Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

OFFLINE REINFORCEMENT LEARNING FOR AUTONOMOUS CYBER DEFENSE AGENTS

Alexander Wei[1], David Bierbrauer[1], Emily Nack[1], John Pavlik[1], and Nathaniel Bastian[1]

[1]Army Cyber Institute, United States Military Academy, West Point, NY, USA

ABSTRACT

Advanced Persistent Threats (APTs) present an evolving challenge in cybersecurity through increasingly sophisticated behavior. Cybersecurity Operations Centers (CSOCs) rely on standard playbooks to respond to myriad cyber threats; however, such methods quickly become outdated, leaving sensitive data at significant risk for theft and exploitation. To provide CSOCs with an advantage, playbooks capable of adapting to threats and environments must be developed for use with modern security orchestration and automation tools. Our methodology proposed herein trains autonomous cyber defense agents through offline reinforcement learning (RL) to address this need. Using the scalable, tailorable Cyber Virtual Assured Network, we simulate an APT conducting data exfiltration to then train an agent using offline RL and compare performance against a myopic policy. Initially, we see improvements in preventing exfiltration while ensuring authorized user access, but it is clear the agent must periodically retrain to account for changing adversarial behaviors.

1 INTRODUCTION

Threats to government, corporate, and other private networks continue to grow in complexity and scale as malicious actors attempt to exploit vulnerabilities for a variety of purposes. In a report from the Federal Bureau of Investigation (FBI), global losses in 2023 rose to $12.5 billion, which increased from $3.5 billion in 2019 (Internet Crime Complaint Center 2024). Such losses impact individual users, along with critical infrastructure and other entities, that organizations and governments rely on for daily activities, as they stem from a wide variety of malicious behaviors and attack vectors. Although the FBI report outlines some specific vectors from complaints received, it does not address the potentially complex actions of Advanced Persistent Threats (APTs) that may act with more nefarious purposes. APTs have become more prominent and are characterized as being well-funded – possibly by nation-states – and acting deliberately to avoid detection while accomplishing their goal. Additionally, APTs will adapt to defensive efforts while repeatedly pursuing their objective (Alshamrani et al. 2019). As a result, much demand is placed on Cybersecurity Operations Centers (CSOCs) to prevent, detect, and respond to these numerous threats.

CSOCs typically employ playbooks to assist professionals in handling their increasing workload. Playbooks for defensive cyber operations (DCO) are typically derived from regulatory and organizational policies, and are further fine-tuned by CSOCs into specific workflows for a given environment (Johns Hopkins University Applied Physics Laboratory 2019). Such processes result in relatively static playbooks that only receive updates when needed and may be reactionary in nature, as opposed to proactive. Such static playbooks, coupled with the increasing demand on CSOCs to defend against APTs and other threats, inevitably lead to significant gaps in cybersecurity.

Modern technologies and methods may help CSOCs close such gaps, by enabling the creation of sophisticated dynamic playbooks for DCO. Security Orchestration and Automated Response (SOAR) platforms provide a first step for efficient allocation of resources and effective response workflows, but still require expertise in building and updating SOAR playbooks. Meanwhile, previous work has shown the potential value of reinforcement learning (RL) to craft dynamic playbooks. One such method employed a hierarchical RL framework to respond to network mapping attempts by a malicious user attempting to avoid detection, allowing a cybersecurity professional to conduct appropriate actions based on the output of a transformer model (Bierbrauer et al. 2023). Combining such a method to develop playbooks for a SOAR platform would certainly help close cybersecurity gaps; however, several limitations existed within the RL framework: it operated within a small network in which the detector model had full visibility of the network environment; the detector was only capable of detecting a specific type of malicious behavior; and it only examined a limited selection of agent policies.

This paper presents a more comprehensive and realistic solution to the dynamic playbook problem for modern environments. Specifically, it addresses a scenario of particular interest to modern military environments: the need to enable access to core services for joint and coalition partners without full visibility of the network and its data. By utilizing a sophisticated simulation environment in the Cyber Virtual Assured Network (CyberVAN) environment, we demonstrate the capability for an agent to learn optimal policies that outperform static, myopic playbooks. As such, this work makes the following contributions:

- Develops a scalable and realistic network environment scenario implemented in CyberVAN.
- Implements an advanced simulation controller, known as the "Game Master" to enable data collection from actual command line interface (CLI) commands conducted by automated agents.
- Revolutionizes playbooks by providing optimal policies based on RL and evolving APT strategies.

We present our work through a realistic military scenario from an APT focused on stealthily exfiltrating data from a mission partner network (a term used to describe an interconnected network with multiple nations involved). The rest of this paper is organized as follows. Section 2 provides a brief overview of recent RL advancements, as well as those specific to cybersecurity. In Section 3, we describe our scenario in detail, along with the problem formulation, data collection, and solution frameworks. We then discuss our findings in Section 4, before reflecting on our work and presenting future work in Section 5.

2 LITERATURE REVIEW

Applications of artificial intelligence (AI) and machine learning (ML) for cybersecurity range from detection to mitigation. Detection problems primarily concern themselves with intrusion, malware, or fraud detection (Ozkan-Okay et al. 2024). For example, Willeke et al. (2023) utilized neural networks to classify malicious traffic from network traffic payloads in a federated setting; meanwhile, malware detection methods that can detect variants through their behavior have also achieved a high level of accuracy (Aslan and Yilmaz 2021). These applications, however, comprise only one step of the overall DCO playbook problem and do not provide any insight into mitigation or prevention procedures.

RL has shown potential in expanding upon malicious behavior detection in the cyber domain. Deep RL has demonstrated efficiency in job scheduling, networking, and routing – which are inherently diverse in domain-specific applications – but have gained popularity in cybersecurity to enhance mitigation techniques (Ozkan-Okay et al. 2024). For instance, Aref et al. (2017) proposed a multi-agent approach to anti-jamming communications in radios to learn a sub-band selection policy to improve device availability. Such applications appear to focus more on highly-specific scenarios in edge-style networks, therefore only touching on appropriate mitigations for APTs. Other RL approaches include development of optimal policies for resource allocation in CSOCs (Hore et al. 2023), which aimed to decrease the gap between APTs and CSOCs by optimizing response to vulnerabilities through appropriate mitigation. Indeed, such work encapsulates more of the overall playbook process, but still does not provide cybersecurity analysts and operators with full workflows for a given threat.

To examine the full threat life cycle in more detail, Wang et al. (2022) described a framework supporting a notion that each step should be treated as a different decision-making task. This framework may be useful in some scenarios where certain tasks may not directly follow from previous ones, but could lead to undesirable results in the long run.

As an example, specific decisions made during a response task could impede recovery efforts, although such decisions were considered optimal at the time. This of course served as inspiration for our previous work – which we discussed in Section 1 – that considered an autonomous agent capable of understanding output from a detection model and responding appropriately (Bierbrauer et al. 2023). Despite its limitations, it showed the feasibility of developing playbooks from an agent's optimal policy.

3 METHODOLOGY

3.1 Scenario Overview

Our work centers on the development of a sophisticated network environment aimed at training an autonomous cyber defense agent to detect and respond to malicious activities, particularly focusing on data exfiltration threats within a simulated military mission setting. Our overarching objective is to leverage this network environment to generate diverse and realistic data that will train an agent to recognize and respond effectively to data exfiltration attempts. To achieve this, we developed a red team playbook specifically designed to simulate real-world cybersecurity threats, providing a comprehensive framework to aid our blue team agent's understanding of the risks associated with data exfiltration and proper mitigation techniques.

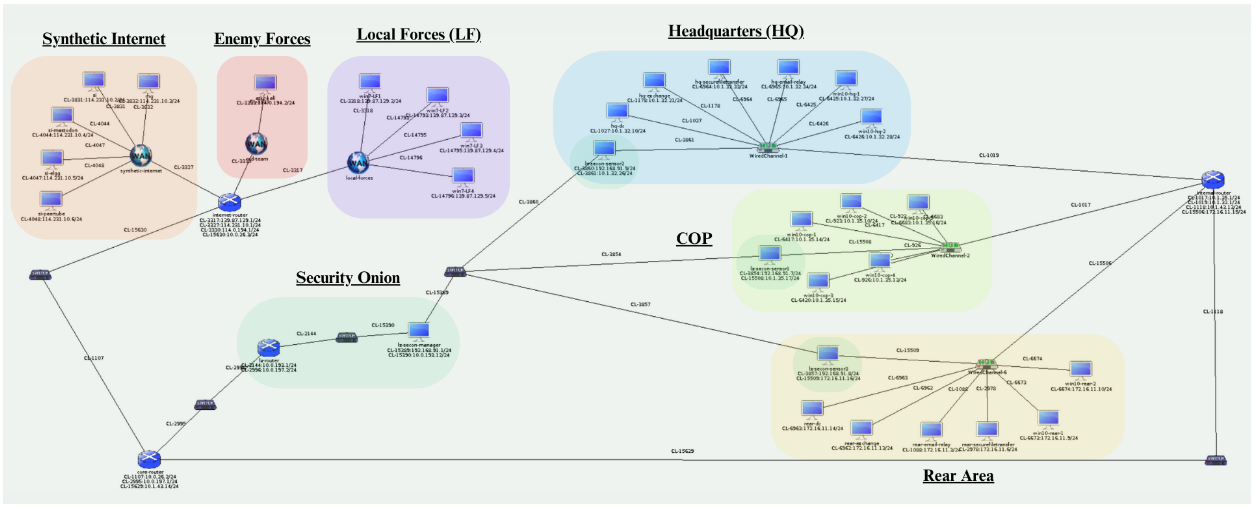


Figure 1: A diagram of the CyberVAN network environment for simulation/emulation.

We utilized CyberVAN (Chadha et al. 2016) to simulate/emulate a realistic, high-fidelity network environment. CyberVAN's advanced discrete-event-simulation and emulation capabilities provide a robust framework for cybersecurity experimentation, operational planning, training, and validation. We construct a realistic tactical network instrumental in the RL training of an agent. The testbed's scalability and flexibility were pivotal in our experimentation, enabling us to mimic real-world cybersecurity scenarios. Our simulated tactical network is inspired by modern mission environments, and comprises several distinct subnets as illustrated in Figure 1, each representing different aspects of the mission partner network:

- **Headquarters (HQ) and Rear Subnets:** These subnets are characterized by a mix of operating systems and devices, including domain controllers, exchange servers, secure file transfer services, email relay servers, and Windows 10 hosts. HQ also hosted a MySQL server with weak admin credentials, which was targeted during experimentation. To enhance monitoring and alerting capabilities, Security Onion sensors were deployed within these subnets, shipping data directly to the Security Onion manager. This setup facilitated real-time monitoring and alerting for potential security threats.

- **Local Force (LF) Subnet:** This subnet is populated with Windows 7 machines, aiming to reflect the presence of older systems in a tactical setting. A common feature across all LF devices is the installation of MySQL client software, essential for simulating benign database interactions. Notably, this subnet does not include Security Onion sensors. This provides a realistic view of a network environment where communication between networks must be allowed, but data privacy for a partner network must be maintained.
- **Command and Control Outpost (COP) Subnet:** Composed entirely of Windows 10 machines, this subnet is equipped with MySQL client software and Elastic Beats agents to ship logs to Security Onion. Similar to the HQ and Rear subnets, Security Onion sensors were also deployed within the COP subnet, ensuring comprehensive monitoring and alerting.
- **Enemy Forces (EF) Subnet:** This subnet is unique in its composition, featuring a Kali Linux machine to simulate a potential adversary within the network environment. This setup allowed for the exploration of cybersecurity defenses against sophisticated attack vectors, and it is important to note that our blue agent is unaware of this subnet's presence initially and cannot observe the specific enemy workstation.

In addition to these subnets, the network also includes a synthetic internet to generate background traffic, mimicking the constant flow of data in a real operational setting. This setup provided us with a robust platform for conducting experimentation, allowing us to simulate real-world scenarios and generate data to help our agent develop effective strategies for enhancing network security.

Our primary threat scenario is an insider threat detailed within the red team playbook. Here, we simulate a situation where an employee, threatened and coerced by an external adversary, uncovers a critical security oversight: the use of weak MySQL credentials. The insider threat, under duress, agrees to access and exfiltrate classified operational data, all while minimizing the risk of detection. This scenario highlights the risks posed by insider threats exploiting legitimate access to sensitive information. The setup enables us to simulate both malicious and benign activities across the network, emphasizing the need to develop robust strategies to mitigate insider threats effectively.

The insider threat scenario involved 10 devices across the COP, LF, EF, and HQ subnets, as detailed in Table 1. Win10-HQ-1 hosted our MySQL server, accessible by both COP and LF. Hosts within the LF subnet had restricted MySQL credentials, limiting their actions to specific tables and activities, simulating benign user interactions in the database. In contrast, the insider threat, located in the COP subnet, had unauthorized administrative access to the MySQL server, granting full control over databases and tables. To covertly exfiltrate data, we employed ping exfitration, a technique where Internet Control Message Protocol (ICMP) packets containing the exfiltrated data were disguised as normal ping requests. These ICMP packets were sent to the atkr-kali device within the EF subnet, using the communication channel to transfer sensitive information. During experimentation, we randomized the targeted insider threat host to ensure variability in the training data for our agent and prevent bias. This approach enables the agent to adapt to different scenarios and devise effective strategies for mitigating threats across various network components, leading to a more comprehensive and robust cybersecurity solution.

Table 1: A breakdown of the devices used for the insider threat scenario.

COP	**LF**	**EF**	**HQ**
Win10-Cop-1	Win7-LF1	atkr-kali	Win10-HQ-1
Win10-Cop-2	Win7-LF2		
Win10-Cop-3	Win7-LF3		
Win10-Cop-4	Win7-LF4		

Navigating the intricacies of our dynamic threat scenario and simulated mission environment reveals a formidable challenge, necessitating the use of RL techniques. Our environment presents a sequential

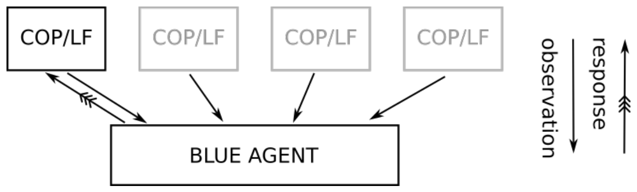


Figure 2: A depiction of the device orchestration and observation collection cycle.

decision problem under uncertainty, requiring strategies to effectively manage incomplete observations. Utilizing the scenario and CyberVAN simulation described above, we collect data to effectively employ offline RL and construct a model of the environment's dynamics that can be explored outside of simulation runtime. Methods like these equip an agent with improved capabilities to detect and respond to data exfiltration attempts. Next, we define our methodology for formulating a Partially Observable Markov Decision Process (POMDP) model and implementing offline RL, then discuss how we integrate this approach with the data generated within our simulation environment to train and evaluate the performance of a blue agent in preventing data exfiltration.

3.2 Partially Observable Markov Decision Processes

We describe here the space of actions $\mathscr{A}$, observations Z, and states S that we implement within the POMDP model. A mixture of benevolent and malicious devices $\{d_1,\ldots,d_n\}$ is present on a network that spans multiple subnets. These devices share access to a MySQL server, and are originators of m distinct shell commands. Device commands are orchestrated with a Game Master, as described in Section 3.4.

We train an intelligent blue agent using a value iteration algorithm to identify the best actions to take, given observations. Actions were executed within the simulation, by selecting a viable device d at random from the network. Figure 2 shows the sequence of gameplay, allowing devices to perform an action, resulting in an observation and accompanying blue agent action.

Action Space - The set $\mathscr{A}$ of blue agent actions:

1. (Do Nothing)
2. (Change User Password) Change the MySQL password of the user associated with device d
3. (SQL Server Shutdown) Perform a physical shutdown of the device hosting MySQL server
4. (Consult LF) Consult an operator to selectively change the password of a user associated with device d with improved accuracy at the cost of response time

State Space - The state space for the blue agent reflects the entire network and is encoded $[S_{d_1}\cdots S_{d_n}]$. For each device $S_d = \mathbb{Z}_m \times \mathbb{Z}_2 \times \mathbb{Z}_2$ is formed by pairing device CLI commands with their outcomes $\{\text{success}, \text{failure}\}$ and benevolence characteristic $\{\text{benign}, \text{malicious}\}$.

Observation Space - The space of observations $Z = \mathbb{Z}_m$ corresponds to the m unique shell commands that are recorded on each execution and traceable through discrete Security Onion alerts. We form $z \in Z$ by removing from $s \in S$ the success and device characteristics. Where T is the function that strips device and success characteristic from a state, we thus define $z_{t_1} = T(s_{t_1})$. This induces the emission function seen in Table 3, and the highlighted observable components of states indicated in Tables 2-4. Therefore, the space of observations is the range of emissions $T(S)$.

Reward - We define a point-wise reward function $R(s,a)$ to penalize adverse events, represented in Table 2. In offline RL, this takes the form of a model $\widehat{R}(s,a)$ computed from historical data generated using the reward profile in Table 4.

Our method relies on several passes through historical event data in order to construct empirical models of state transitions, observations, and reward. Other variants of the POMDP may make use of reward models conditioned on multiple previous states comprising n-grams of event history rather than the single-step digram approach we used. As with the standard MDP, the optimal value functions and the optimal policy can be obtained by solving the Bellman Equation for the expected value of a trajectory $V_{t_1}(s_1), V_{t_2}(s_2), \ldots$,

$$V_t(s) = \max_{a \in \mathscr{A}} \left\{ R_t(s_t, a) + \gamma \sum_{j \in S} P(j|s_t, a) V_{t+1}(j) \right\},$$

where

$$R_k = E_\pi \left[\sum_{i=k}^{\infty} \gamma^i R(s_{t_i}, a_{t_i}) \right], \tag{1}$$

for a reward model R, state transition model P, discount factor $0 < \gamma < 1$, and policy π. The introduction of an *emission function* of states conditioned on observations $P(s|z)$ extends this equation to the POMDP,

$$\begin{aligned} V_t(z) &= \max_{a \in \mathscr{A}} \left\{ E_{s_t}[R_t(s_t, a)] + \gamma \sum_{j \in S} E_{s_t}[P(j|s_t, a) V_{t+1}(j)] \right\} \\ &= \max_{a \in \mathscr{A}} \left\{ \sum_{q \in S} [R_t(q, a) P(q|z)] + \gamma \sum_{j \in S} \sum_{q \in S} [P(j|q, a) V_{t+1}(j) P(q|z)] \right\}, \quad z \in Z. \end{aligned} \tag{2}$$

With l actions, m belief states, and n true states, computing the expected value of a single trajectory for a single action at each belief state takes $\mathscr{O}(lmn^2)$ flops. Selecting the best action of these for each belief state thus requires $\mathscr{O}(lm^2n^2)$ flops for the conditioning on belief states, and comprises a single iteration. This is a substantial increase in the algorithm's complexity. In our work, however, we limit this growth through a direct embedding of the observation space within the fully descriptive state space, which is already restricted to discrete event indicators. We thus seek to present consistent and coherent policies learned for our scenario among the explainable and embedded belief states in the data exfiltration scenario.

Tables 2-4 display how observations are fully embedded within a larger state space. Furthermore, the success or failure of CLI commands on any device corresponds to a well-defined state characteristic measuring both red team and benevolent actor progression through the exfiltration playbook. As a consequence of blue agent controller actions, certain states such as *lf-search-db-failure* are captured implicitly by incurring negative rewards in other undesirable states. This allows for consistency in recording actions that physically disconnect device from network rendering us unable to collect status during simulation, and further reduces the number of POMDP states.

3.3 Offline Reinforcement Learning

3.3.1 Transition, Belief, and Reward Modeling

We train an intelligent agent offline, utilizing historical data from our simulations to build empirical models for state transitions $P(s'|s, a)$, observations $P(s|z)$, and reward $P(R(s, a))$. Such offline training has the benefit of requiring minimal computational power. In our case, the simulated data exhibited random agent policies, whereas the conditioned state diagram transition, observation, and reward point counts are computed and normalized to obtain distributions immediately prior to iteratively solving the Bellman equation (2) for the expected reward along action-state-transition trajectories.

Table 2: The POMDP reward model $\widehat{R}(s,a)$.

State	**Action**			
$s =$ {device}-{cmd}-{success}	**consult_lf**	**change_pw**	**nothing**	**shutdown-mysql**
red-echo-success	0	0	-0.012597	-0.00387
red-echo-failure	0	0	-0.012597	-0.00387
lf-format-success	0	-0.002191	-0.025193	0
lf-format-failure	0	-0.002191	-0.025193	0
red-systeminfo-success	-0.02003	-0.041624	-0.009448	-0.017028
red-systeminfo-failure	-0.02003	-0.041624	-0.009448	-0.017028
lf-mysql_search-success	-0.284170	-0.169782	-0.192100	-0.183437
red-mysql_search-success	-0.161489	-0.144589	-0.204697	-0.079721
lf-mysql_login-success	-0.239104	-0.223455	-0.396798	-0.202012
red-mysql_login-success	-0.088881	-0.065722	-0.425140	-0.020898
lf-cd-success	-0.001252	0.000000	0.000000	-0.000774
lf-exfiltrate-success	0	0	0	0
red-exfiltrate-success	-0.053830	-0.035052	-0.009448	-0.030960

Table 3: The space of observations $Z = T(S)$.

s = {device}-{cmd}-{success}	**Explanation of State** s	**Observation** $T(s)$
lf-mysql_login-success	Benevolent Login Success	mysql login
red-mysql_login-success	Malicious Login Success	mysql login
lf-mysql_search-success	Benevolent Enumerate Tables Success	mysql search
red-mysql_search-success	Malicious Enumerate Tables Success	mysql search
red-mysql_dump-success	Benevolent Save to File Success	mysql dump
red-mysql_dump-success	Malicious Save to File Success	mysql dump

Table 4: The reward points function $R(s,a)$.

State s			**Reward** $R(s,\cdot)$	**State** … s		**Reward** $R(s,\cdot)$
cmd	**Device**	**Success/Failure**	Reward	**Device**	**Success/Failure**	Reward
mysql_login	Benevolent	Success	0	Malicious	Failure	0
mysql_login	Benevolent	Failure	-1	Malicious	Success	-1
mysql_search	Benevolent	Success	0	Malicious	Failure	0
mysql_search	Benevolent	Failure	-1	Malicious	Success	-1
mysql_dump	Benevolent	Success	0	Malicious	Failure	0
mysql_dump	Benevolent	Failure	-2	Malicious	Success	-2
exfiltrate	Benevolent	Success	0	Malicious	Failure	0
exfiltrate	Benevolent	Failure	-3	Malicious	Success	-3

By conducting offline reinforcement learning, we invoke an expected trajectory $\widehat{s}_{t_1}, \widehat{s}_{t_2}, \ldots$ and value function estimate $\widehat{V}^{\pi}(s)$ in place of the explicit trajectory $s_{t_1}, s_{t_2}, \ldots$ of (1). Thus we derive a reward model used in POMDP, shown in Table 2, that can be applied uniformly to all devices on the network without knowing their identifying characteristics.

The distribution of reward points depends on previous state and action for a particular device d and is used to define the empirical reward function $\widehat{R}(s,a) := E_{\pi}(R(s,a))$. The summation of reward values across all devices $D = \{d_1, \ldots, d_n\}$ present in a simulation episode e produces the POMDP reward model,

$$R^e(S,a) = \sum_{d \in D} \widehat{R}(s,a)\,.$$

The values for this reward model are estimated over the reward points history of all games within an episode. These reward points are assigned according to Table 4, which equates a weight with a level of event severity. Explicitly, our reward model considers the average observed reward for actions and states,

$$P(R(s,a)) = \begin{cases} 1\,, & R(s,a) = E_{\pi}(R(s,a)) \\ 0\,, & \text{otherwise}\,, \end{cases}$$

whereas we obtain a transition model $P(s', s, a)$ and a belief model $P(s, z)$ by normalizing the occurrence counts across episodes to produce distribution functions, where $s \rightarrow s'$ for each action a, and $z \Rightarrow s$.

3.3.2 Data

We denote by game the interaction between an actor and the controller (blue agent). An episode consists of multiple games played during a time interval. Games are simulated utilizing a game master described in Section 3.4, recording the history of actor and controller events into a CSV log.

Each row that is entered into the log corresponds to a command executed by an actor. This reflects a change in the RL environment, so we record a minimal set of column values shown in Table 5 to be able to construct the POMDP space of observations, states, and actions that is used in value iteration. A command corresponds to a single actor and produces a new row in the CSV. However, this is augmented to include the complementary set of columns corresponding to the most recent actions of the other actors.

Table 5: A CSV episode history log produced by the Game Master.

Controller Action ,	**Actor *i* Command ,**	**Actor *i* Observation ,**	**Actor *i* Outcome ,**	⋯
Do Nothing ,	lf_mysql_login ,	mysql_login ,	Success ,	⋯
Change User Password ,	lf_mysql_login ,	mysql_login ,	Failure ,	⋯

Between each episode, we modify the actions available to the red agent for performing benign commands. It is important to note that this does not change the actions needed for the red agent to successfully exfiltrate data; rather, it produces different benign states for each episode so we may examine the blue agent's robustness.

3.4 Experimentation

3.4.1 Game Master and Event Orchestration

Central to our experimentation is the utilization of a Game Master, a pivotal software package designed to orchestrate events across hosts within CyberVAN and generate rich data for training and evaluating our RL agent. The Game Master operates in conjunction with custom scripts, allowing for the autonomous execution of actions on each host in the network.

The Game Master's functionality is encapsulated within a python script serving as the backbone of our agent environment. This script defines the state and action space, coordinates events across hosts, and handles the execution, monitoring, and verification of actions. It relies on a configuration file to specify various aspects of the simulation, such as actions performed by different entities (ex. red agent, blue agent, or green agent), simulation duration, and output directories for logs and results.

Additionally, we developed a suite of game scripts that automate various aspects of the simulation environment. Noteworthy scripts include a batch file for executing commands and logging execution details, a shell script for automating file transfers to multiple VMs, and other custom shell scripts for rapid prototyping and real-time testing of game scripts.

3.4.2 Agent Automation and Data Generation

Our framework enables the deployment of autonomous agents within CyberVAN, facilitating the generation of diverse and realistic data. Through python and bash scripting, custom actions can be built and executed autonomously on each host, enabling interactions with other hosts and the generation of relevant data points.

Specifically, we developed four main scripts to accomplish the following: manage and manipulate data related to the simulation environment, ensuring consistency and standardization; facilitate the coordination of controller actions across VMs and queuing of CLI commands to execute on devices; automate the monitoring and analysis of the simulation environment, parsing output, categorizing actions, and calculating rewards based on predefined criteria; and automate the process of running simulations, start necessary VMs, execute monitoring and game master scripts, and clean up the environment post-simulation. Commands are queued for native execution on devices within the turn-by-turn format specified by the game configuration. These actions are queued with a probability of success and will continue until the Game Master detects data exfiltration. Thus, rounds will vary in duration with random sequences of actions. These include the following:

1. (*mysql_login*) Log in to MySQL server from device d,
2. (*mysql_search*) Enumerate tables present on MySQL server,
3. (*mysql_dump*) Export MySQL table to a file on disk,
4. (*exfiltrate*) Exfiltrate the output of *mysql_dump* through ICMP packets.

3.4.3 Scoring and Evaluation

Scoring and evaluating the performance of our RL agent is a critical aspect of our experimentation. We utilize predefined criteria and reward mechanisms, calculated based on real-time success/failure return values from executed actions, to evaluate the efficacy of the agent's decision-making and response strategies. The agent's performance is compared to a myopic baseline policy. The myopic policy is a random selection from all available blue agent actions in $\mathscr{A}$ over all possible devices.

Recalling the data structure of games and episodes among actors and controllers defined in Section 3.3.2, we record the state, observation, and action history $X = (s_1, z_1, a_1), (s_2, z_2, a_2), \dots$ across multiple episodes $e_1, e_2, \dots$. The initial policy is computed using value iteration based on policy value estimates in the first episode e_1, denoted as $\widehat{V}^{\pi}(X_{e_1})$. Subsequent game rounds represent the evolution of the red agent's behavior, enabling us to assess policy degradation without deriving new policies. This data is logged in a structured CSV file generated by `monitoring.py` within the Game Master package, capturing crucial insights into device interactions and outcomes during simulation runs.

For each game round, we evaluate agent performance by it's ability to both allow normal behavior and stop malicious behavior. Equation (3) shows how we calculate Score_α, which corresponds to the true positive rate of the agent; that is, it measures an agent's ability to successfully identify and stop malicious data exfiltration:

$$\text{Score}_\alpha = 1 - \frac{\text{Number of Successful Exfiltration Activities}}{\text{Number of Attempted Exfiltration Activities}} . \tag{3}$$

Score_β in (4) on the other hand provides a measure of how well an agent identifies normal user behavior by calculating the true negative rate:

$$\text{Score}_\beta = 1 - \frac{\text{Number of Failed Benign Activities}}{\text{Number of Attempted Benign Activities}} . \tag{4}$$

To give an overall effectiveness score of an agent, we calculate a weighted average of Score_α and Score_β weighted by the number of non-trusted (m) and trusted (b) devices according to (5):

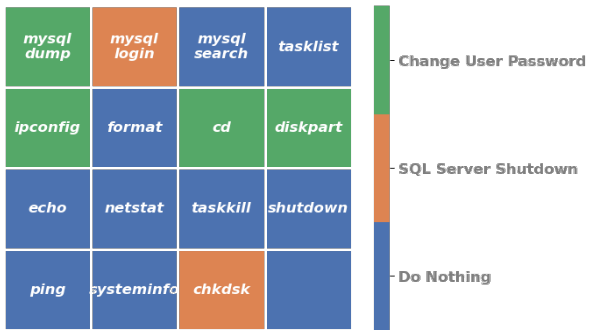
$$\text{Score} = \frac{\text{Number of Successful Outcomes}}{\text{Number of Activities}} = m\,\text{Score}_\alpha + b\,\text{Score}_\beta . \tag{5}$$

The number of non-trusted and trusted devices used for calculating Equation 5 is largely based on the network design and how much knowledge the blue agent possesses for each device. In our scenario, for example, the blue agent has more knowledge of all devices in the COP subnet and believes them to be trusted. The LF subnet, however, is not fully observable and thus each device is not implicitly trusted.

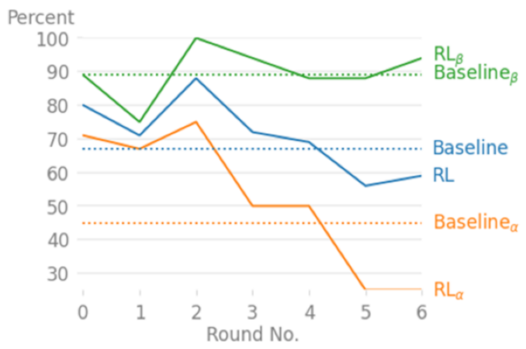
The scoring calculation is based on the state-action history logged during rounds. If specific states are met and appropriate actions are taken by the agent, weighted rewards are assigned. By analyzing these scores over multiple games, we gain valuable insights into the autonomous cyber defense agent's learning progress and decision-making capabilities within our simulated environment. Such insights allow us to assess risks over time associated with allowing our agent to act autonomously, as well as give us a measure of confidence that our agent acts appropriately in certain situations.

4 RESULTS AND DISCUSSION

The optimal policy (RL) is shown in Figure 3. The observations reflect the highlighted observed components to the states shown in Tables 2-4. Notably, due to the offline nature of our agent training this policy was developed and applied in a matter of minutes.



(a) The optimal policy learned by RL agent displayed as color-coded actions for each observation.



(b) A plot of Score$_\alpha$, Score$_\beta$, and overall effectiveness compared with a static playbook (baseline).

Figure 3: A depiction of the optimal policy and RL agent performance.

Figure 3a provides us with a generic output from our program that can be used to derive a dynamic playbook. For instance, detecting the use of the "cd" command to change the current working directory prompts the agent to recommend a procedure to change the user's password to halt the process. Although such an action may impact a normal user negatively, it is seen by the agent as an optimal solution that may prevent malicious exfiltration. Another interesting result from the optimal policy is that the agent learns to shut down our MySQL server when a simple "chkdsk" command is run. This command, which scans an attached drive for file system errors, is typically done by administrators when a device experiences a problem; therefore, a human may see this as a relatively benign event. Our blue agent, on the other hand, recognizes this as part of the red agent's behavior when done on the server itself and takes action to prevent data exfiltration. Much like the response to a "cd" command, this would affect all users for a period of time, which is a trade-off between security and availability.

We see such trade-offs over time with Figure 3b. This plot shows the agent's capability in balancing access for normal users, while preventing malicious activity. Since we implemented our scenario with four devices each from the COP and LF subnets, we see that overall effectiveness results in a simple average of Score$_\alpha$ and Score$_\beta$, thus providing a balanced measure of availability and protection from exfiltration.

Our RL agent largely performs above a baseline myopic policy in rounds 0 through 4. In particular, the RL$_\alpha$ is better than Baseline$_\alpha$ at preventing successful exfiltration activities. The efficacy degrades over time as the red agent adapts in each round to counter the blue agent dynamic playbook. This occurs because we did not conduct value iteration on the blue agent between each round to find new optimal policies. Determining appropriate periods to conduct retraining may be essential to effective policy adaptation.

It is believed that the POMDP problem is NP-hard (Papadimitriou and Tsitsiklis 1987). In general, the values associated with vast spaces of states and observations do not carry any certainty of global policy optimization. However, we restrict our search to a finite set of observations, generated from CLI logs. Event histories, are embedded directly within the space of observations by masking the device profile. This results in a set of screening criteria, generating an administrative policy to integrate with a stream of live monitoring events that we simulate through offline data gathering and training of an agent. RL plays the role of a numerical belief and policy integrator in augmenting the perspective of today's human operators. The codification of events into observations and states together with a sequential reward model may be used in future work to encode real-time constraints such as planned network outages or demand forecasts.

5 CONCLUSION

In this work, we examined the feasibility of training an autonomous cyber defense agent using offline RL to respond to malicious data exfiltration in a realistic network environment. Our scenario utilized a complex environment with both internal and external users to simulate a realistic operational network. By using CyberVAN to build our network, we were capable of simulating highly realistic malicious and benign behaviors from which our agent could train. Additionally, the use of nuanced Game Master software with native-to-device game scripts allows us to orchestrate actions as actual commands across the network.

As a result of our Game Master software, we were able to collect data to use for offline RL training. Our problem formulation allowed us to model the scenario as a POMDP, and we derived optimal policies through value iteration. As a result, we saw marked improvements by our agent in preventing data exfiltration while maintaining resource availability when compared to myopic policies. Over time, however, our work shows that the agent must continually retrain to account for changing malicious behaviors. Despite this, we believe our results lay the foundation for deriving dynamic DCO playbooks that can be incorporated into future SOAR technologies.

Although we greatly expand upon our previous work, several limitations still exist. Our network environment, despite being more realistic than before, still had relatively few devices operating on the network. We considered only value iteration as a solution method for solving our POMDP, which may not scale appropriately for fully-realized networks. Moreover, we did not consider many data exfiltration scenarios and implemented only a few response actions for our agent.

To address these shortcomings, future work will examine scaling the network environment to more devices, users, and accessible services. The simulation environment will ultimately be limited to the memory and CPU cores available to CyberVAN, but such scaling can be accomplished within the currently defined scenario environment and will lend itself to examining other attacker objectives and the feasibility of other solution algorithms. We also will consider other exfiltration vignettes that incorporate additional malicious behaviors focusing on social engineering. Additionally, we will consider other attacker objectives, such as network segmentation. This would aid us in implementing more advanced actions on the part of the red agent and blue agent by improving upon the current Game Master software. We will consider more analysis related to the specific methods and frameworks used. For example, we will consider parameter-tuning of learning methods such as value iteration as well as those related to types of risk within the POMDP reward function. That is, if availability of a network resource is a priority then the rewards should penalize loss of access more heavily.

Utilizing RL to aid in deriving DCO playbooks continues to show promise in detecting and preventing cyber threats. By developing optimal policies that are capable of adapting over time and implementing such policies into advanced SOAR platforms, CSOCs can close the gap with APTs and other cyber threats, helping to reduce the exponentially increasing costs of these threats and benefit all users of digital networks.

ACKNOWLEDGEMENTS

This research used resources from the Cyber Modeling and Simulation Research Lab at the Army Cyber Institute. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Military Academy, U.S. Army, U.S. Department of Defense, or U.S. Government.

REFERENCES

Alshamrani, A., S. Myneni, A. Chowdhary, and D. Huang. 2019. “A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities”. *IEEE Communications Surveys & Tutorials* 21(2):1851–1877.

Aref, M. A., S. K. Jayaweera, and S. Machuzak. 2017, March. “Multi-Agent Reinforcement Learning Based Cognitive Anti-Jamming”. In *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, 1–6. San Francisco, CA, USA.

Aslan, Ö. and A. A. Yilmaz. 2021. “A New Malware Classification Framework Based on Deep Learning Algorithms”. *IEEE Access* 9:87936–87951.

Bierbrauer, D. A., R. M. Schabinger, C. Carlin, J. Mullin, J. A. Pavlik and N. D. Bastian. 2023. “Autonomous Cyber Warfare Agents: Dynamic Reinforcement Learning for Defensive Cyber Operations”. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications V*, edited by L. Solomon and P. J. Schwartz, Volume 12538, 125380E. International Society for Optics and Photonics: SPIE.

Chadha, R., T. Bowen, C.-Y. J. Chiang, Y. M. Gottlieb, A. Poylisher, A. Sapello *et al*. 2016. “CyberVAN: A Cyber Security Virtual Assured Network Testbed”. In *MILCOM 2016-2016 IEEE Military Communications Conference*, 1125–1130. IEEE.

Hore, S., A. Shah, and N. D. Bastian. 2023. “Deep VULMAN: A Deep Reinforcement Learning-enabled Cyber Vulnerability Management Framework”. *Expert Systems with Applications* 221:119734.

Internet Crime Complaint Center 2024. “2022 Internet Crime Report”.

Johns Hopkins University Applied Physics Laboratory 2019. “Introduction to Integrated Adaptive Cyber Defense (IACD) Playbooks”. Accessed April 9, 2024.

Ozkan-Okay, M., E. Akin, Ö. Aslan, S. Kosunalp, T. Iliev, I. Stoyanov *et al*. 2024. “A Comprehensive Survey: Evaluating the Efficiency of Artificial Intelligence and Machine Learning Techniques on Cyber Security Solutions”. *IEEE Access* 12:12229–12256.

Papadimitriou, C. H. and J. N. Tsitsiklis. 1987. “The Complexity of Markov Decision Processes”. *Mathematics of Operations Research* 12(3):441–450.

Wang, W., D. Sun, F. Jiang, X. Chen and C. Zhu. 2022. “Research and Challenges of Reinforcement Learning in Cyber Defense Decision-Making for Intranet Security”. *Algorithms* 15(4):134.

Willeke, M. R., D. A. Bierbrauer, and N. D. Bastian. 2023. “Data-efficient, Federated Learning for Raw Network Traffic Detection”. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications V*, edited by L. Solomon and P. J. Schwartz, Volume 12538, 125380Y. International Society for Optics and Photonics: SPIE.

AUTHOR BIOGRAPHIES

ALEXANDER WEI was a Data Scientist / Engineer at the Army Cyber Institute at West Point. He holds a M.S. degree in Mathematics and a B.A. degree in Mathematics from Tufts University. His email address is alexander.wei@tufts.edu.

DAVID BIERBRAUER was a Research Scientist at the Army Cyber Institute at West Point. He holds a M.S.E. degree in Applied Mathematics and Statistics from Johns Hopkins University and a B.S. degree in Mathematical Sciences with Honors from the United States Military Academy. His email address is david.bierbrauer@westpoint.edu.

EMILY NACK is a Technical Product Manager at the Army Cyber Institute at West Point. She holds a B.S. degree in Game Design and Development with Honors from the Rochester Institute of Technology, along with an A.S. degree in Computer Information Systems from Hudson Valley Community College. Her email address is emily.nack@westpoint.edu.

JOHN PAVLIK was a Senior Research Scientist at the Army Cyber Institute at West Point. He holds a Ph.D. degree in Computer Science from the University of Illinois at Urbana-Champaign, a M.S. degree in Computer Science from the Air Force Institute of Technology, and a B.S. degree in Computer Science from Ohio University. His email address is john.pavlik@westpoint.edu.

NATHANIEL BASTIAN is Division Chief, Data & Decision Sciences and a Senior Research Scientist at the Army Cyber Institute at West Point. He holds a Ph.D. degree in Industrial Engineering and Operations Research from the Pennsylvania State University (PSU), a M.Eng. degree in Industrial Engineering from PSU, a M.S. degree in Econometrics and Operations Research from Maastricht University, and a B.S. degree in Engineering Management (Electrical Engineering) with Honors from the United States Military Academy. His email address is nathaniel.bastian@westpoint.edu.