MIS Quarterly

RESEARCH ARTICLE

RADAR: A FRAMEWORK FOR DEVELOPING ADVERSARIALLY ROBUST CYBER DEFENSE AI AGENTS WITH DEEP REINFORCEMENT LEARNING[1]

Reza Ebrahimi
School of Information Systems, University of South Florida
Tampa, FL, U.S.A. {ebrahimim@usf.edu}

Yidong Chai
School of Management, Hefei University of Technology, Hefei, CHINA, and
Department of Information Systems, City University of Hong Kong, Hong Kong, CHINA {yidongchai@gmail.com}

Weifeng Li
Department of Management Information Systems
University of Georgia, Athens, GA, U.S.A. {weifeng.li@uga.edu}

Jason Pacheco
Department of Computer Science, University of Arizona
Tucson, AZ, U.S.A. {pachecoj@cs.arizona.edu}

Hsinchun Chen
Eller College of Management, University of Arizona
Tucson, AZ, U.S.A. {hsinchun@arizona.edu}

Artificial intelligence (AI) is being widely adopted in modern cyber defense to weave automation and scalability into the operational fabric of cybersecurity firms. Today, AI aids in crucial cyber defense tasks such as malware and intrusion detection to keep information technology (IT) infrastructure secure. Despite their value, cyber defense AI agents can be vulnerable to adversarial attacks. In these attacks, the adversary deliberately manipulates a malicious input by taking a sequence of actions so that a targeted cyber defense AI agent fails to correctly determine its maliciousness. Consequently, the robustness of cyber defense AI agents has raised deep concerns in modern cyber defense. Drawing on the computational design science paradigm, we couple robust optimization and reinforcement learning theories to develop a novel framework, called reinforcement learning-based adversarial attack robustness (RADAR), to increase the robustness of cyber defense AI agents against adversarial attacks. To demonstrate practical utility, we instantiate RADAR for malware attacks—the primary cause of financial loss in cyber attacks. We rigorously evaluate the performance of RADAR as a situated IT artifact against state-of-the-art machine learning and deep learning-based benchmark methods. Incorporating RADAR in three renowned malware detectors shows an adversarial robustness increase of up to seven times, on average. We conclude by discussing contributions to information system research as well as implications for cyber defense stakeholders.

Keywords: Cyber defense, artificial intelligence, adversarial attacks, robust optimization, deep reinforcement learning, computational design science

[1] Gediminas Adomavicius was the accepting senior editor for this paper. Heng Xu served as the associate editor. Yidong Chai is the corresponding author.

Introduction

With the recent increase in the scale and severity of cyber attacks, artificial intelligence (AI) agents have been developed to automate cyber defense in information technology (IT) infrastructure (Apruzzese et al., 2019; Rai, 2017). Cyber defense AI agents have demonstrated the ability to effectively detect and remediate threats at an unprecedented scale (Tolido et al., 2019). New AI agents, including deep neural networks, have shown promise in rapidly detecting unseen malware files and network intrusions (Goodfellow et al., 2018). For instance, AI-based malware detectors can detect unseen mobile Android malware with 96% precision without expensive dynamic malware analysis (Narayanan et al., 2018). Today, leading cybersecurity and IT firms are weaving AI-enabled cyber defense into their operational fabric to protect their IT infrastructure with automated malware detection, intrusion detection, and spam filtering. For example, Avast and Symantec, two major cybersecurity firms, benefit from deep neural networks in their proprietary antivirus products. Endgame uses gradient boosting trees in its open-source malware detector (Anderson et al., 2018; Bloomberg, 2018; Song et al., 2022). A large-scale international survey of IT firms revealed that 69% of them believe they cannot accomplish cyber defense without AI agents (Tolido et al., 2019).

However, AI agents have been found to be vulnerable to adversarial attacks—adversarial data inputs meticulously modified by an adversary to mislead the AI agent (Yuan et al., 2019). Cyber defense AI agents are no exception (Apruzzese et al., 2019). For instance, a previously known malicious executable could be modified to evade an AI malware detector by first changing the signature section and subsequently resetting the file checksum. As another example, malicious network packets could be modified to evade an intrusion detection system by performing encoding followed by fragmentation. In both cases, an adversary crafts adversarial inputs by taking steps of meticulous and hard-to-notice changes to the original malicious input (Goodfellow et al., 2018; Monteiro et al., 2019). The vulnerability of cyber defense AI agents to adversarial attacks is construed as an emerging threat to autonomous cyber defense (Goosen et al., 2018). While cyber defense AI agents are increasingly relied on to protect IT infrastructure, little is known about strengthening the robustness of these agents against adversarial attacks.

In this study, we aim to strengthen the robustness of cyber defense AI agents by leveraging the theories of robust optimization (RO) and reinforcement learning (RL). RO provides a rigorous framework to incorporate the effect of an adversary in the optimization process of learning an AI agent (Bertsimas et al., 2010). This framework promotes a two-player game between the adversary and the AI agent (Madry et al., 2018). In the cyber defense context, RO introduces an adversary who generates evasive adversarial attacks and trains an AI agent that learns from these attacks to improve its robustness. As such, RO establishes that robust defense requires effective adversarial attack emulation (AAE) (Kolter & Madry, 2018).

As adversarial attacks often involve taking a sequence of actions to modify the malicious input against the cyber defense AI agent (Anderson et al., 2018; Fang et al., 2019), AAE can benefit from modeling the steps taken by the adversary in attacking the cyber defense AI agents. RL specializes in examining the sequential interaction of an actor (e.g., adversary) with the environment (e.g., IT infrastructure equipped with an AI-enabled detector) over discrete time steps (Sutton & Barto, 2018). RL is thus well-suited for emulating the steps (i.e., the sequence of actions) an adversary takes to craft adversarial attacks. Additionally, the sequence of actions captured by RL is useful for gaining insights into the adversary's strategies and further enhancing the cyber defense AI agent (Anderson et al., 2018).

Drawing on the computational design science paradigm, we leverage RO and RL to develop a novel ***R**L-based **ad**versarial **a**ttack **r**obustness* (RADAR) framework for enhancing the robustness of cyber defense AI agents. RADAR discovers effective adversarial attacks and prepares cyber defense AI agents to remediate them. Specifically, RADAR offers a novel RL approach to emulate adversarial attacks and presents a novel RL-based RO formalization to strengthen the robustness of cyber defense AI agents against adversarial attacks. Given that malware attacks are the leading threat to IT infrastructure (Bissell et al., 2019), we instantiate our RADAR framework for adversarial malware attacks, evaluate the effectiveness of RADAR against state-of-the-art adversarial malware emulation and RL benchmark methods, and present the utility of RADAR on malware detector robustification. Our experiments showed that RADAR increased the robustness of evaluated malware detectors by seven times, on average, and provided useful insights into designing more robust AI-based malware detectors. Overall, this study makes three major contributions. First, we propose a framework for strengthening the robustness of cyber defense AI agents against adversarial inputs at a large scale. To our knowledge, this is the first robustification framework modeling the adversarial game between the adversary and the cyber defense AI agent. Second, we develop a novel RL method for emulating sequences of adversarial attack actions taken by adversaries. In addition to outperforming the state-of-the-art adversarial attack emulation, our proposed method provides actionable insights into the vulnerabilities of cyber defense AI agents. Third, we design a novel RL-based RO method that is empirically tested to be effective in strengthening the robustness of cyber defense AI agents. The study provides cyber defense practitioners with practical insights, including the significant improvement of adversarial malware detection rate and the behavior of the repeated game between the adversary and malware detector.

Research Background

Four major areas of research are examined. First, we review the information systems (IS) literature on defending IT infrastructure. Second, we explore the theory of robust optimization (RO) as the overarching framework for improving the robustness of cyber defense AI agents. Third, we review AAE to examine effective approaches to automatically emulating adversarial attacks against IT infrastructure. Lastly, we identify state-of-the-art RL methods for operationalizing AAE in cyber defense. Based on the review, we identify research gaps and formalize our research questions.

Defending IT Infrastructure

Recent IS research on defending IT infrastructure can be classified into three major streams: (1) AI-enabled security research, which focuses on developing new IT artifacts based on AI as guided by computational design science (Abbasi et al., 2010; Rai, 2017), (2) behavioral security research that aims to model organizational/individual behavior, policy compliance, and security risk for defending IT infrastructure (Hui et al., 2018), and (3) economic security research focusing on modeling the impact of investment, market effects, and other economic factors on the security of IT infrastructure (Hui et al., 2018). Table 1 summarizes recent IS studies based on their focus, research stream, AI agent's task (if applicable), and whether they assume an adversarial environment. The AI-enabled security research stream often focuses on leveraging AI to design artifacts that learn from human security analysts and automate selected cybersecurity tasks (Ampel et al., 2024; Benjamin et al., 2016, 2019; Ebrahimi et al., 2020; Li et al., 2016; Samtani et al., 2017; Yin et al., 2019; Ebrahimi et al., 2022). However, these artifacts are often not designed to be deployed in adversarial environments, where the adversary can generate adversarial attacks to mislead the artifact. As a result, the security of these IT artifacts can be compromised, which is detrimental to the effectiveness of modern AI-enabled cyber defense (Tolido et al., 2019).

On the contrary, behavioral and economic security studies have provided theoretical justification for an inherently adversarial environment (Burns et al., 2019; Karhu et al., 2018). These studies have incorporated the adversary's behavior into their modeling of security phenomena. For instance, Burns et al. (2019) adopted this view in studying the impact of insiders' behavior on information security. Similarly, Karhu et al. (2018) considered the adversarial role of hostile firms in protecting software resources from exploitation. However, there is a lack of research on how to strengthen the robustness of cyber defense AI agents to adversarial inputs (Goodfellow et al., 2018). We examine the theory of robust optimization as an overarching methodological framework for strengthening the robustness of AI agents in the presence of adversarial inputs.

Robust Optimization (RO) Theory

Traditional AI agents aim to learn from benign training inputs via minimizing a cost (loss) function without considering the scenario in which an adversary could manipulate the inputs. RO expands the optimization function to account for the adversary, which leads to learning a robust AI agent (Madry et al., 2018). RO offers a rigorous, unifying approach that encompasses solving two optimization problems simultaneously: (1) emulating an adversary that aims to mislead an AI agent by maximizing a cost function (known as inner maximization), and (2) learning an AI agent that reduces incorrect defense decisions by minimizing the cost function (known as outer minimization). Given an input x, the inner maximization finds an adversarial input $x+\delta$ that achieves a high cost for the defender, thus increasing the chance of evading the cyber defense AI agent. Accordingly, RO's objective is formulated as a minimax optimization featuring competition between the adversary and the AI agent:

$$\min_{\phi}\left(\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\Delta}\ell(h_{\phi}(x+\delta),y)\right]\right) \tag{1}$$

In this formulation, h_{ϕ} represents the model associated with the cyber defense AI agent. x and y denote the original input (e.g., a malware file) and class label (e.g., benign vs. malicious), respectively. $\mathcal{D}$ is the sampling distribution, and $\delta \in \Delta$ represents a perturbation selected from the set of permissible perturbations (e.g., modifications to the malware file). While the adversary aims to mislead the agent by maximizing the loss (the inner maximization in Equation 1), the AI agent aims to minimize the expected loss ℓ (the outer minimization in Equation 1). This formulation suggests that strengthening the robustness of the AI agent requires effective emulation of the adversarial inputs (i.e., attacks) (Kolter & Madry, 2018). As such, we next examine nascent methods for emulating adversarial attacks against IT infrastructure.

Table 1. Selected Recent IS Literature on Defending IT Infrastructure

Year	Author(s)	Research focus	Research stream	AI agent's task	Adversarial environment assumption
2024	Ampel et al.	Labeling hacker exploits	AI-enabled	Cyber threat detection	No
2022	Ebrahimi et al.	Multilingual cyber threat detection in the dark web	AI-enabled	Cyber threat detection	No
2020	Ebrahimi et al.	Automated cyber threat detection in the dark web	AI-enabled	Cyber threat detection	No
2020	Sen et al.	The impact of cyber attacks on software market	Economic	NA	No
2020	Yoo et al.	The impact of individual self-efficacy on information security	Behavioral	NA	No
2019	Benjamin et al.	Collecting data for AI-enabled security analytics	AI-enabled	Text mining	No
2019	Burns et al.	The impact of insiders' behavior on information security	Behavioral	NA	Yes
2019	Cram et al.	Examining employees' compliance with information security policies	Behavioral	NA	NA
2019	Liang et al.	How individuals cope with IT security threats	Behavioral	NA	No
2019	Yin et al.	De-anonymizing dark web financial transactions	AI-enabled	Transaction classification	No
2019	Yue et al.	The effect of discussions in hacker forums on DDoS attack victims	Economic	NA	No
2018	Karhu et al.	Securing software resources to prevent exploitation from hostile firms	Behavioral	NA	Yes
2018	Vance et al.	Examining users' habituation to security warnings	Behavioral	NA	No
2017	Angst et al.	Reducing the incidence of security data breaches	Economic	NA	No
2017	Hui et al.	The impact of law enforcement on deterring DDoS attacks	Economic	NA	No
2017	Samtani et al.	Designing a malware source code classification system	AI-enabled	Malware detection	No
2017	Temizkan et al.	The impact of software diversity on network security	Economic	NA	No
2016	Benjamin et al.	A computational framework to identify key participants in conversations among cybercriminals	AI-enabled	Hacker Conversation classification	No
2016	Li et al.	A text mining framework for key seller identification in carding shops	AI-enabled	Cybercriminal classification	No

Adversarial Attack Emulation (AAE)

Adversarial attacks are carried out by strategically taking a number of actions to meticulously modify the original input (e.g., malware) so that the functionalities of the original input are preserved and the defense AI agent (e.g., malware detector) is misled into an erroneous decision (e.g., labeling the malware as benign). Functionality-preserving actions can be *additive* or *editing*. Additive actions append content to the input (e.g., adding bytes to a malware executable) and editing actions change the main body of the input (e.g., renaming code sections in a malware executable). Actions can be sensitive to order as the same set of actions taken in different orders could result in different variants (Anderson et al., 2018). Hence, a successful sequence of actions requires the adversary to interact with the cyber defense AI agent and learn from the responses sequentially. The adversary strategically chooses the next actions to achieve evasion based on the prior responses from the cyber defense AI agent (Fang et al., 2019). Such interactions are conducted using various threat models based on the adversary's knowledge, including white-box attacks, where the defense AI agent's model specification is fully known, black-box (BB) attacks, where the adversary only knows the confidence scores from the AI agent, and binary black-box (BBB) attacks where the adversary only knows the binary decision of the agent (Anderson et al., 2018; Qiu et al., 2019). As the BB and BBB threat models are more realistic in the context of attacking IT infrastructure (Rosenberg et al., 2018), we review the existing AAE research within these two threat models and summarize prior research in Table 2 along seven dimensions: the dataset on which they operate, the defense AI agent, research goal, the adversary's action type, threat model, whether they can model the steps (i.e., the sequence of actions) taken by the adversary, and their AAE method. As shown in Table 2, existing AAE often operates on malware files and network traffic datasets and targets a wide range of state-of-the-art cyber defense AI agents, including deep neural networks and gradient boosting trees (GBT).

Our review of the existing AAE literature identifies several research opportunities. First, while most AAE studies focus on the offensive AAE (attack), only a few studies consider strengthening the defense. The robustness of defense AI agents could be strengthened by emulating adversarial attacks and incorporating these adversarial attacks into the optimization of AI defenders (Apruzzese et al., 2019). Second, most research focuses on the black-box setting where the AI agent's confidence score is available. The binary black-box setting, which is more realistic in attacks against the IT infrastructure, is understudied in the current AAE literature (Rosenberg et al., 2018). Third, real-world adversaries typically use a combination of additive and editing actions when interacting with the cyber defense AI agent; however, several existing AAE methods assume actions to be additive, which could limit the coverage of the emulated adversarial attacks. Fourth, prevalent AAE methods fall into three categories: (1) heuristic-based optimization algorithms (e.g., genetic programming, swarm optimization, and hill climbing) (Dang et al., 2017; Dey et al., 2019; Han et al., 2020); (2) deep generative architectures, including generative adversarial networks (GANs) (Lin et al., 2019; Shahpasand et al., 2019; Usama et al., 2019; Zhao et al., 2021) and recurrent neural networks (RNNs) (Hu & Tan, 2018); and (3) deep RL (Anderson et al., 2018; Fang et al., 2019).

Table 2. Selected Significant Black-box and Binary Black-box AAE against IT Infrastructure

Year	Author(s)	Dataset	Defense AI agent	Research goal	Action type	Threat model	Action sequence	AAE method
2024	Zhong et al.	VirusShare	NN	Attack	Additive	BB	No	GAN
2022	Hu et al.	Malwr	NN	Attack	Additive	BB	No	GAN
2021	Zhao et al.	KDD network traffic	NN	Attack	Editing	BB	No	GAN
2020	Han et al.	Network traffic	NN, LR, SVM	Attack	Editing	BB	No	Particle swarm optimization
2019	Castro et al.	VirusTotal	Signature-based	Attack	Additive	BB	No	Randomized perturbations
2019	Chen et al.	VirusShare, Malwarebench mark	CNN	Attack & defense	Additive	BB	No	Enhanced randomized perturbations
2019	Dey et al.	Contagio PDF malware dump	RF & Rule-based	Attack	Additive	BBB	No	Genetic Programming
2019	Fang et al.	VirusTotal	GBT	Attack	Additive	BBB	Yes	Deep RL
2019	Lin et al.	KDD network traffic	SVM, RF, LR	Attack	Editing	BB	No	GAN
2019	Shahpasand et al.	Drebin Android applications	RF, SVM, NN, LR	Attack	Additive	BB	No	GAN
2019	Suciu et al.	VirusTotal, Reversing Labs, FireEye	CNN	Attack	Additive	BB	No	Append benign features
2019	Usama et al.	KDD network traffic	NN, RF, LR	Attack & defense	Editing	BB	No	GAN
2018	Anderson et al.	VirusTotal	GBT	Attack	Editing	BBB	Yes	Deep RL
2018	Hu & Tan	Malwr dataset	RNN	Attack	Additive	BB	No	Generative RNN
2017	Dang et al.	PDF malware	RF	Attack	Additive	BBB	Yes	Hill Climbing

Note: NN: Neural Network, GPT: Generative Pre-trained Transformer, LR: Logistic regression, SVM: Support Vector Machine, RF: Random Forest, GBT: Gradient Boosted Trees, CNN: Convolutional Neural Network, GAN: Generative Adversarial Network, RNN: Recurrent Neural Network

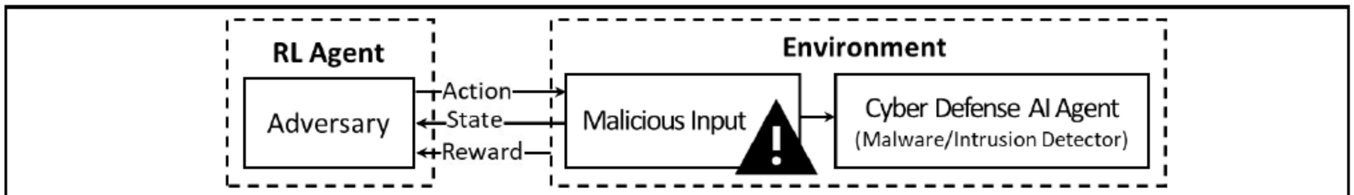

Figure 1. Overview of the Interaction Between the Adversary and Cyber Defense AI Agent in Deep RL Framework to Enable AAE

Neither heuristic-based optimization algorithms nor deep generative architectures are inherently designed to trace and model the adversaries' sequences of actions as the adversary interacts with the cyber defense AI agent over time (Sutton & Barto, 2018). Moreover, GAN-based AAE methods typically require a mapping function projecting the generated representations from the GAN generator to functional adversarial attacks. Such a functionality-preserving mapping function might not always be available for arbitrary contexts, especially when the malware representation used by the defender is unknown (Anderson et al., 2018). As we elaborate in the next subsection, RL presents a natural fit for AAE, as RL is designed to model the adversary's decision-making process in interacting with the cyber defense AI agent and has demonstrated promising results in pioneering AAE research. In particular, RL-based methods optimize the sequence of adversary actions, accommodate both order-sensitive and order-insensitive actions, and preserve the functionality of the original input.

Reinforcement Learning for AAE

RL is a Markov decision process in which an agent interacts with an environment over discrete time steps (Sutton & Barto, 2018). Denoting the environment state space and the agent's action space by $\mathcal{S}$ and $\mathcal{A}$, respectively, and given a state $s_t \in \mathcal{S}$ at time step t, the agent takes an action $a_t \in \mathcal{A}$ based on a policy $\pi(a_t \mid s_t)$ and consequently receives the immediate reward $r(s_t, a_t)$ from the environment. The goal of RL is to learn the optimal policy distribution $\pi(a \mid s)$ that maximizes the return R (expected cumulative discounted reward) over subsequent time steps. Accordingly, RL is particularly suited for modeling the adversary's sequential actions in attacking IT infrastructure, as the adversary iteratively makes attack decisions based on the response from the IT infrastructure.

Learning the policy π is closely tied to estimating the state-action value $Q_\pi(s_t, a_t) = \mathbb{E}_{a\sim\pi}[R_t|s_t, a_t]$ denoting the expected return following policy π. Deep RL has shown breakthrough results in many recent applications by estimating the policy π and the state-action value Q_π with deep neural networks (Silver et al., 2018). In deep RL, an adversary's behavior can be modeled as learning a deep neural network parameterized by θ that estimates policy π. Finding the policy $\pi_\theta(a_t|s_t)$ that maximizes the reward lends itself to solving a stochastic optimization problem for finding parameter θ^* that maximizes the expected cumulative discounted reward across all trajectories:

$$J(\theta^*) = \max_\theta \mathbb{E}_\tau[\textstyle\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)], \quad (2)$$

where $\tau = s_0, a_0, \dots, s_t, a_t$ denotes the trajectory (sequence of state-actions), and $\gamma \in (0,1)$ is a discount factor to ensure the sum is not divergent. Due to the large dimension of states in real-world problems, this optimization often cannot be exactly solved with deterministic optimization techniques such as integer, linear, or dynamic programming (Sutton & Barto, 2018). While these well-established approaches are useful, they often require restrictive assumptions, such as expressing the problem with linear constraints or knowing the environment's characteristics (i.e., transition probabilities of the Markov decision process). RL models the sequential interactions between an RL agent and the environment. The RL framework is well-suited for modeling the adversary's actions in the game between the adversary and the cyber defense AI agent. Figure 1 illustrates an overview of this framework in the context of AAE.

As shown in Figure 1, the RL agent models the adversary who interacts with the environment of an IT infrastructure equipped with a cyber defense AI agent. The adversary applies actions (from a finite discrete action space) on a malicious input to generate an evasive adversarial input (e.g., a malware variant). The environment encompasses the malicious input and the cyber defense AI agent (e.g., malware detector). The state denotes the agent observation reflecting (1) the changes in the malicious input resulting from the adversary's modifications and (2) the internal defense strategy of the cyber defense AI agent. While the former is available to any adversary, the latter is not accessible in black-box settings (Anderson et al., 2018). The state space observed by the RL agent encompasses all possible representations over all adversarial variants. The cyber defense AI agent examines the adversarial input variant and reveals its response to the adversary (benign or malicious). Evasion occurs when an adversarial variant is recognized as benign by the defense AI agent. For a given action and state, a positive reward is given to the adversary upon successful evasion. The adversary strategizes its actions to maximize its cumulative reward.

Numerous deep RL methods have been proposed in the recent literature (Fellows et al., 2019; Fujimoto et al., 2018; Haarnoja et al., 2018; Hasselt et al., 2016; Mnih et al., 2016a; Schulman et al., 2015, 2017; Wang et al., 2017). We present a taxonomy of deep RL methods in Appendix A, featuring three crucial considerations for AAE. These considerations are rooted in real-world AAE domain characteristics: First, for successful AAE, interactions with the AI agent must be minimized to avoid detection by the cyber defense AI agent (Kreuk et al., 2018). As such, deep RL models must be highly sample-efficient (i.e., minimize the number of interactions with the environment). Some recent deep RL methods are focused on high sample efficiency to reduce the number of interactions with the environment. These methods include variational actor-critic (VAC) (Fellows et al., 2019), soft actor-critic (SAC) (Haarnoja et al., 2018), actor-critic with experience replay (ACER) (Wang et al., 2017), and double deep Q-network (DDQN) (Hasselt et al., 2016). These highly sample-efficient deep RL methods are more applicable to the AAE domain. Second, as AAE involves large sequences of byte streams (e.g., file bytes, network traffic), the deep RL method needs to operate on a combinatorially large number of states (Anderson et al., 2018). While this requirement is not met in many recent deep RL methods (including ACER and DDQN), VAC yields breakthrough performance in complex tasks with a large number of states, leading to a 33% higher cumulative reward over SAC in tasks with high-dimensional states (Fellows et al., 2019). Third, the attack vector in AAE is inherently discrete. The action space encompassing possible modifications by the adversary (e.g., editing malware timestamp) is not continuous (e.g., rotating a robotic arm) (Kreuk et al., 2018). Among the highly sample-efficient deep RL methods, ACER, DDQN, and VAC have shown promise in environments with discrete actions. However, ACER and DDQN are not explicitly designed to handle large state spaces encountered in AAE. Extending these methods to handle a high number of states is nontrivial due to the computational complexity of their objective functions (Fellows et al., 2019). VAC's success in high-dimensional state spaces makes it attractive in AAE.

Variational Actor-Critic (VAC)

VAC is a sample-efficient method for high-dimensional state spaces, which operates based on the actor-critic (AC) model (Fujimoto et al., 2018). AC has been shown to provide accurate estimations of the adversary's actions based on the reward feedback from the environment (Fujimoto et al., 2018). This is achieved by directly learning from the gradients of the policy during interacting with the environment, as expressed by the policy gradient theorem (Sutton & Barto, 2018). To solve the objective given in Equation (2), the policy gradient theorem offers a practical way to obtain an unbiased estimate for the gradients of the expected return of state s under policy π_θ, known as value function $V_\theta(s)$ (derivations are given in Appendix C):

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta\,(a_t|s_t) Q_{\pi_\theta}(s_t, a_t)\right], \tag{3}$$

in which $\pi_\theta(a_t|s_t)$ is learned by an *actor* network and $Q_{\pi_\theta}(s_t, a_t)$ is learned by a *critic* network in an AC setting. To conduct this approximation, AC employs two iterative steps, carried out by the *actor* and *critic*:

1. **Policy improvement:** the *actor* estimates a policy π_θ based on a given action-value function Q_{π_θ}. The *actor* is characterized by a neural network, parameterized by θ, that approximates a policy $\pi_\theta(a|s)$, a probability distribution over actions.

2. **Policy evaluation:** the *critic* estimates the action-value function based on the current policy π. The *critic* is characterized by a neural network that accepts a state-action pair and outputs the expected value of state-action pairs to estimate $Q_{\pi_\theta}(s, a)$.

In their seminal work, Fellows et al. (2019) showed that in large state spaces, the approximation is solved more effectively via variational inference (variational AC (VAC)). Variational inference approximates the action posterior $\pi_\theta(a_t|s_t)$ with a tractable family of distributions (Fellows et al., 2019). Despite its usefulness, the standard VAC can benefit from improved gradient estimates when applied to AAE scenarios with a large number of states.

Research Gaps and Questions

Our review of defending IT infrastructure suggests a vital need for cyber defense AI agents that are robust to adversarial attacks. While AAE is an active research area, there lacks a framework that considers both attack and defense to model the game between the adversary and the AI agent. Also, only a few AAE methods comply with a realistic threat model (i.e., binary black box AAE). Furthermore, many AAE methods are mainly additive and thus limited in their coverage of generated attacks. In addition, most AAE methods are limited in their ability to model the adversary's sequential decision-making process in interacting with the cyber defense AI agent. As such, using deep RL to model the sequence of actions and RO to enhance defense could address these gaps. However, state-of-the-art deep RL methods designed for continuous environments need enhancement for discrete actions in cyber attacks. Additionally, it is unclear how RO can be extended to incorporate effective RL-crafted attacks. Given these gaps, the following research questions are posed: (1) How can the RO theory be leveraged in cybersecurity to strengthen the robustness of cyber defense AI agents? (2) How can effective adversarial attacks be emulated with deep RL to establish RO? (3) What mitigation strategies can be learned from utilizing RO and RL to guide the design of more robust cyber defense AI agents?

Proposed Design

To address these research questions, we propose a novel framework, RADAR, that optimizes the robustness of cyber defense AI agents. RADAR operates under three realistic assumptions based on the literature and feedback from three malware analysts in the National Cyber-Forensics and Training Alliance (NCFTA):

1. The adversary's goal is to modify a malicious input into adversarial variants capable of evading the defense AI. To this end, the adversary can apply a sequence of additive or editing actions on a malicious input and observe the state of the input (e.g., characteristics of a malware file).

2. The adversary has no access to the internal information of the defense AI agent, including model parameters, architecture, confidence score, and any pre-trained surrogate model of the defense AI agent. The only information available to the adversary from the defense AI agent is a strictly binary response denoting whether the input is detected as malicious or not.

3. The defense AI agent can only observe the adversarial inputs generated by the adversary and does not have access to the adversary's internal information (i.e., model parameters and architecture).

Under these assumptions, RADAR consists of three inputs, two main phases, and two tangible outputs from each phase. Figure 2 provides an overview of the RADAR framework, including its inputs, phases, and task(s) in each phase; the method to operationalize each phase; and outputs. RADAR accepts adversarial attack vectors, a vulnerable defense AI agent, and malicious input data as its inputs. The adversarial attack vectors determine a set of permissible modifications (i.e., actions) that operate on the malicious input data to mislead a vulnerable cyber defense AI agent. While the adversarial attack vectors are specific to each application domain, reconnaissance sources to identify them may be common. Major reconnaissance sources useful for identifying adversarial attack vectors include current attack literature, domain experts, and cybersecurity reports (presented in Appendix B1).

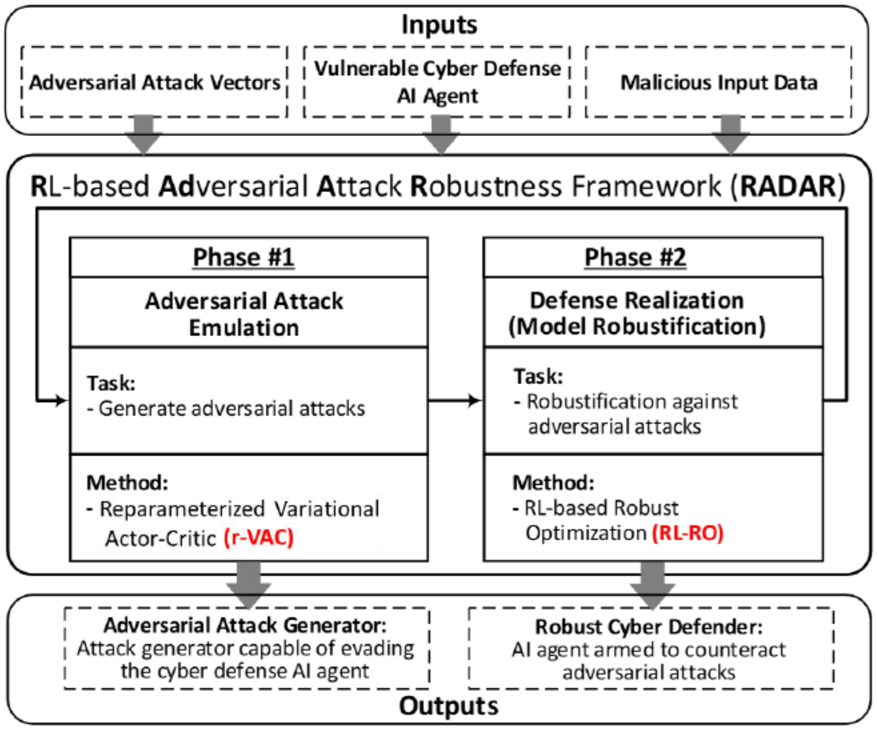
Inputs
Adversarial Attack Vectors
Vulnerable Cyber Defense AI Agent
Malicious Input Data
RL-based Adversarial Attack Robustness Framework (RADAR)
Phase #1
Adversarial Attack Emulation
Task:
- Generate adversarial attacks
Method:
- Reparameterized Variational Actor-Critic (r-VAC)
Phase #2
Defense Realization (Model Robustification)
Task:
- Robustification against adversarial attacks
Method:
- RL-based Robust Optimization (RL-RO)
Adversarial Attack Generator:
Attack generator capable of evading the cyber defense AI agent
Robust Cyber Defender:
AI agent armed to counteract adversarial attacks
Outputs

Figure 2. Overview of the RADAR Framework

Concrete examples of adversarial attack vectors in the malware domain are given in the instantiation of RADAR. Determining these adversarial actions is a prerequisite for conducting AAE in Phase 1. The vulnerable defense AI agent denotes the cyber defense model that is to be robustified against adversarial attacks (e.g., malware or intrusion detection models). In most cybersecurity applications, the malicious input denotes a set of domain-specific sequence data (e.g., malware byte sequence, malicious network traffic) serving as seeds for modification and the construction of adversarial attacks.

Phase 1 is centered around adversarial attack realization. The focal task in this phase is generating adversarial attacks via AAE. At the heart of Phase 1 stands a novel RL-based method called reparametrized variational actor-critic (r-VAC). Phase 1 outputs an adversarial attack generator policy capable of misleading (i.e., evading) the cyber defense AI agent. Phase 2 focuses on defense realization for the mitigation of generated attacks. Accordingly, the primary task in this phase is robustifying the AI agent against adversarial attacks. We propose a novel method called RL-based robust optimization (RL-RO) to operationalize this phase. Phase 2 yields a robust cyber defense AI agent model that counteracts adversarial attacks. These two phases form an iterative sequential process: The learned *stochastic* policy in Phase 1 can be used multiple times to generate different adversarial variants. That is, the defense AI agent could detect some observed variants as malicious while missing others generated under the same policy. Once enough generated samples from that policy are observed in Phase 2, the policy is updated, and the game between the attacker and defender continues. We next detail each phase of RADAR and instantiate the framework for the adversarial malware attack domain.

Phase 1: Adversarial Attack Emulation

Phase 1 aims to emulate the adversary based on the identified adversarial actions. Accordingly, Phase 1 outputs an adversarial attack generator that mimics the adversary's actions to evade a cyber defense AI agent h_ϕ, parametrized by ϕ. In Phase 1, the adversarial attack generator $\Pi_{\theta,h_\phi}(x)$, receives a malicious input x, which is detectable by h_ϕ and yields adversarial input $x+\delta^*$, which is undetected by the cyber defense AI agent, where δ^* represents the optimal modification that leads to the evasion. As the attack generator is represented by the policy, with a slight abuse of notation, we refer to its parameters by θ.

Attack model problem specification: We model the adversarial modification of malicious inputs for evading detection as a Markov decision process with the following components:

Each *state* $s \in \mathcal{S}$ is defined by the representation of a malicious input. The state space $\mathcal{S}$ is defined as the set of all feasible malicious inputs. An *action* $a \in \mathcal{A}$ corresponds to performing an adversarial modification to the malicious input (e.g., malware, network traffic) from action space $\mathcal{A}$. In the context of cyberattacks, the action space is assumed to be finite and discrete. Moreover, the action needs to preserve the integrity of the modified input (e.g., the modified malware variants need to maintain the malicious functionality). The *policy* $\pi(a|s)$ is a function that determines the candidate action to maximize expected future rewards, given the current state. Emulating adversarial attacks is tantamount to learning a policy distribution over a set of actions. The *reward* $r(s,a)$ depends on whether the adversarially modified malicious input can evade the defense model while maintaining functionality. A fixed positive value is awarded when a functionality-preserving evasion occurs. A cumulative reward captures the potential of future evasion from the defense model. The adversary strategizes its actions (on the current and other malicious inputs) to maximize its cumulative reward by creating evasive variants.

In this specification, the adversary performs an action to modify a malicious input. The AI-enabled defense model examines the adversarial input and signals a positive reward to the adversary upon evasion. The reward informs the adversary to strategize its actions to maximize its cumulative reward by generating evasive adversarial attacks. As noted in the RL review, VAC is a viable approach to mimic the adversary's actions. To further improve the variance of gradient estimations in VAC, we propose r-VAC (reparametrized VAC), which reparametrizes actions via approximate sampling, yielding a more effective attack generator (i.e., actor network) in discrete action spaces for cyber defense. The actor network in VAC performs policy improvement, which translates to estimating $\pi_\theta(a|s)$, a distribution over actions, given the states. Learning this action distribution involves gradient estimations over $\pi_\theta(a|s)$, as presented in Equation (3). Jang et al. (2017) and Maddison et al. (2017) showed that improved gradient estimations for discrete random variables (as for actions in AAE) can be achieved by reparameterizing them via approximate sampling. Inspired by their seminal work, we propose reparameterizing the discrete actions in attack generation by sampling from a differentiable Concrete distribution over attack vectors. Concrete distribution is a differentiable distribution that allows sampling from discrete variables in neural networks (Jang et al., 2017). We next detail approximate sampling and its integration with VAC.

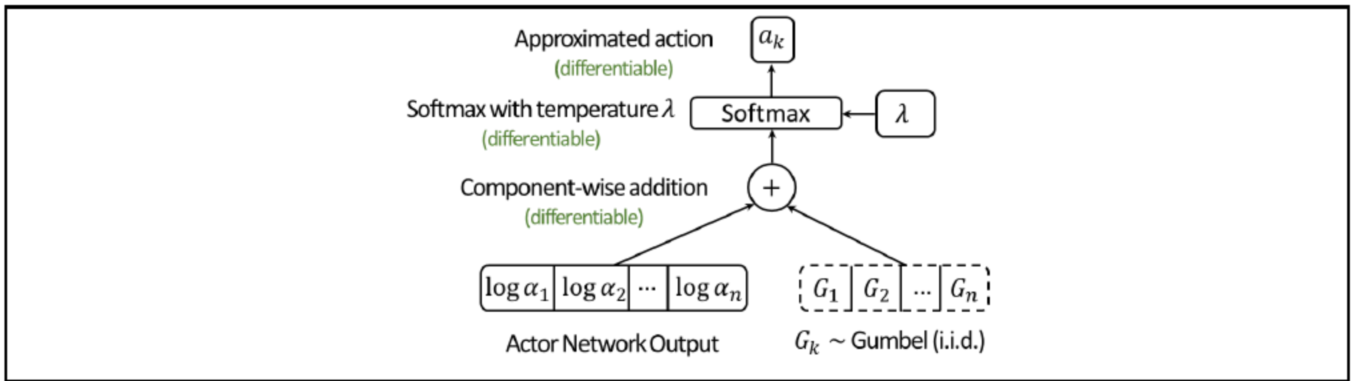

Figure 3. Sampling Graph of Concrete Distribution for Action Distribution $a \sim Concrete(\alpha, \lambda)$ with n Discrete Actions (modified based on Maddison et al., 2017)

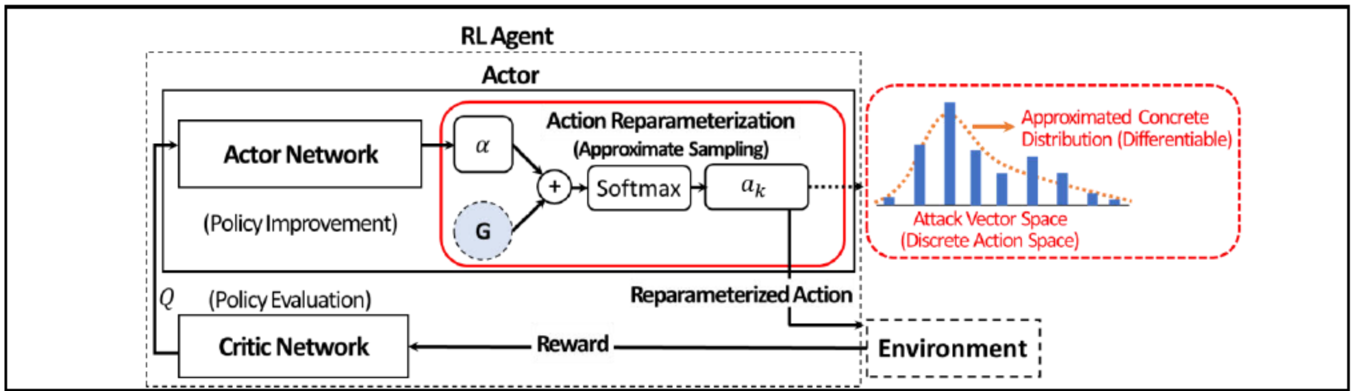

Figure 4. Our Proposed r-VAC for Learning Policies on Discrete Action Spaces by Extending VAC via Approximate Sampling

Approximate Sampling of Actions with Concrete Distribution (Action Reparameterization)

Let α_k denote the probability assigned to each action in the policy by the actor network. Approximate sampling with Concrete distribution reparameterizes the stochastic discrete action variable as a differentiable function of G, a random variable identifying a Gumbel distribution (see the last part in Appendix C). The sampling includes the steps shown in Figure 3. First (bottom left corner of Figure 3), the probability of the sampled discrete action from the actor network is projected to a logarithmic scale for numerical stability (Maddison et al., 2017).

Second (bottom right corner of Figure 3), independent and identically distributed (i.i.d.) random variables are sampled from the Gumbel distribution. Third, the samples from the Gumbel distribution (Jang et al., 2017) are combined with the discrete action probabilities through component-wise addition and a softmax operator parameterized by $\lambda \in (0, \infty)$, which is referred to as "temperature," inspired by thermodynamics (middle of Figure 3). Following this process leads to a Concrete distribution over the outputs of the actor network, which is expressed via the following equation (Maddison et al., 2017):

$$a_k = \frac{\exp((\log \alpha_k + G_k)/\lambda)}{\sum_{i=1}^{n} \exp((\log \alpha_i + G_i)/\lambda)}, \tag{4}$$

where the obtained sample a_k is differentiable and can be used for learning the actor network via gradient ascent. Intuitively, as $\lambda \to 0$, samples from the Concrete distribution become closer to the ones from a categorical (discrete) distribution but the variance of the gradient increases (Maddison et al., 2017). In our experiments, moderate values (e.g., $\lambda = 0.5$) performed well. Figure 4 shows an abstract view of r-VAC, extending VAC by approximate sampling for reparameterizing a discrete action space to obtain better gradient estimates. As shown, the discrete samples are approximated by a differentiable Concrete distribution over the attack vectors (right dashed box). The novelty in the design of r-VAC lies in its ability to effectively perform gradient estimations in environments with discrete action spaces achieved by reparametrizing the action space via approximate sampling. The obtained policy network from r-VAC is an effective adversarial attack generator that yields high-quality attack vectors. This attack generator is useful for creating robust models in Phase 2.

Phase 2: Defense Realization with RO

Phase 2 seeks to automatically improve the defender's robustness using the adversarial attacks generated in Phase 1. To this end, we draw on empirical robust optimization theory to propose RL-based RO (RL-RO), which leverages the RL-enabled attack generator from Phase 1 to improve the defender's robustness. The RL-RO process initially receives a cyber defense AI agent that is vulnerable to adversarial inputs and yields a robust defense AI agent. RO minimizes the AI agent's loss while competing against an adversary that aims to mislead the agent by maximizing the loss. RO's minimax optimization requires the gradients of the inner maximization given in Equation (1): $\nabla_\phi \max_{\delta\in\Delta} \ell(h_\phi(x+\delta), y)$. Based on Danskin's theorem (Danskin, 2012), computing the gradient of a function containing a max term requires finding the maximum point $\delta^*(x)$ first, and then evaluating the gradient at this point: $\nabla_\phi \max_{\delta\in\Delta} \ell(h_\phi(x+\delta^*(x)), y)$, where $\delta^*(x)$ is the optimal modification that maximizes the loss of the cyber defense AI agent. However, finding the exact theoretical maximum is difficult due to irregularities of the loss surface. Kolter and Madry (2018) show that the maximum can be approximated by incorporating a suboptimal attack into the RO's inner maximization. We hypothesize that such relaxations could be useful in obtaining a significantly robustified AI agent. Thus, we approximate the inner maximization by the adversarial attack generator $\Pi_{\theta,h_\phi}(x)$ from the resultant policy network in r-VAC from RADAR's Phase 1. $\Pi_{\theta,h_\phi}(x)$ approximates the optimal adversarial attack sample ($x+\delta^*(x) \cong \Pi_{\theta,h_\phi}(x)$). That is, for a given input sample x, $\max_{\delta\in\Delta} \ell(h_\phi(x+\delta), y) \cong \ell\left(h_\phi\left(\Pi_{\theta,h_\phi}(x)\right), y\right)$, giving rise to RL-based RO:

$$\min_\phi \left(\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell\left(h_\phi\left(\Pi_{\theta,h_\phi}(x)\right), y\right)\right]\right), \tag{5}$$

where $\Pi_{\theta,h_\phi}(\cdot)$ denotes a suboptimal adversarial attack model against the AI agent model h_ϕ. The outer minimization can be solved by stochastic gradient descent (Kolter & Madry, 2018). The outer minimization procedure is given in Appendix D. While a global exact solution to this minimax optimization could be impractical in practice (Arjovsky et al., 2017), suboptimal useful solutions could be empirically found in practice by alternating between the minimization and maximization steps. We show that this approach leads to significantly robustified AI agents in the Evaluation section below.

Instantiation of RADAR for Malware Attacks

Malware attacks have continued to be the leading cause of financial loss and damage to IT infrastructure over the last few years (Bissell et al., 2019). The cost of malware attacks per IT firm has increased by 11% to an average of US$2.6M annually (Bissell et al., 2019). Following the computational design science paradigm (Gregor & Hevner, 2013), we present the instantiation of RADAR as a situated implementation of our design for the malware attack domain. Consistent with (Peffers et al., 2007), our instantiation was rigorously evaluated through a series of quantitative benchmark experiments. We note that the provided instantiation serves as a proof of concept and proof of value for the proposed RADAR framework and is not meant to provide the best solution to the adversarial malware example generation problem. We next introduce our malware research testbed and the instantiation of RADAR's phases for this testbed.

Adversarial Malware Attack Vectors

Adversarial malware attack vectors are designed to yield polymorphic variants that preserve the malicious functionality of original malware executables without requiring access to malware code. Based on the extant malware analysis literature and consultation with two malware analysts, we identified 10 functionality-preserving adversarial actions for the malware attack. Functionality assessment is detailed in Appendix B2. Figure 5 presents these actions, their descriptions (left), and their corresponding point of effect in the malware executable (right). The actions include *remove signature* (RS: unlink the digital signature from the certificate table in the header), *change timestamp* (CT), *break checksum* (BC: setting the file's checksum to an arbitrary value), *remove debug* (RD: unlink the debug section from the file header), *section rename* (SR), *add import* (AI: add a library or function to the import table), *add section* (AS: add a new section to existing sections), *overlay append* (OA: inject bytes into the file), *compression* (CO: compress file), and *decompression* (DC: decompress a packed binary file). Consistent with Anderson et al. (2018) and Fang et al. (2019), for actions such as CT, SR, and AI that accept an operand (e.g., the name of the library to import for AI or the section name to add for AS), the operands for each action were selected from predefined lists. Actions can be repeated multiple times in one sequence. To maintain the Markovian property, no domain-specific limitations are enforced on the order of actions.

Adversarial Malware Attack Vectors

Action ID	Action Name	Description	Action Type
RS	Remove Signature	Unlinks digital signature from certificate table in the header.	Additive
CT	Change Timestamp	Changes Time Date Stamp (TDS).	Editing
BC	Break Checksum	Sets the file's checksum.	Editing
RD	Remove Debug	Unlinks debug section from certificate table in the header.	Additive
SR	Section Rename	Alters sections' name in malware binary.	Editing
AI	Add Import	Adds an imported library and function to the import table.	Additive
AS	Add Section	Adds a new section to existing sections.	Additive
OA	Overlay Append	Appends Bytes to the end of PE file.	Additive
CO	Compression	Compress the binary file.	Editing
DC	Decompression	Decompress a packed binary file.	Editing

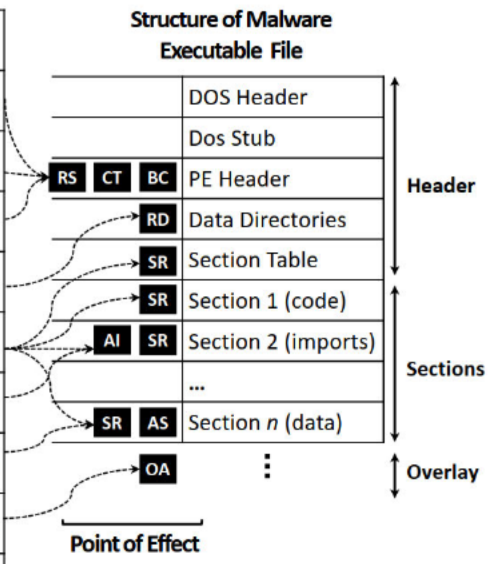

Figure 5. Functionality-Preserving Attack Vectors on Malware Executable File (Left) and their Corresponding Point of Effect (Right).

The provided actions are not meant to be exhaustive, and without lack of generality, can be extended to form long sequences and include more complicated attacks. Iterative application of these actions forms a sequence resulting in a polymorphic, functionality-preserving variant of the original malware executable that evades a targeted malware detector. For example, CT→SR→CO represents an action sequence that changes the file's timestamp (CT), followed by renaming a section (SR) and then compressing the entire file (CO). These actions affect certain parts of the malware executable (right side of Figure 5). As shown on the right side of Figure 5, a malware executable consists of three major parts: *file header, sections,* and *overlay*. The file header includes the metadata (timestamp, file name, data addresses) required to run the executable. The sections include the actual executable content (i.e., code). The overlay denotes the space at the end of the file where the content is not executed. These actions were implemented using LIEF, a Python library that modifies executables at the byte level.

Adversarial Malware Attack Emulation: Phase 1

Generate adversarial malware attacks: The instantiation of Phase 1 involved developing RL-based malware attacks against malware detectors. To this end, we formalized adversarial malware attacks against malware detectors under the RL framework. This formalization entailed characterizing the malware state, adversarial actions, policy, and reward as follows: As the internal state of the detector in the environment was not visible to the adversary, consistent with Anderson and Roth (2018), we used a feature vector to represent the malware state space. To this end, the state of the malware executable was constructed via extracting a holistic set of features represented in a 2,350-dimensional vector extracted from malware files. These features included header metadata, section metadata (section name, size, and characteristics), import and export libraries metadata, counts of human-readable strings (such as file paths, registry keys, and URLs), byte histogram (256 values denoting the counts of each byte value in the executable), and 2D byte-entropy histogram (Saxe & Berlin, 2015). The adversarial actions were selected from a discrete set of functionality-preserving modifications of malware executables shown in Figure 5. The policy was denoted by a neural net determining the next adversarial action based on a given malware state, as noted. The policy's repetitive execution yields an adversarial action sequence required to generate an evasive malware variant (e.g., CT→SR→SR→AI→CO). Consistent with Anderson et al. (2018), an immediate reward $R \in \{0,10\}$ is signaled to the agent after every step, where 0 denotes detection and +10 denotes evasion while preserving original functionality. Under the black-box attack model assumption, the confidence score of the malware detector is not accessible. As such, the reward magnitude denoting the evasion in RL-based AAE is often a fixed positive value (Anderson et al., 2018).

Table 3. AI-Based Malware Detector Used for Evaluating Performance of RADAR in Adversarial Attack Emulation

Detector model	Developer	Architecture	Malware detector type	Feature extraction mechanism	Suitable adversary
LGBM (Anderson & Roth, 2018)	Endgame Inc.	Gradient Boosting Trees	ML-based	Engineered features from file metadata	Editing
MalConv (Raff et al., 2018)	Laboratory of Physical Sciences	CNN variant trained on 1 million malware samples	DL-based	Automatically extract from whole malware binary	Additive
NonNeg (Fleshman et al., 2019)	Laboratory of Physical Sciences	CNN variant with enforcing gradients to be positive	DL- based	Automatically extract from whole malware binary	Additive

Note: ML: machine learning; DL: deep learning

Malware environment: Consistent with Figure 1, the malware environment encompasses the modified malware variant and the malware detector. Aligned with realistic adversarial malware attacks, the environment does not require accessing the source code of the malware executable. The RL environment in our experiments was built on top of the OpenAI Gym RL library and enhanced the "malware-env" environment developed by Anderson et al. (2018). To ensure the same episode length across all RL models, in each episode of the game, the agent can apply up to a maximum of five adversarial actions to evade the malware detector. The agent was given at most three chances to modify a specific malware file to evade the detector. Non-evasive action sequences yielded a zero reward. Consistent with (Anderson et al., 2018; Fellows et al., 2019), ε-greedy exploration strategy was adopted to maintain the balance between exploration and exploitation, and all RL training algorithms were executed on a budget of 10,000 time steps. To construct the environment, we identified three renowned AI-based malware detectors based on their recency and performance in the cybersecurity community. These detectors were also used in Endgame's Malware Evasion Competition in 2019. Since proprietary commercial tools do not provide the opportunity to enhance the detector model, these three detectors are selected to be highly cited open-source tools. As detailed in Table 3, the light gradient boosting method (LGBM) developed by Endgame Inc. uses boosted decision trees and gradient-based sampling (Ke et al., 2017) to achieve efficient and accurate malware detection (Anderson & Roth, 2018). MalConv is a premier open-source deep learning-based malware detector developed by the Laboratory of Physical Sciences, which uses a large convolutional neural network (CNN) trained on over a million malware executables.

NonNeg is a variant of MalConv, developed by the Laboratory of Physical Sciences, that restricts the error gradients to be positive to increase the robustness against adversarial attacks (Anderson & Roth, 2018). The LGBM detector leverages 2,350 features mainly extracted from the metadata section of the malware file and thus is less vulnerable to adding contents to the file overlay (Anderson & Roth, 2018). In contrast, deep learning-based detectors (i.e., MalConv and NonNeg) automatically extract features from the raw content of malware executables. They are hence expected to be more susceptible to additive adversarial actions.

Malware Defense Realization: Phase 2

Robustification against adversarial malware attacks: The instantiation of Phase 2 involved the robustification of LGBM, MalConv, and NonNeg detectors. The original pretrained models for these three malware detectors are available as part of their corresponding open-source packages. These trained models were robustified by conducting the robustification process detailed earlier in RADAR's Phase 2. The robustification algorithm is given in Appendix D.

RADAR's Technical Novelty

The technical novelty of our proposed framework is twofold. First, we propose r-VAC that equips VAC with more effective gradient estimations for AAE. This is achieved by approximating the policy with a differentiable Concrete distribution over discrete attack vectors. Second, we propose RL-RO, which incorporates adversarial inputs generated by r-VAC into RO's minimax optimization. We used the policy network from r-VAC to approximate optimal adversarial attacks to address the inner maximization.

Evaluation

Malware Testbed

Based on the prior malware analysis literature and consulting three malware analysts, we selected VirusTotal as one of the most reputable repositories of emerging malware that aggregates information from multiple contributors worldwide

(Kyadige et al., 2020). To construct our testbed, we obtained an academic license from VirusTotal that provided access to approximately 33.2 GB of recent, working malware executables (28,038 files) collected from 2017 to 2019. Following Raff et al. (2018), to obtain a benign executable dataset, we collected an additional 13,554 (5.04 GB) benign executables from a clean installation of Microsoft Windows. The malware testbed served as the initial seed for generating adversarial attacks and evaluating evasion rates. The benign dataset was used for FPR evaluations. Our malware testbed included six types of malware executables that frequently damage IT infrastructure: adware, botnet, ransomware, rootkit, spyware, and virus. The description of each malware type and the breakdown for each type are given in Table 4.

Consistent with practical guides in Allix et al. (2015) and Pendlebury et al. (2019), to develop robustification against future attacks using the knowledge obtained in the past (rather than future knowledge), malware executables from 2017 and 2018 were used for training, validation, and testing the attack model in RADAR's Phase 1 and malware executables from 2019 were used for the evaluation in Phase 2. As the data was collected from VirusTotal, the distribution of the malware files aligns with their prevalence in the real world. Accordingly, adware is the most common type due to its potential financial gains (Kujawa et al., 2020). The large size of rootkits is due to their common usage for unauthorized access, serving as a fundamental tool in other attack types (Kujawa et al., 2020). However, malware types of relatively smaller sizes, such as ransomware, are among the most detrimental (Bissell et al., 2019). Consistent with RADAR's overview in Figure 2, we first identified the adversarial malware attack vectors.

Experiment Design

To systematically evaluate RADAR and its practical utility in the malware attack domain, we conducted three experiments and a cybersecurity case study, as summarized in Table 5.

Table 4. Malware Samples in the Testbed

Malware type	Description	RADAR's Phase 1 evaluation	RADAR's Phase 2 evaluation
Adware	Shows unwanted ads and forces internet traffic to advertisement sites	2,647 (3.1 GB)	5,714 (14.6 GB)
Botnet	A network of bots connected through the internet	941 (0.47 GB)	2,827 (1.9 GB)
Ransomware	Encrypts data, restricting access until decrypted by adversary	1310 (1.2 GB)	2,118 (3.3 GB)
Rootkit	Grants admin privilege to malware author	1265 (0.94 GB)	4,127 (3.4 GB)
Spyware	Allows malware authors to steal personal information covertly	968 (0.60 GB)	2,203 (1.3 GB)
Virus	Corrupts files on the host system	972 (0.4 GB)	2,946 (2.0 GB)
Total	-	8,103	19,935

Table 5. Experiment Design

Experiment category	Experiment	Description	RADAR phase(s)	Evaluation metric
Performance evaluation	Experiment 1	Compare the performance of r-VAC with state-of-the-art attack methods	Phase 1	Evasion rate
Sensitivity analysis	Experiment 1.1	Measure sensitivity to the reward magnitude		
Performance evaluation	Experiment 2	Measure the reduction in evasion rate before and after robustification with RL-RO in a single game	Phase 2	Evasion rate, false positive rate
Repeated game analysis	Experiment 3	Show latter-round defense strategies are robust against both earlier-round and latter-round attacks in a repeated game	Phase 1 & Phase 2	Evasion rate, false positive rate

Table 6. Overview of Selected Benchmark Methods

Benchmark category	Benchmark approach	Benchmark method name	Reference(s)
Non-RL	Baselines	Random actions Benign feature append (BFA)	Anderson et al., 2018; Chen et al., 2019; Suciu et al., 2019
	Surrogate-based	Surrogate-based RNN	Hu & Tan, 2018
	Meta-heuristic	EvadeHC, GAMMA	Dang et al., 2017, Demetrio et al., 2021
RL-based	Baseline	Policy gradient	Sutton & Barto, 2018
	Q-Learning	Double deep Q-network (DDQN)	Fang et al., 2019; Anderson et al., 2018; Hasselt et al., 2016
		Rainbow	Hessel et al., 2018
	Multi-ARM bandit	MAB-malware	Song et al., 2022
	Actor-critic	Actor-critic with experience replay (ACER)	Anderson et al., 2018; Wang et al., 2017
		Asynchronous advantage actor-critic (A3C)	Mnih et al., 2016b

To rigorously compare RADAR to other alternatives, Experiment 1 compared RADAR's performance on adversarial attack realization to eight leading black-box adversarial malware attack methods from malware attack literature against three identified malware detectors. These methods were selected to represent both non-RL (Surrogate-based, meta-heuristic) and RL-based methods (actor-critic and Q-Learning), as summarized in Table 6. Within the non-RL category, we included two common baseline attacks: random actions (RA) and benign feature append (BFA). Consistent with Anderson et al. (2018), at each step of the sequence, RA selects a random action with uniform probability without learning any policy. Consistent with Suciu et al. (2019), BFA is another widely adopted non-RL baseline method that often performs better than random modifications. BFA injects portions of benign files into the malware executable to generate benign-looking samples. EvadeHC represents a well-established metaheuristic approach that uses hill climbing to operate in black-box settings. GAMMA is a meta-heuristic alternative designed based on genetic programming (Demetrio et al., 2021). EvadeHC's and GAMMA's GitHub implementations were adopted for comparison. Surrogate RNN uses a generative surrogate RNN architecture to operate in black-box settings. Since the authors did not provide the implementation, we carefully implemented the generative RNN based on the specifications mentioned in the paper.

In the RL-based category, policy gradient (Sutton & Barto, 2018) was selected as the baseline for RL-based adversarial attack models. Double deep Q-network (DDQN) is a well-established RL-based binary black-box method that leverages two neural networks for better estimation of the action-value function Q. ACER leads to an effective RL-based binary black-box method that yields the state of the art in malware and other domains by applying variance reduction techniques to reduce the instability of learned policies in the core AC model (Anderson et al., 2018). Rainbow (Hessel et al., 2018) features a more holistic version of DQN that aggregates several advancements in a modern Q-learning RL to improve performance. A3C (Mnih et al., 2016b) features a powerful actor-critic model called asynchronous advantage actor-critic that gained significant popularity for breakthrough performance in Atari games. For RL methods, the implementation in the Intel Lab's Coach RL environment was used. The best parameters for all methods were selected based on the performance on the validation set. Each experiment with optimal parameters was repeated five times, and the average performance was reported (Demšar, 2006). To qualitatively examine the benefits of r-VAC's explainability in AAE, consistent with Anderson et al. (2018), we further analyzed the constituent adversarial actions generated by the learned policy in r-VAC. To this end, we obtained the most frequent adversarial actions leading to the generation of evasive malware variants against LGBM, MalConv, and NonNeg malware detectors. We show how such insights can help mitigate attacks in malware detector design. Given the crucial role of the reward in RL, Experiment 1.1 measured RADAR's sensitivity to the magnitude of reward via performing sensitivity analysis across three selected malware detectors.

To evaluate the performance in defense realization (RADAR's Phase 2) on all three malware detectors, we conducted a series of experiments in which RL-RO was employed on the policy network learned from Phase 1 to improve the robustness of the malware model. Subsequently, the robustified detector was applied to unseen recent malware executables from VirusTotal to assess the effect of robustification on malware detectors. It is helpful to empirically analyze the repeated game set up by the proposed robust optimization between the attack model and the defense AI agent to identify when to stop the robustification process. Since theoretical proof of equilibrium is challenging due to the lack of a closed-form solution for Phases 1 and 2, in

Experiment 3, we assessed whether the latter-round defense models are robust against all earlier-round and current attack models. Lastly, to assess the utility of RADAR in practice, we conducted a real-world case study to demonstrate RADAR's capability of mitigating attack effectiveness in malware detector design.

Evaluation Metrics

We adopted two common robustness metrics in the adversarial malware analytics literature: evasion rate (ER) and false positive rate (FPR). Evasion rate is a widely used metric for measuring the performance of adversarial attack models (Song et al., 2022; Anderson et al., 2018). When all generated adversarial examples are functional, the evasion rate of an AAE method against a given malware detector is defined as $ER = |E|/N$, where E denotes the set of modified malware variants capable of evading the malware detector and N denotes the total number of malware samples generated by the AAE method. A higher evasion rate indicates a stronger attack capable of evading the cyber defense AI agent (e.g., malware detector). FPR is defined as the ratio of benign samples that are falsely detected as malware by the malware detector (Fleshman et al., 2019). Additionally, since FPR plays a vital role in the usability of real-world malware detectors, it is crucial to ensure that while the robustification process reduces the evasion rate, it does not noticeably increase the FPR of the original vulnerable model.

Evaluation Results

Experiment 1: Evaluating RADAR in Adversarial Attack Emulation

We systematically compared r-VAC's performance with the specified benchmark methods against the identified three malware detectors and across six malware categories. Table 7 summarizes the benchmark evaluation results in terms of the evasion rate. Higher evasion rates denote more effective attacks. The last column reports the performance on all malware types combined, which aligns with realistic malware detection scenarios. The results in Table 7 present three main findings. First, RL-based methods yielded more effective adversarial malware variants than other alternatives, including hill climbing-based EvadeHC and surrogate-based RNN methods. This highlights the effectiveness of RL in AAE applications. Second, r-VAC considerably outperformed all the benchmark methods across all six malware categories and against all three malware detectors. The highest performance in each malware category is shown in bold font. On average, r-VAC led to a 19% performance increase against LGBM, 9% against MalConv, and 10% against NonNeg. Third, r-VAC outperformed all benchmark methods on the combined dataset against all three malware detectors (last column in Table 7). The highest performances are underlined and in bold font (28.13% for LGBM, 22.99% for MalConv, and 18.98% for NonNeg). Compared with best-performing methods (ACER and A3C), r-VAC leads to approximately 8% performance gain against LGBM (28.13 vs. 20.67), 5% against MalConv (22.99 vs. 17.58), and 4% against the robustified NonNeg (18.98 vs. 14.43).

These results suggest that enhancing VAC via our proposed r-VAC yields considerably stronger adversarial attacks capable of evading malware detectors (the primary goal of AAE) despite the sparsity of the reward in the malware domain. Furthermore, outperforming other alternatives that can operate in discrete adversarial action spaces (such as DDQN and ACER) indicates that the ability of r-VAC in large malware state spaces is instrumental in achieving these results. To further show generalizability, Appendix B3 provides the results with 10 maximum allowed actions. In addition to evaluating evasive performance, examining evasive emulated attacks is crucial to providing insights into design mitigation strategies. To this end, we identified the dominant adversarial action types (editing vs. additive) that led to successful evasion, along with the top three most evasive action sequences generated by r-VAC in Phase 1 of RADAR against LGBM, MalConv, and NonNeg (Table 8). The second column in Table 8 lists the two most common adversarial actions contributing to evading each malware detector. The third column lists the top three most evasive attacks (i.e., action sequences) for each malware detector. Editing adversarial actions are underlined, whereas additive adversarial actions are not. For LGBM, the most frequent adversarial actions (compression and section rename) were both editing adversarial actions. Compression and section rename were involved in 16% and 14% of the evasive sequences, respectively. For MalConv and NonNeg, while compression was still the most dominant adversarial action in evasive attacks (39% and 36%, respectively), the second most frequent actions were additive (12% import append for MalConv and 10% Remove Debug for NonNeg).

Overall, several observations can be made. First, compression affected all three malware detectors as the most effective adversarial action in emulating attacks. Second, as expected, LGBM was more vulnerable to editing adversarial actions. Additive adversarial actions were not effective in evading LGBM, which is mainly based on features extracted from the metadata that cannot be modified by additive adversarial actions. Third, unlike LGBM, both MalConv and NonNeg were more vulnerable to additive adversarial actions derived from the whole malware executable via deep learning. These findings also provide empirical evidence that the order of actions matters and can contribute to better risk mitigation in modern malware detector design.

Table 7. Evasion Rate Comparisons Against Three Renowned AI-based Malware Detectors Across Six Malware Categories

Malware detector	Model	Adware	Botnet	Ransomware	Rootkit	Spyware	Virus	Combined (all types)
LGBM	Random actions	0.69	1.01	1.65	0.89	1.47	1.98	1.49
	BFA	1.07	1.52	2.44	1.37	2.34	3.03	2.01
	EvadeHC	10.47	5.89	22.66	4.24	12.18	30.50	6.81
	Surrogate RNN	6.47	23.95	13.33	11.49	9.37	15.78	6.29
	PG	5.14	2.28	36.66	10.12	5.62	21.24	6.59
	DDQN	16.64	23.00	44.33**	39.12**	19.8	27.77	9.21
	RainbowDQN	13.87	25.09**	36.33	21.20	13.30	41.27	9.16
	MAB-malware	9.07	4.21	11.25	4.89	10.98	24.13	10.00
	ACER	14.79	30.99	60.11**	27.51	26.87	62.82*	18.50
	A3C	19.72	36.50*	53.11**	25.17	19.37	41.27**	20.67
	r-VAC	**23.42**	**48.29**	**65.22**	**61.15**	**29.53**	**82.40**	**28.13**
MalConv	Random actions	0.72	1.04	0.87	0.83	2.02	1.44	0.96
	BFA	0.92	1.90	1.22	0.96	3.28	2.28	1.67
	EvadeHC	5.44	8.55	2.77	12.31	3.28	14.56	2.10
	Surrogate RNN	7.19	9.12	11.22	6.01	21.09	36.57	10.80
	PG	2.67	5.51	2.33	7.11	2.81	3.18	6.88
	DDQN	6.99	6.08	11.89	16.83	27.5	30.50	14.80
	RainbowDQN	9.14	5.51	17.33	31.05	6.71	26.10	10.17
	GAMMA	3.98	6.13	6.39	3.50	5.34	4.95	12.63
	MAB-malware	21.39	34.59	19.71	30.50	33.33	35.18	17.50
	A3C	6.26	23.38	21.11	23.93	18.75	42.94	15.17
	ACER	17.57	37.07	25.33	29.41	56.09	44.76	17.58
	r-VAC	**28.30**	**44.68**	**26.89**	**50.48**	**65.31**	**48.41**	**22.99**
NonNeg	Random actions	0.59	1.15	1.50	0.86	1.24	1.13	0.88
	BFA	0.77	1.33	2.22	1.09	2.03	1.82	1.57
	Surrogate RNN	6.36	11.97	10.22	12.03	15.00	27.31	1.85
	PG	2.97	10.07	4.22	4.92	10.93	3.64	4.16
	DDQN	12.17	8.37	14.56	29.69	15.46	12.90	9.49
	RainbowDQN	4.93	10.64	11.33	15.59	15.62	19.72	6.47
	GAMMA	15.60	5.20	9.26	7.31	10.08	26.11	11.0
	MAB-malware	12.53	14.37	17.34	18.87	14.93	31.56	14.37
	A3C	11.71	23.19	17.66	15.59	9.38	22.76	14.12
	ACER	8.89	30.99	22.22	24.90	20.63	37.94	14.43
	r-VAC	**15.97**	**48.67**	**26.89**	**46.24**	**31.30**	**41.58**	**18.98**

Note: Numbers in bold indicate the best results. Methods with zero evasion rate across all malware categories were excluded.

Table 8. Dominant Adversarial Actions in Adversarial Malware Attacks with r-VAC in RADAR's Phase 1

Evaded AI agent	Dominant action types in evasive attacks and their frequency	Most evasive action sequences
LGBM	Editing: • Compression: 16% • Section rename: 14%	• Change timestamp → Append import → Change timestamp → Compression • Append import → Break checksum → Section rename → Section rename • Remove signature → Overlay append → Compression → Section rename
MalConv	Editing: • Compression: 39% Additive: • - Import append: 12%	• Compression → Remove debug → Overlay append → Overlay append → Section rename • Remove debug → Remove signature → Import append → Compression • Append import → Overlay append → Change timestamp
NonNeg	Editing: • Compression: 36% Additive: • Remove debug: 10%	• Overlay append → Remove signature → Break checksum → Section rename → Compression • Overlay append → Remove rebug → Break checksum → Compression • Remove debug → Change timestamp → Overlay append → Change timestamp → Overlay append

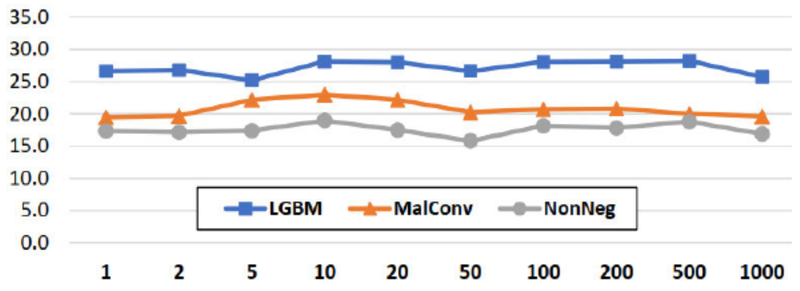

Figure 6. Performance Sensitivity of RADAR to the Reward Magnitude

Table 9. Reduction of Evasion Rate After Robust Optimization Across Three AI-Enabled Malware Detectors

Malware type	Evasion rate (LGBM)		Evasion rate (MalConv)		Evasion rate (NonNeg)	
	Before RL-RO	After RL-RO	Before RL-RO	After RL-RO	Before RL-RO	After RL-RO
Adware	22.4	**8.50**	25.91	**1.76**	26.12	**2.73**
Botnet	15.7	**2.31**	29.81	**8.08***	28.52	**2.87**
Ransomware	13.4	**3.66**	25.88	**3.20**	26.33	**2.41**
Rootkit	18.9	**2.37**	32.46	**4.35**	20.83	**2.1**
Spyware	12.4	**4.01**	27.95	**1.26**	37.01	**3.52**
Virus	17.00	**4.15**	26.35	**6.51**	28.50	**1.81**
Combined	16.63	**4.17 (FPR: 3.9)**	28.06	**4.19 (FPR: 5.0)**	27.89	**2.57 (FPR: 4.8)**

Note: Numbers in bold indicate the best results.

Experiment 1.1: Sensitivity of RADAR to the Reward Magnitude

To ensure that the attack policy learned by RADAR is not drastically affected by the magnitude of reward, we performed a series of sensitivity analysis experiments across all three malware detectors (LGBM, MalConv, and NonNeg). Based on the common reward magnitudes in the RL literature (Anderson et al., 2018), we empirically assessed the evasion rate with a series of reward magnitudes ranging from 1 to 1,000. As it is impractical to test all possible reward values in this range, we systematically selected 10 evaluation points with almost doubling intervals to cover this range efficiently. We repeated each experiment five times for each reward magnitude. Figure 6 plots the average evasion rate of RADAR across malware detectors.

The evasion rates reported in Figure 6 do not provide any evidence in favor of lower vs. higher rewards in this range. Accordingly, to ensure fair comparisons, following Anderson et al. (2018), we chose the reward magnitude of 10 to conduct all RL-based benchmark methods.

Experiment 2: Evaluating RADAR in Defense Realization

We implemented our RL-RO design on the three identified malware detectors (LGBM, MalConv, and NonNeg) to enhance their robustness against adversarial attacks. The evasion rate was then measured on the unseen malware executables for each malware detector. Table 9 compares the evasion rate before and after our robustification with RL-RO across the three identified malware detectors. The results suggest that after adopting RL-RO, all three malware detectors were substantially more robust against unseen malware attacks. RL-RO reduced the average evasion rate by 4, 7, and 11 times for LGBM, MalConv, and NonNeg, respectively. Accordingly, on average, RL-RO decreased the evasion rates across all three malware detectors by approximately 84%, translating into a sevenfold increase in robustness. To ensure that the robustification process does not negatively affect the detectors' performance on correctly recognizing benign samples, we measured the performance of all robustified models on the benign executables obtained from a clean Windows installation (Anderson & Roth, 2018). While the FPR of all robustified models on benign executables stayed the same (3.9%, 5%, and 4.8% for LGBM, MalConv, and NonNeg), the evasion rate was significantly decreased, showing a considerable increase in robustness against adversarial malware attacks.

To further demonstrate the practical utility of RADAR in mitigating adversarial attacks, we later present a cybersecurity case study using Endgame's malware detector, LGBM.

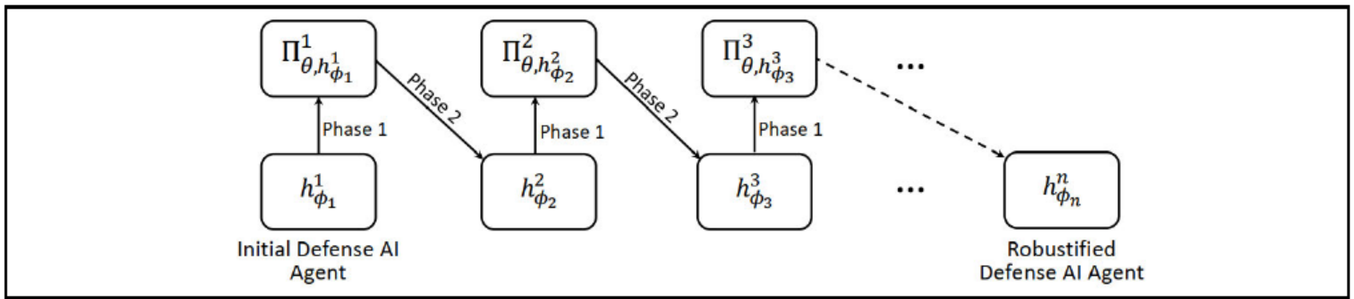

Figure 7. Repeated Game Between the Attack Model (Π) and Defense AI Agent (h) in RADAR

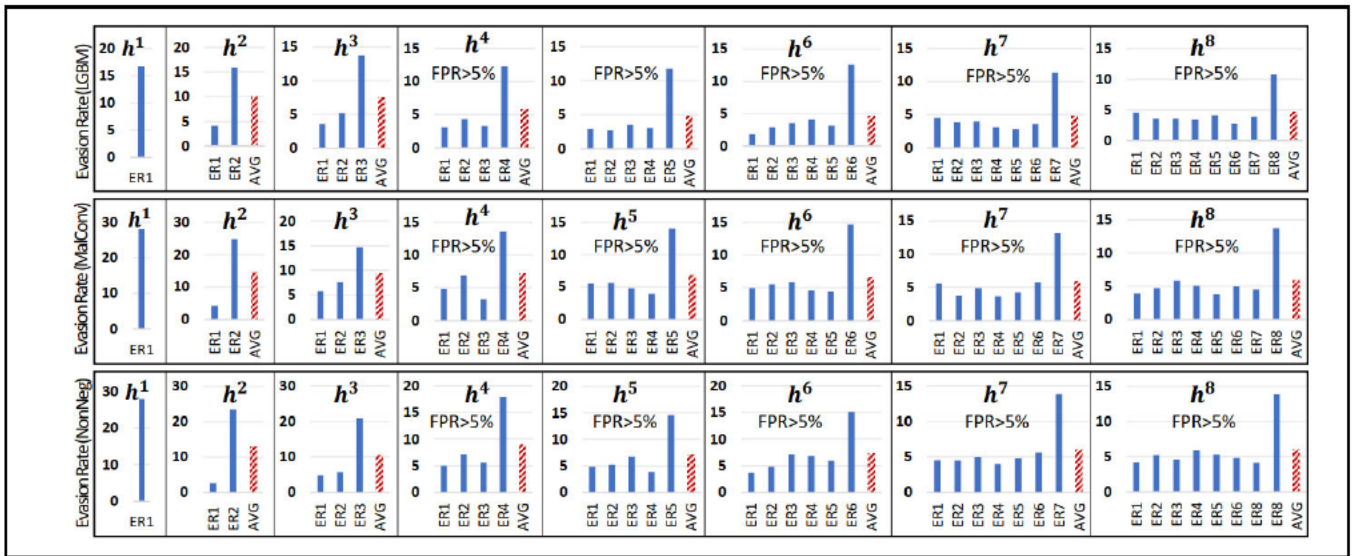

Figure 8. Adversarial Robustness of Defense AI Agents at Different Stages of the Repeated Game

Experiment 3: Empirical Analysis of the Repeated Game in RADAR

To empirically analyze the robustification in a repeated game setup and identify the stopping point, we alternated between Phase 1 and Phase 2 to extend the proposed robust optimization stages between the attack model and defense AI agent beyond the single game. At each stage of the game, an attack policy is learned to evade the current version of the defense AI agent (inner maximization in Equation 4). Subsequently, a robustified defense AI agent is obtained that is robust to the current attack policy (outer minimization in Equation 4). This iterative process continues until the final robustified defense AI agent is achieved (Figure 7).

For each stage, we report the evasion rate of the current attack model (Π_θ^i) against the current defense AI agent ($h_{\phi_i}^i$), evasion rates of previous attack models against the current defense AI agent, and their average (Figure 8). For brevity, the evasion rate of each attack model is denoted by an index (e.g., ER2 denotes the evasion rate of Π_θ^2). To establish a trade-off between the adversarial robustness and false alarm rate (FPR), we monitored the FPR during robustification and stopped the process when FPR started to increase above 5% (Fleshman et al., 2019).

Stopping the game is crucial to ensuring that increasing the adversarial robustness does not compromise the practical usefulness of the malware detector by increasing the false alarm rate (Fleshman et al., 2019). As shown in Figure 8, the original evasion rates of LGBM, MalConv, and NonNeg were 16.63%, 28.06%, and 27.89%, respectively. The average evasion rate for each robustified defense AI agent (denoted by diagonal stripes) monotonically decreased from 10.04%, 14.45%, and 13.01% at the second stage (for h^2) to 4.61%, 5.82%, and 6% at the last stage (for h^8), respectively, and stabilized. This decreasing trend suggests that, on average, latter-round defense AI agents are robust against the previous attack models from former iterations. While this decreasing trend in evasion rate could indicate convergence towards an equilibrium state in which no unilateral change in either attacker or defender would result in an improvement, the FPR starts to constantly increase beyond 5% for all subsequent stages of the robustification from h^4 through h^8. Accordingly, to ensure that robustification does not result in a significant increase of FPR in practice, it is suggested to stop the robustification at h^3. This experiment provides empirical evidence that while it is not practical to discover all adversarial examples in one shot, at each iteration of the repeated game, new adversarial examples are discovered that were not known to the robustified defender in the previous iteration.

Case Study: Mitigating Adversarial Attacks in Malware Detector Design

To demonstrate the practical utility of RADAR in mitigating attack effectiveness, we conducted a case study that focuses on pinpointing the vulnerabilities of a specific malware detector to provide insights into improving malware detector design. Given its popularity in the AI-based malware analysis community, we focused on enhancing LGBM's robustness in this case study. We used RADAR to pinpoint the adversarial actions to which LGBM is most vulnerable. To this end, we leveraged the action sequences that were generated by RADAR and resulted in malware variants that successfully evaded LGBM. Table 10 summarizes the most frequent evasive action sequences of lengths 2 and 3 in each malware category. Two key observations can be made from the most frequent evasive action sequences in Table 10. First, in almost 50% of evasive action sequences, compression appeared as the last adversarial action (holds for botnet, ransomware, rootkit, spyware, and virus). Action sequences ending with compression are shown in bold font.

Second, adversarial actions such as section rename and overlay append that can be applied several times and are applicable to multiple locations of the malware executable were used more often than other adversarial actions. As a result, 75% of the most evasive action sequences in Table 10 included either section rename or overlay append (held for adware, botnet, ransomware, rootkit, spyware, and virus). These action sequences are underlined. Interestingly, these observations could be explained in accordance with human hacker behavior. The first observation suggests applying modification to the file first and obfuscating the content via compression as the last step before deployment. The second observation suggests prioritizing adversarial actions that are repeatable and applicable to multiple parts of malware executables over those that are specific to certain parts. Visualizing the effectiveness of evasive adversarial actions via a heatmap diagram can be useful in gaining insight into the vulnerabilities of the malware detector (Song et al., 2022). Accordingly, Figure 9 visualizes the frequency of evasive adversarial actions within each malware category via a heatmap diagram. The darkness of the cells is proportional to the number of times the action appeared in the action sequences. While the darker cells (associated with compression and section rename) are concentrated on the left side, the right side includes lighter cells (associated with overlay append).

Figure 9 suggests that LGBM was most vulnerable to compression and section rename (marked with the number 1) and least susceptible to byte injections at the end of the file (i.e., overlay append) (marked with the number 2). This can lead to a practical design guideline for training LGBM detectors: to increase the robustness of LGBM against adversarial actions, such as compression and section rename, it could be useful to augment LGBM's training set with the compressed malware files. Furthermore, it could be helpful to add noise to section names in the training set to make the model agnostic to subtle changes in the section names.

Table 10. Most Frequent Evasive Adversarial Action Sequence in Each Malware

Malware type	**Most frequent successful action sequences by length**	
	Length = 2	**Length = 3**
Adware	Change timestamp → Append import	Overlay append → Overlay append → Section rename
Botnet	Remove debug → Section rename	**Overlay append → Remove signature → Compression**
Ransomware	**Overlay append → Compression**	**Remove debug → Remove signature → Compression**
Rootkit	**Remove debug → Compression**	Remove debug → Overlay append → Break checksum
Spyware	Remove signature → Section rename	**Overlay append → Remove debug → Compression**
Virus	**Overlay append → Compression**	Overlay append → Overlay append → Section rename

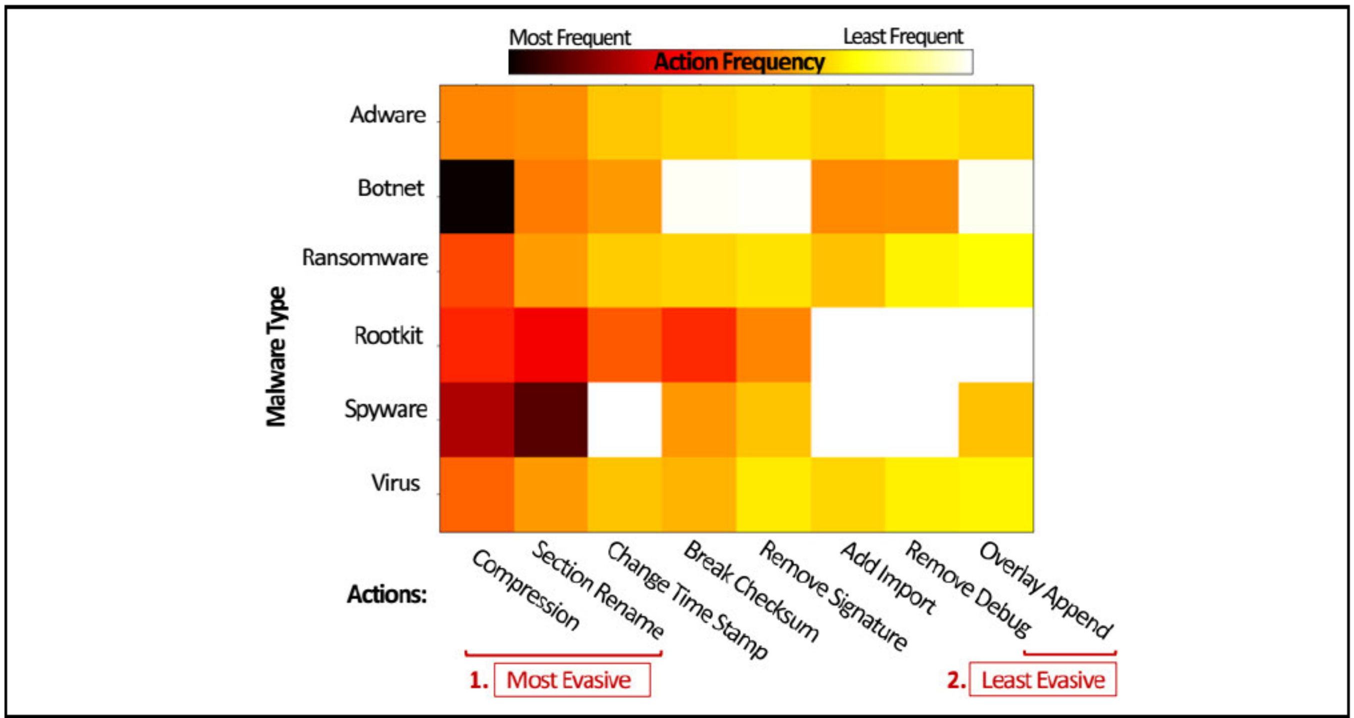

Figure 9. Frequency of Evasive Adversarial Actions for EMBER Malware Detector

Discussion

Guided by the computational design science paradigm (Gregor & Hevner, 2013; Rai, 2017), our study concerns the emerging problem of the robustness of cyber defense AI agents against adversarial attacks. To address the problem, we extensively explored viable solutions and designed a novel IT artifact by drawing on two statistical learning theories: reinforcement learning (RL) and robust optimization (RO). The proposed IT artifact was instantiated in the context of adversarial malware attacks, where a novel RL-based method was implemented to emulate adversarial attacks, and RO was leveraged to strengthen the robustness of malware detectors. The instantiation of our RADAR framework outperformed the state-of-the-art benchmark methods in emulating adversarial malware attacks and robustifying malware detectors and provided actionable insights to inform the design of malware detectors.

Our contribution to the IS knowledge base is threefold. First, our RADAR framework contributes to the IS security research with a novel design for strengthening the robustness of cyber defense AI agents. While the adversarial nature of cyber defense has been recognized in behavioral and economic security research, prior research on cyber defense AI agents rarely examines the effectiveness of the AI agents in such adversarial environments. To this end, the RADAR framework models the adversarial environment with a two-player infinite game, resulting in a robust AI agent. Second, the proposed r-VAC contributes to the prescriptive knowledge of adversarial attack emulation by providing an effective approach to emulating adversaries that take sequential actions to conduct the attack. While most existing AAE methods overlook the sequence of actions for achieving successful adversarial attacks, our r-VAC method shows that explicitly modeling the sequence of actions not only results in more effective adversarial attacks but can also provide actionable insights for improving the design of cyber defense AI agents. Our findings suggest that revealing highly evasive action sequences can help designers discover and mitigate the vulnerabilities of cyber defense AI agents. Third, we contribute to robust optimization by validating and extending the theory in an emerging and high-impact cybersecurity domain. We found that modeling the game between the adversary and the cyber defense AI agent as a robust optimization problem is particularly effective for strengthening the robustness of cyber defense AI agents. Moreover, from a methodological perspective, our RL-RO approach extends the optimization objective to account for adversarial attacks achieved by taking sequences of discrete actions.

In practice, RADAR can serve as a sandbox environment that enables a continuous test-and-improvement loop, enabling the defenders to be one step ahead of the adversary. Such an improvement loop could reduce an organization's vulnerability to unseen cyber attacks significantly. This has positive financial and practical implications across the industry. As shown in our experiments, RADAR effectively reduced malware attack success rates by up to seven times on average. Given the $2.6M annual cost of malware attacks per firm, together with the recent 11% increase in malware attacks to IT firms (Bissell et al., 2019), the adoption of such AI-enabled cyber defense design can potentially generate sizable financial benefits for the industry.

Apart from the financial benefits, we foresee three categories of cyber defense stakeholders that can benefit from our study: cybersecurity managers, cyber defense providers, and cybersecurity analysts. Managers can continuously monitor the security posture of their AI-based cyber defense guided by the vulnerability reports generated from emulating adversarial attacks in RADAR. Cyber defense providers can benefit from the iterative feedback loop embedded in RADAR to increase the robustness of their AI-based cyber defense solutions before large-scale deployment in IT firms. Cybersecurity analysts, including malware analysts and other security practitioners in security operations centers (SOCs), can leverage RADAR as a tool to mitigate unseen attacks such as new adversarially generated malware variants. Although our research can potentially contribute to strengthening cyber defense in security organizations, precautionary measures are needed to ensure ethical usage and prevent unwanted misuse. These measures include restricting access to RADAR's source code, granting access to only known academic research communities, or providing RADAR's functionality as a secure service/API and monitoring its usage. With these measures in place, benevolent use can significantly outweigh malicious usage. Similar successful examples have been observed for other tools in the cybersecurity community (e.g., Kali Linux for penetration testing).

Conclusion, Limitations, and Future Directions

Cyber defense AI agents protecting IT infrastructure are vulnerable to adversarial attacks, introducing an immense security risk and causing disastrous societal outcomes. It is crucial to defend AI agents against adversaries generating large-scale adversarial attacks automatically. This requires keeping up with the arms race between the defender and the adversary. Motivated by this critical need, this study draws on RO and RL theories in the design of a novel RADAR framework for improving the robustness of cyber defense AI agents. RADAR enables an adversarial game between the adversary and defender. Via rigorous evaluation, we showed that under RADAR, emulating adversarial attacks improves the robustness of AI agents. RADAR was instantiated in the malware attack domain, and its performance was extensively measured against state-of-the-art benchmarks. Using RADAR resulted in reducing the malware attack success rate by seven times. We also conducted a case study on a highly reputable AI-based malware detector to show the practical utility of RADAR in designing robust malware detectors. Finally, we discussed contributions to the IS knowledge base and practical implications.

There are several future research directions. First, while our experiments use a set of actions to rigorously evaluate our framework due to limited computing resource constraints, RADAR can accommodate extended action spaces to include more adversarial actions. One promising future direction is to automate the identification and extraction of adversarial actions from salient attacker behavior using imitation learning, augmented intelligence, and generative models such as GPT. Similarly, our state space could be further extended with representation learning to automatically identify useful features of malware executables. Second, RADAR was empirically tested to show that, even with suboptimal emulated adversarial attacks, cyber defense AI agents could still be robustified in practice. Future research could investigate alternative strategies to balance exploration and exploitation and theoretically quantify the quality of emulated adversarial attacks and their effect on robustification using the analysis of regret (Tranos & Proutiere, 2021). Third, the RL technique and the RO objective in RADAR could be further enhanced by the future state of the art and adapted to the cyber defense application domain to reflect domain characteristics. Lastly, though our experiments in the malware attack domain did not suffer from reward sparsity, in other applications, reward sparsity could hamper RL-based adversarial attack emulation. A promising direction for addressing reward sparsity is using a combination of model-based and model-free RL methods.

Acknowledgments

We would like to thank Dr. Jeremiah Birrell from Texas State University for the illuminating discussion and help with characterizing the size of the action space. Yidong Chai is also affiliated with Key Laboratory of Philosophy and Social Sciences for Cyberspace Behavior and Management of Anhui Province. This material is based upon work supported by the National Science Foundation (NSF) under the grants CNS-1936370 (SaTC CORE) and DGE-1921485 (SFS), as well as the National Science Foundation of China (NSFC) under Grants 72342011, 72322019, and 72293581. Weifeng Li is also supported by the Terry-Sanford Research Award.

References

Abbasi, A., Zhang, Z., Zimbra, D., Chen, H., & Nunamaker, J. F., Jr. (2010). Detecting fake websites: The contribution of statistical learning theory. *MIS Quarterly, 34*(3), 435-461. https://doi.org/10.2307/25750686

Allix, K., Bissyandé, T. F., Klein, J., & Le Traon, Y. (2015). Are your training datasets yet relevant? In F. Piessens, J. Caballero, & N. Bielova (Eds.), *Engineering secure software and systems* (pp. 51-67). Springer.

Ampel, B., Samtani, S., Zhu, H., & Chen, H. (2024). Creating proactive cyber threat intelligence with hacker exploit labels: A deep transfer learning approach. *MIS Quarterly, 48*(1), 137-166. https://doi.org/10.25300/MISQ/2023/17316

Anderson, H. S., Kharkar, A., Filar, B., Evans, D., & Roth, P. (2018). *Learning to evade static PE machine learning malware models via reinforcement learning.* arXiv. https://doi.org/10.48550/arXiv.1801.08917

Anderson, H. S., & Roth, P. (2018). *EMBER: An open dataset for training static PE malware machine learning models.* arXiv https://doi.org/10.48550/arXiv.1804.04637

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., & Zaremba, W. (2017). Hindsight experience replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 5048-5058).

Angst, C. M., Block, E. S., D'Arcy, J., & Kelley, K. (2017). When do IT security investments matter? Accounting for the influence of institutional factors in the context of healthcare data breaches. *MIS Quarterly, 41*(3), 893-916. https://www.jstor.org/stable/26635018

Apruzzese, G., Colajanni, M., Ferretti, L., & Marchetti, M. (2019). Addressing adversarial attacks against security systems based on machine learning. In *Proceedings of the 11th International Conference on Cyber Conflict*.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning* (pp. 214-223).

Benjamin, V., Valacich, J., & Chen, H. (2019). DICE-E: A framework for conducting darknet identification, collection, evaluation with ethics. *MIS Quarterly, 43*(1), 1-22. https://www.jstor.org/stable/26847849

Benjamin, V., Zhang, B., Nunamaker, J. F., Jr., & Chen, H. (2016). Examining hacker participation length in cybercriminal internet-relay-chat communities. *Journal of Management Information Systems, 33*(2), 482-510. https://doi.org/10.1080/07421222.2016.1205918

Bertsimas, D., Nohadani, O., & Teo, K. M. (2010). Robust optimization for unconstrained simulation-based problems. *Operations Research, 58*(1), 161-178.

Bissell, K., LaSalle, R. M., & Cin, P. D. (2019). *Ninth annual cost of cybercrime study: Unlocking the value of improved cybersecurity protection*. Accenture and Ponemon Institute. https://www.accenture.com/us-en/insights/security/cost-cybercrime-study

Bloomberg. (2018). *Symantec unveils industry's first neural network to protect critical infrastructure from cyber warfare.* Bloomberg. https://www.bloomberg.com/press-releases/2018-12-05/symantec-unveils-industry-s-first-neural-network-to-protect-critical-infrastructure-from-cyber-warfare

Burns, A., Roberts, T. L., Posey, C., & Lowry, P. B. (2019). The adaptive roles of positive and negative emotions in organizational insiders' security-based precaution taking. *Information Systems Research, 30*(4), 1228-1247. https://doi.org/10.1287/isre.2019.0860

Castro, R. L., Schmitt, C., & Rodosek, G. D. (2019). ARMED: How automatic malware modifications can evade static detection? In *Proceedings of the 5th International Conference on Information Management* (pp. 20-27).

Chen, B., Ren, Z., Yu, C., Hussain, I., & Liu, J. (2019). Adversarial examples for CNN-based malware detectors. *IEEE Access, 7,* 54360-54371.

Cram, W. A., D'Arcy, J., & Proudfoot, J. G. (2019). Seeing the forest and the trees: A meta-analysis of the antecedents to information security policy compliance. *MIS Quarterly, 43*(2), 525-554. https://doi.org/10.25300/MISQ/2019/1511

Dang, H., Huang, Y., & Chang, E.-C. (2017). Evading classifiers by morphing in the dark. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 119-133).

Danskin, J. M. (2012). *The theory of max-min and its application to weapons allocation problems*. Springer.

Demetrio, L., Biggio, B., Lagorio, G., Roli, F., & Armando, A. (2021). Functionality-preserving black-box optimization of adversarial Windows malware. *IEEE Transactions on Information Forensics and Security, 16,* 3469-3478.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7,* 1-30.

Dey, S., Kumar, A., Sawarkar, M., Singh, P. K., & Nandi, S. (2019). EvadePDF: Towards evading machine learning based PDF malware classifiers. In *Proceedings of the International Conference on Security & Privacy* (pp. 140-150).

Ebrahimi, R., Nunamaker, J. F., Jr., & Chen, H. (2020). Semi-supervised cyber threat identification in dark net markets: A transductive and deep learning approach. *Journal of Management Information Systems, 37*(3), 694-722. https://doi.org/10.1080/07421222.2020.1790186

Ebrahimi, R., Chai, Y., Samtani, S., & Chen, H. (2022). Cross-lingual cybersecurity analytics in the international dark web with adversarial deep representation learning. *MIS Quarterly, 46*(2), 1209-1226. https://doi.org/10.25300/MISQ/2022/16618

Fang, Z., Wang, J., Li, B., Wu, S., Zhou, Y., & Huang, H. (2019). Evading anti-malware engines with deep reinforcement learning. *IEEE Access, 7,* 48867-48879.

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., & Levine, S. (2018). *Model-based value estimation for efficient model-free reinforcement learning.* arXiv. https://doi.org/10.48550/arXiv.1803.00101

Fellows, M., Mahajan, A., Rudner, T. G., & Whiteson, S. (2019). VIREL: A variational inference framework for reinforcement learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (pp. 7120-7134).

Fleshman, W., Raff, E., Sylvester, J., Forsyth, S., & McLean, M. (2019). *Non-negative networks against adversarial attacks.* arXiv https://doi.org/10.48550/arXiv.1806.06108

Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning* (pp. 1587-1596).

Goodfellow, I., McDaniel, P., & Papernot, N. (2018). Making machine learning robust against adversarial inputs. *Communications of the ACM, 61*(7), 56-66. https://doi.org/10.1145/3134599

Goosen, R., Rontojannis, A., Deutscher, S., Rogg, J., Bohmayr, W., & Mkrtchian, D. (2018). *Artificial intelligence is a threat to cybersecurity. It's also a solution*. Boston Consulting Group. https://image-src.bcg.com/Images/BCG-Artificial-Intelligence-Is-a-Threat-to-Cyber-Security-Its-Also-a-Solution-Nov-2018_tcm9-207468.pdf

Gregor, S., & Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. *MIS Quarterly, 37*(2), 337-355. https://www.jstor.org/stable/43825912

Grosse, K., Papernot, N., Manoharan, P., Backes, M., & McDaniel, P. (2016). *Adversarial perturbations against deep neural networks for malware classification.* arXiv. https://doi.org/10.48550/arXiv.1606.04435

Ha, D., & Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (pp. 2450-2462).

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the International Conference on Machine Learning.*

Han, D., Wang, Z., Zhong, Y, Chen, W., Yang, J., Lu, S., Shi, X., & Yin, X. (2020). *Practical traffic-space adversarial attacks on learning-based NIDSs.* arXiv. https://doi.org/10.48550/arXiv.2005.07519

Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-learning. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence* (pp. 2094–2100).

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence.*

Hu, W., & Tan, Y. (2018). Black-box attacks against RNN based malware detection algorithms. In *Proceedings of the Workshops at the 32nd AAAI Conference on Artificial Intelligence.*

Hu, W., & Tan, Y. (2022). Generating adversarial malware examples for black-box attacks based on GAN. In *Proceedings of the International Conference on Data Mining and Big Data* (pp. 409-423).

Hui, K.-L., Vance, A., & Zhdanov, D. (2018). *Securing digital assets.* MIS Quarterly Research Curations. https://www.misqresearchcurations.org/blog/2017/5/10/securing-digital-assets-1

Jang, E., Gu, S., & Poole, B. (2017). Categorical reparameterization with Gumbel-Softmax. In *Proceedings of the International Conference on Learning Representations.*

Karhu, K., Gustafsson, R., & Lyytinen, K. (2018). Exploiting and defending open digital platforms with boundary resources: Android's five platform forks. *Information Systems Research, 29*(2), 479-497. https://doi.org/10.1287/isre.2018.0786

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 3146-3154).

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations.*

Kolter, Z., & Madry, A. (2018). Adversarial robustness: Theory and practice. In *Proceedings of Neural Information Processing Systems.*

Kreuk, F., Barak, A., Aviv-Reuven, S., Baruch, M., Pinkas, B., & Keshet, J. (2018). *Adversarial examples on discrete sequences for beating whole-binary malware detection.* arXiv. https://doi.org/10.48550/arXiv.1802.04528

Kujawa, A., Zamora, W., Segura, J., Reed, T., Collier, N., Umawing, J., Boyd, C., Arntz, P., & Ruiz, D. (2020). *State of malware report.* Malwarebytes Labs. https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report.pdf

Kyadige, A., Rudd, E., & Berlin, K. (2020, May 21). Learning from context: A multi-view deep learning architecture for malware detection. In *Proceedings of the 3rd Deep Learning and Security Workshop*.

Li, W., Chen, H., & Nunamaker, J. F., Jr. (2016). Identifying and profiling key sellers in cyber carding community: AZSecure text mining system. *Journal of Management Information Systems, 33*(4), 1059-1086. https://doi.org/10.1080/07421222.2016.1267528

Liang, H., Xue, Y., Pinsonneault, A., & Wu, Y. (2019). What users do besides problem-focused coping when facing IT security threats: An emotion-focused coping perspective. *MIS Quarterly, 43*(2), 373-394. https://doi.org/10.25300/MISQ/2019/14360

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations.*

Lin, Z., Shi, Y., & Xue, Z. (2019). *IDSGAN: Generative adversarial networks for attack generation against intrusion detection.* arXiv https://doi.org/10.48550/arXiv.1809.02077

Maddison, C. J., Mnih, A., & Teh, Y. W. (2017). The Concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations.*

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *Proceedings of the International Conference on Learning Representations.*

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016a). Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning* (pp. 1928-1937).

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016b). Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning* (pp. 1928-1937).

Monteiro, J., Albuquerque, I., Akhtar, Z., & Falk, T. H. (2019). Generalizable adversarial examples detection based on bi-model decision mismatch. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics* (pp. 2839-2844). IEEE.

Narayanan, A., Chandramohan, M., Chen, L., & Liu, Y. (2018). A multi-view context-aware approach to Android malware detection and malicious code localization. *Empirical Software Engineering, 23*(3), 1222-1274. https://doi.org/10.1007/s10664-017-9539-8

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems, 24*(3), 45-77. https://www.jstor.org/stable/40398896

Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., & Cavallaro, L. (2019). TESSERACT: Eliminating experimental bias in malware classification across space and time. In *Proceedings of the 28th USENIX Security Symposium* (pp. 729-746).

Qiu, S., Liu, Q., Zhou, S., & Wu, C. (2019). Review of artificial intelligence adversarial attack and defense technologies. *Applied Sciences, 9*(5), 909-938. https://doi.org/10.3390/app9050909

Racanière, S., Weber, T., Reichert, D., Buesing, L., Guez, A., Rezende, D. J., Badia, A. P., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P., Hassabis, D. Silver, D., & Wierstra, D. (2017). Imagination-augmented agents for deep reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 5690-5701).

Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., & Nicholas, C. (2018). *Malware detection by eating a whole EXE.* arXiv. https://doi.org/10.48550/arXiv.1710.09435

Rai, A. (2017). Editor's comments: Diversity of design science research. *MIS Quarterly, 41*(1), 3-18.

Rosenberg, I., Shabtai, A., Rokach, L., & Elovici, Y. (2018). Generic black-box end-to-end attack against state-of-the-art API call based malware classifiers. In *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses* (pp. 490-510). Springer.

Samtani, S., Chinn, R., Chen, H., & Nunamaker, J. F., Jr. (2017). Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence. *Journal of Management Information Systems, 34*(4), 1023-1053.

Saxe, J., & Berlin, K. (2015). Deep neural network based malware detection using two dimensional binary program features. In *Proceedings of the 10th International Conference on Malicious and Unwanted Software* (pp. 11-20).

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning* (pp. 1889-1897).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal policy optimization algorithms.* arXiv https://doi.org/10.48550/arXiv.1707.06347

Sen, R., Verma, A., & Heim, G. R. (2020). Impact of cyberattacks by malicious hackers on the competition in software markets. *Journal of Management Information Systems, 37*(1), 191-216. https://doi.org/10.1080/07421222.2019.1705511

Shahpasand, M., Hamey, L., Vatsalan, D., & Xue, M. (2019). Adversarial attacks on mobile malware detection. In *Proceedings of the IEEE 1st International Workshop on Artificial Intelligence for Mobile* (pp. 17-20).

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K, & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science, 362*(6419), 1140-1144. https://doi.org/10.1126/science.aar6404

Song, W., Li, X., Afroz, S., Garg, D., Kuznetsov, D., & Yin, H. (2022). Mab-malware: A reinforcement learning framework for black-box generation of adversarial malware. In *Proceedings of the ACM Asia Conference on Computer and Communications Security* (pp. 990-1003).

Suciu, O., Coull, S. E., & Johns, J. (2019). Exploring adversarial examples in malware detection. In *Proceedings of the IEEE Security and Privacy Workshops* (pp. 8-14).

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

Temizkan, O., Park, S., & Saydam, C. (2017). Software diversity for improved network security: Optimal distribution of software-based shared vulnerabilities. *Information Systems Research, 28*(4), 828-849. https://doi.org/10.1287/isre.2017.0722

Tolido, R., van der Linden, G., Delabarre, L., Theisler, J., Khemka, Y., Thieullent, A.-L., Frank, A., Buvat, J., & Cherian, S. (2019). *Reinventing cybersecurity with artificial intelligence: The new frontier in digital security*. Capgemini Research Institute.

Tranos, D., & Proutiere, A. (2021). Regret analysis in deterministic reinforcement learning. In *Proceedings of the IEEE Conference on Decision and Control* (pp. 2246-2251).

Usama, M., Asim, M., Latif, S., & Qadir, J. (2019). Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In *Proceedings of the International Wireless Communications & Mobile Computing Conference* (pp. 78-83).

Vance, A., Jenkins, J. L., Anderson, B. B., Bjornn, D. K., & Kirwan, C. B. (2018). Tuning out security warnings: A longitudinal examination of habituation through fMRI, eye tracking, and field experiments. *MIS Quarterly, 42*(2), 355-380. https://doi.org/10.25300/MISQ/2018/14124

Wang, Z. (2018). Deep learning-based intrusion detection with adversaries. *IEEE Access, 6,* 38367-38384. https://doi.org/10.1109/ACCESS.2018.2854599

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., & de Freitas, N. (2017). Sample efficient actor-critic with experience replay. In *Proceedings of the International Conference on Learning Representations.*

Yin, H. S., Langenheldt, K., Harlev, M., Mukkamala, R. R., & Vatrapu, R. (2019). Regulating cryptocurrencies: A supervised machine learning approach to de-anonymizing the Bitcoin blockchain. *Journal of Management Information Systems, 36*(1), 37-73. https://doi.org/10.1080/07421222.2018.1550550

Yoo, C. W., Goo, J., & Rao, H. R. (2020). Is cybersecurity a team sport? A multilevel examination of workgroup information security effectiveness. *MIS Quarterly, 44*(2), 907-931. https://doi.org/10.25300/MISQ/2020/15477

Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems, 30*(9), 2805-2824.

Yue, W. T., Wang, Q., & Hui, K.-L. (2019). See no evil, hear no evil? Dissecting the impact of online hacker forums. *MIS Quarterly, 43*(1), 73-95. https://doi.org/10.25300/MISQ/2019/13042

Zhang, H., & Yu, T. (2020). Taxonomy of reinforcement learning algorithms. In H. Dong, Z. Ding, & S. Zhang (Eds.), *Deep reinforcement learning: Fundamentals, research and applications* (pp. 125-133). Springer.

Zhao, S., Li, J., Wang, J., Zhang, Z., Zhu, L., & Zhang, Y. (2021). AttackGAN: Adversarial attack against black-box IDS using generative adversarial networks. *Procedia Computer Science, 187,* 128-133. https://doi.org/10.1016/j.procs.2021.04.118

Zhong, F., Cheng, X., Yu, D., Gong, B., Song, S., & Yu, J. (2024). MalFox: Camouflaged adversarial malware example generation based on conv-GANs against black-box detectors. *IEEE Transactions on Computers, 73*(4), 980-893. https://doi.org/10.1109/TC.2023.3236901

About the Authors

Reza Ebrahimi is an assistant professor and the founder of Star-AI Lab in the School of Information Systems at the University of South Florida. Ebrahimi received his Ph.D. in management information systems from the University of Arizona, where he was a research associate at the Artificial Intelligence (AI) Lab in 2021. He received his master's degree in computer science from Concordia University, Canada, in 2016. Reza's dissertation on AI-enabled cybersecurity analytics received the ICIS ACM SIGMIS Best Doctoral Dissertation Award in 2021. His current research is focused on statistical and adversarial machine learning theories for AI-enabled secure and trustworthy cyberspace. Reza has published over 40 articles in peer-reviewed journals, conferences, and workshops, including *MIS Quarterly*, *Journal of Management Information Systems*, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* NeurIPS, ICLR, *IEEE Transactions on Dependable and Secure Computing, Digital Forensics, Applied Artificial Intelligence,* IEEE S&PW, AAAI, IEEE ICDM, and IEEE ISI. He serves as a program chair and program committee member for the IEEE ICDM Workshop on Machine Learning for Cybersecurity (MLC) and IEEE S&P Workshop on Deep Learning Security and Privacy (DLSP). He is also serving as an organizer of the 2025 IEEE S&P Workshop on Human-Machine Intelligence for Security Analytics (HMI-SA). He has contributed to several projects supported by the National Science Foundation (NSF). He is an IEEE Senior Member and a member of the ACM, AAAI, and AIS.

Yidong Chai is a researcher at the Hefei University of Technology and at the City University of Hong Kong. He received his bachelor's degree in information systems from Beijing Institute of Technology, and his Ph.D. degree in management information systems from Tsinghua University. His research fields include machine learning, signal processing, and natural language processing. His work has appeared in journals including *MIS Quarterly, Information Systems Research, Knowledge-Based Systems, and Applied Soft Computing*, as well as conferences and workshops including IEEE TDSC, IEEE TKDE, IEEE S&PW, INFORMS Workshop on Data Science, Workshop on Information Technology Systems, International Conference on Smart Health, and International Conference on Information Systems.

Weifeng Li is an associate professor in the Department of Management Information Systems, the University of Georgia. He received his Ph.D. in management information systems from the University of Arizona. His research interests are the security of artificial intelligence systems and the development of artificial intelligence systems for cybersecurity applications. His methodological focus includes machine learning, natural language processing, and Bayesian modeling. His research has appeared in peer-reviewed journals and conferences, including *MIS Quarterly, Journal of Management Information Systems, IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Dependable and Secure Computing, ACM Computing Surveys*, ICIS, and IEEE Intelligence and Security Informatics. His research has been supported by the National Science Foundation's Secure and Trustworthy Cyberspace program.

Jason Pacheco is an assistant professor in the Department of Computer Science at the University of Arizona in Tucson, Arizona. Before joining the University of Arizona, he completed his Ph.D. in computer science at Brown University with Erik Sudderth. He received the Air Force Office of Scientific Research Young Investigator Program Award for Robust Maximum Entropy Planning, Learning, and Control in Uncertain Environments. He also held a postdoc position at MIT CSAIL with John Fisher III. His work has been published in top-tier conference proceedings such as IEEE/CVF CVPR, NeurIPS, ICML, IEEE S\&PW, and AISTATS. His research interests lie in statistical machine learning, sequential decision-making, and probabilistic graphical models to be used in a variety of domains including computer vision, cybersecurity, computational biology, and computational neuroscience.

Hsinchun Chen is Regents Professor and Thomas R. Brown Chair in Management and Technology in the Management Information Systems Department at the Eller College of Management, University of Arizona. He received his Ph.D. in information systems from New York University. He is the author/editor of 20 books, 300 journal articles, and 200 refereed conference articles covering digital library, data/text/web mining, business analytics, security informatics, and health informatics. He founded the Artificial Intelligence Lab at the University of Arizona in 1989, which has received $50M+ research funding from the NSF, National Institutes of Health, National Library of Medicine, Department of Defense, Department of Justice, Central Intelligence Agency, Department of Homeland Security, and other agencies (100+ grants, 50+ from NSF). His COPLINK/i2 system for security analytics was commercialized in 2000 and acquired by IBM as its leading government analytics product in 2011. The COPLINK/i2 system is used in 5,000+ law enforcement jurisdictions and intelligence agencies in the U.S. and Europe, making a significant contribution to public safety worldwide.

Appendix A

A Taxonomy of Selected Major Recent Deep RL Methods

To systematically identify RL methods suitable for AAE, we expand the taxonomy given by Zhang and Yu (2020) to provide a taxonomy of prevailing deep RL methods. While not meant to be exhaustive, we aim to include recent highly cited deep RL methods. According to the AAE domain requirements noted in the RL review, Table A1 summarizes recent highly adopted RL methods across four dimensions: dependence on a predetermined model of the environment (model-free vs. model-based), sample efficiency (very high, high, and medium), supported action types (discrete vs. continuous), and supported state space size (very high, high, medium).

Table A1. Selected Major Recent Deep RL Methods

Model Dependence	RL Model	Sample Efficiency	State Space Size	Action Space Type
Model-Based	MBVE (Feinberg et al., 2018)	Very high	Medium	Discrete & Continuous
	WM (Ha & Schmidhuber, 2018)	Very high	Medium	Discrete & Continuous
	AlphaZero (Silver et al., 2018)	Very high	Medium	Discrete
	I2A (Racanière et al., 2017)	Very high	Medium	Discrete
Model-Free	PPO (Schulman et al., 2017)	Medium	High	Discrete & Continuous
	A3C (Mnih et al., 2016)	Medium	Medium	Discrete & Continuous
	TRPO (Schulman et al., 2015)	Medium	High	Discrete & Continuous
	VAC (Fellows et al., 2019)	High	Very high	Discrete & Continuous
	SAC (Haarnoja et al., 2018)	High	High	Continuous
	TD3 (Fujimoto et al., 2018)	High	High	Continuous
	HER (Andrychowicz et al., 2017)	High	Medium	Continuous
	ACER (Wang et al., 2017)	High	High	Discrete
	DDQN (Hasselt et al., 2016; Mnih et al., 2015)	High	High	Discrete
	DDPG (Lillicrap et al., 2016)	High	High	Continuous

Note: MBVE: Model-Based Value Estimation, WM: World Model, I2A: Imagination-Augmented Agent, PPO: Proximal Policy Optimization, A3C: Asynchronous Advantage Actor-Critic, TRPO: Trust Region Policy Optimization, VAC: Variational Actor-Critic, SAC: Soft Actor-Critic, DDPG: Deep Deterministic Policy Gradient, TD3: Twin Delayed DDPG, HER: Hindsight Experience Replay, ACER: Actor-Critic with Experience Replay, DDQN: Double Deep Q Network

In AAE, there is often no prior knowledge to model the environment due to its black-box nature. If available, such a model would determine the probability of transitioning to a new state given a current state and action (Feinberg et al., 2018; Ha & Schmidhuber, 2018; Racanière et al., 2017; Silver et al., 2018). In contrast, model-free RL methods are able to operate without requiring or trying to learn environment transition dynamics (Fellows et al., 2019; Fujimoto et al., 2018; Haarnoja et al., 2018; Hasselt et al., 2016; Mnih et al., 2016a; Schulman et al., 2015, 2017; Wang et al., 2017). Accordingly, we focus on the model-free RL paradigm in our design.

Appendix B

Adversarial Attack Vectors

B1. Reconnaissance Sources for Identifying Adversarial Attack Vectors

By consulting several currently active cybersecurity practitioners from academia, government, and industry, we identified four primary reconnaissance sources useful to identify adversarial attack vectors. Table B1 presents the reconnaissance sources for identifying adversarial attack vectors, along with potential information providers and examples. The reconnaissance sources appear in the order of usefulness, voted unanimously by the cybersecurity practitioners.

Table B1. Reconnaissance Sources for Identifying Adversarial Attack Vectors

Reconnaissance source	Providers	Example(s)
Current attack literature	• Academic publications (conferences, journals)	• S&P, CCS, TDSC, TIFS
Domain expert	• Security practitioners	• Malware/network analyst
Cybersecurity reports	• Security Evasion Competitions held by industry • Cybersecurity firms	• Endgame, Microsoft • FireEye, Avast, Mitre
Cyber defense AI agent design documents	• API documentation and source code from academic labs	• Laboratory of Physical Sciences (LPS)

Note: S&P: Security and Privacy, CCS: Computer and Communications Security, TDSC: Transactions on Dependable and Secure Computing, Transactions on Information Forensics and Security

Current attack literature from renowned cybersecurity publications such as Security and Privacy (S&P) and Computer Communications Security (CCS) is crucial to identifying the most recent attack vectors. Additionally, domain experts and security analysts (e.g., malware analysts) can effectively identify attack vectors. Cybersecurity reports from security evasion competitions held by industry, such as the Microsoft Machine Learning Security Evasion Competition (MLSEC) (https://mlsec.io) and periodic reports from cybersecurity firms (e.g., FireEye), are another valuable reconnaissance source to identify attack vectors. Lastly, the cyber defense AI agent's design documents (e.g., API description and source code) could be analyzed by practitioners to identify potential attack vectors. We used three reconnaissance sources (attack literature, domain expert, and MLSEC competition) to identify adversarial malware attack vectors.

B2. Preserving the Functionality of Malware Variants

Permissible adversarial actions must preserve the malicious functionality of the malware. We employed VirusTotal's API, a web-based platform that supports malware analysis, for assessing the functionality of malware executables after modification. VirusTotal offers useful malware behavior reports featuring static and dynamic analysis of the malware executable. These reports describe important information such as network behavior and file access behavior. Using the API, we compared the behavior reports of the modified variants and the original version (unmodified malware executables). In this process, we ensured that the key behavioral indicators of maliciousness in VirusTotal's report stayed the same before and after modification, showing that the modified executables can be run on the operating system and are fully functional.

B3. Extending Experiment 1 by Increasing the Maximum Number of Actions in an Episode

To show generalizability, we conducted a set of experiments that allowed up to 10 actions in each episode and compared the performance of our model with the strongest attack for each malware detector (i.e., A3C for EMBER and ACER for MalConv and NonNeg) from Experiment 1. As shown in Table B3, when up to 10 actions are allowed, the performance gap between the proposed method and the best-performing method further increases.

Table B3. Evasion Rate Comparisons for AI-Based Malware Detectors with Extended Action Sequence Length

Malware Detector	Model	Adware	Botnet	Ransomware	Rootkit	Spyware	Virus	Combined (all types)
LGBM	A3C	15.02%	39.50%	58.50%	39.11%	40.81%	50.22%	28.29%
	r-VAC (ours)	**26.41%**	**55.09%**	**75.19%**	**71.00%**	**36.30%**	**84.44%**	**<u>37.96%</u>**
MalConv	ACER	25.07%	41.05%	31.33%	37.40%	61.10%	50.75%	21.32%
	r-VAC (ours)	**31.03%**	**50.50%**	**34.50%**	**55.40%**	**67.30%**	**52.24%**	**<u>25.87%</u>**
NonNeg	ACER	10.89%	33.00%	25.11%	28.99%	26.55%	38.24%	17.65%
	r-VAC (ours)	**20.34%**	**52.17%**	**29.99%**	**55.03%**	**36.30%**	**45.01%**	**<u>21.65%</u>**

Appendix C

Policy Gradient Theorem and Action Reparameterization

To solve the objective given in Equation (2), the policy gradient theorem computes the gradients of the expected return of state s under policy π_θ, known as value function $V_\theta(s)$:

$$\nabla_\theta V_\theta(s) = \nabla_\theta \mathbb{E}_{a\sim\pi_\theta}[Q_{\pi_\theta}(s,a)] = \nabla_\theta \textstyle\sum_{a\in\mathcal{A}} \pi_\theta(a|s) Q_{\pi_\theta}(s,a) \quad \text{; definition of expectation}$$

$$= \textstyle\sum_{a\in\mathcal{A}} Q_{\pi_\theta}(s,a)\nabla_\theta \pi_\theta(a|s) + \pi_\theta(a|s)\nabla_\theta Q_\theta(s,a) \quad \text{; derivative of the product}$$

$$= \textstyle\sum_{a\in\mathcal{A}} Q_{\pi_\theta}(s,a)\nabla_\theta \pi_\theta(a|s) + \pi_\theta(a|s) \sum_{s'\in S} p(s'|s,a)\nabla_\theta V_\theta(s') \text{ ; Bellman equation}$$

As the second term is recursively defined, the gradient depends on the first term:

$$\textstyle\sum_{a\in\mathcal{A}} Q_{\pi_\theta}(s,a)\nabla_\theta \pi_\theta(a|s) = \sum_{a\in\mathcal{A}} Q_{\pi_\theta}(s,a)\pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a|s) \quad \text{; derivative log trick}$$

$$= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\, Q_{\pi_\theta}(s,a)],$$

which explains the gradient form in Equation (3). In reparameterization with approximate sampling, the discrete action variable is reparameterized by a function $g(\theta, G)$, where G is a random variable admitting Gumbel distribution. Accordingly, $\nabla_\theta \mathbb{E}_a[Q_{\pi_\theta}(s,a)]$ is rewritten as:

$$\nabla_\theta \mathbb{E}_a[Q_{\pi_\theta}(s,a)] = \nabla_\theta \mathbb{E}_G[Q(s, g(\theta, G))]$$

$= \mathbb{E}_G[\nabla_\theta Q(s, g(\theta, G))]$; move expectation under G out of gradient w.r.t. θ, which indicates the gradient is estimated with sampling as noted in “Action Reparameterization.”

Appendix D

RL-Based Robust Optimization (RL-RO) Algorithm

Algorithm 1 presents the outer minimization procedure in RL-RO formally. The algorithm accepts a policy network $\Pi_{\theta,h_\phi}(\cdot)$ learned in RADAR's Phase 1, a defense AI agent's model h_ϕ, and a dataset of malicious inputs $(x, y) \sim \mathcal{D}$ detectable by the defense AI agent, where x and y denote the input malicious samples and their class labels (denoting malicious category), respectively, and $\mathcal{D}$ denotes the sampling distribution. The output of the algorithm is the robust cyber defense AI agent's model $h_{\phi'}$. The algorithm features an iterative procedure continuing until the change in loss (the error rate of the defense AI agent) is lower than a small threshold ϵ. At each iteration, adversarial samples are generated via the learned policy, and the AI agent loss is minimized based on Equation (4). Consistent with the literature, to solve Equation (4), Adam optimizer (Kingma & Ba, 2015) was used to conduct stochastic gradient descent.

```
**Algorithm 1:** Outer Minimization in RL-based Robust Optimization (RL-RO)
**Input**: Policy network $\Pi_{\theta,h_\phi}(\cdot)$, vulnerable AI agent $h_\phi$, set of detectable malicious inputs $(x, y) \sim \mathcal{D}$, constant $\epsilon$
**Output**: Adversarially robust cyber defense AI agent $h_{\phi'}$
**Initialization**: $L' = 0$, $\Delta L = 1$, $\epsilon = 0.001$
**While** the loss change $\Delta L > \epsilon$ **do**:
    set $\boldsymbol{L = 0}$
    **for each** $(x, y)$ **do**:
        Obtain an adversarial sample via $\Pi_{\theta,h_\phi}(x)$
        Update the loss value using Equation (4): $L \leftarrow L + \ell\left(h_\phi\left(\Pi_{\theta,h_\phi}(x)\right), y\right)$
    Update $\phi$ via stochastic gradient descent using Adam optimizer: $\phi \leftarrow \text{Adam}\ (\nabla_\phi L, \phi)$
    Compute the loss change $\Delta L \leftarrow L - L'$
    $L' \leftarrow L$ \\ update the defense AI agent's loss
$\phi' \leftarrow \phi$ \\ Store the robustified model's parameters
**Return** robustified AI agent $h_{\phi'}$.
```

Appendix E

Empirical Efficiency Analysis of RADAR

Due to the lack of a closed-form solution for the RL objective function, guaranteed convergence analysis could be impractical and challenging; therefore, we performed an empirical time analysis of attack generation per malware type to provide insights on the empirical time complexity of RADAR when used to emulate adversarial attacks for an AI-enabled detector (MalConv in this case). Table E1 shows the results in terms of mean elapsed time in seconds.

Table E1. RADAR's Mean Elapsed Time of Attack on each Malware Type against MalConv

	Malware type						
	Adware	**Botnet**	**Ransomware**	**Rootkit**	**Spyware**	**Virus**	**Average**
Mean elapsed time	211.00 s	157.80 s	56.50 s	143.80 s	112.80 s	75.24 s	126.19 s

In all malware types, evaluating the evasiveness of a generated malware variant against a detector took less than a second. As shown in Table E1, generating each adversarial malware variant took 126.19 seconds on average across all malware types. Assessing the functionality via VirusTotal API per malware file was done in less than 5 seconds. The results suggest that RADAR could serve as an empirical robustness evaluation tool for modern detectors at scale. To further improve the scalability, multiple versions of RADAR can be executed concurrently.