GENG5512 Engineering Research Project Part 2

Using Artificial Intelligence to Automate the Deployment of Moving Target Defence Operations

Joo Kai Tay
22489437
School of Engineering, University of Western Australia

Supervisor: Dr. Jin Hong
Department of Computer Science and Software Engineering,
University of Western Australia

Word count: 8000

School of Engineering
University of Western Australia

Submitted: 12 OCTOBER 2024

DECLARATION

In accordance with University Policy, I certify that:

The attached work submitted for assessment is my own work and that all material drawn from other sources has been fully acknowledged and referenced.

Use of AI tools
Students are permitted to use AI tools for general research purposes and to check and improve the quality of written English in their report. Students are not permitted to use AI tools to generate content.

I have used AI tools in the preparation of this report: **Yes**/No (choose one)

If yes, then provide details:
Use of AI tools to check English.

Signed [illegible] Date 12/10/2024

Abstract

Cybersecurity is becoming increasingly critical with the reliance on digital operations in various sectors, making robust cybersecurity measures essential. Traditional cybersecurity methods, focus on static defences that can be ineffective against new or evolving threats. Moving Target Defence (MTD) is a proactive approach to cybersecurity that continuously changes system aspects to make it harder for attackers to gain a foothold. This project leverages machine learning to automate the deployment of MTD operations, enhancing adaptability in cybersecurity strategies.

This project explored the use of machine learning, specifically a Double Deep Q-Network (DQN), to create a system, MTDShield, that can dynamically identify and deploy optimal MTD techniques based on network security posture metrics and attacker information.

The key contributions of this research are:

- Identification and definition of network security posture metrics to inform machine learning models for improved MTD deployment decision-making.
- Introduction of a framework leveraging machine learning models to dynamically identify and deploy optimal MTD techniques. The MTDShield module consists of a feature extraction module, time series analysis module, feature fusion module, and a Q-Network output module to process network security metrics and select an appropriate MTD technique.
- Conducted experiments to evaluate the impact of key hyperparameters such as gamma, epsilon and train start on the performance of the model. Additional experiments were performed to assess the impact of IDS sensitivity on model performance and the contribution of individual modules within the MTDShield system.

The results presented in this research can be used by organisations to add an extra layer of security to their existing MTD techniques, further safeguarding their systems.

Acknowledgements

I’m extremely grateful to my supervisor, Dr Jin Hong for his invaluable patience, feedback and guidance throughout this process. I am also grateful to my course mates for feedback they have provided as I was developing my work.

Lastly, I would be remiss in not mentioning my family for keeping my spirits high and motivated throughout this final year of my studies.

Table of Contents

List of Figures

1 Introduction

Cybersecurity is a preeminent issue in today's interconnected world, with the prevalence of digital operations as the backbone of multiple sectors ranging from banking, transport and health, cybersecurity is of paramount importance [1]. While these industries reap the benefits of evolving technology, the sophistication of cyber threats is also on the rise. Countering these threats require the use of advanced cybersecurity measures to safeguard sensitive data and ensure uninterrupted service delivery [2]. Effective cybersecurity practices not only shield organisations from financial and reputational damage but also bolster trust among customers and stakeholders. Furthermore, regulatory compliance with data protection laws necessitates stringent cybersecurity protocols to avoid legal repercussions [3].

Traditional methods, such as firewalls, IDS and antivirus software, focus on strengthening a static defence to prevent known threats. However, these can be less effective against new or evolving threats that exploit previously unknown vulnerabilities [4]. This is where Moving Target Defence (MTD) comes into play. Unlike static defences, which maintain a consistent configuration and attack surface, MTD continuously changes system aspects such as IP addresses, operating systems and application configurations [4]. This dynamic approach increases the complexity and cost for attackers, as the continuously evolving environment disrupts their reconnaissance efforts and disrupt chained attacks. By preventing attackers from gaining a foothold, MTD enhances overall system security and resilience. MTD's dynamic nature means that even if an attacker discovers a vulnerability, it may no longer be relevant or accessible by the time they attempt to exploit it. This fundamental difference makes MTD particularly valuable in environments where security needs to constantly evolve in response to emerging threats [5].

However, current approaches towards MTD predominantly focuses on optimising specific MTD techniques tailored to specific types of attackers or dedicated MTD methodologies. This specialised concentration often overlooks the broader applicability and integration of MTD strategies across varied threat landscapes and defensive scenarios. As a result, the research tends to delineate narrowly defined attacker models or individual MTD tactics, without addressing the potential for comprehensive or unified defence frameworks that consider multiple strategies that could be taken by attackers and defenders [4]. This approach can inadvertently limit the understanding of MTD's full capabilities and its effectiveness against diverse or evolving cyber threats, thereby constraining the development of more versatile and adaptive cybersecurity solutions.

Machine learning is increasingly used in decision-making through analysing large amounts of data to identify patterns and trends. By feeding a model new data, it enables it to continuously learn, allowing decision-making processes to adapt and improve over time, allowing for reactive outcomes. This project will leverage reinforcement learning models to identify the best combinations of MTD techniques to deploy at strategic moments, based on network security posture metrics and attacker information. To build models capable of predicting the optimal MTD deployment and type, extensive research will first focus on identifying the right metrics for assessing network security posture. Following this, model development and fine-tuning will be essential to ensure that the security framework's is responsive and adaptable but also maximises resource efficiency by deploying defences precisely when and where they are most needed.

The outcomes of this research can provide valuable guidance to organisations seeking to secure their networks by adding an extra layer to traditional MTD techniques, further safeguarding their systems. Specifically, the primary contributions of this work include:

- Identifies and defines network security posture metrics that will inform the machine learning models, improving decision-making for MTD deployment.
- Introduces the application of machine learning models to dynamically identify and deploy optimal MTD techniques, enhancing adaptability in cybersecurity strategies.
- Develops a reinforcement learning model that addresses a diverse range of cyber threats, and can evolve to meet new challenges, expanding beyond specialised MTD approaches.
- Demonstrates how continuous system changes through MTD disrupt attackers' reconnaissance and exploitation efforts, reducing the risk of advanced persistent threats (APTs).

2 Background

MTD techniques are about proactively preventing cyberattacks by manipulating the attack surface of the systems involved. Therefore, when designing a MTD system, the three main design choices revolve around the questions: what to move, how to move and when to move [4].

2.1 What to Move

Changing the specific system configuration attributes, also known as the attack surface, is what the question "What to move" seeks to address. These attack surfaces that can be dynamically altered to thwart attackers include various system or network properties such as IP addresses, port numbers, operating systems or software versions [4]. These changes can occur in the application, OS, VM or hardware layer [6]. To consider "What to move" successful, changes in these configuration attributes must alter the attack surface significantly, causing difficulties and increased effort for attackers [7].

2.2 How to Move

The term "How to move" is integral to the three operational components of MTD classification: shuffling, diversity, and redundancy (SDR). Common techniques used in MTD include artificial diversity and randomisation that either rearrange or randomise various system and network attributes [8, 9]. Each technique can be categorised under shuffling, diversity, or redundancy [4].

2.3 When to Move

The concept of "When to move" in a MTD system focuses on determining the optimal timing for transitioning from the current system state to a new one, thus rendering any information or progress obtained by an attacker in the current state obsolete. There are three primary adaptation strategies for implementing these changes:

- **Reactive Adaptation**: Reactive MTD is designed to counteract an attacker's moves once their actions have been identified, aiming to neutralise any advantage they may have gained [4].
- **Proactive Adaptation**: This approach involves scheduled changes to system configurations or properties, executed at fixed or random intervals [4]. The goal is to ensure that any intelligence gathered by an attacker becomes outdated rapidly, necessitating regular updates to maintain security [10].
- **Hybrid Adaptation**: Combining elements of both reactive and proactive strategies, hybrid adaptations adjust the timing of MTD operations based on specific events or security alerts, while also maintaining a maximum interval between changes to guard against unnoticed threats [11].

A fixed interval approach provides a more proactive defence mechanism, potentially leading to higher security but can be expensive. An optimal MTD interval, therefore, needs to be dynamically determined during runtime to balance cost-effectiveness with effective security measures.

3 Literature Review

Section 3 introduces key concepts related to the integration of multiple MTD techniques and their impact on network security. Section 3.1 explores the challenges and strategies for combining MTD methods, Section 3.2 investigates the role of network security metrics in evaluating defence effectiveness, and Section 3.3 investigates the application of reinforcement learning to enhance MTD strategies. By examining the combined effects of MTD on both attack and defence metrics, and leveraging deep reinforcement learning for adaptive decision-making, this section provides a comprehensive overview of how MTD can be optimised to improve network resilience against evolving threats.

3.1 MTD Combination Techniques

When combining multiple MTD techniques, it’s essential to consider:

- The specific threat being addressed
- The combined impact on network security metrics
- The potential effect on system performance

While matching MTD techniques to specific threats seems ideal, it’s not practical. This approach assumes complete knowledge of attackers’ methods and doesn’t account for new attack strategies. Hong et al. [12] suggested that rather than focusing on countering attacks, MTD should aim to increase the attack efforts to deter attacks on the network in question.

A study by Alavizadeh et al. [13] assessed the effectiveness of combined MTD strategies in a cloud environment using the Hierarchical Attack Representation Models (HARM) framework. They found that combining shuffling, diversity, and redundancy techniques improved all examined security metrics (Redundancy, Return on Attack, Attack Cost, and System Availability) compared to using a single technique. However, this study only considered the attack related metrics as discussed in section 3.2. The deployment of multiple MTD techniques might increase resource consumption, potentially degrading the system's overall performance and adversely affecting the defence related metrics.

3.2 Network Security Metrics

Having a grasp on network security metrics is crucial for assessing the effectiveness and robustness of security policies, detecting vulnerabilities, and ensuring systems are compliant with any regulations in their field. These metrics provide a quantitative basis for understanding the security posture of a network, guiding security investments, and measuring the impact of security improvements. Good security metrics need to be specific, measurable, attainable, repeatable and time dependent [14].

Behi et al. [14] conducted an experimental study to quantify network security. They collected data on various security metrics, calculated a security score for each machine, and then ranked the metrics based on their impact on network security. The most important metrics were system updates, followed by web browser version and OS version.

Another metric used to evaluate network security is the Common Vulnerability Scoring System (CVSS). The CVSS offers a systematic method for evaluating the severity of security vulnerabilities, focusing on their characteristics and resultant effects. The CVSS framework comprises base, temporal, and environmental metrics. As delineated in the CVSS Version 2.0, the base metrics include factors such as Access Vector, Access Complexity, Authentication, Confidentiality, Integrity, and Availability [15]. These metrics provide fundamental insights into the intrinsic attributes of a vulnerability, assessing exploitability, impact, and the potential extent of damage upon exploitation.

Yusuf et al. [16] proposed the Temporal-Hierarchical Attack Representation Model (T-HARM) model to capture network changes and investigate how existing cyber security metrics can be adapted to dynamic networks, rather than the traditional graphical security models which assume that vulnerabilities, nodes and edges are always static. The model leverages the concept of temporal graphs to monitor and evaluate security metrics, which expand upon a base CVSS score by assigning probabilities, impacts, and costs of attacks to create a new layer of security scores. It defines various parameters such as Risk on attack paths (R), Return on attack paths (ROA), and Cost on attack paths (AC). It also calculates the Standard deviation of attack path lengths (SDPL), Probability of attack success on paths (Pr), and a Normalised mean of attack path lengths (NMPL).

Hong et al. [12] extended upon this work by differentiating security metrics into categories for attack and defence. These metrics are designed to evaluate the effectiveness of MTD techniques by monitoring changes in the security posture as the attack surface evolves. A crucial element of the attack surface is the attack paths, and monitoring variations in these paths enables an assessment of the increased effort required for attacks. Specifically, Attack Path Variation (APV) measures shift in these paths as MTD techniques modify the network configuration. Additionally, the exposure time of attack paths is vital, as longer exposures increase the likelihood of a successful attack. The cost of attacks is also significant, estimated through the difficulty of exploiting vulnerabilities as rated by the Common Vulnerability Scoring System (CVSS). The duration it takes for an attacker to breach each point in an attack path is a critical metric; longer attack durations increase the chances of detection. The defence effort metrics primarily focus on the costs associated with deploying MTD techniques within a network. These costs are multifaceted and include the monetary expenses incurred from integrating new MTD technologies. Additionally, there are operational impacts to consider, such as the downtime necessary for replacing each node with a different variant. This process can affect network efficiency, leading to increased service overhead and potential delays in the communication medium, all of which are crucial factors in the overall assessment of defence efforts.

3.3 Deep Reinforcement Learning in MTD

Sections 3.1 and 3.2 lay the foundation for understanding what to deploy and when in MTDs. However, network systems often have security flaws like similarity, determinism, and static configurations, which leave them vulnerable to prolonged observation, analysis, and repeated attacks. Attackers have since identified adaptive strategies that can exploit the vulnerabilities in time-based MTD techniques [18]. Reinforcement learning is particularly effective for this task because it allows the defence agent to learn the adversarial decision-making strategy and continuously adapt its own strategy to prioritise the rewards that lead to the overall stability of the network in both the attack and defence metrics.

Yao et al. [19] introduced the WoLF-BSS-Q strategy for multi-agent Markov games, framed around the Cyber-Kill-Chain model. In these stochastic games, multiple agents interacted in a dynamic environment, where states evolved based on agents' actions and probabilistic rules, where each agent aimed to maximise long-term rewards. WoLF-BSS-Q combined two elements: micro-level strategy selection using the BSS algorithm in a leader-follower dynamic, and macro-level adaptation via the WoLF mechanism, which adjusted learning rates depending on whether the agent is winning or losing. To counter Q-value overestimation, the authors also proposed a "constrained Q-learning" variant, ensuring more accurate and reliable learning.

Zhang et al. [20] introduced EVADE, a reinforcement learning strategy to address key challenges in MTD. EVADE reduced the cost of shuffling network topology in resource-limited environments and tackles scalability issues in multi-objective defence using deep reinforcement learning (DRL). By incorporating a fractal-based environment (FSS), EVADE accelerated DRL training and achieved near-optimal solutions within network adaptation budgets. However, EVADE faced scalability issues due to DRL overhead in an SDN controller.

Kreischer. [21] presented a new strategy that integrates Federated Learning (FL) and Reinforcement Learning (RL) to jointly develop methods for implementing MTD to combat malware attacks. RL adapted more effectively, learning continuously to potentially address even newly emerging threats. To maintain the privacy of each device's behavioural data, the defence selection policy is trained in a federated manner. This collaborative approach could help in effectively responding to an attack globally after it has been initially detected locally. However, this work is limited in the fact that it does not consider the temporal aspects of the network and attacker.

Zhang et al. [22] identified two key challenges in integrating MTD into Delay Tolerant Mobile Networks (DTMN): resource-intensive operations and reliance on current data without considering future events. To address these, they proposed a collaborative mutation-based MTD (CM-MTD) approach, featuring host address mutation (HAM) and route mutation (RM) to disrupt the cyber kill chain. The approach begins with the formulation of a semi-Markov decision process (SMDP) to model time-varying security threats and dynamic MTD scheme deployment. Predictions of security events are made using long short-term memory (LSTM) networks, which serve as the network states within the SMDP. The action space of the SMDP is then refined by removing infeasible actions that do not meet network constraints.

4 Methodology

The goal is to develop a machine learning model that determines the most effective timing and type of MTD based on the calculated network security score. This involves identifying the appropriate features from the existing simulator framework, calculating a network security score and using that to train a machine learning model that will be able to identify appropriate timings and MTD methods. Accordingly, section 4.1 will introduce the MTDShield module, section 4.2 will cover the model training and section 4.3 will cover the model evaluation strategy.

The existing simulator framework highlighted in green consists of three main components: the Network Module (b), MTD module (c) and Attacker module (a) [23] as shown in Figure 1. The network module is a graph comprised of nodes and edges, which symbolise hosts and the connections between. Each node has IP addresses, operating systems, services, vulnerabilities, and user access credentials. The MTD module facilitates MTD operations by deploying MTD techniques into the network on a timed basis. The attacker module conducts attack operations, executing a sequence of attack actions on hosts within the network with the aim to compromise hosts.

The original simulator framework is time-based, deploying MTD into the network on a fixed schedule with no regard for the network security posture. To transform this simulator from a time-based simulator to a reactive one, we introduce the MTDShield plugin highlighted in orange in Figure 1. MTDShield will receive information about the network posture in real time from the network module and use that to determine if an MTD should be triggered. If the model determines that triggering an MTD will improve the network security posture, it will select the most appropriate MTD technique from a list and deploy that into the network via the MTD operations module.

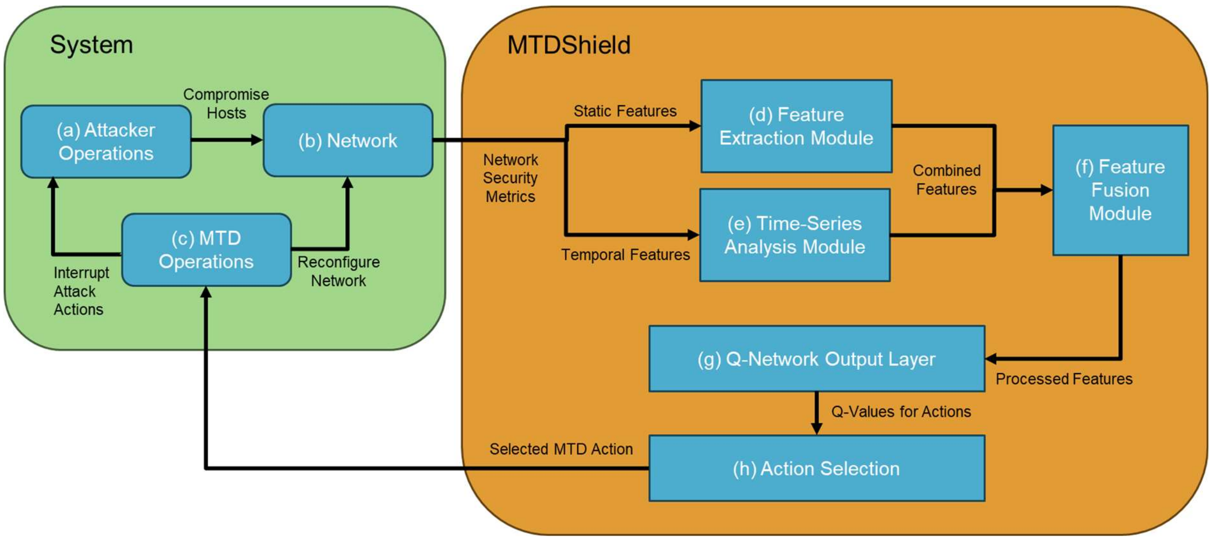

Figure 1: Overview of MTDShield System Architecture

4.1 MTDShield

Reinforcement Learning (RL) offers a powerful framework for addressing dynamic and adaptive environments, which is critical for defending against adversarial attacks in networks

[24]. As highlighted in Section 3.3, adaptive strategies like RL are particularly well-suited for environments where network configurations need continuous adjustment to mitigate threats. Zhang et al. introduced EVADE, which demonstrated how DRL can optimise resource allocation in a resource-limited MTD context while reducing the cost of reconfigurations [20]. However, EVADE faced challenges related to scalability, especially in complex environments with significant DRL overhead.

Building on this, the model proposed in this work addresses key vulnerabilities by formulating the problem as a MDP, where the RL agent updates its policies from interacting with the environment [25]. This work uses a neural network to approximate the optimal action-value function as shown in equation 1 which refers to the highest possible sum of rewards r_t, adjusted by the discount factor γ at each time step t, attainable through a policy $\pi = P(a|s)$, following the observation of state s and selection of action a [26].

$$Q^*(s,a) = \max_{\pi} \mathrm{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a, \pi]$$

Equation 1: Optimal Action-Value Function

In this context, the Deep Q-learning approach stands out due to its ability to handle high-dimensional state spaces and continuously adapt based on network security metrics [27]. The ability to adapt dynamically to changing security conditions in real time, as in Yao's WoLF-BSS-Q strategy [19], is a critical advantage when dealing with network adversaries who employ adaptive strategies.

The proposed model features five key MTD actions: IP shuffle, OS diversity, Service diversity, Complete topology shuffle, and the option not to deploy MTD, which mirrors the multi-objective nature of MTD discussed in works like Zhang et al. [22]. The state space in MTDShield captures comprehensive network security metrics, including the Host Compromise Rate (HCR), Number of Vulnerabilities, Number of Exposed Vulnerabilities, and others, which are essential for evaluating the network's current security posture and guiding the RL agent's decision-making process. These metrics align with the evaluation criteria discussed in section 3.2, providing a foundation for the model to prioritise rewards that ensure the stability of the network.

The choice to implement Deep Q-learning stems from its strength in dealing with discrete actions and continuous feedback, which allows for the effective optimisation of defence actions over time. This method addresses the challenges identified in previous literature regarding the balance between network security and resource management in MTD environments. Furthermore, by leveraging a DQN, the model can generalise across different network states and actions, ensuring scalability and robustness in diverse attack scenarios [28].

4.1.1 Feature Extraction Module

The feature extraction module (d) in Figure 1 transforms raw network security metrics into a high-level representation optimised for the RL algorithm. This is achieved through densely connected neural network layers designed to capture intricate relationships among the input features. The process begins with feeding the raw data into a dense layer of 128 units, allowing the model to capture a broad range of features. A Rectified Linear Unit (ReLU) activation function follows, introducing non-linearity to help the network recognise more complex patterns [29]. To enhance training stability, batch normalisation is applied after each dense layer to standardise the inputs. A second dense layer, with 64 units, refines the features further, reducing the risk of overfitting. This layer is also followed by ReLU and batch normalisation. Finally, a 30% dropout is introduced as an effective regularisation technique that maintains performance [29]. Together, these layers distil the raw metrics into an abstract, high-level representation, making hidden patterns and correlations more accessible for processing by the RL algorithm.

Among the metrics processed, the Host Compromise Ratio (HCR) is an indicator of network security health. It measures the proportion of hosts within a network that are compromised or at high risk due to identified vulnerabilities, providing insights into the extent of malware infections, misconfigurations, or unpatched vulnerabilities. A high HCR signals substantial exposure to potential breaches, necessitating urgent actions such as patch management, system hardening, and user education. Conversely, a low HCR indicates that the network's threat mitigation strategies are effective. By integrating the HCR, the model can better prioritise remediation efforts and bolster network defences.

The module also analyses the Number of Vulnerabilities, which totals the identified security weaknesses within the network environment. This metric is particularly important during the initialisation phase of the network graph in MTDSimTime. It is further refined by the Number of Exposed Vulnerabilities, focusing on vulnerabilities that are accessible to potential attackers due to their presence in publicly accessible systems or areas lacking sufficient network segmentation. Monitoring changes in this metric as MTD solutions are deployed allows the model to assess the ongoing effectiveness of security measures.

The Attack Path Exposure Score quantifies the risk associated with potential attack paths in the network. This score evaluates the vulnerability of interconnected systems and resources that attackers could exploit to access critical assets. It incorporates several factors, including the number and severity of exploitable vulnerabilities (as gauged by CVSS scores), the network's configuration and topology, and the accessibility of critical nodes. Higher scores indicate more significant exposure, suggesting easier navigation for adversaries through the network to target high-value assets.

4.1.2 Time Series Analysis Module

The time series analysis module (e) in Figure 1 is tailored to capture the temporal dependencies inherent in network security data. Much like how Zhang et al. [22] used Long Short-Term Memory (LSTM) architecture to predict network states, MTDShield utilises a LSTM architecture to process sequences of network events, such as variations in Mean Time to Compromise (MTTC) and the intervals between MTD deployments. By analysing the evolution of these metrics over time, the LSTM module can learn long-term dependencies that might be missed by traditional models [30]. The module begins by feeding the time-series input into a 64-unit LSTM layer, which processes the sequential data and returns the entire sequence. A ReLU activation function is applied, followed by batch normalisation to stabilise training.

The data is then passed to a second LSTM layer with 32 units, further refining the temporal patterns, followed again by ReLU and batch normalisation. A dropout of 30% is applied to mitigate overfitting [29]. The resulting output is a temporal feature vector that encodes the dynamic evolution of network metrics, providing valuable insights for predicting future network states based on historical data.

MTCC is a security metric that measures the average duration it takes for an attacker to successfully breach a system or network after identifying a potential vulnerability. In an MTD-enabled environment, the MTTC provides crucial insights into the effectiveness of these adaptive security measures. Ideally, by continually shifting the attack surface, MTD should significantly increase the MTTC, giving defenders more time to detect, respond to, and neutralise threats.

Downtime and Operational Impact for Node Replacement refers to the temporary loss of service and associated disruptions that occur when a network node is replaced, updated, or reconfigured during an IP Shuffle or OS Diversity. This metric is particularly relevant when deciding when the most appropriate time is to deploy an MTD technique as the availability of the network must be considered alongside security.

Time since last MTD is a metric used to measure the elapsed time since the last MTD technique was deployed into the network. This metric is important as the larger this gets; the more time the attack surface has remained static which increases the likelihood that reconnaissance efforts from adversaries would pay off.

4.1.3 Feature Fusion Module

The feature fusion module (f) in Figure 1 combines the outputs from both the feature extraction and time series analysis modules to create a unified representation that incorporates both spatial and temporal aspects of the network security data. This is achieved by concatenating the abstract feature vector from the feature extraction module with the temporal feature vector from the time series analysis module. The concatenated vector is then passed through a 64-unit dense layer to learn joint representations of these features, followed by a ReLU activation function to introduce non-linearity. Batch normalisation is applied to ensure stable training by normalising the inputs, and a 30% dropout is added to reduce overfitting [29]. This fusion of features allows the model to integrate both static and time-dependent information.

4.1.4 Q-Network Output Module

The Q-network output layer (g) is responsible for generating the final action-value predictions that guide the reinforcement learning agent's decisions. After processing the input features through the preceding layers, the Q-network outputs a set of five Q-values, one for each MTD technique plus the option to do nothing. These Q-values represent the expected future rewards for choosing a specific technique in each state [31]. The output layer typically consists of a dense layer where the number of units corresponds to the number of possible actions. This allows the model to estimate the value of each action, enabling the agent to select the action with the highest Q-value [28]. The output layer plays a critical role in the decision-making process, as it directly influences the agent's policy by providing actionable insights based on the learned patterns from the environment.

4.2 Model Training

In traditional Deep Q-Networks (DQN), a single main neural network is used to approximate the Q-values for each state-action pair. The agent uses this main network to update the Q-values based on the Bellman equation, which involves predicting the future reward by considering the maximum Q-value for the next state [32]. Specifically, the agent selects the action that maximises the Q-value for the next state and uses it to update the current Q-value estimate. However, this approach can lead to instability in the learning process because these incremental updates to Q may cause the policy to diverge significantly due to the correlation between action and target values [28, 33]. This work addresses both challenges using a combination of experience replay and Double DQN learning.

Double Q-learning addresses a key limitation of traditional Q-learning: the overestimation of Q-values. This overestimation occurs because the main Q-network is responsible for selecting the best action and evaluating the value of that action. This creates a correlation between the Q values and the target values $r + \gamma \max_{a'} Q(s', a')$. The consequence of this correlation is that the network may not be accurate in its predictions and can consistently favour overoptimistic values, leading to suboptimal policies. As shown in Figure 2 below, Double Q-learning solves this by introducing two separate networks: a main network for selecting actions and a target network for evaluating their values [33]. By decoupling action selection from value estimation, the algorithm reduces the likelihood of overestimating the Q-values. The primary Q-network is used to select the action that appears to be the best, while a target network, that is updated periodically, is used to compute the value of that action. This separation reduces the correlation in the value estimation process, resulting in more stable and accurate learning [34].

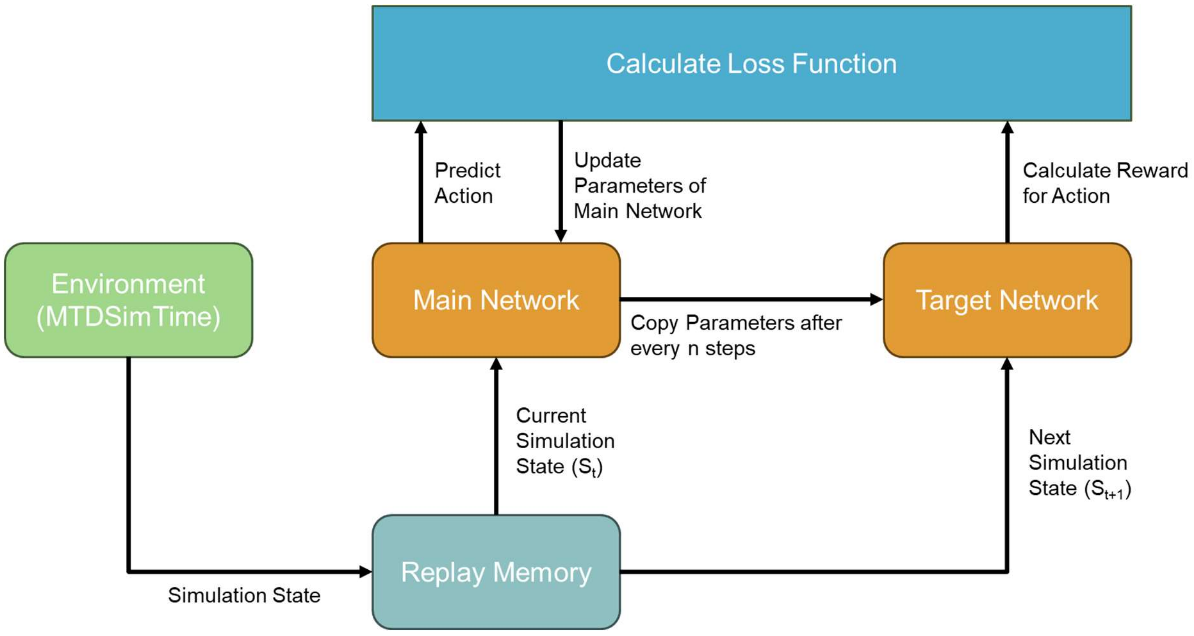
Calculate Loss Function
Predict Action
Update Parameters of Main Network
Calculate Reward for Action
Environment (MTDSimTime)
Main Network
Copy Parameters after every n steps
Target Network
Current Simulation State (S_t)
Next Simulation State (S_{t+1})
Simulation State
Replay Memory

Figure 2: Double DQN Model Training

4.2.1 Double DQN Action Selection

In Double DQN, the update rule is modified to reduce overestimation by using two separate networks as shown in Figure 2 above: the main Q-network and the target Q-network. The main Q-network is responsible for selecting the best action in the next state by choosing the action with the highest Q-value, while the target Q-network is used to evaluate the value of that action. This decoupling of action selection and value estimation helps reduce the bias in overestimating Q-values [33]. Specifically, instead of directly using the maximum Q-value from the same network, Double DQN updates the Q-value by selecting the action using the main network and estimating its value using the target network. As shown in Figure 2, the target network is updated less frequently, only copying the weights from the Main Network every n steps. This ensures that it provides a more stable estimate [35]. This approach prevents the instability caused by overoptimistic Q-value estimates, leading to more reliable learning and improving the overall performance of the agent.

4.2.2 Experience Replay

The second method to combat instability in DQN is experience replay. This mechanism randomises the data that is used to update the weights of the main model in each step which removes the correlations in the observation sequence. To perform experience replay, each time step t will result in an experience packet $e_t = (s_t, a_t, r_t, s_{t+1})$ which will be stored in the Replay Memory. During the back-propagation step, minibatches of experience packets will be randomly drawn from the Replay Memory and used to update the weights of the main network.

5 Evaluation

The evaluation methodology involves systematically testing the reinforcement learning-based MTD system through a series of experiments to assess its performance under different configurations. Section 5.1 describes the baseline scenario in which the models will be evaluated against. In section 5.2, various hyperparameters are adjusted to understand their impact on the model's learning process and effectiveness. In section 5.3, attacker detection rate is studied to evaluate how the model adapts to different threat scenarios. Finally in section 5.4, ablation studies are performed to isolate and test the significance of different model components, such as the feature extraction and time-series analysis modules. By removing these elements and comparing the performance with the complete model, the contribution of each component to the overall system is measured. These experiments collectively aim to determine the optimal configuration of the model for effective and efficient MTD deployment in real-time network defence.

5.1 Baseline Evaluation

The performance of each model will be assessed using five key metrics: Attack Success Rate (ASR), Mean Time to Compromise (MTTC), Attack Path Exposure (APE), Return on Attack (ROA), and Risk. The goal is to explore a method that will allow us to easily compare sets of scores between models. The approach that we chose was to run the simulation with no MTD applied, simply running the Network and Attacker modules to get baseline scores for each of the five metrics. We will then normalize subsequent scores against these baseline values by dividing them, the obtain a value that represents a multiple of this baseline value. This allows for easy comparison between different models.

5.2 Hyperparameter Evaluation

Section 5.2 explores the impact of key hyperparameters—gamma, epsilon, and train start—on the performance of a reinforcement learning model used to enhance network defence. Each parameter plays a crucial role in balancing short-term versus long-term decision-making, exploration versus exploitation, and early training speed versus stability. By conducting experiments with varying values for each hyperparameter, this section aims to uncover how these settings influence the model's ability to learn an optimal defence policy.

5.2.1 Impact of Gamma (γ)

The gamma parameter, or discount factor, in a reinforcement learning model determines how much future rewards are considered when the agent is learning. A higher gamma value (closer to 1) emphasises long-term rewards, for example, prolonging system integrity consistently through reducing the system's Attack Path Exposure (APE) or minimizing the Mean Time to Compromise (MTTC) across multiple attack attempts or balancing resource efficiency with defence by optimising resource usage while maintaining effective defence strategies, minimising the need for frequent changes to the network that might increase overhead. Conversely, a lower gamma value (closer to 0) makes the agent focus more on immediate rewards, such as immediately triggering MTD to mitigate a vulnerability which may lead to a quicker but less optimal decision [36]. The most common values of gamma used in other experiments range from 0.9 to 0.99 as they focus on long term strategies. This experiment will instead vary gamma values from 0.50 to 0.99 to observe how the agent's focus on short-term versus long-term rewards affects its ability to learn an optimal policy. The lower values are used to study the trade-off between immediate and future rewards in faster-changing environments [33, 35].

Figure 3 shows the different model performance when varying gamma. In general, the chart follows a bell-shaped pattern, with model performance increasing as gamma grows from 0.5 to 0.85 before dropping as gamma approaches 0.99. The best performing model was gamma_0.85 (10.77) while the worst performing model was gamma_0.5 (9.46) which is a 12.16% difference.

Taking a deeper dive into the individual metrics, we observe that Risk and ROA affected the overall score of the models the most with the range being as much as 41.78% and 40.26%. While the difference in APE was less than the top 2 at 10.62%, it still accounted for a large overall swing due to the relatively large absolute value when compared to the other metrics.

We observe that ASR and MTTC had less influence on the overall performance throughout the 10 models tested, with only a 3.53% and 6.57% difference respectively from the top performing to the worst performing models.

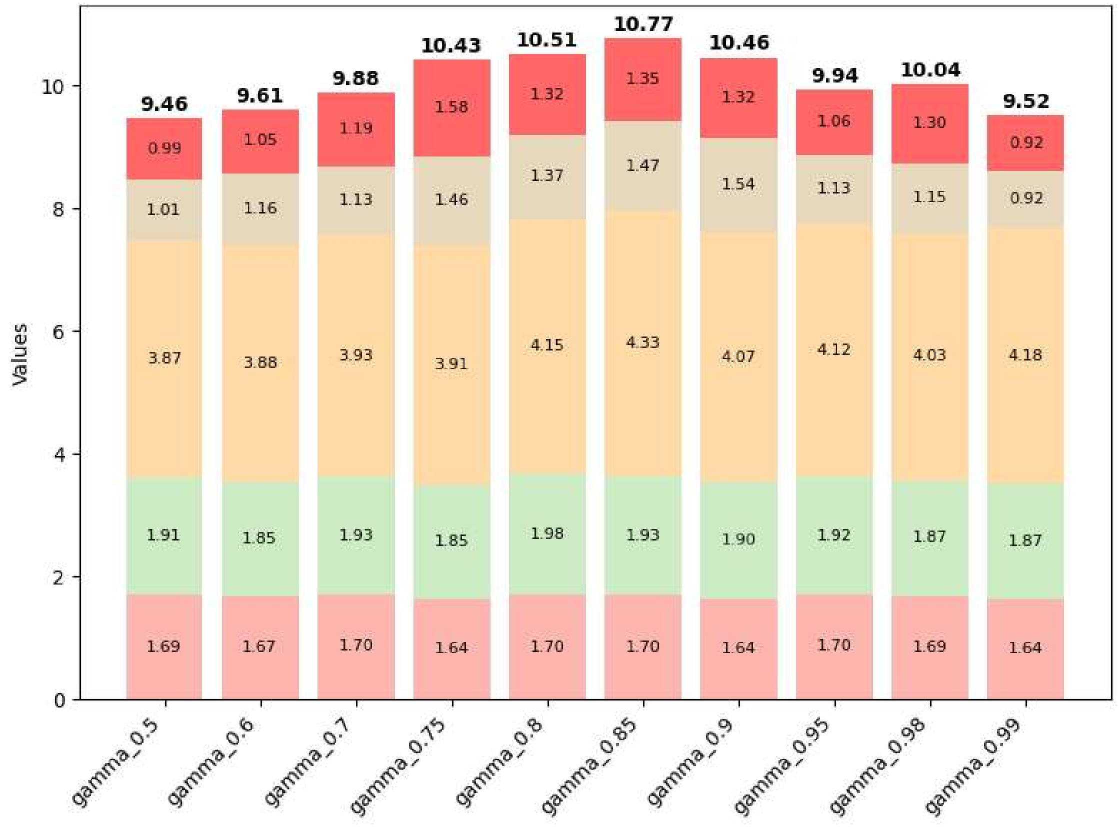

Figure 3: Impact of Gamma on model performance (Top- down: Risk, ROA, APE, MTCC, ASR)

5.2.2 Impact of Epsilon

The impact of changing epsilon in a reinforcement learning model directly influences the agent's exploration versus exploitation behaviour. Epsilon controls the likelihood of the agent taking random actions (exploration) versus selecting the action it believes to be optimal based on its learned policy (exploitation). A high initial epsilon encourages more exploration, which allows the agent to gather diverse experiences and better understand the environment, but it may slow down the learning process initially [29]. As epsilon decays over time, the agent shifts towards exploitation, relying more on its learned policy. The epsilon decay rate determines how quickly this transition occurs. A slower decay maintains exploration for a longer period, potentially helping the agent avoid suboptimal policies, while a faster decay leads to quicker exploitation, which may improve performance early but risks missing better strategies. This experiment will involve testing various epsilon values (ranging from 0.5 to 1.0) which range from moderate to high exploration. The rationale for not testing below 0.5 is that the model would lean towards exploitation too early on, which may lead to premature convergence and suboptimal policies due to insufficient exploration [28]. Different gradual decay rates (from 0.980 to 0.998) allow us to view the impact of changing epsilon without compromising model performance too significantly [34].

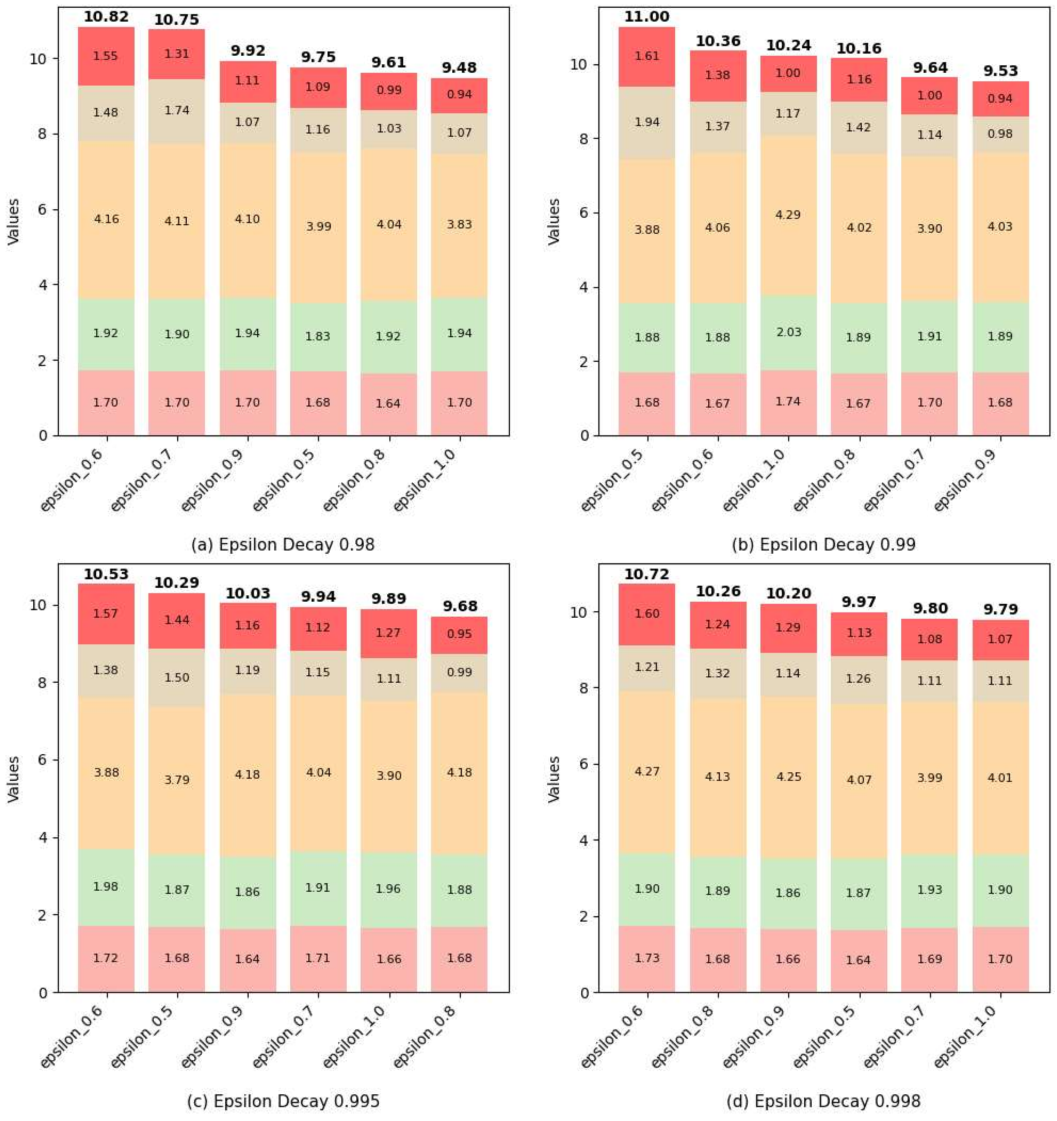

Figure 4: Impact of Epsilon and Epsilon Decay on Model Performance

Figure 4 shows 4 subplots that each represent a level of epsilon decay and the varying levels of epsilon for that decay value. In general, the lower values of epsilon (0.5, 0.6 and 0.7) outperformed the higher values of epsilon (0.8, 0.9 and 1.0). The best performing model was epsilon_0.5 with a decay of 0.99 (11.00) in Figure 4b while the worse performing model was epsilon_1.0 with a decay of 9.48 in Figure 4a which is a 13.82% difference.

While the individual metrics within each model fluctuates, it should be noted that the best performing model in each subplot had the highest score in Risk. As for the range, varying epsilon and epsilon decay had a greater impact on the individual metrics as the variance was higher compared to gamma. Risk and ROA once again had the greatest difference with a percentage change of 41.61% and 49.48%. APE, MTCC and ASR had swings of 11.66%, 9.85% and 5.75% respectively.

5.2.3 Impact of Train Start

The train start parameter plays a crucial role in the efficiency and stability of reinforcement learning. Train start determines when the model begins training by specifying how many experiences must be collected in memory before training starts. A smaller train start allows the model to begin learning earlier, but may result in overfitting to initial, possibly random, experiences, while a larger train start ensures more diverse experience collection before learning begins, leading to more stable updates. This experiment will vary train start values (500 to 2000 experiences) to explore the trade-off between early training speed and long-term stability. By testing a range of train start values, this experiment seeks to understand how different amounts of pre-training experience affect the balance between speed and stability in the simulated environment.

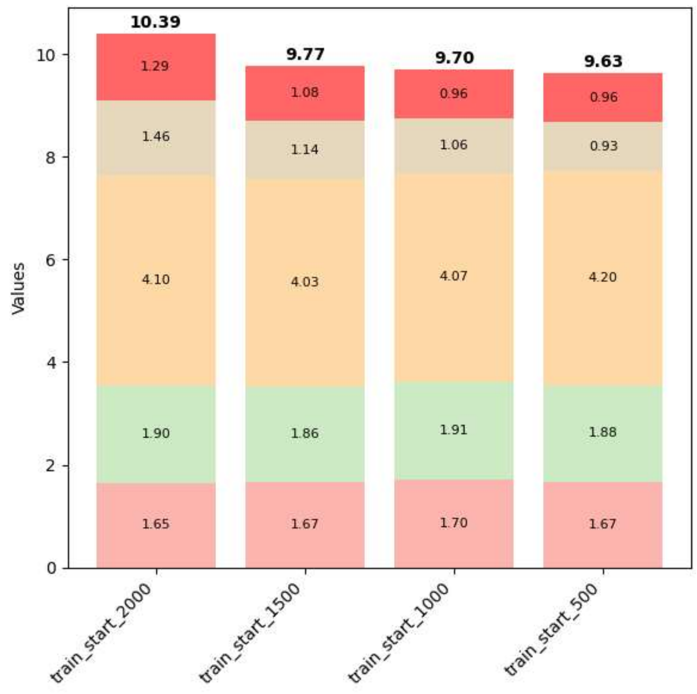

Figure 5: Impact of Train Start on Model Performance

Figure 5 above shows that having a train start of 2000 (10.39) was the best performing model of the 4 tested. This model outperformed the next closest model by 5.97% and the worst performing model by 7.31%. Much like what was observed in sections 5.2.1 and 5.2.2, most of the gains that this model made were in the Risk and ROA metrics.

5.3 Attacker Detection Rate Evaluation

Most traditional network systems will have some form of an Intrusion Detection System (IDS) which is an application that monitors network traffic and searches for known threats and suspicious or malicious activity [37]. However, as attacker evasion techniques and zero-day attacks become more and more sophisticated, the ability of an IDS to offer credible and timely information becomes less impactful.

This experiment simulates the ability of an IDS by feeding information about the actions the attacker is taking to the model during the training process. The goal is to assess how varying levels of IDS performance will influence the model's performance. The experiment is structured by training multiple models, each receiving different amounts of information about the attacker's behaviour, ranging from 0% (no knowledge of the attacker's actions) to 100% (full knowledge of the attacker's actions). The experiment aims to uncover how varying degrees of attacker detection affect the MTD system's ability to anticipate and react to threats.

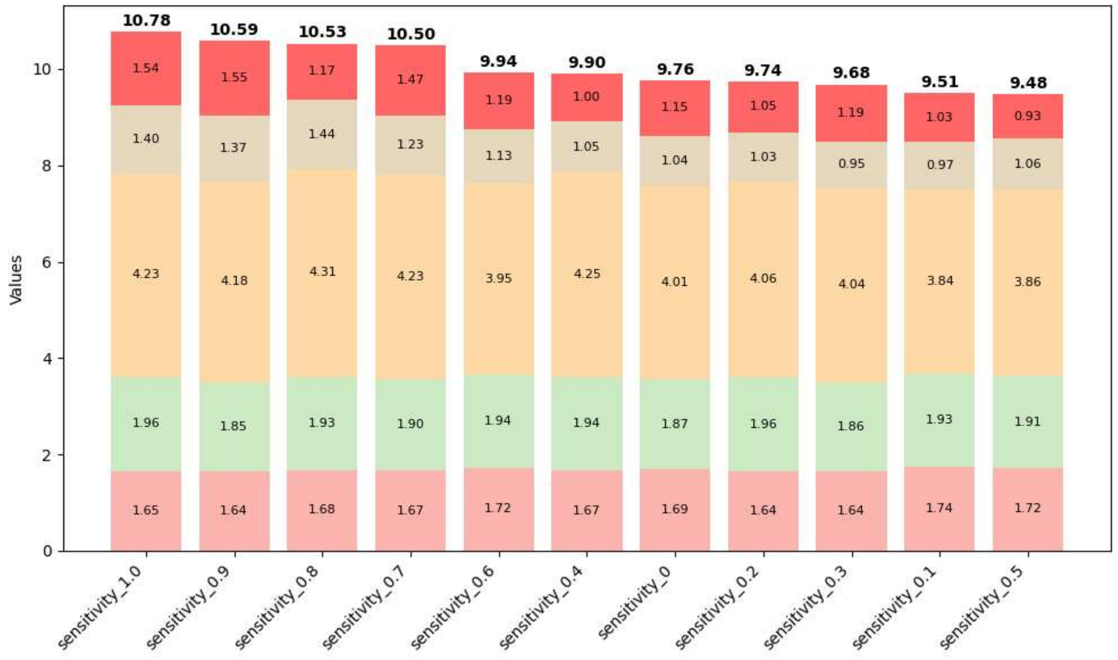

Figure 6: Impact of Attacker Detection Rate on Model Performance

Figure 6 above shows the impact of adjusting attacker detection rate on model performance. The first 4 models' sensitivity_1.0 – sensitivity_0.7 demonstrate a trend where performance decreases marginally as detection rate decreases (2.60%). However, it can be observed that there is a cutoff after sensitivity_0.7 where the performance drops by 5.33% from sensitity_0.7 – sensitivity_0.6. From there on, the performance of the models no longer seems correlated with their attacker detection rate.

5.4 Ablation Studies

This experiment serves to evaluate the contributions of various components and feature sets within the model, we conducted a series of ablation studies. These studies are designed to systematically assess the impact of removing or altering specific model elements, such as the static feature extraction module and the time-series LSTM layers. The model leverages both static features and time-series data to make predictions. This ablation study isolates these factors to identify critical features and architectural elements that contribute meaningfully to the model's predictive capabilities, while also highlighting potential areas of redundancy or unnecessary complexity.

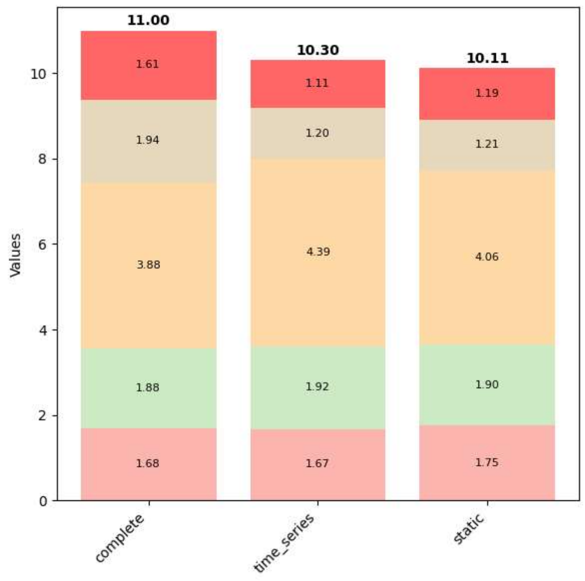

Figure 7: Impact of Removing Modules on Model Performance

Figure 7 compares the best performing model from the previous sections with one that only has a time series or static feature extraction module. The best performing model outperformed the other 2 models by 6.36% and 8.09% respectively.

However, when examining the individual security metrics, while the complete model outperformed the other two in Risk (31.06% and 26.09%) and ROA (38.14% and 37.63%), it under performed in the APE metric by 13.14% and 4.6% compared to the two incomplete models.

6 Discussion

The results obtained in Section 5 align with the specific simulation conducted in MTDSimTime and the parameters chosen for that context. However, these results may vary for networks with different configurations or when using other MTD techniques. By applying the evaluation methods described, we can analyse how each parameter impacts the relevant metrics. This, in turn, allows us to identify suitable parameter ranges for different types of network environments.

6.1 Gamma Values

When examining model performance in relation to gamma values in Section 5.2.1, it was observed that the performance followed a bell-shaped curve. The top-performing models were within the range of 0.75 to 0.9, with performance declining outside this range. While this may initially seem like similar outcomes at both ends of the spectrum, a closer analysis of individual metrics reveals the nuanced effects of gamma on model performance.

For models with higher gamma values (>0.9), we observe an increased APE score compared to models with lower gamma values (<0.75). A higher gamma in reinforcement learning encourages the agent to prioritize long-term strategies, which focus on proactively reducing vulnerabilities and securing potential attack paths. On the other hand, models with lower gamma values show proportionally higher Risk and ROA scores. These models tend to focus on short-term rewards, such as triggering MTD actions immediately to mitigate ongoing attacks, leading to the observed higher Risk and ROA scores.

Based on these observations, we can conclude that the higher performance of the models in the middle of the range is due to the balance between short term and long-term rewards. Gamma values that are too low emphasises actions that yield better results in the short term, which could lead to overlooking strategies that may lead to better network performance in the long term. On the other hand, due to the limited run-time of the simulation, placing too much importance on future rewards with high gamma values, which often delay beneficial short-term rewards, led to a scenario where these long-term benefits never fully materialised.

6.2 Epsilon Values

Figure 4 in Section 5.2.2 showed 24 different models with varying combinations of epsilon and epsilon decay values. It was noted that the best performing models were the ones that started with low epsilon values (0.5 – 0.7) which runs contrary to the epsilon-greedy policies that other reinforcement learning models tend to favour [28]. One of the factors that contributed to this evaluation is the Risk metric. When observing the 4 subplots in Figure 4, it can be observed that while the other four metrics tend to fluctuate, the top performing model in each plot had the highest Risk score. This reinforces the idea that a lower epsilon leads a model towards exploitation more quickly, doubling down on established, proven actions that are known to work, which shows in the greatly improved Risk score.

However, this quick shift towards exploitation has negative consequences for the longer-term metrics such as APE, where the top performing model suffers in, having one of the lowest APE scores of the 24 tested models. This model which has an epsilon of 0.5, the lowest starting value, and a quick decay of 0.99 quickly shifted towards exploitation. This led it to lack the time to explore a wide variety of actions and scenarios which caused the long-term metrics to suffer.

We can conclude that the low epsilon models favour known, high-reward actions which lead to stronger immediate and consistent results. This strategy works well in the simulated environment due to the limited runtime, but more studies need to be conducted to determine their performance in extended trials.

6.3 Train Start

Section 5.2.3 showed the results of adjusting the train start hyperparameter. While there was a small improvement by increasing the train start due to improved quality of experiences stored in the memory buffer, this value shouldn't be pushed too far due to the increased computational burden it imposes on the system when training.

6.4 Attacker Detection Rate

In Section 5.3, we explored simulating an IDS within the simulator to assess the impact of the attacker's detection rate on model performance. For detection sensitivities in the range of 0.7 to 1.0, there was a clear trend: model performance improved as the detection rate increased. This suggests that these models were able to integrate the information they received about the attacker's actions into their strategies, leading to better overall performance.

However, a cutoff point was observed at a sensitivity of 0.7. Beyond this threshold, model performance no longer correlated with the detection rate, with performance scores becoming inconsistent and lacking any clear trend. We hypothesize that this is due to the model receiving incomplete information about the attacker's actions. Once the detection sensitivity falls below 0.7, the model can only acquire attacker information sporadically, resulting in performance levels like those of models with no information at all.

6.5 Ablation Studies

In section 5.4, we conducted an ablation study where we removed either the time-series or static feature extraction module from the model to study the effects on performance. While the complete model outperformed both incomplete ones, the difference in performance was not as large as expected. It seems that despite losing one of the critical elements of the model, it is still able to make sound judgements on when and what MTD techniques to deploy.

7 Future Works

This research has opened several avenues for further exploration and improvement in the field of MTD automation using artificial intelligence, however, as discussed in Section 6, there is still room for more studies to be conducted on real networks, increased runtime and more ablation studies.

7.1 Testing on Real-World Datasets

The results discussed in Sections 5 and 6 are all specific to the MTDSimTime simulator which is a representation of a network. Therefore, the findings made in the simulator may not necessarily be applicable to real networks. Integrating external intrusion datasets into the simulator would provide a valuable opportunity to test the robustness and generalisability of the MTDShield in real-world scenarios. By incorporating real-world data, the system's ability to respond to actual cyber threats could be evaluated, offering insights into its practical application in defending against contemporary attack methods. This integration would enhance the simulator's realism and allow for more comprehensive testing of the proposed MTD strategies.

7.2 Increased Runtime and Simulation Space

As discussed in Sections 6.1 and 6.2, certain values of gamma and epsilon have stood out as leaders in this study which run contrary to the values used in many other RL studies. One of the reasons could be the size of the network resulting in smaller runtime. The models need to be tested on larger, more realistic networks to determine if these parameters can still come out on top.

7.3 Network Metrics Optimisation

One key area for future work involves investigating which parameters and features can be optimised to enhance model performance. Section 6.5 conducted ablation studies where it was discovered that the removal of entire modules only dropped the performance by 8%. More studies need to be conducted into the individual input metrics to determine which ones have the most impact on model performance.

8 Conclusion

This thesis demonstrated how the AI-driven MTD system MTDShield could be leveraged to automatically adjust the attack surface in real time. By utilising a reinforcement learning model, the system was able to continuously learn from network security posture metrics—such as the number of vulnerabilities, exposed vulnerabilities, MTTC, APE, and HCR. The deep Q-learning model used in this research deployed a range of MTD techniques—such as IP address shuffling, OS diversity, service diversity, and topology changes—based on these metrics, which resulted in significant improvements in network resilience. The results showed reduced ASR and increased MTTC, indicating that the proposed system successfully prolonged the time required for attackers to breach the network.

Key findings from the evaluation experiments demonstrate the importance of hyperparameter tuning in reinforcement learning, balancing short-term actions with long-term network security benefits with the choice of gamma and epsilon values. Models with gamma (between 0.75 and 0.90) and epsilon (favouring early exploitation) values exhibited the best performance.

One of the most important takeaways from this research is the demonstration of how dynamic, AI-driven MTD systems can substantially improve cybersecurity outcomes. Unlike static defences, which remain vulnerable to reconnaissance and prolonged observation, MTD introduces unpredictability by continuously altering system configurations. The reinforcement learning model was able to learn when and how to apply these changes, striking a balance between security efficacy and operational impact. By optimising MTD deployments based on real-time conditions, the system reduced the attack surface at critical moments, disrupted attackers' strategies, and minimised unnecessary MTD deployments, which could be costly in terms of system performance and downtime.

In conclusion, this thesis has developed a novel, AI-enhanced MTD framework that dynamically adjusts to evolving threats and network conditions. By utilising deep reinforcement learning, this approach not only enhances security but does so in a manner that maximises resource efficiency, reducing both operational costs and risks. This work paves the way for more intelligent, responsive cybersecurity systems.

9 References

[1] V. Sardana and S. Singhania, ‘Digital Technology in the Realm of Banking: A Review of Literature’, *Int. J. Res. Finance Manag.*, vol. 1, no. 2, pp. 28–32, Nov. 2018, doi: https://doi.org/10.33545/26175754.2018.v1.i2a.12.

[2] S. Al-Bassam and A. Al-Alawi, ‘The Significance of Cybersecurity System in Helping Managing Risk in Banking and Financial Sector’, *J. Xidian Univ.*, vol. 14, no. 7, pp. 1523–1536, Nov. 2019, doi: https://doi.org/10.37896/jxu14.7/174.

[3] D. Thaw, ‘The Efficacy of Cybersecurity Regulation’, *Ga. State Univ. Law Rev.*, vol. 30, no. 2, pp. 287–374, 2014 2013.

[4] J.-H. Cho *et al.*, ‘Toward Proactive, Adaptive Defense: A Survey on Moving Target Defense’, *IEEE Commun. Surv. Tutor.*, vol. 22, no. 1, pp. 709–745, 2020, doi: 10.1109/COMST.2019.2963791.

[5] S. Sengupta, A. Chowdhary, A. Sabur, A. Alshamrani, D. Huang, and S. Kambhampati, ‘A Survey of Moving Target Defenses for Network Security’, *IEEE Commun. Surv. Tutor.*, vol. 22, no. 3, pp. 1909–1941, 2020, doi: 10.1109/COMST.2020.2982955.

[6] H. Alavizadeh, D. S. Kim, J. B. Hong, and J. Jang-Jaccard, ‘Effective Security Analysis for Combinations of MTD Techniques on Cloud Computing (Short Paper)’, in *Proc. Int. Conf. Inf. Security Pract. Exp., 2017*, pp. 539–548. doi: 10.1007/978-3-319-72359-4_32.

[7] G. Cai, B. Wang, W. Hu, and T. Wang, ‘Moving target defense: state of the art and characteristics’, *Front. Inf. Technol. Electron. Eng.*, vol. 17, no. 11, pp. 1122–1153, Nov. 2016, doi: 10.1631/FITEE.1601321.

[8] N. Ben-Asher, J. Morris-King, B. Thompson, and W. J. Glodek, “Attacker Skill Defender Strategies And The Effectiveness Of Migration Based Moving Target Defense In Cyber Systems,” in *Proc. 11th Int. Conf. Cyber Warfare Security (ICCWS)*, 2016, p. 21.

[9] Y. Huang and A. K. Ghosh, “Introducing Diversity And Uncertainty To Create Moving Attack Surfaces For Web Services,” in *Moving Target Defense*. New York, NY, USA: Springer, 2011, pp. 131–151. doi: 10.1007/978-1-4614-0977-9_8.

[10] D. C. MacFarland and C. A. Shue, “The SDN Shuffle: Creating A Moving-Target Defense Using Host-Based Software-Defined Networking” in Proc. 2nd ACM Workshop Moving Target Defense (MTD), 2015, pp. 37–41. doi: 10.1145/2808475.2808485.

[11] R. Zhuang, S. Zhang, A. Bardas, S. A. DeLoach, X. Ou, and A. Singhal, Investigating The Application Of Moving Target Defenses To Network Security,” in *Proc. 6th Int. Symp. Resilient Control Syst. (ISRCS)*, 2013, pp. 162–169. doi: 10.1109/ISRCS.2013.6623770.

[12] J. B. Hong, S. Y. Enoch, D. S. Kim, A. Nhlabatsi, N. Fetais, and K. M. Khan, ‘Dynamic Security Metrics For Measuring The Effectiveness Of Moving Target Defense Techniques’, *Comput. Secur.*, vol. 79, pp. 33–52, Nov 2018, doi: 10.1016/j.cose.2018.08.003.

[13] H. Alavizadeh, J. B. Hong, J. Jang-Jaccard, and D. S. Kim, “Evaluation For Combination Of Shuffle And Diversity On Moving Target Defense Strategy For Cloud Computing” in *Proc. 17th IEEE Int. Conf. Trust Security Privacy Comput. Commun. (TrustCom)*, 2018, pp. 573–578. doi: 10.1109/TrustCom/BigDataSE.2018.00087.

[14] M. Behi, M. GhasemiGol, and H. Vahdat-Nejad, ‘A New Approach to Quantify Network Security by Ranking of Security Metrics and Considering Their Relationships’, *Int. J. Netw. Secur.*, vol. 20, no. 1, Jan. 2018, doi: 10.6633/IJNS.201801.20(1).15.

[15] V. R. Kebande, I. Kigwana, H. S. Venter, N. M. Karie, and R. D. Wario, ‘CVSS Metric-Based Analysis, Classification and Assessment of Computer Network Threats and Vulnerabilities’, in *2018 Int. Conf. Adv. Big Data, Comput. Data Commun. Sys. (icABCD)*, 2018, pp. 1–10. doi: 10.1109/ICABCD.2018.8465420.

[16] S. E. Yusuf, M. Ge, J. B. Hong, H. K. Kim, P. Kim, and D. S. Kim, “Security modelling and analysis of dynamic enterprise networks,” in *Proc. IEEE Int. Conf. Comput. Inf. Technol. (CIT)*, 2016, pp. 249–256. doi: 10.1109/CIT.2016.88.

[17] L. Wang, M. Albanese, and S. Jajodia, ‘Network Hardening: An Automated Approach to Improving Network Security’, Cham Springer International Publishing, 2014.

[18] M. L. Winterrose, K. M. Carter, N. Wagner, and W. W. Streilein, ‘Adaptive Attacker Strategy Development Against Moving Target Cyber Defenses’, in *Adv. Cyber Sec. Anal. Decis. Sys.*, 2020, pp. 1–14. doi: 10.1007/978-3-030-19353-9_1.

[19] Q. Yao, Y. Wang, X. Xiong, P. Wang, and Y. Li, ‘Adversarial Decision-Making for Moving Target Defense: A Multi-Agent Markov Game and Reinforcement Learning Approach’, *Entropy*, vol. 25, no. 4, Art. no. 4, Apr. 2023, doi: 10.3390/e25040605.

[20] Q. Zhang, J.-H. Cho, T. J. Moore, D. D. Kim, H. Lim, and F. Nelson, ‘EVADE: Efficient Moving Target Defense for Autonomous Network Topology Shuffling Using Deep Reinforcement Learning’, in *Appl. Cryptography Netw. Secur.*, 2023, pp. 555–582. doi: 10.1007/978-3-031-33488-7_21.

[21] J. Kreischer, ‘Federated Reinforcement Learning for Private and Collaborative Selection of Moving Target Defense Mechanisms for IoT Device Security’, Master’s Thesis, University of Zurich, Zürich, Switzerland, 2023. doi: 10.5167/uzh-255748.

[22] T. Zhang *et al.*, ‘When Moving Target Defense Meets Attack Prediction in Digital Twins: A Convolutional and Hierarchical Reinforcement Learning Approach’, *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3293–3305, Oct. 2023, doi: 10.1109/JSAC.2023.3310072.

[23] W. Zhang, *MTDSimTime*. [Online]. Available: https://github.com/MoeBuTa/MTDSimTime

[24] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, ‘Deep Reinforcement Learning: A Brief Survey’, *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.

[25] S. H. Lim, H. Xu, and S. Mannor, ‘Reinforcement Learning in Robust Markov Decision Processes’, in *Adv. Neural Inf. Process. Sys.*, Curran Associates, Inc., 2013.

[26] S. E. Li, ‘Deep Reinforcement Learning’, in *Reinforcement Learning for Sequential Decision and Optimal Control*, S. E. Li, Ed., Singapore: Springer Nature, 2023, pp. 365–402. doi: 10.1007/978-981-19-7784-8_10.

[27] K. Rajwar and K. Deep, ‘Q-Learning-Driven Framework for High-Dimensional Optimization Problems’, in *2024 IEEE Congr. Evol. Comput. (CEC)*, Jun. 2024, pp. 1–8. doi: 10.1109/CEC60901.2024.10611751.

[28] V. Mnih *et al.*, ‘Human-level Control Through Deep Reinforcement Learning’, *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[30] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, ‘Long Short-Term Memory Networks for Anomaly Detection in Time Series’, presented at the 23rd Eur. Symp. on Artif. Neural Netw., Apr. 2015.

[31] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, *An Introduction to Deep Reinforcement Learning*. now, 2018. doi: 10.1561/2200000071.

[32] J. N. Tsitsiklis and B. Van Roy, ‘An Analysis of Temporal-difference Learning with Function Approximation’, *IEEE Trans. Autom. Control*, vol. 42, no. 5, pp. 674–690, May 1997, doi: 10.1109/9.580874.

[33] H. van Hasselt, A. Guez, and D. Silver, ‘Deep Reinforcement Learning with Double Q-Learning’, in *Proc. 30th AAAI Conf. on Artif. Intell.*, Feb. 2016, pp. 2094–2100.

[34] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, ‘Dueling Network Architectures for Deep Reinforcement Learning’, in *Proc. 33rd Int. Conf. Mach. Learn. - Vol. 48*, 2016, pp. 1995–2003.

[35] T. P. Lillicrap *et al.*, ‘Continuous Control with Deep Reinforcement Learning’, Jul. 05, 2019, *arXiv*: arXiv:1509.02971. doi: 10.48550/arXiv.1509.02971.

[36] R. S. Sutton and A. G. Barto, ‘Reinforcement Learning: An Introduction’. MIT Press, 2014

[37] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, ‘Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges’, *Cybersecurity*, vol. 2, no. 1, p. 20, Jul. 2019, doi: 10.1186/s42400-019-0038-7.