Enhancing Exfiltration Path Analysis Using Reinforcement Learning

Riddam Rishu[1], Akshay Kakkar[1], Cheng Wang[1,*], Abdul Rahman[1], Christopher Redino[1], Dhruv Nandakumar[1], Tyler Cody[2], Ryan Clark[1], Daniel Radke[1] and Edward Bowen[1]

[1]*Deloitte & Touche LLP*
[2]*National Security Institute, Virginia Tech*

Abstract

Building on previous work using reinforcement learning (RL) focused on identification of exfiltration paths, this work expands the methodology to include protocol and payload considerations. The former approach to exfiltration path discovery, where reward and state are associated specifically with the determination of optimal paths, are presented with these additional realistic characteristics to account for nuances in adversarial behavior. The paths generated are enhanced by including communication payload and protocol into the Markov decision process (MDP) in order to more realistically emulate attributes of network based exfiltration events. The proposed method will help emulate complex adversarial considerations such as the size of a payload being exported over time or the protocol on which it occurs, as is the case where threat actors steal data over long periods of time using system native ports or protocols to avoid detection. As such, practitioners will be able to improve identification of expected adversary behavior under various payload and protocol assumptions more comprehensively.

Keywords

reinforcement learning, exfiltration, penetration testing, cyber terrain

1. Introduction

In previous work [1], RL was employed to expose exfiltration[1] (also called exfil) paths within networks through carefully engineering a reward system to account for the nodes with a network topology considered as cyber terrain [3]. Additionally, [1] and [4] did not consider the payload size or protocol preference during the exfiltration operation: these proposed methodologies are only realistic for nominal payload sizes. Additional modelling restrictions, such as how much data is sent and at what rate the data is being moved, must be considered for large volume exfiltration operations [5, 6, 7]. These constraints reflect realistic cyber security paradigms that model adversarial activity where exfiltration operations often prefer a protocol (e.g. tunneling

CAMLIS'23: Conference on Applied Machine Learning for Information Security, October 19–20, 2023, Arlington, VA
*Corresponding author.
rrishu@deloitte.com (R. Rishu); akshkakkar@deloitte.com (A. Kakkar); chengwang@deloitte.com (C. Wang); abdulrahman@deloitte.com (A. Rahman); credino@deloitte.com (C. Redino); dnandakumar@deloitte.com (D. Nandakumar); tcody@vt.edu (T. Cody); ryanclark4@deloitte.com (R. Clark); dradke@deloitte.com (D. Radke); edbowen@deloitte.com (E. Bowen)

[1]NIST 800-53r5 [2] states specifically that exfiltration lies within security control SC-07(10) for boundary protection to prevent unauthorized data movement (exfiltration).

exfil traffic through domain name system, (DNS)) to deter detection while obfuscating intent [8, 7, 6, 9, 5].

The previous literature's drawbacks are discussed in [1] and the topic of using RL for conducting post-exploitation activities such as exfiltration is still under-studied. Previous work [4], for example, employs ontological models of the agent with actions defined using common software modules. While this may be useful in some capacity, it suffers from aligning to network structure, path structure, and cyber terrain, thereby limiting its ability to anchor agents to the *real* computer network. In addition, the output is not operationally interpretable where security operations center (SOC) and cyber analysts can action results [6, 9, 7, 5].

This paper presents a framework for using RL methods for discovering exfiltration paths in network models while accounting for attacker preferences in payload and protocol preferences. The key contributions of this paper are twofold:

1. An approach for modeling data exfiltration on networks that accounts for choices in protocol with varying size of payload.
2. The implementation of RL-based algorithms for discovering exfiltration paths in network models.

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

The remainder of this work begins with a background on the use of RL for penetration testing followed by an exploration of the methods for modeling defensive terrain and discovering exfiltration paths. Then, the experimental design is described for evaluating the proposed approach, followed by an analysis of the experimental results, and a discussion of the findings. Lastly, this paper concludes with remarks on modeling decisions, a summary of the work, and possible avenues of future research.

2. RL and Penetration Testing

2.1. Reinforcement Learning

RL is a framework where an agent learns to optimize its behaviour by interacting with its environment [10]. A Markov decision process (MDP): $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ is often used to model the environment, where S is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function and $\gamma \in (0, 1]$ is the discount factor, which determines the present value of future rewards. The agent's behavior is characterized by its policy π, which is a probabilistic distribution over actions given a state. For deterministic policies, the action taken in state s can be denoted as $\pi(s)$. Corresponding to each time step, the agent observes a state s_t, on which it takes an action a_t according to $\pi(a|s_t)$, and transitions to a new state s_{t+1} and receives a reward $r_t = r(s_t, a_t, s_{t+1})$. The cumulative discounted reward is called the *return* and is defined as $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$. The RL agent aims to learn an optimal policy π^*, which maximizes the expected return from each state. RL algorithms can be categorized

into three groups: value function-based (or critic-only) methods, policy gradient (or actor-only) methods, and actor-critic methods.

Value function-based methods such as Q-learning [11] or deep Q-networkx (DQN) [12] learn optimal policies by first estimating the optimal action-value function $Q^*(s,a)$:

$$\begin{aligned} Q^*(s,a) &\equiv \max_{\pi} Q^{\pi}(s,a) \\ &\equiv \max_{\pi} \mathbb{E}_{\pi}\big[G_t | s_t = s, a_t = a\big], \end{aligned} \tag{1}$$

which can be obtained by solving the Bellman equation:

$$Q^*(s,a) = \mathbb{E}_{s'}\big[r + \gamma \max_{a'} Q^*(s',a') | s,a\big]. \tag{2}$$

Then, an optimal policy π^* is derived by selecting the action that yields the largest Q-value:

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

On the other hand, policy gradient approaches focus on directly parameterizing the policy $\pi(a|s;\theta)$ and optimizing a performance measure $J(\theta)$ such as the expected return $\mathbb{E}[G_t]$ via gradient ascent. Such methods often suffer from high variance and may result in slow learning. Thus, to reduce the variance, actor-critic methods use an estimate of the value function $V_{\pi}(s) \equiv \mathbb{E}_{\pi}[G_t|s_t = s]$ as a baseline when estimating the policy gradient $\nabla J(\theta)$ [13]. The critic is responsible for learning the value function while the actor updates policy parameters by using the estimated policy gradient. In particular, the policy gradient can be estimated as

$$\nabla J(\theta) \approx \mathbb{E}\big[\nabla_{\theta} \log \pi(a_t|s_t;\theta) A_t\big], \tag{4}$$

where $A_t = Q(s_t,a_t) - V(s_t)$ represents the *advantage* of taking action a_t at state s_t.

Policy gradient methods are prone to performance collapse as a result of large policy updates, which can be challenging to recover from because the agent will have been trained on the experience produced by bad policies. To improve training stability, Proximal Policy Optimization (PPO) [14] uses a clipped surrogate objective function:

$$\mathscr{L}(\theta) = \mathbb{E}\Big[\min\big(\rho_t(\theta)A_t, \mathrm{clip}\big(\rho_t(\theta), 1-\epsilon, 1+\epsilon\big)A_t\big)\Big], \tag{5}$$

where $\rho_t(\theta) = \pi_{\theta}(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ is the probability ratio of the new policy over the old policy. The advantage function A_t is often estimated using the generalized advantage estimation [15], truncated after T steps:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{6}$$

$$\text{where } \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t). \tag{7}$$

To support exploration, an entropy bonus $\beta H(\theta)$ is often added to the objective function (5), where β is a coefficient.

2.2. RL applications in penetration testing

Deep RL has been applied to cybersecurity broadly [13], but only recently it has been employed as a tool for penetration testing [16, 17, 18, 19, 20, 21, 22, 23]. There are a number of different approaches, but most only consider privilege escalation on a target host as the learning task. Gangupantulu et. al proposed to use concepts of cyber terrain to help enrich task design and reward shaping [23]. This concept spurred the development of several task-specific uses of RL for penetration testing, including crown jewel analysis [24], discovering exfiltration paths [1], and exposing surveillance detection routes [25].

As with Gangupantulu et al.[24], the presented RL approach here solves a more complex task and acts as a focused tool for cyber operators to increase the effectiveness of operator workflow in penetration testing. RL for penetration testing has made frequent use of DQN [17, 21, 22, 23, 24]. As an alternative, Nguyen et al. proposed an RL-based approach that makes use of two agents: one for iteratively scanning the network to build a structural model and another for exploiting the constructed model [26]. In this study, Nguyen et al.'s double agent architecture is combined with the PPO algorithm to train the RL agents.

3. Methods

In this section, we present the details of the exfiltration model, the protocol-based path selection criteria, and the complete RL formulation. While the model incorporates several assumptions, it's fundamentally based on a data-driven approach using scan data. This reliance on scan data not only ensures empirical robustness but also permits iterative refinements, as newer or more comprehensive data become available, to progressively approach a more accurate representation of reality.

3.1. Exfiltration Simulation Overview

The approach proposed here expands on previous models for data exfiltration in that it can model paths for different payload sizes. The exfiltration campaign is modeled based on three tasks consisting of (i) Connection, (ii) Path Selection and (iii) Exfiltration. The attacker initially attempts to gain control of some of the known target hosts which are externally connected via an internet connection to serve as the point of exfiltration. Once control of the target host is gained, an exfiltration path is selected based on the preferences for an exfiltration protocol. The attacker then tries to exfiltrate data packets from the compromised host. The three tasks are designed to function so that if an attacker discovers a new exfiltration host, the path selection module determines whether a better path exists and adjusts the exfiltration path accordingly.

The agent explores the network and gathers information on neighboring hosts by taking the subnet scan action. In order for the scan to be successful, the agent must first gain access to the underlying host, which can be achieved by executing an exploit action. Multiple exploits may exist for a given machine, with each targeting a specific Common Vulnerabilities and Exposures (CVE) vulnerability.

Once a foothold is gained on a new host, the agent updates candidate exfiltration paths that consist of each of the compromised hosts. It will then decide which path is preferred to carry out

the exfiltration based on the predetermined Exfiltration protocol strategy. If another target is captured later, the path selection task evaluates the new paths available and, if a new preferred path is discovered, the path is updated and the payload is reset to its original value.

After identifying the preferred exfiltration path, the agent can start sending parts of the payload to the exit node. The task is completed if the entire payload is uploaded from the initial node. In order to evade firewall detection, the agent should avoid frequent and large uploads. To hide its activity, the agent may take a sleep action that simply does nothing for a period of time.

3.2. Network Firewalls

As in [27], any exfiltration traffic will be monitored by network firewalls, which are placed between each of the subnets and the public Internet. Upon detection of unusual traffic patterns, the administrator will be alerted and an emergency firewall update will be conducted. Examples of suspicious activities include the following:

- the total egress volume exceeds `max_upload_volume`;
- the total active time surpasses `max_upload_time`.

Table 1 lists the values of firewall-related parameters used in the experiments.

Table 1
List of Firewall Parameters

Firewall Parameter	Value
max_upload_volume (MB)	5000
max_upload_time (minutes)	4
update_frequency (hours)	24

Firewalls are also updated periodically. In particular, a wall-clock is introduced to simulate the real time of an attack campaign. Different actions will increase the clock time by different amounts depending on the their complexity. Both the regular update and the emergency update will patch the vulnerabilities and block the outbound traffic from the compromised hosts.

3.3. Protocol-Based Path Selection

Exfiltration activities within attacker campaigns are typically carried out by exploiting a common protocol as these are deemed generally safer and less likely to be detected by security monitoring. Standard protocols, such as Hypertext Transfer Protocol Secure (HTTPS), are often used to carry out data exfiltration. By using common protocols used by enterprise applications, it's more likely these protocols are available. It's also more likely these protocols are not monitored as closely by security detection methods. As an example, by using the same protocol used by databases to backup their data to cloud services, attackers emulate the database backup expected by security rules and do not raise alerts in monitoring systems.

Path selection is determined by maximizing the utilization of this protocol across as many hosts in an exfiltration network path as possible. This is not always the shortest path. A path maximizing the use of the chosen protocol is often more advantageous, even when this

path touches more nodes in the victims network. The path selection algorithm accounts for contingencies when end-to-end use of the designated exfiltration protocol is unavailable. The algorithm prioritizes finding a complete path using the given protocol over a shortest path possible. The next level of criterion considered are the length of the path and rewards accumulated. If multiple paths are identified with the same exfiltration protocol coverage, the shortest path will be prioritized. When no complete path can be created using the exfiltration protocol, the algorithm searches for the shortest path exposed to the maximum use of the protocol. The reward function calculates the highest rewarded path using existing reward mechanisms, shortest number of hosts, and maximum use of the exfiltration protocol.

3.4. Reinforcement Learning Formulation

3.4.1. State Space

The state has the following features for every host:

- Address,
- Operating system,
- Services and processes,
- Discovery value and status,
- Infection value and status,
- Access level information.

Host's address is denoted by its subnet ID and local ID. The operating system, service and process features have a value of one of if they are present at the host and zero otherwise. Similarly, the discovery and infection status are one if the host is discovered or compromised and zero otherwise. The discovery and infection values represent the reward for successfully discovering and compromising a host, respectively. Additional features are defined for target hosts:

- Connection status,
- Time since infection,
- Remaining payload size,

The connection status can be connected, not connected, or isolated (i.e., blocked by firewalls). The time since infection is measured by the wall-clock rather than time steps. Finally, the remaining payload size indicates how much left to upload. The exfiltration task is complete when the remaining payload size becomes zero.

3.4.2. Action Space

There are four types of actions for the RL agent: *subnet scan*, *exploit*, *upload*, and *sleep*. Each action requires specification of a target host, except for the sleep action, which simply does nothing for a given period of time. Multiple exploits targeting at different vulnerabilities may be available for a given host. Two uploading actions with different speed are available at each target - one with a rate of 100MB/s and another with rate 1MB/s.

Table 2
List of actions

Action Type	Time
Subnet Scan	30
Exploit	10
Upload	10
Sleep	60

Clock-time increases differently based on the action's result and complexity. Table 2 lists the assigned clock time for each action. For not applicable actions, such as performing as subnet scan without access to the underlying host, the clock time will only move forward by one second.

3.4.3. Reward Function

The reward function consists of a positive value for achieving sub-goals such as discovering or exploiting a host and a negative value that accounts for the action's cost. An action with higher cost is more likely to trigger the defense terrain. Specifically, we follow the approach in [1] and assign action' cost based on the services running on the target system. The idea is that even though the adversaries may not know the exact defense mechanism or strength, they can still infer the presence of defense based on the host's service information. In particular, we categorized the services into three groups, high-risk, medium-risk, and low-risk. The actual cost of an action then depends on its type (scan, exploit, or upload) and the target's service profile.

Rewards are given based on how much of the exfiltration path chosen is covered by the exfiltration protocol, For example if out of the 6 hosts in the exfiltration path 3 hosts have exfiltration protocol running then 50 percent of the reward configured will be given to the agent. The agent receives positive reward on uploading a partial payload from the infected host, upon finishing sending the entire payload, the agent is given a large bonus reward. However, if exfiltration is detected by network firewalls, then the agent will receive a penalty equal to the total accumulated rewards gained on the originating host and the host will be isolated. That is, the agent will lose all rewards from discovery, infection and partial uploads. Table 3 lists rewards used in this study.

Table 3
List of rewards

Reward Type	Value
Discovery	1000
Exploit	1000
Exfiltration Protocol Path	1000
Upload (per unit)	0.1
Upload (bonus)	10000

4. Experiments

In this section we present the experiment details and the results, and discuss key characteristics of the attack paths learned by the RL agent.

4.1. Network Description

We have designed two experiment networks. The first experiment network has 10 subnets and a total of 56 hosts. Each subnet contains between 3 and 12 hosts. The attacker agent is assumed to have gained an initial foothold on host (8, 2) in subnet 8, which is not directly connected to the Internet. One particular machine (2, 0) from subnet 2 is designated as the exfiltration host. Subnet 2 is directly accessible from the internet whereas, other subnets are private and are not directly accessible from the Internet. The exfiltration host has Dynamic Host Configuration Protocol Server (DHCPS) running as a service, which is chosen as the Exfiltration Protocol. The second experiment network has 101 subnets and a total of 1444 hosts. This network is remarkably bigger than the one used previously. Each subnet contains between 3 and 50 hosts. The attacker agent is assumed to have gained an initial foothold on host (44, 5) in subnet 44, which is not directly connected to the Internet. A host connected to the internet (5, 10) from subnet 5 is designated as exfiltration host. The exfiltration host has running HTTPS service, which is chosen as the Exfiltration Protocol.

4.2. Training Details

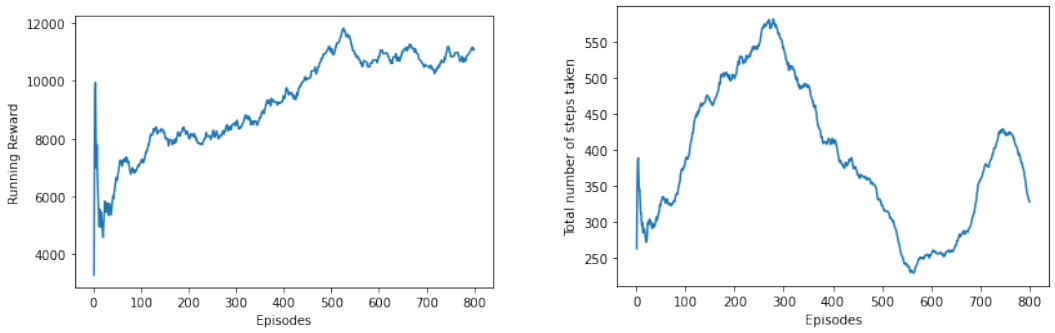
The RL agent is trained in an episodic fashion for both the networks using the well-known PPO algorithm. An episode ends when the initial host either completes sending payload to the exfiltration host or is isolated by firewalls. The target payload is set to be 10,000MB. Both the actor and the critic are approximated by a two-layer feed-forward neural network, where the first layer has 64 neurons, and the second layer has 32 neurons. Other key hyperparameters are listed in Table 4. For the first network the RL agent is trained for 800 episodes and for the second network RL agent is trained for 1000 episodes.

Table 4
List of hyperparameters

Hyperparameter	Value
Critic learning rate (α_w)	0.0003
Actor earning rate (α_θ)	0.0003
Discount factor (γ)	0.99
Horizon (T)	2048
Minibatch size	32
Epochs	5
GAE parameter (λ)	0.95
Clipping parameter (ϵ)	0.2
Entropy coefficient (β)	0.02

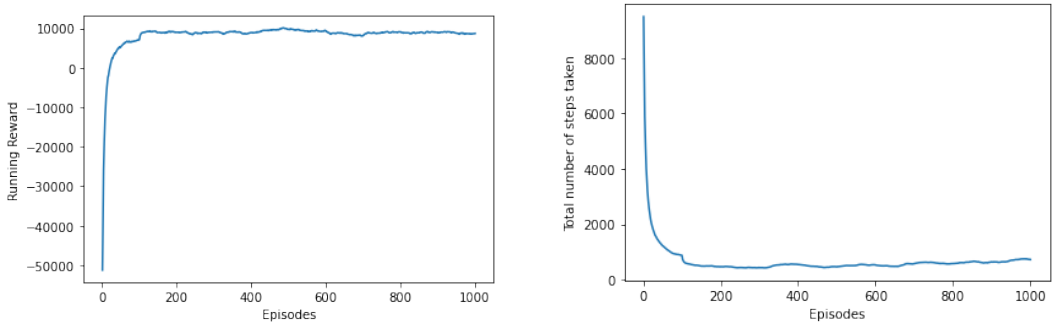
5. Results

For the first network, episode rewards over training runs are presented in Fig. 1a and episode length in Fig. 1b, and for the second network, episode rewards over training runs are presented in Fig. 2a and episode length in Fig. 2b. Training is observed to be stable for both networks, and the RL policy converges in 800 episodes for the first network and in 1000 episodes for the second network. Fig. 1a shows that the sum of rewards in an episode for the first network steadily increases to almost 12,000, and Fig. 2a shows that the sum of rewards in an episode for second network steadily increases to a little more than 10,000. During the same intervals, the episode length gradually decreases for both simulations. This suggests that as training goes on, the RL agent completes the attack task more efficiently and takes fewer random actions.



(a) (b)

Figure 1: Average episode reward (left) and length (right) from the first network.



(a) (b)

Figure 2: Average episode reward (left) and length (right) from the second network.

Table 5 reports statistics on the length and rewards from the generated attack paths for the first network. On average, the RL agent finishes the task in 389 steps and receives a total reward of 9298. Table 6 reports statistics on the length and rewards from the generated attack paths for the second network. On average, the RL agent finishes the task in 670 steps and receives a total reward of 8252.

Table 5
Statistics of the generated attack paths for First Network

	Steps	Rewards
Mean	389	9298
Std	98	1618
Min	229	3292
Max	582	11814

Table 6
Statistics of the generated attack paths for Second Network

	Steps	Rewards
Mean	670	8252
Std	608	3898
Min	427	-51249
Max	9493	10103

Table 7
List of main steps taken by the RL agent for First Network

Action	Target
Subnet Scan	(8, 2)
Exploit	(4, 2)
Subnet Scan	(4, 2)
Exploit	(2, 0)
Exploit	(6, 0)
Subnet Scan	(6, 0)
Exploit	(5, 1)
Upload(10 MB)	(8, 2)
Upload(1000 MB)	(8, 2)
Sleep(NoOp)	-

Due to the stochastic nature of the learned policy, the RL agent may take some unnecessary or redundant actions such as exploiting unimportant hosts or subnet scans. After pruning the output trajectory, key steps in the attack for the simulation of the first network can be identified as shown in Table 7.

For the first network, the agent gains a foothold on host (8, 2) in subnet 8, from which it triggers a subnet scan which leads to the discovery of other hosts in the same subnet and in the connected subnets, subnet 4 and subnet 6. The agent then exploits the host (4, 2) in subnet 4 and it is chosen as a host for further exploitation to make an exfiltration path. A subnet scan is triggered from the host (4, 2), which discovers the hosts present in connected subnets i.e., subnet 2 and ultimately discovers the target or exfiltration host (2, 0), which is then compromised to forge an exfiltration path i.e., $(8, 2) \rightarrow (4, 2) \rightarrow (2, 0)$. In search of availability of better paths, agent exploits the host (6, 0) in subnet 6, and triggers a subnet scan from that host, discovering hosts on connected subnet i.e., subnet 5. This scan discovers host (5, 1) in subnet 5 and is later exploited to forge another exfiltration path i.e., $(8, 2) \rightarrow (6, 0) \rightarrow (5, 1) \rightarrow (2, 0)$.

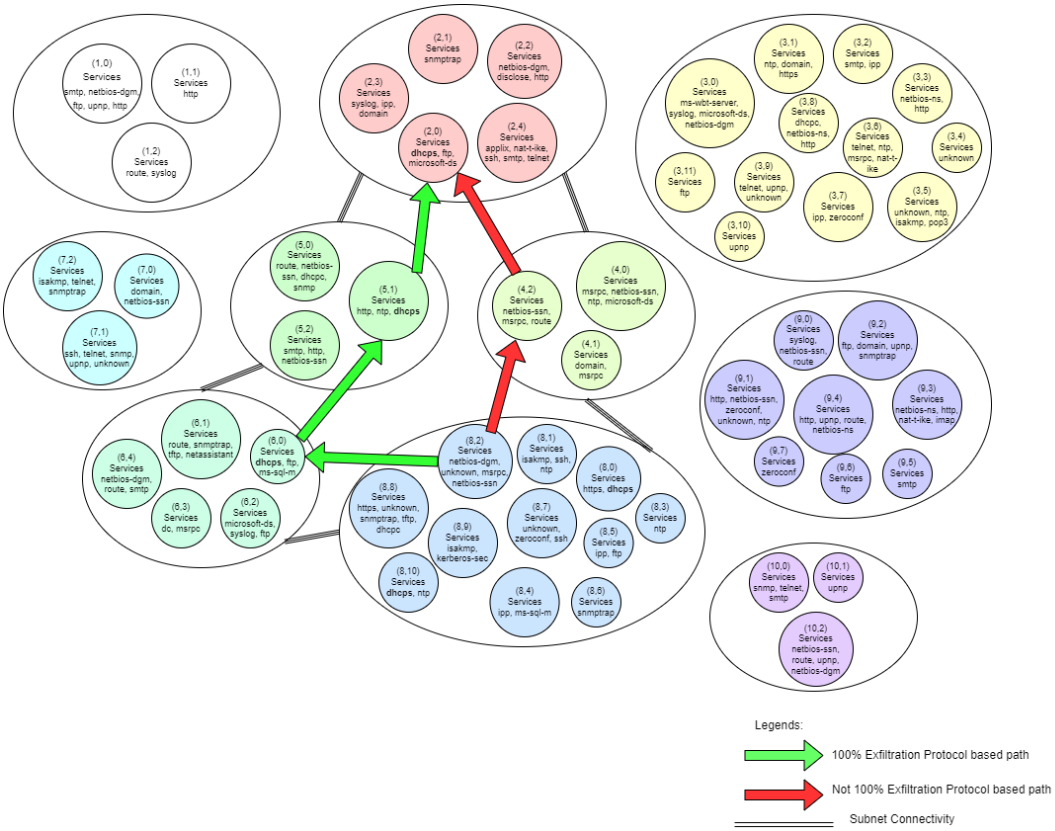


Figure 3: Network diagram showing the two exfiltration paths found. The green path is preferred over the read one as it utilizes the same protocol (i.e., DHCP).

The path explored earlier i.e., $(8, 2) \rightarrow (4, 2) \rightarrow (2, 0)$ is not a complete exfiltration protocol-based path, since there is no DHCP (exfiltration protocol) service running on host (4, 2) as shown in Fig. 3. However, the second path discovered i.e., $(8, 2) \rightarrow (6, 0) \rightarrow (5, 1) \rightarrow (2, 0)$ is a complete exfiltration protocol-based path since the same service (i.e., DHCP) is running on both hosts (6, 0) and (5, 1) as shown in Fig. 3. Noticeably, the agent chooses the second path over first path to upload payload as it is 100 percent protocol-based path and is the optimal path, even though the first path discovered is shorter in length.

For the second network, the agent has a foothold over the host (44, 5) in subnet 44. Upon performing various subnet scans and exploits, the agent gets a hold over the host (24, 18), and ultimately discovers and exploits the target or exfiltration host (5, 10). This led to development of exfiltration path i.e., $(44, 5) \rightarrow (24, 18) \rightarrow (5, 10)$. The host (24, 18) has HTTPS service running on it, hence the path forged is a complete protocol-based path. The capability of the agent to forge a 100 percent protocol-based path over such a big network indicates that the model is scalable as well.

For both networks the agent takes appropriate sleep actions in between the upload actions so that there is no unusual traffic pattern and cyber defenses are not triggered.

The agent found paths in both networks that utilize a single network protocol. In real-life

scenarios, attackers try to use a single protocol to avoid increasing attack complexity and reduce the risks of inconsistencies or errors, which can lead to a greater possibility of detection. Choosing to exfil data using existing network protocols that the network defenses (firewalls, IDS) know about also reduces the risk of discovery by traffic anomaly detection algorithms. Using standard protocols for exfiltration while considering traffic timing and volume replicates previously documented Tactics, Techniques, and Procedures (TTP)s[28].

While novel exfiltration methods that use non-standard protocols exist,Domain Name Service (DNS), Network Time Protocol (NTP), or Internet Control Message Protocol (ICMP), they typically require complex setup for execution [8]. They also are usually more closely monitored by defensive measures for volume and anomalous behaviors than standard protocols due to their usage in previous exfiltration operations [8]. Data exfiltration requires more network volume and can be more stealthily sent over less strictly monitored or eccentric channels [29].

6. Conclusion

The current gap within the cybersecurity industry involves contextualizing and quantitatively prioritizing the efficacy of deployed security controls to enable sense-making for security practitioners and network defenders. In this paper, we address this gap through applying RL for exfiltration path analysis enhanced by integrating protocol and payload considerations. Our work demonstrates that an RL agent can effectively find an exfiltration path with maximum exfiltration protocol coverage and can perform exfiltration using this preferred path without being detected by security infrastructure (i.e., firewalls). Our results identify optimal paths that provide insights for operators, analysts, and defenders to evaluate the value of currently deployed security controls which influence (i.e., isolate or eliminate) the connections within the path. As a result, the operations community can utilize this data to formulate task lists for securing enterprise networks.

This RL approach identified the most likely hosts and services used when exfiltrating data while capturing variable metrics used in network risk assessments. The strength of this approach was validated through identification of intentional network misconfigurations that mimic real-world vulnerabilities. In future work we consider expanding the risk formalism to increase its sophistication and maturity, which will drive increased applicability and relevance.

References

[1] T. Cody, A. Rahman, C. Redino, L. Huang, R. Clark, A. Kakkar, D. Kushwaha, P. Park, P. Beling, E. Bowen, Discovering exfiltration paths using reinforcement learning with attack graphs, in: 2022 IEEE Conference on Dependable and Secure Computing (DSC), IEEE, 2022, pp. 1–8.

[2] N. I. of Standards, Technology, Security and Privacy Controls for Federal Information Systems and Organizations, Technical Report NIST Special Publication 800-53 Revision 5, U.S. Department of Commerce, Washington, D.C., 2020.

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

[4] R. Maeda, M. Mimura, Automating post-exploitation with deep reinforcement learning, Computers & Security 100 (2021) 102108.
[5] J. Ahmed, H. H. Gharakheili, Q. Raza, C. Russell, V. Sivaraman, Real-time detection of dns exfiltration and tunneling from enterprise networks, in: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), IEEE, 2019, pp. 649–653.
[6] A. Nadler, A. Aminov, A. Shabtai, Detection of malicious and low throughput data exfiltration over the dns protocol, Computers & Security 80 (2019) 36–53.
[7] M. Zhan, Y. Li, G. Yu, B. Li, W. Wang, Detecting dns over https based data exfiltration, Computer Networks 209 (2022) 108919.
[8] J. Zhang, L. Yang, S. Yu, J. Ma, A dns tunneling detection method based on deep learning models to prevent data exfiltration, in: Network and System Security: 13th International Conference, NSS 2019, Sapporo, Japan, December 15–18, 2019, Proceedings 13, Springer, 2019, pp. 520–535.
[9] A. Das, M.-Y. Shen, M. Shashanka, J. Wang, Detection of exfiltration and tunneling over dns, in: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2017, pp. 737–742.
[10] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT press, 2018.
[11] C. J. C. H. Watkins, Learning from delayed rewards (1989).
[12] 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 518 (2015) 529–533.
[13] T. T. Nguyen, V. J. Reddi, Deep reinforcement learning for cyber security, arXiv preprint arXiv:1906.05799 (2019).
[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).
[15] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, arXiv preprint arXiv:1506.02438 (2015).
[16] M. C. Ghanem, 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.
[17] J. Schwartz, H. Kurniawati, Autonomous penetration testing using reinforcement learning, arXiv preprint arXiv:1905.05965 (2019).
[18] M. C. Ghanem, T. M. Chen, Reinforcement learning for efficient network penetration testing, Information 11 (2020) 6.
[19] S. Chaudhary, A. O'Brien, 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.
[20] M. Yousefi, N. Mtetwa, Y. Zhang, 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.
[21] A. Chowdhary, D. Huang, J. S. Mahendran, D. Romo, Y. Deng, A. Sabur, Autonomous security analysis and penetration testing, in: 2020 16th International Conference on Mobility, Sensing and Networking (MSN), IEEE, 2020, pp. 508–515.

[22] Z. Hu, R. Beuran, 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.
[23] R. Gangupantulu, T. Cody, P. Park, A. Rahman, L. Eisenbeiser, D. Radke, R. Clark, Using cyber terrain in reinforcement learning for penetration testing, arXiv preprint arXiv:2108.07124 (2021).
[24] R. Gangupantulu, T. Cody, A. Rahman, C. Redino, R. Clark, P. Park, Crown jewels analysis using reinforcement learning with attack graphs, arXiv preprint arXiv:2108.09358 (2021).
[25] L. Huang, T. Cody, C. Redino, A. Rahman, A. Kakkar, D. Kushwaha, C. Wang, R. Clark, D. Radke, P. Beling, et al., Exposing surveillance detection routes via reinforcement learning, attack graphs, and cyber terrain, arXiv preprint arXiv:2211.03027 (2022).
[26] H. V. Nguyen, S. Teerakanok, A. Inomata, T. Uehara, The proposal of double agent architecture using actor-critic algorithm for penetration testing., in: ICISSP, 2021, pp. 440–449.
[27] C. Wang, A. Kakkar, C. Redino, A. Rahman, S. Ajinsyam, R. Clark, D. Radke, T. Cody, L. Huang, E. Bowen, Discovering command and control channels using reinforcement learning, in: SoutheastCon 2023, IEEE, 2023, pp. 685–692.
[28] Mitre att&ck framework®, 2021. URL: https://attack.mitre.org.
[29] B. Sabir, F. Ullah, M. A. Babar, R. Gaire, Machine learning for detecting data exfiltration: A review 54 (2021).