Beyond Conventional Defenses: Proactive and Adversarial-Resilient Hardware Malware Detection using Deep Reinforcement Learning

Zhangying He
California State University, Long Beach
zhangying.he01@student.csulb.edu

Houman Homayoun
University of California, Davis
hhomayoun@ucdavis.edu

Hossein Sayadi
California State University, Long Beach
hossein.sayadi@csulb.edu

ABSTRACT

This research investigates the vulnerability of machine learning-enabled Hardware Malware Detection (HMD) methods to adversarial attacks, a pressing concern undermining their efficacy against malware threats. While prior adversarial learning research primarily centered on image classification and/or overlooked adversarial attacks in HMDs, we delve into the distinctive challenges posed by adversarial attacks in the context of tabular data from processors' performance counters. This paper introduces a proactive and robust multi-phased adversarial learning and defense framework based on Deep Reinforcement Learning (DRL). In the initial phase, highly effective adversarial attacks are employed to circumvent ML-based detection mechanisms. Subsequently, an efficient deep reinforcement learning technique based on Advantage Actor Critic (A2C) is presented to predict adversarial attack patterns in real-time. Next, ML models are fortified through adversarial training to enhance their defense capabilities against both malware and adversarial attacks. To achieve greater efficiency, an RL-based constraint controller using an Upper Confidence Bounds (UCB) algorithm is proposed that dynamically assigns adversarial defense responsibilities to specialized RL agents based on different performance constraints. The results demonstrate the proposed framework's effectiveness, indicating up to 86% boost in F1-score for defending against adversarial attacks across all models, leading to detection rate of 96.1% for the top-performing adaptive malware detector.

KEYWORDS

Adversarial Learning, Cybersecurity, Hardware Malware Detection, Machine Learning, Reinforcement Learning

1 INTRODUCTION

The escalating prevalence of security vulnerabilities within modern computing systems has rendered them increasingly susceptible to complex cyberattacks. In this evolving landscape, robust malware detection has become crucial for preserving user data integrity and confidentiality. Conventional software-centric detection approaches have exhibited performance inefficiencies, catalyzing the emergence of Hardware Malware Detection (HMD) [4, 8, 18].

HMD methods harness low-level microarchitectural features captured by Hardware Performance Counters (HPCs) and employ Machine Learning (ML) techniques to discern between malicious software and legitimate programs. In this research, we have identified and addressed five key challenges linked to ML-enabled hardware malware detection methods.

Challenge 1- ML Models' Susceptibility to Adversarial Attacks: ML models, despite achieving high accuracy, are susceptible to adversarial attacks where cleverly crafted input data can mislead the model's predictions [20]. In hardware malware detection, this vulnerability can be exploited by attackers to evade detection or compromise the model's integrity, posing a significant challenge to security.

Challenge 2- Adaptation of Methods to Tabular Data: While adversarial learning and defense methods have excelled in image classification, they face challenges when applied to tabular data—an increasingly vital aspect in emerging fields like cybersecurity [1]. The distinctive characteristics and vulnerabilities of tabular data pose a unique adaptation hurdle compared to image-based datasets.

Challenge 3- Complex Task of Preserving Original Malware Functionality: In the context of cybersecurity, ensuring that adversarial attacks do not disrupt the original functionality of malware is a complex task. Adversarial attacks mask malware as benign while maintaining its harmful actions, causing a tougher challenge in altering feature vectors than manipulating visible image pixels.

Challenge 4- Vulnerabilities in Hardware Malware Detectors: HPC-based malware detectors, while advantageous due to their prevalence and hidden nature, are not immune to vulnerabilities [5, 6, 8, 9]. Adversarial attackers can exploit these weaknesses by devising strategies to manipulate or obfuscate features, potentially evading detection and posing a threat to the integrity of detection systems.

Challenge 5- Lack of Robust and Adaptive Adversarial Defense: Previous efforts focused on generating valid adversarial attacks that are imperceptible to both human observers and well-tuned machine learning models [6, 16]. To navigate the dynamic cybersecurity landscape and effectively counter evolving threats, an essential challenge lies in crafting a resilient and adaptable defense approach.

This work introduces an effective adversarial learning and defense approach tailored for hardware malware detection. To this aim, we utilize unlabeled data to train a highly discriminative Deep Reinforcement Learning (DRL) agent for predicting adversarial attacks. Subsequently, we present a multi-phased framework involving adversarial training to fortify classical ML models' robustness. Then, we develop specialized RL-based agents to dynamically select ML models at run-time, aligning with the preferred performance requirements (e.g., latency, memory footprint, accuracy, etc.). Table 1 presents a compilation of previous research alongside a comparative analysis of our proposed approach concerning adversarial attacks and learning techniques. Our proposed adversarial-resilient malware detection approach excels in detecting adversarial attacks,

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3658252

Table 1: Comparative analysis of prior adversarial attacks and learning methods with our proposed approach

Research	Perturbed Features	Attack Type	Attack Success Rate	Adversarial Defense Approach	Defense Improvement Evaluation	Adaptive Learning
[10]	pdf files	Inference integrity (MA)	90+%	✗	✗	✗
[22]	APK/DEX file call graph	Inference integrity (MA)	93%	✗	✗	✗
[15]	PE header matadata	Inference integrity (MA)	98%	✗	✗	✗
[14]	API call sequences	Inference integrity (MA)	100%	✗	✗	✗
[20]	APKs	Inference integrity (MA)	50%	✗	✗	✗
[7]	API calls, permissions	Inference integrity (MA)	69%	Defensive distillation, adversarial training	adversarial training: up to 6% (misclassification rate)	✗
[6]	HPCs	Inference integrity ($M \Leftrightarrow B$)	97%	✗	✗	✗
[9]	HPCs	Inference integrity (MA)	✗	Moving target defense	up to 31.5%(accuracy), 22.6%(precision)	✗
[5]	HPCs	Inference integrity ($M \Leftrightarrow B$)	97%	Adversarial training	up to 63.1%(accuracy), 35.1%(precision), 63.2%(recall), 70.1%(F1-score)	✗
[8]	HPCs	Inference integrity (MA)	80%	Adversarial training, randomized classifier	✗	✗
This Work	**HPCs**	**Inference integrity (MA)**	**100%**	**Adversarial training, RL-based dynamic malware defense**	**up to 86%(F1-score), 47%(accuracy), 63%(AUC), 64%(precision), 87%(recall), 87%(TPR)**	✓

M ⇔ B: malware as benign and benign as malware, MA: malware attack

learning from them, enhancing the resiliency of HMD, and dynamically adapting to counteract evolving threats. It also equips the research community with a powerful tool to combat adversarial threats effectively. The main contributions of this work include:

- We present a customized adversarial generation method tailored to tabular hardware data in HMDs that achieves a remarkable success rate of 100% in producing adversarial attacks. This approach underscores the susceptibility of ML-based HMDs, reducing the detection rates by up to 79%.
- To address the challenges posed by the generated adversarial attacks, we introduce a multi-phased adversarial learning framework that combines deep reinforcement learning with classical ML models. To this end, first, we train a highly discriminative DRL-based adversarial predictor using unlabeled data and feedback rewards to predict adversarial attacks, which achieves 100% in identifying adversarial attacks from uncertain streaming data.
- Subsequently, we propose a two-stage proactive and performance-aware approach: the first stage employs adversarial training to enhance the ML model's robustness, and the second stage introduces specialized RL-based agents. These agents dynamically select optimal models at run-time based on predefined performance criteria, ensuring robust and adaptive malware detection.

2 PROPOSED ADVERSARIAL-RESILIENT MALWARE DETECTION FRAMEWORK

2.1 System Configuration and Feature Analysis

- *Data Acquisition:* We conducted our experiments on a computer system with an 11th Gen Intel Core i7 processor running Ubuntu 22.04.2 operating system. Data collection was facilitated using the Perf tool available on Linux, with a sampling time of 10ms. We gathered performance counter features from a diverse range of benign and malware applications. To create a diverse and representative dataset, we executed over 3,000 malware and benign applications encompassing a wide range of domains and functionalities. Leveraging the scripting capabilities of Perf, we automated the performance monitoring process and seamlessly integrated it into our proposed methodologies. This allowed us to collect a wide range of hardware events (+30 events) from diverse applications efficiently. We collected and analyzed malware applications from online repositories like VirusShare and VirusTotal, encompassing various classes such as Worms, Viruses, Botnets, Ransomware, and more. To isolate and gather HPC information, we utilized Linux Containers (LXC) that provide direct access to hardware counters, setting it apart from common virtual platforms like VirtualBox, which emulates HPCs.

- *Feature Engineering:* We further apply an enhanced feature engineering approach, encompassing data cleaning, standard scaling, and Mutual Information (MI)-based feature selection to extract relevant hardware features. MI quantifies the mutual dependency between variables [17], and is employed to select the most informative HPC features highly relevant to the target label (exhibit a higher mutual dependency). Regarding each feature set X and label Y, the MI measure denoted as $I(X, Y)$, is obtained by estimating the marginal entropies $H(X)$, $H(Y)$, and the joint entropy $H(X, Y)$, which is equal to $I(X, Y) = H(X) + H(Y) - H(X, Y)$. We use the Scikit Learn library's *mutual_info_classif* algorithm. We then rank the estimated mutual information values between each feature and the target label, selecting the top hardware features (*LLC-load-misses, LLC-loads, cache-misses, cpu/cache-misses/*). The top four HPCs are used to create training and testing sub-datasets in an 80:20 ratio, with further 80:20 splitting for training and validation.

2.2 Threat Model

In our proposed framework, our primary emphasis lies on detecting run-time inference attacks. Potential adversaries can leverage known malware as a basis for crafting adversarial attacks, exploiting familiar attack vectors and tactics to evade detection. Attackers gather HPC features data from malware samples, following a process similar to training ML defenders. They manipulate the feature vectors to make them imperceptible to MLs, aiming to trick the models into classifying the adversarial samples as benign applications instead of recognizing them as malware. This aids attackers in launching successful malware attacks. In our threat model, it is assumed that attackers lack direct access to the ML defense models' training data and their parameters, ensuring they operate without privileged information about the defense mechanisms. Nevertheless, in the event of attackers attempting to steal or alter the ML defense models, we incorporate periodic ML model validation, described in Subsection 2.7. This process ensures that the core ML defense mechanisms remain unaltered, by malicious actors. Furthermore, during the inference stage, attackers possess the capability to access the inference data. These adversarial vectors can be plotted through updating firmware, system reboot with the uploaded malicious image to the device, and Man-In-The-Middle (MITM) attack. This access enables them to launch adversarial attacks by introducing modifications to the underlying HPC data. In this scenario, attackers do not directly modify the attack executable but instead manipulate the underlying features that pass through the anti-malware defense system during inference. The generated adversarial attacks retain the original malware executable but incorporate crafted values designed to evade detection. While previous research has primarily concentrated on attacking inference integrity [20, 22], this study aims to undermine the system's malware detection capabilities, thereby increasing the success rate of actual malware attacks.

2.3 Overview of the Proposed Methodology

Figure 1 presents an overview of our proposed adversarial-resilient framework for hardware malware detection. In Adversarial Attacks, depicted in the upper-left corner of the Figure, attackers can utilize

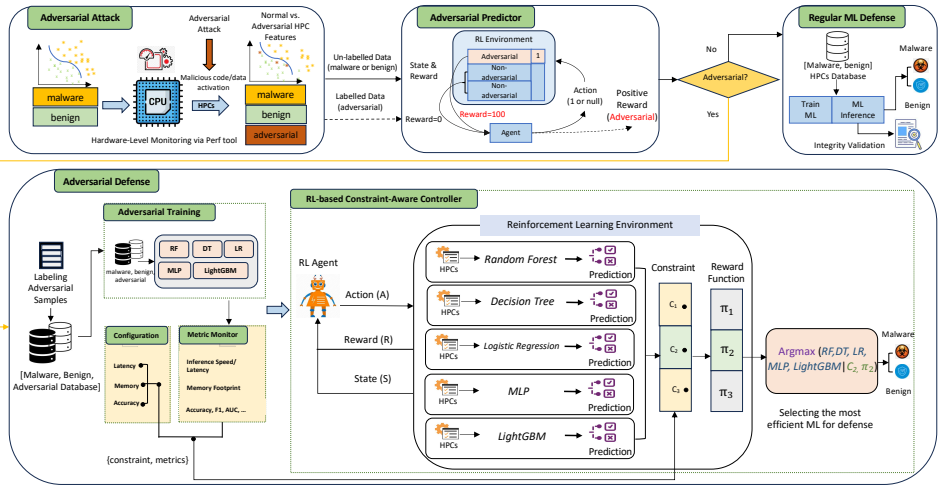

Figure 1: Overview of the proposed adversarial attack detection/prediction and defense framework for hardware malware detection.

diverse methods to present malicious firmware as a genuine update. Upon installation of this adversarial firmware onto a device, often without users' awareness, attackers gain the ability to execute attacks immediately, delay them, or trigger them based on specific conditions. During such attacks, malicious code is triggered, and instructions are made to rewrite the malware HPCs with the adversarial vectors contained in the data section of the firmware. Hence, during the inference stage, incoming HPC data can represent either malware, benign, or adversarial malware attacks. Initially, the system is equipped with legitimate malware and benign HPC data, and well-trained ML models as its primary defense mechanisms. However, when a hacker initiates an imperceptible adversarial attack, there is a high likelihood that it will evade detection by the ML-enabled malware defender, being misclassified as benign.

To effectively counter adversarial attacks, our method incorporates an Adversarial Predictor. This predictor assesses whether incoming data is adversarial, legitimate malware, or benign, leading to different defense strategies: 1) In the case of legitimate malware or benign behavior, the system employs pre-trained ML models to detect and defend against malware and benign data. 2) When the data is identified as adversarial, the system labels it as such (noting that adversarial malware exhibits distinct feature characteristics compared to legitimate malware, as explained in Section 2.4). The newly labeled adversarial data is then merged with the existing malware and benign data, as shown in the adversarial defense module (bottom left) in Figure 1. This merged HPC database $[Malware, Benign, Adversarial]$ is used to train both ML models and adversarial-resilient RL agents (constraint-aware controller). We trained five different ML models (Random Forest, Decision Tree, Logistic Regression, MLP, LightGBM) and one Neural Network (NN) (2 CONV and 3 FC layers), each offering distinct characteristics such as robust detection, speedy inference, and compact size. Furthermore, leveraging the merged adversarial data, we trained three RL agents using the Upper Confidence Bounds (UCB) algorithm. These RL agents (described in Section 2.6.1), specialize in various constraint-aware scenarios to enhance the system's efficiency.

2.4 Adversarial Attacks Generation

The definition of adversarial attacks sparks debate in the research community. Perspectives vary on their common properties, yet coherence, relevance, and imperceptibility compared to the original data garner consensus. For instance, the work in [3] defines adversarial examples as inputs that closely resemble natural inputs but are incorrectly classified, while [21] characterizes them as imperceptible, non-random perturbations. Extending the concept of imperceptibility to tabular data, [1] introduces the weighted l_p norm of the perturbation, considering feature importance. Notably, various features contribute differently to predictions, and the most critical features are subject to closer investigation by algorithms or human experts. Building upon [1]'s LowProFool method, we formalize the definition of generating adversarial attacks on tabular HPCs data for hardware malware detection as described below.

To generate an adversarial attack based on a feature matrix X containing individual incoming HPC data points (x_i), each associated with a true label $s = f(x)$, the objective is to introduce a perturbation r to x_i to create an adversarial instance $g(r)$, formulated as follows:

$$g(r) = L(x + r, t) + \lambda ||r \odot v||_p^2 \quad (1)$$

The generated adversarial sample, denoted as $g(r)$, consists of two components. The first component, $L(x + r, t)$, represents the loss value of the model, ensuring that the predicted label of the adversarial instance matches the intended target label, expressed as $f(x + r) = t$. The second component, $\lambda ||r \odot v||_p^2$, serves as a regularizer aimed at minimizing the perturbation r. Here, v is a vector representing feature importance, which can be computed using a feature importance algorithm. The parameter λ controls the weight assigned to the feature importance in the adversarial sample. This is crucial because adding significant perturbation to $g(r)$ can increase the likelihood of detection by human experts or ML models. Hence, the ultimate goal of $g(r)$ is to minimize perturbation values, enabling the generated adversaries to cross label boundaries while deceiving both human experts and ML models.

Algorithm 1 Adversarial Samples Generation

Input: malware HPCs (X_{mal} matrix) and set target label y_t is benign
Output: adversarial malware samples X_{adv}
- Clip X_{mal}'s min, max value of X_{mal}
- Train a Logistic Regression model on X_{mal} as imperceptibility evaluator
while *steps of generating adversaries* **do**
  Fit X_{mal}'s feature important $v = \{v_1, v_2, v_3, v_4\}$
  Calculate l_p norm loss and its gradient
  Generate perturbation and apply clipped min, max values
  Evaluate if $y_t == benign$ using LR model
  Assign the best imperceptible perturbation at each step
end
- Test X_{adv} by pre-trained LR model, output attack success rate.

Our proposed adversarial generation process is further outlined in Algorithm 1. We leveraged the Adversarial Robustness Toolbox (ART) [12], a Python-based machine learning security library. ART provides tools to assess, defend, certify, and verify machine learning models against various adversarial threats. We customized ART's LowProFool algorithm [1] and enhanced it with an imperceptibility evaluator using Logistic Regression (LR) from Scikit Learn. Once adversaries are generated, we use the trained LR model (on legitimate malware and benign data) to evaluate the adversarial success rate resulted in 100%. Specifically, we focused on modifying malware to deceive the system into classifying it as a benign application, thus increasing the likelihood of a successful attack.

2.5 Adversarial Attacks Prediction

2.5.1 RL with Unlabelled Data. Training an RL agent with unlabeled data addresses various challenges, including imitating learning and learning a reward function that discriminates specific data characteristics. In the context of hardware malware detection, where incoming data can be dynamic, encompassing adversarial attacks, legitimate malware, or benign samples, we focus on training an RL agent highly proficient in recognizing patterns associated with adversarial attacks. Previous studies have shown the initial effectiveness of using unlabeled data to train a discriminating RL agent. In particular, when unlabeled data exceeds labeled data in size, it substantially enhances the RL agent's ability to learn an effective reward policy for accurate predictions. Evaluation in this RL learning scenario is based on episode rewards rather than the conventional accumulated rewards. We developed a reinforcement learning-based approach applied to tabular-based hardware data derived from HPC registers of the underlying processor. Given the RL's heightened discriminative capacity when one dataset is more substantial than the other, and considering our limited adversarial data, we employed the existing limited adversarial data as labeled, while treating legitimate malware and benign data as unlabeled, assigning them a "None" label. During RL training, we assigned a high reward value (100) to labeled adversarial data, while unlabeled data received a reward of (0). Upon completion of training, the RL agent effectively learned to provide reward values, distinguishing between adversaries (labeled) and non-adversaries (unlabeled), aligning with our prediction requirements for incoming data.

2.5.2 Adversarial Predictor Training. We customize our RL environment using OpenAI's Gym baseline class [2], with state and action space defined as follows. We trained the adversarial predictor using an Advantage Actor Critic (A2C) algorithm [11] in TensorFlow, with both Actor and Critic using MLP with 4-hidden layers. A2C is a deep RL technique that combines RL and deep learning. Learning rates were set to 0.0005 for the Actor and 0.001 for the Critic. The Actor predicts an action ([1, nan]) after SoftMax activation, while the Critic assesses the quality (distance) of the predicted action from the Actor and computes a loss value using the Mean Square Error function. In our experiments, the adversarial predictor (agent) strives to classify incoming data as an adversarial attack (actions) to deceive the ML defender as effectively as possible by rendering the HPC data imperceptible (environment). We update the episode reward for each incoming data point, as independent events. This scenario conforms to a Markov Decision Process (MDP) consisting of states, actions, rewards, and a discount factor as stated below:

- *States:* 4-tuple representing the top four HPCs, whether from adversarial attacks, legitimate malware, or benign sources.
- *Actions:* adversarial attack or nan.
- *Rewards:* 100 for adversarial attack, 0 for nan.
- *Discount Factor:* Percentage of past experience (accumulated rewards) in the current decision (0.99 in our work).

2.6 Defending against Adversarial Attacks

The adversarial defense module detects malware attacks regardless of whether the data is adversarial or not. However, detecting malware is subject to resource constraints, especially in resource-limited devices. As depicted in Figure 1, our proposed approach addresses three primary constraints: accuracy in malware detection, inference latency, and memory footprint, offering an adversarial resilient and cost-aware malware detection solution.

2.6.1 Constraint-Aware Controller. To respond to pre-configured constraint requirements, we incorporate a constraint-aware controller into our proposed approach. The adversarial defense modules initially employ a merged dataset containing adversarial, malware, and benign samples to train a diverse set of classical ML models, each possessing distinct strengths and cost considerations. Some excel in high detection accuracy, while others prioritize faster inference or smaller model sizes. We develop the constraint-aware controller as an RL agent, allowing it to adapt dynamically to changing run-time variables like constraints, metrics, and incoming data from its environment. We select the Upper Confidence Bound RL as the learning algorithm due to its lightweight nature, imposing minimal overhead in terms of parameter size and inference latency for adaptive scheduling of the ML models at run-time. Furthermore, we evaluate all ML models on test sets and store their performance metrics in the Metric Monitor modules. These metrics are then passed to the reward function in the RL environment to guide in selecting the ML model that best meets the run-time configured constraints. Based on the constraint requirements, three types of RL agents are trained, each of which incorporates five classical ML models (excluding NN due to its low performance in adversarial learning) into its environment with the following specializations:
- *Agent 1:* Faster inference - Trained by a UCB agent to select the fastest inferring ML while ensuring high detection accuracy.
- *Agent 2:* Smaller memory footprint - Trained by a UCB agent to minimize usage while maintaining accurate predictions.
- *Agent 3:* Efficient malware detection - Trained by a UCB agent for accurate detection of adversarial and malware attacks, prioritizing models with low latency and memory usage as rewards.

2.6.2 RL-based Constraint Controller Training. In this RL decision-making process, each specialized agent aims to select the most optimized ML model from the five available in its environment. The goal is to predict whether incoming HPC data represents a malware attack (action) effectively. The decision criteria for choosing an ML model consider two factors: first, the model's ability to correctly detect malware attacks, and second, its ability to meet the configured constraints. This decision process aligns with the MDP:
- *States:* 14-tuple for the top four HPC features, the predictions of the five ML models for malware detection (including adversarial and regular attacks, and benign), and the passed constraints for the five ML models. This constraint serves as a run-time variable in the RL environment's reward function, forming a reward policy. The

RL agent interacts with these states, selects an action (which ML model to use), and learns a reward policy for maximum return with respect to the pre-configured performance constraint.
- *Actions:* 2-tuple for malware or benign selection.
- *Rewards:* A reward of 1 is assigned for correct predictions (malware or benign), while a reward of 0 is given for incorrect predictions.

2.7 ML Model Integrity Validation

Our framework is designed to protect the integrity and security of the deployed ML models against potential tampering by unauthorized entities. To achieve this, we employ offline hashing techniques to verify the stability of our ML defense models. Once the ML models are deployed for defense, we periodically generate hash values using the SHA-256 algorithm for the model path combined with deployment timestamps. These generated hashes are then compared against stored records to confirm the models' integrity. Moreover, we conduct regular assessments using a reserved offline validation set to evaluate the ongoing performance of our ML defense models. Metrics including accuracy, F1-score, True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR), and False Negative Rate (FNR) are systematically monitored. These metrics collectively provide a comprehensive view of the model's performance and serve as indicators of any potential modifications. If metrics deviate from our established records, indicating possible alterations to the models, immediate corrective action is taken. We restore the verified model and conduct a thorough system investigation to rectify any security breaches or unauthorized modifications.

3 EXPERIMENTAL RESULTS

Non-Adversarial Hardware Malware Detection. To examine the baseline hardware malware detection, various distinct branches of classical ML and NN models were implemented using legitimate malware and benign data captured from HPCs. This evaluation focuses on the models' ability to distinguish malware from benign data, as reflected in the results in Table 2. Notably, Random Forest and LightGBM models achieved F1-scores of 87% and 88%, respectively, while MLP and NN outperformed them with a 92% and 93% F1-score, respectively. In this context, MLP and NN stand out as the top-performing malware detectors. Additionally, it is observed that RF and LightGBM exhibit a 19% FNR, whereas MLP and NN boast a significantly lower 6% and 5% FNR. This highlights MLP's advantage in maintaining a secure operational environment by minimizing instances of wrongly classifying malware as benign. However, there is a trade-off, as MLP also incurs a slightly higher false positive rate (9%) compared to RF and LightGBM (4%), resulting in a marginally increased likelihood of false alarms.

Hardware Malware Detection under Adversarial Attacks. In this scenario, adversarial attacks are launched by applying imperceptible perturbations to HPC features, allowing malicious functions to masquerade as benign applications undetected by the ML models. The Adversarial Attack results in Table 2 show a significant deterioration in the ML models' detection performance across all metrics. This enhances the transferability of adversarial samples across all learning algorithms [13]. Furthermore, the downward blue arrow lines in Figure 2 illustrate a substantial downgrade in the system's malware detection capabilities, with RF decreasing by 71%, MLP by 34%, and LightGBM by 79% due to the impact of these

Table 2: Performance results for three scenarios: a) regular malware detection without adversarial attacks consideration, b) malware detection under adversarial attacks, and c) adversarial training, after detecting adversarial attacks.

Scenario	ML	ACC	F1	AUC	TPR	FPR	FNR	TNR
malware attack	RF	0.88	0.87	0.93	0.81	0.04	0.19	0.96
	DT	0.85	0.84	0.89	0.79	0.09	0.21	0.91
	LR	0.87	0.87	0.90	0.88	0.15	0.12	0.85
	MLP	0.92	0.92	0.94	0.94	0.09	0.06	0.91
	LightGBM	0.89	0.88	0.93	0.81	0.04	0.19	0.96
	NN	0.93	0.93	0.94	0.95	0.10	0.05	0.90
adversarial attack	RF	0.32	0.16	0.57	0.10	0.25	0.90	0.75
	DT	0.52	0.56	0.77	0.45	0.34	0.55	0.66
	LR	0.54	0.59	0.78	0.48	0.33	0.52	0.67
	MLP	0.53	0.58	0.77	0.49	0.40	0.51	0.60
	LightGBM	0.30	0.09	0.51	0.05	0.20	0.95	0.80
	NN	0.67	0.80	0.83	1.0	1.0	0.0	0.0
adversarial defense	RF	0.92	0.93	0.97	0.90	0.05	0.10	0.95
	DT	0.92	0.94	0.96	0.92	0.09	0.08	0.91
	LR	0.83	0.88	0.91	0.88	0.26	0.12	0.74
	MLP	0.95	0.96	0.97	0.97	0.10	0.03	0.90
	LightGBM	0.93	0.95	0.98	0.92	0.05	0.08	0.95
	NN	0.33	0.0	0.83	0.0	0.0	1.0	1.0

adversarial attacks in bypassing the detection mechanism. In addition, RF and LightGBM models exhibit a 90% and 95% false negative rate, while MLP demonstrates a false negative rate of 51%, mistakenly categorizing malware as benign. Conversely, their TPR are notably low. Depicted in Figure 3-(a) (yellow bars), MLP achieves a TPR of 49%, indicating a 51% chance of false alarms, while RF and LightGBM worsen with TPRs of 10% and 5%, respectively losing credibility as effective and reliable malware defenders. As indicated in Table 2, the NN trained on legitimate malware data misclassifies all data as suspicious malware, exhibiting an FPR of 100%. It struggles to distinguish between four numerical HPC values across malware, adversarial, and benign categories.

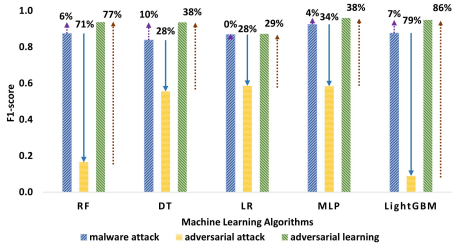

Figure 2: Adversarial attacks downgrade MLs' F1-score by up to 79%. Through adversarial training, enhanced MLs increase the F1-score over regular malware detection by up to 10%, and adversarial attacks by up to 86%.

Adversarial Predictor's Performance. The proposed RL-based adversarial predictor, acting as the initial line of defense, achieves a flawless 100% F1-score, accuracy, precision, and recall. Notably, the detection of adversarial attacks relies on feedback through the reward value rather than predictions from the DRL agent. The predictor is trained to discriminate between adversarial and non-adversarial HPC patterns, even when dealing with imbalanced and unlabeled data. During inference, it determines whether an input is adversarial if the feedback reward is positive or non-adversarial if it receives a zero reward. Figure 3-(b) illustrates the reward values in inference, where approximately 2,000 adversarial samples are followed by around 4,000 non-adversarial (malware, benign) samples.

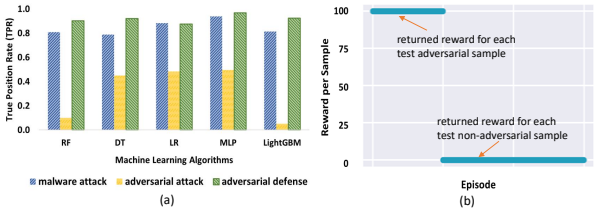

Figure 3: (a) True Positive Rate (TPR) drops during adversarial attacks (yellow bars) but improves with adversarial training (green bars) compared to regular malware attacks (blue bars), (b) Adversarial learning predictor distinguishes adversarial attack samples from non-adversarial ones via distinct feedback reward values.

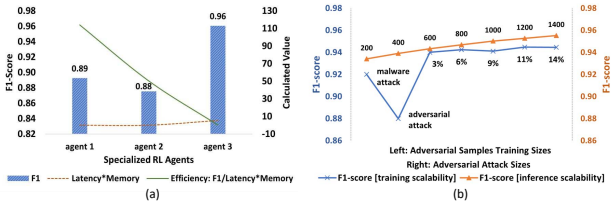

Figure 4: (a) Specialized RL agents adapt by selecting the best ML as a run-time defender based on predefined constraint requirements (detection rate (F1-score), Overhead: latency*memory, Efficiency Metric: F1/latency*memory), (b) Scalability analysis with varying sizes of adversarial samples in training and inference.

The DRL-based predictor adeptly discerns HPC patterns, assigns rewards, and offers insights on predicting adversarial attacks.

Adversarial Learning. As presented in Table 2, our adversarial defense strategy showcases enhanced detection performance through adversarial learning. Initially, the adversarial data is identified by the adversarial predictor (DRL agent) and then labeled before being merged with the existing legitimate malware and benign data. Using this combined database, various types of ML models are developed, resulting in an overall performance boost, as illustrated with the green bars in Figure 2. Notably, MLP shows a 4% increase in F1-score compared to regular malware detection without adversarial training (purple up-arrow lines on top of blue bars), while Decision Tree and Random Forest exhibit improvements of 10% and 6%, respectively. The findings underscore the effectiveness of adversarial training in enhancing the robustness of ML models. It is also observed that adversarial learning doesn't enhance the performance of NN in varied scenarios, reinforcing the ongoing debate about NN's effectiveness with tabular data [19] despite its excellence in processing images and sequential data.

Constraint Management for Adversarial Defense. Figure 4-(a) presents a performance and cost analysis of three constraint configuration options. Agent 1, optimized for cost-effectiveness, achieves a fair F1-score of 89% and an AUC of 96%. Agent 2, with relatively lower latency and memory footprint, serves as a moderate RL agent compared to agents 1 and 3. Agent 3 excels with an F1-score exceeding 96%, an AUC of 97%, 95% precision, and 97% recall, albeit with a slightly longer inference time (0.005 ms) and a larger memory footprint (1.06 MB). In contrast, Agent 1 boasts an extremely fast inference time of 0.0002 ms and a compact 47 KB memory size. Depending on the specific application requirements, Agent 1 is ideal for pre-configurations demanding swift inference and minimal memory usage while maintaining a dynamic adversarial defense (with an 89% detection rate). Agent 3's high detection performance is appealing for applications where a slightly longer inference time (around 0.005 ms) and a 1 MB model size are acceptable.

Scalability Analysis. We conducted a comprehensive scalability analysis of adversarial learning across both training and inference phases, illustrated in Figure 4-(b). In the training phase, our focus was on assessing the impact of varying sizes of adversarial attack data on inference detection performance. As shown in the blue line, the F1-score drops during adversarial attacks, followed by improved detection performance through adversarial training as the number of training samples increases. Initially, we observe improved detection performance as the number of adversarial samples for training increased. However, this enhancement plateaued as the attack scale remained fixed despite a growing pool of adversaries. The impact of adversarial sample size on larger-scale attacks warrants further investigation. In contrast, during the inference phase, our robust model, trained adversarially, displayed heightened resilience against increasing volumes of adversarial attacks (orange line). This trend not only underscores the effective scalability of our adversarial learning approach with growing data but also highlights its robustness when confronted with escalating adversarial challenges.

4 CONCLUSION

This study addresses adversarial attack challenges in Hardware Malware Detection (HMD), focusing on vulnerabilities in tabular data sourced from performance counter registers. We propose a proactive and multi-phased adversarial learning and defense framework based on Deep Reinforcement Learning (DRL). This framework encompasses adversarial attack generation, real-time prediction of attack patterns, and adversarial training as a defense for ML models. To ensure adaptability, we integrate a performance-aware constraint controller for dynamic allocation of defense responsibilities to specialized RL agents based on pre-defined performance constraints. The experiments highlight the impact of adversarial attacks causing a 79% decrease in MLs' F1-score, countered by adversarial training that elevates it by up to 86%. The results confirm the efficacy of the proposed approach, achieving up to 96.1% detection performance across various ML-based malware detectors.

5 ACKNOWLEDGMENT

This work is supported by the National Science Foundation under Award No. 2139034.

REFERENCES

[1] V. Ballet et al. 2019. Imperceptible Adversarial Attacks on Tabular Data. arXiv:1911.03274 [stat.ML]
[2] G. Brockman et al. 2016. OpenAI Gym. arXiv:1606.01540
[3] N. Carlini and D. Wagner. 2017. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. arXiv:1705.07263
[4] J. Demme et al. 2013. On the Feasibility of Online Malware Detection with Performance Counters. In *ISCA*. ACM, 559–570.
[5] A. Dhavlle et al. 2021. HMD-Hardener: Adversarially Robust and Efficient Hardware-Assisted Runtime Malware Detection. In *DATE*. 1769–1774.
[6] S. M. P. Dinakarrao et al. 2019. Adversarial Attack on Microarchitectural Events Based Malware Detectors. In *DAC*. ACM, Article 164, 1–6 pages.
[7] K. Grosse et al. 2017. Adversarial Examples for Malware Detection. In *ESORICS 2017*. Springer, Cham, 62–79.
[8] K. Khasawneh et al. 2017. RHMD: Evasion-resilient Hardware Malware Detectors. In *MICRO*. ACM, 315–327.
[9] A. P. Kuruvila et al. 2021. Defending Hardware-Based Malware Detectors Against Adversarial Attacks. *IEEE TCAD* 40, 9 (2021).
[10] Y. Li et al. 2020. A feature-vector generative adversarial network for evading PDF malware classifiers. *Information Sciences* 523 (2020), 38–48.
[11] V. Mnih et al. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *ICML*. 1928–1937.
[12] M. Nicolae et al. 2019. Adversarial Robustness Toolbox v1.0.0. arXiv:1807.01069
[13] N. Papernot et al. 2016. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. arXiv:1605.07277
[14] I. Rosenberg et al. 2018. Generic Black-Box End-to-End Attack Against State of the Art API Call Based Malware Classifiers. In *RAID*. Springer, Cham, 490–510.
[15] I. Rosenberg et al. 2020. Query-Efficient Black-Box Attack Against Sequence-Based Malware Classifiers. arXiv:1804.08778
[16] I. Rosenberg et al. 2021. Adversarial Machine Learning Attacks and Defense Methods in the Cyber Security Domain. arXiv:2007.02407
[17] B. C. Ross. 2014. Mutual Information between Discrete and Continuous Data Sets. *PLoS ONE* 9 (2014).
[18] H. Sayadi et al. 2018. Ensemble Learning for Effective Run-time Hardware-based Malware Detection: A Comprehensive Analysis and Classification. In *DAC*. 1–6.
[19] R. Shwartz-Ziv and A. Armon. 2021. Tabular Data: Deep Learning is Not All You Need. arXiv:2106.03253
[20] O. Suciu et al. 2018. When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks. In *USENIX Security*. 1299–1316.
[21] C. Szegedy et al. 2014. Intriguing Properties of Neural Networks. arXiv:1312.6199
[22] P. Xu et al. 2020. MANIS: Evading Malware Detection System on Graph Structure. In *SAC*. ACM, 1688–1695.