EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

Mitigating Adversarial Reconnaissance in IoT Anomaly Detection Systems: A Moving Target Defense Approach based on Reinforcement Learning

Arnold Osei[1], *Yaser Al Mtawa*[1], *and Talal Halabi*[2*]

[1]Department of Computer Science, University of Winnipeg, Winnipeg, MB, Canada
[2]Department of Computer Science, Laval University, Québec, QC, Canada

Abstract

The machine learning (ML) community has extensively studied adversarial threats on learning-based systems, emphasizing the need to address the potential compromise of anomaly-based intrusion detection systems (IDS) through adversarial attacks. On the other hand, investigating the use of moving target defense (MTD) mechanisms in Internet of Things (IoT) networks is ongoing research, with unfathomable potential to equip IoT devices and networks with the ability to fend off cyber attacks despite their computational deficiencies. In this paper, we propose a game-theoretic model of MTD to render the configuration and deployment of anomaly-based IDS more dynamic through diversification of feature training in order to minimize successful reconnaissance on ML-based IDS. We then solve the MTD problem using a reinforcement learning method to generate the optimal shifting policy within the network without a prior network transition model. The state-of-the-art ToN-IoT dataset is investigated for feasibility to implement the feature-based MTD approach. The overall performance of the proposed MTD-based IDS is compared to a conventional IDS by analyzing the accuracy curve for varying attacker success rates. Our approach has proven effective in increasing the resilience of the IDS against adversarial learning.

Received on 23 February 2024; accepted on 01 July 2024; published on 10 July 2024

Keywords: Adversarial Machine Learning, Anomaly detection systems, IoT security, Threat mitigation, Reinforcement Learning

doi:10.4108/eetiot.6574

1. Introduction

The Internet of Things (IoT) has become an intrinsic technology in various automated industries and large-scale smart city and government services including wearable health devices and autonomous transportation [1]. A major concern about IoT systems is security, which has become more difficult to implement in embedded IoT devices having limited computational resources to run firewalls and advanced cyber defense algorithms [2]. IoT devices and networks are known to be increasingly vulnerable to security attacks on data integrity and service availability [3, 4]. With increased connectivity, IoT devices can become compromised and used as zombies. Hackers can control these devices remotely and exploit them for illegal purposes like carrying out large-scale distributed denial of service (DDoS) attacks, e.g., via a Control & Command (C&C) server [5].

At the IoT network level, intrusion detection systems (IDS) constitute one of the essential defense tools that monitor the entire IoT network traffic [6]. As most IoT devices are deficient in computational and memory resources, most IoT networks rely on network-based IDS to provide security for the collective nodes within the network [7]. These IDS can either be signature-based or anomaly-based [8]. However, adversaries continue to devise new ways of orchestrating stealthy attacks that are usually cunningly evasive and could render signature-based detection defunct. This is why there have been growing research interests in building anomaly-based IDS to detect unknown attacks. These

*Corresponding author. Email: talal.halabi@ift.ulaval.ca

IDS often rely on machine learning (ML) algorithms to detect malicious behavior [9–11]. Nonetheless, this reliance adds to the IoT network's threat surface due to the presence of adversarial learning attacks.

Conventional IDS in IoT networks are as susceptible to attacks as the nodes within these networks. Little attention has been placed on protecting IDS, but the damage caused by a cyber attack that compromises an IDS could be far more detrimental than that caused by compromising nodes within the network, since the IDS's mission is to protect the network. An adversarial attack stealthily injects an input to an ML model that is purposely designed to cause the model to make a mistake in its predictions despite resembling a valid input when observed [12]. Adversarial attacks may belong to three categories: In white-box attacks, the adversary possesses knowledge of the training and testing datasets of a given IDS, what ML model and miscellaneous techniques are employed by the IDS, among other things. In gray-box attacks, the adversary only has partial knowledge of the network. Lastly, black-box attacks require that the adversary has no knowledge of the network, and launches the attack blindly and arbitrarily.

In this paper, we aim to proactively protect anomaly-based IDS in IoT from potential adversarial learning attacks using a novel Moving Target Defense (MTD) approach. Our goal is to prevent or mitigate the impact of successful reconnaissance within the system, where stealthy adversaries attempt to probe the ML-based IDS in order to collect useful information about its design and operation in order to eventually craft subverting attack patterns. Although researchers using MTD have made important strides within the community [13], no work has yet considered implementing MTD on IoT gateways involving IDS components. The contributions of this paper are described as follows:

- A novel, stochastic game-theoretic MTD model based on architecture decentralization and learning diversification to protect ML-based IDS in IoT networks against adversarial threats. The model decomposes the target system into a number of logical sub-components and apply MTD to mitigate the impact of the adversary's reconnaissance over time by expanding the exploration surface in terms of features.

- A reinforcement learning (RL) solution to autonomously optimize MTD timing and operations within the network in an adaptive fashion without prior knowledge of the adversarial behavior and environment model. The proposed RL algorithm learns the optimal defense policy via trial and error in order to avoid unnecessary configuration shifts.

- An extensive experimental validation of our proposed threat prevention solution using the state-of-the-art, real-world ToN-IoT dataset [14]. Our results are promising and demonstrate the effectiveness of the MTD-enabled IDS in reducing the impact of adversarial reconnaissance while maintaining high detection accuracy.

The remainder of the paper is structured as follows. Section 2 provides background information and discusses the related work. Section 3 describes the proposed MTD model. Section 4 presents our RL-based solution to the MTD timing problem. Section 5 describes our experiments and results. Finally, Section 2 concludes the paper.

2. Background Information and Related Work

2.1. Adversarial Learning Threats to IDS

The Fast Gradient Sign Method (FGSM) is an adversarial learning attack that uses the concept of gradient descent in neural networks, which is in turn an iterative optimization process to minimize the adversarial error during an attempt to compromise a given deep learning (DL) model [15]. This results in data points that look indistinguishable from the original ones but results in serious misclassification when passed through a DL model. FGSM is one of the most common adversarial learning attacks in the literature and is mostly used in image misclassification. The Barrage of Natural Transforms (BaRT) randomly sets up the classifier to be vulnerable to a number of transforms based on the criteria which the ML model uses in its prediction. Some transforms could be: Noise injection, FFT perturbation and Color precision reduction (for image classification) [16]. In a jacobian based saliency attack (JSMA), by analyzing the jacobian matrix of outputs with respect to inputs, one is able to deduce how the output probabilities behave given a slight modification of an input feature. In [17], JSMA was implemented against a multilayer perceptron model using the CIDS and TRAbID datasets for network traffic classification. All these adversarial threats can render the IoT network highly vulnerable to compromise by subverting the performance of the IDS.

Generally, the defense approaches against an adversarial learning attack on IDS can be grouped into reactive and proactive [18]:

- Reactive: Involves carrying out patches on a given network, based on the kind of adversarial attacks experienced. This could be done in an iterative manner to enhance the robustness of the IDS.

- Proactive: Involves altering the underlying architecture or learning procedure of the IDS, e.g., by adding more layers, training the detection model in real time with adversarial attack samples, or increasing the sensitivity of loss/activation functions. The loss function is given by: $f(x+\delta) = f(x)$, Where δ is the input injected by the adversary to cause perturbation and impact the prediction accuracy of the IDS.

To defend against FGSM, gradient masking (that naturally transforms a threat model from a white/gray box into a black box) is used to mask the model's output with respect to its input [19]. Adversarial training could also be used, where the IDS is trained with some adversarial examples to make it immune to adversarial attacks, but this can sometimes lead to label leakage and over-fitting as the adversarial examples generated during the training phase may not be present during the predictive/testing phase [20]. The authors in [21] came up with a novel model for adversarial training that involves feature scattering in a given latent space. They generate the feature-scattering adversarial examples in an unsupervised manner as a deliberate attempt to address possible label leakage. A novel generative adversarial network (GAN)-based adversarial defense method called Cowboy was proposed by [22]. This approach both detects and defends against adversarial attacks by using both the discriminator and generator of a typical GAN trained on the same dataset. The method is inspired by hypothesizing that adversarial samples ought to exist out of the data pipeline understudied by a GAN.

2.2. Moving Target Defense in IoT Systems

In contrast to reactive security approaches where a cyber attack may cause damage to the system before counteraction is initiated, MTD is a cyber defense paradigm that proposes to proactively make systems and networks dynamic to increase the difficulty for an attacker to be successful with exploits in the first place. With MTD, there is a full-on acknowledgment that systems are always going to be vulnerable against zero-day threats regardless of how many times the attack surface is shrunk through patching, because there is an unfair asymmetrical advantage that attackers have on static systems over time. Hence, the reconnaissance activities of the attacker on a static network would render the attacker in a favorable position to launch a successful attack [23]. It is also easier to create backdoors in traditional systems if system parameters are unchanging/static. Furthermore, resource-constrained devices and networks may not be able to add complex security set-ups and hence are most vulnerable to attacks. So far, MTD research in IoT networks is typically focused on the nodes, and is oblivious to the vulnerabilities of the IDS gateways through which all traffic converge and diverge, which can also be exploited by resourceful adversaries in order to penetrate the network.

MTD proposes the movement of system parameters at certain periods so that the state of the network at T_0 is different from the state of the network at some arbitrary period $T_0 + \delta t$, thereby making it difficult for the attacker to do any proper reconnaissance. The goal is not to reactively reduce the attack surface with traditional countermeasures like patches, but to proactively keep moving the attack surface to make it seemingly impossible or very difficult to be successful with an attack. The key design questions to consider while setting up an MTD-based network are: what to move, how to move it, and when to move it. With regards to what parameters to move/change, the moving parameter (MP) could be the data (e.g., formats), software, network (e.g., IP addresses, port numbers), platform (e.g., OS, firmware), runtime environment (e.g., RAM address space), or even hardware (e.g., routines in enterprise switch brands). With regards to how to move, the MP can be made to move from one configuration to another via randomized shuffling or using a predefined optimization algorithm to create diversification. Finally, the timing problem when implementing MTD involves determining the trigger to initiate the move in the network, which could be based on a specific time or event, or a combination of both.

In a given network, a typical use-case could be random re-assignment of IP addresses and port numbers. This can thwart the reconnaissance activities of the attacker. For example, using scanning tools like nmap will yield different results for each scan and will therefore not be useful knowledge to the attacker. Also, MTD in IoT networks may involve constantly changing the communication protocols between nodes and the gateway (e.g., WiFi, Bluetooth, Zigbee, etc.). Such diversification would make it difficult for an adversary to complete her exploit as each protocol is completely different from the other, with no correlation whatsoever.

Because of the uncertainty created by changing configurations on the network, it is possible to quantitatively attempt to describe the degree of uncertainty of a given network with a chosen MP by evaluating the number of states the MP is capable of taking on, and the probability that it takes on a certain state. This can be modeled using Shannon's entropy [24]. For a uniform probability distribution, Shannon's entropy will directly depend on the number of states available for a given MP [24]. This theoretical inclination further buttresses the following point: Instead of reducing the attack surface, MTD rather enlarges the ”exploration surface” domain for attackers, and then moves the attack surface as a sub-domain with the exploration domain, thereby making it difficult for attackers to orchestrate attacks. That said, even though more states of an MP translate to higher uncertainty and hence greater difficulty for the attacker, it could be much more difficult and costlier to have numerous states, depending on the MP in question. For example, it is easier for the defender to deploy multiple states of IP addresses compared to multiple firmware. This also means for an attacker, it may be easier to break through 254 different states of a node's IP address than 5 different states of a node's firmware. Hence there ought to be some qualitative representation of weighing the cost of states of a given MP, for both the attacker and the defender.

MTD is a relatively new area of research that is rapidly gaining momentum, especially for low-resource networks like IoT [25]. The authors in [26] implement an MTD model that involves shuffling proxies of which clients ought to connect to access system resources. The authors in [27] introduce a novel MTD against network reconnaissance. It is a software-defined technique called the Sniffer Reflector. This MTD architecture is basically set up to prevent successful network probing by the attacker by providing forged responses to network scans. The authors in [28] propose a novel MTD framework employing IoT-enabled data replication to replicate sensory and control signals in cyber-physical systems. This framework combines two layers of uncertainty, hence reducing the arbitrary attacker's ability to learn about the IoT network over time. It also reduces the impact of false data injection attacks on a given system model.

An MTD system typically requires a continuous adaptation of system configurations to be able to effectively hinder attacks, which can cause some overhead on the resources of a given system. On the other hand, limiting adaptations in a bid to reduce the overhead could make it much easier for attackers to successfully execute attacks. Therefore, determining the right time to make adaptations with the objective of minimizing long-term costs highlights the importance of the MTD timing problem. For instance, the authors in [29] propose a cost-aware MTD model to make smart and optimized adaptations by analyzing the trade-off between reducing system overhead and increasing the resilience of the system to attacks. Also, the authors in [30] propose a game-theoretic MTD model to address the timing problem when proactively facing DoS attacks. The model provided a guided framework to ensure that the defending system moved at on optimal time to yield the most resilience for the least cost on network performance.

3. The Proposed MTD-enabled IDS

Our MTD model against adversarial learning in IoT anomaly detection systems is mainly based on a novel feature shuffling mechanism that can be incorporated into the training phase of the ML-enabled IDS to shield it from reconnaissance. To provide the research community with a solid theoretical foundation that could become the basis for future works on MTD (e.g., using different MP), the designed shuffling mechanism fundamentally relies on a stochastic game-theoretic formulation between the IDS and the attacker, which we describe in detail in this section.

3.1. System Model

The proposed IDS architecture is based on logical decentralization and aggregation. This was inspired following the analysis of the ToN–IoT network dataset [14]. First, the 45 features of the ToN-IoT dataset were reduced to 15 prime features after conducting the dimensionality reduction technique of selecting the features that contribute the most variance in the dataset. Features with minuscule variance contributions were eliminated. This is done to reduce the overhead of applying our MTD approach. In our architecture, the typical IDS is logically split into i decentralized IDS components. Each IDS component is trained with a unique combination of n prominent features, and every combination is different for each IDS component. For each instance of traffic that goes through the IDS architecture, it is transmitted in parallel to all IDS components to be classified as either normal or malicious traffic based on the feature combinations that each IDS component was trained with. The classification outcome is then aggregated and a common classification result is chosen by virtue of a simple majority rule. Features are subsequently reshuffled during the trigger of the next shuffle iteration, which is dependent on the action taken by the IDS as part of the stochastic game.

The attacker is assumed to have unlimited resources for reconnaissance allowing her to constantly bombard IDS components with data instances as part of her adversarial probing goals (e.g., finding out what combination of features is being used by a given logical component). If one or more components within the architecture start to record detection rates α that are relatively lower than other components, this implies that it may have been compromised. When half or more IDS components have been deduced to be compromised, this triggers the IDS architecture to shuffle features and re-train its components. Training of the IDS is performed online when a shuffle is triggered. The IDS components need to collaborate with each other in other to classify a given traffic instance. Therefore, they ought to be trained with the same instances for meaningful results. The only difference is that each IDS component's classifier interprets a given traffic instance based on the feature combination assigned to it, but the traffic instances must be the same for all. The model may adopt a strict policy to ensure that the accuracy of the MTD-based IDS always exceeds a certain threshold. Hence, the combination of features could be constrained by the total accuracy criterion set. For example, if it is desired that after training and testing, the accuracy obtained from aggregating the results of each IDS component should not be less than 97% , then this obviously restricts the number of combinations possible - valid combinations. After reshuffling and training is complete, the IDS components are replaced in real time with the newly trained ones.

3.2. Game Characterization

Game Theory has been extensively used to evaluate situations where individuals have conflicting objectives. A game may be defined as a strategic interaction between two or more entities (players), which act in such a manner as to maximize their wins and minimize their losses (whether cooperatively or competitively) [31]. Pay-offs or Utilities are the quantifiable motivations that players get for executing corresponding actions. The characteristics of the proposed stochastic game can be described as follows.

- **States:** Let S_v be the set of all possible states in the game. The i IDS components constitute a state. They take on binary values: "1" to indicate that the IDS has been compromised, and "0" to indicate that the IDS component has not been compromised yet. The transition from one state to another, or choosing to remain in the same state, can be influenced by several factors: one or more IDS components within a conglomerate being supposedly compromised (by inferring from the relative detection rate), the defender just trying to maximize their rewards, etc.

- **Actions:** Let A_A be the set of all possible actions available to the attacker, and A_D be the set of all possible actions available to the defender. Naturally, the number of valid actions heavily depend on the operational goals of the game, for example, allowing only feature combinations that result in a certain accuracy percentage of the overall IDS. In our model, the agent's actions are selected to be between shuffling the IDS configurations or not. The attacker actions are selected to be between probing the IDS parameters or not.

- **Transition probabilities:** Let $P(s'|s, a_A, a_D)$ represent the transition probability from state s to state s' given the actions of the attacker (a_A) and defender (a_D). This captures the stochastic nature of the game, where the next state depends on the current state and the actions taken.

- **Payoff functions:** Let $R_A(s, a_A, a_D)$ represent the payoff received by the attacker when in state s and taking action a_A, given the defender's action a_D. Let $R_D(s, a_A, a_D)$ represent the payoff received by the defender when in state s and taking action a_D, given the attacker's action a_A.

- **Strategies:** Let $\pi_A(s)$ be the attacker's strategy, which determines the action a_A to be taken in state s. Let $\pi_D(s)$ be the defender's strategy, which determines the action a_D to be taken in state s.

- **Value functions:** Let $V_A(s)$ represent the expected cumulative payoff for the attacker starting from state s, considering the attacker's strategy π_A and the defender's strategy π_D. Let $V_D(s)$ represent the expected cumulative payoff for the defender starting from state s, considering the attacker's strategy π_A and the defender's strategy π_D.

Figure 1 depicts a scenario in our dynamic game, which is considered a semi-perfect information game. This is because the attacker is aware of when there is a change in state within the system. As compromising just one IDS component is not sufficient to launch an evasion attack (by virtue of the majority rule principle), the attacker is able to tell that her attack was impactful when the malicious payload goes through undetected (false negatives increase significantly because at a completely compromised state, the IDS is ineffective at detecting malicious traffic). After the termination of an episode due to the game being over, the IDS architecture is triggered to change the state by executing the "shuffle" action, and the attacker knows this because the false negatives would have tremendously dropped after retraining. The defender on the other hand is aware of the possible move of the attacker by virtue of the fact that there is a significant disparity in the relative detection rate among the IDS components.

The stochastic zero-sum game model formulation can be expressed as follows [32]:

$$V_A(s) = \max_{\pi_A(s)} \left[\min_{\pi_D(s)} \left\{ R_A(s, a_A, a_D) + \sum_{s'} P(s'|s, a_A, a_D) \cdot V_A(s') \right\} \right] \tag{1}$$

$$V_D(s) = \min_{\pi_A(s)} \left[\max_{\pi_D(s)} \left\{ R_D(s, a_A, a_D) + \sum_{s'} P(s'|s, a_A, a_D) \cdot V_D(s') \right\} \right] \tag{2}$$

where the players' value functions are recursively defined, considering the expected cumulative payoffs and the transition probabilities.

Any stochastic game-theoretic model can be further expanded to include reinforcement learning as a potential solution. This is because this ML paradigm is based on the trial-and-error approach, and so an agent and its environment can be modeled as playing repetitive games of trial-and-error until an optimal solution is found. If we are able to model our IDS architecture so that a given state provides information about how many IDS components have been compromised or not, and the transition to the next state is dependent on the attacker's success rate in compromising IDS components, assuming a perpetual attacker (an attacker constantly probing), and also that the transition to the next state is dependent on the defending system's choice of action (whether to stay in a state or shuffle), then the perpetual attacker's impact can be incorporated as part of the environment that the defending system (agent) has to learn from.

The majority rule criteria for the IDS architecture to maintain high classification accuracy implies that if more than half of the IDS components are compromised, then the game is over for the defender, and hence the termination of an episode. Hence, there would exist different ways that a game/episode could terminate, and hence reinforcement learning is used to ascertain the most optimal policy for the defender based on the testbed, as well sub-optimal strategies that could approach the optimal strategy.

3.3. Threat Model

As is the case with most ML-based systems, anomaly-based IDS are prone to adversarial attacks. The kind of adversarial attacks that the attacker is able to launch depends on the information that she is privy to about the system. Our MTD model defends against gray-box adversarial attacks and assumes that: 1) the attacker has knowledge of the entire feature space that the IDS architecture uses to train and test traffic (datasets); 2) the attacker has knowledge of the IDS architecture including its decentralization-aggregation approach; and finally 3) she is cognisant of the splitting number of prominent features among the IDS components. Nonetheless, the attacker is oblivious to the innerworking of the feature shuffling mechanism the IDS uses for the dissemination of features among the logical IDS components. Specifically, the attacker is unaware of the exact feature combination per IDS component at any given time. Consequently, if this is figured out by the attacker, she is able to inject successful adversarial examples into that IDS component. Also, if the attacker had knowledge of the feature shuffling mechanism, his information about the entire system would be complete and hence this would have been a white-box attack model. Therefore, on the spectrum of gray-box attack scenarios, this is the worst-case scenario from the defender's perspective. This is particularly important because in assuming that the attacker has enough knowledge to compromise a given network, we are able to address most of the loopholes within a network for worst-case eventualities which rarely happen, but are very possible.

Typically, with unrestricted access to the feature space of the network traffic of the IDS architecture, the two categories of adversarial injection attacks that could be launched are: Data poisoning and evasion attacks. The former is implemented during the training phase of the IDS, while the latter is usually executed in the testing phase of the IDS (i.e., traffic transmission in real time). There has been significant progress in protecting systems against data poisoning attacks [33]. Hence, we only focus on evasion attacks due to their potential stealth. An evasion attack typically involves manipulating a given instance of traffic so that it is misclassified by the IDS. Ideally, because the attacker has knowledge of what features the IDS uses for classification, this attack should be easy to execute over time by constantly probing the IDS components. However, the dynamic shuffling of unique combination of features among IDS components makes it difficult for the attacker to know what combination of features are used for a given IDS component at a given time t, rendering the information gathered by the attacker unworthy of being used. The goal of MTD in any case is to stretch the time t that an attacker requires to compromise a network as practically as possible - bearing in mind the cost involved in terms of system resources and performance degradation [34]. Hence, allowing both the defender and the omnipotent attacker to play the *game of features* provides insights on how long the game can go on before the entire IDS is compromised.

3.4. Uncertainty Analysis of our MTD Approach

The feature space diversification of the theoretic model highlights the different possible feature combinations of the IDS components. For i components of a given IDS architecture, each component is represented by a bit. Assuming 5 IDS components, the IDS architecture is represented by [00000] bits. The flip of a bit from 0 to 1 denotes the compromise of an IDS component. The termination of a single iteration of the game is triggered when more than half the IDS components ($i/2$) have 1 as their bit values. It is important to re-emphasize that the objective of MTD is to create as much uncertainty in the system as possible, so as to make it difficult for an arbitrary attacker to successfully complete the reconnaissance phase and launch an attack over time.

Let $X = i/2$. The diversification of the proposed IDS architecture can be measured by the sum of combinations of all possible bits that can take the value 0 before termination at $X = i/2$. This is bounded by:

$$C_i^i + C_{i-1}^i + C_{i-2}^i + C_{i-3}^i + C_{i-4}^i + ... + C_{i-X}^i$$

where

$$C_X^i = \frac{n!}{X!(i-X)!} \tag{3}$$

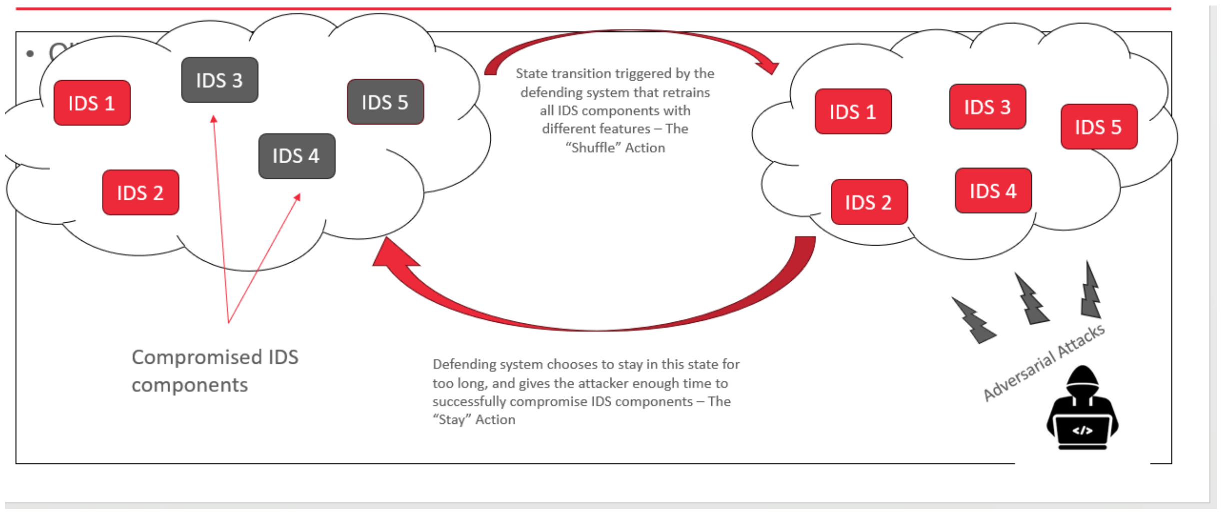

IDS 1
IDS 3
IDS 5
IDS 4
IDS 2
State transition triggered by the defending system that retrains all IDS components with different features – The "Shuffle" Action
IDS 1
IDS 3
IDS 5
IDS 2
IDS 4
Compromised IDS components
Defending system chooses to stay in this state for too long, and gives the attacker enough time to successfully compromise IDS components – The "Stay" Action
Adversarial Attacks

Figure 1. Diagram showing dynamic game scenario.

On the other hand, the diversification based on feature combinations (MP) is given by:
C_f^n If f denotes the number of unique features per IDS component, the total diversification of the system is thus given by the product of the two expressions as follows:

$$(C_i^i + C_{i-1}^i + C_{i-2}^i + C_{i-3}^i + ... + C_{i-X}^i) \times C_f^n$$

Entropy is a measure of the randomness or uncertainty within the system. The correlation between entropy and uncertainty implies that the greater the entropy within the system, the higher the uncertainty and difficulty for the attacker to successfully gather reconnaissance and launch an attack [24] [23]. The entropy $H(X)$ is given by [35]:

$$H(x) = -\sum_{x \in X} p(x) log(p(x)) \tag{4}$$

Based on the diversification expressions above, the maximum probability that the theory stipulates for which the IDS architecture will not be compromised is given by:

$$(C_i^i(\frac{1}{2})^i(\frac{1}{2})^0 + C_{i-1}^i(\frac{1}{2})^{i-1}(\frac{1}{2})^1 + C_{i-2}^i(\frac{1}{2})^{i-2}(\frac{1}{2})^2 + C_{i-3}^i(\frac{1}{2})^{i-3}(\frac{1}{2})^3 + ... + C_{i-X}^i(\frac{1}{2})^{i-x}(\frac{1}{2})^x) \times \left(C_f^n(\frac{1}{n})^3\right)$$

The $\frac{1}{2}$ probability of the IDS components represents a uniform probability of a bit either being 0 or 1. The $\frac{1}{n}$ probability of the feature set represents a uniform probability of training a given IDS component from an n feature space. The entropy then is given as:

$$H(x) = -\sum_{j=0}^{x} \left(C_{i-j}^i \left(\frac{1}{2}\right)^j \left(\frac{1}{2}\right)^{i-j} \times C_f^n(\frac{1}{n})^f\right) \times \log_2 \left(C_{i-j}^i \left(\frac{1}{2}\right)^j \left(\frac{1}{2}\right)^{i-j} \times C_f^n(\frac{1}{n})^f\right) \tag{5}$$

Based on these theoretical expressions, the following can be deduced: 1) The measure of uncertainty can be increased by increasing the features space n from which IDS components can be trained; 2) The measure of uncertainty can be increased by reducing the number of feature combinations f that each IDS component is trained with, given a feature-space pool n (this obviously would have an impact on the accuracy of each IDS component because the less features one trains an IDS with, the less information it has to properly classify a given data instance); and 3) The measure of uncertainty can be increased by increasing the number of IDS components i in a given IDS architecture (but this will incur more computational overhead). In all cases, it is preferred to use entropy as the ultimate measure of uncertainty because it provides a tighter and systematic way of looking at which parameters matter the most when it comes to increasing the uncertainty within the system [23].

4. Reinforcement Learning for Solving the MTD Timing Problem

In RL, the agent undertakes a sequence of actions that yield feedback signals from the environment in the form of rewards or punishments. The agent learns over time based on the actions taken and the rewards/punishments received, and the environment evolves based on the agent's actions. The environment state transitions can be seen as stochastic sequences or

Markov decision processes [36]. The agent's ultimate task is to find an optimal policy that yields the most rewards after experiencing several episodes of the RL game [37]. In our MTD model, we propose a new reward function for the agent which depends on the number of compromised components i_c and the number of shuffled components $i_s = i - i_c$. It also depends on the action taken (whether "shuffle" or "stay"), and if the action is taken at a time when the number of compromised IDS components i_c is below or above the threshold $i/2$. We setup our reward function according to algorithm 1, where the parameters a, b, c, and d are used to tune the reward estimation be the system administrator according to the importance of actions. Based on this model, the defender has two (2) actions: Stay or Shuffle. The "Stay" action indicates a decision by the defender to remain in the same state based on pre-defined objectives of the model. For example: Reward threshold, number of compromised components, etc. The "Shuffle" action indicates a decision by the defender to change state. This also means the "Shuffle" action would involve different feature combinations and how that affects metrics like the overall performance of the IDS architecture.

The agent's decision to trigger a shuffle or not would either result in an episodic game ending quickly or going on for as long as possible, provided the agent chooses to shuffle before at least half of the IDS components within the architecture are compromised. However, choosing to shuffle continuously results in computational and memory costs on the IDS architecture due to retraining IDS components with different feature combinations iteratively for every trigger move. Ultimately, the agent would want to

```
Input  : $i_c, i$
Output: reward
Function compute_reward():
    reward ← $i - i_c$;
    if agent_action == 1 and $i_c \leq i/2$ then
        cost ← $a * i_c$;
        return (reward - cost);
    end
    else if agent_action == 1 and $i_c > i/2$ then
        cost ← $b * i_c$;
        return (reward - cost);
    end
    else if agent_action == 0 and $i_c \leq i/2$ then
        cost ← $c * i_c$;
        return (reward - cost);
    end
    else if agent_action == 0 and $i_c > i/2$ then
        cost ← $d * i_c$;
        return (reward - cost);
    end
```

Algorithm 1: The reward function of the agent

learn about the environment over time for a given number of episodes and total corresponding rewards yielded based on the actions taken, and to find the closest optimal policy or sub-optimal policy so as to determine when to initiate a move. The model is set up so that the attacker constantly attempts to launch a successful evasion attack. This would naturally consist of sub-actions of feature combinations. For the sake of simplicity of the model, the attacker's action is to constantly attempt to evade, and is incorporated into the environment the defense agent has to learn. Given an arbitrary pool of prominent features f_p to shuffle from, if each IDS component was shuffled with f features, then the number of combinations possible for each IDS component from the pool is given by: $C_f^{f_p}$. To represent each IDS components' combination by using bits, we obtain $\log_2(C_f^{f_p})$ bits. This relation indicates that each IDS component's feature combination is represented as an x-bit binary number, which is extremely important in the implementation phase, where it is much easier and more convenient to put an abstraction on feature combinations with binary representations. Hence, an evasion attack is deemed successful if for any arbitrary $\log_2(C_{f_{split}}^{f})$-bit feature combination of each IDS component, the attacker's feature combination matches.

The Q-learning algorithm adopted to solve the MTD timing problem is shown in algorithm 2. Given a policy function $\pi(s)$, which indicates a set of "action-paths" to follow as a function of the present state s, $V_\pi(s)$ denotes the expected utility at a given state and $U(R, \gamma)$ is the sum of discounted rewards (where γ is the discount factor and is equal to 1 if the reward function is not affected by the circumstances of the future) [38]. The RL problem is formulated as follows:

$$Q(s, a) = \mathbb{E}[R(s, a, s') + \gamma V_\pi(s')] \tag{6}$$

$$V_\pi(s) = max_a Q(s, a) \tag{7}$$

$$\pi(s, a) = argmax_a Q(s, a) \tag{8}$$

$Q_\pi(s, a)$ is referred to as the Q-value at a chance node, which is the quality of a state action-pair. Essentially, the optimal value at any state s would be the Q-value that is maximized over a set of actions. The optimal policy $\pi(s, a)$ is the state-action combinational subset that maximizes the quality function (Q-value), as seen in algorithm 2. The bell's optimality equations are feasible solutions only when there is a working model for the environment. This is where model-free RL solutions come in handy. We leverage the Q-learning technique which has information about which state-action in any given policy yields the most rewards. At each step k, there is a re-evaluation of the value of a state (or the quality of a state-action pair Q) until

```
Input   : state space, action space, Q table,
          $\alpha$, $\gamma$, $\epsilon$, num-episodes,
          max-steps-per-episode, iteration-count
Output: old-state, new-state, agent-action,
          reward, accuracy
$state\_space \leftarrow 32$;
$action\_space \leftarrow 2$;
$q\_table \leftarrow$ np.zeros($state\_space, action\_space$);
$\alpha \leftarrow 0.5$;
$\gamma \leftarrow 0.9$;
$\epsilon \leftarrow 0.1$;
$num\_episodes \leftarrow 100$;
$max\_steps\_per\_episode \leftarrow 2500$;
$iteration\_count \leftarrow \{\}$;
$k \leftarrow 0$;
for $episode$ to $num\_episodes$ do
    $observation\_action\_pair \leftarrow []$;
    $observation\_action\_observation \leftarrow []$;
    $Game.reset(restart = True)$;
    $state \leftarrow$ binary_to_decimal(old_state) $-1$;
    for $step$ to $max\_steps\_per\_episode$ do
        if $random.uniform() < \epsilon$ then
            $agent\_action \leftarrow$ agent_action_space.sample();
        else
            $agent\_action \leftarrow$ argmax($q\_table[state]$);
        end
        $observation, agent\_action \leftarrow$ Game.step
          (Game.agent_action_space.sample());
        $pair \leftarrow old\_state, agent\_action$;
        $observation\_action\_pair$.append($pair$);
        $triplet \leftarrow$
          $old\_state, agent\_action, new\_state, reward, acc$;

        $observation\_action\_observation$.append($triplet$);
          $state \leftarrow binary\_to\_decimal(observation) - 1$;
        $next\_state \leftarrow binary\_to\_decimal(new\_state) - 1$;
        $td\_error \leftarrow reward +$
          $\gamma \cdot$ (np.max($q\_table[next\_state]$) $-$
          $q\_table[state][agent\_action]$);
        $q\_table[state][agent\_action] \leftarrow$
          $q\_table[state][agent\_action] + \alpha \cdot td\_error$;
          $state \leftarrow next\_state$;
        if $Game.done1$ then
            $iteration\_count[k+1] \leftarrow$
              $observation\_action\_observation$;
            $k \leftarrow k+1$;
            break;
        end
    end
end
```

Algorithm 2: Q-learning process in our MTD model

termination.

$$Q^{new}(s_k, a_k) = Q^{old}(s_k, a_k) + \alpha[r_k + \gamma max_a Q(s_{k+1}, a) - Q^{old}(s_k, a_k)], \quad \forall k \in [1...K] \tag{9}$$

where K is the total number of steps. Q-learning is an off-policy RL technique, which implies that the quality of the next state-action pair is explorative, and will capture even sub-optimal Q-values in an attempt to find the optimal policy - the argmax that maximizes the Q-value for the next state. Its off-policy quality makes it possible to keep track of almost all episodes that lead to termination. There is a tunable parameter ϵ to add some randomness in exploring sub-optimal policies. In algorithm 2, the Q-table is updated iteratively, and the agent's choice of action gains optimality over time.

5. Experiments and Results

5.1. Experimental Setup

We validate our MTD solution using the state-of-the-art IoT security dataset called ToN-IoT, which was generated by simulating MQTT-based traffic [39] - an MQTT machine-to-machine network architecture of 12 sensors, a broker, a camera, and an attacker. Also, five scenarios of data records were retrieved: Normal operation, aggressive scan, UDP scan, Sparta SSH brute-force attack, and MQTT brute-force attack. Three abstraction levels of features are also employed in the composition of this dataset (after extraction from raw pcap files): packet features, unidirectional flow features, and bidirectional flow features. The MTD parameter n was chosen to be 3 based on the analysis performed on the ToN-IoT dataset. It was found that for an IDS component to be trained on a combination of prominent features that have enough variance for decent classification of traffic instances, they ought to be trained with at least three features.

The anomaly-based IDS relies primarily on a ML classifier. In our experiment, we set-up a random forest classifier consisting of ten decision trees as our ML model for the IDS. The random forest is a meta estimator that understudies and fits a given dataset into two or more decision trees, and uses the average classification of each decision tree for a more accurate and robust classification. Our experimental MTD-enabled IDS consists of 5 different random forest classifiers (logical IDS components). After feature selection was conducted on the ToN-IoT dataset, the chosen features were recorded in a mutable list (called the pool of features). During the training phase of the IDS components, each was randomly trained on three unique combinations of features from that feature pool. To compromise a given IDS component, we use the

CleverHans library for our adversarial learning attack implementation. It is an open source python library for conducting adversarial attacks on ML models. Adversarial examples were generated by feeding the test traffic through the cleverhans.torch.attack.noise class, which generated the adversarial examples for the ToN-IoT test data samples by injecting adversarial noise that led to the misclassification of data instances.

From the python repository, the Scikit-Learn library was used to implement the random forest classifier model. The Pandas library was used for reading the ToN-IoT dataset as a data frame into the working environment, and then for the various manipulations of the data frame. The Numpy library was used for handling data fragments as arrays, tinkering with them, and re-writing them back on to the data frame for further analysis. The OpenAI gym library was used to provide the baseline environment for implementing the RL algorithm and incorporating the incessant adversarial attacks as part of the the agent's environment. The machine used for the experiment is an Intel64 Family 6 Model 165 Stepping 2 GenuineIntel Processor with 7,968 MB of RAM.

The experimentation steps can be summarized as follows. First, we set up and train two separate IDS: A conventional IDS and the MTD-based IDS, each using the random forest ML model as the classifier, using the ToN-IoT dataset as the traffic specimen. Next, we measure and record the accuracy test traffic going through the conventional IDS. Using the cleverhans adversarial framework, we inject adversarial noise into the test traffic and measure the accuracy drop, then we do this for increasing odds of success of the attack launch (i.e., $0.5, 0.6, 0.8, 1$). For each adversarial attack attempt on the MTD-based IDS, the RL algorithm learns the corresponding reward obtained progressively over time and influences in real time what action the agent takes over the course of a number of episodic runs. The odds of the attacker are increased accordingly from 50% *to* 100%. The accuracy at each of these odds is recorded, as well as the CPU and memory usage amidst adversarial attacks and MTD deployment.

5.2. Results Analysis

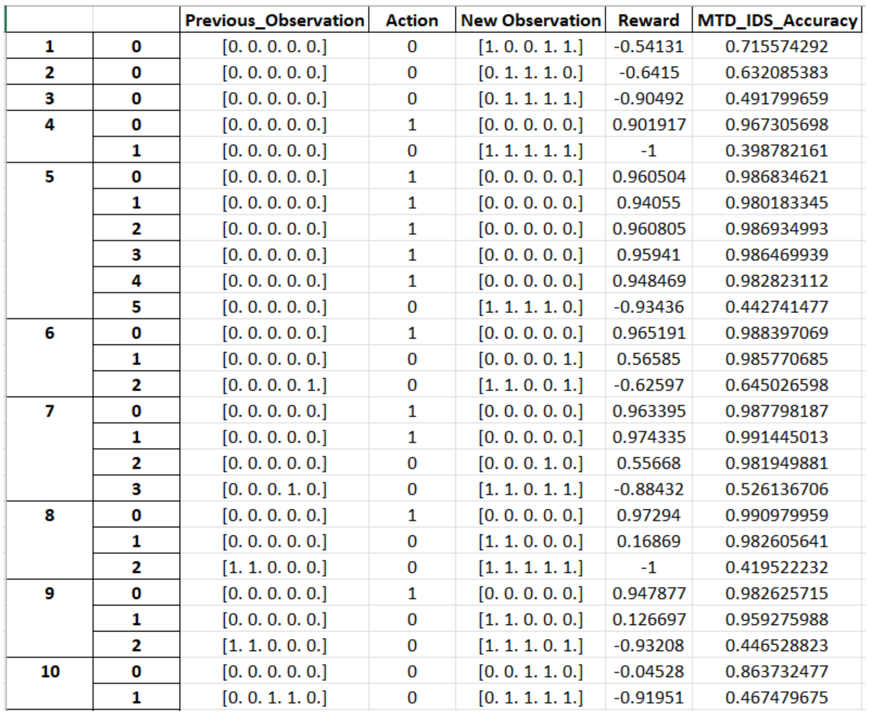
As seen in Figure 2, the simulation testbed for the MTD-based IDS is set up so that the IDS defending agent goes through various scenarios of which an episode could terminate, and the associated reward is obtained after an episodic termination. The attacker's incessant adversarial attacks are incorporated as part of the environment and the testbed also records in real time how the overall accuracy of the MTD-based IDS is doing for each episodic step. Resetting to a new episode or triggering the "action" results in the retraining of the 5 IDS components (indexed $0 - 4$). Figure 3 provides a microscopic view into what goes on within an episodic step: The combination of features used to train a specific IDS component, what each IDS component's accuracy is after training, and the combined IDS accuracy after training (with or without an adversarial attack).

One of the biggest concerns of having an MTD-based system is coming up with a systematic scheme to know when to initiate an adaptation in an optimized or quazi-optimized manner. Here, we leveraged RL to guide the behavior of the agent and allow it to learn in real time what actions to take and when to take them to maximize gain and minimize cost. Figure 5 shows the cumulative reward of the MTD-based IDS agent over 250 episodes. The results show the agent's reward was increasing over the number of episodes, which is a strong indication that learning was taking place.

Figure 4 shows the overall accuracy comparison of the conventional and MTD-based IDS for various attack success rates. The various success rates of the attacker were simulated using the python library Numpy's uniform probability distribution (using the same seed value for both the conventional and MTD-based IDS for objectivity and neutrality), and changing the odds incrementally. For example, as can be seen from the figure, at 50 % odds of success, the attacker has an equal chance of success and failure for both the conventional and MTD-based IDS. In this case, there was no trigger for an adversarial attack, hence the reason the conventional IDS has an accuracy of close to 100 %. Neither was attacked at this success rate, however the slight drop in the accuracy of the MTD-based IDS is due to the implementation of MTD (the decentralization and aggregation in the MTD architecture). As the attacker's success keeps increasing however, it can be seen that the accuracy degrades significantly for the conventional IDS, whereas for the MTD-based IDS, it degrades much less comparatively.

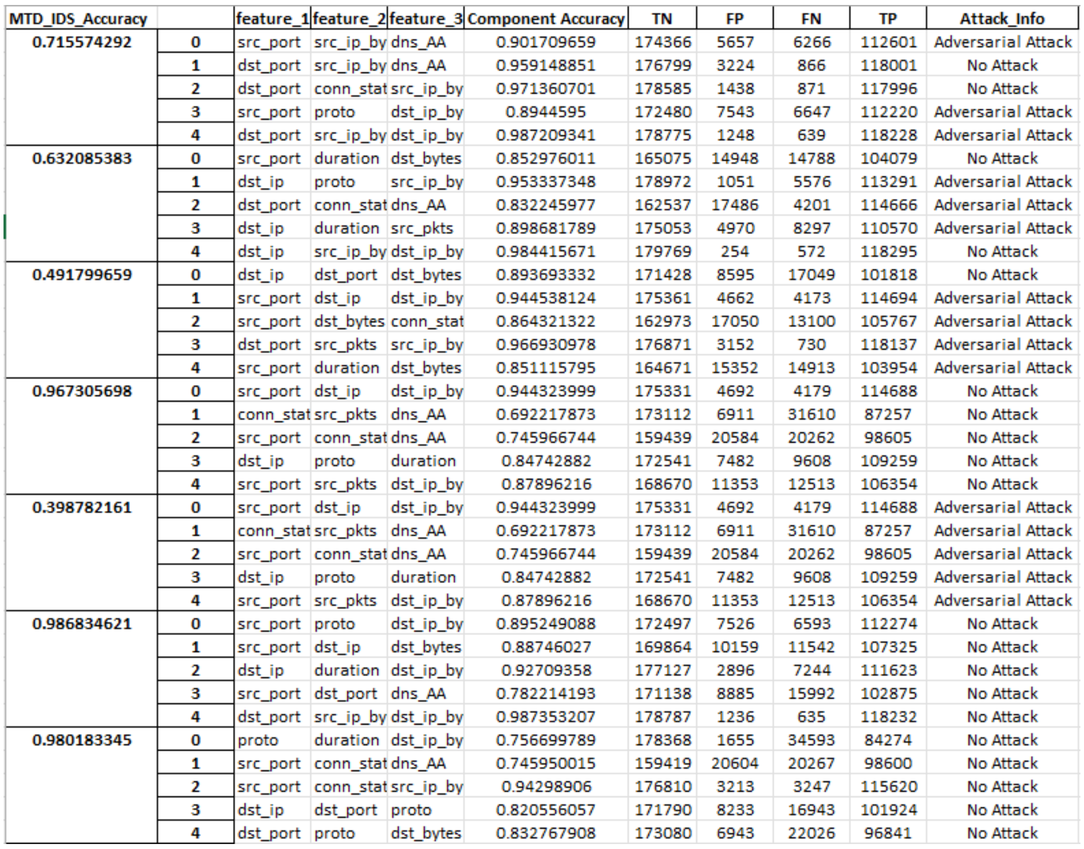
Figure 5 shows the progression of the agent's reward duting learning. Figure 6 shows a positive correlation between the reward accrued and the overall accuracy of the MTD-based IDS. This also informs us that the MTD-enabled IDS agent was truly learning to optimize and maximize its overall accuracy. It must be clarified though that the system's inherent costs were not incorporated into the reward function of the agent during the RL training phase. The agent's reward was simply a function of the number of compromised IDS components and number of shuffled IDS components, as ratios of the total number of IDS components within the system. In our future work, we intend to include the network cost within the reward function of the agent.

In Figure 7, we show the CPU usage of the MTD-based IDS in comparison to the conventional IDS for different success rates of the attacker. As can be deduced, the constraints on CPU resources increase for every increase in the adversarial attack success



		Previous_Observation	Action	New Observation	Reward	MTD_IDS_Accuracy
1	**0**	[0. 0. 0. 0. 0.]	0	[1. 0. 0. 1. 1.]	-0.54131	0.715574292
2	**0**	[0. 0. 0. 0. 0.]	0	[0. 1. 1. 1. 0.]	-0.6415	0.632085383
3	**0**	[0. 0. 0. 0. 0.]	0	[0. 1. 1. 1. 1.]	-0.90492	0.491799659
4	**0**	[0. 0. 0. 0. 0.]	1	[0. 0. 0. 0. 0.]	0.901917	0.967305698
	1	[0. 0. 0. 0. 0.]	0	[1. 1. 1. 1. 1.]	-1	0.398782161
5	**0**	[0. 0. 0. 0. 0.]	1	[0. 0. 0. 0. 0.]	0.960504	0.986834621
	1	[0. 0. 0. 0. 0.]	1	[0. 0. 0. 0. 0.]	0.94055	0.980183345
	2	[0. 0. 0. 0. 0.]	1	[0. 0. 0. 0. 0.]	0.960805	0.986934993
	3	[0. 0. 0. 0. 0.]	1	[0. 0. 0. 0. 0.]	0.95941	0.986469939
	4	[0. 0. 0. 0. 0.]	1	[0. 0. 0. 0. 0.]	0.948469	0.982823112
	5	[0. 0. 0. 0. 0.]	0	[1. 1. 1. 1. 0.]	-0.93436	0.442741477
6	**0**	[0. 0. 0. 0. 0.]	1	[0. 0. 0. 0. 0.]	0.965191	0.988397069
	1	[0. 0. 0. 0. 0.]	0	[0. 0. 0. 0. 1.]	0.56585	0.985770685
	2	[0. 0. 0. 0. 1.]	0	[1. 1. 0. 0. 1.]	-0.62597	0.645026598
7	**0**	[0. 0. 0. 0. 0.]	1	[0. 0. 0. 0. 0.]	0.963395	0.987798187
	1	[0. 0. 0. 0. 0.]	1	[0. 0. 0. 0. 0.]	0.974335	0.991445013
	2	[0. 0. 0. 0. 0.]	0	[0. 0. 0. 1. 0.]	0.55668	0.981949881
	3	[0. 0. 0. 1. 0.]	0	[1. 1. 0. 1. 1.]	-0.88432	0.526136706
8	**0**	[0. 0. 0. 0. 0.]	1	[0. 0. 0. 0. 0.]	0.97294	0.990979959
	1	[0. 0. 0. 0. 0.]	0	[1. 1. 0. 0. 0.]	0.16869	0.982605641
	2	[1. 1. 0. 0. 0.]	0	[1. 1. 1. 1. 1.]	-1	0.419522232
9	**0**	[0. 0. 0. 0. 0.]	1	[0. 0. 0. 0. 0.]	0.947877	0.982625715
	1	[0. 0. 0. 0. 0.]	0	[1. 1. 0. 0. 0.]	0.126697	0.959275988
	2	[1. 1. 0. 0. 0.]	0	[1. 1. 1. 0. 1.]	-0.93208	0.446528823
10	**0**	[0. 0. 0. 0. 0.]	0	[0. 0. 1. 1. 0.]	-0.04528	0.863732477
	1	[0. 0. 1. 1. 0.]	0	[0. 1. 1. 1. 1.]	-0.91951	0.467479675

Figure 2. The first 10 episodes of the game played by the MTD-based IDS agent, and the corresponding reward and accuracy obtained in each episode.



MTD_IDS_Accuracy		feature_1	feature_2	feature_3	Component Accuracy	TN	FP	FN	TP	Attack_Info
0.715574292	**0**	src_port	src_ip_by	dns_AA	0.901709659	174366	5657	6266	112601	Adversarial Attack
	1	dst_port	src_ip_by	dns_AA	0.959148851	176799	3224	866	118001	No Attack
	2	dst_port	conn_stat	src_ip_by	0.971360701	178585	1438	871	117996	No Attack
	3	src_port	proto	dst_ip_by	0.8944595	172480	7543	6647	112220	Adversarial Attack
	4	dst_port	src_ip_by	dst_ip_by	0.987209341	178775	1248	639	118228	Adversarial Attack
0.632085383	**0**	src_port	duration	dst_bytes	0.852976011	165075	14948	14788	104079	No Attack
	1	dst_ip	proto	src_ip_by	0.953337348	178972	1051	5576	113291	Adversarial Attack
	2	dst_port	conn_stat	dns_AA	0.832245977	162537	17486	4201	114666	Adversarial Attack
	3	dst_ip	duration	src_pkts	0.898681789	175053	4970	8297	110570	Adversarial Attack
	4	dst_ip	src_ip_by	dst_ip_by	0.984415671	179769	254	572	118295	No Attack
0.491799659	**0**	dst_ip	dst_port	dst_bytes	0.893693332	171428	8595	17049	101818	No Attack
	1	src_port	dst_ip	dst_ip_by	0.944538124	175361	4662	4173	114694	Adversarial Attack
	2	src_port	dst_bytes	conn_stat	0.864321322	162973	17050	13100	105767	Adversarial Attack
	3	dst_port	src_pkts	src_ip_by	0.966930978	176871	3152	730	118137	Adversarial Attack
	4	src_port	duration	dst_bytes	0.851115795	164671	15352	14913	103954	Adversarial Attack
0.967305698	**0**	src_port	dst_ip	dst_ip_by	0.944323999	175331	4692	4179	114688	No Attack
	1	conn_stat	src_pkts	dns_AA	0.692217873	173112	6911	31610	87257	No Attack
	2	src_port	conn_stat	dns_AA	0.745966744	159439	20584	20262	98605	No Attack
	3	dst_ip	proto	duration	0.84742882	172541	7482	9608	109259	No Attack
	4	src_port	src_pkts	dst_ip_by	0.87896216	168670	11353	12513	106354	No Attack
0.398782161	**0**	src_port	dst_ip	dst_ip_by	0.944323999	175331	4692	4179	114688	Adversarial Attack
	1	conn_stat	src_pkts	dns_AA	0.692217873	173112	6911	31610	87257	Adversarial Attack
	2	src_port	conn_stat	dns_AA	0.745966744	159439	20584	20262	98605	Adversarial Attack
	3	dst_ip	proto	duration	0.84742882	172541	7482	9608	109259	Adversarial Attack
	4	src_port	src_pkts	dst_ip_by	0.87896216	168670	11353	12513	106354	Adversarial Attack
0.986834621	**0**	src_port	proto	dst_ip_by	0.895249088	172497	7526	6593	112274	No Attack
	1	src_port	dst_ip	dst_bytes	0.88746027	169864	10159	11542	107325	No Attack
	2	dst_ip	duration	dst_ip_by	0.92709358	177127	2896	7244	111623	No Attack
	3	src_port	dst_port	dns_AA	0.782214193	171138	8885	15992	102875	No Attack
	4	dst_port	src_ip_by	dst_ip_by	0.987353207	178787	1236	635	118232	No Attack
0.980183345	**0**	proto	duration	dst_ip_by	0.756699789	178368	1655	34593	84274	No Attack
	1	src_port	conn_stat	dns_AA	0.745950015	159419	20604	20267	98600	No Attack
	2	src_port	conn_stat	src_ip_by	0.94298906	176810	3213	3247	115620	No Attack
	3	dst_ip	dst_port	proto	0.820556057	171790	8233	16943	101924	No Attack
	4	dst_port	proto	dst_bytes	0.832767908	173080	6943	22026	96841	No Attack

Figure 3. Microscopic view into the first 8 episodes (indexed by their accuracy values) showing the feature combinations each IDS component was trained with.

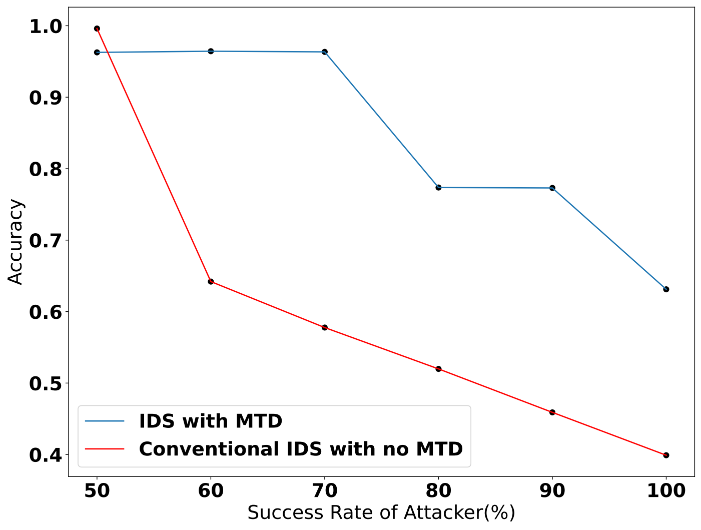


Figure 4. Accuracy of both IDS architectures for varying attack success rates.

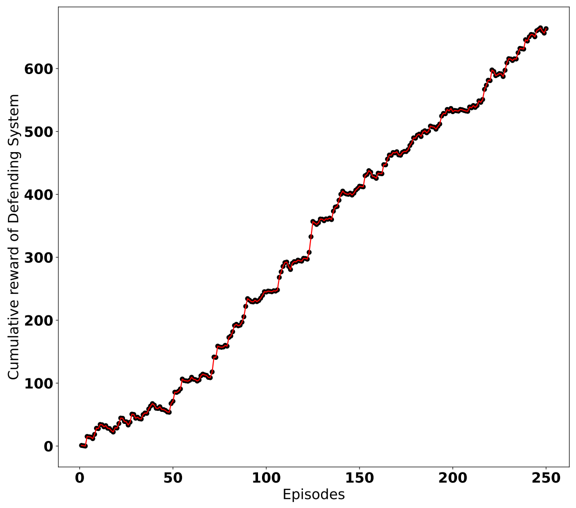


Figure 5. Cumulative reward of the agent during the learning process (250 episodes).

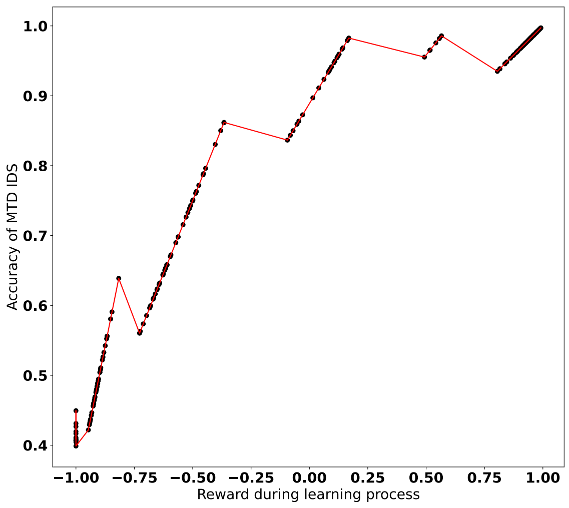


Figure 6. Accuracy vs reward of the MTD-based IDS agent during the game.

rate. This is more so for the MTD-based IDS because it requires shuffling of features and re-training of IDS components to mitigate attacks. It was observed that the biggest contributing factor to the high CPU usage in the MTD-based IDS is the re-training of IDS components every time an MTD move is triggered. The conventional IDS still incurs more CPU usage because of the increase in the number of successful adversarial probing by the attacker. Memory usage also went up for the MTD-based IDS for different attacker's success rates, as seen in Figure 8. This was not surprising, given the complexity of the MTD architecture that was implemented. The memory usage cost incurred by the conventional IDS as a result of more frequent adversarial activity pales in comparison to the cost incurred by the MTD-based IDS for all instances, and justifiably so. Nonetheless, the resource drainage from CPU and memory usage were not infused into the reward function of the learning agent and may have influenced the reason for the wide gaps in the curves shown.

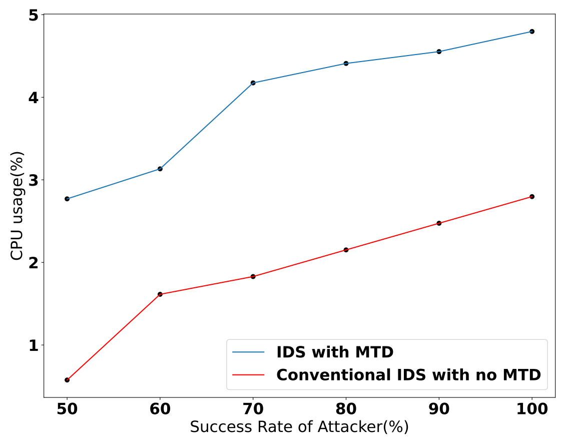


Figure 7. CPU usage of both the MTD-based IDS and conventional IDS for varying success rates of the attacker.

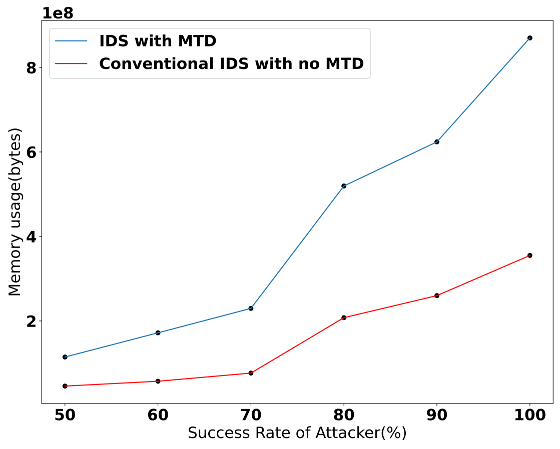


Figure 8. Memory usage of both the MTD-based IDS and conventional IDS for varying success rates of the attacker.

6. Conclusion

Anomaly-based IDS in IoT networks mostly rely on machine learning, and hence are vulnerable to adversarial threats. In this paper, we designed a novel MTD solution based on feature shuffling to allow the IDS to counter stealthy reconnaissance efforts and prevent successful evasion attacks. Our solution is based on a solid game-theoretic problem formulation between the defense system and the adversary, which we solve using reinforcement learning to orchestrate the configuration shifts in an adaptive fashion. The results show that our MTD-enabled IDS is more resilient to adversarial attacks. The impact of our findings is indisputable and will undoubtedly contribute significantly to the MTD literature. We plan to replicate the experiments in a real-world IoT testbed to achieve stronger validation on various datasets and investigate other potential moving parameters that may allow us to expand the exploration surface of the IDS.

Acknowledgement

This research is partially supported by the Natural Sciences & Engineering Research Council of Canada (NSERC) [Grant no. RGPIN-2021-04342].

References

[1] IHS, S. (2018) Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 (in billions) .

[2] Makhdoom, I., Abolhasan, M., Lipman, J., Liu, R.P. and Ni, W. (2018) Anatomy of threats to the internet of things. *IEEE Communications Surveys & Tutorials* **21**(2): 1636–1675.

[3] Giraldo, J., Urbina, D., Cardenas, A., Valente, J., Faisal, M., Ruths, J., Tippenhauer, N.O. *et al.* (2018) A survey of physics-based attack detection in cyber-physical systems. *ACM Computing Surveys (CSUR)* **51**(4): 1–36.

[4] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z. *et al.* (2017) Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)*: 1093–1110.

[5] Şendroiu, A. and Diaconescu, V. (2018) Hide'n'seek: an adaptive peer-to-peer iot botnet. *architecture* **3**: 5.

[6] Santos, L., Rabadao, C. and Gonçalves, R. (2018) Intrusion detection systems in internet of things: A literature review. In *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)* (IEEE): 1–7.

[7] Elrawy, M.F., Awad, A.I. and Hamed, H.F. (2018) Intrusion detection systems for iot-based smart environments: a survey. *Journal of Cloud Computing* **7**(1): 1–20.

[8] Zarpelão, B.B., Miani, R.S., Kawakani, C.T. and de Alvarenga, S.C. (2017) A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications* **84**: 25–37.

[9] Al-Garadi, M.A., Mohamed, A., Al-Ali, A.K., Du, X., Ali, I. and Guizani, M. (2020) A survey of machine and deep learning methods for internet of things (iot) security. *IEEE Communications Surveys & Tutorials* **22**(3): 1646–1685.

[10] Alsoufi, M.A., Razak, S., Siraj, M.M., Nafea, I., Ghaleb, F.A., Saeed, F. and Nasser, M. (2021) Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review. *Applied sciences* **11**(18): 8383.

[11] Patel, V., Choe, S. and Halabi, T. (2020) Predicting future malware attacks on cloud systems using machine learning. In *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)* (IEEE): 151–156.

[12] Vorobeychik, Y., Kantarcioglu, M., Brachman, R., Stone, P. and Rossi, F. (2018) *Adversarial machine learning*, **12** (Springer).

[13] Sengupta, S., Chowdhary, A., Sabur, A., Alshamrani, A., Huang, D. and Kambhampati, S. (2020) A survey of moving target defenses for network security. *IEEE Communications Surveys & Tutorials* **22**(3): 1909–1941.

[14] Booij, T.M., Chiscop, I., Meeuwissen, E., Moustafa, N. and den Hartog, F.T. (2021) Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets. *IEEE Internet of Things Journal* **9**(1): 485–496.

[15] Huang, S., Papernot, N., Goodfellow, I., Duan, Y. and Abbeel, P. (2017) Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* .

[16] Mahmood, K., Gurevin, D., van Dijk, M. and Nguyen, P.H. (2021) Beware the black-box: On the robustness of recent defenses to adversarial examples. *Entropy* **23**(10): 1359.

[17] Ayub, M.A., Johnson, W.A., Talbert, D.A. and Siraj, A. (2020) Model evasion attack on intrusion detection systems using adversarial machine learning. In *2020 54th annual conference on information sciences and systems (CISS)* (IEEE): 1–6.

[18] Zhong, Y., Zhu, Y., Wang, Z., Yin, X., Shi, X. and Li, K. (2020) An adversarial learning model for intrusion detection in real complex network environments. In *Wireless Algorithms, Systems, and Applications: 15th International Conference, WASA 2020, Qingdao, China, September 13–15, 2020, Proceedings, Part I 15* (Springer): 794–806.

[19] Lee, H., Bae, H. and Yoon, S. (2020) Gradient masking of label smoothing in adversarial robustness. *IEEE Access* **9**: 6453–6464.

[20] Xu, H., Ma, Y., Liu, H.C., Deb, D., Liu, H., Tang, J.L. and Jain, A.K. (2020) Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing* **17**: 151–178.

[21] Zhang, H. and Wang, J. (2019) Defense against adversarial attacks using feature scattering-based adversarial training. *Advances in Neural Information Processing Systems* **32**.

[22] Santhanam, G.K. and Grnarova, P. (2018) Defending against adversarial attacks by leveraging an entire gan. *arXiv preprint arXiv:1805.10652* .

[23] Navas, R.E., Cuppens, F., Cuppens, N.B., Toutain, L. and Papadopoulos, G.Z. (2020) Mtd, where art thou? a systematic review of moving target defense techniques for iot. *IEEE internet of things journal* **8**(10): 7818–7832.

[24] Zhuang, R., DeLoach, S.A. and Ou, X. (2014) Towards a theory of moving target defense. In *Proceedings of the first ACM workshop on moving target defense*: 31–40.

[25] Mercado-Velázquez, A.A., Escamilla-Ambrosio, P.J. and Ortiz-Rodriguez, F. (2021) A moving target defense strategy for internet of things cybersecurity. *IEEE Access* **9**: 118406–118418.

[26] Jia, Q., Sun, K. and Stavrou, A. (2013) Motag: Moving target defense against internet denial of service attacks. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)* (IEEE): 1–9.

[27] Wang, L. and Wu, D. (2016) Moving target defense against network reconnaissance with software defined networking. In *Information Security: 19th International Conference, ISC 2016, Honolulu, HI, USA, September 3-6, 2016. Proceedings 19* (Springer): 203–217.

[28] Giraldo, J.A., El Hariri, M. and Parvania, M. (2022) Moving target defense for cyber–physical systems using iot-enabled data replication. *IEEE Internet of Things Journal* **9**(15): 13223–13232.

[29] Wang, H., Li, F. and Chen, S. (2016) Towards cost-effective moving target defense against ddos and covert channel attacks. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*: 15–25.

[30] Osei, A.B., Yeginati, S.R., Al Mtawa, Y. and Halabi, T. (2022) Optimized moving target defense against ddos attacks in iot networks: When to adapt? In *GLOBECOM 2022-2022 IEEE Global Communications Conference* (IEEE): 2782–2787.

[31] Von Neumann, J. and Morgenstern, O. (2007) Theory of games and economic behavior. In *Theory of games and economic behavior* (Princeton university press).

[32] Shoham, Y. and Leyton-Brown, K. (2008) *Multiagent systems: Algorithmic, game-theoretic, and logical foundations* (Cambridge University Press).

[33] Chen, H. and Koushanfar, F. (2023) Tutorial: Toward robust deep learning against poisoning attacks. *ACM Transactions on Embedded Computing Systems* **22**(3): 1–15.

[34] Evans, C. and Hamkins, J.D. (2013) Transfinite game values in infinite chess. *arXiv preprint arXiv:1302.4377* .

[35] Carter, T. (2007) An introduction to information theory and entropy. *Complex systems summer school, Santa Fe* .

[36] Li, Y. (2017) Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274* .

[37] Ding, G., Aghli, S., Heckman, C. and Chen, L. (2018) Game-theoretic cooperative lane changing using data-driven models. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE): 3640–3647.

[38] Brunton, S.L. and Kutz, J.N. (2022) *Data-driven science and engineering: Machine learning, dynamical systems, and control* (Cambridge University Press).

[39] Hindy, H., Tachtatzis, C., Atkinson, R., Bayne, E. and Bellekens, X. (2020) Mqtt-iot-ids2020: Mqtt internet of things intrusion detection dataset. *IEEE Dataport* .