Discovering Exfiltration Paths Using Reinforcement Learning with Attack Graphs

Tyler Cody[a*], Abdul Rahman[b], Christopher Redino[b],
Lanxiao Huang[a], Ryan Clark[b], Akshay Kakkar[b],
Deepak Kushwaha[b], Paul Park[c], Peter Beling[a], Edward Bowen[b]
[a]National Security Institute, Virginia Tech
[b]Deloitte & Touche LLP
[c]Deloitte Consulting LLP
[*]Corresponding Author: tcody@vt.edu

***Abstract*—Reinforcement learning (RL), in conjunction with attack graphs and cyber terrain, are used to develop reward and state associated with determination of optimal paths for exfiltration of data in enterprise networks. This work builds on previous crown jewels (CJ) identification that focused on the target goal of computing optimal paths that adversaries may traverse toward compromising CJs or hosts within their proximity. This work inverts the previous CJ approach based on the assumption that data has been stolen and now must be quietly exfiltrated from the network. RL is utilized to support the development of a reward function based on the identification of those paths where adversaries desire reduced detection. Results demonstrate promising performance for a sizable network environment.**

***Index Terms*—attack graphs, reinforcement learning, exfiltration paths, penetration testing, cyber terrain**

I. Introduction

The National Institute of Standards and Technology (NIST) special publication 800-53 revision 5 states that exfiltration[1] (also called exfil) is the unauthorized movement of data within a network [1]. Many times, cyber attacks are considered successful if they exfiltrate data for monetary, disruptive, or competitive gain. Detection of exfiltration can be plagued with technical challenges as adversaries routinely encapsulate data within typically allowable protocols (e.g., http(s), DNS) which make it significantly harder to defend. Additionally, adversaries have been known to prefer traversing certain network paths for data theft to reduce detection and tripping cyber defenses so they do not raise suspicions.

Heisting data requires two different plans: a plan to get to the data and a plan to exfiltrate the data without getting caught. Much effort in the cybersecurity industry is devoted to identifying and preventing points of weakness that allow authorized (i.e., adversarial) entry into a network. The most common exfiltration opportunity is moving data from a local network to an adversary network via the internet. To perform this, an adversary must gain access to the data on an organization's network, then send the data to a place off their network. Most organizations are focused on preventing network access, which leaves gaps in defenses for access from the network to the internet.

Much of the literature on automating penetration testing using RL has a focus on the way networks can be accessed (i.e., infiltration [2]–[10]). And while some consider using RL to detect exfiltration [11], [12], RL for conducting post-exploitation activities like exfiltration are under-studied [13]. Maeda and Mimura apply deep RL to do exfiltration, however, they do not use a standard attack graph construct, but rather define states using an ontological model of the agent and define actions using task automation tools. [13]. Their approach has several limitations:

- The RL agent's inputs and outputs are greatly abstracted away from network structure, path structure, and cyber terrain, thereby limiting the ability to anchor agents to the *real* computer network.
- The exfiltration methodology does not leverage automated frameworks for attack graph construction like MulVal [14] or the vulnerability- and bug-reporting communities (e.g., via the Common Vulnerability Scoring System (CVSS) [15]).
- The output of the RL-based exfiltration method is not easily interpretable in terms of networks, their paths and configurations, and risks preferences regarding their traversal.

Whereas Maeda and Mimura's use of task automation tools and ontologies make their proposed exfiltration method a highly automated means of actually performing exfiltration, this paper presents an alternative approach more tailored to automating exfiltration path discovery for cyber operator workflows.

This paper presents an RL method for discovering exfiltration paths in attack graphs. This paper proposes and combines:

1) An approach for modeling service-based defensive cyber terrain in dynamic models of attack graphs.
2) An RL-based algorithm for discovering the top-N exfiltration paths in an attack graph.

The presented methodology is aligned with a focus on network structure and configuration, path analysis, and cyber terrain.

[1]NIST 800-53r5 [1] states specifically that exfiltration lies within security control SC-07(10) for boundary protection to prevent unauthorized data movement (exfiltration).

978-1-6654-2141-6/22/$31.00

It maintains MulVal's focus on scalability and leverages the vulnerability- and bug-reporting communities via CVSS. Its outcomes can be directly understood as paths through networks, as is highlighted in a detailed discussion of the results. To support reproducibility, the RL solution methods, experimental design, and network model are specified in great detail.

This paper is structured as follows: First, background on RL for penetration testing and on constructing Markov decision processes (MDPs)from attack graphs is given. After, the methods for modeling defensive terrain and discovering exfiltration paths are presented. Then, experimental design is described, results are presented, and findings are discussed. The paper concludes with remarks on modeling decisions, a synopsis, and a statement on future work.

II. RL and Penetration Testing

A. Reinforcement Learning Preliminaries

RL describes the paradigm of learning by interaction with an environment [16]. This contrasts directly with supervised learning where an oracle is queried for ground-truth labels. More formally, it describes a set of solution methods for approximate dynamic programming [17]. It also addresses challenges associated with large and complex environments by approximating various aspects of planning and decision-making.

With respect to RL, agents learn by taking actions in environments $\mathcal{E}$ and receiving rewards. Commonly, environments $\mathcal{E}$ are modeled as MDPs. Finite MDPs are tuples $\{S, A, \Phi, P, R\}$ where S and A are states and actions, $\Phi \subset S \times A$ are admissible state-action pairs, $P : \Phi \times S \rightarrow [0, 1]$ is the probability transition function, and $R : \Phi \rightarrow \mathbb{R}$ where $\mathbb{R}$ are the reals is the expected reward function. An agent interacts with an environment $\mathcal{E} = \{S, A, \Phi, P, R\}$ by taking actions a_t and receiving states s_{t+1} and rewards r_{t+1}.

The learning procedure can be described in general terms as follows. Let R_t be the discounted sum of future rewards,

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{1}$$

where $\gamma \in (0, 1)$ is a discount factor. The action value function $Q^\pi(s, a)$ can then be defined as

$$Q^\pi(s, a) = \mathbb{E}[R_t | s_t = s, a], \tag{2}$$

where π is a policy mapping states and actions (s, a) to the probability of picking action a in state s. The learning procedure aims to find the optimal action value function $Q^*(s, a)$,

$$Q^*(s, a) = \max_\pi Q^\pi(s, a). \tag{3}$$

Deep Q-learning (DQN) approximates Q^* with a neural network $Q(s, a; \theta)$, where θ are parameters of the neural network [18], [19].

Alternatively, instead of learning the Q function, policies can be parameterized and learned directly. In policy gradient methods, the reward function is defined as

$$J(\theta) = \sum_s d^\pi(s) V^\pi(s) = \sum_s d^\pi(s) \sum_a \pi_\theta(a|s) Q^\pi(s, a), \tag{4}$$

where $d^\pi(s)$ denotes the stationary distribution of Markov chain for π_θ. According to policy gradient theorem, the gradient $\nabla_\theta J(\theta)$ is given by

$$\nabla_\theta J(\theta) \propto \sum_s d^\pi(s) \sum_a Q^\pi(s, a) \nabla_\theta \pi_\theta(a|s). \tag{5}$$

The policy gradient theorem provides a basis for learning a parameterized policy. However, it suffers from high variance of gradient and instability. To overcome this, the value of the state $V^\pi(s)$, the value of using policy π in state s is introduced as the baseline:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \tag{6}$$

where $A^\pi(s, a)$ is called the advantage. And the gradient is now given as

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(s, a) A_\pi(s, a)] \tag{7}$$

These gradients serve as the basis for the advantage actor-critic method (A2C), a standard policy gradient method in deep reinforcement learning [20].

B. RL for Penetration Testing

While deep RL has been applied to cybersecurity broadly [21], it has only recently been pursued as a tool for penetration testing [2]–[10]. While approaches and uses vary greatly, many use *attack graphs* to model the network [22]. Note, attack graphs model the network formed by computer vulnerabilities and exploits, creating an abstraction that does not necessarily match the topology of the physical network, as shown in Figure 1. The use of attack graphs is a distinguishing character of RL for penetration testing from RL for cybersecurity broadly [21].

Most frequently, RL is tasked with simply traversing a network from one initial node to one terminal node—(i.e., finding paths through networks) [2]–[9]. Gangupantulu et al., in contrast, emphasize a more complex task of using RL to perform CJ analysis [10]. Here, similar to Gangupantulu et al., the presented RL method solves a more complex task and serves as a targeted tool for cyber operators to improve the efficiency of operator workflow in penetration testing. It does not automate exfiltration entirely.

RL for penetration testing has made frequent use of DQN [3], [7]–[10]. Nguyen et al., alternatively, propose an RL-based approach to penetration testing that uses two agents: one for iteratively scanning the network to build a structural model and another for exploiting the constructed model [23]. In our first attempt at performing RL on the network presented later on, we attempt to use the DQN solution method but it did not converge, leading us to explore alternative agents. We use Nguyen et al.'s double agent method where both agents are A2C. We compare to the standard A2C algorithm.

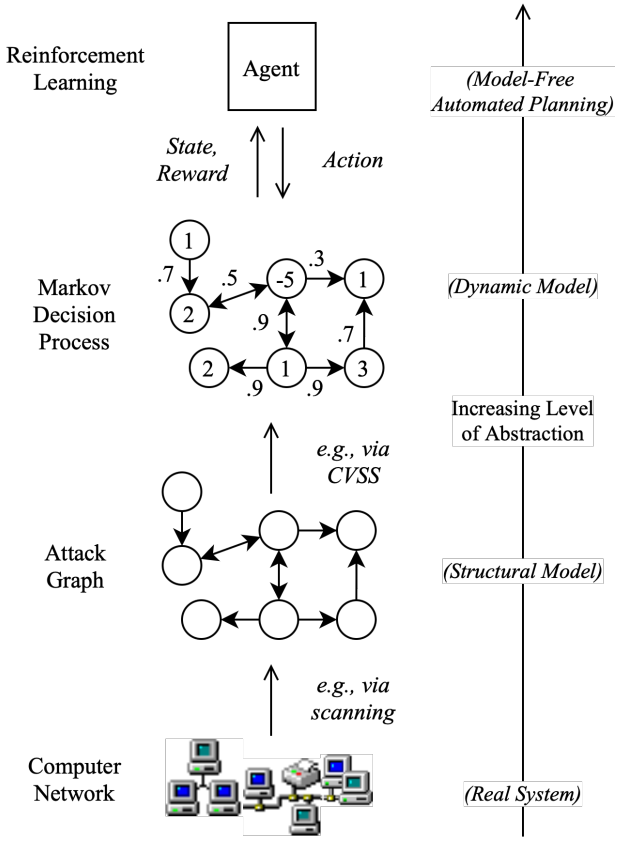

Fig. 1: RL for penetration testing requires abstracting from real computer networks described by information such as packet flows, into the mathematical models with which RL agents interact.

III. MDPs from Attack Graphs

There are many solution methods for modeling attack graphs [24]. Key trade-offs relate to scalability, observability, accuracy, and reliability. In particular, partially observable Markov decision processes (POMDPs) are well-argued to be a more realistic representation of computer networks than MDPs [25]. In POMDPs, actions are stochastic and network structure and configuration are uncertain. But POMDPs have not been shown to scale to large networks and require modeling many prior probability distributions [26]. Additionally, while RL for MDPs is well-established, RL for POMDPs is still under fundamental development [27]. Currently, MDPs are the standard in RL for penetration testing [2]–[10].

The CVSS [15], [28] is used as a scalable approach for adding behavior to attack graphs [29], [30]. It is the numerical representation of an information security vulnerability. These scores represent an attempt at providing a standardized way of evaluating the severity of threats posed by a particular vulnerability. This takes into consideration both how easy it is to exploit this vulnerability, and also how severe the consequences of such an exploit would be.

While some authors in RL for penetration testing use alternative methods [2]–[5], CVSS is emerging as a standard approach to modeling MDPs over attack graphs for RL [6]–[10]. Gangupantulu et al. draw from the literature to define a *vanilla* CVSS-MDP for point-to-point network traversal [9].

CVSS-MDPs use the attack graph to define the state-action space $S \times A$ and CVSS to define the reward R and transition probabilities P [9]. CVSS-MDP assigns transition probabilities P using the attack complexity, where the ranks low, medium, and high are associated with transition probabilities of 0.9, 0.6, and 0.3. The reward for arriving at a host is given by,

$$\textit{Base Score} + \frac{\textit{Exploitability Score}}{10}.$$

The agent receives -1 reward for each step and receives 100 reward for arriving at the terminal node. Episodes terminate when the terminal state is reached after number of steps (i.e., actions).

While CVSS scores are useful in practice and currently considered an industry standard, it is important to remember that a measure of threat severity is not the same as a measure of risk and that they do not generalize to give information that's useful for evaluating an entire attack path through a network. From the perspective of an attacker, a greater risk means a greater chance of detection. While the CVSS scores of vulnerabilities do inform the probability of success of any particular exploit in the models here, the real driving force of RL agent behavior should be centered around concepts of terrain [31]. The details of this reward engineering of terrain are given in the following.

IV. Methods

The following subsections describe the presented methods for adding service-based risk penalties as defensive terrain in CVSS-MDPs and the algorithm for discovering the top-N exfiltration paths in a network, shown in Algorithm 1.

A. Defensive Terrain in CVSS-MDPs

Gangupantulu *et al.* argue that models of cyber terrain can be layered onto CVSS-MDPs, and do so by adding firewalls between subnets, and assigning protocol-specific negative reward and transition probabilities for traversing firewalls [9]. Gangupantulu *et al.* later layer on additional notions of cyber terrain by using RL as part of a methodology for modeling footholds and pivot points nearby the 2-hop network of CJ nodes [10].

We propose a new approach for modeling service-based defensive terrain in CVSS-MDPs. Instead of explicitly defining the defenses in the states of the MDP, we make assumptions similar to what a human attacker would make: even if the attacker cannot detect a defense directly, by their experience they can infer the presence of defenses based on the services available on a given host. Common network defenses can include host-based antivirus and malware detection software, inter-subnet router firewalls, or authentication log tracking.

We engineer rewards for defensive terrain that are additive, or, otherwise put, are layered on top of the CVSS-MDP

Algorithm 1 Exfiltration Paths via RL (EP-RL)

Require: MDP, initial node i, exit nodes J, N
Ensure: N paths from initial node to top-N exit nodes
  for i in N **do**
    $path \leftarrow f_{RL}(MDP, i, J)$ ▷ Optimal path $i \rightarrow j, j \in J$
    $paths \leftarrow store(path)$
    $J \leftarrow J \setminus j$ ▷ Remove j from J
  end for
  return $paths$

rewards. A quantified negative reward structure is used to itemize the cost of attacker actions. The criteria of interest are (1) *a risk hierarchy applied to service categories* and (2) *the type of action performed by the agent on a host*. The requirement to implementing these criteria is to unify them in a way the agent could enumerate. This is achieved by creating an array of actions and services and applying an individual reward to each combination. The negative reward can be assigned using (1) action type, here, exploiting or scanning, and (2) services.

Services are derived from four principal categories: authentication, data, security, and common. To create a negative reward, a hierarchy of costs associated with attacking these services was applied. When performing an exploiting action, this hierarchy applies authentication as a reward of -6, data as a reward of -4, while both security and common have a reward of -2. When performing a scanning action, the reward is increased by 1 (i.e., -5, -3, -1, respectively). These rewards represent a combination of factors highlighting the risk to organizations presented from these services. Different organizations or operators may prefer a different scaling. It is important to note that the values of these negative rewards are relative, and as such they can be as a set scaled together to represent different risk preferences. When taking an action on a host with multiple services, the agent applied the highest cost to the action's reward. This was a decision that presumes a leading practice approach by security practitioners to apply security controls based upon the 'riskiest' service found on a host (i.e., a service known to be at greater exposure to the network edge or greater business loss if exploited). By syncing our rewards to this presumption, the agent calculates a more realistic quantitative measurement of risk as it attempts to converge to an optimal attack path.

B. Discovering Exfiltration Paths with RL

In contrast to Gangupantulu et. al.'s CJ analysis method [10], our discovering exfiltration paths method uses multiple terminal states corresponding to the various exit nodes of interest and only a single initial node. The agent then interacts with the network in an episodic fashion to learn which is the best exit node with respect to expected reward. To provide a comprehensive path analysis for cyber operators using the tool, the top-N exit nodes are found by iteratively solving the MDP to find the best exit node, removing the best exit node, and solving the MDP again. This algorithm is described in Algorithm 1. Notably, the agent iteratively solves the problem of finding a path to a single exit in the joint set of exits. This avoids the brute force approach of creating an MDP for each exit node, solving each MDP, then ranking the paths.

V. Experimental Design

In the following subsections the network, state-action space, and RL algorithm implementation are described.

A. Network Description

The experimental network where the simulations are run was derived from an architectural leading practices approach to represent enterprise network configurations and deployments. The network contains:

- Defined Subnets - 9
- Defined Hosts - 26
- Types of Operating Systems - 2
- Privilege Access Levels - 3
- Network Services - 9
- Host Processes - 6
- Network Firewall Rulesets - 39

The network is visualized in Figures 3, 4, and 5.

Subnets are constructed to represent a grouping of hosts with commonly segregated services utilized for enterprise information technology administration to include server services, database services, client workstation networks, edge and DMZ services, and core services that orchestrate least-privilege or zero-trust security (i.e., domain controllers and public-key infrastructure) [32]. Hosts and network firewall rulesets are configured to deliver a representation of common ITS communication requirements between these subnets that allow daily functions of an enterprise ITS department and business operations.

The services within this network are laid out with the presumption of common security controls and monitoring software one would see within an enterprise network. These presumptions include the following expectations:

1) Authentication services are exposed to the internet through a Virtual Private Network (VPN).
2) Web services are exposed to the internet through a secured edge network zone (DMZ).
3) Services exposed to the internet are monitored.
4) Firewalls are monitored at a higher rate than other network devices.
5) Security services have the most inherited security controls.
6) Authentication services and firewall services, if exploited, have the greatest secondary and tertiary impacts to a network's overall security profile.
7) Network security rules only apply allowlists.
8) Host and network assets apply principles of least-privilege when authorizing privileges for account access and use.

B. Environment Description

For the environment, each host is represented by an 1D vector that contains its status (compromised, reachable, discovered or not) and configurations (address, services, operating system and processes). The environment combines all the vectors for hosts in the network as a entire state tensor. Thus, each state contains descriptions of all hosts. The actions are defined as an operation performed on a specific target host. The actions consist of 6 primitive actions for scanning, exploiting, or privilege escalation. The action type and target host configuration must align or the action will fail. For the environment, the initial host for exfiltration is set on (6, 0), while the terminal hosts are set on (1, 0), (2, 0) and (4, 0), which are all connected to the public internet, where (a, b) denotes host b in subnet a. The initial node is set as compromised, reachable and also discovered at the beginning to make it possible for the agent to perform further actions. The exfiltration goal is reached if the agent compromises any host among them and obtains root access. If the goal is reached, the agent is given a high reward (set as 100 for our experiment).

C. RL Implementation

The experiment is conducted based on two models: A2C model and the double agent architecture [23]. Both agents in double agent use the A2C algorithm. For both, the learning rate is set as 0.001 and the discounted factor is set as 0.99. We use Adam as the optimizer of our networks. Both of the models are trained for 4,000 episodes with a maximum of 3,000 steps in each episode. If the maximum number of steps is reached, the episode terminates and the agent receives 0 terminal reward. Both the A2C model and the structuring agent of double agent use deep neural networks (DNNs) with three fully connected layers of size 64, 32, and 1 and the exploiting agent of the double agent uses a DNN with two fully connected layers of size 10 and 1. All DNNs use tanh activation functions for non-output layers and softmax for the output layer.

D. Sensitivity Analysis

The experiments run the A2C and double agent algorithms to convergence. To study the effect of the scale of service-based penalties on the convergence of the agents and on the paths they discover, the exploiting and scanning service-based penalties are scaled by a factor of 1.3, 1.0, and 0.7. These values correspond to risk preferences that we term risk-averse, risk-neural, and risk-accepting, respectively.

VI. Results

To observe the convergence of our models, we plot the steps and the reward versus episodes and the result is shown in Fig. 2. It can be observed that both of our models converge within 1,000 episodes. It could also be noticed that double agent converges slower than the A2C agent, which is expected considering that the double agent is more complex and contains two A2C models that learn simultaneously.

A. RL Performance

When reviewing the A2C and double agent as they reach convergence, A2C uses a similar amount of episodes to reach an optimal path regardless of the scaling factor. In the double agent model, the risk-accepting agent reaches an optimal path much quicker than the risk-neutral and risk-adverse agents. Additionally, the double agent model converges quickly at first, and then plateaus. This suggests the double agent model can quickly arrive at near optimal policies.

B. Agent's Behavior Compared to Human Expectations

The paths results from the nine experiments are shown in Table I. The simulations represent the top three paths for all three risk preferences. In addition to the paths, the table shows the number of steps, the reward and the cumulative probability score. These cumulative probability score are not directly the CVSS scores of the exploits, but are proprietary scores designed to play similar role. It is important to note that the ordering of these scores is *not* the same as the ordering of the reward. This is in agreement with the expectation that the measure of risk (tracked by the reward) does not have to track linearly with a vulnerability score.

Unbeknownst to the rest of the authors, the cyber security operations expert who crafted the simulated networks for the generated attack paths and network topology included two intentional misconfigurations within the host-service assignments. These misconfigurations simulated real-world experiences resolving enterprise network incidents where exfiltration of data occurred. The primary goal of these misconfigurations is to represent flaws in network design that were exploited by actual attackers for exfiltration in actual enterprise networks. If the agents can deduce (without explicit design) this misconfiguration, it would be a compelling example of how this reward engineering can produce human like behavior.

The significant configuration within our results was on host (3,2). The PKI service was extended into subnet (3), the server subnet. In published leading practices for securing PKI [32], this service is included in the most privileged tier of an environment and requires a specific privileged account authorization 'Crytpographic Operator' [32]. As such it should only be accessible utilizing hardware and software with enhanced security controls. These leading practices also require this service to only reside in a secure subnet alongside other servers and appliances with similarly privileged security requirements. When this service is allowed to operate as a node within the general server subnet (i.e., regular business applications), it exposes the network firewall rules to exploitation when exfiltrating data from the private key repository database.

In the resulting optimal path diagrams identified by the agent, host (3,2) was the most traversed node. Reviewing this result shows that the misconfiguration was successfully identified and exploited by the agent when defining an optimal path.

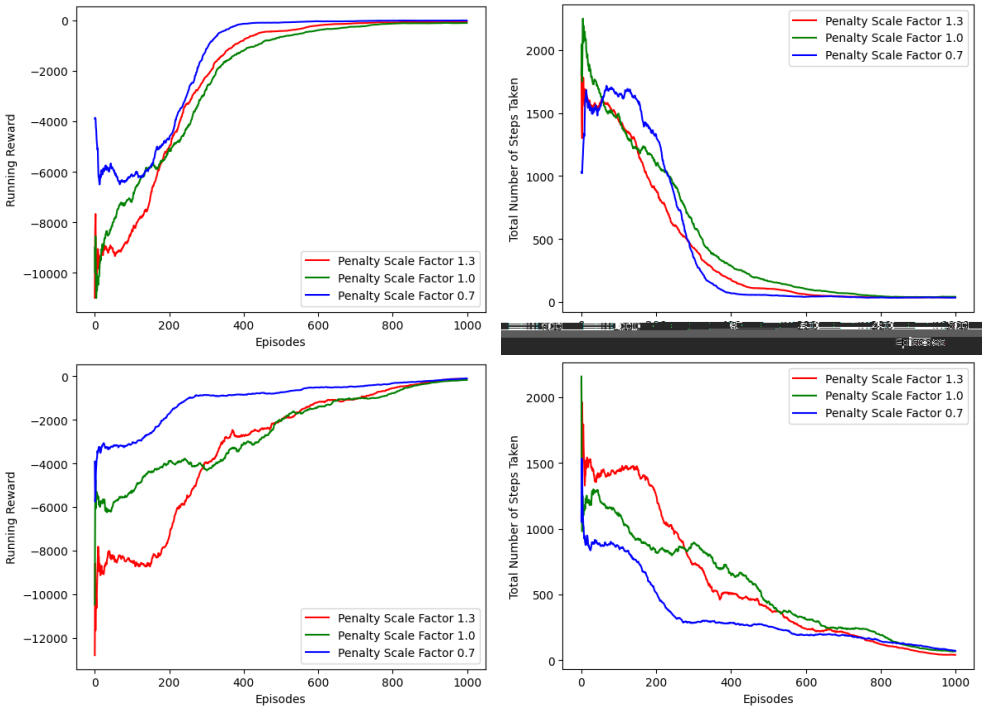

Fig. 2: Agent learning over episodes. The left plots show the average reward over episodes and the right plots show the average number of steps taken over episodes. The top plots are the results from running RL using an A2C agent and the bottom plots show the results when leveraging the double agent methodology [23]. The color of the lines reflects how heavily incentivized an agent is to avoid detection: blue for risk-accepting, green for risk-neutral, and red for risk-averse.

Path Rank	Scale Factor	Path	Steps	Reward	Cumulative Probability Score
Best Path	0.7	$(6,0) \rightarrow (3,0) \rightarrow (2,0)$	12	57.8	$2.9 + 2.9 = 5.8$
	1.0	$(6,0) \rightarrow (3,2) \rightarrow (1,0)$	11	62	$2.9 + 2.9 = 5.8$
	1.3	$(6,0) \rightarrow (3,0) \rightarrow (1,0)$	5	68.3	$1.9 + 2.9 = 4.8$
Second Best Path	0.7	$(6,0) \rightarrow (3,2) \rightarrow (1,0)$	19	46.9	$2.9 + 4.9 = 7.8$
	1.0	$(6,0) \rightarrow (3,0) \rightarrow (2,0)$	16	24	$2.9 + 4.8 = 7.7$
	1.3	$(6,0) \rightarrow (3,2) \rightarrow (1,0)$	19	33.1	$1.9 + 2.9 = 4.8$
Third Best Path	0.7	$(6,0) \rightarrow (3,0) \rightarrow (1,1) \rightarrow (4,0)$	15	41.3	$1.9 + 1.9 + 2.4 = 6.2$
	1.0	$(6,0) \rightarrow (3,2) \rightarrow (1,0) \rightarrow (4,0)$	24	17	$3.9 + 1.9 + 7.5 = 13.3$
	1.3	$(6,0) \rightarrow (3,2) \rightarrow (1,0) \rightarrow (4,0)$	22	-6.1	$1.9 + 2.9 + 6.3 = 11.1$

TABLE I: Table of the top-3 exfiltration paths found by double agent. *Scale factor* denotes the risk-accepting (0.7), risk-neutral (1.0), and risk-averse (1.3) scaling of the penalty for services. *Path* gives the shortest path from the initial node to exit node, and is derived from the set of actions taken by the converged agent in an episode. *Steps* and *reward*, in contrast, refer to the optimal performance of the agent in an episode (i.e., not just the actions taken to form the *path*). *Cumulative probability score* reports a custom, CVSS-like vulnerability scoring of the *path*.

VII. Discussion

A. Benefits of Approach

The presented methods can provide security defenders and operators three immediate benefits:

1) Iterate security control implementations within enterprise networks by *prioritizing the most impactful controls first*.
2) *Quantify decreased risk factor* for each iteration of new security controls via the reward.
3) Deliver integrity to the results by *matching the attacker actions taken to expected actions for each risk preference*.

This RL approach associates to the integrity component of the cybersecurity CIA triad (confidentiality, integrity, and availability). Upon completion of the modeling simulations, the results were analyzed by a cybersecurity operations expert with certification in security architecture and experience resolving incident response from nation-state and APT attack groups. This review found that experiment results matched the expected results for the simulated networks. Relevant criteria for this decision include:

- Risk-adverse agent takes very few steps when the entire network is exposed.
- Risk-accepting agent will achieve a greater reward in more secure networks because of its ability to move faster

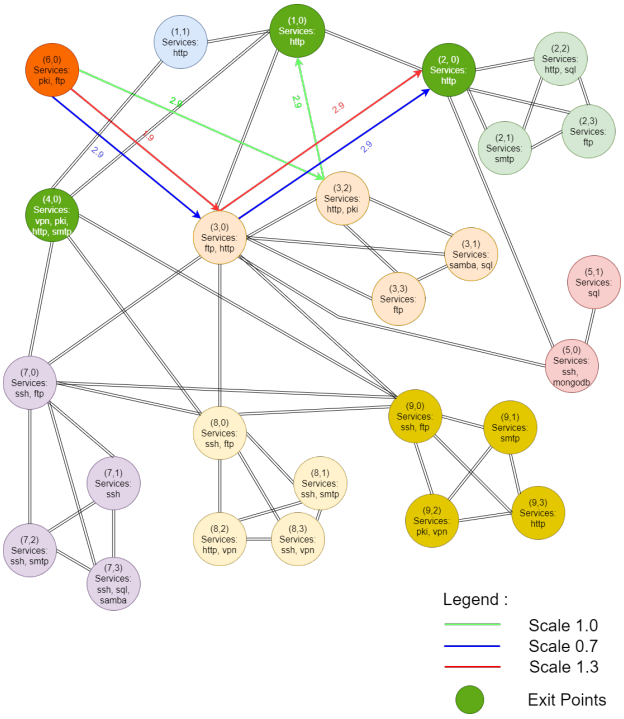

Fig. 3: Network diagram showing *Best Path* in Table I. The color of the edge reflects risk preference and the color of nodes encodes the subnet.

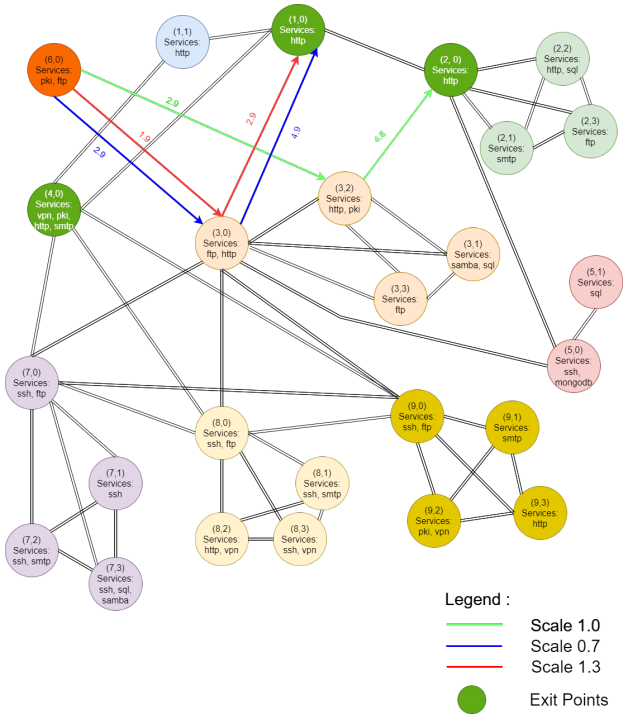

Fig. 4: Network diagram showing *Second Best Path* in Table I. The color of the edge reflects risk preference and the color of nodes encodes the subnet.

than stealthier actors.

- The optimal path will often be the same for various risk profiles, matching the A2C modeling convergence trends.
- Utilizing misconfigurations of security services within a network is a high-likelihood of success for attackers.
- Data exploitation is more likely through servers and services than through client workstations.
- When operating in more secure networks, the agent consistently creates simple exfiltration paths but requires additional unsuccessful scanning actions to achieve this same convergence.

B. Remarks on Payload

Within the current scope of this work, there is no consideration for the size of the payload extracted, or the rate at which it is removed. If the payload is a small amount of (critical) data, this simulation can be considered an approximation of reality. If the amount of data exfiltrated becomes large enough that this approximation fails, then additional modeling considerations need to be considered, such as encoding rates of transfer and amount of data into the states of the environment. Payload size, while being a calculable statistic for security operations, is often measured for security in a binary manner. If the payload size from one server to another server, or for one firewall at a given time of day, is of sufficient variation from the expected thresholds, an alert will trigger for security or network operations. While malicious actions can create this, non-malicious actions can create this as well. Common ITS operations such as database backups, system update downloads, or unexpected network configuration changes can each create a pattern that alerts security or network teams to heavy payloads on a network. Without a way to compensate for these additional variables, the value of adding payload sizes in this work was negligible.

VIII. Conclusion

In this paper, we have provided security practitioners and network defenders a quantitative methodology using RL to identify optimal paths for data exfiltration. In our experiments, the presented RL approach identified the most likely hosts and services used when exfiltrating data and captured metrics used in network risk assessments. The strength of this approach was validated through identification of intentional network misconfigurations that mimic real-world vulnerabilities.

Future work should consider integration with other RL for penetration testing tasks. In addition, expanding the risk formalism to increase its sophistication and maturity will drive increased applicability and relevance. Review of payload size extraction and subsequent rates are also should also be included for future studies.

IX. Acknowledgements

This work was made possible through the collaboration between Dr. Michael Ambroso, Lead for the AI/ML for Cyber Testing Innovation Pipeline group—funded by Deloitte's Cyber Strategic Growth Offering led by Deborah Golden,

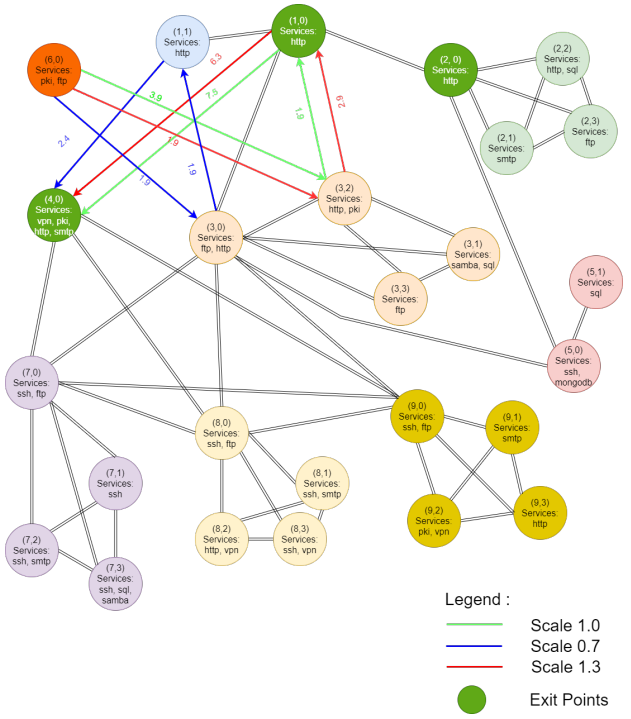

Fig. 5: Network diagram showing *Third Best Path* in Table I. The color of the edge reflects risk preference and the color of nodes encodes the subnet.

and Dr. Laura Freeman, Director of the Hume Center for National Security and Technology's Intelligent Systems Lab at the Virginia Polytechnic Institute and State University.

REFERENCES

[1] N. I. of Standards and Technology, "Security and privacy controls for federal information systems and organizations," U.S. Department of Commerce, Washington, D.C., Tech. Rep. NIST Special Publication 800-53 Revision 5, 2020.

[2] M. C. Ghanem and T. M. Chen, "Reinforcement learning for intelligent penetration testing," in *2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*. IEEE, 2018, pp. 185–192.

[3] J. Schwartz and H. Kurniawati, "Autonomous penetration testing using reinforcement learning," *arXiv preprint arXiv:1905.05965*, 2019.

[4] M. C. Ghanem and T. M. Chen, "Reinforcement learning for efficient network penetration testing," *Information*, vol. 11, no. 1, p. 6, 2020.

[5] S. Chaudhary, A. O'Brien, and S. Xu, "Automated post-breach penetration testing through reinforcement learning," in *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2020, pp. 1–2.

[6] M. Yousefi, N. Mtetwa, Y. Zhang, and H. Tianfield, "A reinforcement learning approach for attack graph analysis," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 212–217.

[7] A. Chowdhary, D. Huang, J. S. Mahendran, D. Romo, Y. Deng, and A. Sabur, "Autonomous security analysis and penetration testing," in *2020 16th International Conference on Mobility, Sensing and Networking (MSN)*. IEEE, 2020, pp. 508–515.

[8] Z. Hu, R. Beuran, and Y. Tan, "Automated penetration testing using deep reinforcement learning," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2020, pp. 2–10.

[9] R. Gangupantulu, T. Cody, P. Park, A. Rahman, L. Eisenbeiser, D. Radke, and R. Clark, "Using cyber terrain in reinforcement learning for penetration testing," *arXiv preprint arXiv:2108.07124*, 2021.

[10] R. Gangupantulu, T. Cody, A. Rahman, C. Redino, R. Clark, and P. Park, "Crown jewels analysis using reinforcement learning with attack graphs," *arXiv preprint arXiv:2108.09358*, 2021.

[11] S. Venkatesan, M. Albanese, A. Shah, R. Ganesan, and S. Jajodia, "Detecting stealthy botnets in a resource-constrained environment using reinforcement learning," in *Proceedings of the 2017 Workshop on Moving Target Defense*, 2017, pp. 75–85.

[12] M. Albanese, S. Jajodia, and S. Venkatesan, "Defending from stealthy botnets using moving target defenses," *IEEE Security & Privacy*, vol. 16, no. 1, pp. 92–97, 2018.

[13] R. Maeda and M. Mimura, "Automating post-exploitation with deep reinforcement learning," *Computers & Security*, vol. 100, p. 102108, 2021.

[14] X. Ou, S. Govindavajhala, A. W. Appel *et al.*, "Mulval: A logic-based network security analyzer." in *USENIX security symposium*, vol. 8. Baltimore, MD, 2005, pp. 113–128.

[15] P. Mell, K. Scarfone, S. Romanosky *et al.*, "A complete guide to the common vulnerability scoring system version 2.0," in *Published by FIRST-forum of incident response and security teams*, vol. 1, 2007, p. 23.

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[17] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[21] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *arXiv preprint arXiv:1906.05799*, 2019.

[22] J. P. McDermott, "Attack net penetration testing," in *Proceedings of the 2000 workshop on New security paradigms*, 2001, pp. 15–21.

[23] H. V. Nguyen, S. Teerakanok, A. Inomata, and T. Uehara, "The proposal of double agent architecture using actor-critic algorithm for penetration testing." in *ICISSP*, 2021, pp. 440–449.

[24] T. Gonda, T. Pascal, R. Puzis, G. Shani, and B. Shapira, "Analysis of attack graph representations for ranking vulnerability fixes." in *GCAI*, 2018, pp. 215–228.

[25] E. Miehling, M. Rasouli, and D. Teneketzis, "A pomdp approach to the dynamic defense of large-scale cyber networks," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2490–2505, 2018.

[26] D. Shmaryahu, G. Shani, J. Hoffmann, and M. Steinmetz, "Constructing plan trees for simulated penetration testing," in *The 26th international conference on automated planning and scheduling*, vol. 121, 2016.

[27] P. Zhu, X. Li, P. Poupart, and G. Miao, "On improving deep reinforcement learning for pomdps," *arXiv preprint arXiv:1704.07978*, 2017.

[28] H. Joh and Y. K. Malaiya, "Defining and assessing quantitative security risk measures using vulnerability lifecycle and cvss metrics," in *Proceedings of the 2011 International Conference on Security and Management (SAM'11)*, vol. 1, 2011, pp. 10–16.

[29] L. Gallon and J. J. Bascou, "Using cvss in attack graphs," in *2011 Sixth International Conference on Availability, Reliability and Security*. IEEE, 2011, pp. 59–66.

[30] M. Keramati, A. Akbari, and M. Keramati, "Cvss-based security metrics for quantitative analysis of attack graphs," in *ICCKE 2013*. IEEE, 2013, pp. 178–183.

[31] G. Conti and D. Raymond, *On cyber: towards an operational art for cyber conflict*. Kopidion Press, 2018.

[32] M. Corporation. (2021) Implementing least-privilege administrative models. [Online]. Available: https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/implementing-least-privilege-administrative-models