Deterring Adversarial Learning in Penetration Testing by Exploiting Domain Adaptation Theory

Shreyas Bera, Liam Glenn, Abhay Raghavan, Emma Meno, Tyler Cody, Peter A. Beling
National Security Institute
Virginia Tech
Arlington, VA, USA

***Abstract*—Artificial intelligence (AI) and machine learning (ML) are increasingly being used in cyber operations. Because of techniques like adversarial learning, the performance of network defenses can degrade quickly. Thus, there is an increasing need for adaptable, dynamic network defenses. Correspondingly, there has been a rise in the use of reconfiguration schemes like moving target defense in software-defined networks. However, moving target defense methods target individual adversaries and rely on an in-depth understanding of an adversary's utility function. In contrast, domain adaptation theory suggests that learning agents are sensitive to distributional changes in their inputs, regardless of their utilities. In this paper, we identify several kinds of network changes that deter adversaries by exploiting vulnerabilities in their learned assumptions. We use an open source network attack simulator, NASim, to conduct experiments on reinforcement learning (RL)-based penetration testers. We measure the time-to-relearn in order to compare the efficacy of different network changes at deterring adversaries. We find that by focusing on shifting the learning domain as a defensive strategy, we are able to degrade the performance of multiple adversaries simultaneously. With our methodology, cyber defenders have tools that allow them to raise the sophistication and cost needed by adversaries to remain a threat to network operations over time.**

***Index Terms*—adversarial learning, penetration testing, cybersecurity, reinforcement learning**

I. Introduction

Artificial intelligence (AI) and machine learning (ML) are becoming increasingly prevalent in cyber operations [1], including network defense. However, techniques such as adversarial learning can quickly degrade the performance of network defenses [2]. Therefore, there is a growing need for adaptable and dynamic network defenses that can respond to changes in attack patterns. To address this challenge, reconfiguration schemes like moving target defense (MTD) have been used in software-defined networks [3]. However, these methods target individual adversaries and rely on a thorough understanding of their utility functions.

In contrast, domain adaptation theory suggests that learning agents are sensitive to distributional changes in their inputs, regardless of their utilities [4]. In this paper, we propose a new defensive strategy based on domain adaptation theory that focuses on shifting the learning domain to deter adversaries by exploiting vulnerabilities in their learned assumptions.

We identify several kinds of network changes that can be used to deter adversaries and measure their efficacy using an open source network attack simulator, NASim [5]. Specifically, we conduct experiments on reinforcement learning (RL)-based penetration testers [6] and measure the time-to-relearn in order to compare the efficacy of different network changes at deterring adversaries. Our results demonstrate that by focusing on shifting the learning domain, we can degrade the performance of multiple adversaries simultaneously and increase the cost needed by adversaries to remain a threat to network operations over time.

The remainder of this paper is organized as follows. In Section II, we provide a brief overview of related work on penetration testing with RL and on domain adaptation. In Section III we describe our methodology, including the network changes we propose and the experimental setup. In Section IV we present and discuss our results. Finally, in Section V, we draw conclusions and discuss future work.

II. Background

A. Penetration Testing with RL

Many modern systems are cyber-physical, and thus modern test and evaluation (T&E) includes both cyber and physical aspects. In cyber T&E, vulnerability assessment and penetration testing are two primary, related, and distinctive activities [7]. Vulnerability assessment is a proactive measure that helps organizations identify and address potential weaknesses in their systems, while penetration testing is a reactive measure that tests the ability of an organization's defenses to withstand attacks. Both activities are critical to a comprehensive security strategy, and should be performed on a regular basis to ensure that an organization's defense remain effective over time.

The desire for cheaper and higher frequency penetration testing has led to research in automation. RL for automated penetration testing is a recent area of interest by the community [6], [8]–[19]. RL for penetration testing treats the activity as a sequential decision making process, specifically, as a Markov decision process (MDP). MDPs consist of states, actions, rewards, transition probabilities, and the admissible set of actions for each state. RL agents optimize MDPs by maximizing the expected sum of future discounted rewards. That is, they take the actions that result in the most rewards over time, subject to a discount rate that relates the value of rewards in the future to the value of rewards in the present.

RL for penetration testing is a kind of adversarial learning that can degrade network defenses over time. By re-configuring networks, there is a chance that the adversarial learning can be deterred or even taken advantage of. Typically, moving target defense is used to optimize defensive re-configurations. The setting for MTD is a game between a defender and attacker inspired by notions of prey evading predators. MTD seeks advantageous change to systems and stratagems as a defensive policy. Typically, specific knowledge regarding the payoffs or adversary capabilities is exploited in MTD solution methods. But these game theoretic solutions have drawbacks, for example, they typically try to address a single payoff structure, i.e., only one kind-of adversary at a time. In this paper, motivated by domain adaptation theory, we explore whether changes made based on the transition probabilities of the MDP describing the network can be used to deter multiple adversaries at once.

B. Domain Adaptation Theory

Domain adaptation theory is a subfield of learning theory concerned with the effect that changing domains has on learning and predictive performance [4]. Domain adaptation theory has shown that learned models are sensitive to changes in input data (such as distribution and complexity) at a fundamental level [20]. This finding has been validated empirically [21]. In other words, without knowing exactly what adversaries are doing or what their payoffs are, if can know roughly how hard it is for them to adapt to a change in general, then we have a powerful mechanism that can degrade the performance of many adversaries at once. This paper explores whether this is a possible means of deterring adversarial learning against network defenses.

C. NASim

In our paper, we use NASim as an experimental testbed [5]. NASim simulates a network containing target host(s) with pre-established "firewalls" and rewards the agents for successful privilege escalation on the sensitive hosts. NASim enables developers to create RL-based penetration testing agents for custom or pre-built network scenarios built with *.yaml configuration files. The configuration files define the networks structure as well as define various costs and probabilities. NASim also has an implementation for deep Q-learning (DQN), a neural network based approach to RL.

The network configuration is defined by the number and size of the subnets in the network, how the different subnets in the network are connected, the address, OS, services, processes, and firewalls for each host in the network, and which communication is prevented between subnets. The penetration testing agents are defined by the set of exploits available, the set of privilege escalation actions available, the cost of performing each scan type (service, OS, process, and subnet), and the target hosts on the network and their value.

At each time step, the state space that the RL agent observes consists of all hosts on the network and whether or not there was an error resulting from the last action (connection error, permission error, or undefined error). The actions consist of scanning, exploiting, and privilege escalation. The actions for each subnet and host are represented individually. For example, a host with two available exploits and two available privilege escalations would have 4 actions, plus 3 more actions for service, OS, and process scans. If other hosts have the same exploits available, they will be represented as additional dimensions of the action space. The actions are concatenated and indexed by a vector. More information on the agents can be found in the NASim documentation [5].

III. Methods

We use the pre-built network scenario 'small.yaml'[1]. This scenario contains 4 subnets, 8 hosts, 2 operating systems, 3 services, 2 processes, 3 exploits, and 2 privilege escalations. There is a subnet connected to directly to the internet, a buffer subnet between the internet-connected host and users, a user subnet, and a final subnet that is the furthest from the internet and also contains the target host. More information on 'small.yaml' can be found in the NASim documentation [5].

To represent 3 different agents, we designed 3 different reward functions using the scan cost parameters of the configuration files. Agents A, B, and C correspond to scan costs of 1, 3, and 5, respectively. For the agents, we used the out-of-the-box DQN implementation provided by NASim, which includes ϵ-greedy linear annealing and experience replay. The ϵ-greedy approach takes a random action with probability ϵ and takes the greedy action (i.e., the action that maximizes reward) with probability $1-\epsilon$. Linear annealing linearly decreases the value of ϵ as training progresses. Experience replay stores a buffer of past data that is randomly sampled to remove correlations between successive observations and smooth distributional changes in the data.

We identified changes to the success probabilities as our method for testing whether changes to learning processes can be used to degrade the performance of all three agents at once. Other changes, like removing certain hosts or actions, led to changes in the state and actions space, which complicated the ability to compare performance before and after the change occurs. For the changes, we created differences in the probability of success of actions by ±10% on exploits and privilege escalations. These changes simulate changes to a network's firewall that a network administrator could make by changing configurations or adding defensive systems. The changes made were: 10% decrement to the daclsvc privilege escalation, 10% decrement to the tomcat privilege escalation 10% decrement to exploit success, and 10% increment to exploit success.

IV. Results

First, we trained agents A, B, and C on the unchanged 'small.yaml' network. The agents were trained for 10,000 episodes with 1×10^6 exploration steps for linearly annealing ϵ from 0.95 to 0.05.

[1]We changed the target hosts to only contain the (4, 0) target in increase the learning speed.

TABLE I
Reward and Steps on Original Network by Agent

Agent	Reward (Mean, Std)	Steps (Mean, Std)
A	(82, 5.3)	(9.7, 3)
B	(75, 11)	(9.4, 3)
C	(74, 8.0)	(9.2, 3)

When ϵ is 0.95, there is a 95% chance of taking a random action. By the end of linear annealing, there is only a 5% chance of taking a random action. The results are shown in Table I. The terminal reward for escalating privileges on the target host is 100 and there is a -1 reward for each action taken, except for scans, which have -1, -3, and -5 rewards for agents A, B, and C, respectively. Thus, the agents performed well, and a slight decrease in reward can be seen as scan cost increases. The number of steps are similar, indicating the scan cost difference drove the difference in performance. There is variance to the results since the action selection is stochastic and the action success probabilities are stochastic.

We tested the changes one by one for each agent. In all cases, there was a significant drop off in performance, which can be seen in Figure 1. This shows that the agents were very fragile to change. It could also be a symptom of overfitting to the particular network configuration. To test how long the adversaries would be deterred based on each change, we retrained the agents on each change for 10,000 episodes and identified where during retraining their reward stabilized. Figure 1 shows the retraining rewards plotted against the number of episodes for each change for agent A. The tomcat change had significantly more variance from episode to episode than the other methods, and it took many episodes for that variance to decrease. Similar trends were found for agents B and C. The number of episodes to converge is summarized in Table II. We can see that agents A, B, and C each take longer to retrain, in general, respectively. We can also notice that the changes to privilege escalations (daclsvc and tomcat) take longer to retrain than changes to exploits. In practice, this would suggest that increasing security for privilege escalation will degrade adversary performance for longer than increasing security around exploits. Moreover, the trend for each agent is similar, suggesting that the changes to the learning process had similar effects for all agents desipte the differences in their reward function. This suggests that challenges in domain adaptation can be exploited to affect multiple adversaries at once.

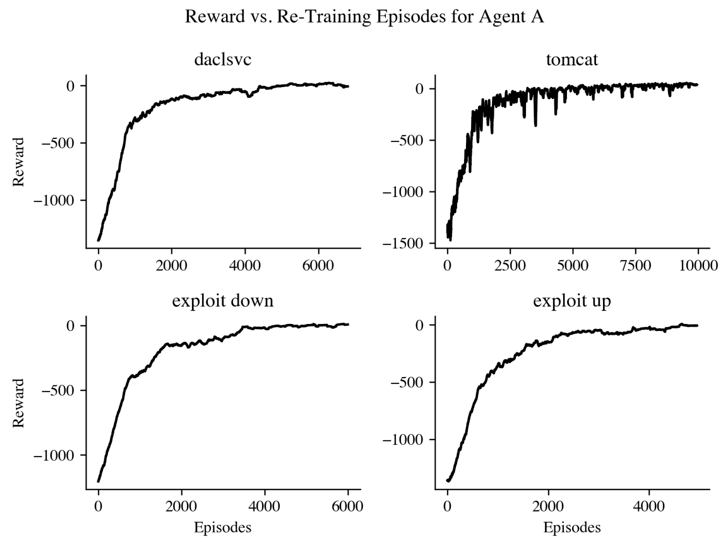

Fig. 1: A plot showing the number of re-training episodes it takes Agent A to converge after changes have occured to the network configuration.

TABLE II
NUMBER OF RE-TRAINING EPISODES TO CONVERGE

Agent	daclsvc	tomcat	exploit down	exploit up
A	6400	10000	6100	4800
B	7500	10000	5200	5500
C	8100	10000	7400	6300

V. CONCLUSION

Our proposed strategy, based on domain adaptation theory, focuses on shifting the learning domain to deter multiple adversaries simultaneously. We have demonstrated the existence of this phenomena using a case study in RL for penetration testing. By exploiting vulnerabilities in the adversaries' learned assumptions, we can increase training cost needed by adversaries to remain a threat to network operation over time.

There are several limitations to our study that should be addressed in future work. Originally, we tried to use the 'medium.yaml' network which had more subnets with more hosts, and thus the changes would like create larger differences in the optimal path. However, the 'medium.yaml' network took too long to train the DQN agents. Future work should implement more efficient RL algorithms and consider larger networks where the optimal path changes. Also, we only varied agents A, B, and C by varying their reward function. Future work should consider different levels of sophistication, e.g., by comparing various reinforcement learning algorithms, some which have transfer learning to help in adaptation and some that do not. Lastly, we avoided changes to the state and action space by focusing on changes to transition probabilities. Future work should address changes that alter the presence of hosts, the topology of connectivity between hosts, and the available exploits and privilege escalations.

ACKNOWLEDGEMENTS

Project sponsored by the National Security Agency under Grant Agreement Number H98230-21-1-0322. The United States Government is authorized to reproduce and distribute reprints notwithstanding any copyright notation herein.

REFERENCES

[1] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE Access*, vol. 8, pp. 222 310–222 354, 2020.
[2] A. Alotaibi and M. A. Rassam, "Adversarial machine learning attacks against intrusion detection systems: A survey on strategies and defense," *Future Internet*, vol. 15, no. 2, p. 62, 2023.
[3] S. Sengupta, A. Chowdhary, A. Sabur, A. Alshamrani, D. Huang, and S. Kambhampati, "A survey of moving target defenses for network security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1909–1941, 2020.
[4] I. Redko, E. Morvant, A. Habrard, M. Sebban, and Y. Bennani, "A survey on domain adaptation theory: learning bounds and theoretical guarantees," *arXiv preprint arXiv:2004.11829*, 2020.
[5] J. Schwartz and H. Kurniawatti, "NASim: Network attack simulator," https://networkattacksimulator.readthedocs.io/, 2019.

[6] T. Cody, "A layered reference model for penetration testing with reinforcement learning and attack graphs," in *2022 IEEE 29th Annual Software Technology Conference (STC)*. IEEE, 2022, pp. 41–50.

[7] K. Thakur, M. Qiu, K. Gai, and M. L. Ali, "An investigation on cyber security threats and security models," in *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*. IEEE, 2015, pp. 307–311.

[8] M. C. Ghanem and T. M. Chen, "Reinforcement learning for efficient network penetration testing," *Information*, vol. 11, no. 1, p. 6, 2019.

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

[10] T. Cody, P. Beling, and L. Freeman, "Towards continuous cyber testing with reinforcement learning for whole campaign emulation," in *2022 IEEE AUTOTESTCON*. IEEE, 2022, pp. 1–5.

[11] 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.

[12] T. Cody, A. Rahman, C. Redino, L. Huang, R. Clark, A. Kakkar, D. Kushwaha, P. Park, P. Beling, and 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.

[13] Y. Li, J. Yan, and M. Naili, "Deep reinforcement learning for penetration testing of cyber-physical attacks in the smart grid," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 01–09.

[14] 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.

[15] R. Gangupantulu, T. Cody, A. Rahma, C. Redino, R. Clark, and P. Park, "Crown jewels analysis using reinforcement learning with attack graphs," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2021, pp. 1–6.

[16] T. Y. Zhou, Y. C. Zang, J. H. Zhu, and Q. X. Wang, "NIG-AP: a new method for automated penetration testing," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 9, pp. 1277–1288, 2019.

[17] R. Gangupantulu, T. Cody, P. Park, A. Rahman, L. Eisenbeiser, D. Radke, R. Clark, and C. Redino, "Using cyber terrain in reinforcement learning for penetration testing," in *2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. IEEE, 2022, pp. 1–8.

[18] F. M. Zennaro and L. Erdodi, "Modeling penetration testing with reinforcement learning using capture-the-flag challenges: trade-offs between model-free learning and a priori knowledge," *arXiv preprint arXiv:2005.12632*, 2020.

[19] K. Tran, M. Standen, J. Kim, D. Bowman, T. Richer, A. Akella, and C.-T. Lin, "Cascaded reinforcement learning agents for large action spaces in autonomous penetration testing," *Applied Sciences*, vol. 12, no. 21, p. 11265, 2022.

[20] T. Cody and P. A. Beling, "A systems theory of transfer learning," *arXiv preprint arXiv:2107.01196*, 2021.

[21] T. Cody, S. Adams, and P. A. Beling, "Empirically measuring transfer distance for system design and operation," *IEEE Systems Journal*, vol. 16, no. 3, pp. 4962–4973, 2022.