International Journal of Interdisciplinary and Multidisciplinary Research (IJIMR)

ISSN 2456–4567

Industrial Control Network Cyber Security Orchestration Using Reinforcement Learning

1st Author Muhammad Afzal Nazim 2nd Author Bakhtawar Sarfraz
Department of Computer Science
Government College University Faisalabad, Layyah Campus

Abstract
To minimize the effect on the network as much as possible, the nodes should coordinate their responses across a wide number of nodes based on incorrect indications of penetration. Most sophisticated assaults may take months to be developed and show no visible signs of development before being carried out. The sequential choice issue is difficult to solve due to the vast observation and action areas and the lengthy time horizon involved, as illustrated in Table 1. In this paper, we propose approaches for scaling deep reinforcement learning in order to address the challenge of orchestration of industrial control networks for cyber security. This neural architecture utilizes the principles of attention and size complexity and is only reliant on the size of the protected network. An early exploration training program is implemented to relieve the difficulties early exploration brings. The findings of the trials suggest that the suggested techniques have a better chance of converging on effective policies than baseline methods, as far as learning sample complexity and complexity of policies are concerned.

Introduction

Increasingly, hackers attack ICS-linked computer networks. As a result, they are tough to track down and stop. Intrusion detection systems monitor networks and notify security orchestrators to possible intrusions because of the large number of false alarms. It's clear that many safety warnings are not heeded due of the significant number of false alarms. Attackers who employ advanced persistent threats (APTs) exploit this to their advantage by stealthily spreading malicious code throughout a target network, lying dormant for extended periods of time before making an attempt. The loss of industrial control systems (ICS) in which control functions are transferred, resulting in loss of function, physical damage to equipment, and even the loss of human life may be a consequence of a given attack on the ICS. as a result of the typical network APT attacks, attackers may cause an attack on a critical system, physical damage to equipment, and even loss of life.Because they lack the required resources, human security teams are unable to react to every possible intrusion warning and efficiently neutralize all threats. This study demonstrates that it is possible to build an automated cyber security orchestrator (ACSO) to help human analysts in industrial control systems (ICS) network security analyses by automatically examining and reducing network security threats.The first thing we need do in order to take on this challenge is to think of a discrete-time sequential decision-making problem model, which is then refined. The suggested model captures many of the difficulties related to the underlying choice problem even while it is agnostic to particular network architectures and communication patterns. We built a simulator that can provide a huge amount of trial data in a short amount of time to help with this.

Due to the fact that explicit models of APT attack dynamics and warning behaviors have not yet been established, many sequential decision techniques are inadequate to address this issue. in the absence of explicit models, reinforcement learning (RL) may be used to train policies for complicated tasks. An observation and action space that is very high-dimensional results as the number of nodes on the protected network is increased. Deep reinforcement learning is said to be problematic in learning discrete actions, because generalization across actions is difficult. In a similar line, stochastic learning sample generation has also been shown to be inefficient in large input spaces.

APT assaults are made difficult to detect so that network penetrations are hidden while the attack is ongoing. Many authors are in agreement that it is much more difficult to solve a partly observable issue than a completely visible one because of the hidden information involved. Organizations must be prepared to react rapidly to breaches, even if these incidents may continue for months. Time-steps with great precision are required to capture the decision-making process, which leads to lengthy time spans. Longer time horizons and restricted incentives have been found to substantially enhance sample complexity.

The techniques and training materials outlined in this article may be applied to many other areas. A unique attention-based architecture is presented that can learn over a wide range of network nodes, without the need for new factors to be addressed. While parameter complexity network (PCN) training speed is substantially increased, it is also demonstrated to reach a better performing solution compared to the baseline architecture. Under supervision, pretraining lowers the overall high-risk classifications and temporal difference loss, which reduces the possibility of running into exploratory issues.

The technique of solution suggested on the basis of neural network designs and training methodologies was compared to other methods, all of which included expert-developed policies. By implementing the tests, the suggested method is shown to be scalable, and thus autonomous network security agents could be developed via deep RL. These approaches may be utilized to expand the current approach, which is focused on scalability, to problems involving larger input spaces, output spaces, or greater time horizons utilizing the methodologies described.

Background
ReinforcementLearning

A sequential choice problem is structured like a diagram in which the states are represented by circular shapes that progress in a possibly stochastic manner. An agent performs activities that generate changes in state transition distributions and receives incentives in return. An awful lot of the time, the condition of the environment is unknown. Based on the equation o Z(o | s, a), agents in partly visible areas get noisy observations.

In sequential choice issues, the value V (s) = E t tr(st, at) for all states during the duration of the trajectory is maximized by designing an action sequence that optimizes for V (s) = E t tr(st, at) for all states throughout the trajectory. Recurrent trial and error with the environment using reinforcement learning techniques yields a policy: o:t, which maps a history of observations to actions as a consequence of recurrent trial and error. Activities are carried out in each experiment, in line with the present policy in force, until a terminal condition is achieved. In the end, the policy is modified to boost the projected system performance.

Initial Intrusion	Lateral Movement	Discovery	Privilege Escalation	Execution
Through social engineering, the attacker obtains control of a low-privilege node.	To guarantee persistence and visibility, the attacker takes control of other nodes.	The attacker looks for data on the network and the physical process that is being attacked.	The attacker collects credentials and compromises nodes in order to carry out the attack.	Malicious code is sent by the attacker, which causes data loss, process interruption, or equipment damage.

Table 1: APT Assault Evolution

In the MITRE ATT&CK paradigm, this depicts the usual progression of an APT attack at the tactical level. The procedure begins on the left with an initial incursion and continues for many months before being carried out.

Reinforcement learning using deep neural networks (or deep RL) is a type of reinforcement learning in which neural networks are used to represent the learnt policy. [14] Deep Q-Network (DQN) learning is a commonly used forecasting technique for given input history. It calculates Q(ot:t, a) as: Q(ot:T, a) = E r(st, A) + V (st+1) where Q(ot:t, a) represents the expected value Q(ot:t, a), which is defined as Q(ot:T, a) = E r(st, A) + V (st+1). the action with the greatest anticipated value is specified as an argmax (ot:t, a).

Often, massive quantities of data are required to effectively train Deep RL models. In direct relation to the complexity of the task and the size of the neural network, the number of trials needed to complete a task rises as the size of the neural network increases. Incremental reinforcement learning agents keep moving in new directions with each new trial. Roughly speaking, the probability of discovering a good route via random search is low, thus issues that need a very wide input space, output space, or temporal horizon require much more investigation.

Networked Industrial Control Systems

Industrial control systems are control networks used for monitoring and managing a coordinated physical process, such as an assembly line or a power plant. In order to gather sensor data and to give control instructions to the equipment itself, industrial equipment is normally outfitted with simple operational technology (operational technology(OT) computers such as programmable logic controllers [PLCs]) computer systems. In order to provide remote access and control, these PLCs are connected to information technology networks.

More interconnected than ever, industrial control systems (ICS) networks are at risk of cyber-attacks. While there is a higher degree of risk involved, it is necessary to have a strong network of contacts when a process involves components that are difficult to reach or are located in different locations all over the world. There has to be cooperation among distribution substations which are located in remote locations and may be found across vast geographical areas, for example. To try to address this issue, some networks use an architecture that has a low-privilege node placed on an accessible level and a process-critical node located on a more isolated subnet.

High-profile infrastructure assaults have in the recent past included sophisticated persistent threats (APTs). In its threat taxonomy, the Defense Science Board designates advanced persistent threats as the top two threat categories, which are often well-funded and are prepared to put in a considerable amount of work to reach their objective. To skirt around the network topology, attackers may inject malware at the start of the assault into a low-privilege node to compromise the target network. The APT will utilize this node for performing internal network reconnaissance, gaining control of other privileged nodes, and executing malicious acts against critical infrastructure until it has achieved the authority required to meet

its primary objective, which may include disrupting crucial services. The MITRE ATT&CK framework provides an example of an APT assault procedure that is based on the MITRE ATT&CK framework techniques.

Automated intrusion detection systems (AIDS) struggle to identify the presence of reconnaissance and lateral movement. During the assault campaign, the bulk of the time is spent on lateral movement. It may take months to finish. You can quickly carry out an assault if an attack has been planned. To fight against an APT assault, it is critical to identify and secure the compromised nodes during the staging stages.

The techniques used in many cyber security fields, such as virus identification and anomaly detection, have been explained. While many reinforcement learning techniques have been used in the field of cyber security, many varied methods have been applied. The article by Nguyen and Reddi has an in-depth review of recent research in this field, which includes a description of the research that has already been conducted. This research's focus is restricted to one thing: the kind of network the defender agent is protecting, the actions the defender agent has available, and the specific assault that is being defended against. Gupta and Yang provide an example of a method limited to filtering signals that spoof networked sensor readings, which is based on the example they provide.

Problem Formulation

Reducing the amount of time that defensive measures cause network operations to be disrupted is the goal when it comes to preventing an APT from disrupting a group of PLCs. The figure to the right shows how the Purdue enterprise reference architecture (PERA) is limited in the degree to which it safeguards the organization's engineering level (2) and plant level (1). All PLCs are at level 1; only workstations and servers are in level 2. (Programmable logic controllers). Approximately eighty-three nodes are part of the network in question, among which are 25 level 2 workstations, five level 1 workstations, and fifty PLCs.

In addition to ICS network states, the state space of the APT attacker states has to be included as well. Network state is an aggregate representation of all nodes in the network. The system seems to have certain amounts of access and control over the server and workstation nodes, as well as where they are situated on the network. The state of a PLC is used to show whether or not an assault has harmed it. APT, the actual APT sub-goal, and all the resources at your disposal are defined in the APT state.

The ACSO agent knows the network setup and PLC states, but if a node is hacked, the agent is notified. ACSO investigations are responsible for both actively and passively produced threat intelligence. On one hand, these may cause a lot of false positives. On the other hand, the identification of hacked nodes could provide a large number of false alarms. On the opening scene of each episode, the APT has managed to successfully compromise a single level 2 workstation and to access the system's data. After 25 PLCs have been disrupted by APT assaults, or after 5,000 hours of simulation, whichever comes first, the program concludes. To create a choice step, you must finish one hour of simulation world time. The discount factor is equal to 0.999, thus it takes one hour to accomplish one decision step.

It is up to the ACSO to decide whether it is appropriate to take action on a particular node or if it should do nothing each stage of the process. Workstations have seven valid workstation activity categories, whereas servers have six, and PLCs have two. Across the nodes in the network, there are a total of 329 different actions that may be carried out. In each step, the APT performs certain actions that are chosen according to the current state and an attacker model that uses random numbers. It is important to keep in

mind that certain actions will take time, which may be measured in time increments. Agents may be asked to engage in extra activities throughout this time period.

The prize money given for each stage is calculated according to how much of the PLCs are functioning and how much ACSO operations are causing disturbance. This is a reward function outline:

$$r(s, a) = r_{\text{PLC}}(s, a) + r_{\text{IT}}(s, a) + r_{\text{Term}}(s, a) \quad (1)$$

$$= 1 - 0.04 \sum_{p \in \text{PLCs}} 1\{p \text{ disrupted}\} \quad 1.0 - \sum_{a \in A_t} \text{cost}(a) + \frac{1}{\gamma} 1\{s_{\text{time}} \geq t_{\max}\} \quad (2)$$

The set of all actions done at time step t is referred to as the at set.

First, any twenty-five PLCs interrupted is compensated by a process stop in the first term. The use of activity costing means that when an ACSO action happens, a penalty is applied, and the cost is allocated to each action depending on how it is thought to impact network operations. Prior to devising a plan of action, specialists from the fields of network and security were interviewed to determine the expenses. With regard to low-disruptive operations, such as a reboot, a workstation will cost just 0.01 dollars, but a server reimaging will be more disruptive and cost around 0.05 dollars. Once the period is over, the agent gets paid no matter how long it takes to complete the episode (tmax). The optimum state value is not able to vary since the 1/ magnitude of the terminal reward guarantees that the terminal reward always sits at exactly.

This and future study need a well-constructed Julia ICS network attack simulation (INASIM). This is the first open-source APT assault simulation capable of producing data in the numbers needed for deep reinforcement learning on a single machine that is also free, according to the authors' understanding. In addition to having a bespoke network simulation, INASIM includes an APT attacker model that may be customized. Thanks to the Python bindings that have been provided, the simulation is compatible with both the POMDP and the Open AI Gym frameworks. To provide a quick synopsis of the simulation and the attacker model, below is a short description of the simulation and the baseline attacker model utilized in the experiment. You'll find more information on the environmental states, activities, and dynamics as well as the study parameters utilized in the tables below in the Appendix.

The network simulation determines the network topology, node statuses, and the allowed behaviors of the APT and ACSO agents. In the event of an investigation, the ACSO may examine node statuses or take steps to protect compromised nodes. If the adversary has control over a compromised node, the APT may use it to look for and infect other vulnerable nodes whereas if it has not yet gained control of a compromised node, the APT may use it to help facilitate further control over an already hacked node. Unlike the APT, which has the ability to only function if a system is infected, the ACSO can work on any machine in the network. Every node has been compromised, and the damage to each node is visible. Depending on the type of penetration, the APT may carry out several activities: one may be carried out from the host machine while another may be executed from the targeted node. Similarly, the degree of compromise on the node has an impact on how defensive ACSO operations affect the node. For example, in order to resist attempts by the ASO to reset the password on a node with "escalated privileges," an APT may operate on a node with "escalated privileges."

A stochastic finite state machine is used as an accurate depiction of the basic APT agent. In Table 1, the machine states and assault phase strategies are shown as linked entities. In each time step, the APT modifies the state of its machine to better reflect the belief it has of the network's connection. A stochastic rules-based sub-policy has been planned in advance to generate machine states depending on the exit criteria for each state. In each episode of the APT, there may be one objective, and the attack may employ one assault method. The APT is a successful attempt if it manages to disrupt or destroy the PLC-controlled process or equipment.

Solution Methods

In order to learn a policy from our attention-based neural network, we have enhanced the DQN algorithm and included it into the overall framework. According to the Rainbow DQN research, we revised our basic strategy. Double-DQN, prioritized experience playback, and n-step TD loss are among the many improvements.

$$L_{TD}\ Q_\pi(h_t, a) = \left[\sum_{\tau=t+1}^{t+n} \gamma^{\tau-t} r_\tau + \gamma^n Q_\varphi\ h_{t+n}, \operatorname{argmax}_{a'} Q_\pi(h_{t+n}, a')\right] - Q_\pi(h_t, a) \quad (3)$$

In where Q represents the estimate of the policy network's action value, and Q represents the estimate of the target network's action value. The Huber-loss norm is symbolized by the letter to compute the loss of each update, it is needed to account for the fact that a network update's impact differs based on batches of importance-weighted samples from an experience replay buffer.

Workers were given with a task reward in addition to one based on Ng, Harada, and Russell [34], which was derived (1). To incentivize the agent to work even harder, this reward was created.

$$r_{shape}(s, a, s') = \gamma(A\delta_W + B\delta_S) \quad (4)$$

An APT is considered to be compromised if the number of workstations and servers have increased from states to state s′. $\delta_W$and $\delta_S$reflect a change in the number of workstations and servers affected by the APT from state s to state s′, while A and B represent weight factors. Weighted Sum Equations 1 and 4 were used throughout the training phase. As far as assessment was concerned, only equation (1) was used.

The training parameters of the network were tuned using a grid search in which ten level 2 workstations were paired with three level 1 workstations and thirty PLCs were used. Shaping reward weight, observation-history interval, target network update frequency, and -greedy exploration decay schedule was all investigated. After 500 episodes, the performance strategy generated by the parameter configuration that outperformed the others in terms of average performance was judged to be the best option. PyTorch was utilized both for training and constructing neural networks in this experiment. Training parameters and other practical information on training is also included in the appendix.

Neural network

The more variables that are added to the ACSO problem, the greater the size of the input and output spaces. The ICS network's nodes the vector o(I) is a list of which stage I in the process the relevant alert or action was noticed. Workstation nodes have 16 components, server nodes have 14 pieces, and PLC nodes have 7 parts. Each ICS node will generate a unique observation vector, which includes 792 distinct items. This observation will include a full 792 items in the overall observation for the given time-step.

While the issue is only partly apparent, the network input for each step was based on a history of previous observations, which was essential because this is a multi-step problem. To obtain the whole input, the network has access to information about earlier phases of time t. As a consequence, the resulting value was Ht = Ot:t. A total of 256 preceding steps were used in this study, yielding a total input size of about 202,752. There are 329 valid actions at each time step, and this adds up to 329 total items in the output space.

Neural networks specialized for a particular task may derive benefit from the specific structure of input data, making the network less complicated. Using picture spatial invariance, such as with convolutional nets, is implemented by recurrent networks, for example. Additionally, recurrent networks encode sequence correlation. In order to handle the enormous quantity of input data without incurring substantial loss in explanatory power or encountering an insurmountable increase in size, we utilized attention processes to construct the neural network illustrated in Fig. 3. Using attention methods on a task that includes numerous interchangeable inputs may lead to an improvement in learning efficiency.

The temporal attention graph is shown in Figure (a), which alternates layers of multi-headed attention with 1D temporal convolution. The global attention sub-graph is shown in Figure (b), which alternates completely linked layers with multi-headed attention. To make gradient back-propagation easier, a skip connection is provided.

Each node's history is used to build a distinct sub-graph that is then used to embed the network's structure into a single latent state vector. This results in a large sub-graph with different components that forms a single higher-level structure. An increase in the number of nodes does not result in an increase in the total number of network parameters. Once the individual node states have been layered, they are stacked and added to a global attention sub-graph. This process is repeated until the desired number of layers has been reached. This sub-graph provides the network with information about the qualities of nodes nearby that are important to an action's value function, since it looks at the attributes of nodes near other nodes. Node vectors are passed on to the feed-forward output sub-graphs with contextualization. Nodes of the same type, such as input sub-graphs, have parameters that are shared throughout the whole network.

Figure 4 also shows the sub-graphs of temporal and global attention. Through time-ordered convolution in the time dimension, the network may be trained to priorities parts of the input sequence and increase their temporal attention by alternating dot-product attention layers with multiple-headed convolution. The multi-headed dot-product attention mechanisms alternate with affine layers in a cyclic manner in the global self-attention sub-graph. When relevant information from any other node in the network is included in the output representation for node I this structure makes it feasible. In order to divide the output space into distinct sub-graphs for each node, including a learnt global context into the latent representation of each node is feasible.

Over 683,000 parameters were under consideration in the neural network design. When the number of ICS nodes to be examined is taken into consideration, no requirement for this size modification exists. To establish a baseline, a convolutional neural network was used. Four 1D temporal convolutional layers are included as the first network layers in the baseline network. These levels are followed by a fully connected layer. Every network node adds 1,240,329 more parameters to the network. The baseline network was

built to mimic the baseline network as closely as feasible without causing significant changes to the latent variables' latent dimensions. The appendix describes the architecture of the convolutional network utilized in this research in some depth.

Pre-Training Method

Using greedy random exploration to investigate the issue proved to be time-consuming. We used an Alpha Star expert training-like supervised pre-training stage to expedite learning for our participants. In order to get the sample trajectories, it was essential to utilize stochastic rules-based expert policies. The next instruction to the agent was to minimize a composite loss term that applied to the whole collection of data under consideration.

We were able to find the pre-training loss term using a distributional DQN composite loss. "It's like this: We have sustained a total loss of $20,000 in monetary assets.

The pre-training loss term was based on a composite loss for distributional DQN [41]. The loss is defined as

$$L_P(Q) = L_{TD}(Q) + \lambda L_{LM}(Q) \quad (5)$$

where L_{TD} is the temporal difference loss defined in eq. (3), $_{LM}$ is a large-margin classificationloss, and λ is a weighting hyper-parameter. The margin loss term is defined as

$$L_{LM}(Q) = \max_{a \in A} Q(h, a) + l(a_E, a) - Q(h, a_E) \quad (6)$$

$$l(a\ , a) = \quad Q(h, a) - Q(h, a_E) + \delta, \quad \text{if } a_E \quad a \quad (7)$$

In where aE is the expert policy's chosen action and margin, a point is referred to as aE This quote suggests that expert policy acts should have a value that is equivalent to, or greater than, the value of all other policies. To ensure estimates are Bellman-consistent, TD-loss component must be included in the pre-training loss. The benefit of include the TD-loss component in the pre-training loss is that it helps to avoid forgetting during early learning.

Experiments

Using the training technique and DQN algorithm, we trained our neural network to deal with the ACSO problem. The research examined training strategies on a group of 100 experiments, and although the APT objective and attack vector were fixed, training efficacy was assessed. We were able to calculate the impact of each new feature by calculating the relationship to our previous practices and techniques. The combined policies' performance as well as the samples' effectiveness in terms of learning were examined in these studies. predictive statements

The findings of this combined loss were encouraging, and it was used to pre-train both network topologies. To finish, the relevant unscripted footage was utilized to augment each of the previously trained networks, resulting in a total of 500 episodes. Using the hyper-parameters that were discovered during the grid-search procedure, the RL training was carried out. Pre-training was used to measure the effect of training each design using DQN before training, training each design using 5 randomly chosen networks, in order to assess the differences between the outcomes. An appendix of in-depth training material is provided.

The results shown in Figure 4 illustrate the learning performance of each experiment design. For randomly started networks, the best performing seed is provided in this section. It is shown by way of

equation (3) and a discounted task return value of 1.25-million-episode steps, that the average TD loss of Equation (3) is 42 TD. Additionally, note that the TD loss of our neural network quickly degrades regardless of whether or not pre-training is used, whereas pre-training only slows down the rate of TD loss decline. The actions the agent selects are more likely to be comparable to the taught strategy than the other way around given the action-value margin in equation (6). As a result, the policy runs across fewer newly observed-action pairings, and as a result, the policy must spend longer learning about their values. In other words, after almost 200,000 footsteps, the TD-decay was increased by pre-training. The decision to use the CNN in the pre-inconsistent training may be owing to the fact that the CNN is more sensitive to the action-value estimations produced before to the inconsistent training.

Both the TD-loss and the discounted return's learning performance follow the same patterns, and with our approach, we tend to converge sooner in most instances. Also, we were consistently better converged than the convolutional network in most instances. Training caused our design to perform considerably better and get to converged performance much sooner than the TD-loss trends. The performance of the convolutional architecture was negatively impacted by the training prior to that.

Network	Pre-Train	Return	ActionCost	PLCDowntime	CompromiseTime
Ours	Yes No	940.4±0.0 **941.0±0.4**	0.59±0.01 0.59±0.01	**0.0±0.0** **0.0±0.0**	**0:2 _ 0:0** 1:0 _ 0:0
Conv	Yes No	791:5 _ 2:7 738:6 _ 5:0	**0:26 _ 0:01** 0:57 _ 0:01	6:2 _ 0:2 5:0 _ 0:3	3:9 _ 0:0 3:5 _ 0:1
Expert	-	906:9 _ 4:5	0:72 _ 0:01	0:2 _ 0:0	1:3 _ 0:2

Table 2: Evaluation Performance:

The results of the assessments in Table 2 show the assessment performance of the best policy in each configuration, with the highest score being the greatest performance. An average total number of 100 episodes is calculated for each of the four performance indicators, and together with their corresponding standard errors, is shown in the table. the amount of returned funds reduced by the time it took to earn the money (1). For an ACSO, the cost of taking an action is 10 times the standard cost of a defensive action step. The total number of hours that PLCs were off throughout each 5,000-hour episode is described as PLC downtime. With regard to the meaning of compromise time, which refers to the total number of IT nodes that are in an off-nominal condition for an entire simulated hour, the phrase is defined as: In addition, an in-depth look is given to the expert policy that was used to produce the pre-training data.

Also shown are our neural network's superior results when trained with both convolutional and expert policies, as compared to a convolutional network and an expert policy, respectively. A randomized network will have similar expected return and investment costs to a pre-trained network, and that is why we used a randomized network in this research. Surprised by the lower average number of compromised nodes each hour than anybody had expected, the pre-trained policy not only found a more effective policy as it learnt, but it did so use the higher learning rate as well. Even though both of the convolutional networks were trained using trajectories developed by experts, none of the networks was able to best the trajectories developed by experts. Even though their overall return was lower, the novel networks fared better than the convolutional networks when it came to preventing future PLC downtime. By making these assumptions, we may infer that the methods it discovered were more aggressive and successful in restricting APT's development than previously believed. Additionally, the expert policy required an

average action cost that was greater than that of both of the new networks together. In other words, despite the excellent policies that prevented PLCs from failing, they were less efficient than learning methods because of the data.

Conclusions

A POMDP model for solving the issue of controlling industrial network security via orchestration of cyber-related processes is presented in this article (ICS). A cheap and efficient ICS network attack simulation has been developed, enabling researchers to carry out deep RL studies both now and in the future. Deep reinforcement learning was shown to be possible using a new DQN architecture that accounted for the size of the observation and action areas. The pre-training we gave our architecture enhanced our architecture's learning capability.

To ensure the legitimacy of our findings, we conducted several experiments in our research of the effect of each contribution on learning efficiency and convergent policy performance. Our suggested network design, together with the pre-training, has shown to enhance learning rate and resulting performance. According to the study's findings, deep reinforcement learning may be used to create autonomous security agents that could help minimize the likelihood of advanced persistent threats. Some of the solutions that have been suggested may be applied to new discrete space issues that exist in the actual world. The findings showed that implicit bias training may increase training efficiency while being necessary to create an acceptable policy.

It will be necessary to create a more accurate simulation of real-world network dynamics in order to further this study. More creative variants include simulating assaults using virtual machine kernels and real-world vulnerabilities and creating simulators that use virtual machine kernels and real-world vulnerabilities. Due to the computational and security difficulties presented by the latter, the likelihood of the former succeeding is greater. In order to effectively connect rewards to processes, another approach to explore is incorporating a model of an actual ICS process and assigning RL rewards based on the process' effectiveness. Various additional attack routes, like spoofing of IDS signals or ACSO computer assets, should also be explored.

Additional solution approaches will be researched in connection with the suggested network design. In contrast to sliding window histories, recurrent neural networks and variational networks may provide more accurate representations of the network's state than these other options. Attention processes and graph neural networks may be used to create different topologies for learnt rules to be applied into ICS networks. Phasic policy gradientis an alternative to the DQN approach, although other ways like this will be looked at as well.

A fixed, stochastic attacker model is considered in this study. This study will investigate the extent to which adversarial learning may be utilized to produce difficult attacker methods to improve the resilience of learnt rules. Another point to note is that it was also chosen to represent the warning and transition dynamics as stationary, even though the dynamics are dynamic in the actual world. Additionally, these dynamics will be placed through a stress test. Special attention will be paid to ensuring the safety of key industrial control systems in order to explore how to guarantee and verify ACSO operations.

References

1. T.Alladi,V.Chamola,“Industrialcontrolsystems:Cyberattacktrendsandcountermeasures,”Co mputerCommunications,vol.155,pp.1–8,2020.
2. R.Langner,“Stuxnet:Dissectingacyberwarfareweapon,”IEEESecurityandPrivacy,vol.9,no. 3, pp. 49–51, 2011.
3. G. Liang, F. Luo, S. R. Weller, J. Zhao, “The 2015 ukraine blackout: Implicationsforfalsedatainjectionattacks,”IEEETransactionsonPowerSystems,vol.32,no.4,p

p.3317–3318,2016.
4. O.Vinyals,I.Babuschkin,W.M.Czarnecki,M.Mathieu,A.Dudzik,J.Chung,D.H.Choi,R.Powell,T .Ewalds,D.Horgan,P.Georgiev,J.Oh,M.Kroiss,I.Danihelka,A.Huang,L.Sifre,D.Hassabis,C.Ap ps,andD.Silver,"GrandmasterlevelinstarcraftIIusingmulti-agent reinforcement learning," Nature, vol. 575, no. 7782, pp. 350–354, 2019.
5. 6. C.-J. Hoel,L. Laine,K. Wolff, K. Driggs-Campbell, and M. J. Kochenderfer, "Combiningplanning and deep reinforcement learning in tactical decision making for autonomous driving,"IEEE Transactionson Intelligent Vehicles,vol.5,no.2,pp.294–305,2019.
6. G. Dulac-Arnold,B. Coppin, R. Evans, andP. Sunehag, "Reinforcement learninginlargediscreteactionspaces,"ComputingResearchRepository,2015.arXiv:1512.07679.
7. T. T. NguyenandN. D. Nguyen, S. Nahavandi,, "Deep reinforcement learning for multia-gent systems: A review of challenges, solutions, and applications," IEEE Transactions onCybernetics,vol.50,no.9,pp.3826–3839,2020.
8. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A.Riedmiller, "Playing atari with deep reinforcement learning," Computing Research Repository, 2013.

1. ***Figure:***

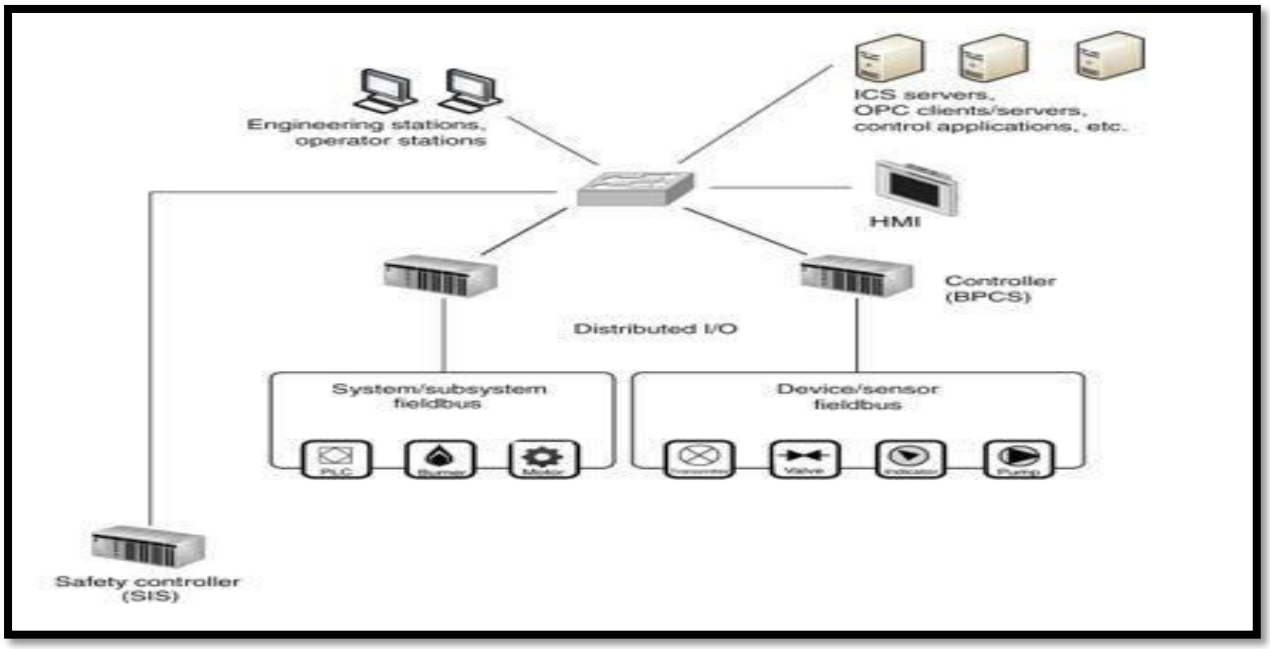


Figure 1: Simulated Network Architecture

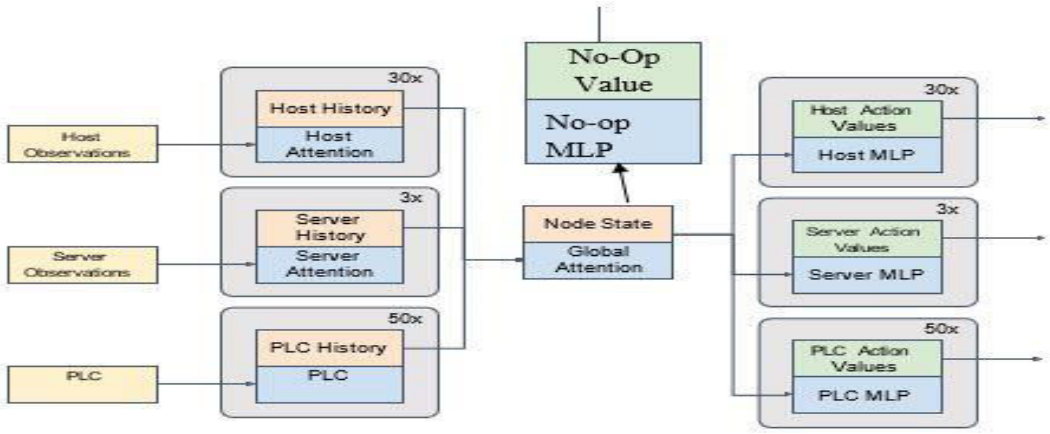


Figure 2: Attention Network

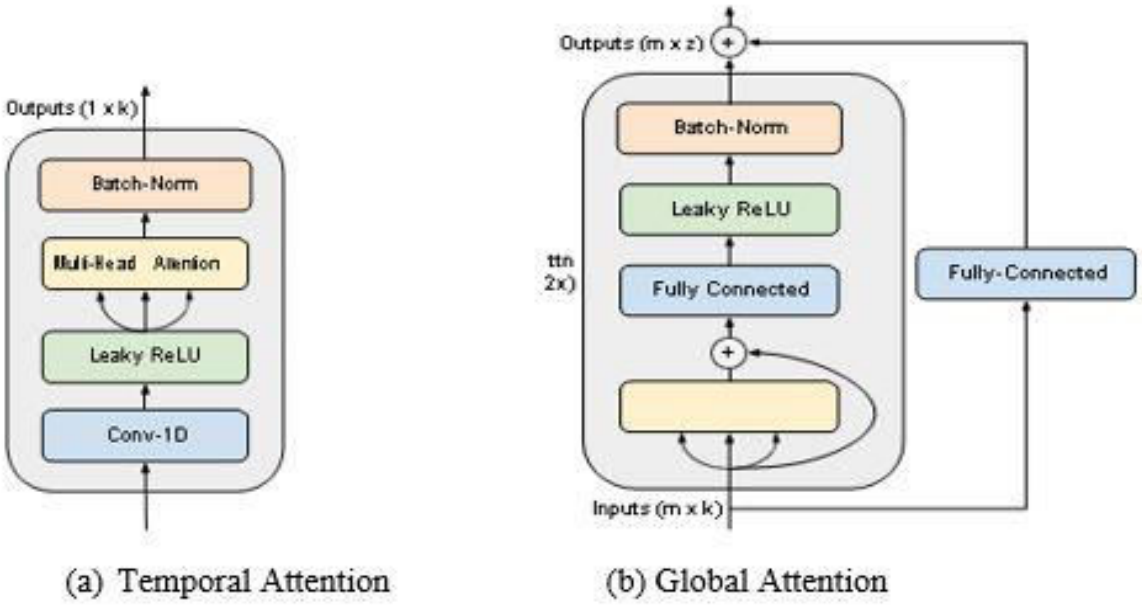
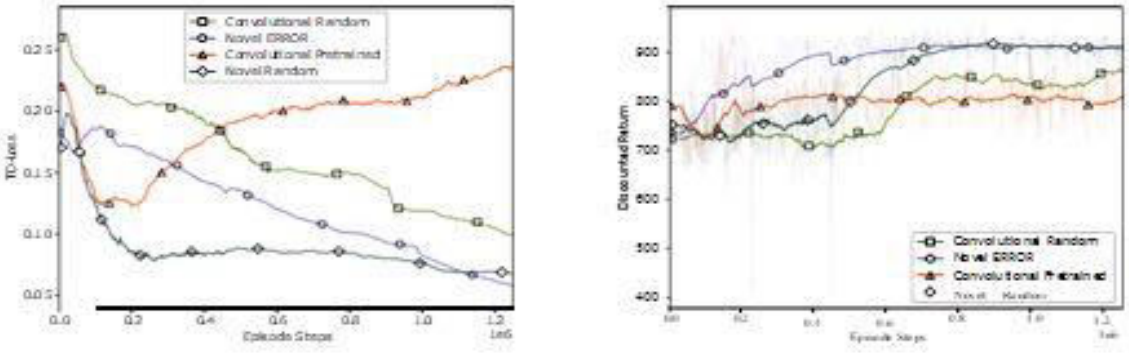


Figure 3: Attention Sub-Graphs



(a) TD-Loss (b) Discounted Return

Figure 4: Training Results