PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

Autonomous cyber warfare agents: dynamic reinforcement learning for defensive cyber operations

David Bierbrauer, Robert Schabinger, Caleb Carlin, Jonathan Mullin, John Pavlik, et al.

David A. Bierbrauer, Robert M. Schabinger, Caleb Carlin, Jonathan Mullin, John A. Pavlik, Nathaniel D. Bastian, "Autonomous cyber warfare agents: dynamic reinforcement learning for defensive cyber operations," Proc. SPIE 12538, Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications V, 125380E (12 June 2023); doi: 10.1117/12.2663093

SPIE.

Event: SPIE Defense + Commercial Sensing, 2023, Orlando, Florida, United States

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023 Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

Autonomous Cyber Warfare Agents: Dynamic Reinforcement Learning for Defensive Cyber Operations

David A. Bierbrauer[a], Robert M. Schabinger[b], Caleb Carlin[b], Jonathan Mullin[b], John A. Pavlik[a], and Nathaniel D. Bastian[a]

[a]Army Cyber Institute, United States Military Academy, West Point, NY 10996, USA
[b]DCI Solutions, Aberdeen Proving Ground, MD 21005, USA

ABSTRACT

In this work, we aim to develop novel cybersecurity playbooks by exploiting dynamic reinforcement learning (RL) methods to close holes in the attack surface left open by the traditional signature-based approach to Defensive Cyber Operations (DCO). A useful first proof-of-concept is provided by the problem of training a scanning defense agent using RL; as a first line of defense, it is important to protect sensitive networks from network mapping tools. To address this challenge, we developed a hierarchical, Monte Carlo-based RL framework for the training of an autonomous agent which detects and reports the presence of `Nmap` scans in near real-time, efficiently and with near-perfect accuracy. Our algorithm is powered by a reduction of the state space given by a transformer, `CLAPBAC`, an anomaly detection tool which applies natural language processing to cybersecurity in a manner consistent with state-of-the-art. In a realistic scenario emulated in `CyberVAN`, our approach generates optimized playbooks for effective defense against malicious insiders inappropriately probing sensitive networks.

Keywords: Defensive Cyber Operations, Reinforcement Learning, Monte Carlo, Transformers, Emulation

1. INTRODUCTION

Incidents in the cyberspace domain continue to grow in frequency and scale. Cybersecurity Operations Centers (CSOCs) typically treat these incidents as emergencies by following standard response procedures. Most organizations will develop customized procedures from frameworks like the National Institute of Standards and Technology (NIST) *Computer Security Incident Handling Guide*, which defines a four step Incident Response Life Cycle as seen in Figure 1.[1] With this starting framework, CSOCs will create defensive cyber operations playbooks, which serve as action plans that document sets of steps they follow to recover from or even prevent cyber events.[2] The CSOCs adapt their procedures according to organizational policies, leading to the development of specific workflows for handling specific types of events.[3]

For example, an organizational policy may restrict users from attaching thumb drives to workstations to prevent installation of potential malware. When a user violates this policy, the CSOC receives an alert and conducts an investigation. Once the investigation concludes, the CSOC's playbook may specify re-imaging of the workstation before closing the incident. This of course solves the immediate issue of the policy violation, but may not improve the overall security of the network. If we assume the thumb drive did contain malicious code designed to exploit a specific vulnerability on one of the organization's servers, then the malicious actor may find another way to introduce the code to the network. The CSOC remains unaware that the true threat potentially remains, and the playbook fails to adapt appropriately. If instead the playbook dynamically changed to enable the use of a honeypot - or decoy - server to trap the malicious actor when the code executed, then the CSOC would have a greater understanding of the actor's intent and the vulnerability itself while ensuring the policy remains intact.

This overall playbook could also involve several intermediate steps that require optimization in order to increase effectiveness. These steps may involve specific commands or examination of several logs that reduce a CSOC's ability to act swiftly against a potential threat. Depending on the expertise of CSOC personnel, this could have disastrous implications for a network.

Send correspondence to: nathaniel.bastian@westpoint.edu.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

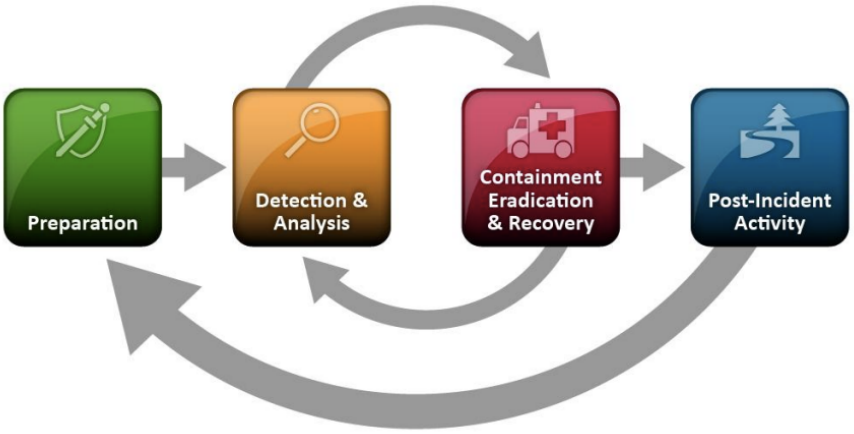

Figure 1. A framework from NIST depicting their four-step incident response life cycle.

Thus, we see several challenges associated with the current use of defensive cyber operations (DCO) playbooks. First, they may lack flexibility to adapt to malicious intent or tactics. Second, steps involved in specific workflows may not be optimized for certain scenarios or environments. As such, cyber defenders may not achieve parity with malicious actors who constantly change tactics and exploit numerous vulnerabilities in information systems and networks.

Although this lack of flexibility impacts many organizations, it could have devastating effects on the military's ability to achieve information dominance on the modern battlefield. As the United States Army pivots to more interconnected devices and platforms at the edge of its networks,[4] cyber defenders need the ability to sense, understand, decide, act, and assess more quickly than their adversaries to counter threats.[5] This becomes increasingly difficult as computer networks increase in size and scope while resources remain pooled in enterprise CSOCs. To aid cyber defenders in maintaining an advantage over advanced persistent threats (APTs), we explore applications of reinforcement learning (RL) in computer network scenarios that will not only ensure information security, but also provide adaptable workflows to changing environments. These dynamic DCO playbooks will enable defenders to rapidly adjust to evolving threats. This paper makes the following contributions:

- Formulates the dynamic DCO playbook generation problem as a Markov Decision Process.
- Solves this problem using transformer modeling and Monte Carlo based hierarchical reinforcement learning.
- Implements this solution methodology in an advanced computer network emulation environment.
- Demonstrates the feasibility of enhancing CSOC operations with trained agents.

Our paper is structured as follows. First, we discuss related works in Section 2. We then describe how we formulate the DCO playbook optimization problem as a MDP and discuss our cutting-edge RL approach to its solution in Section 3. We follow this with an explanation of the computational environment utilized for our experiments in Section 4 before providing our results in Section 5. We then conclude our paper by examining our limitations and plans for future work in Section 6.

2. RELATED WORKS

Many surveys have been conducted on the application of RL to cybersecurity problems. One such survey by Adawadkar and Kulkarni focused on three key problem areas: building intrusion detection and prevention systems; identity and access management; and characteristics of Internet of Things (IoT) devices and networks.[6] Although all three areas may have some impact on our approach, the IoT applications are of particular interest. One such application relates to optimal resource allocation such as the work done by Dutta and Biswas where they formulate a multi-agent MDP for maximizing network throughput in wireless sensor networks.[7] Much of

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

the IoT literature also focused on solving these problems for unmanned aerial vehicles (UAVs). For example, Yang *et. al.* utilized a deep RL model for improving task execution efficiency in UAVs for load-balancing across mobile-edge networks.[8] Research into these applications, which might prove useful for IoT-style edge networks, rarely focus on the direct problem of defending these networks from malicious activity. Therefore, we must explore how RL has been applied in more traditional enterprise scenarios for cybersecurity.

A recent article by Wang *et. al.* proposed a framework for such research based on the threat life cycle. Namely, the authors focused on penetration-testing, design, response, and recovery as different decision-making tasks for cybersecurity.[9] The response and recovery tasks hold particular interest as they encompass many of the tasks a CSOC may perform. During the response task, defenders must have the capability to detect abnormalities and respond appropriately. One application of this task focused on training an autonomous agent to defend a software-defined network. Specifically, the authors train an agent to protect critical servers while introducing some adversarial methods to disrupt the training process.[10] Recovery, meanwhile, typically occurs after response and requires defenders to restore functionality or mitigate residual risk from an event.[9] The associated literature with recovery is fairly limited, with some focusing on recovery of critical power infrastructure.[11,12] Separating these tasks into different decision models may have utility in some scenarios, but doing so may not give cybersecurity professionals optimal workflows to enable holistic network health. That is, the optimal response decision may lead to suboptimal recovery decisions; therefore, we consider a more comprehensive approach to achieve a desired end-state.

Specific research for CSOC operations have used optimization-based approaches for cybersecurity problems. Ganesan *et. al.*[13] proposed a generalized optimization model using mixed-integer programming (MIP) for scheduling cybersecurity analysts to minimize risk (*e.g.*, maximize significant alert coverage by analysts) and maintain risk under a predetermined upper bound. The article tested the optimization model and its scalability on a set of given sensors with varying analyst experiences, alert generation rates, system constraints, and system requirements. Shah *et. al.*[14] presented a novel two-step sequential MIP optimization method that was used in the development of a new decision-support business model for outsourcing the alert analysis process. It was demonstrated that through this model, a CSOC could effectively deliver its alert management services. Shah *et. al.*[15] further investigated the problem of allocating clusters of sensors to analysts for investigation within a CSOC. There were two essential properties that must be met in the above grouping and allocation process: 1) meeting the cluster's requirement for specific analyst expertise mix, complete tool coverage that allows the analysts to handle the type of alerts generated, and analyst credentials such as security clearances; and 2) minimizing and balancing the number of unanalyzed alerts among clusters at the end of the daily work shift because an imbalance or a large number of unanalyzed alerts among clusters due to factors such as lack of analyst credentials or tooling expertise in a cluster would pose a security risk to the organization. The authors modeled and solved this cybersecurity resource allocation optimization problem using a MIP model.

Other research for CSOC problems applied RL through the lens of sequential decision-making. Ganesan *et. al.*[16] used a RL-based stochastic dynamic programming optimization model that incorporated estimates of future cyber alert rates and responds by dynamically scheduling cybersecurity analysts to minimize risk (*i.e.*, maximize significant alert coverage by analysts) and maintain the risk under a predetermined upper bound. The authors tested the dynamic optimization model and compared the results to an integer programming model that optimized the static staffing needs based on a daily-average alert generation rate with no estimation of future alert rates (static workforce model). Molina-Markham *et. al.*[17,18] proposed an RL-enabled cyber defense model but acknowledged the complexity of the environment and described it as a collection of games with constantly drifting rules. Molina-Markan *et. al.* then expanded on their previous claim of the complexity of the rules of network defense and showed that their approach was necessary to facilitate the development of RL network defenders that are robust against attacks aimed at the agent's learning. Hore, Shah, and Bastian[19,20] proposed a novel framework, Deep VULMAN, consisting of a deep RL agent and an integer programming method to optimize the cyber vulnerability management process for a CSOC. Their sequential decision-making framework, first, determined the near-optimal amount of resources to be allocated for mitigation under uncertainty for a given system state and then determined the optimal set of prioritized vulnerability instances for mitigation. Their proposed framework outperformed the current methods in prioritizing the selection of important organization-specific vulnerabilities, on both simulated and real-world vulnerability data, observed over a one-year period.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

3. METHODOLOGY

3.1 Preliminaries

In order to understand our mathematical formulation and solution approach to the DCO playbook optimization problem, we first discuss the fundamentals of MDPs along with model-based versus model-free RL algorithms. We then provide an overview of transformer modeling.

3.1.1 Markov Decision Processes

MDPs are powerful analytical tools that generalize standard Markov models by embedding the sequential decision process in the model and allowing multiple decisions in multiple time periods. The basic definition of a discrete-time MDP contains five components. The decision epochs, $t = 0, 1, \ldots, N$, are the set of points in time at which decisions are made (seconds, minutes, etc.). The state space S is the set of all possible values of dynamic information relevant to the decision process. For any state $s \in S$, A_s is the action space, the set of possible actions that the decision maker can take at state s. Transition probabilities, $p_t(\cdot|s, a)$, are the probabilities that determine the state of the system in the next decision epoch, which are conditional on the state and action at the current decision epoch. Finally, the reward function, $r_t(s, a)$, is the immediate result of taking action a at state s. Thus, these five components collectively define an MDP.

A decision rule is a procedure for action selection from A_s for each state at a particular decision epoch, namely $d_t(s) \in A_s$. We can drop the index s from the expression and use $d_t \in A$, which represents a decision rule specifying the actions to be taken at all states, where A is the set of all actions. A policy π is a sequence of the decision rules to be used at each decision epoch and defined as $\pi = (d_0, \ldots, d_{N-1})$. The objective of solving an MDP model is to find the optimal policy that maximizes a measure of long-run expected rewards. Note that future rewards are often discounted over time. In the absence of a discounting factor, if we let $u_t^*(s_t)$ be the optimal value of the total expected reward when the state at time t is s and there are $N-t$ periods to the end of the time horizon, then the optimal value functions and the optimal policy giving these equations can be obtained by iteratively solving the following recursive equations (known as the Bellman equations):

$$u_N^*(s_N) = r_n(s_n) \quad \forall \quad s_N \in S \tag{1}$$

$$u_t^*(s_t) = \max_{a \in A_s} \left\{ r_t(s_t, a) + \lambda \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j) \right\} \text{ for } t = 0, \ldots, N-1, s_t \in S \tag{2}$$

where $r_n(s_n)$ denotes the terminal reward that occurs at the end of the process when the state of the system at time N is s_n, and λ represents the discount factor ($0 < \lambda < 1$) specifying the relative importance of rewards.

At each decision epoch t, the optimality equations given by the second equation chooses the action that maximizes the total expected reward that can be obtained for periods $t, t+1, \ldots, N$ for each state s_t. For a given state s_t and action a, the total expected reward is calculated by summing the immediate reward, $r_t(s, a)$, and future reward, which is obtained by multiplying the probability of moving from state s_t to j at time $t+1$ with the maximum total expected reward $u_{t+1}^*(s)$ for state j at time $t+1$ and summing over all possible states at time $t+1$. A finite-horizon MDP model is appropriate for systems that terminate at some specific point in time. At each stage, we choose the following:

$$a_{s_t,t}^* \in \arg\max_{a \in A_s} \left\{ r_t(s_t, a) + \lambda \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j) \right\} \text{ for } t = 0, \ldots, N-1 \tag{3}$$

where $a_{s_t,t}^*$ is the best action maximizing the total expected reward at time t for state s.

In other situations, an infinite-horizon MDP model is more appropriate, in which case the use of a discount factor is sufficient to ensure the existence of an optimal policy. The most commonly used optimality criterion for infinite-horizon ($N \to \infty$) problems is the total expected discounted reward. In an infinite-horizon MDP, the following very reasonable assumptions guarantee the existence of optimal stationary policies: stationary

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

(time-invariant) rewards and transition probabilities, discounting with λ, and discrete state and action spaces. Optimal stationary policies still exist in the absence of a discount factor when there is an absorbing state with immediate reward zero). In a stationary infinite-horizon MDP model, the time indices can be dropped for the reward function and transition probabilities, leaving the following Bellman equations:

$$V(s) = \max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) V(j) \right\} \text{ for } s \in S \tag{4}$$

where $V(s)$ is the optimal value of the MDP for state s (the expected value of future rewards discounted over an infinite horizon). The optimal policy consists of the actions maximizing this set of equalities.

MDPs may be classified according to the time horizon in which the decisions are made: finite- and infinite-horizon MDPs. Finite horizon and infinite-horizon MDPs have different analytical properties and solution algorithms. Because the optimal solution of a finite-horizon MDP with stationary rewards and transition probabilities converges to that of an equivalent infinite-horizon MDP as the planning horizon increases and infinite-horizon MDPs are easier to solve and to calibrate than finite-horizon MDPs, infinite-horizon models are typically preferred when the transition probabilities and reward functions are stationary.

The optimality (Bellman) equations can be solved using dynamic programming (DP). DP is an algorithmic technique for solving an optimization problem by breaking it down into simpler subproblems and utilizing the fact that the optimal solution to the overall problem depends upon the optimal solution to its subproblems. DP is mainly an optimization over plain recursion. The idea is to simply store the results of subproblems, so that we do not have to re-compute them when needed later. To use DP, the problem must have overlapping subproblems (*i.e.*, finding its solution involves solving the same subproblem multiple times). Also, the overall optimal solution must be able to be constructed from the optimal solutions of its subproblems. For more details on MDPs, DP and other algorithmic solution approaches, please refer to Puterman,[21] Bersetkas,[22] and Powell.[23]

3.1.2 Reinforcement Learning

Reinforcement Learning (RL) is a computational approach to goal-directed learning from interaction, which is used for sequential decision-making under uncertainty. The core idea is learning from interaction between a decision-making agent and its environment to achieve a goal (despite uncertainty about the environment).[24] In simple terms, RL is focused on learning what to do — how to map situations to actions — so as to maximize a numerical reward signal. Thus, RL is concerned with creating agents that learn to make good sequences of decisions using evaluative feedback within a simulated and/or emulated environment; evaluative feedback occurs after the agent makes a decision and is informed if the decision was good or not. The RL problem may be formalized as a MDP to define the interaction between a learning agent and its environment in terms of states, actions and rewards as depicted in Figure 2. The *policy* defines the learning agent's way of behaving at a given time (mapping of states to actions), the *reward signal* defines the goal of the RL problem, and the *value function* reflects the total amount of reward an agent can expect to accumulate over the future. As discussed above, a core piece of using DP to solve an MDP is that the probability of moving from one state to another state (*i.e.*, state transition probabilities) is known. When this holds true for an RL problem, we refer to this as *model-based* (planning). In many real-world settings, however, the environment behavior is unknown, so we refer to this as model-free (trial-and-error). In the model-free RL setting, a simulated and/or emulated environment is needed to generate the state transition probabilities.[24] For the purposes of our work we utilize the model-free setting, and we describe the specific emulation environment used in Section 4.

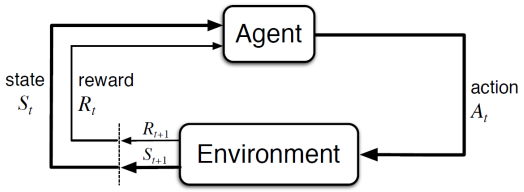

Figure 2. A visual depiction of agent-environment interaction that takes place in reinforcement learning.[24]

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

3.1.3 Transformer Modeling

A transformer model is a neural network which is able to efficiently learn context and meaning by tracking relationships in sequences of tokens. Traditionally, the tokens in question have been taken to be words in natural languages, as the transformer model was pioneered with the goal of revolutionizing the task of translation.[25] However, this is not a necessary restriction and, in fact, transformer modeling can be applied equally well to more abstract sequences of tokens. Bommasani *et. al.* described such transformer models as "foundation models" as they train on broad data and have a wide range of potential applications, such as computer vision, chat bots, and robotic manipulation.[26] Such models have a wide range of capabilities due to how they utilize positional encoders to tag elements and find patterns between them without the need for large, labeled data sets.[27]

Transformer architectures continue to evolve,[28,29] but it is worth highlighting a novel core feature of transformer models which makes the efficient delivery of self-supervised learning possible: multiplicative self-attention. Self-attention allows the model to learn relationships between the tokens in input sequences, relationships which may vary depending on context, directly from data sets collected in the wild with very mild curation (see Figure 3) and electing to employ a multiplicative rather than additive paradigm for the attention architecture boosts performance via highly-optimized standard libraries for matrix multiplication.

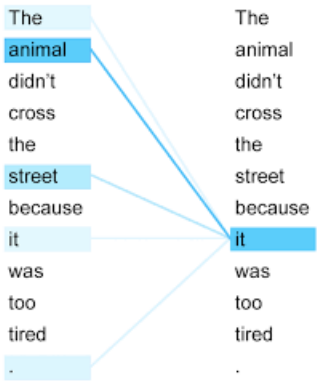

Figure 3. Self-attention allows the transformer to automatically see that parts of an input sequence of tokens are correlated.

While transformer models have been a mainstay of natural language processing since their inception, they were applied to machine languages in the context of cybersecurity more recently. In Steverson *et. al.*,[30] their `CALBAC` program was able to successfully train transformer models to recognize statistical differences between in-sample and out-of-sample Windows event log files, thus allowing for the creation of powerful file-based anomaly detection software. In this work, we use `CLAPBAC`,[31] a network-based anomaly detection toolkit which enables the training of models with a transformer architecture analogous to those produced with `CALBAC` but which train on the headers of network packet captures rather than on system log files. Trained `CLAPBAC` models return a non-negative real number for each inspected packet at inference, where numbers close to zero indicate that the inspected packet header is statistically similar to packets in the training data sample and larger numbers suggest that the inspected packet header is anomalous in some way.

3.2 Scenario Design and MDP Formulation

There is a growing awareness in the cybersecurity community that full network traffic emulation is much preferred over simulation when the mission is to train autonomous cyber defense agents with a high degree of realism.[17,18] In this paper, we employ `CyberVAN`,[32] a general software suite for emulation which aims to support experimentation on virtual networks comprised of virtual machines (VMs) running common operating systems connected to each other through virtual hubs, routers, and switches. In order to train a scanning defense agent, we start with a network broken down into three subnets, each containing one two CPU core VM running `Windows 7` and one two CPU core VM running `Ubuntu 18.04 LTS`, connected to the internet through a router. Our network monitoring with `CLAPBAC` and subsequent RL training is carried out on a server with twelve CPU cores which taps the entire network. Realistic background traffic on the desktop VMs involving activities such as sending

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

emails and transferring files is emulated using an additional utility called `ConsoleUser`. A visualization of the `CyberVAN` scenario used for our experimentation can be found in Figure 4.

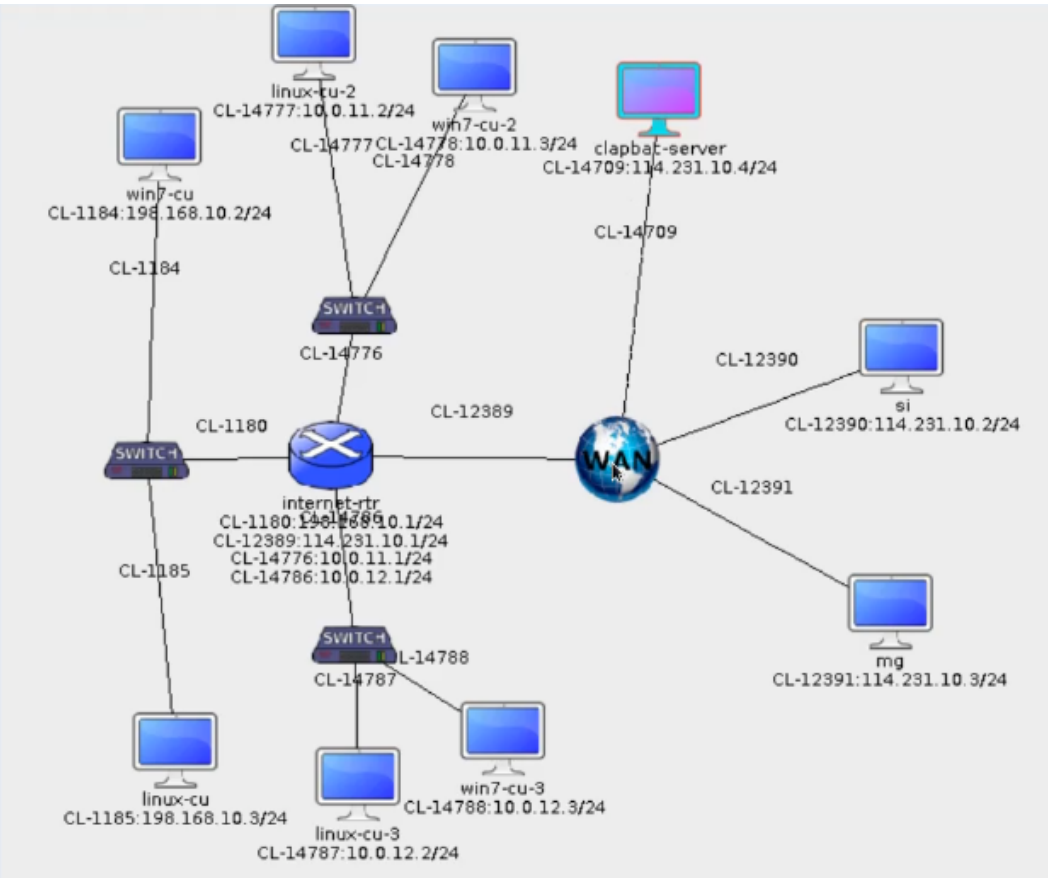

Figure 4. A schematic of our `CyberVAN` scenario. The "internet" in our scenario is modeled by a synthetic internet server (`si`) hosting an array of virtual IP addresses, together with a media gremlin (`mg`) node in a supporting role.

With this scenario in mind, we envision an experiment where a malicious actor – henceforth referred to as the *red agent* – takes over the first subnet and then scans virtual internet protocol (IP) addresses hosted on the synthetic internet server at random using the `Nmap` network mapping utility. These malicious scans occur every eight seconds during training data collection. Our defender – the *blue agent* – must learn to distinguish scans from background traffic using RL with the output of our `CLAPBAC` model for each inspected packet as a starting point. Our scenario most naturally defines an MDP where the analysis of each packet constitutes an episode. Further details of our MDP and training experimentation are described below.

3.2.1 States

In our experimentation, the blue agent cultivates a coarse awareness of the common operating picture (COP) by continuously monitoring the network traffic with `CLAPBAC`. The initial state space exposed to the blue agent is therefore at least $X \times M$ dimensional, where X is the number of monitored IP addresses and M is the set of packet header parameters. In practice, an essential reduction of the state space is provided by the output of our `CLAPBAC` model at inference time, as the model provides a positive real number score to each packet inspected, larger numbers implying a correspondingly larger probability of anomaly. In marginal situations where the anomaly score is not sufficiently indicative by itself, the blue agent may elect to perform either a cursory or a rigorous inspection of the questionable packet source VM for clues in order to decide whether or not a threat alert response is warranted. If the blue agent elects to perform a cursory inspection which turns out to be inconclusive, it enters a state where it has the option to follow up with a rigorous inspection. As a control, states

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

are also provided to the blue agent during training which should never be visited in any learned policy, as they are highly disfavored compared to the other available choices.

3.2.2 Actions

Due to the state space reduction afforded to us by `CLAPBAC`, our agent is able to consider and issue a response to each network packet using the relatively light-weight `clapbac-server`. When each packet arrives off the wire, the agent will decide if it looks innocuous, if it looks malicious, or that some form of investigation of the potentially affected device is warranted. A control action is also provided where the agent logs onto the device and counts the files in the root folder. This action is of course a waste of time and should never be selected by a trained agent. Action a_3 always succeeds but might take longer because it requires a careful correlation of `Nmap` and `CLAPBAC` log data to know for certain whether a malicious scan occurred. Action a_4 succeeds only 2/3 of the time but may be quicker, as it just logs onto the potentially affected device and checks for the presence of the `Nmap` executable. The logic behind a_4 is that, `Nmap` is ubiquitous in cybersecurity and could very well have been installed for legitimate work-related reasons. Finally, we found it useful for the richness of our scenario to allow the agent to learn whether exit status 0 or exit status 1 means that something went right with the execution of a `bash/cmd` command. A visualization of the state-action space can be found in Figure 5.

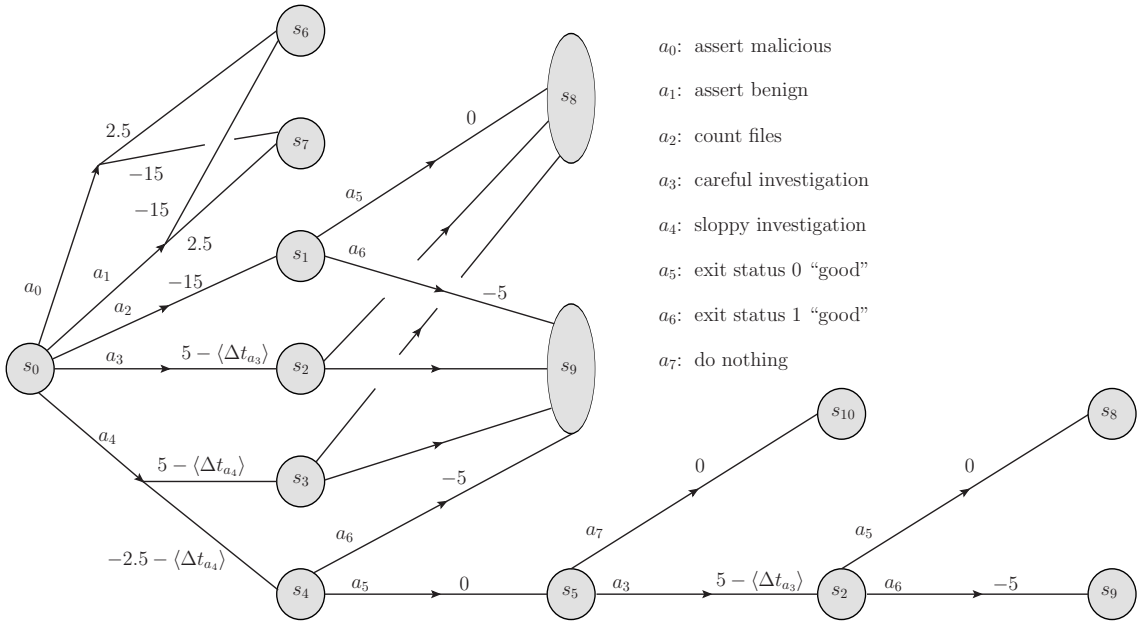

Figure 5. State-action diagram for blue agent. There is a state s_0 for each value of the `CLAPBAC` score. Assigned awards for each state-action pair are provided after each transition arrow. In a few cases, an oracle program assists the training by directing the transition to the next state according to whether or not the prior action was a success. The actions and rewards for transitioning out of states s_2 and s_3 are suppressed, as they are the same as from s_1 and s_4.

3.2.3 Rewards

Our RL training regimen must provide the blue agent rewards for the actions described above. The reward signal is designed to encourage investigation in marginal situations; the reward provided for a successful a_0 or a_1 must be relatively small because the average investigation time durations $\langle t_{a_3} \rangle$ and $\langle t_{a_4} \rangle$ turned out to be significantly smaller than expected. For a_3, we found it appropriate to assign an award of $5 - \langle t_{a_3} \rangle$, to reflect the cost of performing a careful investigation. On the other hand, a_4 can either succeed ($5 - \langle t_{a_4} \rangle$ reward) or fail ($-2.5 - \langle t_{a_4} \rangle$ reward). In contrast to a_0 and a_1, a lesser penalty for failure is given here since it may have a small amount of value to do an investigation independent of the outcome; there will in any case be some trace on disk that a marginal `CLAPBAC` score was investigated, and it may actually be useful to an administrator to know about the presence of the `Nmap` executable on a client machine if it is not supposed to be installed there. The

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

agent will learn through our penalization of action a_6 with a reward of -5 that exit status 1 means something did not succeed on the source VM the agent logged onto via `ssh` from `clapbac-server`. RL training unfolds on a packet-by-packet basis, allowing for the training of an independent blue agent for each value of the `CLAPBAC` score; experimentally, rounding to the hundreds place provides a higher resolution than what would be needed for deployment. The specific rewards available at decision epoch t for specific actions can be found in Table 1.

Table 1. **Blue Agent Actions and Rewards**. The specific actions the blue agent can perform at decision epoch t along with specified rewards for actions on networked devices within our scenario.

Action		Description	Reward
$(t = 0,\ s = s_0)$	assert malicious	correctly flagged from high `CLAPBAC` score	2.5
$(t = 0,\ s = s_0)$	assert malicious	erroneously flagged from high `CLAPBAC` score	-15
$(t = 0,\ s = s_0)$	assert benign	correctly flagged from low `CLAPBAC` score	2.5
$(t = 0,\ s = s_0)$	assert benign	erroneously flagged from low `CLAPBAC` score	-15
$(t = 0,\ s = s_0)$	count files	count files in root folder of source device	-15
$(t = 0,\ s = s_0)$	careful investigation	check source device to see whether a scan occurred	$5 - \langle t_{a_3} \rangle$
$(t = 0,\ s = s_0)$	sloppy investigation	`Nmap` was not installed on source device	$5 - \langle t_{a_4} \rangle$
$(t = 0,\ s = s_0)$	sloppy investigation	`Nmap` was installed on source device	$-2.5 - \langle t_{a_4} \rangle$
$(t = 1,\ s = s_1)$	exit status 0 "good"	return code 0 means `ssh` command worked	0
$(t = 1,\ s = s_1)$	exit status 1 "good"	return code 1 means `ssh` command worked	-5
$(t = 1,\ s = s_2)$	exit status 0 "good"	return code 0 means `ssh` command worked	0
$(t = 1,\ s = s_2)$	exit status 1 "good"	return code 1 means `ssh` command worked	-5
$(t = 1,\ s = s_3)$	exit status 0 "good"	return code 0 means `ssh` command worked	0
$(t = 1,\ s = s_3)$	exit status 1 "good"	return code 1 means `ssh` command worked	-5
$(t = 1,\ s = s_4)$	exit status 0 "good"	return code 0 means `ssh` command worked	0
$(t = 1,\ s = s_4)$	exit status 1 "good"	return code 1 means `ssh` command worked	-5
$(t = 2,\ s = s_5)$	do nothing	satisfied by failed sloppy investigation	0
$(t = 2,\ s = s_5)$	careful investigation	check source device to see whether a scan occurred	$5 - \langle t_{a_3} \rangle$
$(t = 3,\ s = s_2)$	exit status 0 "good"	return code 0 means `ssh` command worked	0
$(t = 3,\ s = s_2)$	exit status 1 "good"	return code 1 means `ssh` command worked	-5

3.3 RL Solution Approach

Our solution approach to the RL problem at hand employs hierarchical RL with agents trained using Monte Carlo methods.

3.3.1 Hierarchical RL

Hierarchical approaches to RL consider decomposable tasks that employ several different tiers of agents with different responsibilities. Here, we consider the detection of anomalous network activity as one major task and the development of an appropriate mitigation response as a second major task. Our `CLAPBAC` model may be considered to be a pre-trained agent which addresses the first major task. From this point of view, we are really training a multitude of blue subagents with RL, one for each value of the `CLAPBAC` score, to address the second major task. The key advantage of this approach is that it produces a helpful factorization of the state space for the entire problem, allowing for a convenient split of the whole into two manageable, linked subtasks.

3.3.2 Monte Carlo Methods

Monte Carlo methods allow us to generate sample transitions between states, thereby not requiring complete probability distributions of all possible transitions (which would be required to use an exact DP approach). In simple terms, Monte Carlo methods sample and average returns for each state-action pair needed to approximate optimal policies. The overall concept follows the idea of generalized policy iteration, where a value function is approximated for a current policy, a policy is then repeatedly improved with respect to the current value function, iterating until the approximate policy and approximate value function approach optimality.[24] We leverage the Monte Carlo Exploring Starts (MCES) algorithm, depicted in Algorithm 1. In the MCES algorithm, the action-value function, or Q-function, is estimated by averaging the Monte Carlo returns, and the policy is improved by

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

choosing actions that maximize the current estimate of the Q-function. Exploration is performed by "exploring starts," that is, each episode begins with a randomly chosen state and action and then follows the current policy.[24]

Algorithm 1: Monte Carlo Exploring Starts

Input: $\pi(s) \in A(s)$ (arbitrarily), $\forall s \in S$
$Q(s,a) \in \mathbb{R}$ (arbitrarily), $\forall s \in S$, $a \in A(s)$
$Returns(s,a) \leftarrow$ empty list, $\forall s \in S$, $a \in A(s)$
Output: $\pi \approx \pi^*$
for each episode do
  Choose $S_0 \in S$, $A_0 \in A(S_0)$ randomly such that all pairs have probability > 0
  Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  for *t=T-1,T-2,...,0* **do**
    $G \leftarrow \gamma G + R_{t+1}$
    if $S_t, At \notin \{S_0, A_0, S_1 1, A_1, \ldots, S_{t-1}, A_{t-1}\}$ **then**
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
      $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$
    end
  end
end

4. COMPUTATIONAL EXPERIMENTATION

We are now in a position to delve deeper into the details of our data collection run and subsequent RL training. Initially, we pre-trained our `CLAPBAC` model on a few days worth of captured background network traffic generated by `ConsoleUser`. For this pre-training, it was useful to move our packet captures out of `CyberVAN` to a separate GPU server. On a single `NVIDIA GeForce RTX 3090` GPU, training a `CLAPBAC` model took just a few hours. All of the RL training, on the other hand, was carried out on the twelve CPU core `clapbac-server` VM in no more than a few days time. In the end, addressing software engineering challenges related to the data collection process itself consumed the most significant fraction of our research time.

One of the most non-trivial aspects of our work was interfacing with specialized software responsible for running and controlling the in-scenario actions of our `CyberVAN` experiments from the command line of the Peraton Labs testbed. This orchestration program, `GameMaster`, was developed to provide a seamless, multi-purpose RL gym for `CyberVAN` emulation environments. Despite the relative maturity of `CyberVAN` as a network emulation software suite,[32] we believe `GameMaster` to be the first program of its kind. `GameMaster` can either instruct scenario VMs to run commands sequentially in the foreground or persistently in the background. The latter functionality is what is required for the persistent `CLAPBAC` monitoring of the virtual network. `GameMaster` is also built with the concept of episodes hardcoded to allow for the convenient organization of RL. During an experiment, the flow of control may pass seamlessly from VM to VM, conditioning on the previous in-scenario action, by virtue of the message-passing capability of `GameMaster`. A schematic of the program's operation with further detail is provided in Figure 6.

Instead of using Algorithm 1 verbatim for the RL training of the blue subagents, we elected to instead collect data containing 15,000 scanning events in parallel over five independent instances of our `CyberVAN` scenario using the testbed resources. Before commencing with our full-scale production run, it was necessary to look at a smaller data sample, focusing on a random selection of smaller, intermediate, and larger `CLAPBAC` scores in the range [3.5,8] to evaluate whether our reward signal would allow for an adequate trade-off between immediately asserting maliciousness or benignness and performing investigations; originally, we had considered assigning an award of 5 to subagents that successfully carried out actions a_0 or a_1, but this did not end up leading to a reasonable number of investigations being performed in marginal situations.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

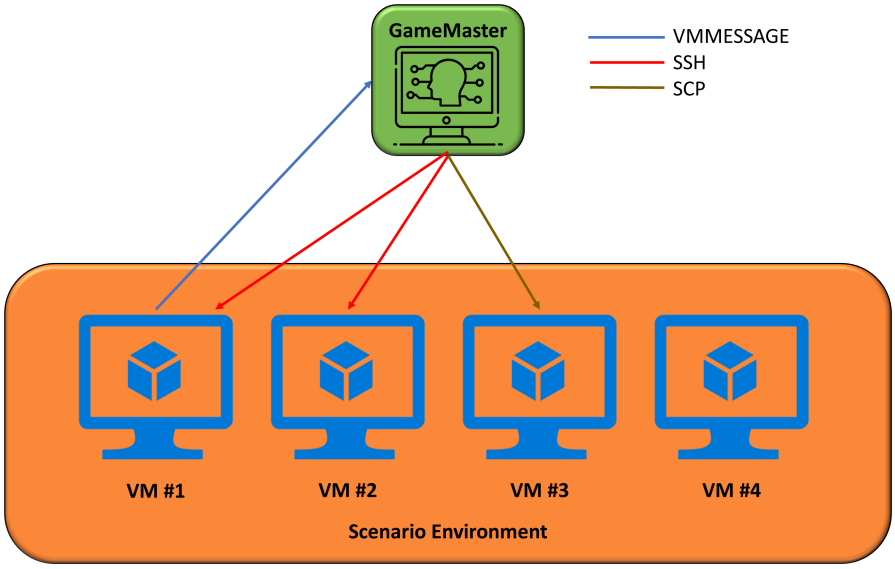

Figure 6. A schematic illustration of the functionality of the `GameMaster` program responsible for orchestrating runs of our experiments in `CyberVAN`. Its ability to intercept messages from scenario VMs in real time to trigger actions and subsequent transitions to new system states was integral to the success of our research experimentation.

After data collection was complete, we estimated the value of each state-action pair using our data set offline without performing policy improvement at intermediate stages. Training offline was possible by virtue of the fact that it is easy to record the time stamps of the `Nmap` scans when they occur in-scenario and align them with the time stamps of the `CLAPBAC` packet meta data after the fact in order to assign ground truth values to each packet record. With labeled packet data in hand, it is immediate to determine what the outcome would be for each of the blue agent's actions when in any possible state because both $\langle t_{a_3} \rangle$ and $\langle t_{a_4} \rangle$ can be accurately measured in less than an hour as a preliminary.

While our unorthodox line of attack might at first sight seem suboptimal, an offline hybrid approach was worthwhile because it allowed for the aggressive use of experience replay and saved on establishing thousands of slow `ssh` connections from `clapbac-server` to user VMs in-scenario. For the experience replay, individual packets scored by `CLAPBAC` were used twenty times each with potentially different randomly-chosen Monte Carlo trajectories. Milking the collected data in this manner allowed us to obtain a highly-accurate valuation of the entire state-action space far more quickly. To construct a complete policy, for a handful of `CLAPBAC` scores where a sparsity of data made real RL training impossible, it was necessary to extrapolate the learned policy from nearby values of the `CLAPBAC` score. In such extreme cases, we found it appropriate to manually assign an initial action a_1 (assert benign) to define a policy for scores below a threshold of 5.33 and an initial action a_0 (assert malicious) to define a policy for scores above a threshold of 5.33. While we had a particular value of the threshold in mind from legacy experimentation, manual policy construction was only actually triggered out on the tails of the `CLAPBAC` score distribution (see caption of Figure 7), ensuring that the above-described procedure cannot be sensitive to the precise value of the threshold chosen.

5. RESULTS AND DISCUSSION

Upon completing the experimentation and RL training described above, we can simply read off the optimal policy for each value of the `CLAPBAC` score by selecting the state-action pair which maximizes the Q-function at each state visited by the blue subagent associated to the given `CLAPBAC` score as its optimal trajectory unfolds. For exceptionally small values of the `CLAPBAC` score, the optimal policy is to assert that the packet is benign immediately and for exceptionally large values of the `CLAPBAC` score the optimal policy is to assert that the

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

packet is malicious immediately. For intermediate values of the CLAPBAC score, the subagents learn that the rigorous investigation provided by action a_3 is best (see Figure 7 below for a visualization). Most probably, no incentive ever existed in our experimentation for a subagent to prefer the cursory investigation provided by action a_4 because it turns out that the difference between $\langle t_{a_3} \rangle$ and $\langle t_{a_4} \rangle$ is not appreciably different from zero; while not evident to us at the outset, it was the case that both $\langle t_{a_3} \rangle$ and $\langle t_{a_4} \rangle$ were completely dominated by the overhead required to establish a `ssh` connection from `clapbac-server` to a user VM.

After hours of testing, no erroneous actions were observed, suggesting that the combination of CLAPBAC and our blue subagents result in a model nearly perfect in its performance. Furthermore, our deployment testing revealed that investigations are rarely triggered by scanning events. Initially, the non-triviality of our experimentation was thrown into question when we discovered that a human could easily identify the packets originating from Nmap in the CLAPBAC meta data by eye if the scans were done in a naive way; it turns out that the default packet lengths for the SYN scans initiated by Nmap are not coincident with the packet lengths of any of the default background traffic generated by the ConsoleUser utility in CyberVAN. Fortunately, upon further investigation, we found that the packet length must not have been the only handle available to our CLAPBAC model, as our trained agent performed flawlessly in deployment even after our Nmap scans were performed more carefully during training by tacking on a random amount of filler to the tail end of each packet sent out by Nmap to randomize their lengths.

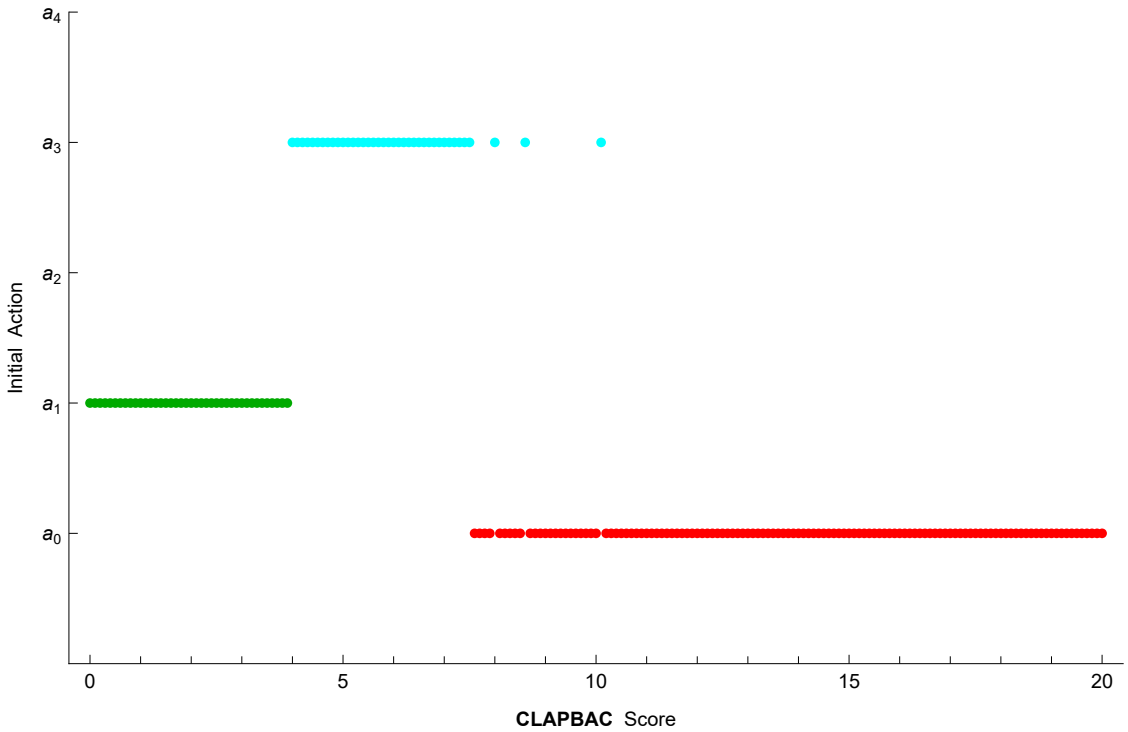

Figure 7. Which of the initial actions enumerated in Figure 5 are selected by the trained blue subagents associated to each value of the CLAPBAC score; for clarity, only every tenth data point in our set is plotted above. In practice, it is never advantageous to initially select actions a_2 or a_4 and CLAPBAC scores larger than 10 or smaller than .25 are uncommon.

6. CONCLUSION

Our principal goal in undertaking this research was to demonstrate that it is possible to distill useful cybersecurity playbooks as a result of training models with RL in high-fidelity emulations of real-world cybersecurity scenarios. As noted in the introduction, a playbook for a given scenario is closely related to but more flexible than the optimal policy learned via RL. A human-in-the-loop (*i.e.*, CSOC analyst) could look at the optimal policy depicted in Figure 7 and immediately distill the playbook where the actions a_1 (assert benign) and a_0 (assert

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

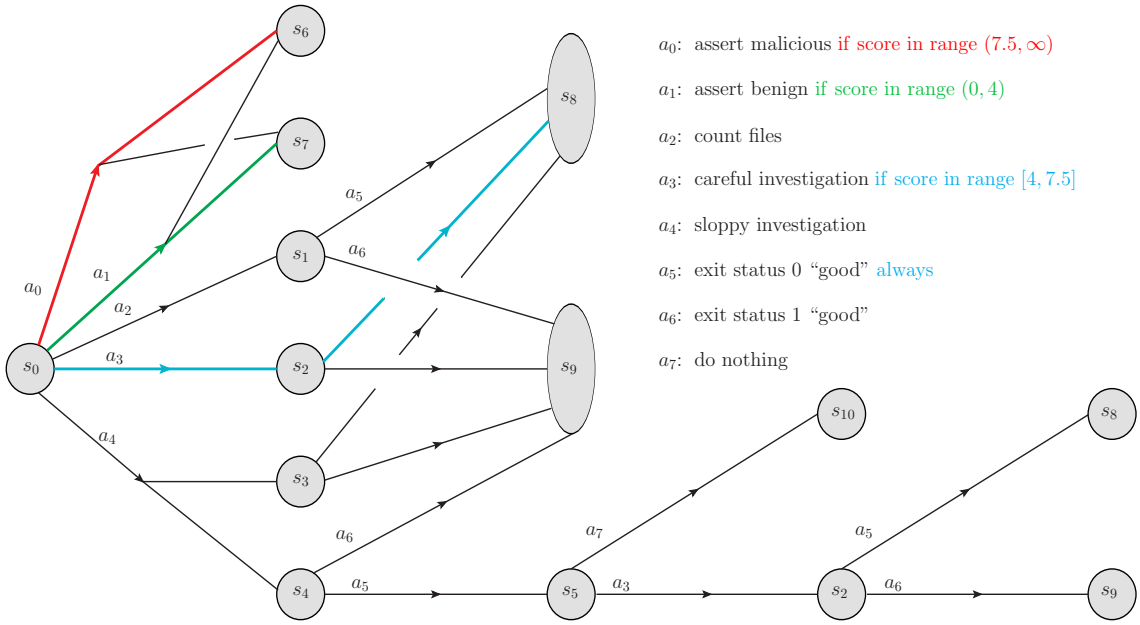

Figure 8. Playbook extracted from the learned policy of Figure 7, superimposed on the state-action diagram of Figure 5.

malicious) would be taken on either extreme, with rigorous investigations ordered for packets with `CLAPBAC` scores in the interval $[4, 7.5]$; this playbook is a curation of the optimal policy (state-action pairing) which would likely deliver nearly identical performance in practice (see Figure 8 for a visualization). In future work, it would be of interest to develop new scenarios in which the blue agents would inspire the synthesis of richer, dynamic playbooks, possibly by allowing the strategies of the red agents to drift once they start being effectively countered by the blue agents during RL training. It is also worth investigating whether a more organic approach which avoids fine-tuning MDPs is achievable by specifying only terminal rewards in the state-action diagram. At the same time, further fundamental research into non-stationary RL is required to overcome limitations of our current paradigm. Indeed, we find it improbable that a comprehensive solution to the dynamic DCO playbook generation problem will be forthcoming otherwise. While it is not yet clear precisely what RL strategy will ultimately win out, non-stationary RL remains an important area of active research in its own right[33,34] with the potential to be a game-changer in the domain of cybersecurity.

Acknowledgments

We thank Peraton Labs for the usage of `CyberVAN` testbed resources for the data collection and RL training aspects of this work. This work was supported in part by the U.S. Army Combat Capabilities Development Command (DEVCOM) C5ISR Center under Support Agreement No. USMA21056. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Military Academy, the United States Army, the Department of Defense, or the United States Government.

REFERENCES

[1] Cichonski, P., Millar, T., Grance, T., and Scarfone, K., "Computer security incident handling guide," Tech. Rep. NIST Special Publication (SP) 800-61, Rev. 2, National Institute of Standards and Technology, Gaithersburg, MD (2012).

[2] Bartock, M., Cichonski, J., Souppaya, M., Smith, M., Witte, G., and Scarfone, K., "Guide for cybersecurity event recovery," Tech. Rep. NIST Special Publication (SP) 800-184, National Institute of Standards and Technology, Gaithersburg, MD (2016).

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

[3] Johns Hopkins University Applied Physics Laboratory, "Introduction to Integrated Adaptive Cyber Defense (IACD) Playbooks," Retrieved from https://www.iacdautomate.org/intro-to-playbooks-and-workflows (2019).

[4] Wigness, M., Abdelzaher, T., Russell, S., and Swami, A., "Internet of battlefield things: Challenges, opportunities, and emerging directions," in [*IoT for Defense and National Security*], Douglass, R., Gremban, K., Swami, A., and Gerali, S., eds., pp. 5–22, IEEE Press, John Wiley & Sons (2023).

[5] Ross, R. J., "Information advantage activities: A concept for the application of capabilities and operational art during multi-domain operations," *The Cyber Defense Review* **6**(4), pp. 63–74 (2021).

[6] Adawadkar, A. M. K. and Kulkarni, N., "Cyber-security and reinforcement learning — a brief survey," *Engineering Applications of Artificial Intelligence* **114**, 105116, pp. 1–18 (2022).

[7] Dutta, H. and Biswas, S., "Distributed reinforcement learning for scalable wireless medium access in IoTs and sensor networks," *Computer Networks* **202**, 108662, pp. 1–14 (2022).

[8] Yang, L., Yao, H., Wang, J., Jiang, C., Benslimane, A., and Liu, Y., "Multi-UAV-enabled load-balance mobile-edge computing for IoT networks," *IEEE Internet of Things Journal* **7**(8), pp. 6898–6908 (2020).

[9] Wang, W., Sun, D., Jiang, F., Chen, X., and Zhu, C., "Research and challenges of reinforcement learning in cyber defense decision-making for intranet security," *Algorithms* **15**(4), 134, pp. 1–23 (2022).

[10] Han, Y., Rubinstein, B. I. P., Abraham, T., Alpcan, T., de Vel, O. Y., Erfani, S. M., Hubczenko, D., Leckie, C., and Montague, P., "Reinforcement learning for autonomous defence in software-defined networking," in [*9th International Conference on Decision and Game Theory for Security*], Bushnell, L., Poovendran, R., and Başar, T., eds., pp. 145–165, Lecture Notes in Computer Science vol. 11199, Springer, Cham. (2018).

[11] Wu, J., Fang, B., Fang, J., Chen, X., and Tse, C. K., "Sequential topology recovery of complex power systems based on reinforcement learning," *Physica A: Statistical Mechanics and its Applications* **535**, 122487, pp. 1–13 (2019).

[12] Wei, F., Wan, Z., and He, H., "Cyber-attack recovery strategy for smart grid based on deep reinforcement learning," *IEEE Transactions on Smart Grid* **11**(3), pp. 2476–2486 (2020).

[13] Ganesan, R., Jajodia, S., and Cam, H., "Optimal scheduling of cybersecurity analysts for minimizing risk," *ACM Transactions on Intelligent Systems and Technology* **8**(4), 52, pp. 1–32 (2017).

[14] Shah, A., Ganesan, R., Jajodia, S., and Cam, H., "An outsourcing model for alert analysis in a cybersecurity operations center," *ACM Transactions on Intelligent Systems and Technology* **14**(1), 2, pp. 1–22 (2020).

[15] Shah, A., Ganesan, R., Jajodia, S., and Cam, H., "Optimal assignment of sensors to analysts in a cyber security operations center," *IEEE Systems Journal* **13**(1), pp. 1060–1071 (2019).

[16] Ganesan, R., Jajodia, S., Shah, A., and Cam, H., "Dynamic schedule of cybersecurity analysts for minimizing risk using reinforcement learning," *ACM Transactions on Intelligent Systems and Technology* **8**(1), 4, pp. 1–21 (2016).

[17] Molina-Markham, A., Miniter, C., Powell, B., and Ridley, A., "Network environment design for autonomous cyberdefense," (2021).

[18] Molina-Markham, A., Winder, R. K., and Ridley, A., "Network defense is not a game," (2021). Retrieved from https://arxiv.org/pdf/2104.10262.pdf.

[19] Hore, S., Shah, A., and Bastian, N. D., "An artificial intelligence-enabled framework for optimizing the dynamic cyber vulnerability management process," *Proceedings of the 39th International Conference on Machine Learning*, pp. 1–21 (2022) (2022).

[20] Hore, S., Shah, A., and Bastian, N. D., "`Deep VULMAN`: A deep reinforcement learning-enabled cyber vulnerability management framework," *Expert Systems with Applications*, 119734, pp. 1–17 (2023).

[21] Puterman, M. L., [*Markov Decision Processes Discrete Stochastic Dynamic Programming*], Wiley Interscience (2005).

[22] Bertsekas, D. P., [*Dynamic Programming and Optimal Control*], vol. 1, Athena Scientific, 3rd ed. (2005).

[23] Powell, W. B., [*Reinforcement Learning and Stochastic Optimization*], Wiley Interscience (2019).

[24] Richard S. Sutton, A. B., [*Reinforcement Learning, An Introduction*], MIT Press, 2nd ed. (2020).

[25] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I., "Attention is all you need," *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6000–6010 (2017).

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

[26] Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M., Krishna, R., Kuditipudi, R., Kumar, A., Ladhak, F., Lee, M., Lee, T., Leskovec, J., Levent, I., Li, X. L., Li, X., Ma, T., Malik, A., Manning, C. D., Mirchandani, S., Mitchell, E., Munyikwa, Z., Nair, S., Narayan, A., Narayanan, D., Newman, B., Nie, A., Niebles, J. C., Nilforoshan, H., Nyarko, J., Ogut, G., Orr, L., Papadimitriou, I., Park, J. S., Piech, C., Portelance, E., Potts, C., Raghunathan, A., Reich, R., Ren, H., Rong, F., Roohani, Y., Ruiz, C., Ryan, J., Ré, C., Sadigh, D., Sagawa, S., Santhanam, K., Shih, A., Srinivasan, K., Tamkin, A., Taori, R., Thomas, A. W., Tramèr, F., Wang, R. E., Wang, W., Wu, B., Wu, J., Wu, Y., Xie, S. M., Yasunaga, M., You, J., Zaharia, M., Zhang, M., Zhang, T., Zhang, X., Zhang, Y., Zheng, L., Zhou, K., and Liang, P., "On the opportunities and risks of foundation models," (2021). Retrieved from https://arxiv.org/pdf/2108.07258.pdf.

[27] Merritt, R., "What is a transformer model?," Retrieved from https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/ (2022).

[28] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K., "`BERT`: Pre-training of deep bidirectional transformers for language understanding," (2019). Retrieved from https://arxiv.org/pdf/1810.04805.pdf.

[29] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I., "Improving language understanding by generative pre-training," (2018). Retrieved from https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[30] Steverson, K., Carlin, C., Mullin, J., and Ahiskali, M., "Cyber intrusion detection using natural language processing on Windows event logs," *2021 International Conference on Military Communication and Information Systems*, pp. 1–7 (2021).

[31] Hampton, D., Steverson, K., Mullin, J., and Doersch, M., "Self-supervised transformer models for real-time anomaly detection in raw packet headers," *2022 IEEE Military Communications Conference*, in press (2022).

[32] Chadha, R., Bowen, T., Chiang, C.-Y. J., Gottlieb, Y. M., Poylisher, A., Sapello, A., Serban, C., Sugrim, S., Walther, G., Marvel, L. M., Newcomb, E. A., and Santos, J., "`CyberVAN`: A cyber security virtual assured network testbed," *2016 IEEE Military Communications Conference*, pp. 1125–1130 (2016).

[33] Chandak, Y., Theocharous, G., Shankar, S., White, M., Mahadevan, S., and Thomas, P. S., "Optimizing for the future in non-stationary MDPs," *Proceedings of the 37th International Conference on Machine Learning*, pp. 1414–1425 (2020).

[34] Chandak, Y., Shankar, S., Bastian, N. D., da Silva, B., Brunskill, E., and Thomas, P. S., "Off-policy evaluation for action-dependent non-stationary environments," *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*. Retrieved from https://arxiv.org/pdf/2301.10330.pdf (2022).

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 14 Jun 2023
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use