Reinforcement Learning based Autonomous Cyber Attack Response in Nuclear Power Plants

Pavan Kumar Vaddi[a], Yunfei Zhao[a] and Carol Smidts[a]
[a] Department of Mechanical and Aerospace Engineering, The Ohio State University, Columbus, USA,
vaddi.3@osu.edu, zhao.2263@osu.edu, smidts.1@osu.edu

Abstract: Cyber-attacks on digital industrial control systems are becoming increasingly frequent. Given the rise of digitalization in nuclear power plants and the potentially hazardous consequences of a successful cyber-attack on nuclear power plants and similar safety-critical systems, it is imperative that research should be focused on industrial control system cyber-attack detection and mitigation. In this paper we explore the use of reinforcement learning to develop an autonomous cyber-attack response system for nuclear power plants, specifically the digital feedwater control system of a pressurized water reactor. The cyber-attacks are modeled as Stackelberg games between the defender i.e., the plant operator and the attacker. The system state transition probabilities are defined using probabilistic risk assessment. The optimal defender strategy is computed using multi-agent Q-learning, where the Stackelberg equilibrium over the current Q-values is used at every update. The advent of digital twins for nuclear power plants enables us to simulate a wide variety of cyber-attacks for as many instances as needed to fully train the reinforcement learning agents.

1. INTRODUCTION

The process of digitalization of industrial control systems (ICSs) has enabled automation, and improved control, monitoring and diagnostics. However, digitalization has also opened such systems to a new type of threat, the cyber-attacks. In contrast to the traditional information technology (IT) systems, where a cyber-attack primarily affects confidentiality, privacy and availability of systems with implications largely in the digital realm, a cyber-attack on an industrial control system that interacts with the physical world can have ramifications in the physical realm and in worst case scenarios can even endanger human life along with causing economic losses due to equipment damage. Well known examples of cyber-attacks on industrial control systems include the Stuxnet incident in 2010 that crippled Iran’s Natanz nuclear fuel enrichment facility [1], and the Colonial pipeline ransomware incident in 2021 [2] that caused fuel shortages in the United States. Additionally, the Triton malware identified in the Triconex PLCs in Saudi Arabia’s petrochemical plants in 2017 was nicknamed as the “most murderous malware” [3]. Varuttamaseni et al. [4] presented a list of cyber events, both intentional attacks and unintentional events in nuclear facilities from various countries. The consequences of cyber-attacks can be severe in the context of nuclear systems or nuclear reactors, where a potential incident can release radioactive material into the environment, rendering the surroundings inhospitable for several decades or even centuries. Hence it is important to detect cyber-attacks. However, it is also important to assume that cyber-attacks are inevitable and to be appropriately prepared to mitigate such attacks.

It is implicit that the response of a NPP under a cyber-attack is dependent on the actions of both the defender i.e., the plant operator working to protect the plant from any damage and the attacker i.e., the malicious actor trying to inflict damage on the plant. Taking this into consideration Zhao et al. [5], [6] and Maccarone and Cole [7] presented two player non-cooperative general sum stochastic games based frameworks with the defender and the attacker as the players, in which the defender’s response strategy is formulated taking the possible future attacker actions and the corresponding long term consequences into consideration along with the current state. Zhao et al. [5], [6] assume that both players act simultaneously and have complete information, and use the concept of Nash equilibrium to compute optimal strategies in infinite horizon as well as finite horizon cases, while Maccarone and Cole [7] applied the Bayesian games framework to the case in which each player has incomplete information about the other, and considered simultaneous games as well and hierarchical games. In contrast to simultaneous games, a hierarchical game also known as a Stackelberg game involves one of the players known as the leader, enforcing their strategy on the other player known as the follower [8]. Planning based methods were used to solve the games and compute the optimal response strategies.

Cyber-attacks on nuclear facilities are generally implemented by enemy states with unlimited resources. Hence the formulation of an optimal response strategy for a wide variety of scenarios is of vital importance. Deploying planning based methods which involve construction of the game models, i.e., identifying the system states and state transitions, and searching for optimal response strategies for a large number of such complex cases is cumbersome and may not be feasible. The reinforcement learning framework, in which an agent learns and evolves by interacting with the environment, and algorithms such as Q-learning which are model-free in nature provide a valid framework to 'learn' the optimal cyber-attack response strategies for a large number of possible cyber-attacks. While learning in real time during a cyber-attack is dangerous, the use of digital twins for nuclear power plants that enables the simulation of a wide variety of cyber-attacks for as many instances as needed to train the RL agent makes the pursuit of such an idea completely realistic.

In this paper, we explore the use of reinforcement learning (RL), specifically the multi-agent Q-learning algorithm to compute the optimal cyber-attack response strategy. The interaction between the attacker and the defender is modelled as a Markov game solved using the concept of Stackelberg equilibrium. We implement a case study using the digital feedwater control system (DFWCS) of a PWR.

2. REINFORCEMENT LEARNING BASED APPROACH

2.1. Reinforcement Learning

In reinforcement learning, an agent interacts with or acts upon an environment or the system, sequentially at every time step, and uses the resultant feedback to update its course of action in order to achieve a long-term goal [9]. The feedback (not to be confused with the feedback in the context of control systems) received after each action is called the reward and the long-term goal is to maximize the cumulative reward [9]. For the scope of this paper, we assume that the agent can observe the system completely and the agent follows stationary behavior. Formally a reinforcement learning problem can be defined by the tuple $(S, A, \pi, P, R, \boldsymbol{\gamma})$ as follows [9]:

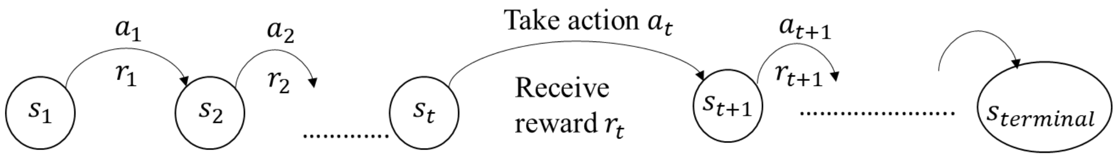
- At time step t, the environment is in a state $s_t \in S$, where S represents the set of all possible states of the environment. The terms environment and system will be used interchangeably.
- A is the action space of the agent and the set possible actions may depend on the current system state.
- At each state s, the probabilities of taking different actions i.e., the manner in which an agent behaves is defined by the policy π, which is a mapping from S to the probabilities of taking different actions. The probability that the agent takes an action a_t at state s_t under policy π is denoted by $\pi(a_t|s_t)$. For stationary behavior, $\pi(a_t|s_t) = \pi(a|s)\ \forall\ t$.
- $P: S \times A \times S \to [0,1]$ is the state transition probability mapping.
- $R: S \times A \times S \to \mathbb{R}$ is the reward function. At any timestep t, if the environment is in state s_t, the agent takes action a_t and the environment transitions to state s_{t+1}, the agent receives an immediate reward $r_t = R(s_t, a_t, s_{t+1})$. The reward r_t is a real number.
- At each time step t, the agent observes the state of the environment s_t, implements the action $a_t \in A$ on the environment, receives the immediate reward r_t and the environment transitions to a new state s_{t+1}, with a probability of transition $p(s_{t+1}|s_t, a_t)$ under a Markovian assumption.
- The discounted cumulative reward obtained by the agent over the course of time is $G_t = \sum_{j=0}^{\infty} \gamma^j r_{t+j}$, where r_{t+j} is the reward received j time steps after t, and $\gamma \in [0,1]$ is the discount factor that represents the weight assigned to future rewards. The agent's objective is to maximize the expected cumulative reward $\mathbb{E}_\pi[\sum_{j=0}^{\infty} \gamma^j r_{t+j}]$.
- A value function $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$ is defined for every state s, that represents the expected cumulative reward obtained starting from state s if the policy π is followed and t is any time step.
- Additionally, an action-value function, also known as Q-value, $q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$ is defined for every state-action pair (s, a) representing the expected cumulative

reward if action a is taken at state s, and then the policy π is followed subsequently from any time step t. Equation (1) presents the Bellman equation for the Q-value function [9].

$$Q(s_t, a_t) = \sum_{s_{t+1}\in S}\left(p(s_{t+1}|s_t, a_t) \times \left[R(s_t, a_t, s_{t+1}) + \gamma \times \sum_{a_{t+1}\in A}(\pi(a_{t+1}|s_{t+1}) \times Q(s_{t+1}, a_{t+1}))\right]\right) \quad (1)$$

One of the most popular algorithms used to solve a reinforcement learning problem is the Q-learning algorithm. Q-learning is a Monte Carlo based method used in combination with a dynamic programming based update for learning from experience [9]. It is implicit that in an optimal policy, at every state the agent takes an action that maximizes the Q-value. In Q-learning, these Q-values are "learned" by the agent through interacting with the environment over many episodes. Q-values are initialized randomly at the beginning of the learning process. Each episode consists of the agent starting out from an initial state, taking actions, transitioning into subsequent states until a terminal state is reached. The Q-values are updated at each state encountered in every episode. **Figure 1** represents an episode in Q-learning.

Figure 1. An episode in Q-learning.



During the learning process, at each state of every episode the agent's action is selected using a policy based on the Q-values learned from previous episodes. In a greedy policy, the action that maximizes the Q-value is selected. For example, in an episode i, the agent's action a_t at system state s_t is chosen according to equation (2), where Q_{i-1} represents the Q-values learned by the agent upto the episode $i-1$.

$$a_t = \arg\max_a Q_{i-1}(s_t, a) \quad (2)$$

Alternatively, ε-greedy policy, in which a random action is chosen with a probability ε to allow for exploration can be used to choose the agent's action. This allows the agent to explore actions that are different from those dictated by previous experience. The subsequent state s_{t+1} in episode i given the current state s_t and action a_t is sampled using the state transition probabilities. If a_{t+1} is the action taken by the agent in the state s_{t+1}, the Q-value update Bellman equation presented in equation (1) then transforms into equation (3), for the sample $(s_t, a_t, s_{t+1}, a_{t+1})$ in episode i.

$$Q_i(s_t, a_t) = R(s_t, a_t, s_{t+1}) + \gamma \times Q_{i-1}(s_{t+1}, a_{t+1}) \quad (3)$$

If the agent is behaving in an optimal manner, the action a_{t+1} at state s_{t+1} is chosen such that, $Q_{i-1}(s_{t+1}, a_{t+1})$ is maximum i.e., the agent chooses the optimal action based on the Q-values learned until episode $i-1$. Using this reasoning, the Q-value of the state-action pair (s_t, a_t) is updated using equation (4), a dynamic programming based equation.

$$Q_i(s_t, a_t) \leftarrow R(s_t, a_t, s_{t+1}) + \gamma \max_a Q_{i-1}(s_{t+1}, a) \quad (4)$$

It can be observed that in equation (4), the Q-value of the state action pair (s_t, a_t) in episode i is updated completely based on the current sample $(s_t, a_t, s_{t+1}, a_{t+1})$, and $Q_{i-1}(s_t, a_t)$ learned from the previous experience is completely discarded. Equation (5) presents the Q-value update equation, which combines the Q-values learned previously, with the current updates using a learning parameter $\alpha \in (0,1]$ [9]. It can be noticed that, when convergence to optimal policy is achieved, equations (4) and (5) will become identical.

$$Q_i(s_t, a_t) \leftarrow (1-\alpha) \times Q_{i-1}(s_t, a_t) + \alpha \times [R(s_t, a_t, s_{t+1}) + \gamma \max_a Q_{i-1}(s_{t+1}, a)] \quad (5)$$

It is implicit that in practical implementation only one Q-matrix is used and updated iteratively in every episode for efficient computation. As in the case of cyber-attacks, many realistic scenarios involve multiple agents acting on the environment simultaneously and trying to maximize their individual rewards. Since the environment is affected by the actions of all the agents, the corresponding reward received by every individual agent is dependent on actions of all other agents. Hence it is implicit that it is not ideal if each agent unilaterally tries to optimize its own reward as a function of its individual actions. Littman [10] introduced Markov games as a reinforcement learning framework in the context of multiple agents working either co-operatively or competitively. We model the interaction between the attacker and the defender as a Markov game with infinite time horizon and use multi-agent reinforcement learning to compute an optimal response strategy.

2.2. Multi-Agent Framework using Markov Games

A player in the context of game theory is equivalent to an agent in reinforcement learning. The terms agent and player will be used interchangeably. In the context of this paper, only two players, the defender and the attacker are considered. As presented by Littman [10] and Zhao et al.[5], a Markov game of two players (defender and attacker) is defined by the tuple $(\boldsymbol{S}, \{A, D\}, \{\pi^{\boldsymbol{a}}, \pi^{\boldsymbol{d}}\}, P, \{R^{\boldsymbol{a}}, R^{\boldsymbol{d}}\}, \boldsymbol{\gamma})$**,** where:

- $S = \{s_1, s_2, s_3 \ldots\}$ is the set of all possible states of the system.
- $D = \{d_1, d_2, d_3 \ldots\}$ is the defender's action space and $A = \{a_1, a_2, a_3 \ldots\}$ is the attacker's action space. It is implicit that the set of feasible defender and attacker actions depends on the system state.
- $\pi^{\boldsymbol{d}}$ and $\pi^{\boldsymbol{a}}$ are the action policies of the defender and the attacker respectively. The quantity $\pi^{\boldsymbol{d}}(d_i|s_j)$ is the probability that the defender takes action i at system state s_j.
- $P: S \times D \times A \times S \to [0, 1]$ is the state transition probability mapping, where $p(s_j|s_i, d_k, a_l)$ is the probability that the system transitions to state s_j when the defender takes action d_k and the attacker takes action a_l when the system state is s_i.
- $R^{\boldsymbol{d}}: S \times D \times A \times S \to \mathbb{R}$ is the reward function of the defender and $R^{\boldsymbol{a}}: S \times D \times A \times S \to \mathbb{R}$ is the attacker's reward function. The quantities $r^{\boldsymbol{d}} = R^{\boldsymbol{d}}(s_i, d_k, a_l, s_j)$ and $r^{\boldsymbol{a}} = R^{\boldsymbol{a}}(s_i, d_k, a_l, s_j)$ are the immediate rewards received by the defender and the attacker respectively, when the system is state s_i, the defender takes action d_k, the attacker takes action a_l and the system transitions to state s_j. Similarly,
- $\boldsymbol{\gamma}$ is the discount factor.
- Each player tries to optimize his or her expected cumulative rewards $\mathbb{E}\left[\sum_{j=0}^{\infty} \gamma^j r_{t+j}^d\right]$ and $\mathbb{E}\left[\sum_{j=0}^{\infty} \gamma^j r_{t+j}^a\right]$.

In addition, the action-value functions of the defender $Q^{\boldsymbol{d}}$ and the attacker $Q^{\boldsymbol{a}}$ are defined as well. It is implicit that for every state s, the state transitions and the corresponding rewards are dependent on both attacker and defender actions. Hence, $Q^{\boldsymbol{d}}$ and $Q^{\boldsymbol{a}}$ should be defined as functions of defender and attacker action pairs. Table 1 presents example Q-values for an arbitrary state in a two-player game. As shown, there are three possible defender actions and attacker actions, and the Q-values are defined for the defender as well as the attacker for a total of nine action pairs. In the normal-form representation of a two player Markov game [8], $Q^{\boldsymbol{d}}$ and $Q^{\boldsymbol{a}}$ are matrices for every state s. In the convention used in this paper, (i, j) element of the Q-value matrices for both the defender and attacker, denotes the Q-value for defender action i and attacker action j at a particular state s.

Table 1. Example Q-values in one state in multi-agent setting.

		$Q^{\boldsymbol{d}}$ – defender Q-values				Q^a – Attacker Q-values		
Attacker Actions		1	2	3		1	2	3
Defender Actions	1	3.72	-4.27	3.5		-5.75	-4.8	-3.67
	2	-7.5	-2.25	-2.75		4.52	-3.61	-2.50
	3	-2.94	-7.6	1.67		-3	-2.54	4.57

The Q-learning algorithm presented in section 2.1 for a single agent can be expanded to the multi-agent case. Equations (6) and (7) present the Q-value update equations for the defender and the attacker respectively in episode i in a multi-agent reinforcement learning setting as presented by Hu and Wellman [11] and Kononen [12].

$$Q_i^d(s_t, d_t, a_t) = (1-\alpha) \times Q_{i-1}^d(s_t, d_t, a_t) + \alpha \times [r_t^d + \gamma \times Optimal\,(Q_{i-1}^d(s_{t+1}, d_{t+1}, a_{t+1}))] \tag{6}$$

$$Q_i^a(s_t, d_t, a_t) = (1-\alpha) \times Q_{i-1}^a(s_t, d_t, a_t) + \alpha \times [r_t^a + \gamma \times Optimal\,(Q_{i-1}^a(s_{t+1}, d_{t+1}, a_{t+1}))] \tag{7}$$

It is clear that defender and the attacker are learning simultaneously. While Hu and Wellman [11] used Nash equilibrium to compute the optimal Q-values, Kononen [12] presented the asymmetric learning case with Stackelberg equilibrium. It is implicit that, action selection during the episodes is performed using the optimal Q values. In this paper we use the Stackelberg equilibrium for multi-agent Q-learning.

2.3. Stackelberg Equilibrium

In the Stackelberg equilibrium concept for two player games i.e., in a two player Stackelberg game one of the players acts as the leader and the other is a follower [8], [12]. In contrast to Nash equilibrium, where the two players have symmetric roles, a Stackelberg equilibrium is asymmetric and hierarchical in nature, where the leader can enforce their strategy (action) while the follower responds to it in a rational manner, i.e., in a manner that optimizes their reward. The leader-follower approach of the Stackelberg equilibrium concept is appropriate in the context of cyber security, where the defender as the leader enforces a security strategy, in response to which an attacker as the follower observes the implemented strategy and launches an offensive action that optimizes their reward [7], [13], [14]. The Stackelberg equilibrium solution for the two-player game with the defender as the leader, and given Q^d and Q^a matrices at a state s can be computed as follows as presented by Kononen [12]. The procedure involves a two-step backward calculation. In the first step the follower's optimal response to every one of leader's actions is identified. In the second step the leader's action that generates the optimal reward given that the follower responds with the actions identified in the first step is obtained.

- Identify the attacker action that generates the maximum reward (in this case Q-value) for every possible defender action, as shown in equation (8).

  $$a_{\mathcal{S}}(d_i) = \underset{a, d_i \in D}{\arg\max}\, Q^a(s, d_i, a) \tag{8}$$

  where D is the defender's (leader's) action space,
  $d_i \in D$ is the defender's (leader's) action, and
  $a_{\mathcal{S}}(d_i)$ is the optimal response by the attacker (follower) for defender's (leader's) action d_i.

  For the example presented in Table 1, it can be observed that $a_{\mathcal{S}}(d = 1) = 3$, $a_{\mathcal{S}}(d = 2) = 1$ and $a_{\mathcal{S}}(d = 3) = 3$ i.e., when the defender's action is 1, the optimal attacker action is 3, when the defender's action is 2, the optimal attacker action is 1, and when the defender's action is 3, the optimal attacker action is 3.

- Identify the defender action that generates the maximum defender reward, for the attacker actions calculated above, as shown in equation (9).

  $$d_{\mathcal{S}} = \underset{d \in D}{\arg\max}\, Q^d(s, d, a_{\mathcal{S}}(d)) \tag{9}$$

  where D is the defender's (leader's) action space,
  $a_{\mathcal{S}}(d)$ is the optimal response by the attacker (follower) for defender action d, and
  $d_{\mathcal{S}}$ is the optimal defender action.

  The pair $(d_{\mathcal{S}}, a_{\mathcal{S}}(d_{\mathcal{S}}))$ is the pure strategy Stackelberg equilibrium solution, where each player has one definite action/strategy to implement. Alternatively, a mixed strategy equilibrium solution provides a probability distribution on the players' action spaces and will be studied in further research. For the example presented in Table 1, when the defender-attacker action pair is (1,3), the defender reward is 3.5, when the defender-attacker action pair is (2,1), the defender reward is -7.5 and when the defender-attacker action pair is (3,3), the defender reward is 1.67. It can be observed that out of these values, the maximum reward is 3.5. Therefore, the Stackelberg equilibrium strategy in this example with the defender as the leader is $d = 1$ and $a = 3$. So, according to the computed pure strategy Stackelberg equilibrium solution, the defender (leader) enforces action -1 out of actions 1, 2 and 3 initially, to which the attacker (follower) responds with action – 3.

It can be observed from the above example that, for the defender (leader) to compute and enforce their strategy, they should have knowledge of the attacker's (follower's) Q-values to estimate the attacker's optimal response for every one of their actions. This is the result of the assumption that the defender (leader) is aware of the attacker's (follower's) rewards. We also assume that the attacker (follower) can always observe the strategy enforced by the defender (leader). It is realistic to expect that the defender is unaware of the attacker's rewards and attacker (follower) cannot completely observe the defender's strategy. Such cases will be studied in future research.

In the case study presented in this paper we use Stackelberg equilibrium with the defender as the leader in the multi-agent Q-learning and compare the results with the case where the attacker is the leader i.e., the case where the attacker implements their offensive action, and the defender responds with an optimal action. It is our opinion that the case in which the attacker acts as the leader i.e., the case where the attacker (leader) is aware of the defender's (follower's) rewards, the attacker (leader) can enforce a strategy to which the defender responds optimally, and the defender (follower) can completely observe the attacker's (leader's) actions is realistic. For example, it can be the case of an insider attack, in which a disgruntled or a radicalized employee can lay traps beforehand and lead the game.

3. CASE STUDY

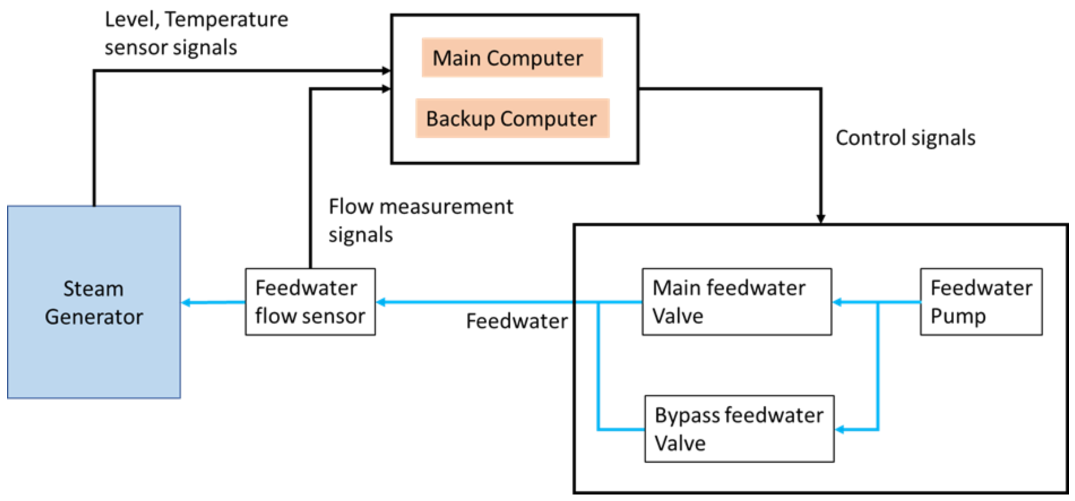
In this case study we implement the multi-agent Q-learning discussed above to compute the optimal cyber-attack defense strategy for the digital feedwater control system (DFWCS) of a pressurized water reactor (PWR). The system states, the possible attacker and defender actions as presented by Zhao et al. [5], [6] are discussed below.

3.1. The System and System States

Figure 2 depicts a schematic of the digital feedwater control system of a pressurized water reactor as presented in the NUREG CR-6942 [15]. The components main computer (MC) and backup computer (BC) act as controllers, whereas the components feedwater pump, main feedwater valve and bypass feedwater valve are mechanical components and act as actuators that receive control signals from the MC and BC. Additionally, there are three sensors, the feedwater flow sensor, the steam generator level sensor and a temperature sensor that are part of the feedback loop of the control system and send signals to the MC and BC. The steam generator acts as a tank. It is important to observe here that the pump, the valves and the steam generator are mechanical components and cannot be subjected to cyber-attacks, whereas the computers and the sensors are digital components and can be compromised. So, as presented by Zhao et al. [5], [6], in this analysis only the digital components are considered, and the sensors are grouped together as one set of digital components for simplification.

The main computer is the primary controller of the system and is in use under normal operating conditions. As the name suggests the backup computer is backup to the main computer and is in standby when the main computer is in use. The backup computer is used only when the main computer is out of operation either due to a failure or a cyber-attack. In this analysis, component failures are not explicitly considered. The system is controlled manually by the PWR operator when the backup computer is out of operation as well, either due to a failure or a cyber-attack. So, a control mode variable with two states automatic or manual is considered to represent the mode of control. The control mode is automatic when either the MC or the BC is used, and manual when the operator directly controls the system. Once the control is switched from MC to BC, it cannot be reverted, and similarly, once the control is switched from BC to manual mode, it cannot be reverted within the session or episode. It is assumed that the control can be transferred only from the MC to the BC, and then from the BC to manual control in that order, and any other transfers such as from the BC to the MC or from the MC to manual control are not possible. Additionally, an approximate model, that can be used as a substitute for the sensors as a countermeasure against cyber-attacks is considered [5], [6]. The approximate model can predict the system physical variables such as flowrate, temperature and level to a certain degree of accuracy. We assume that the approximate model cannot be compromised. The operator can decide to use the approximate model instead of sensors in both automatic and manual control modes, and this switch cannot be reverted as well.

Figure 2. Schematic of a PWR DFWCS



It is assumed that, even though the system can be controlled using either the approximate model or manual control or both, it is not as safe as using sensors and computers. In addition to digital components, the reactor core with two states either normal or damaged is considered as well. The episode in Q-learning ends once the reactor core is damaged or when we reach a state when the system cannot be compromised anymore. **Table 2** lists the components and modes along with their corresponding states as presented in [5], [6].

Table 2. List of components/modes and their states.

Component	States
1. Sensors 2. Main Computer 3. Backup computer	1. Normal and in use. (1) 2. Normal and not in use. (2, NU) 3. Compromised and in use. (3) 4. Compromised and not in use. (4, NU)
4. Control Mode	1. Automatic (1) 2. Manual (2)
5. Sensing Mode	1. Sensors are used. (1) 2. Approximate model is used. (2)
6. Reactor core	1. Normal (1) 2. Damaged. (2)

As seen from the Table 2 a total of $4^3 \times 2 \times 2 \times 2 = 512$ states are possible. However, it is implicit that once the reactor core is damaged, the states of the remaining components are of no consequence. Similarly, the states in which both MC and BC are normal and in use or the state in which both the sensors and approximate model are used, are physically impossible. Zhao et al [5], [6] reduced the total number of possible system states to just 16, as presented in the Table 3. The physical system state is represented through a vector with 6 components, [State of sensors, MC state, BC state, Control mode, Sensing Mode, Reactor core state].

Table 3. System States.

	Vector	**Description**	**SAFE or UNSAFE**
1	[1 1 2 1 1 1]	Automatic mode with normal MC and normal sensors	SAFE
2	[3 1 2 1 1 1]	Automatic mode with normal MC, compromised sensors	UNSAFE
3	[1 3 2 1 1 1]	Automatic mode with compromised MC, normal sensors	UNSAFE

4	[1 NU 1 1 1 1]	Automatic mode with normal BC, and normal sensors	SAFE
5	[3 3 2 1 1 1]	Automatic mode with compromised MC and sensors	UNSAFE
6	[3 NU 1 1 1 1]	Automatic mode with normal BC and compromised sensors	UNSAFE
7	[NU 1 2 1 2 1]	Automatic mode with normal MC and approximate model	SAFE
8	[1 NU 3 1 1 1]	Automatic mode with compromised BC and normal sensors	UNSAFE
9	[NU 3 2 1 2 1]	Automatic mode with compromised MC and approximate model	UNSAFE
10	[3 NU 3 1 1 1]	Automatic mode with compromised BC and sensors	UNSAFE
11	[1 NU NU 2 1 1]	Manual with normal sensors	SAFE
12	[NU NU 1 1 2 1]	Automatic mode with normal BC and approximate model	SAFE
13	[3 NU NU 2 1 1]	Manual with compromised sensors	UNSAFE
14	[NU NU 3 1 2 1]	Automatic mode with compromised BC and approximate model	UNSAFE
15	[NU NU NU 2 2 1]	Manual with approximate model.	SAFE (terminal state)
16	[X X X X X 2]	Core damaged - END	UNSAFE (terminal state)
NU – Not in use.		X – the component state is of no consequence.	

3.2. Attacker and Defender Actions

Table 4 presents the list of all possible attacker and defender actions as presented by Zhao et al. [5], [6]. The attacker can either compromise the sensors, or the MC or the BC, whereas the defender can switch from sensors to approximate model, or switch control from MC to BC or from BC to manual mode. Additionally, we consider the case in which the attacker and the defender do nothing i.e., take no action. It is assumed that the defender and the attacker can take only one action at a time. For example, the attacker cannot compromise both the MC and the sensors in one action. Additionally, certain actions are impossible in some system states. Consider the system state 1, i.e., the system is in automatic control mode with normal MC and sensors. As discussed in section 3.1, the defender cannot switch directly to manual control. So, defender action 3 is invalid. Similarly, it is assumed that a component not in use cannot be compromised and hence attacker action 3 is invalid as well. **Table 5** lists the state dependent attacker and defender action spaces as presented by Zhao et al. [5], [6]. It is assumed that both the operator and the defender are always successful in implementing their actions.

Table 4. List of possible attacker and defender actions.

Attacker Actions	Defender Actions
1 – Compromise the sensors	1 – Switch from sensors to approximate model.
2 – Compromise the main computer	2 – Switch control from MC to BC.
3 – Compromise the backup computer	3 – Switch control from BC to manual mode.
4 – Do Nothing.	4 – Do nothing.

Table 5. State dependent attacker and defender action spaces.

Physical system state	Attacker action space	Defender action space	Physical system state	Attacker action space	Defender action space
1	1, 2, 4	1, 2, 4	9	4	2, 4
2	2, 4	1, 2, 4	10	4	1, 3, 4
3	1, 4	1, 2, 4	11	1, 4	1, 4
4	1, 3, 4	1, 3, 4	12	3, 4	3, 4
5	4	1, 2, 4	13	4	1, 4

6	3, 4	1, 3, 4	14	4	3, 4
7	2, 4	2, 4	15	4	4
8	1, 4	1, 3, 4	16	4	4

3.3. State Transitions and Reward Functions

As presented by Zhao et al. [6], we consider that when the system is in an unsafe state i.e., when at least one component is compromised, we assume that the feedwater control system has failed and as result there is a probability of 10^{-3} that the system transitions into the core damaged state (state 16). This is equal to the probability that the auxiliary feedwater system fails given the feedwater system has already failed. The distinction between the states where the attacker has compromised the sensors or the controller or both will be studied in further research.

Similarly, when the system is in its design state i.e., in state 1 with the main computer and sensors in normal state and in use, with the backup computer on standby and the system in automatic control mode using sensors, we assume that the probability of transition to core damage is 10^{-5}. This is equal to the probability of DFWCS failure and auxiliary system failure. It is implicit that use of approximate models instead of sensors, and use of manual control instead of the computers, is prone to errors. As a result, we assume that the transition probability to core damage increases to 10^{-4} in state 15 and use interpolation to quantify the transition probabilities for the states in between, as listed in Table 6. For example, in state 4, backup computer is used for control along with normal sensors, and the main computer is out of operation. As a result, we assume that the probability of transition to terminal state is 3.34×10^{-5}, which is higher compared to the one in state 1.

Table 6. Transition probabilities to terminal state

States	Probability of transition to core damage state.
16 – terminal.	(Already in core damage state)
5, 10 – Both controller and sensors are compromised and in use. 2, 3, 6, 8, 9, 13, 14 – one component is compromised.	10^{-3}
1 – initial state.	10^{-5}
4 – Automatic mode with normal BC and sensors. 7 – Automatic mode with normal MC and approximate model.	3.34×10^{-5}
11 – Manual control with normal sensors. 12 – Automatic mode with normal BC and approximate model.	6.67×10^{-5}
15 - Manual mode with approximate model	10^{-4}

The objective of a reinforcement learning problem is to compute the action policies that optimize the expected cumulative discounted reward. Hence, the selection of reward functions must be given utmost importance. In this analysis we divide the reward function into two components, as presented in equation (10).

$$r = r_{action} + r_{transition} \tag{10}$$

The term r_{action} represents the cost incurred by the agent i.e., the operator or the attacker for taking an action. We assume that the cost of taking any action i.e., any one of actions 1, 2 and 3 in Table 4 is equal to \$ 10,000 for the attacker and the defender incurs no cost to take any action, as discussed by Zhao et al. [6].

Additionally, both agents get an immediate reward $r_{transition}$ whenever there is a state transition due to their actions. This immediate reward is used as an incentive to encourage the agents' behavior in the correct direction [16]. When there is a transition to the terminal state, the operator receives a negative reward that is 10 billion dollars in worth [6] that represents the consequences of core damage, and we assume that the attacker receives a positive reward of identical value.

The benchmark system presented in [15] has fault tolerance capabilities. For example, the main computer and the backup computer verify and validate the readings from the sensors. Hence it is assumed that the states in which two components are compromised (states 5 – sensors and main computer are compromised and 10 – sensors and backup computer are compromised) are more advantageous to the attacker compared to states in which only one component is compromised. So, it is assumed that the attacker receives an immediate positive reward of $10,000 when they compromise a single component i.e., when there is a transition to states 2, 3, 6, 8, 9, 13 and 14 due to their actions and $20,000 when there is a transition to the states 5 and 10 i.e., states in which two components are compromised. The defender receives a negative reward of identical value when these transitions occur irrespective of their actions. Similarly, when there is a transition to a safe state i.e., 4, 7, 11 and 12, the defender receives a positive reward, and the attacker receives a negative reward of worth $10,000, and when there is a transition to state 15, a state in which no component can be compromised, the reward is equal to $20,000. The attacker and defender do not receive any immediate reward and do not incur any cost of action when they take no action and when there is no state transition. In this case study, the immediate rewards for transitions are assumed to be in the same order as the cost incurred by the attacker for taking an action. Further research is needed to accurately quantify the rewards to represent more realistic situations.

It can be observed that the defender acting proactively and switching to manual control mode and approximate model, which cannot be compromised by the attacker is safe in the immediate run in the context of cyber security. However, these modes of operation are error prone and unsafe in the long run, and the increased probabilities of transition to core damage as presented in Table 6 represents the penalties levied on the defender for such actions.

3.4. Results

A total of 500000 episodes were used for learning. The attacker and defender actions were selected randomly for the first 100000 episodes to allow for exploration and an ε-greedy policy based action selection was used for the remaining episodes with $\varepsilon = 0.1$. The discount factor γ was set to 0.8, and the learning rate α was set to 0.4. A sensitivity analysis of these parameters will be performed in future research. It is important to remember that Stackelberg games are hierarchical in nature, in which the leader takes an action, and the follower responds optimally. Table 7 presents the pure strategy Stackelberg equilibrium solutions i.e., the optimal defender and attacker actions at every state for the case in which defender leads as well as the case in which the attacker leads.

The computed pure strategies appear to be logical. For example, in state 1 where the system is operating in automatic control mode with normal sensors, and normal main computer, as the leader, the defender's optimal strategy is to switch to the approximate model (action - 1). It is assumed that the defender takes this action pre-emptively when an intrusion is discovered in the network i.e., when a cyber-attack may be imminent, which is the starting point of this research and not randomly, out of extreme caution even when there is no indication of cyber-attack. The attacker's response to this strategy is to compromise the main computer. These actions will result in a transition to state 9, where the system is in automatic control mode with compromised main computer and approximate model. The defender's optimal strategy in state 9 is to switch the control from main computer to backup computer (action 2 as shown in Table 7 for state 9).

Table 7. Stackelberg Equilibrium Solutions for Different States

	Defender as the Leader		Attacker as the leader	
State	Defender's action	Attacker's action	Defender's action	Attacker's action

1	1	2	2	2
2	1	2	2	2
3	2	1	2	1
4	3	1	3	3
5	1	4	2	4
6	1	4	1	3
7	4	2	2	4
8	3	1	1	1
9	2	4	2	4
10	3	4	1	4
11	4	1	4	1
12	3	4	3	4
13	1	4	1	4
14	3	4	3	4
15	4	4	4	4
16	4	4	4	4

Similarly, when the attacker is the leader, the optimal action in state 1 is to compromise the main computer (action – 2), to which the defender's response is to use the backup computer (action – 2). These actions result in a transition to state 4, i.e., the system is in automatic control mode with normal backup computer and normal sensors. In state 4, it can be observed that the attacker's strategy is compromise the backup computer and the defender's response is to switch to manual control.

It can be observed that as the leader, the defender is initially prioritizing the use of approximate model which cannot be subjected to cyber-attacks. It is not possible to compromise both the main computer and the backup computer at the same time. So, when the defender is switching to the approximate model, it is impossible to reach the states in which two components are compromised at the same time i.e., states 5 and 10. As mentioned in section 3.3, transitioning to states 5 and 10 is more advantageous to the attacker and will result in a large negative penalty to the defender. Similarly, when the attacker is the leader, the priority is on compromising the main computer initially, which will lead the defender to switch to backup computer, thereby providing the attacker with additional opportunities to compromise multiple components.

4. CONCLUSIONS

Cyber-attacks on industrial control systems, and the corresponding interactions between the attackers and the operators have been modeled using game theory, as competitive Markov Decision Problems (MDPs). In this paper, we presented the use of a multi-agent reinforcement learning approach, specifically the multi-agent q-learning algorithm with Stackelberg equilibrium to solve those MDPs and to compute the defender's optimal response strategy against cyber-attacks. The results can be used to support an operator when the system is facing a cyber-attack. It is worth noting that the reinforcement learning approach used in this research belongs to the model-free paradigm. This means that the attacker and defender strategies are learned through experience. This is in contrast to model-based approaches where the MDP model is first explicitly constructed and the value functions of individual states, and the action-value functions, and the resultant attacker and defender strategies are explicitly computed using planning based methods.
As a case study, the method was tested using a simplified version of the DFWCS of a PWR, and the pure strategies of the attacker and the defender were computed for two separate cases. In the first case the defender is the leader and in the second case, the attacker is the leader. The strategies for the defender for these two cases were compared and the obtained pure strategies were found to be logical. In future research, the proposed method will be extended to finite-horizon games, mixed Stackelberg strategies and the use of a physics based dynamic model of the system.

Acknowledgements

This research is being performed using funding received from the DOE Office of Nuclear Energy's Nuclear Energy University Program.

References

[1] Falliere, Nicolas, Liam O. Murchu, and Eric Chien., "W32.Stuxnet Dossier." Symantec Corp, Feb. 2011. Accessed: May 04, 2022. [Online]. Available: https://pax0r.com/hh/stuxnet/Symantec-Stuxnet-Update-Feb-2011.pdf

[2] S. Corbet and J. W. Goodell, "The reputational contagion effects of ransomware attacks," *Finance Res. Lett.*, p. 102715, Feb. 2022, doi: 10.1016/j.frl.2022.102715.

[3] "Triton is the world's most murderous malware, and it's spreading," *MIT Technology Review*. https://www.technologyreview.com/2019/03/05/103328/cybersecurity-critical-infrastructure-triton-malware/ (accessed Dec. 10, 2021).

[4] A. Varuttamaseni, R. A. Bari, and R. Youngblood, "Construction of a Cyber Attack Model for Nuclear Power Plants," Idaho National Lab. (INL), Idaho Falls, ID (United States), INL/CON-17-41862, May 2017. Accessed: May 04, 2022. [Online]. Available: https://www.osti.gov/biblio/1378337

[5] Y. Zhao, L. Huang, C. Smidts, and Q. Zhu, "A game theoretic approach for responding to cyber-attacks on nuclear power plants," *in proceedings of 11th Nuclear Power Instrumentation and Control - Human Machine Interface Technologies conference,* February 9-14, 2019, Orlando, FL.

[6] Y. Zhao, L. Huang, C. Smidts, and Q. Zhu, "Finite-horizon semi-Markov game for time-sensitive attack response and probabilistic risk assessment in nuclear power plants," *Reliab. Eng. Syst. Saf.*, vol. 201, p. 106878, Sep. 2020, doi: 10.1016/j.ress.2020.106878.

[7] L. T. Maccarone and D. G. Cole, "Bayesian games for the cybersecurity of nuclear power plants," *Int. J. Crit. Infrastruct. Prot.*, vol. 37, p. 100493, Jul. 2022, doi: 10.1016/j.ijcip.2021.100493.

[8] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, 2nd edition. Philadelphia: Society for Industrial and Applied Mathematics, 1999.

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, Second edition. Cambridge, Massachusetts: The MIT Press, 2018.

[10] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*, W. W. Cohen and H. Hirsh, Eds. San Francisco (CA): Morgan Kaufmann, 1994, pp. 157–163. doi: 10.1016/B978-1-55860-335-6.50027-1.

[11] J. Hu and M. P. Wellman, "Nash q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, no. null, pp. 1039–1069, Dec. 2003.

[12] V. Kononen, "Asymmetric multiagent reinforcement learning," in *IEEE/WIC International Conference on Intelligent Agent Technology, 2003. IAT 2003.*, Oct. 2003, pp. 336–342. doi: 10.1109/IAT.2003.1241094.

[13] A. Sinha, F. Fang, B. An, C. Kiekintveld, and M. Tambe, "Stackelberg Security Games: Looking Beyond a Decade of Success," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, Jul. 2018, pp. 5494–5501. doi: 10.24963/ijcai.2018/775.

[14] D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe, "Stackelberg vs. Nash in Security Games: An Extended Investigation of Interchangeability, Equivalence, and Uniqueness," p. 31.

[15] T. Aldemir *et al.*, "NUREG/CR-6942: Dynamic Reliability Modeling of Digital Instrumentation and Control Systems for Nuclear Reactor Probabilistic Risk Assessments," 2007.

[16] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, 1999, vol. 99, pp. 278–287.