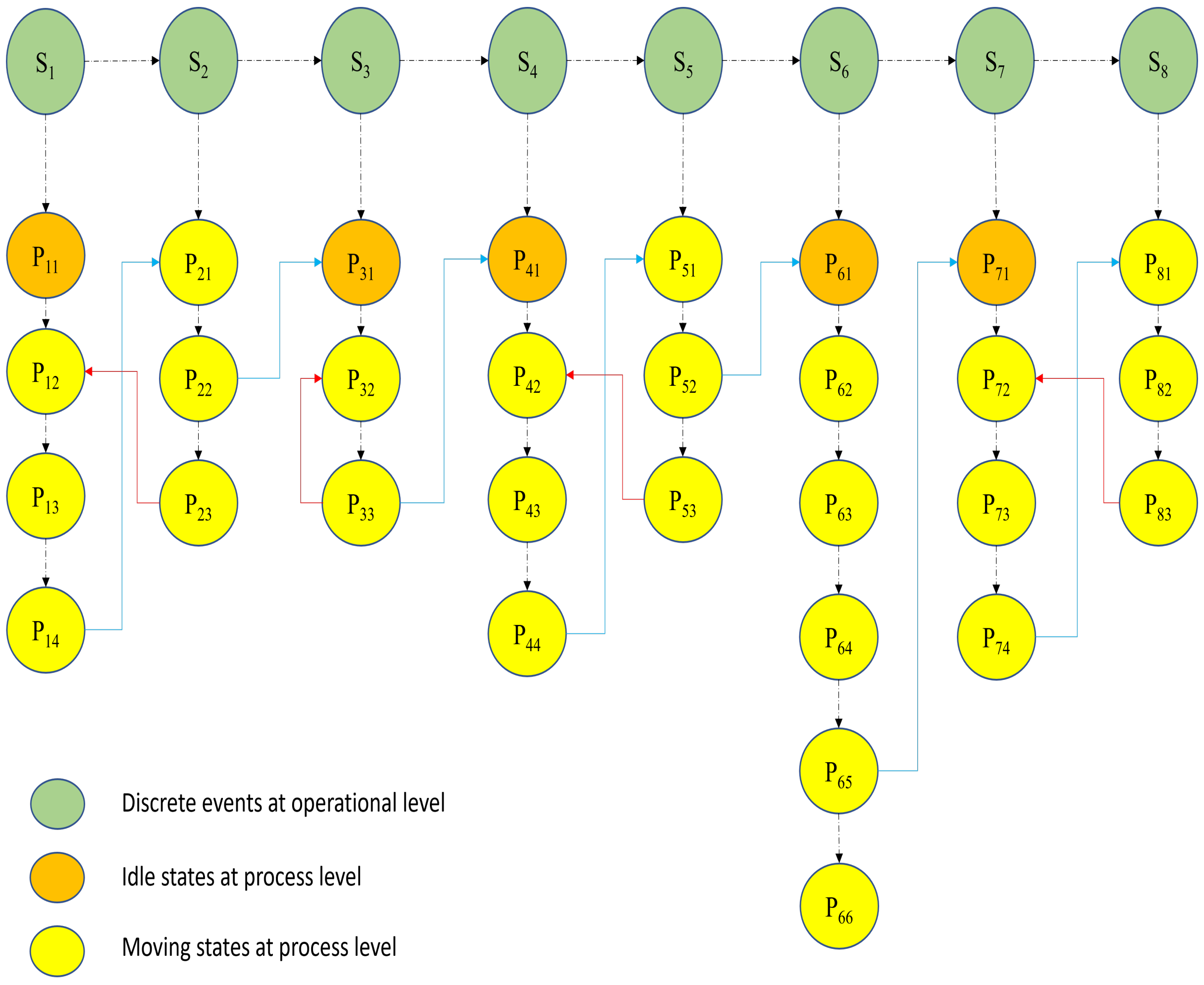


Figure 3.3: Discrete event at the process and operational level controller.

Physical assets	**Discrete events at Operational level controller**	**Discrete events at Process level controller**
R1	pick raw material from warehouse	Idle — Go to inventory — Grab the box — Pickup the box
	place raw material on AGV	Move to AGV — Place on AGV — Go to inventory
AGV	Start AGV	Idle — Move to AGV start — Move to AGV end — Move to start
R2	pick raw material from AGV	Idle — Go to AGV — Grab the box — Pickup the box
	place raw material on conveyor	Move to conveyor — Place on conveyor platform — Go to AGV
Conveyor	Start conveyor	Idle — Move to P1 — Move to P2— Hold the process — Move to P3 — Move to P1
R3	pick raw material from AGV	Idle — Go to P3 — Grab the box — Pickup the box
	place raw material on conveyor	Move to QC — Place on QC — Go to P3

Table 3.2: Discrete event at the process and operational controller level.

3.2 Attack identification layer

The second layer in the presented recovery architecture is the attack identification layer. The objective of this layer is to investigate and conduct common cyber attacks and monitor its impact on the cyber manufacturing systems. Moreover to evaluate the severity of attacks on cyber manufacturing systems a unique ranking system is developed. Details of this layer are discussed in the following section.

3.2.1 Investigation of cyber attacks

The cyber manufacturing systems established in systems layer is open to cyber-attacks. The research investigates six cyber-manufacturing attacks. These cyber-attacks include phishing attacks, sniffing and spoofing attacks, malware attacks, cross-site scripting attacks, SQL data injection attacks, and DNS rebinding attacks. For the proposed work, three goals are defined for the attackers to achieve. These consist of intellectual property theft, premature failure of products, and modification of process level controller's sensor data to increase downtime of the production. Defined goals assist in analyzing the process of conducting the attacks on the testbed. The six attacks are explained in depth in the following sections.

3.2.1.1 Phishing attack

The phishing attack is commonly used by cyber-attackers to steal information (Mityukov et al. 2019). This attack is achieved through duplication of the webpage and negligence by the user of the webpage. In this attack,

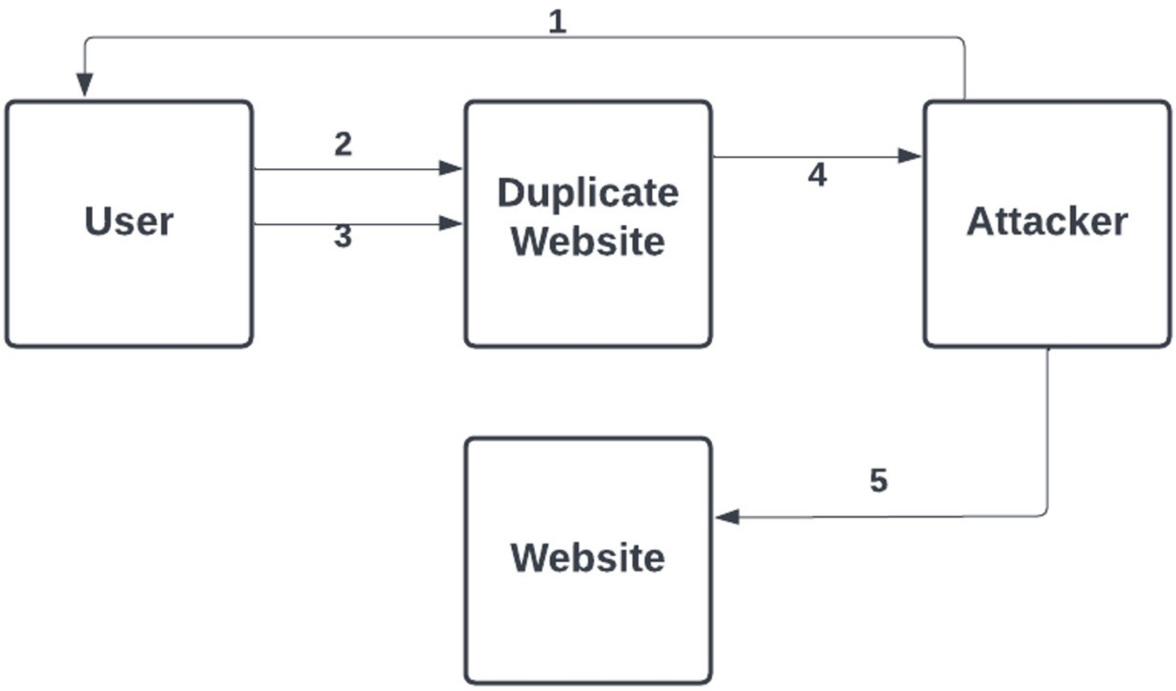


Figure 3.4: Phishing attack on CMS.

the attacker creates a duplicate webpage. It visually appears similar to the user, thus the users of the webpage do not verify the domain name of the webpage. Users enter their login information in the duplicate webpage assuming it is the real webpage. Once submitted the attacker keeps a record of the user's login information and redirects users to the original webpage. Thus attackers achieve their objective of stealing login credentials or user information.

The attacker can implement this attack on the CMS testbed to steal user credentials and use the stolen credential to alter the submitted file. The attack is conducted in four steps as shown in Figure 3.4. First, the attacker creates a duplicate webpage that appears visually similar to the original webpage created for the users to submit the order in the testbed. This is achieved by using similar colors, blocks, and fonts as that of the original webpage. Second, the user enters the login information without

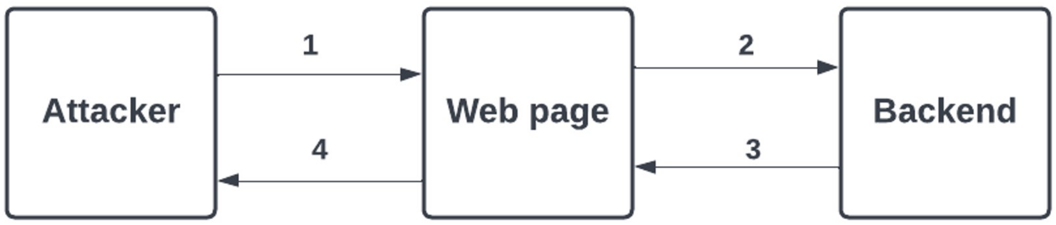


Figure 3.5: SQL attack on CMS.

verifying the domain address of the webpage. Third, after the user submits the login information the attacker's page stores the information and redirects the user to the original webpage. Fourth, the attacker used the stolen login information to enter the webpage and modify the submitted order. The modification of orders includes modification of file submitted, payment information such as credit card information, and historical information of the orders submitted by the user. The attacker achieves two goals through this attack. The stolen information results in intellectual property theft and modification of the orders results in premature failure of products or parts.

3.2.1.2 Structured query language (SQL) data injection attack

The structured query language (SQL) is a popular way to organize and retrieve information from the database. The database includes information such as past and present orders submitted, user personal information, user's payment history, and employee details (Alnabulsi, Islam, and Mamun 2014). Attacking this database can lead to fulfilling the attacker's objective of intellectual property theft and the expansion of the attack can result in premature failure of the parts.

This attack can be conducted on the testbed in 4 steps as shown in Figure 3.5. The attackers exploit the vulnerability within the syntax used by the programmer to verify the user information from the backend. The programmer failed to separate the code and data. Hence with correct logical operations or SQL commands the attacker can get access to the database. First, the attacker writes 1=1 in the block of the login page. Second, since 1=1 is always true the SQL would output all the backend information. This information is viewed by the attacker on the webpage. Similarly, an attacker can use commands such as DROP to delete the entire row or database. Thus, the attacker achieves two objectives by implementing this attack on the testbed.

3.2.1.3 Malware attack

These encompass various types of attacks such as viruses, trojans, and ransomware. The implementation of these attacks is made through USB, fraudulent email, or a link clicked by an employee, as illustrated in Figure 3.6 (H.-m. Kim and K.-h. Lee 2022). Once the malware is inside the computer system, it blocks access to the file in exchange for money or deletes important system files to disrupt production.

The attack on the testbed can originate from an employee who ignores a safety policy and clicks on a malicious link. After the user clicks on the link, the malware enters the system and looks for potential vulnerabilities. It identifies the important files and blocks the employee from accessing those files in demand of money. If the employee provides money the attacker decrypts the file. However, if the employee fails to comply with

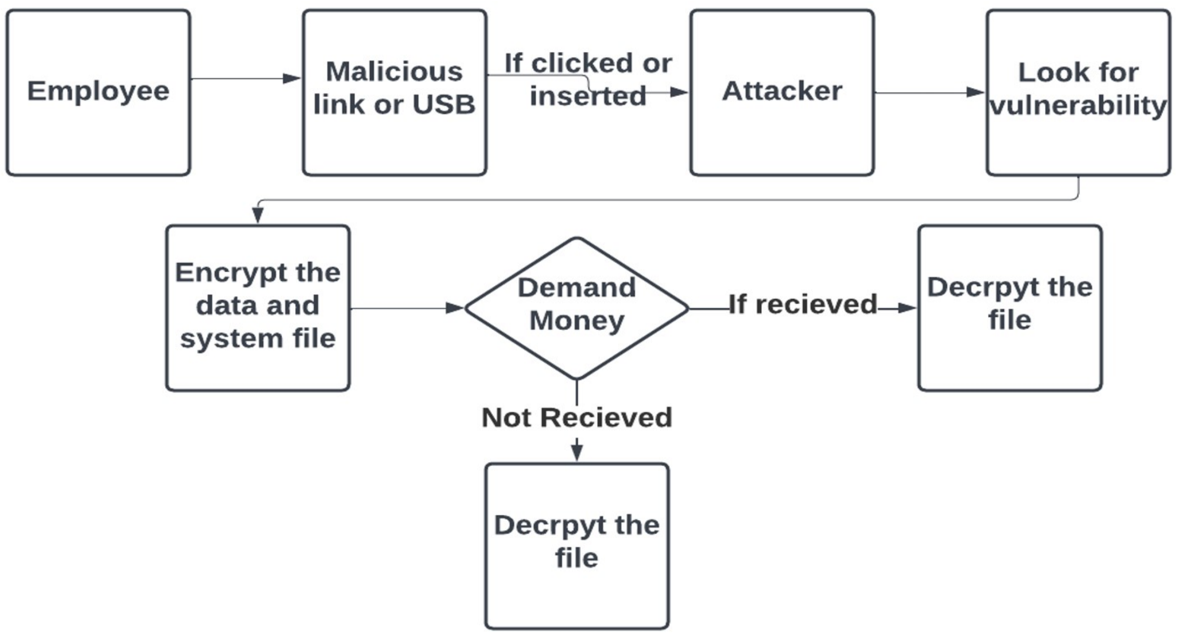


Figure 3.6: Malware attack on CMS.

the attacker's demand the file is deleted. Hence, this attack can disrupt production and steal client and employee information.

3.2.1.4 Cross-site scripting attack

Cross-site scripting (XSS) attacks are a type of data injection attack, in which malicious scripts are injected into otherwise benign and trusted websites (Pan et al. 2017). XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser-side script, to a different end user. The end user's browser has no way to know the originality of the page and thus execute the script. Because it considers that the script is from a trusted source, the malicious script can access sensitive information retained by the browser and used with that site.

Figure 3.7 shows how an attacker can use the cross-site scripting attack

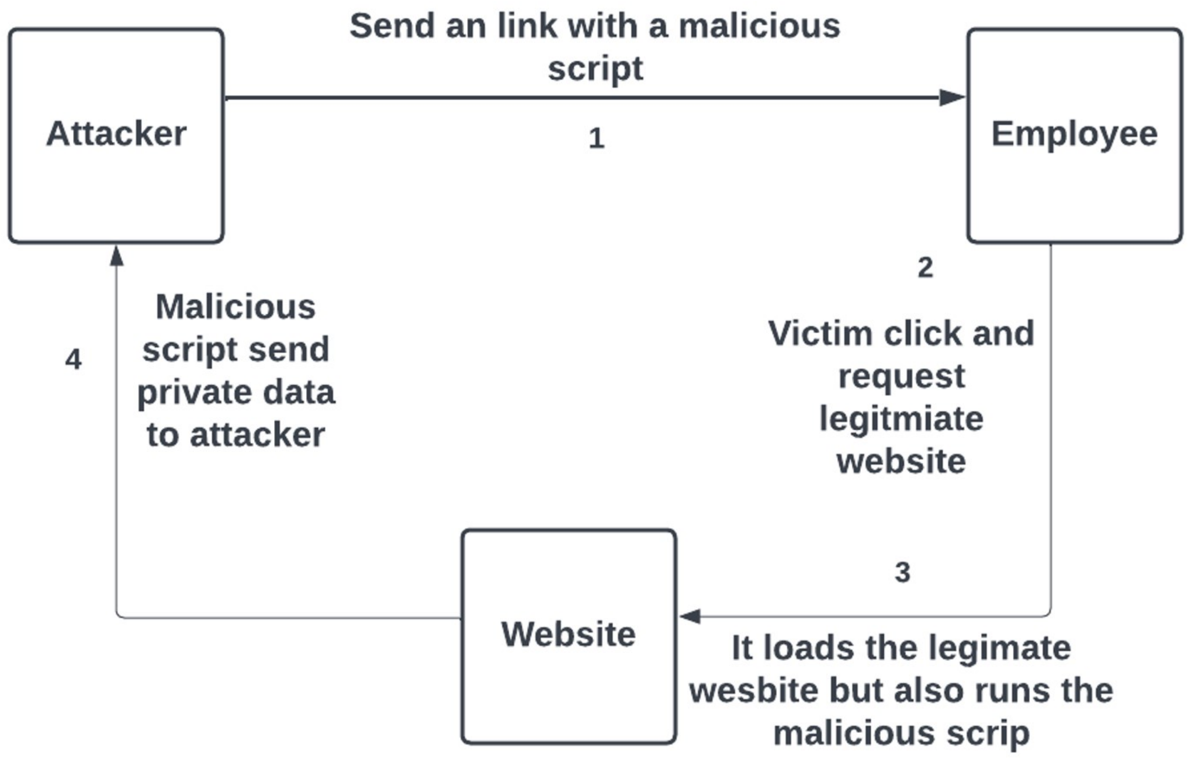

Attacker
Send an link with a malicious script
1
Employee
2
Victim click and request legitmiate website
3
It loads the legimate wesbite but also runs the malicious scrip
Website
4
Malicious script send private data to attacker

Figure 3.7: Cross site scripting attack on CMS.

to achieve the goal of intellectual property theft. The attacker sends an email to an employee, which is clicked by the employee. It results in the execution of malicious scripts. The malicious script then sent details of the employee, which includes login information to the attacker's database. Thus, giving the attacker complete information to access the system, where he/she can successfully modify the file submitted. Additionally, access the control system where he/she can make the process behave erratically. Thus, this attack achieves all three objectives. However, a countermeasure called as same origin policy ensures that these attacks are avoided.

3.2.1.5 Sniffing and spoofing attack

Sniffing attacks on the testbed are conducted by intercepting the communication between the main controller and physical entities. Since the data transmitted are not encrypted, a simple Wireshark scan results in information, readable by the attacker (Kevin 2019b). Through the implementation of this attack, the attacker achieves the goal of accessing intellectual information. This information includes process control information, product information, and user information. The attack is expanded by the attacker by spoofing the information. The spoofed information causes abnormal behavior in the manufacturing process and premature failure of the product.

Sniffing and Spoofing are common network attacks. Here the attacker monitors the data of the packets and then modifies the data inside the packets. Encryption-enabled packets are protected from modification. However, in this research, these data are not encrypted thus a successful attack can be conducted. The user submits an STL file on the webpage The 3D printer has Wi-Fi that enables sharing of the file from the controller machine to the 3D printer. The attacker conducts the attack in three steps as shown in Figure 3.8: (i) the attackers look at all the communication happening on the wireless network. They filter packets and focus on communication between 192.168.60.01 to 192.168.60.10, (ii) analysis is conducted on the packets to understand the information, and iii.) a spoofing attack is conducted by sending a modified packet request from the attacker's machine to the 3D printer. Thus, the attackers are successful in modifying the speed command of the G-code generated file and thus making the 3D printer behave abnormally.

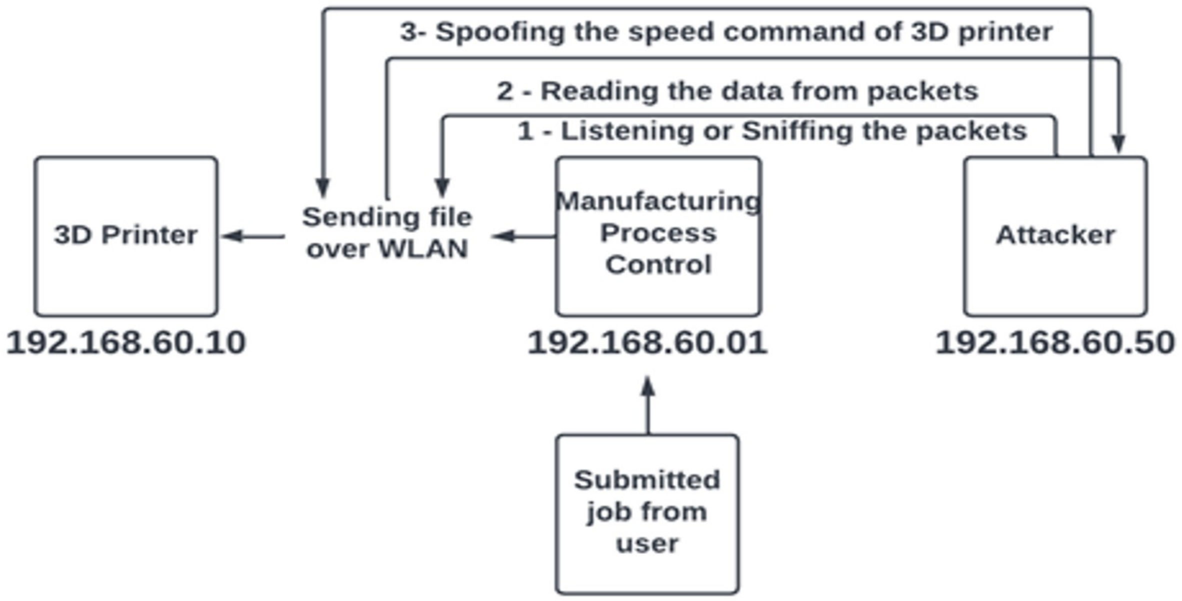

3- Spoofing the speed command of 3D printer
2 - Reading the data from packets
1 - Listening or Sniffing the packets
3D Printer
Sending file over WLAN
Manufacturing Process Control
Attacker
192.168.60.10
192.168.60.01
192.168.60.50
Submitted job from user

Figure 3.8: Sniffing and spoofing attack on CMS.

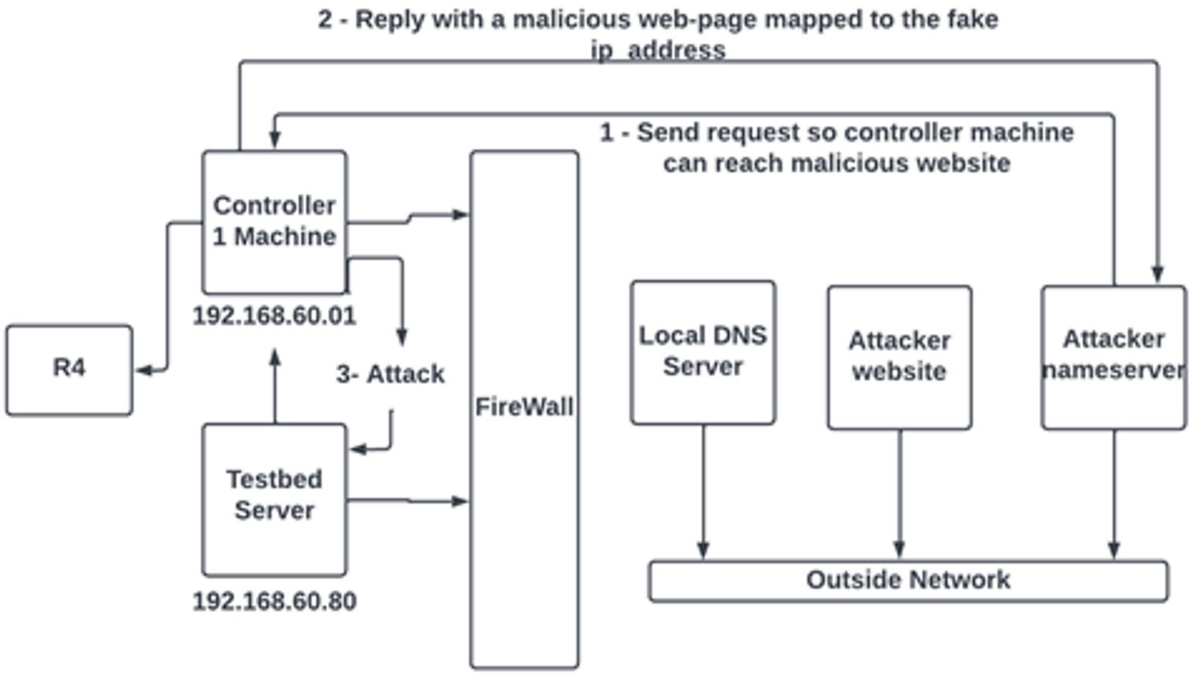


Figure 3.9: DNS rebinding attack on CMS.

3.2.1.6 Domain name system (DNS) rebinding attack

The domain name system is analogous to the phonebook for the internet. It maps the domain name to the IP address (Kevin 2019a). Since an IP address is unique for every network and device, they maintain unique records. So, fetching them is easy whenever they communicate with any web servers. In the testbed, the attacker can successfully gain access to the web server. The web server controls the sequence of operations within the testbed. These attacks cause maximum damage to the manufacturing systems as they overcome the countermeasure implemented to overcome the XSS attack.

This attack is demonstrated for the robotic arm in the testbed controlled by Arduino. The control commands are communicated to Arduino through Raspberry PI. In real manufacturing plants, these process is done via PLC and SCADA systems. The robotic arm is controlled via a controller machine. This controller machine interacts with the webserver to ensure that the robot is in the sequence required to complete the task. The testbed server is protected through firewalls and passwords. The firewall blocks communication to the testbed server from outside the local network. The password and same origin policy countermeasure within the webserver prevents the cross-site scripting attack. Only the controller machine is allowed to communicate to the outside network. The DNS rebinding attack is conducted in two steps as shown in Figure 3.9: i.) the first step requires attaching the password through the application programming interface (API) and ii.) the second step involves sending an HTTP request to the robotic arm servers command API.

After the employee opens the request from the attacker's website in a browser, it sends a DNS request to the attacker's resolver and retrieves the address of the malicious server, which is 192.168.60.1. Once it is uploaded to the employee browser, the malicious script in the attacker's website attempts to trigger another DNS resolution for its domain. However, the resolver will return 192.168.50.2 instead of 192.168.60.01. So, the attacker website is rebound to the controller machine 1 IP address. The malicious script keeps sending requests to the attacker's website that eventually reaches the private server. Since the testbed browser won't recognize these requests as cross-origin, the malicious website can read the data and impart change in the control of the robotic arm 4. Therefore, the attacker can change the sequence of movement of the robotic arm.

3.2.2 Ranking system

The six cyber-attacks selected for analysis provide an intuition on how these attacks can be conducted on the testbed. Demonstration of each attack on the testbed is infeasible. Hence, a metric system is developed to evaluate cyber-attacks. Three metrics are established for each attack: the sophistication of the attack, the ability of the attacks to bypass security measures, and if the attacks are successfully conducted, the economic loss to manufacturing plants.

First, the sophistication of cyber-attacks means the level of difficulty the attacks possess. These attacks have low, medium, and high levels of difficulty. If an attack has a low level of difficulty, it receives a score of 1. Examples of such attacks are phishing and SQL data injection attacks. If an attack spreads and impacts other processes a score of 3 is assigned. These attacks are on a medium level of difficulty. Examples of such attacks are cross-site scripting attacks, sniffing and spoofing attacks, and malware attacks. However, if attacks are a combination of different attacks, it is highly sophisticated and hence receive a score of 5. A DNS rebinding attack is a combination of sniffing and spoofing and a cross-site scripting attack. Thus it receives the highest score of 5.

Second, the attack's ability to bypass the security system demonstrates the skill of the attackers. The attacker receives a score of 1 for attacks that cannot bypass the prevention system. Examples of such attacks are phishing and SQL data injection attacks. However, if an attack is successful but is detected by the detection system a score of 3 is assigned to the attacker. These attacks include malware and cross-site scripting attacks. If an attack successfully bypasses the detection and prevention systems, it receives a score of 5. DNS rebinding and sniffing and spoofing attacks bypass the security system of the testbed, hence it receives a score of 5.

Third, the attack's ability to maximize the economic loss of the manufacturing plant demonstrates the severity of the attacks. For, manufacturing plant data is the lowest priority, hence intellectual property theft receives a score of 1. If an attack is successful in modification of the product for premature failure receives a score of 3. However, the maximum damage to any manufacturing facility is the loss of production hours. If an attack is successful in incurring the loss of production hours by disrupting the manufacturing process or damage to the machines, then it receives a score of 5.

Based on the metric and scale defined above, a severity score for each attack is calculated as shown in Table 3.3. SQL data injection attacks and phishing attacks receive low scores as they are blocked by the prevention system. However, if these attacks are successful due to users negligence, then their impact on the testbed is limited to intellectual property theft. Malware attacks are blocked by the prevention system too, but if the attacks are successful their impact on the testbed is high. Cross-site scripting attack before the countermeasures were implemented, was impactful. Therefore, these attacks had less impact on the testbed. Sniffing and Spoofing attacks and DNS rebinding attacks bypass the security systems. Both attacks successfully modify the sensor data and incur an economic loss to the manufacturing facilities. Hence these two attacks have high severity scores.

Attacks	**Attacker Goal**	**Metrics**			**Total**
		Difficulty of the attack	Detection and Prevention	Cost to the manufacturing systems	
Phishing	IP theft	1	3	1	5

Attacks	**Attacker Goal**	**Metrics**			**Total**
		Difficulty of the attack	Detection and Prevention	Cost to the manufacturing systems	
Phishing	Damage to product	1	3	1	5
	Damage manufacturing process	0	0	0	0
Severity Score					10
SQL	IP theft	1	1	1	3
	Damage to product	1	1	3	5
	Damage manufacturing process	0	0	0	0
Severity Score					8

Attacks	**Attacker Goal**	**Metrics**			**Total**
		Difficulty of the attack	Detection and Prevention	Cost to the manufacturing systems	
Malware	IP theft	3	1	1	5
	Damage to product	3	1	3	7
	Damage manufacturing process	3	1	5	9
Severity Score					21
XSS	IP theft	3	1	1	5
	Damage to product	3	1	3	7
	Damage manufacturing process	3	1	5	9

Attacks	**Attacker Goal**	**Metrics**			**Total**
		Difficulty of the attack	Detection and Prevention	Cost to the manufacturing systems	
Severity Score					21
Sniffing and Spoofing	IP theft	3	3	1	7
	Damage to product	3	3	3	9
	Damage manufacturing process	3	3	5	11
Severity Score					27
DNS	IP theft	5	5	1	11
	Damage to product	5	5	3	13

Attacks	Attacker Goal	Metrics			Total
		Difficulty of the attack	Detection and Prevention	Cost to the manufacturing systems	
DNS	Damage manufacturing process	5	5	5	15
Severity Score					39

Table 3.3: Severity score for the attacks.

3.3 Data auditing and detection layer

The function of this layer is to collect and monitor the process and operational level control data for the cyber manufacturing systems. A data collection plan includes a thorough analysis of sensors, the position of sensors in the manufacturing processes, and the frequency of data to be collected. An example of its implementation is using an accelerometer sensor to measure the change in vibration of the CNC milling process.

In the testbed established in the systems layer, data collection can be classified into four types: i.) cyber data, ii.) operational level data, iii.)

process level data, and iv.) plant level data. The cyber data includes the status of the job, resource allocation, network packet information, and task scheduling. This data is captured through application and network log files. The plant level data includes machine utilization rate, machine availability, and overall equipment effectiveness. These data can be captured by counting the number of defective products, the number of total production, the time study of the manufacturing process, etc., These are formula-based metrics used by manufacturing plants to monitor the efficiency of the plant. The operation and process level data is captured by attaching different sensors such as the current sensor, the temperature sensor, the position sensor, the camera sensor, the acoustic sensor, and the accelerometer sensor to the manufacturing processes.

Additionally, a detection algorithm is required to monitor and generate an alert for anomalous behavior. There exist many detection algorithms as explained in the introduction. These algorithms have demonstrated great accuracy in detecting and generating alerts. Thus, this work does not include a detection algorithm and assumes that there exists a detection algorithm that generates alerts to trigger the recovery layer.

3.4 Recovery layer

The proposed framework is illustrated in Figure 3.10. The operation of manufacturing systems consists of two modes: the default mode and the recovery mode. In the absence of an attack or when an attack remains undetected the manufacturing systems operate on the default controller.

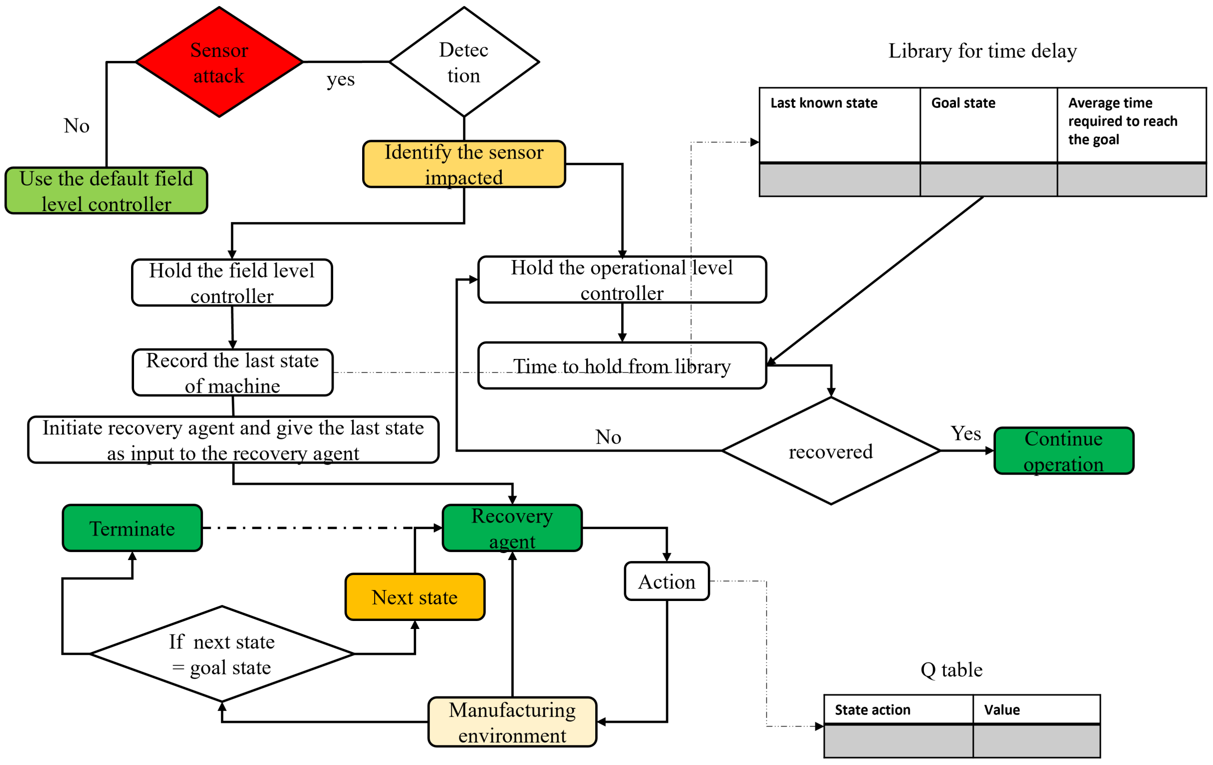

Sensor attack
Detec tion
yes
No
Use the default field level controller
Identify the sensor impacted
Library for time delay
Last known state
Goal state
Average time required to reach the goal
Hold the field level controller
Hold the operational level controller
Record the last state of machine
Time to hold from library
Initiate recovery agent and give the last state as input to the recovery agent
No
recovered
Yes
Continue operation
Terminate
Recovery agent
Next state
Action
If next state = goal state
Q table
State action
Value
Manufacturing environment

Figure 3.10: Recovery of CMS.

However, upon detecting an attack, the controller is switched to a recovery mode. Within the recovery mode, there are two key controller components: the process-level controller and a supervisor/operational-level controller.

Upon detecting a false data injection attack (also known as sniffing and spoofing attacks) on the sensor, the system generates an alert to point at the specific system and sensor under attack. The system then records the last state of the system and transmits this information to the recovery agent. This recovery agent is trained to navigate the system from its current position to a predefined goal position, aiming to restore the system's normal functioning. Meanwhile, the supervisory/operational controller operating on discrete events, delays the other process by the time proportional to the time required by the agent for system restoration. This approach allows the processes dependent on the impacted process to be in synchronization after the field-level recovery is achieved. Hence achieving an overall manufacturing systems recovery.

3.4.1 Challenges with the recovery layer

The proposed work faces several challenges and it is important to address them for the efficiency and reliability of the proposed recovery framework.

Incorrect last recorded state: A significant challenge occurs when the last recorded state of the cyber manufacturing systems is incorrect due to the time delay between the attack occurring and detection (Prasad, Swanson, and Y. Moon 2022). During this time the cyber manufacturing systems may move into an undesired state. Hence using this recorded state may lead the controller to an incorrect estimation. To overcome this challenge the existing literature implemented a checkpoint protocol Kong et al. 2018, recording the correct states and updating them with new correct states as the system moved forward in time. While this seems a reasonable approach when the final destination of the system is unknown. However, in the proposed work we can leverage the fact that the predefined operating conditions of the cyber manufacturing systems are known. As a result, an agent can be trained to drive the system to these predefined goal states, even if the starting state is incorrect.

Avoiding redundant agents: Ideally, each controller would require its own agent, but this approach could lead to the development and training of redundant agents. Since the agents are trained to navigate the manufacturing systems to predefined goals, a more efficient solution is to employ a singular agent for the impacted process. Meanwhile, the non-attacked controller can continue to be guided by the default controller, avoiding the need for multiple redundant agents.

Recording time delay for operation/supervisory controller: Due to the discrete events within the supervisory controller it is a challenge to know and add a time delay in real-time. Hence after training, this work maintains a library of the average time required by the agent to recover from a starting state to a final state. The library is stored offline and as the agent starts the recovery we add this time delay from the library to the respective process.

3.4.2 Reinforcement learning based recovery

3.4.2.1 Markov decision process

Developing a successful recovery strategy hinges on accurately estimating the future state based on the current state of the system, even when the current state is compromised due to a false data injection attack on the sensor. Despite the loss of information about the trajectory leading to the current state, the current state itself remains relevant for making predictions state Sutton and Barto 2018. This characteristic aligns with the Markov property, indicating that the recovery environment can be treated as a Markov decision process (MDP). MDP is a mathematical framework widely used to model problems with discrete time horizons, signifying the presence of both a starting state and a termination state. The Markov property suggests *that the future state of the system is conditionally independent of the past given the current state.* In the context of recovery from cyber attacks, the current state, though compromised, provides sufficient information for making predictions about subsequent states.

By formulating the recovery environment as an MDP, it becomes possible to apply MDP principles and methodologies to develop and optimize recovery strategies. This approach allows for a systematic and mathematically grounded exploration of decision-making processes, aiding in the design of effective recovery policies that take into account the dynamic nature of the system and the uncertainties introduced by the attacks. An MDP is defined by a five-state tuple (S, A, P, R, γ), where:

- S represents the set of states of the system s
- A is the set of control actions a
- P denotes the probability of transitioning from one state s to another state $s{+}1$
- r is the reward assigned when a state transition occurs
- γ is the discount factor, a value between 0 and 1. If γ is 0, only immediate rewards are considered, while if it's 1, rewards in future time steps are also taken into account.

$$r(s,a) = \mathbb{E}\left[R_{t+1} | S_t = s, A_t = a\right] \tag{3.1}$$

The primary objective of MDPs is to maximize expected rewards given by equation 3.1. The agent receives a reward R_{t+1} for the action chosen from a set of actions $A_t = a$ given a state $S_t = s$. The expected reward is given to the agent at the end of the episode. Two main approaches are commonly employed to solve this objective function: dynamic programming and reinforcement learning. Dynamic programming involves breaking the problem into simpler steps at different time points. Bellman's principle of optimality is a fundamental concept in dynamic programming Kirk 1970. It asserts that an *optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.* In essence, it separates the current decision made at time step t from future decisions $t + 1$. Dynamic programming leverages recursion for expected rewards to find the optimal solution. However, a key challenge lies in determining the state transition probabilities, which can be computationally expensive, even when the system model is known Sutton and Barto 2018. This challenge often leads to the exploration of alternative approaches, such as reinforcement learning, which can be more practical and effective in some scenarios.

3.4.2.2 Reinforcement learning

Reinforcement learning has experienced a notable surge as an alternative control framework, offering a sample-based approach to address the limitations of traditional model-based approaches Sutton and Barto 2018. What sets reinforcement learning apart from other machine learning algorithms is its distinctive methodology — it constructs its dataset through active exploration and exploitation of the environment. In this paradigm, an agent is trained to achieve a specific goal while receiving rewards for favorable actions and penalties for unfavorable ones. The agent's performance is solely assessed based on a scalar reward function, which quantifies the success or failure of its actions in the learning process. This unique characteristic of reinforcement learning makes it a powerful and adaptable tool, particularly suitable for scenarios where explicit models are challenging to define or compute. In the context of reinforcement learning the agent is represented as a 4-tuple (S, A, R, γ), where:

- S represents the set of states of the system s
- A is the set of control actions a
- R is the reward assigned when a state transition occurs
- γ is the discount factor, a value between 0 and 1. If γ is 0, only immediate rewards are considered, while if it's 1, rewards in future time steps are also taken into account.

The operation of reinforcement learning can be illustrated through a simplified example, as depicted in Figure 3.11. In this scenario, the agent, acting as the controller, aims to pick and place objects within a tray. At time t_0, the agent observes the state of the objects in the environment (S_t). Upon this observation, the agent selects an action from the set of control actions (A_t), which in this case involves adjusting the current or voltage to manipulate the robotic arm's motor. Following the execution

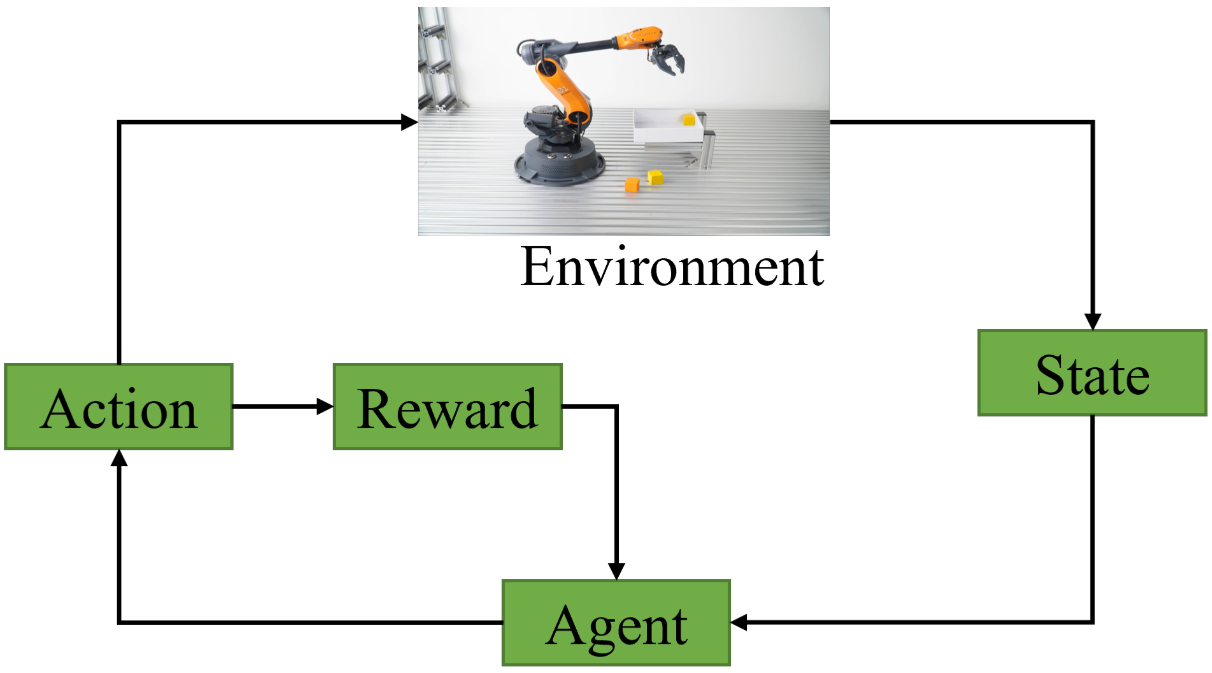


Figure 3.11: Toy example of reinforcement learning

of these actions, the agent receives a reward (R_{t+1}) determined during the modeling process, along with the next state of the system.

In each time step, the agent establishes a mapping from states to probabilities of selecting actions, known as the policy ($\pi_t(a|s)$). Reinforcement learning algorithms dictate how the agent adjusts its policy based on its experiences. Therefore, many reinforcement learning algorithms involve estimating the value function, as defined by equation 3.2, or the state-action value function, as given by equation 3.3. Estimating value functions enables the agent to understand the value of being in a particular state s following a policy π while estimating state-action value functions enables the agent to understand the value of taking a specific action a in state s following a policy π.

$$v_\pi(s) = \mathbb{E}_\pi\,[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s] \tag{3.2}$$

$$q_\pi(s,a) = \mathbb{E}_\pi\,[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a] \tag{3.3}$$

Thus, in the next chapter, this thesis demonstrates the functioning of the recovery architecture with emphasis on the recovery layer.

Chapter 4

Experimental validation

To assess the performance of recovery architecture the following chapter conducts multiple experiments. The experimental setup consists of real and virtual cyber manufacturing systems, setup, and training of reinforcement learning agents. The chapter ends with establishing key metrics to validate the performance of the recovery architecture.

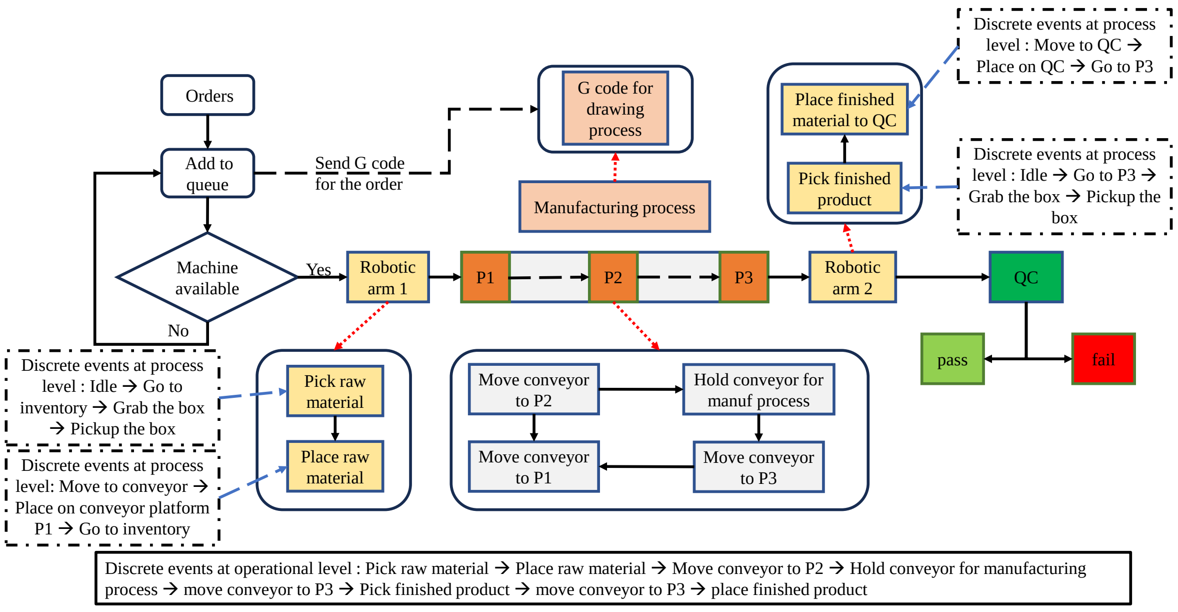

Orders
Add to queue
Send G code for the order
G code for drawing process
Manufacturing process
Place finished material to QC
Pick finished product
Discrete events at process level : Move to QC → Place on QC → Go to P3
Discrete events at process level : Idle → Go to P3 → Grab the box → Pickup the box
Machine available
Yes
No
Robotic arm 1
P1
P2
P3
Robotic arm 2
QC
pass
fail
Discrete events at process level : Idle → Go to inventory → Grab the box → Pickup the box
Discrete events at process level: Move to conveyor → Place on conveyor platform P1 → Go to inventory
Pick raw material
Place raw material
Move conveyor to P2
Hold conveyor for manuf process
Move conveyor to P1
Move conveyor to P3
Discrete events at operational level : Pick raw material → Place raw material → Move conveyor to P2 → Hold conveyor for manufacturing process → move conveyor to P3 → Pick finished product → move conveyor to P3 → place finished product

Figure 4.1: Experiment: Cyber manufacturing systems.

4.1 Systems layer

Cyber manufacturing systems are a complex and integrated framework that encompasses a range of processes, resources, and interactions, all working synchronously to produce the desired end product. From the testbed demonstrated in Figure 3.2 this work selects a portion of two robotic arms, a conveyor belt, and a drawing manufacturing process as depicted in Figure 4.1. These components operate in synchronization to manufacture products based on client specifications. The discrete events are simulated at the process level and operational/supervisory level.

In Figure 4.1 for the process level, there exists an independent controller designed to perform discrete process events. For example, the pick raw material discrete event at the operational level for robotic arm 1 has four process discrete events: Idle, go to inventory, grab the box, and pick up the box. The operational/supervisory has eight discrete events: pick raw material, place raw material, moving conveyor to P2, hold conveyor for the manufacturing process, move conveyor to P3, pick finished product, move conveyor to P3 and place finished product. This chapter uses programmable logic controllers to sequence the operational/supervisory discrete events. To mimic the operations of future manufacturing systems, client orders are submitted through a web-based application developed in the Python programming language. Upon receiving these orders, they are stored in a local database and sequenced according to the first-in, first-out algorithm. This modeling approach enables to emulate the behavior of future manufacturing systems.

In this simulation, all physical systems, except the drawing process, are modeled using the Python programming language, in conjunction with independent controllers and programmable logic controllers. The simulation is conducted over an 8-hour workday. Each step in the simulation represents a millisecond, signifying that every control decision made by the controller occurs at the millisecond level, ensuring precise control and monitoring of the manufacturing processes.

4.2 Attack identification layer

This work considers malicious attackers capable of executing false data injection attacks on sensors through sniffing and spoofing attacks. Specifically, this attacker can manipulate sensor data by introducing alteration before the data reaches the controller, thereby compromising the manufacturing systems's integrity. The attack designed in this paper can randomly select one sensor from the simulator and initiate a false data injection attack by adding or subtracting the sensor measurement before processed by the controller. The attacker operates with two primary objectives in mind:

Threat 1: Driving the system towards physical limits: One of the attacker's goals is to push the manufacturing systems to their physical limits. The attacker achieves this goal by adding positive values to the sensor measurement. This can result in increased stress and potential wear and tear on the system's components, potentially causing long-term damage or reducing the system's operational lifespan.

Threat 2: Forcing system shutdown: The attacker's second objective is to instigate a system shutdown. The attacker achieves this goal by adding negative values to the sensor measurement. The intent is to disrupt normal operations and potentially cause production downtime, leading to financial losses for the organization.

Both of these attacker objectives can have detrimental consequences for the manufacturing systems, ranging from reduced equipment lifespan and operational inefficiencies to costly shutdowns and potential safety hazards.

4.3 Data auditing and detection layer

The manufacturing control process parameter received by the conveyor is the distance and time to hold. These parameters are sent in the form of electrical signals to the motor controlling the conveyor belt. Assuming a constant voltage supply, there are three parameters to be collected: i.) current, ii.) position and iii.) time to hold the process. Gikfun ACS712 current sensors and HC-SR04 ultrasonic sensors collect current and position data. The time to hold the conveyor platform depends on the time required to finish the manufacturing process. These data are supplied to the detection algorithm. Moreover, the parameters received by the robotic arm 1 and 2 are position in degree and current. This information is captured by an encoder attached to the machine and Gikfun ACS712 current sensors.

The success of this work depends on a robust detection algorithm (Belikovetsky, Solewicz, et al. 2017; Prasad and Y. Moon 2021; Wu and Young B Moon 2019). However, because it is extensively studied in literature we do not study it in this thesis.

4.4 Recovery layer

The reinforcement learning framework serves as a powerful abstraction for addressing the challenge of goal-directed learning through interaction (Sutton and Barto 2018). In the context of this paper, the objective of the recovery agent is to navigate the cyber-attacked manufacturing systems back to their predefined conditions, making the reinforcement learning framework well-suited for this problem.

The development of the recovery agent, depicted in Figure 4.2 is an enlarged view of the recovery agent from Figure 3.10. It plays a crucial role in restoring the cyber-attacked manufacturing systems to their desired

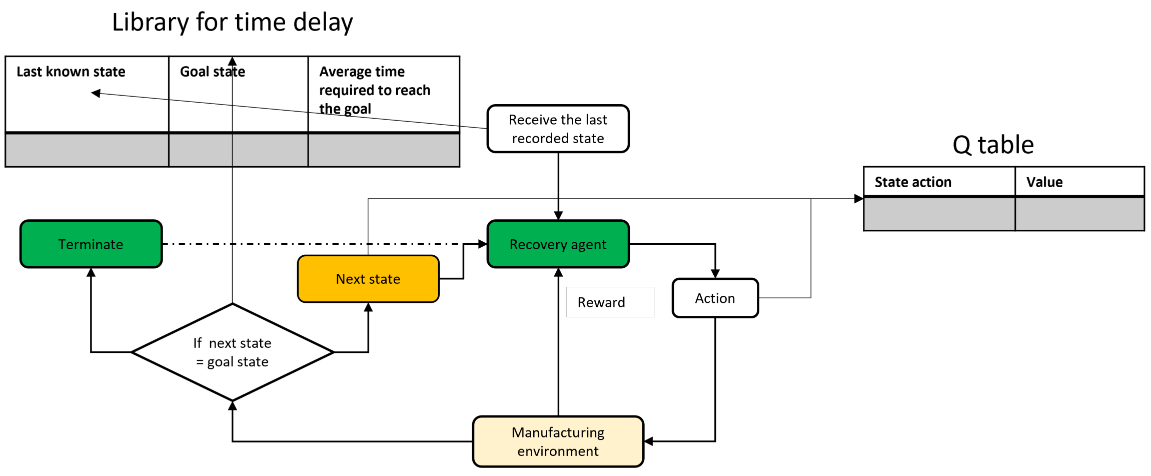

Library for time delay
Last known state
Goal state
Average time required to reach the goal
Receive the last recorded state
Q table
State action
Value
Terminate
Next state
Recovery agent
Reward
Action
If next state = goal state
Manufacturing environment

Figure 4.2: Reinforcement learning-based recovery agent (Enlarged view of the Figure 3.10).

state. At the heart of this process is the Q-learning algorithm, an integral part of reinforcement learning that enables the agent to learn a policy, mapping states to actions, through iterative exploration and exploitation. The agent initiates its recovery process from the last recorded state of the system, serving as its starting point. The primary objective of the recovery agent is to navigate the system back to its predefined operating conditions, overcoming the disruptions caused by cyber-attacks.

We implemented two different approaches for training reinforcement learning-based recovery agents. The game environment Unity was used to simulate conveyor movement within the factory and a manufacturing simulator was developed in Python that constitutes an operational level controller and process level controller. Each of these processes is explained in detail in the following section.

4.4.1 Game environment for training

Reinforcement learning has shown significant results in the gaming environment (Mnih et al. 2013; Silver et al. 2018; Lample and Chaplot 2017), and hence we explored different gaming environments that would allow us to simulate the cyber manufacturing systems. Reinforcement learning recovery agents have characteristics such as exploration and exploitation (Sutton and Barto 2018). Exploration allows the agent to explore the state space and exploitation allows the agent to use the actions that result in a maximum cumulative reward.

Raw material on platform	Manuf. process status	Finished object removal status	Current position of the agent	Sequences required by the agent to reach the goal
1	0	0	[P1, P2]	P2 - P3 - P1
1	1	0	[P2, P3]	P3 - P1
0	1	1	[P3, P1]	P1

Table 4.1: Agent's observation for the raw material on the platform, manufacturing process status, finished object removal status.

Here, we demonstrate the simulation of a conveyor platform where the agent's goal is to correct the position of the conveyor platform by maximizing the cumulative reward. Because of these characteristics, it is dangerous to train the agent on the physical system and hence a virtual recovery agent is developed inside a Unity3D platform.

The process of developing a recovery agent through this process is explained in Figure 4.3. The first step in this process is to identify variables of interest that the agent should observe from the environment. After iden-

Current position of the conveyor	Future position	Reward assigned	Simulation status
[P1, P2]	If it goes to P1	-1	Simulation ends
[P1, P2]	If it goes to P2	+0.3	Simulation continues
[P2, P3]	If it goes to P2	-1	Simulation ends
[P2, P3]	If it goes to P3	+0.3	Simulation continues
[P3, P1]	If it goes to P3	-1	Simulation ends
[P3, P1]	If it goes to P1	+0.4	Simulation continues

Table 4.2: Agent's observation for the raw material on the platform, manufacturing process status, finished object removal status.

tifying the variable, we develop a computer-aided design (CAD) model of the conveyor platform. Solidworks is used to create the CAD model and then export the assembly into a unified robotics description format (URDF) (Feder, Giusti, and Vidoni 2022). This format allows Unity3D to read the CAD model from Solidworks and use it for simulation (Akharas, Hennessey, and Tornoe 2020). Unity3D has a communicator that connects with Python to allow training of the recovery agent. Inside Unity3D, the recovery agent uses an MLAgents package to communicate with the conveyor platform. The MLAgents package requires to define the characteristics of the agent (Juliani et al. 2020).

The characteristics of the agent are defined as the following:

1. **Agent:** The agent is the conveyor platform. Inside MLAgent's

The current position of the recovery agent	Checkpoint is ON for following positions	End simulation is ON for following positions
[P1, P2]	P2 and P3	P1
[P2, P3]	P3	P2
[P3, P1]	P1	P3

Table 4.3: Checkpoint and end simulation.

Parameters	Values (t)
Trainer type	Proximal Policy Optimization (PPO)
Batch size	10
Learning rate	[0, 1]
Beta	5.0e-4
Epsilon	0.2
Lambda	0.99
Hidden units	128
Number of layers	2
Gamma	[0, 1]
Summary frequency	1000
Maximum steps	3000

Table 4.4: Training parameters for the recovery agent.

python package, this agent is responsible for generating observations and performing actions it receives from behavior.

2. **Behavior/Action:** The action in our experiment is to increase or decrease the current supplied to the platform. It is continuous data. In each step, the agent is given randomly one action to perform from these sets of possible actions.

3. **State/Observation:** The agent observation in the experiment is a tuple of:

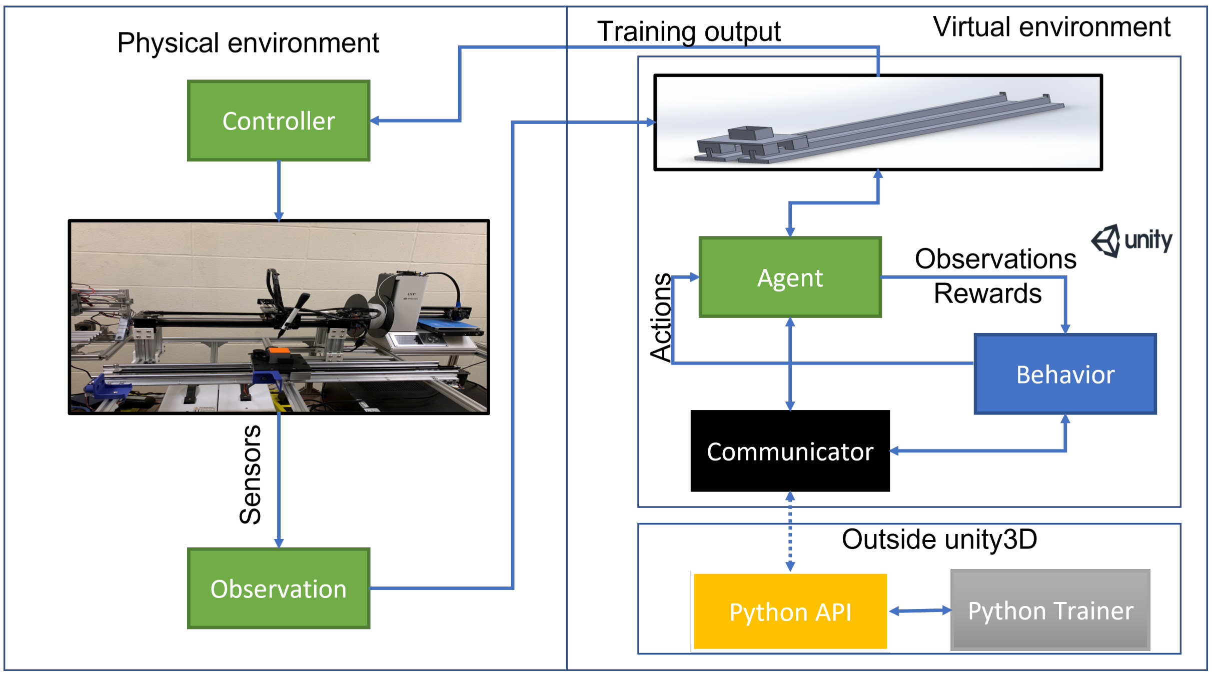


Figure 4.3: Virtual environment training for reinforcement learning based recovery agent inside Unity3D.

- Current position of the conveyor platform: This position is the starting position of the conveyor at each episode.
- Current supplied: This is the action taken in the last time step, and is appended to the state/observation because this assists in the agent's learning.
- Length of the conveyor: The total length of the conveyor. In our simulation, the total length of the conveyor is 1000 mm.
- Distance to goal: In our simulation, there are three goal positions P1, P2 and, P3. In our experiment, three new variables are created that generates the current status of the conveyor platform and future sequences that the agent must follow to reach the goal. These three variables are the raw material on the platform, manufacturing process status, and finished object

removal status.

- Raw material on the platform: This variable can take two values 0 and 1 as shown in Table 4.1. This is provided through the pressure sensor placed on our conveyor platform. This value is zero if there is no material on the platform and 1 if there is material on the platform.
- Manufacturing process status: This variable takes three values. The status can be 0, 1, or 2. Here, 0 indicates that the process is not started, 1 indicates that the process is currently going on and 2 indicates the process is finished. To simplify the model in this experiment only 0 and 1 are considered as shown in Table 4.1. Here 0 is for the process not started and 1 for the manufacturing process is finished.
- Finished object removal status: These variable takes two values as shown in Table 4.1. The status is 0 if the object is not removed and is still on the platform and 1 if the object is removed.

4. **Reward:** The agent's goal in our simulation is to maximize the reward earned by the agent. In any given situation the agent would randomly start in one of these three positions [P1, P2], [P2, P3], and [P3, P1]. Here [P1, P2] represents that the agent is between P1 and P2. Similarly [P2, P3] and [P3, P1] represents that the agent is in between P2 and P3 or P3 and P1. From the start positions and the actions performed by the agent, the agent would receive

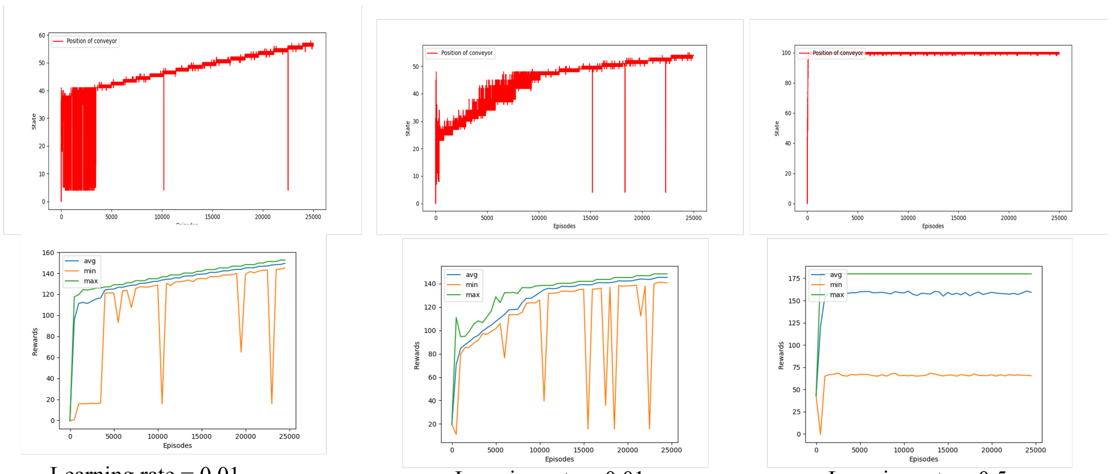
rewards/punishment based on the positions they end up in. These positions and rewards assigned are presented in Table 4.2.

5. **Environment:** The environment for the agent is the whole conveyor platform. Moreover, the challenge in this experiment was to assign rewards for the position and end the simulation if it goes to undesired positions. Hence two variables have been created that turn on the rewards and end the simulation if the conveyor reaches an undesired position. These two variables are shown in Table 4.3. For example, if the agent starts in the current position [P1, P2], the reward is ON for the position P2 and P3, and the agent is motivated to move to position P2 and then to position P3. However if the agent decides to move in the other direction i.e, P1 the agent is penalized by receiving a negative reward and the simulation end is turned ON for the position P1.

6. **Algorithm:** The reinforcement learning algorithm implemented is OpenAI's proximal policy optimization (PPO) (**PPO**). The choice of using PPO for the experiment was made due to its ease of implementation and tuning of hyperparameters. Reinforcement learning includes many dynamic parts compared to supervised learning and hence optimization via policy gradient is a challenge. To overcome this PPO is proposed by OpenAI that allows ease of implementation and tuning of hyperparameters. The training parameters used in this experiment are provided in Table 4.4.

State

Reward



Learning rate = 0.01
Discount rate = 0.01

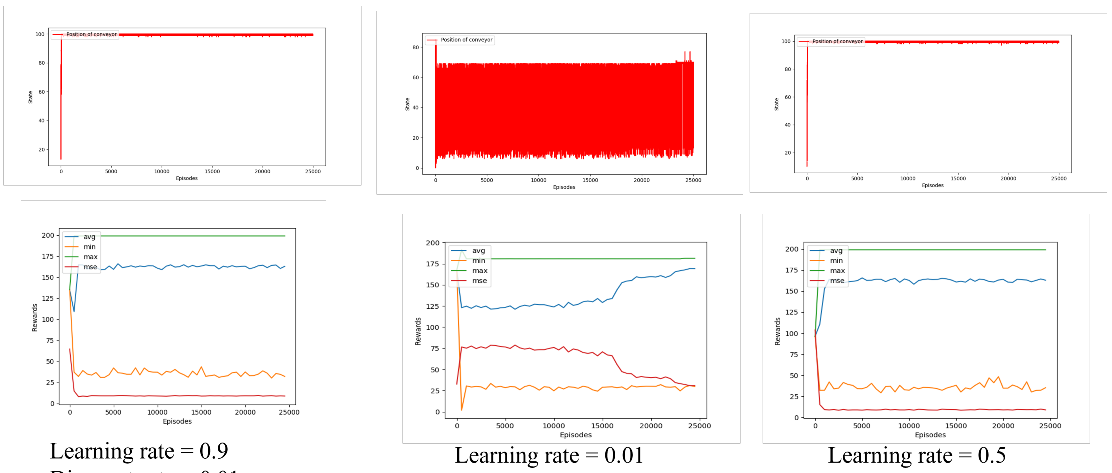
Learning rate = 0.01
Discount rate = 0.5

Learning rate = 0.5
Discount rate = 0.01

Figure 4.4: Training: Conveyor starting state between 10-15cm.

State

Reward



Learning rate = 0.9
Discount rate = 0.01

Learning rate = 0.01
Discount rate = 0.5

Learning rate = 0.5
Discount rate = 0.5

Figure 4.5: Training: Conveyor starting state between 10-75cm.

4.4.1.1 Result

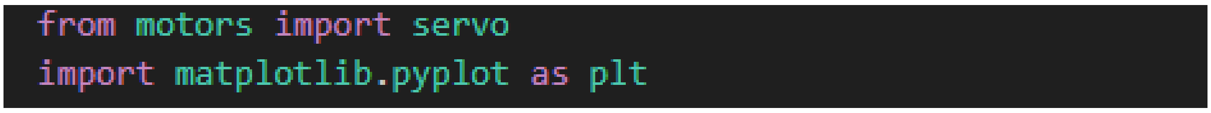
The goal of the recovery agent is to control the conveyor platform from the three possible positions [P1, P2], [P2, P3], and [P3, P1] and continue the sequence given in the Table 4.1. These conveyor platform positions are given to an agent as a random starting state at each episode. The random starting state represents an attacked conveyor platform position the agent would receive as input. Since there is no way to determine the attacked conveyor platform position before the attack, the randomness allows the agent to be trained for all possible attacked conveyor platform positions. The recovery agent with no prior knowledge about the conveyor platform starts with random actions, i.e., increasing the current or decreasing the current. This leads the conveyor to move in a forward direction or backward direction. One episode is one simulation, where the simulation ends if the agent moves the conveyor platform on one of the positions mentioned in Table 4.3.

Figures 4.4 and 4.5 illustrate the training of the agent from any initial state. Moreover, the fine-tuning of hyperparameters demonstrates a significant influence on the agent's training. Although the maximum reward attainable in a simulation is 200, it varies depending on the agent's starting state due to changes in rewards. The agent's loss decreases as expected, representing the gradient progress of the agent toward normal functioning. The optimal reward is achieved with a hyperparameter combination of a discount rate of 0.01 and a learning rate of 0.5, as depicted in Figure 4.5. Despite conventional literature suggesting a preference for a lower learning rate, our choice of 0.5 is justified by the agent's heavy penalty for moving towards an unsafe state. A higher learning rate enables the agent to rapidly learn the environment and map corresponding actions

4.4.1.2 Limitations

The game environment assists in agent training and allows the agent to train efficiently, hence reinforcement learning has shown amazing results within the game simulation environment. However, there are challenges in developing such an environment:

- The Manufacturing environment is very complex to be constructed within the game environment. To develop the conveyor platform in the game environment, this work had to decompose the real conveyor from the testbed and measure the length of the conveyor platform, the length of the belt, the motor implemented, and other physical characteristics of the conveyor.
- The Unity platform does not have an in-built operational level controller. To develop such a controller or synchronize the whole manufacturing systems multiple URDF files are required to be created in Solidworks and imported within the Unity environment.
- Communication among all the process level manufacturing controllers is required for a successful recovery, which is challenging to develop. It requires each process to be independently controlled by the reinforcement learning agent and hence multi-agent system is out of the scope of the work.



```
from motors import servo
import matplotlib.pyplot as plt
```

Figure 4.6: Code Snippet: Import library for R1.

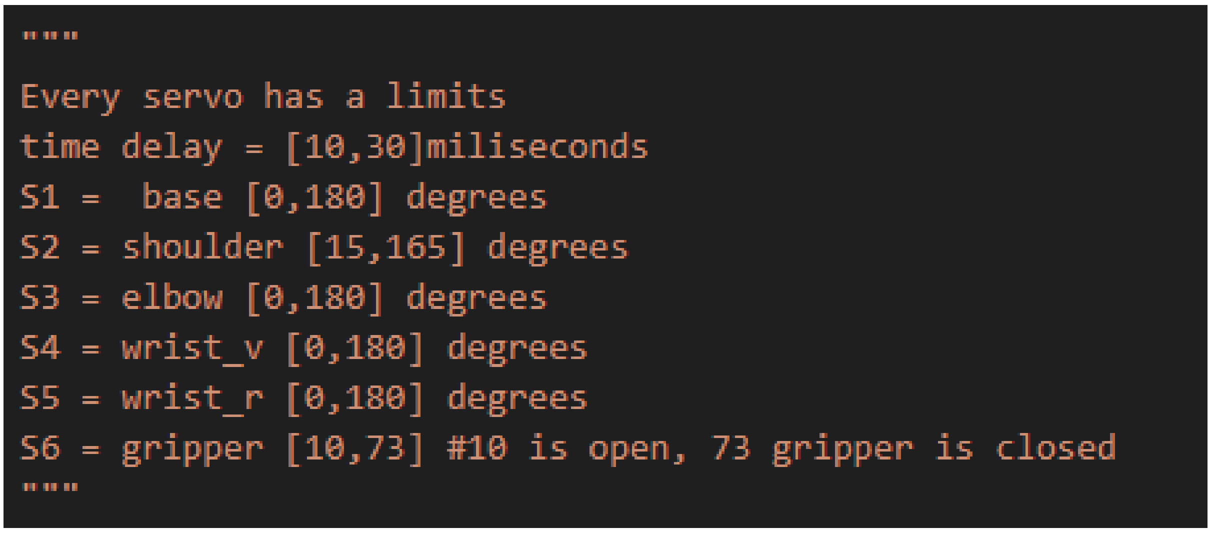
4.4.2 Cyber Manufacturing simulator in Python

To overcome the challenges mentioned in section 4.4.1.2 an alternate approach is used for simulating a cyber manufacturing systems. This approach uses Python to develop a cyber manufacturing systems simulator. The manufacturing simulator consists of two robotic arms, one conveyor, and a drawing manufacturing process. Our goal is to simulate the operational level controller and process level controller and hence we develop the simulator from scratch.

4.4.2.1 Simulation of robotic arm 1

. Robotic arm 1 has 6 servo motors and hence the first step in this simulation is to import the library servo motors. These servo motors are the actuators of the robotic arm. Additionally, to visualize the working of the motor we import the library from matplotlib as shown in Figure 4.6

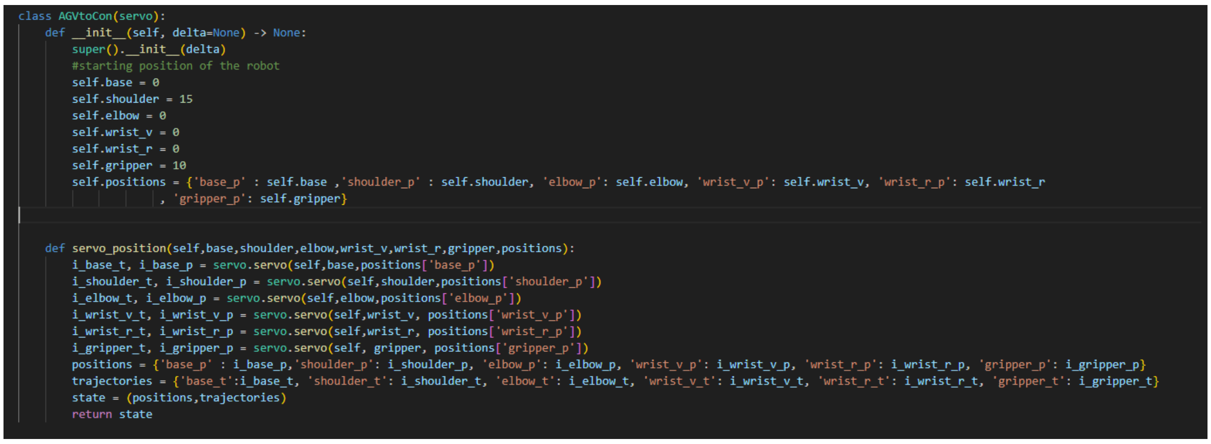
The next step is identifying the limitations of each motor and the time delay of the motors. To simulate this step we performed trial and error on the real testbed shown in Figure 4.1 and identified the limitations for each motor as shown in Figure 4.7. On average, we choose a time delay for each step taken by the servo motor as 20 milliseconds. This time delay is learned through experiments conducted on real systems. These time delays are part of the **servo** class.



```
"""
Every servo has a limits
time delay = [10,30]miliseconds
S1 =  base [0,180] degrees
S2 = shoulder [15,165] degrees
S3 = elbow [0,180] degrees
S4 = wrist_v [0,180] degrees
S5 = wrist_r [0,180] degrees
S6 = gripper [10,73] #10 is open, 73 gripper is closed
"""
```

Figure 4.7: Code Snippet: Motor constraints for R1.

We construct a class **AGVtoCon** shown in Figure 4.8 to develop the control algorithm that will move the robotic arm from an idle position to pick the raw material and place it on top of the conveyor platform. This class has an initialization and a servo position module. Within the initialization module, we define the idle state of each of the motors and create a dictionary to store the information. The second module is the servo position and this module is the controller of the robotic arm. This module input is the position of **base, shoulder, elbow, wrist-vertical, wrist-rotation** and **gripper's** along with the **dictionary position**. Within this module, we define the sequence in which the motor would align itself to perform the picking and placing operation. The module returns the trajectories traveled by the motor and the final position of the motors during each operation.



```python
class AGVtoCon(servo):
    def __init__(self, delta=None) -> None:
        super().__init__(delta)
        #starting position of the robot
        self.base = 0
        self.shoulder = 15
        self.elbow = 0
        self.wrist_v = 0
        self.wrist_r = 0
        self.gripper = 10
        self.positions = {'base_p' : self.base ,'shoulder_p' : self.shoulder, 'elbow_p': self.elbow, 'wrist_v_p': self.wrist_v, 'wrist_r_p': self.wrist_r
                        , 'gripper_p': self.gripper}


    def servo_position(self,base,shoulder,elbow,wrist_v,wrist_r,gripper,positions):
        i_base_t, i_base_p = servo.servo(self,base,positions['base_p'])
        i_shoulder_t, i_shoulder_p = servo.servo(self,shoulder,positions['shoulder_p'])
        i_elbow_t, i_elbow_p = servo.servo(self,elbow,positions['elbow_p'])
        i_wrist_v_t, i_wrist_v_p = servo.servo(self,wrist_v, positions['wrist_v_p'])
        i_wrist_r_t, i_wrist_r_p = servo.servo(self,wrist_r, positions['wrist_r_p'])
        i_gripper_t, i_gripper_p = servo.servo(self, gripper, positions['gripper_p'])
        positions = {'base_p' : i_base_p,'shoulder_p': i_shoulder_p, 'elbow_p': i_elbow_p, 'wrist_v_p': i_wrist_v_p, 'wrist_r_p': i_wrist_r_p, 'gripper_p': i_gripper_p}
        trajectories = {'base_t':i_base_t, 'shoulder_t': i_shoulder_t, 'elbow_t': i_elbow_t, 'wrist_v_t': i_wrist_v_t, 'wrist_r_t': i_wrist_r_t, 'gripper_t': i_gripper_t}
        state = (positions,trajectories)
        return state
```

Figure 4.8: Code Snippet: AGVtoCon class for R1.

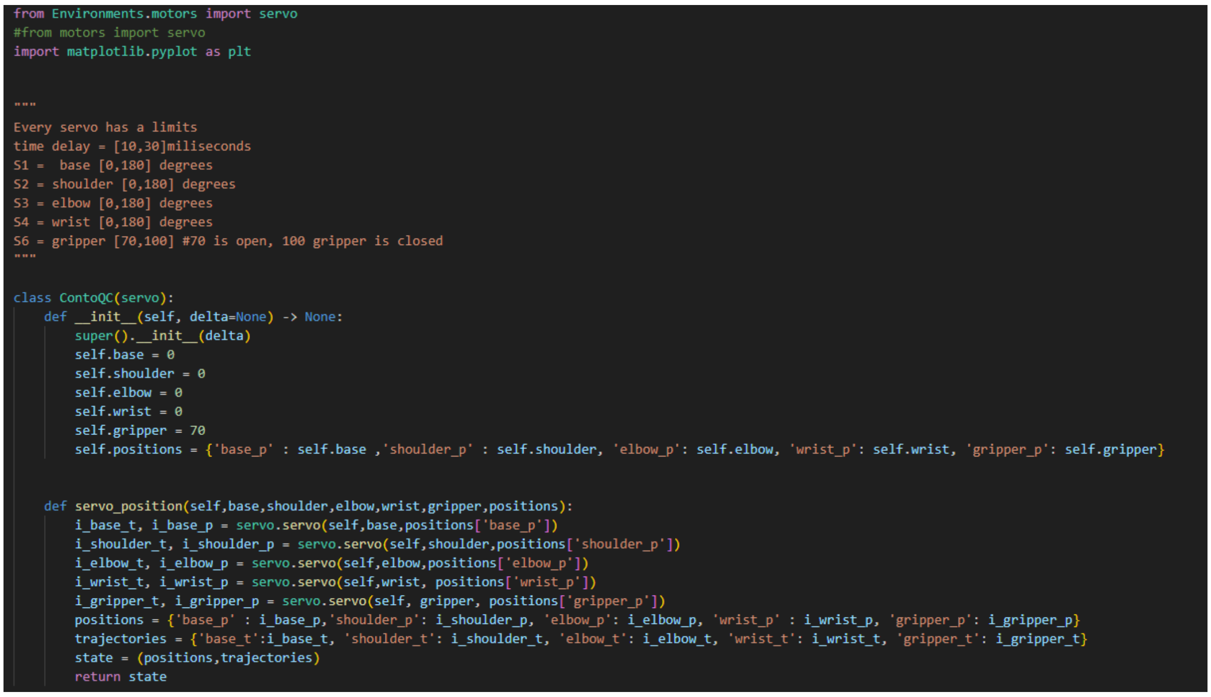
4.4.2.2 Simulation of robotic arm 2

To replicate the robotic arm 2 from the testbed we include only 5 servo motors. The development of code is similar to the one described in 4.4.2.1. The code is shown in Figure 4.9.

4.4.2.3 Simulation of conveyor

The conveyor simulator includes importing a stepper motor library because the testbed uses the stepper motor. The class of conveyor shown in Figure 4.10 includes the initialization and conveyor module. Within the initialization module, we define the steps of the stepper motor which is 200 according to our real testbed. The conveyor module mimics the behavior of the real system by either moving, stopping, or holding the platform. The modules take input as positions, the status of manufacturing, and the status of the robotic process, along with the time required to finish the robotic 2-picking process and time to hole the manufacturing process.

Figure 4.11 shows the movement of the conveyor platform. On the x-axis is the time and the y-axis is the distance travelled in mm. The



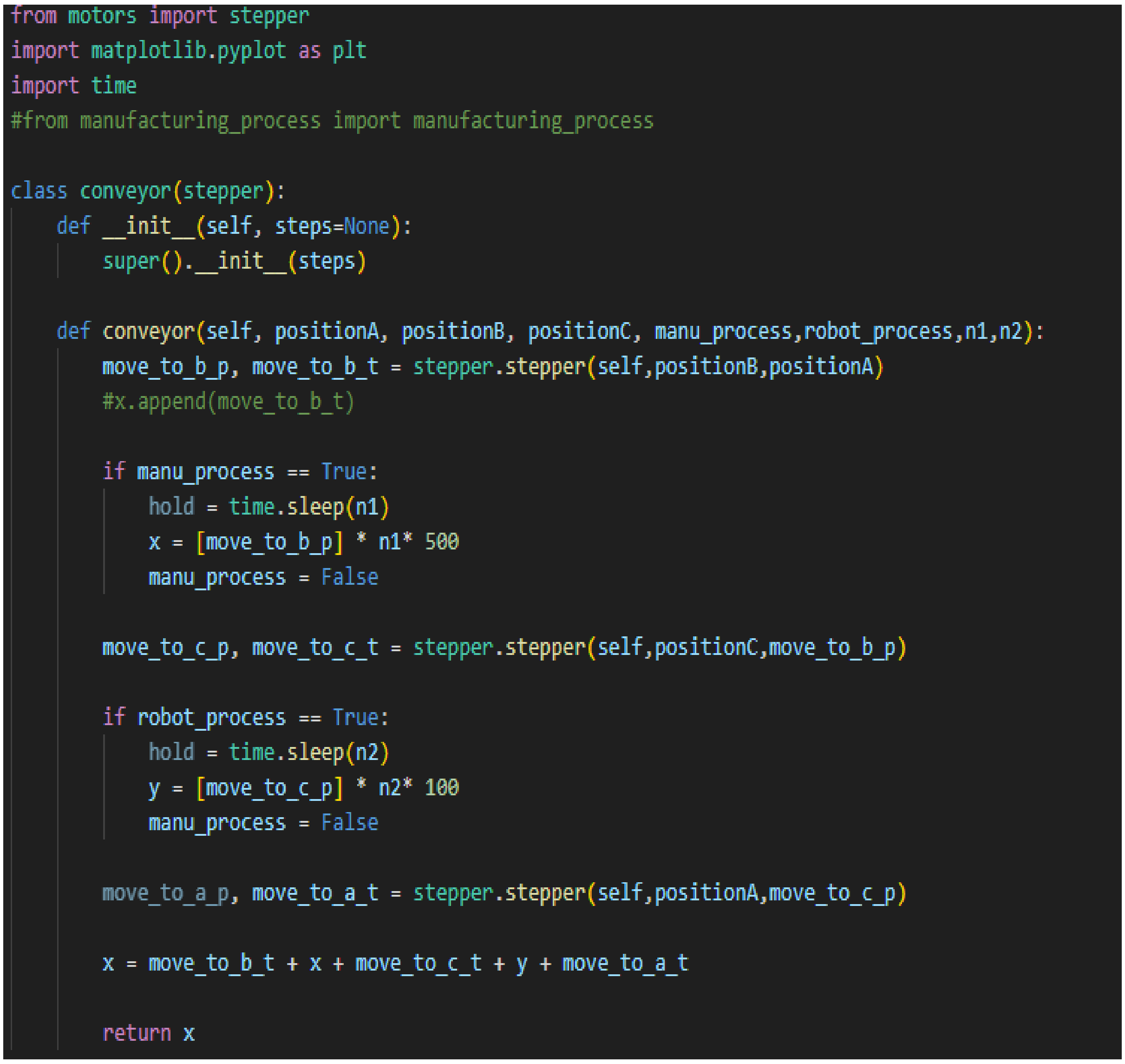
```python
from Environments.motors import servo
#from motors import servo
import matplotlib.pyplot as plt


"""
Every servo has a limits
time delay = [10,30]miliseconds
S1 =  base [0,180] degrees
S2 = shoulder [0,180] degrees
S3 = elbow [0,180] degrees
S4 = wrist [0,180] degrees
S6 = gripper [70,100] #70 is open, 100 gripper is closed
"""

class ContoQC(servo):
    def __init__(self, delta=None) -> None:
        super().__init__(delta)
        self.base = 0
        self.shoulder = 0
        self.elbow = 0
        self.wrist = 0
        self.gripper = 70
        self.positions = {'base_p' : self.base ,'shoulder_p' : self.shoulder, 'elbow_p': self.elbow, 'wrist_p': self.wrist, 'gripper_p': self.gripper}


    def servo_position(self,base,shoulder,elbow,wrist,gripper,positions):
        i_base_t, i_base_p = servo.servo(self,base,positions['base_p'])
        i_shoulder_t, i_shoulder_p = servo.servo(self,shoulder,positions['shoulder_p'])
        i_elbow_t, i_elbow_p = servo.servo(self,elbow,positions['elbow_p'])
        i_wrist_t, i_wrist_p = servo.servo(self,wrist, positions['wrist_p'])
        i_gripper_t, i_gripper_p = servo.servo(self, gripper, positions['gripper_p'])
        positions = {'base_p' : i_base_p,'shoulder_p': i_shoulder_p, 'elbow_p': i_elbow_p, 'wrist_p' : i_wrist_p, 'gripper_p': i_gripper_p}
        trajectories = {'base_t':i_base_t, 'shoulder_t': i_shoulder_t, 'elbow_t': i_elbow_t, 'wrist_t': i_wrist_t, 'gripper_t': i_gripper_t}
        state = (positions,trajectories)
        return state
```

Figure 4.9: Code Snippet: ContoQC class for R2.

conveyor system has four tasks and three positions as shown by the data collected from the ultrasonic sensor in Figure 4.11:

- moves to the manufacturing process after the raw material is placed on the platform [P1 - P2]
- holds the conveyor platform until the manufacturing process is completed [P2 and Hold],
- move the platform to the warehouse robotic arm [P2 - P3],
- return to the start position to receive the next raw material [P3 - P1]

The positions P1, P2, and P3 referred to in Figure 4.11 represent the first, second, and third positions of the conveyor platform. The trajectory tracked by virtual conveyor as shown in Figure 4.12 is similar to the real



```
from motors import stepper
import matplotlib.pyplot as plt
import time
#from manufacturing_process import manufacturing_process

class conveyor(stepper):
    def __init__(self, steps=None):
        super().__init__(steps)

    def conveyor(self, positionA, positionB, positionC, manu_process,robot_process,n1,n2):
        move_to_b_p, move_to_b_t = stepper.stepper(self,positionB,positionA)
        #x.append(move_to_b_t)

        if manu_process == True:
            hold = time.sleep(n1)
            x = [move_to_b_p] * n1* 500
            manu_process = False

        move_to_c_p, move_to_c_t = stepper.stepper(self,positionC,move_to_b_p)

        if robot_process == True:
            hold = time.sleep(n2)
            y = [move_to_c_p] * n2* 100
            manu_process = False

        move_to_a_p, move_to_a_t = stepper.stepper(self,positionA,move_to_c_p)

        x = move_to_b_t + x + move_to_c_t + y + move_to_a_t

        return x
```

Figure 4.10: Code Snippet: Conveyor.

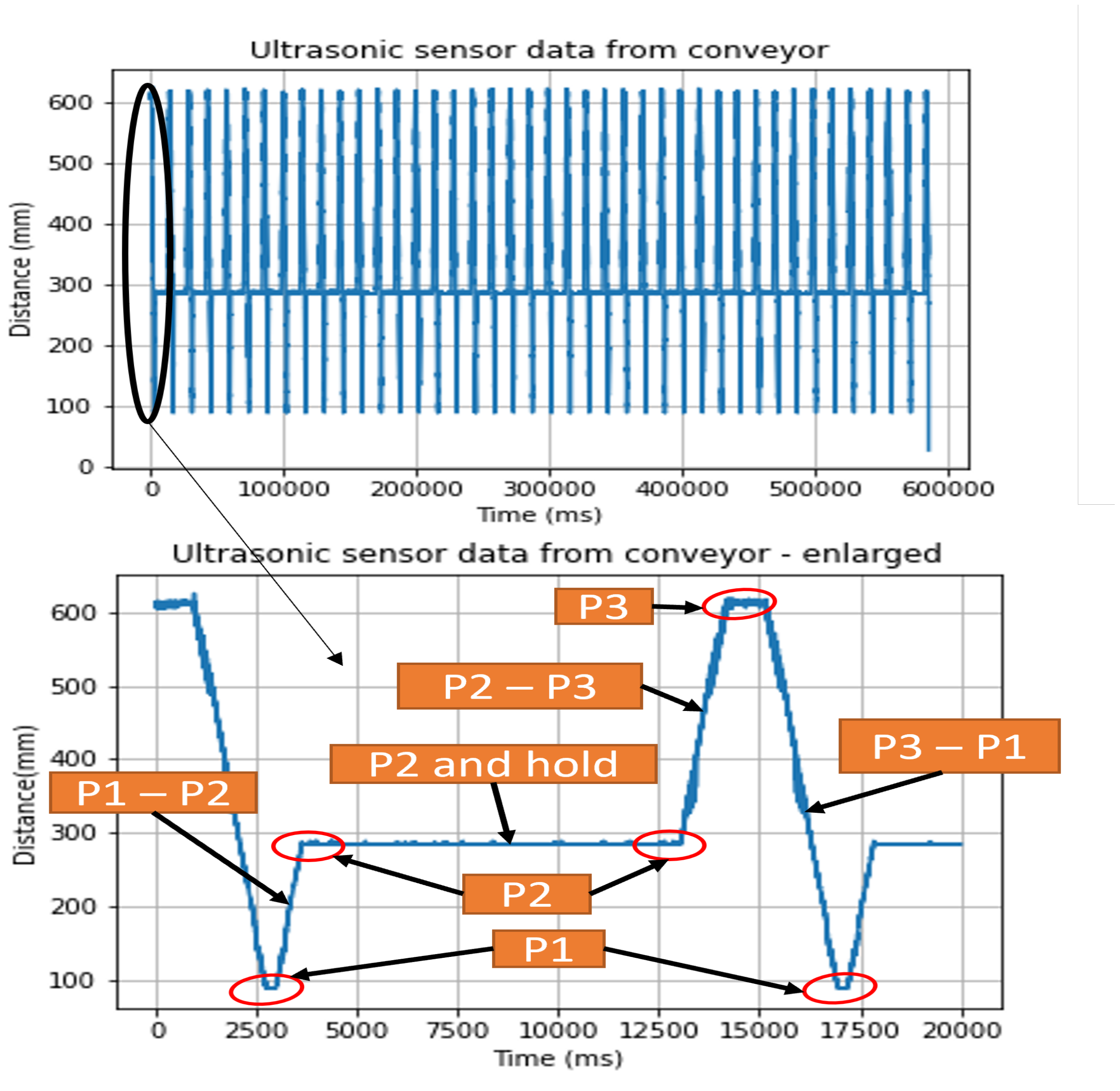


Figure 4.11: Data from the ultrasonic sensor.

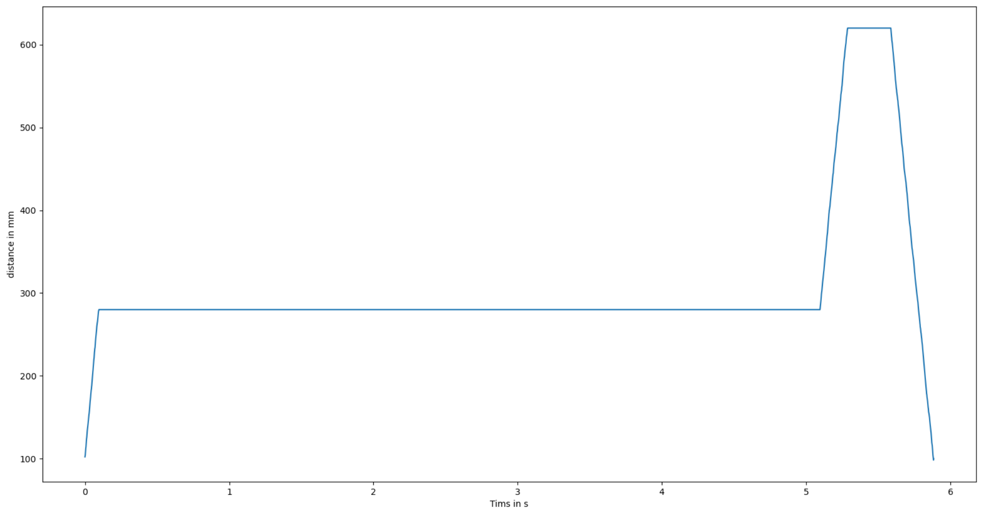
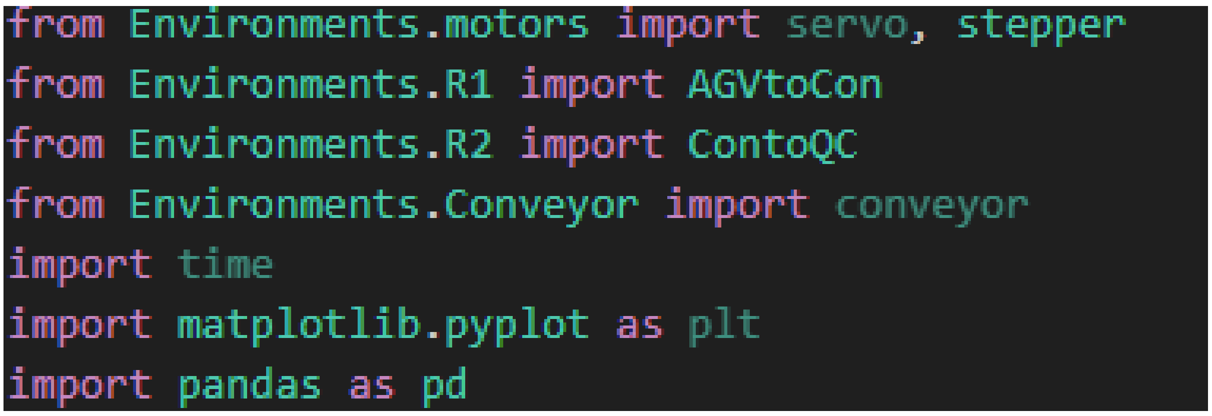


Figure 4.12: Virtual conveyor.



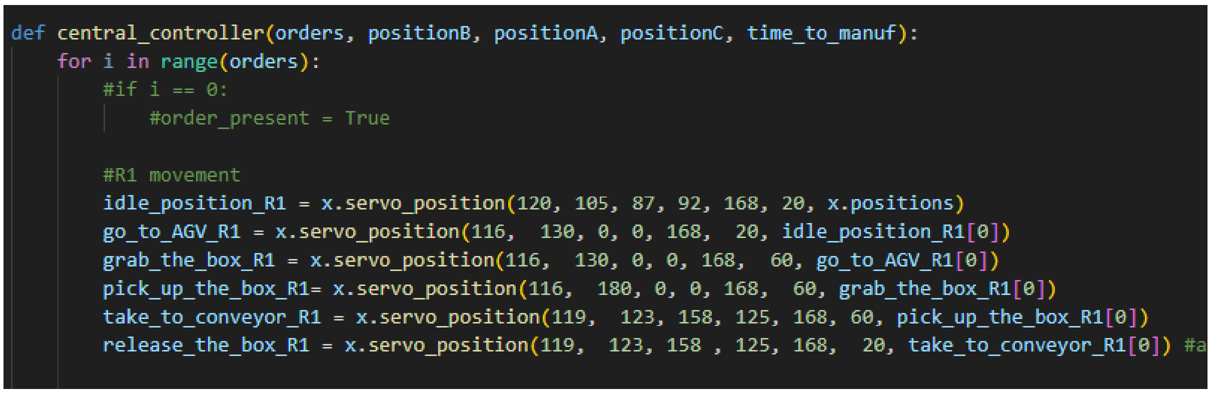
```
from Environments.motors import servo, stepper
from Environments.R1 import AGVtoCon
from Environments.R2 import ContoQC
from Environments.Conveyor import conveyor
import time
import matplotlib.pyplot as plt
import pandas as pd
```

Figure 4.13: Code Snippet: Library import in Central Controller.

conveyor. Hence this virtual environment can be used by the reinforcement learning agent to be trained.

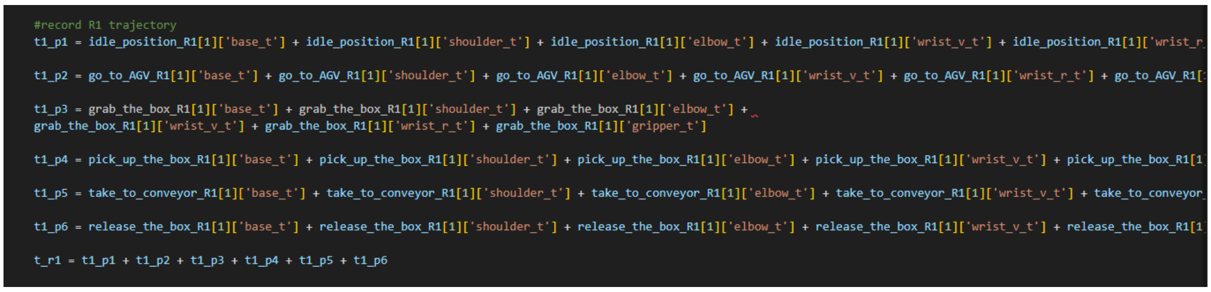
4.4.2.4 Simulation of operational level controller

While the above controller simulates the behavior of the process level controller it is paramount for the success of the recovery layer that it also controls the sequence of operation. Hence the simulation of operational controllers allows us the flexibility to develop such a simulator.



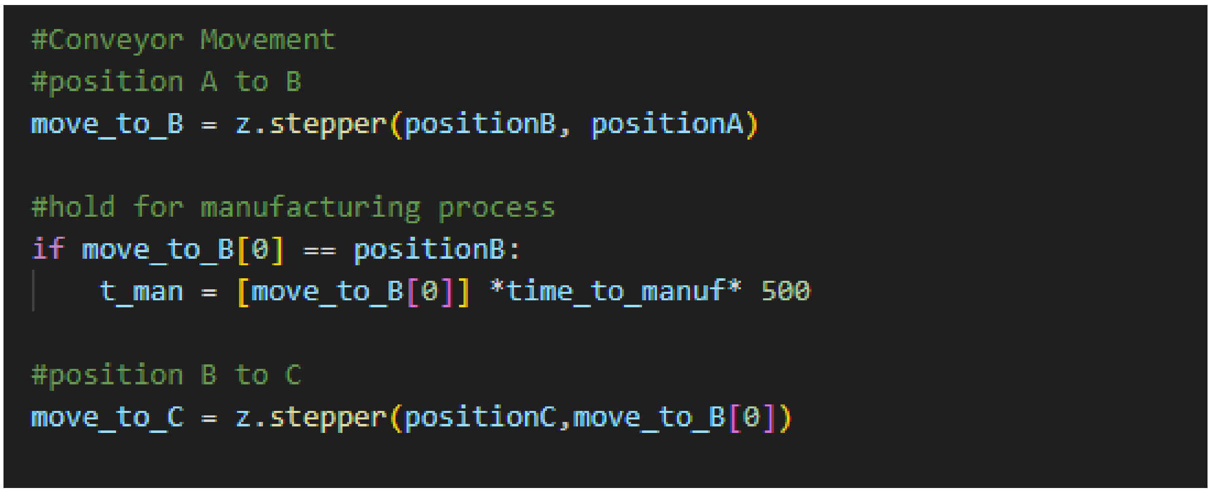
```
def central_controller(orders, positionB, positionA, positionC, time_to_manuf):
    for i in range(orders):
        #if i == 0:
            #order_present = True

        #R1 movement
        idle_position_R1 = x.servo_position(120, 105, 87, 92, 168, 20, x.positions)
        go_to_AGV_R1 = x.servo_position(116,  130, 0, 0, 168,  20, idle_position_R1[0])
        grab_the_box_R1 = x.servo_position(116,  130, 0, 0, 168,  60, go_to_AGV_R1[0])
        pick_up_the_box_R1= x.servo_position(116,  180, 0, 0, 168,  60, grab_the_box_R1[0])
        take_to_conveyor_R1 = x.servo_position(119,  123, 158, 125, 168, 60, pick_up_the_box_R1[0])
        release_the_box_R1 = x.servo_position(119,  123, 158 , 125, 168,  20, take_to_conveyor_R1[0]) #a
```

Figure 4.14: Code Snippet: Position of Robotic Arm defined within central controller.



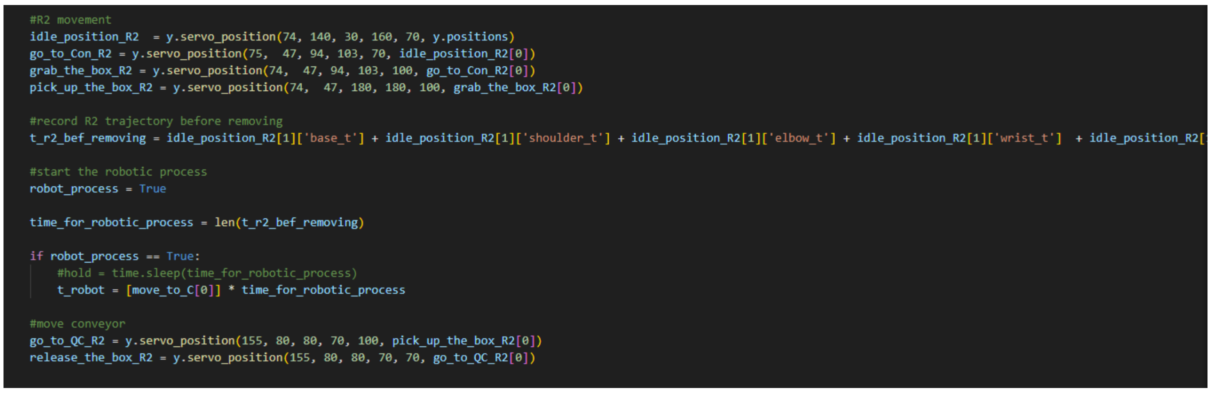
```
#record R1 trajectory
t1_p1 = idle_position_R1[1]['base_t'] + idle_position_R1[1]['shoulder_t'] + idle_position_R1[1]['elbow_t'] + idle_position_R1[1]['wrist_v_t'] + idle_position_R1[1]['wrist_r

t1_p2 = go_to_AGV_R1[1]['base_t'] + go_to_AGV_R1[1]['shoulder_t'] + go_to_AGV_R1[1]['elbow_t'] + go_to_AGV_R1[1]['wrist_v_t'] + go_to_AGV_R1[1]['wrist_r_t'] + go_to_AGV_R1[

t1_p3 = grab_the_box_R1[1]['base_t'] + grab_the_box_R1[1]['shoulder_t'] + grab_the_box_R1[1]['elbow_t'] +
grab_the_box_R1[1]['wrist_v_t'] + grab_the_box_R1[1]['wrist_r_t'] + grab_the_box_R1[1]['gripper_t']

t1_p4 = pick_up_the_box_R1[1]['base_t'] + pick_up_the_box_R1[1]['shoulder_t'] + pick_up_the_box_R1[1]['elbow_t'] + pick_up_the_box_R1[1]['wrist_v_t'] + pick_up_the_box_R1[1

t1_p5 = take_to_conveyor_R1[1]['base_t'] + take_to_conveyor_R1[1]['shoulder_t'] + take_to_conveyor_R1[1]['elbow_t'] + take_to_conveyor_R1[1]['wrist_v_t'] + take_to_conveyor_

t1_p6 = release_the_box_R1[1]['base_t'] + release_the_box_R1[1]['shoulder_t'] + release_the_box_R1[1]['elbow_t'] + release_the_box_R1[1]['wrist_v_t'] + release_the_box_R1[1

t_r1 = t1_p1 + t1_p2 + t1_p3 + t1_p4 + t1_p5 + t1_p6
```

Figure 4.15: Code Snippet: Trajectories and position captured of the robotic arm 1.



```
#Conveyor Movement
#position A to B
move_to_B = z.stepper(positionB, positionA)

#hold for manufacturing process
if move_to_B[0] == positionB:
    t_man = [move_to_B[0]] *time_to_manuf* 500

#position B to C
move_to_C = z.stepper(positionC,move_to_B[0])
```

Figure 4.16: Code Snippet: Positions for conveyor platform.



```
#R2 movement
idle_position_R2  = y.servo_position(74, 140, 30, 160, 70, y.positions)
go_to_Con_R2 = y.servo_position(75,  47, 94, 103, 70, idle_position_R2[0])
grab_the_box_R2 = y.servo_position(74,  47, 94, 103, 100, go_to_Con_R2[0])
pick_up_the_box_R2 = y.servo_position(74,  47, 180, 180, 100, grab_the_box_R2[0])

#record R2 trajectory before removing
t_r2_bef_removing = idle_position_R2[1]['base_t'] + idle_position_R2[1]['shoulder_t'] + idle_position_R2[1]['elbow_t'] + idle_position_R2[1]['wrist_t']  + idle_position_R2[

#start the robotic process
robot_process = True

time_for_robotic_process = len(t_r2_bef_removing)

if robot_process == True:
    #hold = time.sleep(time_for_robotic_process)
    t_robot = [move_to_C[0]] * time_for_robotic_process

#move conveyor
go_to_QC_R2 = y.servo_position(155, 80, 80, 70, 100, pick_up_the_box_R2[0])
release_the_box_R2 = y.servo_position(155, 80, 80, 70, 70, go_to_QC_R2[0])
```

Figure 4.17: Code Snippet: Positions and trajectories capturing for robotic arm R2.

To develop this simulator we begin by importing the library of stepper motor, servo motor, robotic arm 1, robotic arm 2, and conveyor simulator developed in sections 4.4.2.1, 4.4.2.2, 4.4.2.3 as shown in Figure 4.13. We defined a module **central-controller** shown in Figure 4.14 that performs the following operation:

- **R1 pickup and drop object:** In the real testbed we know the position of the object and hence we define a variable that takes this position and assigns it as an input to the R1 controller. This is demonstrated in Figure 4.14, 4.15.

- **Conveyor moves to P2 position:** Upon receiving the object on the platform we send a binary signal, similar to led glowing in the testbed. These indicates that the object is placed on the platform and the conveyor can move to the position P2. Since in the testbed through ultrasonic sensor we know the position, we hard code this value in the simulation as shown in Figure 4.16.

- **Manufacturing process**. Developing a simulator of a drawing process is difficult also it is not important for this work hence we get to know the time for processing from the G-code software and use that as input in the testbed and also in the simulator.

- **Conveyor moves to P3 position:** Upon finishing the manufacturing process the conveyor is moved to P3. Through the ultrasonic sensor, we get the position and hardcode this number in our simulator. This is demonstrated in Figure 4.17.
- **R2 pickup and drop finished product:** The robotic arm 2 is given a binary signal to represent the platform reaching P3 and this initiates the robotic arm. Like R1 we are aware of these numbers through the real testbed. The central controller tells the robotic arm 2 to pick the product from P3 and drop it in QC.
- **Conveyor moves to P1 position:** Upon R2 removing the product from the platform the platform moves to P1. This is demonstrated in Figure 4.17.

4.4.2.5 Simulation of recovery agent

In this section, we discuss the characteristics of developing an environment for reinforcement learning agents. The crucial elements of observation, rewards, and actions are defined for the above-defined simulator as follows:

Environment: The manufacturing systems simulation consists of two robotic arms, one conveyor and a manufacturing process. This constitutes the agent's environment. Additionally, the sensors providing information on positions, current, and the status of discrete events within the plant are integral components of the environment. Table 4.5 shows the detailed environment the agent observes. In the Table, the discrete events are represented as binary information.

Machines	Observations
Robotic arm 1	$\{P_1, \dots P_6\}$ in degree $\{I_1, \dots I_6\}$ in mA
Robotic arm 2	$\{P_1, \dots P_5\}$ in degree $\{I_1, \dots I_5\}$ in mA
Conveyor belt	$\{P_1, P_2, P_3\}$ in mm $\{I_1\}$ in mA
Manufacturing process	Time to process in s

Discrete events	R1 pickup object: {1,0}
	R1 drop object: {1,0}
	Conv moves to P2 position: {1,0}
	Manufacturing process: {1,0}
	Conv moves to P3 position: {1,0}
	R2 pickup finished product: {1,0}
	Conv moves to P1 position: {1,0}
	R2 drop finished product for QC: {1,0}

Table 4.5: Agent's environment.

State/Observation Space: Considering the complexity of the environment, it is essential to provide the agent with a subset of information as its state/observation space. In this paper, the agent observes the information only for the sensor detected by the detection system. Additionally, the agent receives information about the discrete event to which the detection belongs. For example, if the *conveyor moves to P2 position* is currently in motion then after detection the agent will receive: [(Discrete events: [0,0,1,0,0,0,0,0]),(current state of the conveyor: P_1').[1],(Goal state: P_2)].

[1]In Table 4.5 P_1 is the true state, however since after the attack the state is not true and hence represented as P_1'.

Action Space: The action space is straightforward, encompassing discrete control steps such as increasing the current, decreasing the current, or maintaining it at a constant level. In the environment, this is represented as 0,1 and 2 numbers. However inside the controller simulator, number 0 corresponds to decreasing the current, 1 corresponds to constant, and 2 corresponds to increasing the current.

Reward: Given the goal-directed nature of the learning, any state other than the goal state is deemed incorrect. Therefore, the agent receives a reward only when it reaches the goal state; otherwise, it incurs a negative reward in all other states. The reward function is defined in equation 4.1, where T_{goal} is the predefined condition and T_{act} is the actual status of the agent. The agent is given a positive five reward as depicted in 4.2 if the difference between the current state of the system and the goal state of the system is within the bounds and negative elsewhere. This approach ensures that the agent is penalized for each action taken outside the goal state, aligning with the overarching goal of returning the manufacturing systems to its predefined conditions.

$$r = T_{goal} - T_{act} \tag{4.1}$$

where;

$$\begin{cases} r = 5 & -2 \leq r \leq 2 \\ r = - \mid r \mid & else \end{cases} \tag{4.2}$$

Algorithm: The Q-learning algorithm is employed iteratively until the agent successfully reaches its goal (Dietterich 2000). Concurrently, a

Q Table is constructed to store the learned values that guide the agent's decision-making process. Simultaneously, a library is developed to track the average time required by the agent to reach its goal during the learning process. This library is utilized by the supervisory/operational level recovery. During the training phase, the starting state of the system is randomly selected from the predefined physical limits associated with each actuator. This randomness in the selection process ensures that the agent encounters a diverse set of starting conditions, mirroring the potential scenarios it might face in real-world applications. This approach helps the agent generalize its learning, making it robust across various initial states. It is a model-free algorithm and learns the value of an action in a particular state by using equation 4.3. In this equation Q(s, a) represents the Q-value for state-action pair (s, a), α is the learning rate, r is the immediate reward, γ is the discount factor, s' is the next state, and a' is the action in the next state. The Q Table is updated as the agent explores and exploits the environment.

$$Q(s,a) = (1-\alpha) * Q(s,a) + \alpha * [r + \gamma * max(Q(s',a'))] \tag{4.3}$$

This work uses the OpenAI's gym environment (Brockman et al. 2016) to access the action space and observation space. The recovery agent includes four modules as shown in Figure 4.18:

- **Initialization:** In this module the observation space and action space are defined.

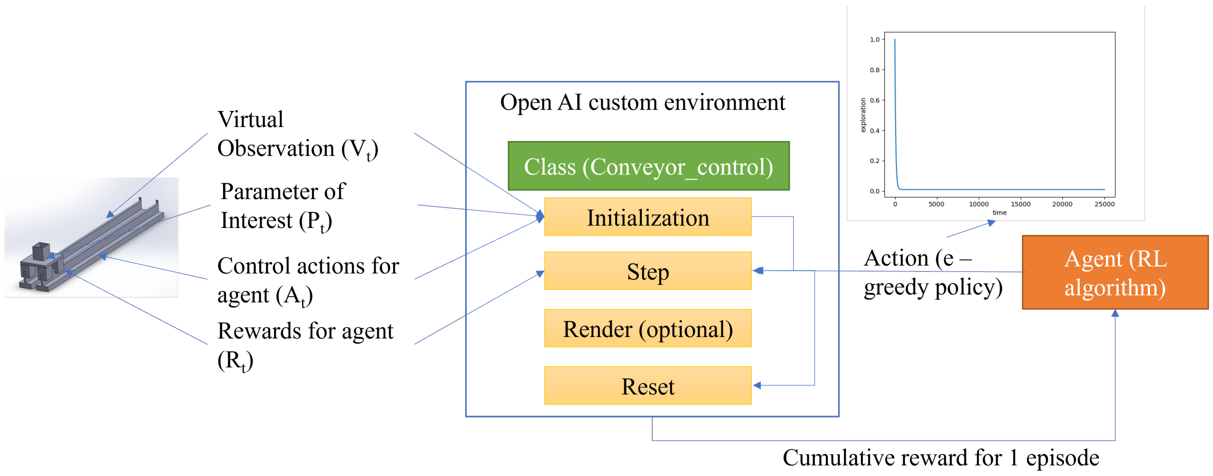

Virtual Observation (V_t)
Parameter of Interest (P_t)
Control actions for agent (A_t)
Rewards for agent (R_t)
Open AI custom environment
Class (Conveyor_control)
Initialization
Step
Render (optional)
Reset
exploration
time
Action (e – greedy policy)
Agent (RL algorithm)
Cumulative reward for 1 episode

Figure 4.18: Open AI environment.

- **Step:** In this module the simulator defined in section 4.4.2.1, 4.4.2.2, 4.4.2.3 , 4.4.2.4 are communicated with agent. This forms the environment where rewards are defined for the action taken by the agent.

- **Render:** This work does not use rendering.

- **Reset:** The reset function allows the functioning of termination and truncation. We implement constraints in the form of motor limits and time limits. If crossed the environment is terminated.

4.5 Results

The Q learning algorithm has two main hyperparameters, discount rate and learning rate. Hence our work tested different combinations of these hyperparameters to identify the combination that would work well for the testing of the agent. Figure 4.19 demonstrates the average reward earned by the agent for two different starting conditions: closer to shutting

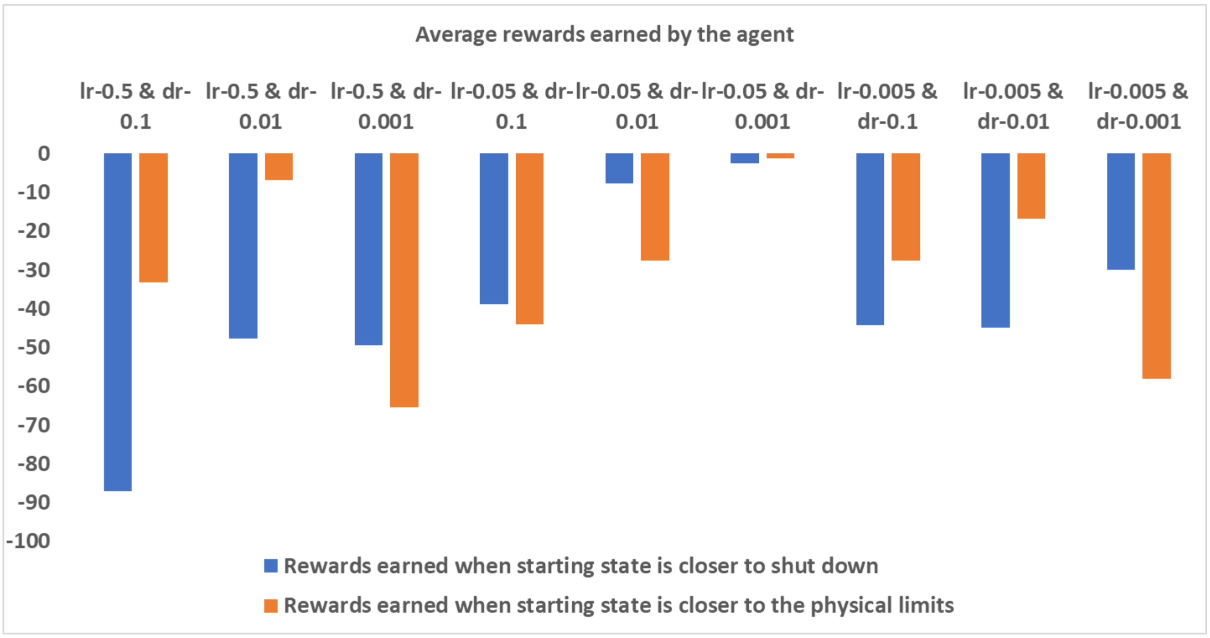


Figure 4.19: Average reward earned by the agent.

down the plant and closer to the physical limitations of the motor. The combination of a learning rate of 0.05 and a discount rate of 0.001 achieves the best result. Since the agent had to control the systems within 200 time steps, an observation is that a lower learning rate would flourish only if the training time is increased.

To assess the performance of the recovery strategies outlined in this paper, comparisons are made against two reference benchmarks: (i) restarting the process and (ii) a manually tuned PID (Proportional-Integral-Derivative) controller. This comparative analysis will provide valuable insights into the effectiveness of the proposed recovery strategies, enabling manufacturers to make informed decisions about their adoption and implementation.

4.5.1 Recovery from threat 1

Threat 1 involves false data injection attacks on sensors, pushing the manufacturing systems towards the physical limits of the actuator. In Figure 4.20 and 4.21, the recovery strategies of the recovery agent and PID controller are demonstrated as they restore the system states to normal conditions. The PID controller's approach comprises a two-step recovery process. Initially, it brings the system back to a checkpoint state, representing the last known safe state, before resuming normal operation. In contrast, the reinforcement learning agent, trained to navigate directly from the starting state to normal operation, executes a more direct recovery path. As a result, the agent exhibits greater speed compared to the manual PID controller. The implementation of a time delay library, as evident in Figure 4.22 and 4.23, serves a crucial role in pausing discrete events. This ensures seamless movement and operational continuity, even in the face of disruptions caused by false data injection attacks on sensors.

4.5.2 Recovery from threat 2

Threat 2 represents false data injection attacks on sensors intended to move the manufacturing systems toward shutting down the plant by driving the sensor values toward zero. In Figure 4.24, the recovery strategies of the recovery agent and PID controller are depicted as they bring the system states back to normal conditions. Once again, the time delay library, illustrated in Figure 4.25, proves instrumental in pausing discrete events, ensuring a smooth transition despite the false data injection attack on the

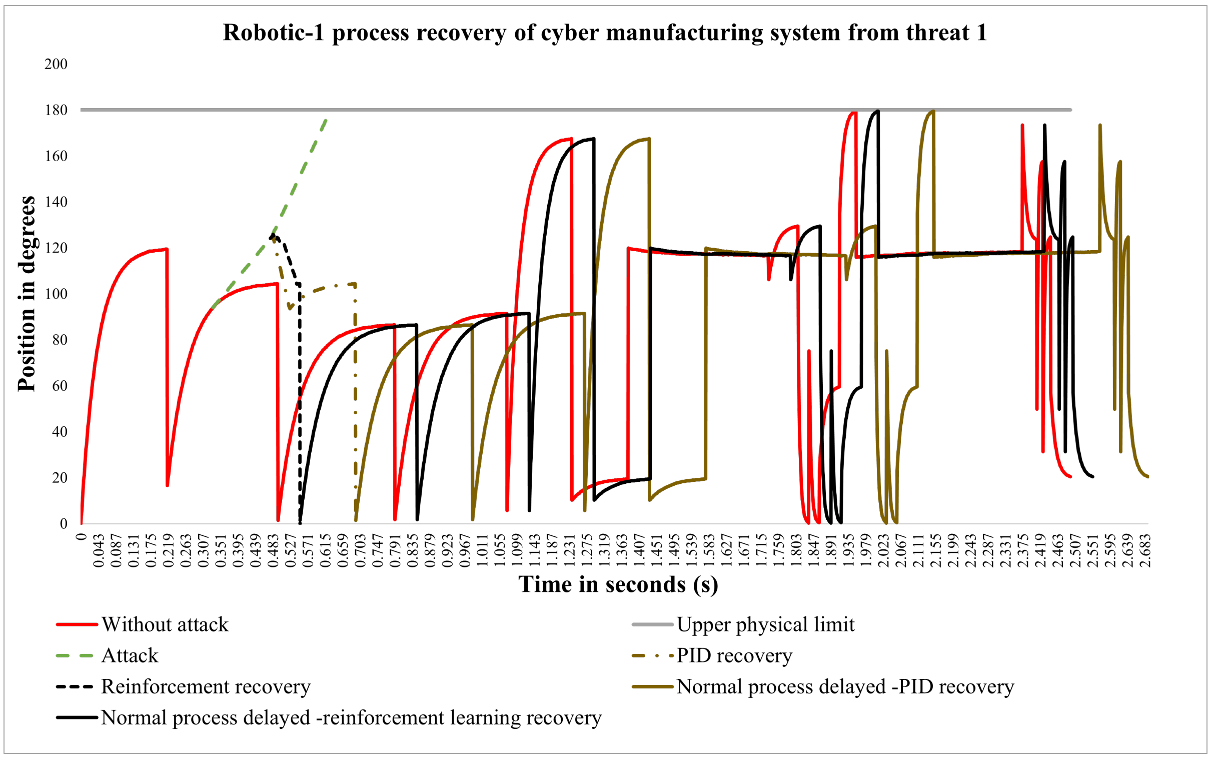


Figure 4.20: Process level recovery of robotic arm 1 from threat 1 with no recovery, reinforcement learning recovery, PID recovery.

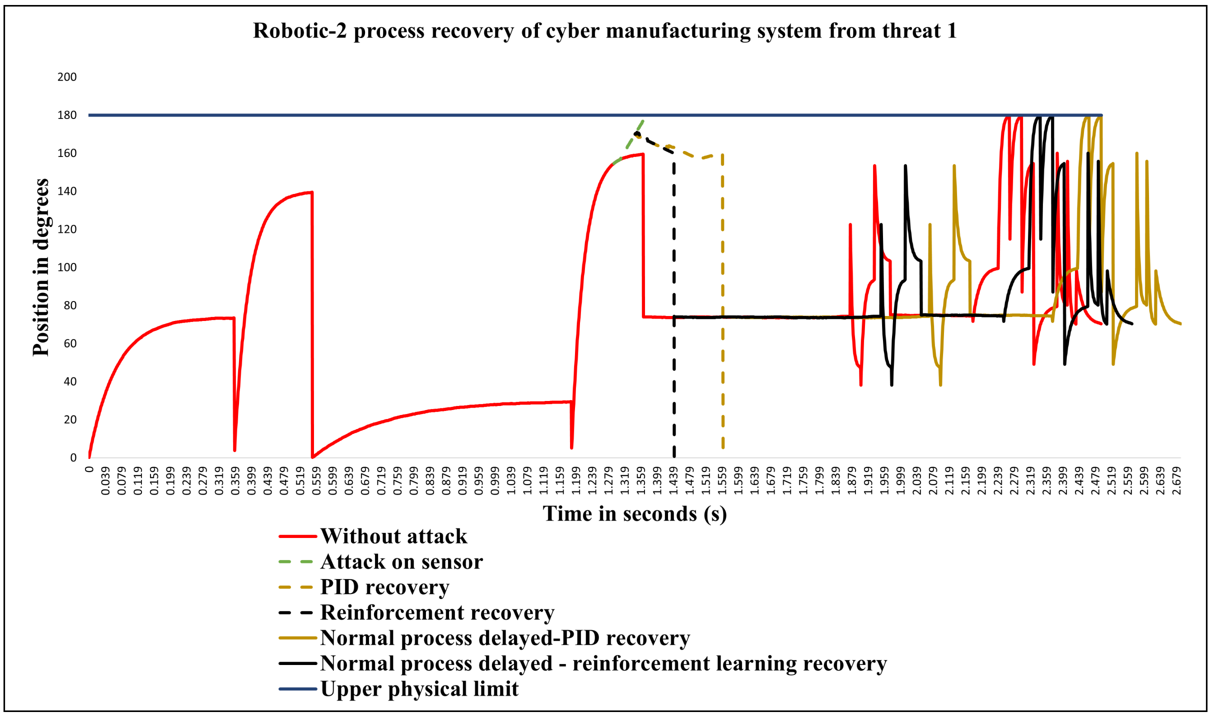


Figure 4.21: Process level recovery of robotic arm 2 from threat 1 with no recovery, reinforcement learning recovery, PID recovery.

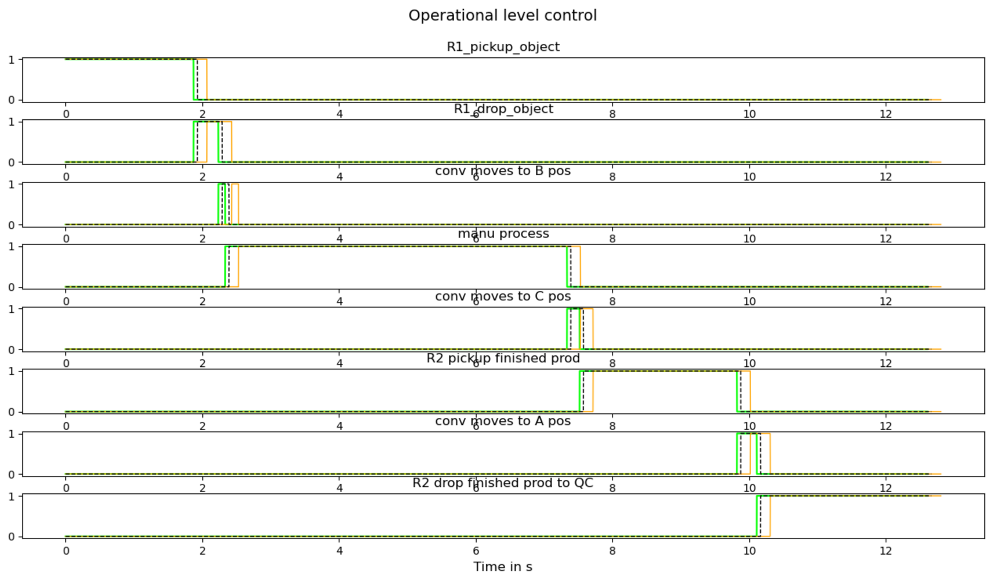


Figure 4.22: Operational level recovery of CMS from threat 1 with no recovery, reinforcement learning recovery, PID recovery. The lime color represents the normal functioning of manufacturing systems. The black color represents recovery with reinforcement learning and the orange color represents recovery with PID.

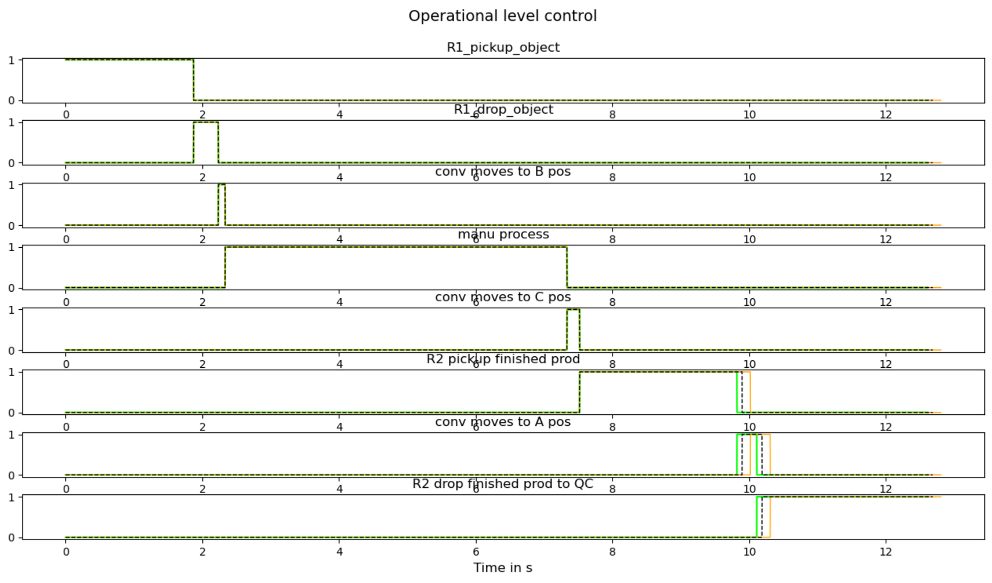


Figure 4.23: Operational level recovery of CMS from threat 1 with no recovery, reinforcement learning recovery, PID recovery. The lime color represents the normal functioning of manufacturing systems. The black color represents recovery with reinforcement learning and the orange color represents recovery with PID.

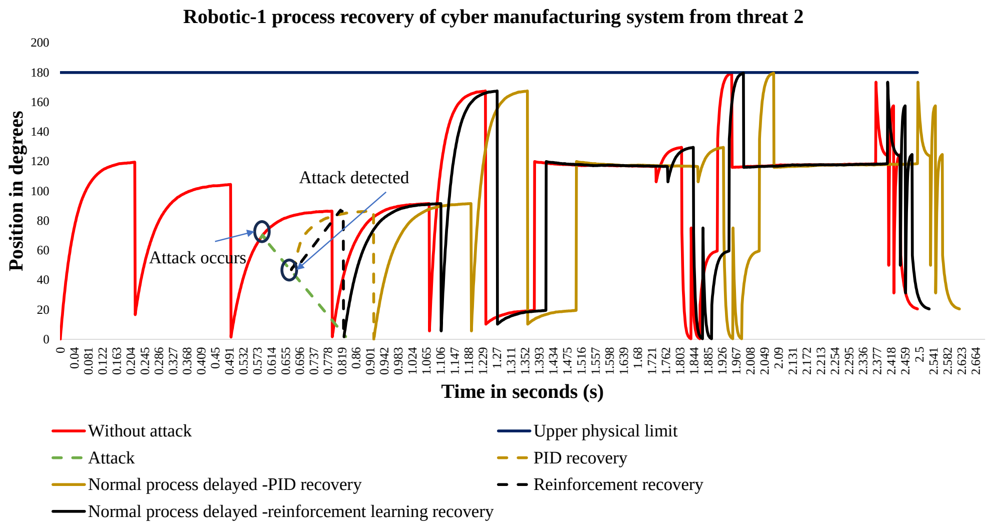


Figure 4.24: Process level recovery of robotic arm 1 from threat 2 with no recovery, reinforcement learning recovery, PID recovery.

sensor. The impact of manual tuning PID versus reinforcement learning is notable, emphasizing the efficiency of the reinforcement learning agent in such recovery scenarios.

The recovery agent's efficiency in swiftly restoring the system to normal conditions, coupled with its seamless handling of time delays, positions it as a robust solution for mitigating the impact of Threat 1 and Threat 2. The direct path taken by the reinforcement learning agent showcases its effectiveness in recovering from false data injection attacks on sensors in manufacturing systems.

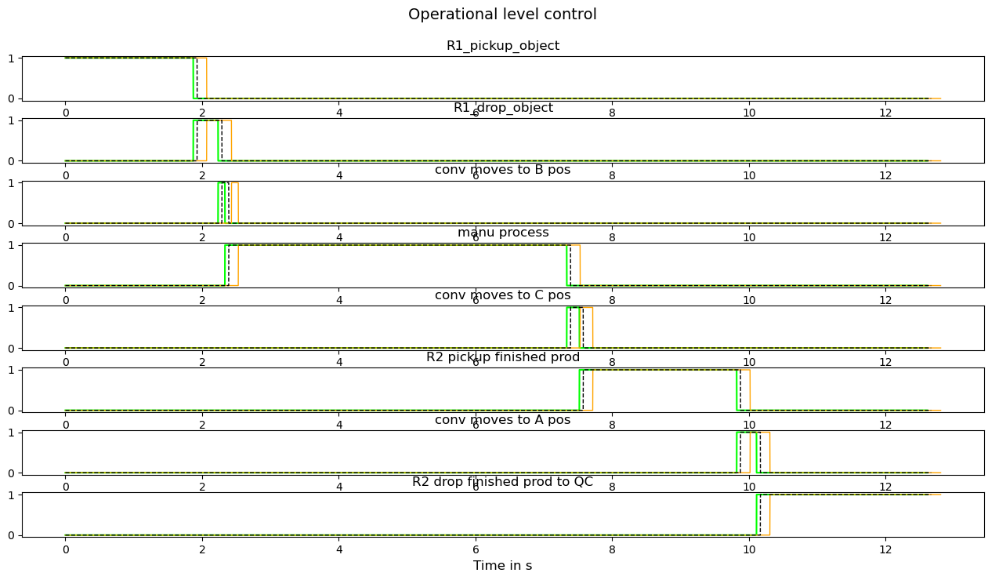


Figure 4.25: Operation level recovery of cyber manufacturing systems (R1) from false data injection attacks on sensors (Threat 2). The lime color represents the normal functioning of manufacturing systems. The black color represents recovery with reinforcement learning and the orange color represents recovery with PID.

4.6 Metrics to evaluate the performance of the agent

However, validating the performance of the recovery strategies presented in this paper requires the establishment of key metrics. Three primary metrics have been selected for this purpose, drawing inspiration from traditional key performance indicators commonly used by manufacturing plants. These metrics are essential for evaluating the effectiveness of the recovery strategies and making informed comparisons. The three chosen metrics are:

Downtime: Downtime represents the total time during which the manufacturing process is non-operational, often due to disruptions or issues, and is given by 4.4. Minimizing downtime is a critical objective in manufacturing, as it directly impacts production efficiency and profitability. The recovery strategies introduced in this paper will be assessed based on their ability to reduce downtime.

$$Downtime = \frac{\text{Time when machine is non operational}}{\text{Total production time}} \tag{4.4}$$

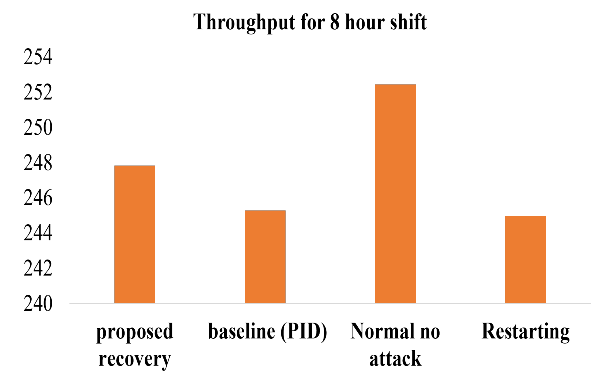
Efficiency: Efficiency reflects how well resources are utilized in the manufacturing process to produce the desired output and is given by 4.5. Maximizing efficiency is a fundamental objective for manufacturing operations, as it directly impacts production costs and resource allocation. The proposed recovery strategies will be analyzed for their efficiency improvements.

$$Efficiency = \frac{\text{Actual cycle time}}{\text{Expected cycle time}} \tag{4.5}$$

Throughput: Throughput represents the rate at which products are processed or manufactured within a given time frame and is given by 4.6. It's a fundamental indicator of production capacity and efficiency. Maximizing throughput is a key goal for manufacturers, as it directly impacts the rate of production and, ultimately, revenue generation. The recovery strategies will be analyzed in terms of their impact on throughput improvement.

$$Throughput = \frac{\text{Units produced}}{\text{Time}} \tag{4.6}$$

Figures 4.26 provide a visual representation of the recovery agent's per-



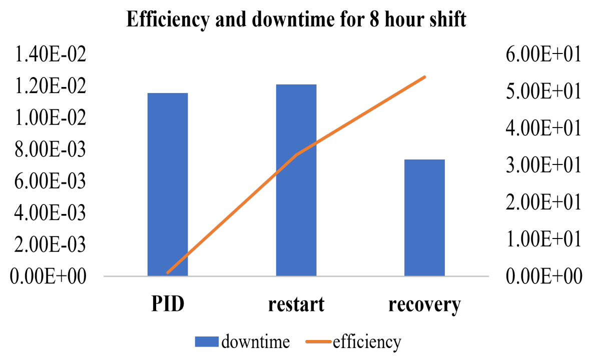


Figure 4.26: Metrics to evaluate the performance of the recovery agent

formance relative to restarting and manually tuning the PID controller. The simulation results reveal a notable decline in the manufacturing systems's performance due to a false data injection attack on the sensor. However, the primary objective of minimizing downtime in the cyber manufacturing systems is effectively addressed by the reinforcement learning-based recovery strategy. A key observation from the figures is that the parts produced through the reinforcement learning-based approach are significantly higher compared to the alternative methods. This highlights the resilience of the recovery mechanism, which, even after a system attack, mitigates the impact and allows for continued production. The ability of the reinforcement learning agent to reduce downtime and enhance overall production output underscores its efficacy in handling and recovering from cyber threats in the manufacturing environment.

Chapter 5

Discussion and Conclusion

This chapter highlights the novelty of the research proposed, its key contribution and further discusses the limitations of this work.

5.1 Contribution

This research contributes to the advancement of understanding reinforcement learning agent-augmented recovery strategies in cyber-manufacturing systems. The key contributions of this research are outlined.

The introduced recovery systems architecture is the first of its kind for cyber manufacturing systems. This architecture comprises a systems layer, an attack identification layer, a data auditing and detection layer, and a recovery layer. While previous efforts primarily focused on recovery from an error perspective, our work emphasizes recovery from the standpoint of cyber threats. This framework is designed to be a generalized and flexible structure capable of accommodating various cyber manufacturing systems. Within the attack identification layer, we specifically address cyber-attacks such as false data injection attacks (commonly known as sniffing and spoofing attacks), which have the potential to manipulate manufacturing systems sensors. Consequently, it was essential to analyze these attacks in the context of cyber manufacturing systems. Prior research was predominantly limited to phishing or SQL injection attacks within manufacturing systems, overlooking the critical information associated with sniffing, spoofing attacks, and domain name rebinding attacks on manufacturing systems.

The recovery layer is a multi-stage process. The first stage involves the recovery of the process level controller, followed by the second stage, which achieves continuous functioning through the operational level controller. In the first stage, the implementation of reinforcement learning is introduced to recover the manufacturing systems from any state. During training, the agent is trained on an epsilon-greedy policy, theoretically exploring the entire state space. The development of recovery agents addresses multiple challenges, including distinguishing between normal and abnormal states, predicting future states based on the current state, taking sequential actions to move the process from an abnormal state to a normal state, and controlling the process. All challenges associated with the reinforcement learning agent are successfully addressed within this thesis.

Technical challenges in the implementation of reinforcement learning are also addressed. Reinforcement learning agents learn through interaction with the environment, and training them in a real environment is challenging due to their characteristic of taking random actions. To overcome this, we trained the agents in a virtual environment, which presented the first challenge of designing a suitable virtual environment. The second challenge involved creating a reward function for the agents to learn effectively. Lastly, we fine-tuned the exploration and exploitation characteristics of the recovery agent.

The second stage of continuous operation ideally requires multiple agents. However, the approach of introducing a time delay effectively addresses this issue. The development of a table that incorporates the time delay for different training agents allows us to delay the process as soon as an attack is detected. This has enabled the operational level to continuously function.

5.2 Discussion

5.2.1 Reliance on detection system

The success of the proposed work is heavily dependent on the accuracy of the detection system. Choosing an appropriate detection system introduces several challenges, as there is often a tradeoff between achieving higher true positives and minimizing false negatives. The effectiveness of the overall recovery strategy hinges on the reliability and precision of the detection system in identifying and alerting the presence of false data injection attacks on sensors

5.2.2 Stability of the cyber manufacturing systems

The challenge of defining stability in control systems, especially in the context of reinforcement learning (RL), is indeed a complex task. Traditionally, Lyapunov functions have been employed to analyze the stability of dynamical systems (Lyapunov 1992). However, the lack of a clear mathematical model and understanding of the RL algorithms poses a unique set of challenges in establishing stability for RL-based control systems. One potential avenue for assessing stability in RL applications is to examine the convergence of rewards. A stable RL algorithm should exhibit consistent and convergent reward values over time. Tracking the trend of rewards during training and ensuring that they stabilize within an acceptable range can provide insights into the stability of the learned policy.

Moreover, the trade-off between exploitation and exploration in RL is a crucial factor. A stable RL algorithm should strike a balance between exploiting the learned policy and exploring new actions to adapt to changes or uncertainties. Monitoring this trade-off and ensuring that the agent converges to a robust policy without drastic fluctuations is indicative of stability. Additionally, leveraging insights from control theory and adapting them to the unique characteristics of RL algorithms could contribute to addressing the stability concerns in these complex cyber-physical systems.

5.2.3 Multiple sensor attacked at the same time

As of now, this work focuses solely on addressing a single attack on the system. However, in scenarios where multiple attacks occur simultaneously, the proposed strategy would need to be adapted to incorporate a multi-agent recovery. Here each agent responds to an alert and coordinates among themselves to ensure successful recovery. This adjustment becomes crucial to ensure the effectiveness of the recovery strategy in the face of multiple concurrent attacks on the cyber manufacturing systems.

5.2.4 Complexity of the simulator

While the proposed simulation serves as a promising proof of concept, it's essential to acknowledge that real-world manufacturing scenarios involve not just a singular process but multiple processes running in parallel. Expanding the scope of this work to accommodate and address the challenges posed by multiple parallel processes would enhance its applicability and relevance in complex manufacturing environments.

5.2.5 Sample Inefficiency and Real-World Transfer Challenges

Reinforcement learning indeed faces the challenge of sample inefficiency, particularly in scenarios with limited and repetitive real-world data. In the manufacturing domain, where patterns in data can be highly similar, training an efficient agent becomes a challenging task due to the scarcity of diverse samples. To overcome this hurdle, it is crucial to explore strategies for improving sample efficiency. Techniques such as data augmentation, ensemble learning, or leveraging domain knowledge to generate synthetic data could be explored.

Furthermore, transferring the learned policies from simulation to the real-world manufacturing testbed is a significant challenge. Discrepancies between the simulated environment and the actual system can lead to a lack of generalization. Addressing this issue may involve refining the simulation model to better match the real-world dynamics or adopting techniques like domain adaptation to bridge the gap between simulation and reality. Another critical consideration is synchronizing the decisions made by the reinforcement learning agent with the hardware's clock timing in the manufacturing systems. Ensuring that the agent's decisions align seamlessly with the physical processes is essential for the successful implementation of the proposed recovery strategy. Fine-tuning the agent's temporal aspects and addressing any timing mismatches are crucial steps in achieving effective real-world deployment.

5.3 Conclusion

In the realm of resilient manufacturing systems, this paper brings significant contributions by addressing the critical aspect of recovery. While existing literature predominantly emphasizes prevention and detection strategies, the proposed work focuses on recovery. The key contributions of this paper can be summarized as follows. This paper pioneers the development of a robust recovery strategy for cyber manufacturing systems. The approach leverages reinforcement learning to guide the system back to its normal operating conditions after being subjected to false data injection attacks on sensors. By utilizing the principles of reinforcement learning, the recovery process becomes adaptive, learning from the system's environment and efficiently navigating toward normalcy. Using a manufacturing systems simulator this work demonstrates that our proposed work can successfully recover the system and ensure that it continues its functioning by minimizing the downtime, and outperforms the manually tuned PID controller. By contributing pioneering methodologies for recovery and continuous operation, this paper significantly advances the field of resilient manufacturing systems. It addresses a critical gap in existing research and provides practical insights into enhancing the overall robustness and adaptability of cyber-physical manufacturing environments.

Bibliography

Afonso, Francisco, Carlos Silva, Adriano Tavares, and Sergio Montenegro (2008). “Application-level fault tolerance in real-time embedded systems”. In: *2008 International Symposium on Industrial Embedded Systems*. IEEE, pp. 126–133.

Akharas, Ismail, Michael P Hennessey, and Eric J Tornoe (2020). “Simulation and Visualization of Dynamic Systems in Virtual Reality Using SolidWorks, MATLAB/Simulink, and Unity”. In: *ASME International Mechanical Engineering Congress and Exposition*. Vol. 84546. American Society of Mechanical Engineers, V07AT07A039.

Akowuah, Francis, Romesh Prasad, Carlos Omar Espinoza, and Fanxin Kong (2021). “Recovery-by-learning: Restoring autonomous cyber-physical systems from sensor attacks”. In: *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, pp. 61–66.

Al Faruque, Mohammad Abdullah, Sujit Rokka Chhetri, Arquimedes Canedo, and Jiang Wan (2016). “Acoustic side-channel attacks on additive manufacturing systems”. In: *2016 ACM/IEEE 7th international conference on Cyber-Physical Systems (ICCPS)*. IEEE, pp. 1–10.

Alnabulsi, Hussein, Md Rafiqul Islam, and Quazi Mamun (2014). “Detecting SQL injection attacks using SNORT IDS”. In: *Asia-Pacific World Congress on Computer Science and Engineering*. IEEE, pp. 1–7.

Andersson, Kristin, Bengt Lennartson, and Martin Fabian (2006). “Restarting flexible manufacturing systems; Synthesis of restart states”. In: *2006 8th International Workshop on Discrete Event Systems*. IEEE, pp. 201–206.

Arul Prakash, Sakthi Kumar, Tobias Mahan, Glen Williams, Christopher McComb, Jessica Menold, and Conrad S Tucker (2020). “Detection of system compromise in additive manufacturing using video motion magnification”. In: *Journal of Mechanical Design* 142.3, p. 031109.

Bayens, Christian, Tuan Le, Luis Garcia, Raheem Beyah, Mehdi Javanmard, and Saman Zonouz (2017). “See no evil, hear no evil, feel no evil, print no evil? malicious fill patterns detection in additive manufacturing”. In: *26th USENIX Security Symposium (USENIX Security 17)*, pp. 1181–1198.

Belikovetsky, Sofia, Yosef Solewicz, Mark Yampolskiy, Jinghui Toh, and Yuval Elovici (2017). “Detecting cyber-physical attacks in additive manufacturing using digital audio signing”. In: *arXiv preprint arXiv:1705.06454*.

Belikovetsky, Sofia, Mark Yampolskiy, Jinghui Toh, Jacob Gatlin, and Yuval Elovici (2017). “dr0wned–{Cyber-Physical} attack with additive manufacturing”. In: *11th USENIX Workshop on Offensive Technologies (WOOT 17)*.

Brandman, Josh, Logan Sturm, Jules White, and Chris Williams (2020). “A physical hash for preventing and detecting cyber-physical attacks in additive manufacturing systems". In: *Journal of Manufacturing Systems* 56, pp. 202–212.

Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba (2016). "Openai gym". In: *arXiv preprint arXiv:1606.01540*.

Chenm, Chu Xin and MM Trivedi (1991). "A task planner for sensor-based inspection and manipulation robots." In: *International Symposium on Intelligent Robots*, pp. 591–603.

Chhetri, Sujit Rokka, Arquimedes Canedo, and Mohammad Abdullah Al Faruque (2016). "Kcad: kinetic cyber-attack detection method for cyber-physical additive manufacturing systems". In: *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, pp. 1–8.

Cruz Salazar, Luis Alberto, Daria Ryashentseva, Arndt Lüder, and Birgit Vogel-Heuser (2019). "Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of selected approaches mapping four agent patterns". In: *The International Journal of Advanced Manufacturing Technology* 105.9, pp. 4005–4034.

Der Jeng, Mu (1997). "Petri nets for modeling automated manufacturing systems with error recovery". In: *IEEE Transactions on Robotics and Automation* 13.5, pp. 752–760.

Dietterich, Thomas G (2000). "Hierarchical reinforcement learning with the MAXQ value function decomposition". In: *Journal of artificial intelligence research* 13, pp. 227–303.

Espinoza-Zelaya, Carlos and Young Bai Moon (2022). “Resilience Enhancing Mechanisms for Cyber-Manufacturing Systems against Cyber-Attacks”. In: *IFAC-PapersOnLine* 55.10, pp. 2252–2257.

Feder, Maddalena, Andrea Giusti, and Renato Vidoni (2022). “An approach for automatic generation of the URDF file of modular robots from modules designed using SolidWorks”. In: *Procedia Computer Science* 200, pp. 858–864.

Gatlin, Jacob, Sofia Belikovetsky, Samuel B Moore, Yosef Solewicz, Yuval Elovici, and Mark Yampolskiy (2019). “Detecting sabotage attacks in additive manufacturing using actuator power signatures”. In: *IEEE Access* 7, pp. 133421–133432.

Greenberg, Andy (2019). *A Guide to LockerGoga, the Ransomware Crippling Industrial Firms.* Tech. rep. WIRED,

Haerder, Theo and Andreas Reuter (1983). “Principles of transaction-oriented database recovery”. In: *ACM computing surveys (CSUR)* 15.4, pp. 287–317.

Hojjati, Avesta, Anku Adhikari, Katarina Struckmann, Edward Chou, Thi Ngoc Tho Nguyen, Kushagra Madan, Marianne S Winslett, Carl A Gunter, and William P King (2016). “Leave your phone at the door: Side channels that reveal factory floor secrets”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 883–894.

Juliani, Arthur, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange (2020). “Unity: A general platform for intelligent agents”. In: *arXiv preprint arXiv:1809.02627*. URL: https://arxiv.org/pdf/1809.02627.pdf.

Kevin, Du (2019a). *Domain Name System and Attacks.” Computer Internet Security: A Hands-on Approach, Second Edition.* Self published.

— (2019b). *Sniffing and Spoofing.” Computer Internet Security: A Hands-on Approach, Second Edition.* Self published.

Kim, KH and Howard O. Welch (1989). “Distributed execution of recovery blocks: An approach for uniform treatment of hardware and software faults in real-time applications”. In: *IEEE transactions on Computers* 38.5, pp. 626–636.

Kim, Ho-myung and Kyung-ho Lee (2022). “Iiot malware detection using edge computing and deep learning for cybersecurity in smart factories”. In: *Applied Sciences* 12.15, p. 7679.

Kirk, E. Donald (1970). *Optimal Control Theory: An Introduction.* Prentice-Hall.

Klein, Inger, Peter Jonsson, and Christer Bäckström (1999). “Efficient planning for a miniature assembly line”. In: *Artificial Intelligence in Engineering* 13.1, pp. 69–81.

Kong, Fanxin, Meng Xu, James Weimer, Oleg Sokolsky, and Insup Lee (2018). “Cyber-physical system checkpointing and recovery”. In: *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS).* IEEE, pp. 22–31.

Kovalenko, Ilya, Efe C. Balta, Dawn M. Tilbury, and Kira Barton (2023). “Cooperative Product Agents to Improve Manufacturing System Flexibility: A Model-Based Decision Framework”. In: *IEEE Transactions* *on Automation Science and Engineering* 20.1, pp. 440–457. DOI: `10.1109/TASE.2022.3156384`.

Krundyshev, V and M Kalinin (2020). “Prevention of cyber attacks in smart manufacturing applying modern neural network methods”. In: *IOP conference series: materials science and engineering*. Vol. 940. 1. IOP Publishing, p. 012011.

Labor Statistics, Bureau of (2019). *Manufacturing, durable manufaturing and nondurable manufacturing*. Tech. rep. Bureau of Labor Statistics.

Lample, Guillaume and Devendra Singh Chaplot (2017). “Playing FPS games with deep reinforcement learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1.

Langner, Ralph (2011). “Stuxnet: Dissecting a cyberwarfare weapon”. In: *IEEE Security & Privacy* 9.3, pp. 49–51.

Lee, Jay, Behrad Bagheri, and Hung-An Kao (2015). “A cyber-physical systems architecture for industry 4.0-based manufacturing systems”. In: *Manufacturing letters* 3, pp. 18–23.

Lee, Robert.M, Michael Assante, and Tim Conway (2014). “German Steel Mill Cyber Attack”. In: *Industrial Control System*.

Lepuschitz, Wilfried, Alois Zoitl, Mathieu Vallée, and Munir Merdan (2011). “Toward Self-Reconfiguration of Manufacturing Systems Using Automation Agents”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 41.1, pp. 52–69. DOI: `10.1109/TSMCC.2010.2059012`.

Loborg, Peter (1994). “Error recovery in automation-an overview”. In: *AAAI Spring Symposium on Detecting and Resolving Errors in Manufacturing Systems*, pp. 94–100.

Loborg, Peter and Anders Törne (1996). “Towards error recovery in sequential control applications”. In: *The 2nd World Automation Congress, WAC’96.* Citeseer.

Lyapunov, Aleksandr Mikhailovich (1992). “The general problem of the stability of motion”. In: *International journal of control* 55.3, pp. 531–534.

Mavrikios, Dimitris, Nikolaos Papakostas, Dimitris Mourtzis, and George Chryssolouris (2013). “On industrial learning and training for the factories of the future: a conceptual, cognitive and technology framework”. In: *Journal of Intelligent Manufacturing* 24, pp. 473–485.

Michael, Bartock, Cichonski Jeffrey, Murugiah Souppaya, Matthew Smith, Greg Witte, and Karen Scarfone (2016). *Guide for Cybersecurity Event Recovery.* Tech. rep. National Institute of Standards and Technology.

Mityukov, EA, AV Zatonsky, PV Plekhov, and NV Bilfeld (2019). “Phishing detection model using the hybrid approach to data protection in industrial control system”. In: *IOP Conference Series: Materials Science and Engineering.* Vol. 537. 5. IOP Publishing, p. 052014.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller (2013). “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602.*

Moore, Samuel Bennett, William Bradley Glisson, and Mark Yampolskiy (2017). “Implications of malicious 3D printer firmware”. In: Proceedings of the 50th Hawaii International Conference on System Sciences.

Office of, Management and Budget (2015). *Cybersecurity Strategy and Implementation Plan (CSIP) for the Federal Civilian Government, OMB Memorandum 16-04, October 30.* Tech. rep. Office of Management and Budget (OMB).

Pan, Yao, Jules White, Douglas Schmidt, Ahmad Elhabashy, Logan Sturm, Jaime Camelio, and Christopher Williams (2017). “Taxonomies for reasoning about cyber-physical attacks in IoT-based manufacturing systems”. In.

Prasad, Romesh, Seyed Alireza Zarrin Mehr, and Young Moon (2023). “Recovery systems architecture for cyber-manufacturing systems against cyber-manufacturing attacks: Reinforcement learning approach”. In: *Manufacturing Letters* 35, pp. 851–860.

Prasad, Romesh and Young Moon (2021). “Adaptive Intrusion Detection System for Cyber-Manufacturing System”. In: *ASME International Mechanical Engineering Congress and Exposition.* Vol. 85567. American Society of Mechanical Engineers, V02BT02A010.

— (2022a). “Architecture for Preventing and Detecting Cyber Attacks in Cyber-Manufacturing System”. In: *IFAC-PapersOnLine* 55.10, pp. 2246–2251.

— (2022b). “Comprehensive Analysis of Cyber-Manufacturing Attacks Using a Cyber-Manufacturing Testbed”. In: *ASME International Me-*

chanical Engineering Congress and Exposition. Vol. 86649. American Society of Mechanical Engineers, V02BT02A015.

Prasad, Romesh, Matthew K Swanson, and Young Moon (2022). “Recovering From Cyber-Manufacturing Attacks by Reinforcement Learning”. In: *ASME International Mechanical Engineering Congress and Exposition*. Vol. 86649. American Society of Mechanical Engineers, V02BT02A014.

Rais, Muhammad Haris, Ye Li, and Irfan Ahmed (2021). “Spatiotemporal G-code modeling for secure FDM-based 3D printing”. In: *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, pp. 177–186.

Ron, Ross, Winstead Mark, and McEvilley Michael (2022). *Engineering Trustworthy Secure Systems*. Tech. rep. National Institute of Standards and Technology.

Salazar, Luis A Cruz and Oscar A Rojas Alvarado (2014). “The future of industrial automation and IEC 614993 standard”. In: *2014 iii international congress of engineering mechatronics and automation (ciima)*. IEEE, pp. 1–5.

Shi, Zhangyue, Chen Kan, Wenmeng Tian, and Chenang Liu (2021). “A Blockchain-based G-code protection approach for cyber-physical security in additive manufacturing”. In: *Journal of Computing and Information Science in Engineering* 21.4, p. 041007.

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. (2018). “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419, pp. 1140–1144.

Smara, Mounya, Makhlouf Aliouat, Saad Harous, and Al-Sakib Khan Pathan (2022). "Robustness improvement of component-based cloud computing systems". In: *The Journal of Supercomputing*, pp. 1–33.

Song, Jinwoo, Harika Bandaru, Xinyu He, Zhenyang Qiu, and Young B Moon (2020). "Layered image collection for real-time defective inspection in additive manufacturing". In: *ASME International Mechanical Engineering Congress and Exposition*. Vol. 84492. American Society of Mechanical Engineers, V02BT02A006.

Song, Jinwoo, Chunxi Wang, Charlélie Saudrais, Matthew K Swanson, Emily Ann Greaney, and Young B Moon (2020). "Cyber-Manufacturing System Testbed Development: Adversarial Insider Manipulation". In: *Procedia CIRP* 93, pp. 180–185.

Song, Zhengyi and Young Moon (2017). "Assessing sustainability benefits of cybermanufacturing systems". In: *The International Journal of Advanced Manufacturing Technology* 90, pp. 1365–1382.

Song, Zhengyi and Young B Moon (2016). "Performance analysis of cybermanufacturing systems: A simulation study". In: *Product Lifecycle Management for Digital Transformation of Industries: 13th IFIP WG 5.1 International Conference, PLM 2016, Columbia, SC, USA, July 11-13, 2016, Revised Selected Papers 13*. Springer, pp. 592–605.

Sousa, Mario de and Adriano A Santos (2007). "Management of Replicated IEC 61499 Applications". In: *2007 5th IEEE International Conference on Industrial Informatics*. Vol. 1. IEEE, pp. 231–236.

Staff, Reuters (2017). *Renault-Nissan resumes nearly all production after cyber-attack.* Tech. rep. WIRED,

Standards, National Institute of and Technology (2018). *The NIST Cybersecurity Framework.* Tech. rep. National Institute of Standards and Technology.

Sturm, Logan D, Christopher B Williams, Jamie A Camelio, Jules White, and Robert Parker (2014). “Cyber-physical vulnerabilities in additive manufacturing systems”. In: *2014 International Solid Freeform Fabrication Symposium.* University of Texas at Austin.

— (2017). “Cyber-physical vulnerabilities in additive manufacturing systems: A case study attack on the. STL file with human subjects”. In: *Journal of Manufacturing Systems* 44, pp. 154–164.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction.* MIT press.

Tittus, Michael, Sven-Arne Andreasson, Anders Adlemo, and JE Frey (2000). “Fast restart of manufacturing cells using restart points”. In: *Proc. of the 4th World Automation Congress.* Citeseer.

Watanabe, Aki and Ken Sakamura (1995). “Design fault tolerance in operating systems based on a standardization project”. In: *Twenty-Fifth International Symposium on Fault-Tolerant Computing. Digest of Papers.* IEEE, pp. 372–380.

Wellener, Paul, Steve Shepley, Ben Dollar, Laaper Stephen, Ashton Manolian Heather, and Beckoff David (2019). *Cybersecurity for Smart Factories.* Tech. rep. Deloitte research center for energy and industrial group.

Worley, Michael, Christopher Caridi, Michelle Alvarez, Karlina Bakken, Yannick Bedard, Michele Brancati, Christopher Bedell, Joshua Chung, Scott Craig, Joseph DiRe, John Dwyer, Emmy Ebanks, Richard Emerson, Charlotte Hammond, and Kevin Henson (2023). *X-Force Threat Intelligence Index*. Tech. rep. IBM.

Wu, Mingtao (2019). "Intrusion Detection for Cyber-Physical Attacks in Cyber-Manufacturing System". PhD thesis. Syracuse University.

Wu, Mingtao and Young B Moon (2017). "Taxonomy of cross-domain attacks on cybermanufacturing system". In: *Procedia Computer Science* 114, pp. 367–374.

— (2019). "Intrusion detection system for cyber-manufacturing system". In: *Journal of Manufacturing Science and Engineering* 141.3, p. 031007.

— (2020). "Alert correlation for detecting cyber-manufacturing attacks and intrusions". In: *Journal of Computing and Information Science in Engineering* 20.1, p. 011004.

Wu, Mingtao, Zhengyi Song, and Young B Moon (2019). "Detecting cyber-physical attacks in CyberManufacturing systems with machine learning methods". In: *Journal of intelligent manufacturing* 30, pp. 1111–1123.

Xiao, Claud (2013). *Security Attack to 3D Printing*. Tech. rep. Keynote at XCon2013.

Yampolskiy, Mark, Wayne E King, Jacob Gatlin, Sofia Belikovetsky, Adam Brown, Anthony Skjellum, and Yuval Elovici (2018). "Security of additive manufacturing: Attack taxonomy and survey". In: *Additive Manufacturing* 21, pp. 431–457.

Zhang, Lin, Pengyuan Lu, Fanxin Kong, Xin Chen, Oleg Sokolsky, and Insup Lee (2021). "Real-time attack-recovery for cyber-physical systems using linear-quadratic regulator". In: *ACM Transactions on Embedded Computing Systems (TECS)* 20.5s, pp. 1–24.

Zhang, Lin, Kaustubh Sridhar, Mengyu Liu, Pengyuan Lu, Xin Chen, Fanxin Kong, Oleg Sokolsky, and Insup Lee (2023). "Real-Time Data-Predictive Attack-Recovery for Complex Cyber-Physical Systems". In: *2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, pp. 209–222.

Vita
Romesh Prasad

PERSONAL STATEMENT

A curious person currently in his final year of Ph.D. training, with 4 years of research experience in the field of computational research, simulation and modeling and 3 years of research experience in applying deep reinforcement learning for various applications. This training makes me an independent researcher and skilled employee in applying analytical tools to real-world problems. A confident presenter at the conference, three years of teaching assistant experience, and the ability to communicate complex ideas to wider audiences.

RESEARCH INTERESTS

resiliency, safe reinforcement learning, unmanned manufacturing systems

EDUCATION

Syracuse University, Syracuse, NY Dec 2023

Ph.D., Department of Mechanical and Aerospace Engineering

Thesis Title: Deep Reinforcement learning based resilient manufacturing control system

Syracuse University, Syracuse, NY May 2019

M.S., Department of Mechanical and Aerospace Engineering

University of Mumbai, Mumbai, India May 2016

B.S., Mechanical Engineering

PROFESSIONAL RESEARCH EXPERIENCE

National Renewable Energy Laboratory CO, USA

Deep Reinforcement Learning Research Intern 2023

- Applied deep reinforcement learning research on inverter-based frequency controller of power grid
- Developed custom reinforcement learning agent environment, Python, OpenAI, NumPy extensively used
- Developed model-free hybrid action for on policy gradient algorithms such as PPO, A2C and DQN
- Applied multi-agent reinforcement learning for attacker versus defender policies
- Models was trained on stable-baselines3 PyTorch, scenarios run-on High-Performance-Computing (HPC)

PUBLICATIONS

- (2023) R. Prasad and Y.B. Moon, "Recovery of cyber manufacturing systems from false data attacks on sensors using reinforcement learning," Journal of Manufacturing Systems, In Pipeline.

- (2023) R. Prasad and Y.B. Moon, “Recovery systems architecture for cyber attacks on manufacturing systems,” NAMRC, Jersey City, NJ.
- (2022) R. Prasad and Y. B. Moon, “Comprehensive Analysis of cyber-manufacturing attacks using a cyber-manufacturing testbed,” ASME 2022 International Mechanical Engineering Congress and Exposition (IMECE), Columbus, Ohio.
- (2022) R. Prasad, M. K. Swanson, and Y. B. Moon, “Recovery from cyber-manufacturing attacks by reinforcement learning,” ASME 2022 International Mechanical Engineering Congress and Exposition (IMECE), Columbus, Ohio.
- (2022) R. Prasad and Y. B. Moon, “Architecture for Preventing and Detecting Cyber-Attacks in Cyber-Manufacturing Systems,” 10th IFAC Conference on Manufacturing Modelling, Management and Control (MIM), Nantes, France.
- (2021) F. Akowuah, R. Prasad, C. O. Espinoza and F. Kong, “Recovery-by-learning: Restoring autonomous cyber-physical systems from sensor attacks,” 2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications, (RTCSA) pp. 61–66.