

Towards Attacker Type Agnostic Cyber Defense Agents

Erick Galinkin, Emmanouil Pountourakis, Spiros Mancoridis

¹Drexel University
eg657@drexel.edu

Abstract

With computing now ubiquitous across government, industry, and education, cybersecurity has become a critical component for every organization on the planet. Due to this ubiquity of computing, cyber threats have continued to grow year over year, leading to labor shortages and a skills gap in cybersecurity. As a result, many cybersecurity product vendors and security organizations have looked to artificial intelligence to shore up their defenses. This work considers how to characterize attackers and defenders in one approach to the automation of cyber defense – the application of reinforcement learning. Specifically, we characterize the types of attackers and defenders in the sense of Bayesian games and, using reinforcement learning, derive empirical findings about how to best train agents that defend against multiple types of attackers.

Introduction

The use of machine learning in cybersecurity has grown substantially as a way to simultaneously increase the speed and reduce the cost of cyber threat detection and response. One machine learning technique that has seen considerable interest but seemingly little adoption in deployed systems is reinforcement learning. While reinforcement learning is notoriously unstable and can be difficult to evaluate, it has also famously seen tremendous success in cases like AlphaGo (Silver et al. 2016). The translation of cybersecurity problems into a Markov decision process is a key step in applying techniques like reinforcement learning, and frequently overlap with game theoretic framings of these same cybersecurity problems.

Today, security orchestration, automation, and response (SOAR) frameworks are used by cybersecurity practitioners to reduce the human labor associated with responding to cybersecurity incidents. These SOAR systems can orchestrate disparate tools within workflows to get additional data, automate remediation actions, and often have other, diverse capabilities. While some SOAR systems have integrated AI/ML capabilities (Kinyua and Awuah 2021), they often lack true automation in the containment and recovery phases of the Identification, Containment, Eradication, Recovery (ICER) cycle. Indeed, most SOAR systems rely on

expert written `if-then` rules to take limited actions when some event is triggered. Our work aims to use deep reinforcement learning to fill this gap.

One difficulty in many reinforcement learning and game theoretic approaches is the notion of “type” in the sense of Harsanyi (Harsanyi 1967). That is, the games *a priori* fix the parameters of the attacker and defender to specify goals and capabilities. In reality, cyber defenders observe a wide variety of attacker types whose tactics, techniques, and procedures (TTPs) may overlap considerably even if their objectives differ. This work considers two attacker types with substantially divergent objectives: a ransomware attacker who seeks to gain control of 80% of a target network and an advanced persistent threat actor who seeks to access data on a single high-value node. We provide detailed definitions of attacker and defender types below. In this work, we provide an extension of the YAWNING-TITAN (Andrew et al. 2022) reinforcement learning framework to allow specification of different attacker types for independent learning agents. Additionally, we demonstrate that a model trained across adversary types in a self-play setting yields a robust, attacker type agnostic defensive agent. We further demonstrate that even when defenders have seen only a single type of attacker, the learned policies are transferable, albeit suboptimal, against unseen adversary types.

Background

Reinforcement learning (RL) has been widely explored in cybersecurity (Nguyen and Reddi 2021; Adawadkar and Kulkarni 2022) as a way to automate detection and response. If we take a graph view of systems we need to defend – a natural representation for computer networks – the data regarding systems is necessarily high-dimensional, capturing the states of individual machines and their network connections. Reinforcement learning, and deep reinforcement learning in particular, serves to effectively reduce this dimensionality and make learning tractable. Deep RL has seen many applications in cybersecurity, including defense of cyber-physical systems (Feng and Xu 2017); phishing detection (Chatterjee and Namin 2019); and moving target defense (Li and Zeng 2023).

This work grounds our experiments using a partially observable stochastic Bayesian game, building on the stochastic Bayesian game (Albrecht and Ramamoorthy 2013).

In this work, we build upon the YAWNING-TITAN (YT) reinforcement learning framework (Andrew et al. 2022) which uses Proximal Policy Optimization (PPO) (Schulman et al. 2017) as its default reinforcement learning algorithm. This particular algorithm has been adopted in a number of applications, including cybersecurity (Nguyen and Reddi 2021; Adawadkar and Kulkarni 2022; Galinkin, Pountourakis, and Mancoridis 2024) as a powerful on-policy deep RL algorithm. In contrast to standard YT, we use self-play between two independent networks and extend the framework itself to include “multi-type training”. Our implementation of self-play also differs from standard YT by incorporating partial observability and noise (Galinkin et al. 2023) to reflect the presence of false positive alerts commonly observed in practice and more accurately frame the difference between the true state of the environment and the observed space.

In addition to standard PPO, our work also considers a Hierarchical PPO (HiPPO) algorithm (Li et al. 2020). The HiPPO algorithm leverages a high-level and low-level policy network where a higher-level manager algorithm does not take an action in the space, but rather conditions the low-level policy networks. At some fixed interval, the manager receives a new observation and decides which low-level policy to commit to over the next interval. In our case, these low-level policies reflect the “skills” of detecting and responding to our different attacker types.

Methodology

We build upon a partially observable stochastic Bayesian game with noise (Galinkin et al. 2023). In contrast with traditional stochastic Bayesian games (Albrecht and Ramamoorthy 2013) which uses Harsanyi-Bellman ad hoc coordination, the partial observability of the game means that there is presently no standard solution concept. We consider two attacker types in our assessment. These attacker types have the same actions available to them and thus, differ in terms of their objectives.

1. Ransomware: an attacker who receives a reward for controlling 80% or more of the target network
2. APT: an attacker who aims to access information on a single high-value node

We elect to use these attacker types for several reasons. First and foremost, ransomware and APT-style attackers are often considered the highest priority threats for many major businesses. Secondly, these attacker types have distinctly different objectives, making it easy to contrast them against each other. In each game, a number of nodes are created, one of which is a “high-value target”. The ransomware actor is indifferent to whether or not the node they have compromised is that target, and treats each node with an equal value. By contrast, the APT actor considers every node except for the high-value target to have a value of zero, with all reward confined to exfiltrating data from that single node.

In line with prior work (Galinkin, Pountourakis, and Mancoridis 2024), we train our agents in an environment with a shared state space but two separate observation spaces. This approach allows the attacking agent to learn through play

and requires two distinct instances of proximal policy optimization (PPO) (Schulman et al. 2017) – one for the attacker and one for the defender – to train the agents. Since the observation spaces differ and the hidden information is key to our methodology, methods like multi-agent DDPG (Lowe et al. 2017) are not suitable, as these methods will leak hidden information to the other agent.

We train our agents in four different settings:

1. Ransomware: The attacking agent has a ransomware objective throughout training
2. APT: The attacking agent has an APT objective throughout training
3. Alternating: Two attacking agents – one Ransomware and one APT – are instantiated. At each training epoch, one is chosen at random to play against the defender.
4. Hierarchical: Two attacking agents – one Ransomware and one APT – are instantiated. The defender uses a hierarchical PPO model with one high-level and two low-level policies. At each training epoch, an attacking agent is chosen at random to play against the defender.

Environment

The environment is based on a partially observable stochastic Bayesian game with noise (Galinkin et al. 2023) that is intended to reflect realistic conditions. This game setting reflects the fact that neither the attacker nor the defender has full, true knowledge of the full state space, and that the outcomes of their actions are not fully determined – there is a probabilistic outcome associated with their action succeeding or failing. The game is defined as a tuple $\Gamma = (S, A, P, R, \Theta)$ where:

- $S = \langle V, E \rangle$ is the state space
- $A = \{A_{\mathcal{A}}, A_{\mathcal{D}}\}$ is the action space with $A_{\mathcal{A}}, A_{\mathcal{D}}$ representing the attacker’s and defender’s action spaces, respectively
- $P : S \times S \rightarrow [0, 1]$ is the state transition function representing the probability that a compound action $a_t = \langle a_{\mathcal{A}}, a_{\mathcal{D}} \rangle$ in state s at time t will yield some state s' at time $t + 1$
- R the expected immediate reward of taking action a in state s
- $\Theta = \{\Theta_{\mathcal{A}}, \Theta_{\mathcal{D}}\}$ the type spaces for attackers and defenders.

The state space represents a computer network that the defender is tasked with defending and which the attacker seeks to compromise to achieve their own goal. This is modeled as a network graph where each node is a defender-owned computer and each edge is a network connection between two nodes in that graph. Each node v is a tuple $(v_{\square}, v_{\alpha}, v_{\delta})$ that defines the true state and the hidden information:

1. $v_p \in [0, 1]$: The “vulnerability” of a particular node – the probability that an attack will be successful.
2. v_{α} : The true value of whether the node has been compromised by the attacker, visible only to the attacker.
3. v_{δ} : Defender-visible attribute that indicates whether an alert has been triggered on the node.

In our game, we fix the action spaces $A_{\mathcal{A}}, A_{\mathcal{D}}$ for attacker and defenders, subject to the actions available in YAWNING-TITAN (Andrew et al. 2022), such that all attacker and defender types share a relevant action space. $A_{\mathcal{A}}$ is comprised of five actions:

- **Basic Attack:** Compromise and make accessible some adjacent $v \in V$ with probability given by v_v , the “vulnerability” of the particular node.
- **0-day Attack:** Compromise and make accessible some adjacent $v \in V$ even if $v_v = 0$.
- **Move:** Move from some compromised $v \in V$ to another accessible $v' \in V$
- **Do Nothing:** Take no action
- **Execute:** End the game and realize rewards for all compromised $v \in V$

$A_{\mathcal{D}}$ is comprised of seven actions

- **Reduce Vulnerability:** For some $v \in V$, slightly decrease the probability, p that a basic attack will be successful
- **Make Node Safe:** For some $v \in V$, reduce the probability that a basic attack will be successful to 0.01
- **Restore Node:** For some $v \in V$, reset the node to its initial, uncompromised state
- **Scan:** With some probability, detect the true compromised status of each $v \in V$
- **Isolate:** For some $v \in V$, remove all $e \in E$ connected to it
- **Reconnect:** For some $v \in V$, restore all $e \in E$ that have been disconnected
- **Do Nothing:** Take no action

The costs and rewards that specify $R_{\mathcal{A}}, R_{\mathcal{D}}$ are defined by the player’s type, detailed below. Since our attackers and defenders leverage reinforcement learning, the actual function computing R is learned by the “critic” value function of the agent.

Attacker Type Definition

Using the partially observable stochastic Bayesian game as the basis of our analysis, we must consider the type, in the sense of Harsanyi (Harsanyi 1967, 1968), of our attacker and defender. In this setting, the type, $\theta_{\mathcal{A}}$ of an attacker is drawn from the set of all types $\Theta_{\mathcal{A}}$. This type is uniquely defined by the objective of the attacker, characterized by their reward function.

In cybersecurity, threat actor attribution is frequently used and forms a natural analogy with types. However, attack attribution is a notoriously difficult practice (Perry, Shapira, and Puzis 2019) and is one reason why so much of threat intelligence relies on private companies selling this information to defenders. There are many models for performing attribution of threat activity to a particular group, the most prominent of which is the diamond model of intrusion analysis (Caltagirone, Pendergast, and Betz 2013) that contains four core features: adversary, capability, infrastructure, and victim, pictured in Figure 1, and links those features together to discover knowledge of malicious activity.

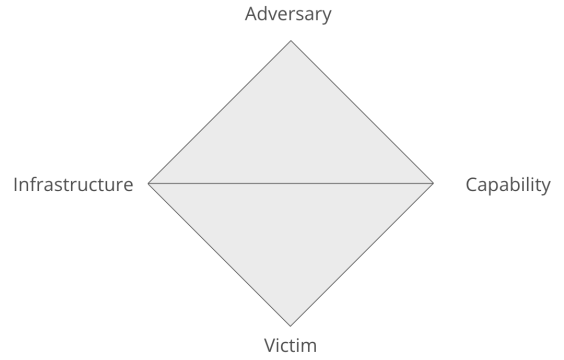


Figure 1: The Diamond Model of intrusion analysis: Adversary, Victim, Capability, and Infrastructure. Edges represent the relationships between features that can be used to analyze and discover malicious activity.

The diamond model also includes meta-features like timestamp, phase, result, direction, methodology, and resources. In essence, threat intelligence asks and aims to answer the question: “who is conducting the attack, how is it being conducted, and what do they want?” A full discourse on the diamond model is beyond the scope of this paper but the outcome of this type of analysis is typically an identification of an intrusion set or threat actor group that has a natural analogy to an attacker’s type.

This attacker type mapping may be extremely granular *e.g.*, identifying activity associated with a particular military unit ala Mandiant’s pioneering report on APT 1 (Intelligence 2013). In many cases, identification is far more coarse – simply identifying that the threat actor group is some kind of advanced persistent threat or cybercrime group. In this work, we take a coarse view of attacker types.

Defender Type Definition

While attacker types have a clear parallel to threat actor groups, defender types have largely remained ill-defined. Our defender type should capture attributes of how a defensive environment is set up – the detection of malicious activity and potential false positives. It should also capture the “maturity” of the organization as it represents their ability to respond to threats. In our setting, we consider a number of parameters that contribute to the optimal strategy of a defensive agent – specifically:

- The probability of detecting an attack p
- The false positive rate of detection q
- The cost c associated with defender actions – a proxy for the organization’s maturity
- The payoff function u associated with preventing or failing to prevent an attack

Since defenders have the same objective – maintaining the confidentiality, integrity, and availability of a system – and will learn a strategy parameterized by the above, we say that a defender’s type $\theta_{\mathcal{D}}$ consists of the above, plus their learned strategy $\pi_{\mathcal{D}}$.

In contrast with attackers, defenders lack a taxonomization and classification. To this end, we suggest characterizing both p and q – which are typically positively correlated – both individually and in relation to each other. In lieu of a proper taxonomy, we offer some example archetypes below.

- **Cautious:** $1.0 > p > 0.7$, $0.3 > q > 0.2$ – An organization generally accepting of false positives. Typifies an organization with a large security budget or some managed security service providers. Note that in general, a false positive rate exceeding 30% is impractical to manage for nearly any organization.
- **Balanced:** $0.7 > p > 0.5$, $0.2 > q > 0.1$ – Organization with a somewhat constrained security budget who cannot afford to respond to large numbers of false positives. Typifies organizations like a medium-sized business.
- **Constrained:** $0.5 > p > 0.3$, $0.1 > q > 0.0$ – Organizations with a limited security budget who can afford only to respond to a limited number of threats. Typifies organizations like K-12 education or small businesses.

Throughout this work, we fix our defender type and assume that we are operating with a balanced defender who has values of $p = 0.6$ and $q = 0.1$, consistent with prior work (Galinkin et al. 2023). That is, we assume the defender has configured their tooling such that they correctly detect 60% of attacker activity but mischaracterize benign activity as malicious 10% of the time. Additionally, we define the defender’s utility $u = r - \sum_t^T c_t$ where r is 0 if the defender fails to prevent the attack and 5000 for successfully eliminating the attacker; T is the total length of the episode; and c_t is the cost of the action taken at timestep t . We note that the reward value for winning is derived by choosing the value of the highest cost action available to the defender (10) and multiplying it by the maximum number of possible timesteps per round (500). Modifying these settings impacts the learned policy and should assume realistic values that may be derived empirically from an organization’s tooling and incident investigations.

Agent Training

Training reinforcement learning agents via self-play is challenging in terms of convergence to a globally optimal policy due to the inherently adversarial nature of such training. To control for potential differences in outcomes related to hyperparameters, we opt to use the same set of hyperparameters across all agents during training. To this end, we experimented with a number of learning rates and training step sizes. Our agents were ultimately trained for 3500 steps using an Adam optimizer with a learning rate of 0.0003 for the actor and 0.0005 for the critic, with an update batch size of 64. For the actor, values of $\{0.00005, 0.0001, 0.0003, 0.0005, 0.001\}$ were tried, and for the critic, values of $\{0.0001, 0.0003, 0.0005, 0.001, 0.0015\}$ were tried, where the critic learning rate was always higher than the actor value for stability. We also experimented with training step sizes of $[1000, 2000, 3500, 5000, 10000]$ and found that rewards were generally stable within 3000 epochs, even

with very low learning rates. No other hyperparameters were modified.

Multi-Type Training

Two of our defender agents, Alternating and Hierarchical, are trained in multi-type scenarios. In these scenarios, two attacker agents are instantiated from scratch and learn according to their objective – either the ransomware or APT objective. During training, one of the two attackers is chosen at random to be the “active” attacker, and the defending agent plays against the active attacker using their current policy. The results are recorded and the agent policy networks are trained over the course of the training run.

In the Alternating case, the defender is a single PPO (Schulman et al. 2017) agent that acts against attackers in the game. This agent follows the same architecture as the standard PPO agent, with a single actor-critic network learning an action and value policy for deciding what action to take given a particular state. By contrast, in the hierarchical case, we leverage HiPPO (Li et al. 2020) and establish a manager network who chooses a subpolicy at some interval k . We consider a “coarse” attacker type: a family of attackers $\Phi_{\mathcal{A}} \subseteq \Theta_{\mathcal{A}}$ which may consist of a “granular” set of individual $\theta_{\mathcal{A}}$. In our case, since $\Theta_{\mathcal{A}}$ consists of only two attacker types, we instantiate only two networks. For each attacker family, we instantiate a subpolicy network π_{sub} . Thus, at every k timesteps of the game, the manager network assesses the subpolicy which is best performing and chooses actions from that policy for the next k steps, before re-evaluating.

Results

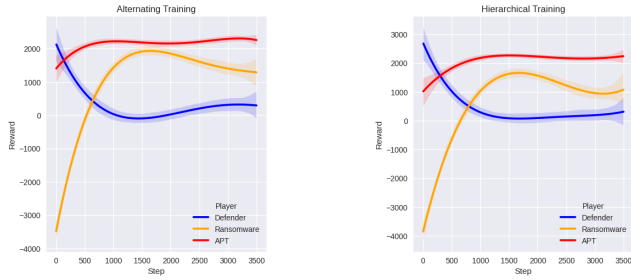
Our evaluation results demonstrate a number of findings:

1. Defenders who see multiple attacker types during training achieve better rewards on average
2. A simple change in the attacker’s condition for victory yields a distinctly different action policy
3. Learned defender policies have transferability to unseen attacker types

Training

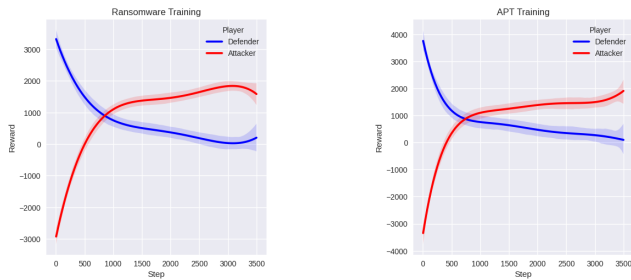
From our training curves in Figure 2, we can observe that in all settings, defensive agents initially start out with high levels of reward and converge over time to near-zero reward. Although our game is not zero-sum, there is an inverse relationship between attacker and defender rewards. In the alternating (Figure 2a) and hierarchical (Figure 2b) settings, we note that APT rewards are fairly consistent across all training runs, suggesting that more defender adaptation is occurring against the ransomware opponent. This is reasonable, considering that the highest scores achieved during training for the defender – and thus, the learned best policy – is likely weighted toward ransomware defense. We also highlight that the combination of training curve smoothing and random attacker assignment overestimates the early returns of the APT model in Figures 2a and 2b. In the case of single-type training, pictured in Figures 2c and 2d, we note that the reward for learned policies are fairly stable from

training step 1500 onward. During the course of training, the alternating model played against the APT attacker 1701 times and the ransomware player 1799 times. The hierarchical model played against APT 1767 times and ransomware 1733 times.



(a) Training time rewards for defender, ransomware, and APT in alternating setting.

(b) Training time rewards for defender, ransomware, and APT in hierarchical setting.



(c) Training time rewards for defender and ransomware attacker

(d) Training time rewards for defender and APT attacker

Figure 2: Training curves for all four training settings. Note that curves are smoothed using a best fit line with order 5.

Evaluation

Evaluating reinforcement learning findings is a challenge, even more so in our case due to the stochasticity inherent in the environment. In line with Agarwal *et al.* (Agarwal *et al.* 2021), we consider score distributions and the interquartile mean for our evaluation runs in addition to the mean scores achieved. Our evaluation environment is a 50 node network with edges randomly instantiated at each run, ensuring at least 60% connectivity between nodes and no unconnected nodes. The difference in scores between Figure 3 and Figure 4 demonstrate the value of examining the interquartile mean. While Figure 3 shows that the APT-optimized defender does, in fact, perform best on the APT attacker, the hierarchical defender has the best overall average score (383.28) across both attacker types. On the other hand, Figure 4 demonstrates the truly poor performance of the APT-optimized defender against ransomware attackers and has the defender trained in the alternating setting achieving the best reward (-326.01) across attacker types, with the hierarchical defender performing only marginally worse (-369.52). In both cases, however, it suggests that the agents

who observed both attacker types in training have a meaningful advantage, despite having seen fewer individual instances of each attacker type than either of the specialized models.

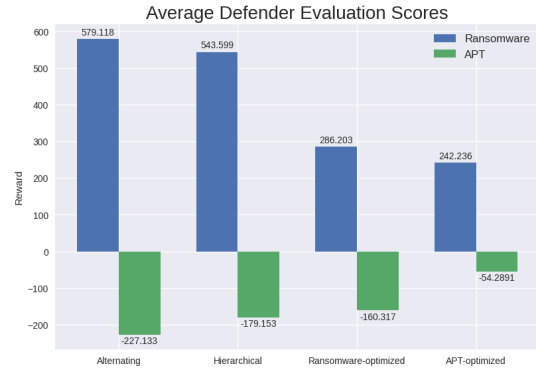


Figure 3: Mean evaluation reward for each defender against ransomware and APT-type attackers

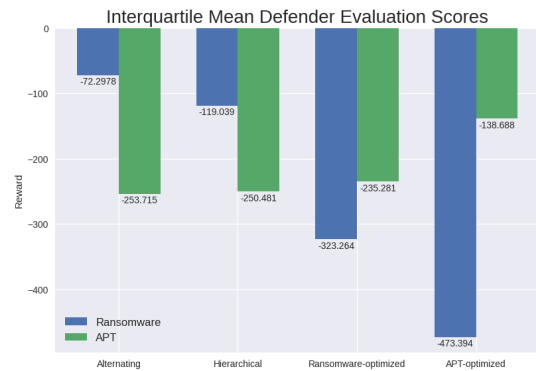


Figure 4: Interquartile mean evaluation reward for each defender against ransomware and APT-type attackers. Note that all rewards in this graph are negative; less negative values are better.

Figure 5 shows the win rates of each defender against both types of attacker. As the negative values in Figures 3 and 4 suggest, the attacker achieves their objective in most scenarios, likely due to the “balanced” type of the defender being fixed, as mentioned in the section describing defender types. In fact, the highest win rate of all, the alternating defender against ransomware-type attackers, is only 31%. Across all defenders, no single defender manages to achieve better than a 4.6% win rate versus APT-type attackers. On average, the hierarchical defender achieves the best win rate (17.6%), closely tailed by the alternating defender (17.25%). We note that these generally disappointing win rates for defenders are, at least in part, due to our assumptions around the detection rate p and false positive rate q , and a higher p or lower q would likely yield more impressive outcomes for the defender.

The score distributions in Figure 6 suggest a number of findings. First, across all defenders, the training curves fol-

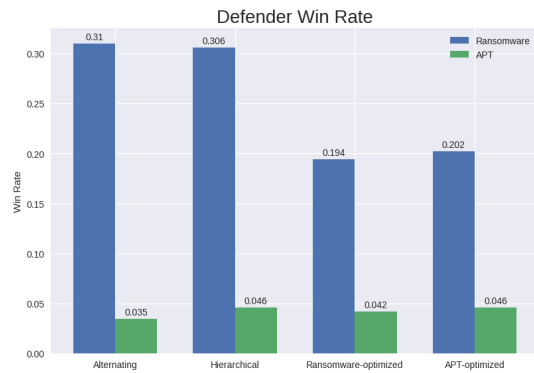


Figure 5: Evaluation win rate for each defender against ransomware and APT-type attackers

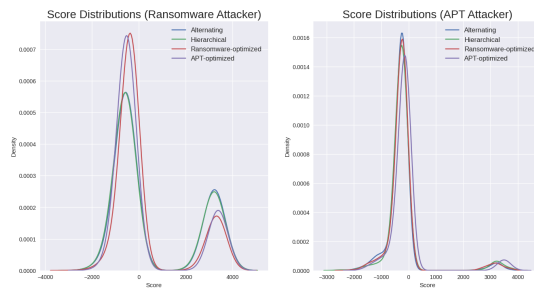


Figure 6: Evaluation score distribution for each defender against ransomware and APT-type attackers. Note the probability density on the y-axis and score values on the x-axis differ between the two charts.

low a very similar shape against both attacker types and have probability density concentrations at or around the same score value. This suggests that the learned policies are fairly similar, though the individual actions and corresponding win rates differ. Another observation is that although there is transferability of a defender’s learned skills – as evidenced by the APT-optimized defender’s performance against ransomware – the attacker’s learned policies are distinct. As described in our methods section, costs are incurred for actions taken, but there is no penalty other than the cost incurred by actions for losing. By comparing the two score distribution charts in Figure 6, we observe that there is substantially more variance in ransomware outcomes than in APT outcomes, and that APT outcomes tend to have a high density of low-negative losses. From this, we can infer that APT type attackers, having a much more directed goal of compromising and exfiltrating data from a particular target, are learning a policy that quickly uncovers the target and compromises it, resulting in shorter games with relatively smaller losses.

Conclusion

In this work, we have demonstrated that defensive agents trained via reinforcement learning self-play are capable of learning strategies to mitigate multiple types of attackers. In particular, we have shown that agents trained in multi-type

settings – our “alternating” and “hierarchical” agents – perform meaningfully better on average than agents trained to specialize against one particular attacker type. We highlight that as additional attacker types are introduced, the hierarchical agent has an advantage for scaling in needing only to learn the high-level policy. Although we leverage HiPPO (Li et al. 2020) in this work, online algorithms for mixtures of experts like exponential weighting (Littlestone and Warmuth 1994) potentially offer a major advantage in quickly adapting to novel threats. We wish to explore the pros and cons of this sort of online learning methodology in future work.

One limitation of this work is that YAWNING-TITAN’s network node states restrict the available attacker and defender action spaces. Although the policies learned by our two attacker types do differ in a meaningful way, this limitation makes it difficult to introduce additional attacker types who may differ in the actual techniques used. In future work, we seek to apply our findings to more fully-featured simulated environments with richer states and actions such that we can emulate a larger number of adversaries and apply our findings in real-world environments.

References

- Adawadkar, A. M. K.; and Kulkarni, N. 2022. Cybersecurity and reinforcement learning—A brief survey. *Engineering Applications of Artificial Intelligence*, 114: 105116.
- Agarwal, R.; Schwarzer, M.; Castro, P. S.; Courville, A. C.; and Bellemare, M. 2021. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34: 29304–29320.
- Albrecht, S. V.; and Ramamoorthy, S. 2013. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 1155–1156.
- Andrew, A.; Spillard, S.; Collyer, J.; and Dhir, N. 2022. Developing Optimal Causal Cyber-Defence Agents via Cyber Security Simulation. In *Workshop on Machine Learning for Cybersecurity (MLACyber)*.
- Caltagirone, S.; Pendergast, A.; and Betz, C. 2013. The diamond model of intrusion analysis. Technical report, Center For Cyber Intelligence Analysis and Threat Research Hanover Md.
- Chatterjee, M.; and Namin, A.-S. 2019. Detecting phishing websites through deep reinforcement learning. In *2019 IEEE 43rd annual computer software and applications conference (COMPSAC)*, volume 2, 227–232. IEEE.
- Feng, M.; and Xu, H. 2017. Deep reinforcement learning based optimal defense for cyber-physical system in presence of unknown cyber-attack. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–8. IEEE.
- Galinkin, E.; Pountourakis, E.; Carter, J.; and Mancoridis, S. 2023. Simulation of Attacker Defender Interaction in a Noisy Security Game. In *AAAI-23 Workshop on Artificial Intelligence for Cyber Security*.
- Galinkin, E.; Pountourakis, E.; and Mancoridis, S. 2024. The Price of Pessimism for Automated Defense. In *Inter-*

national Conference on Decision and Game Theory for Security, 45–64. Springer.

Harsanyi, J. C. 1967. Games with incomplete information played by “Bayesian” players, I–III Part I. The basic model. *Management science*, 14(3): 159–182.

Harsanyi, J. C. 1968. Games with incomplete information played by “Bayesian” players part II. Bayesian equilibrium points. *Management Science*, 14(5): 320–334.

Intelligence, M. 2013. APT1 Exposing One of China’s Cyber Espionage Units. Technical report, Mandiant Intelligence.

Kinyua, J.; and Awuah, L. 2021. AI/ML in Security Orchestration, Automation and Response: Future Research Directions. *Intelligent Automation & Soft Computing*, 28(2).

Li, A.; Florensa, C.; Clavera, I.; and Abbeel, P. 2020. Sub-policy Adaptation for Hierarchical Reinforcement Learning. In *International Conference on Learning Representations*.

Li, H.; and Zeng, Z. 2023. Robust Moving Target Defense Against Unknown Attacks: A Meta-reinforcement Learning Approach. In *Decision and Game Theory for Security: 13th International Conference, GameSec 2022, Pittsburgh, PA, USA, October 26–28, 2022, Proceedings*, volume 13727, 107. Springer Nature.

Littlestone, N.; and Warmuth, M. K. 1994. The weighted majority algorithm. *Information and computation*, 108(2): 212–261.

Lowe, R.; Wu, Y. I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30.

Nguyen, T. T.; and Reddi, V. J. 2021. Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8): 3779–3795.

Perry, L.; Shapira, B.; and Puzis, R. 2019. No-doubt: Attack attribution based on threat intelligence reports. In *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 80–85. IEEE.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.