Using Artificial Intelligence to Automate the Deployment of MTD Operation

Wai Him Ho (22701889)
Supervisor: Jin Hong

Word Count: 7940

Date of Submission: October 13, 2024

School of Engineering
University of Western Australia

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

If yes, then provide details: Double check English

Signed Wai Him Ho Date October 13, 2024

Project Summary

In the field of cybersecurity, Moving Target Defense (MTD) is recognized for its adaptive approach to addressing evolving cyber threats. Rather than striving to eliminate all vulnerabilities within a system, MTD enhances system dynamism by continuously reconfiguring its components to modify the attack surface. Despite substantial research on MTD, a notable gap remains in the integration of Artificial Intelligence (AI) with MTD, particularly in examining various metrics as features. To address this gap, this research proposes a double Q-network reinforcement learning model and aims to investigate how different MTD intervals, network sizes and MTD techniques influence the effectiveness of employing diverse security metrics as features, thereby enhancing the performance of the reinforcement learning model.

Our evaluation focuses on four key metrics: risk (R), return on attack (ROA), attack path exposure (APE), and attack success rate (ASR). The hybrid model, which integrates both static and time-based features, emerged as the most effective optimization strategy when compared to individual metric approaches, outperforming previous MTD schemes in longer MTD intervals, with most models achieving at least 30% better. The findings also indicate that the performance of hybrid metric optimization is affected by the inclusion of metrics that do not directly reflect the network's security posture. Such metrics can dilute overall optimization effectiveness, highlighting the critical importance of careful metric selection in designing MTD strategies. While the hybrid model demonstrated better results, its performance declined in shorter MTD intervals, revealing a potential limitation in rapidly changing environments. On the other hand, the optimization of ASR consistently showcased robust performance across various network sizes and configurations during short intervals. Moreover, results shows the importance of choosing MTD type. Models that deployed diversity technique can outperform models that used shuffling technique by up to 140% in high MTD interval. Overall, this research demonstrates the effect of MTD type, MTD interval and network size on metrics optimization in the proposed reinforcement learning model.

Acknowledgements

The proposed reinforcement learning model was developed by Wai Him Ho (22701889) and Joo Kai Tay (22489437). Joo Kai is mainly responsible for the model development and Wai Him is responsible for the data collections.

Table of Contents

List of Figures

List of Tables

1 Introduction

The rapid expansion of the public use of internet has led to a significant rise in information security risks. The increasing number and variety of cyberattacks have made cybersecurity a critical concern for not just individuals but also companies and government entities.[1] Traditional static defense mechanisms have become ineffective due to advanced attacks such as Advanced Persistent Threat(APT) [2], which can perform long-term vulnerability analysis and penetration testing on the target, making firewalls and signature-based detection obsolete. Moving Target Defence offers a solution by dynamically modifying the attack surface [3] of the system, increasing uncertainty and raising the cost of attacks while preserving its fundamental functionality. It offers a cost-effective solution, as it can often be implemented using existing system components and technologies, avoiding the need for extensive time and resources to develop entirely new defense strategies [3]. Furthermore, the recent rise of artificial intelligence (AI) has introduced novel and sophisticated approaches to cybersecurity, including the integration of AI with defense techniques like MTD for more complex and adaptive protection [4].

Many existing work on MTD focuses on developing a single timeliness-based MTD technique against a specific attacker type. Research on the optimal deployment of hybrid MTD techniques is limited, and the effectiveness of a single MTD is often insufficient [5]. Although hybrid MTD approaches have been proposed [6, 7, 8, 9], they often overlook the consideration of multiple attacker types and rarely account for the availability and performance of the system [10, 11, 12] after deploying the MTD techniques. Due to the limitations of manually selecting MTD configurations based on experience [13], other research has been done in the area of automating MTD techniques, ranging from MTDeep [14] to reinforcement learning based MTD deployment [15]. However, most of these studies focus on automating the deployment of a single MTD technique against a specific attacker type. There remains a gap in using machine learning to optimally deploy one or more MTD against various attacker types.

This project will investigate the optimal automated deployment of MTD techniques against attackers using a reinforcement learning model. The goal of the model is to minimize network vulnerabilities and maximise network security. To achieve this, the MTD simulator MTDSimTime [16], an extension of MTDSim [17], will serve as a simulation and testing environment for building the machine learning model. This research aims to enhance the robustness of existing MTD studies. Previous research often overlooked thorough investigation of adversaries, resulting in a lack of validity in real-life cyber-attack scenarios. Metrics for optimizing rewards in reinforcement model also only consider Common Vulnerability Scoring System (CVSS) metrics [18] such as impact score, and complexity. Moreover, although there have been investigations on different security metric [19], previous works have not consider examining their utility as features for reinforcement learning models. Automating the deployment of multiple MTDs is essential due to the increased complexity when considering both security and performance metrics across various adversaries.

Therefore, this project seeks to bridge different aspects of MTD research to advance the state of the art. Key contributions include:

- Review previous works on the combinations of MTD techniques, the automation of MTD, and the related security metrics and incorporate them into this research
- Propose a reinforcement learning model that uses Double Deep Q-network learning

to deploy different MTD techniques against adversary.

- Examine how different metrics optimization in the reward and choices of MTD techniques affects the performance of the models under different system configurations.
- Examine the effectiveness of the proposed model against previous MTD deployment schemes

2 Literature Review

2.1 Combinations of Moving Target Defense

MTD techniques can be classified into three categories: shuffling, diversity, and redundancy [3]. Each combination of MTD techniques(e.g. Shuffling and Diversity) should be examined and assessed individually due to their emergent properties [3].

Alavizadeh et al. proposed an approach that combines both Shuffling and Redundancy Moving Target Defense (MTD) techniques and evaluated its effectiveness using a 2-layer Hierarchical Attack Representation Model (HARM) [6, 7]. The paper used HARM and SHARPE to measure two security metrics: system risk and reliability, to evaluate the MTD techniques. Additionally, two Network Centrality Measures (NCMs), betweenness centrality and closeness centrality, were used to rank the most crucial VMs in the cloud to improve security analysis. The results showed that deploying redundancy (R) and the combination of shuffle and redundancy (S + R) increased system risk, with S + R increasing risk significantly more than S alone. However, S + R outperforms S significantly in terms of reliability. While S + R reduced more risk compared to R, R achieved slightly better reliability compared to S + R. In addition to their previous work [6], Alavizadeh et al. introduced the unattackability metric [7]. The paper also concluded that the combination of two MTD techniques, such as S + R, contributed to a higher likelihood of increasing the unattackability of the system.

2.1.1 Shuffling and Diversity

Rajakumaran et al. proposed a shuffling and diversity hybrid MTD technique against DoS mitigation in Amazon web service [8]. The experiment incorporated a 2-layer HARM security model and used the Importance Measure(IM) to assess the vulnerability. Resource constraints, delay in proxy switchover, and reduction in attack probability were used as the other important metrics. The experiment showed that the attack probability is 0.5% when using either shuffle, diversity, or redundancy MTD technique where the proposed hybrid MTD reduced the attack probability down to 0.1%.

Wenxiao Zhang evaluated various combinations of shuffling and diversity(S+D) MTD strategies across different scenarios using the proposed MTDSimTime simulator [16]. Two shuffling MTD techniques (complete topology shuffle and IP shuffle) and two diversity MTD techniques (service diversity and OS diversity) were implemented. These techniques were executed either randomly, alternately, or simultaneously to combat adversaries. The paper evaluated both single and pairwise hybrid MTD techniques and found that network-shuffle-based MTD techniques had a similar impact on the MTTC metric score. There was no significant difference between using these techniques in combination with other MTD techniques or using them individually. However, the S + D

MTD technique demonstrated more favorable outcomes, yielding an average improvement of approximately 20% in the MTTC score compared to using shuffling techniques alone.

J.B. Hong et al. evaluated the combined effects of shuffle and diversity MTD techniques against two attack scenarios [9] through a simulation framework MTDSim [17]. An extra layer is added to the 2-layer HARM model to capture services provided by each host to closely model the attack behavior to improve the attack simulation. The paper used the Cyber Kill Chain [20] and MITRE ATT&CK Framework [21] to simulate realistic attack scenarios. The simulation tested against single, double, and triple MTD techniques. The experiment concluded that the best-performing results between them were similar, and using only the best single MTD is the best option in the context of the measured metrics.

2.1.2 Diversity and Redundancy

Very few papers discuss the hybrid uses of diversity and redundancy MTD techniques and the majority of the research has been done in other combinations of MTD techniques. J. H. Cho speculated that combining diversity or redundancy may not require shuffling as frequently as using shuffling alone since the diversity of the system will decrease the vulnerabilities that will be spotted by the attackers [3].

M. C. Lucas-Estañ et al. evaluated the utilization of diversity and redundancy in wireless networks to enhance the reliability and latency for mobile industrial applications [22]. They created a prototype that implements diversity and redundancy for the wireless connections between robots and conducted trials that assess the reliability and latency of wireless connections. However, the study only explored the concepts of diversity and redundancy independently and also did not involve the consideration of moving target defense.

2.1.3 Shuffling, Diversity and Redundancy

Alavizadeh et al. evaluated the effectiveness of Moving Target Defense (MTD) techniques concerning security and economic metrics in a cloud environment [23]. They employed a 2-layer Hybrid Attack-Resistance Model (HARM) to model the cloud, and Importance Metrics (IM) were utilized to measure the closeness and betweenness of virtual machines (VMs). The study employed four security metrics - system risk, attack cost, return on attack, and reliability - to assess the effectiveness of the combined MTD techniques. A combination of shuffling, diversity, and redundancy (S + D + R) was deployed with five VMs on the cloud-band for the experiment. The results showed that the values of attack cost (AC) in diversity (D)-only scenarios were lower than those of AC in S + D + R scenarios. Additionally, the deployment of S + D + R led to better reliability. Conversely, the Return on Attack (RoA) values for S + D + R were also lower than those in D-only scenarios, indicating that the attacker is less likely to attack again. The paper concluded that deploying S + D + R improved all aspects of the security and performance metrics used in the experiment.

2.1.4 Discussion

Previous works generally suggest that there is an advantage to deploying multiple MTDs over a single MTD. However, existing metrics such as attack effort [9] are not sufficient in capturing the effects of hybrid MTD deployments and there is a need for more diverse

metrics. Moreover, the capability and intelligence of the attackers are underdeveloped [16, 9]. For example, the attacker only focuses on one target node even if it is too difficult to compromise [9]. There is also no substantial analysis of the effectiveness of hybrid MTD techniques against a variety of adversaries. Besides, there can be considerations of other security models other than HARM [6, 7, 23] to assess MTD deployment. Finally, event-based MTD and hybrid MTD can result in more system degradation [5, 16] due to more abrupt disconnections between a connected client and a server and trade-offs between security and performance are neglected. Therefore, extensive investigation is needed to inquire how different MTD techniques can be combined to maximize benefits while maintaining system performance.

2.2 Automation of Moving Target Defense

Zhu Y et al. conducted a holistic survey on MTD and an overview of MTD automation based on previous works. They categorized MTD automation techniques into affordable, optimized, and self-adaptive [15]. Affordable intelligent MTD focuses on reducing high overhead and minimizing additional overhead when triggering MTD. Optimized intelligent MTD involves defensive strategy solutions tailored to specific types of attacks and generalized types of attacks. Reinforcement learning is considered a suitable choice for optimized intelligent MTD. The paper gave an example of a Deep Deterministic Policy Gradient(DDPG) deep reinforcement algorithm that routes randomly against eavesdropping attacks. Self-adaptive intelligent MTD is categorized into machine learning, machine learning with legacy defense mechanisms such as dynamic honey-pot defense, and machine learning with game theories. The paper concludes that self-adaptive is the most advanced and effective type among all 3 types of intelligent MTD proposed.

J.H Cho et al. introduced an Efficient Moving Target Defense (EVADE) system that alters network topology periodically to thwart attackers [24]. They utilized the VERN algorithm to rank vulnerabilities and implemented a lightweight solution search algorithm (FSS) to expedite training. Their hybrid MTD approach combined greedy MTD with density optimization and Deep Reinforcement Learning-based MTD, triggering periodic shuffling of network topology. Comparisons were made between Deep Q-learning Networks (DQN) and Proximal Policy Optimization (PPO), alongside their hybrid versions (S-G-DQN and S-G-PPO with EVADE), and two baseline methods (GA and Random) across various network types. The study considered epidemic attacks and state manipulation attacks (SMAs). Results indicated the proposed algorithm outperformed alternatives in enhancing performance and reducing security vulnerabilities. EVADE-based models excelled in sparse and dense networks, exhibiting resilience against SMAs. However, considerations for adaptive attacks on deep reinforcement learning agents and comprehensive evaluation of attack coverage are areas for further exploration.

Tao Zhang et al. proposed an Intelligence-Driven Host Address Mutation(ID-HAM) scheme to slow down network reconnaissance using automation. The scheme utilized deep reinforcement learning with the Markov decision process(MDP) to describe time-varying network conditions in HAM [25]. Mininet was used to simulate and perform security analysis to evaluate the effectiveness of ID-HAM based on four metrics - SMT formalization, defense performance, convergence performance, and network performance. The paper examined reconnaissance attacks and considers the scanning process of the adversary. The proposed algorithm decreased a maximum of 25% times scanning hits while only influencing the quality of service(QoS) of communication slightly. The adap-

tivity of ID-HAM was also better than its two counterparts RHM and FRVM.

Chowdhary et. al. proposed a multi-agent reinforcement learning framework with a zero-sum game dynamic in a Software-defined Network for obtaining the optimal Moving Target Defense strategy [18]. Compared to previous work, the paper used domain-specific reward modeling which uses the CVSS score of vulnerability, the difficulty of compromising a vulnerability, the attacker and defender's effort, and the effect of MTD countermeasures as rewards. The modeling considered the performance impact induced by defensive countermeasures as a part of the reward modeling since deploying MTD actions can have some impact on the network. The experimental results suggested that the defender can mimic the attack's action using a reinforcement learning policy and obtain a higher reward using reinforcement learning. Hence, the proposed reinforcement algorithm outperforms the random MTD deployment strategy with uniform action probabilities.

Eghtesad et. al. proposed a multi-agent partially-observable Markov decision process for Moving Target Defense [13]. The paper combined the Deep Q-learning algorithm and Double Oracle algorithm to find the optimal MTD by solving the mixed strategy Nash equilibrium(MSNE) of a game and equates finding the MSNE of the game as the method to find adaptive MTD policies. The algorithm is experimented in a two-player general-sum game between the attacker and the defender. The results showed that the double oracle algorithm converges around four pieces of training for each player, thus demonstrating that it can efficiently find the MSNE. The experiment also showed that using heuristics and NoOP as initial policy spaces resulted in very similar payoffs, which indicated that the algorithm can converge to near-optimal solutions regardless of the initial policy space.

Sengupta et al. proposed a Bayesian Strong Stackelberg Q-learning (BSS-Q) approach to advance upon previous works where incomplete information and the complexity of the adversaries were not considered [26]. The paper showed that BSS-Q converges to a game theoretic model called the Bayesian Stackelberg Markov Games (BSMGs) that models uncertainty over attacker types. The strategy of the defender starts with a uniform random combination of heuristic strategies and alternates between different strategies until it converges to Strong Stackelberg Equilibrium (SSE) for the BSMG. Two experimental scenarios were also conducted using MTD for web applications and cloud networks, demonstrating that using BSS-Q performs better than existing baseline solutions. However, the paper only considered single follower types and future research is needed to consider other possible attacker types.

Yao et al. proposed the WoLF-BSS-Q algorithm, which can adjust its strategies according to the game process while maintaining its performance compared to classic reinforcement algorithms [27]. The BSS algorithm is used to select the strategy for MTD, and the WoLF algorithm is used to dynamically adjust those strategies by the game process. The approach experiments in an OpenAI Gym-style multi-agent game simulator. The WoLF-BSS-Q algorithm is the optimum and converges to the maximal rewards compared to Nash-Q and URS-Q. The algorithm outperformed due to its ability to select the action with the greatest reward by adjusting the action selection probability. However, the attack agent was straightforward and could not identify the dynamic defense strategy of the defense agent. Furthermore, it did not consider the incomplete information aspect of the multi-agent Markov game, which can cause deviation in the results.

2.2.1 Discussion

Research towards using automation to deploy MTD against multiple adversaries remains deficient, mostly considering only two attacker types, often in the reconnaissance stage [24]. Furthermore, the complexity of the adversary is often lacking and not adaptive enough. For instance, the model described in literature [27] does not consider cheating as a method to consume defense resources and cannot identify the dynamic defense strategy of the proposed defense agent. Another study [24] acknowledges the necessity for ongoing research on characterizing different attacker types. Although research has been conducted on advanced adversaries [28], they have not been used to assess the effectiveness of intelligent MTD deployment. This literature [28] categorizes intelligent attacks targeting MTDs in an SDN environment into two main types: target attacks aiming to determine MTD properties and existing attacks improved through intelligent attack planning. Even though there have been considerations for using reinforcement learning to deploy multiple MTDs or heuristics in previous works [15, 18], there has been limited research done on it.

2.3 Security Metrics

Metrics assessing the effectiveness of Moving Target Defense (MTD) techniques can be categorized as either static or dynamic [29]. Static metrics concentrate on providing an overall assessment of security measures, while dynamic metrics evaluate the adaptability and efficacy of MTD strategies against intelligent adversaries.

1) Static metrics: Alavizadeh et al. have incorporated commonly used static metrics including the attack cost(AC), return on attack(RoA), system risk(R) and reliability into evaluating the security of the cloud environment within the context of the Hierarchical Attack Representation Model (HARM) [23, 30, 31, 32]. Wenxiao Zhang further extended the time domain aspect of MTDSim [17] and updated the implementation of attack cost and introduced the Mean Time to Compromise metric [16].

2) Dynamic metrics: Hong et al. used a Temporal Hierarchical Attack Representation Model(T-HARM) to evaluate the effectiveness of security metrics for dynamic networks [19]. The assessment included path metrics such as mean of attack path lengths(MAPL) and number of attack paths(NAP) and demonstrated that different security metrics can respond to changes in the network configuration by changing their values. Sharma et al. proposed a set of dynamic security metrics that examine the effectiveness of moving target defense techniques in software-defined network(SDN) domain [33]. Three security metrics were proposed: network and host address-based metrics, attack path-based metrics, and attack stage-based success metrics.

2.3.1 Discussion

Dynamic metrics are particularly valuable given the adaptive and dynamic aspects involved in this project. Path-based metrics, such as attack path variability (APV) [19], and attack stage-based success metrics, including exploit success probability (ESP) [19] and attack impact (AI) [19], have not received much research attention in terms of their incorporation into reinforcement learning models for deploying MTD. This project aims to leverage both static and dynamic metrics as features for the proposed model.

3 Methodology

3.1 System Overview

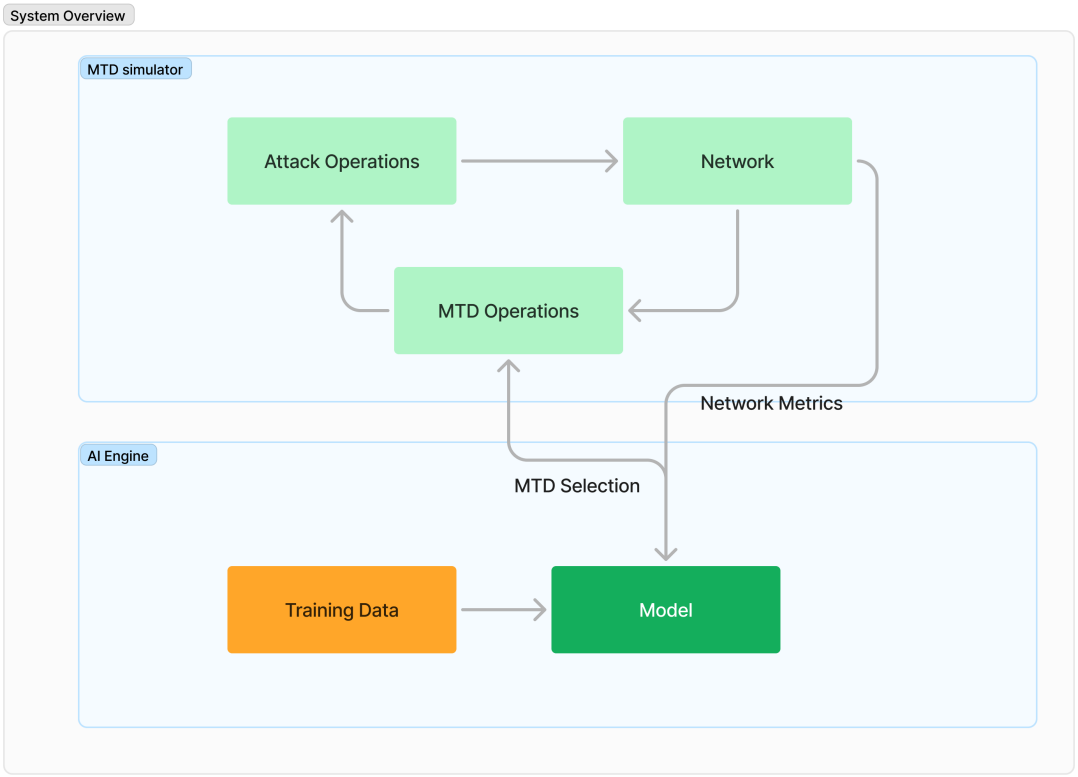

Figure 1: System Overview

As shown in Figure 1, the system consists of the AI engine and the MTD simulator. The AI engine is the proposed model of the project. In this project, the MTD simulator MTDSimTime [16] is used as the infrastructure to create the MTD AI model. It is used as a platform for the MTD AI model to interact with the adversary. It is also used for collecting results for the evaluation of the model.

3.1.1 Environment and Players

There are two players, the defender and the adversary. As shown in Figure 1, the interaction between them will occur in the MTDSimTime simulator [16]. The simulator uses a 3-layer HARM model which consists of the Host Attack Graph(AG), the Services on the Host Attack Graph(AG), and the Service Attack Tree(AT). Currently, the simulator can only deploy four MTD techniques either randomly, alternatively, or simultaneously against limited attacker types. In this project, this simulator will be extended to include more purposeful deployment and also consider a wide range of MTD techniques. Adversary was built using the Cyber Kill Chain [20] and MITRE ATT&CK [21] framework to try to compromise as many hosts as possible.

3.1.2 Static Degrade Factor

The default value for static degrade factor is set to be 2000ms. The 2000ms threshold is not as aggressive as 1000ms and it can allow more meaningful and proper deployment of MTD to avoid disrupting the previous deployment. When period between the last MTD trigger and the current time exceed the static degrade factor, the system will be forced the trigger a random MTD based on the selected MTD deployment type and reset the interval counter. This check will happen before the network metrics being feed into the MTD AI model. This approach ensures periodic system changes that enhance security by preventing attackers from gaining too much familiarity with the state of the network. The assumption is that when the network stay static for too long it will become more vulnerable to attackers. Having a dynamic network topology can reduce the attack surface of the system[24].

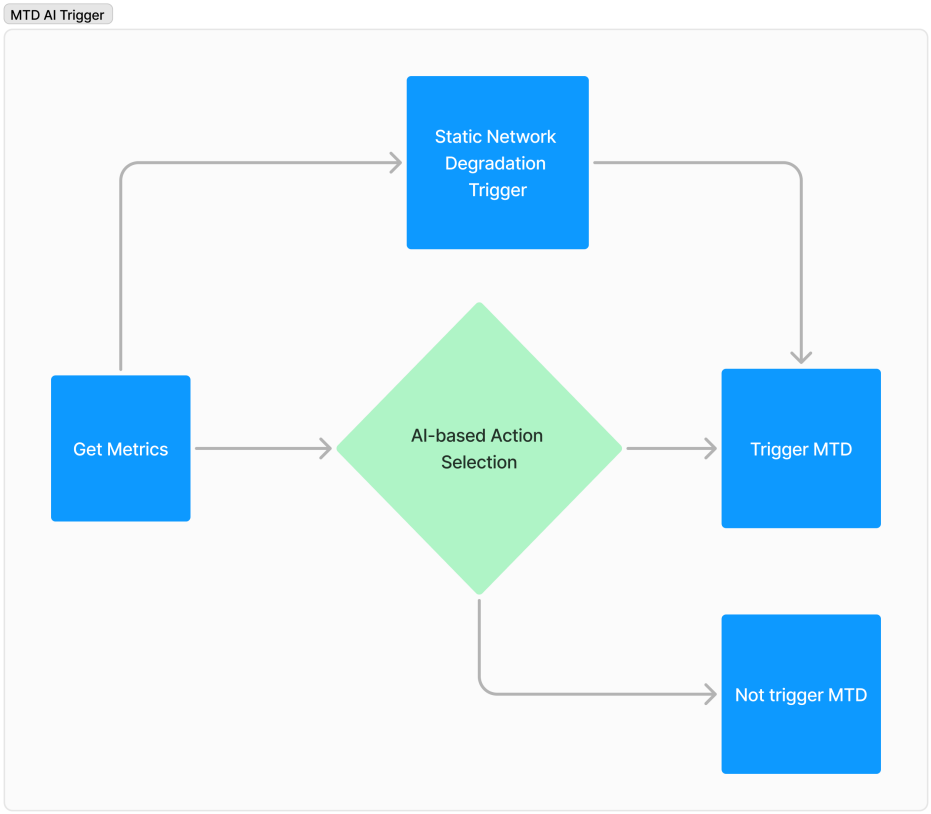

Figure 2: MTD Trigger Procedure

3.2 Model

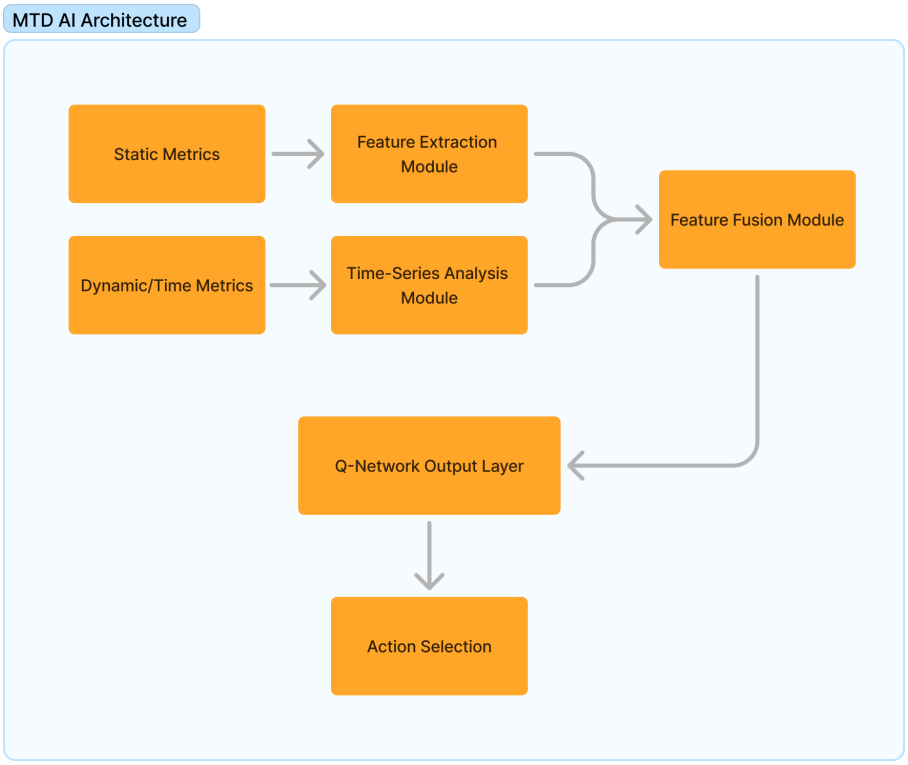

Figure 3: MTD AI Model Architecture

3.2.1 Double Deep Q-network learning

The reinforcement model that this project will be using is the double deep-Q-network model. Compared to other conventional reinforcement models such as the Stackelberg Q-learning model, the deep Q-learning model is more flexible and easier to adapt to different frameworks [34]. The advantage of double Q-learning is that it reduces the bias caused by single Q-learning by maintaining two separate Q-value estimates. This prevents overestimation since two Q-values are independently updated [35]. In the original model Q-learning, the aim is to learn the state-action-value(or Q) function, which measures the overall expected reward for taking action a(t) at state s(t) with $r(t) \in \mathbb{R}$ being the intermediate reward as listed in Equation (1). However, when states and actions are broken down into smaller, specific parts, they can lead to a large number of combinations. This can create a problem called the curse of dimensionality [36].

$$Q(s(t), a(t)) = \boldsymbol{E}_{\boldsymbol{\pi}} \left(\sum_{r=0}^{\infty} \gamma^{\tau} r(t + \tau + 1) \mid s(t), a(t) \right) \tag{1}$$

Deep Q-network improves Q-learning by approximating the Q-function with a neural network. The deep-Q-network learns by minimizing the loss function presented in Equation(2) and seeks to find an optimal policy π^* that produces the maximum

expected reward from all states where $\pi^* = arg\max \mathbb{E}[R|\pi]$ [37].

$$\mathbb{L}(t) = \frac{1}{2}(\underbrace{r(t) + \gamma \max_{a \in \mathbb{A}} Q(s'(t), a, \theta(t))}_{\text{target}} - \underbrace{Q(s(t), a(t), \theta(t))}_{\text{actual}})^2 \tag{2}$$

3.2.2 Neural Network Architecture

Figure 4: Feature Fusion Module

There are two feature modules in the architecture to handle two different types of features as shown in Figure 4. Static feature extraction module is used to process static features, which are the features that do not change over time (e.g. host compromised ratio). It contains two fully connected dense layers with 128 and 64 neurons each and two respective ReLU activation. The dense layers apply linear transformation to the input while ReLU activation introduces non-linearity to ensure only positive numbers are passed down. The purpose of the dense layers is to learn about the relationships between static features of the input data while ReLU activation can increase complexity. Two batch normalization layers are used to ensure stability. The time series module is used to process dynamic features, which are features that are more adaptive and

reflect the current state of the network(e.g. MTD frequency). This module consists of two LSTM (Long Short-Term Memory) layers that have 64 and 32 units respectively along with activation, normalization and dropout layers. The purpose of LSTM is to counter the vanishing gradient problem in traditional RNN(recurrent neural network) so that it can process long periods of small time steps. The 30% dropout is to ensure training efficiency and reduce overfitting. Both these feature modules will be feed into the feature fusion module which will go through the final dense and ReLu layers. The feature fusion module will then go through the Q-Network output layer which will calculate the reward and update the model.

3.2.3 Actions

The action a^t is represented by $a^t = \{MTD_1, MTD_2, \ldots, MTD_{12}, MTD_{13}, \ldots, MTD_{\sum_{i=1}^{n}}, \text{Null}\}$ where MTD_i is the deployment of the i^{th} MTD technique in the MTD set. MTD_ij represents deploying both the i^{th} and j^{th} MTD technique such as Figure(2) [15]. Null means no actions will be taken.

3.2.4 Rewards

The rewards for the model should be designed in a way that trains the model to minimize system vulnerabilities. The reward at step t is given by $R(s_t, a_t, s_{t+1})$, where $R(s_t, a_t, s_{t+1}) = f(N_{t+1}) - f(N_t)$ where $f(x)$ is an evaluation function. The accumulative reward is categorized as $R = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$. Only the selected features will have an affect on the calculation of the optimization of the rewards. Metrics optimization that seeks to optimize multiple metrics will be using a weighted sum as the reward.

$$R_t = \sum_{i=1}^{n} w_i \cdot f_i(x_t) \tag{3}$$

where w_i is a constant, either positive or negative, based on the feature:

$$\begin{cases} w_i > 0 & \text{if } f_i \text{ reflects dynamic behavior of the network} \\ w_i < 0 & \text{if } f_i \text{ reflects vulnerabilities of the network} \end{cases} \tag{4}$$

To ensure equal contribution of each feature to the sum, each feature will be normalized using min-max normalization against previous data in the memory in each reward calculation. The reason why min-max normalization is used instead of other normalization is that it is assumed that the more samples it gets eventually most data will be in the middle and the normalization can be consistent since it does not affect the upper or lower boundary. The agent will not start learning from its experiences until it has collected at least 1000 samples (training start parameter). This ensures that the agent has a sufficient amount of varied experiences to learn from, rather than starting to train with too little data.

3.3 Experiment Setup

3.3.1 Fixed Parameters

Table 1: Learning Parameters

Parameter	Value
Discount Rate (γ)	0.95
Exploration Rate (ϵ)	1.0
Minimum Exploration Rate (ϵ_{min})	0.01
Exploration Decay Rate (ϵ_{decay})	0.995
Training Start	1000
Episodes	100

Table 2: Simulator Settings

Parameter	Value
Start Time	0
Finish Time	15000
Total Nodes	150
Static Degrade Factor	2000(ms)

The learning parameters regulate the reinforcement learning model's training, using a 0.95 discount rate to balance immediate and future rewards. The agent starts with full exploration and decays to a minimum of 0.01 at a rate of 0.995, beginning training after collecting 1000 samples over 100 episodes. Simulator settings run trials from 0 to 15000 to gather adequate data. The number of nodes in the network stays constant to reflect different density of the network when the network size parameters changes.

3.3.2 Features

The following metrics that will be used for the reinforcement learning model:

Table 3: Metrics and Symbols

Symbol	Description
h_i	The i^{th} host
h	A host in the system
AP	List of all possible attack paths in the network.
$R_h(t)$	Risk of compromising host h at time t
$CSP_h(t)$	Compromise success probability of host h at time t.
$AI_h(t)$	Attack impact of host h at time t.
RoA_h	Return on Attack Cost for host h
SAP	Set of shortest attack paths at time t.
$\|ap\|$	Length of the attack path ap.
$SAPV(t)$	Shortest Attack Path Variability at time t, indicating the changes in shortest attack paths over time.
n	Number of hosts on the shortest attack path
$V_{new}(h)$	New Vulnerability Percent for host h
C_t	Number of compromised hosts at time t
T_{host}	Total number of hosts in the network
A_t	Number of attempted attacks by adversary at time t
NAV_{t_k}	Network Address Variability at time t_k, indicating changes in the network addresses over time.
$AC(h_i)$	Attack cost is defined as the time to exploit
N_{MTD}	The number of executed MTDs
$Time_{now}$	Current time of the simulator
$Time_{LME}$	The time since last MTD is executed
`SCAN_PORT`	Scans a target system to identify open ports and services running on those ports.
`EXPLOIT_VULN`	Attempts to exploit a known vulnerability in a service or application to gain unauthorized access or control.
`BRUTE_FORCE`	Uses automated methods to guess passwords or encryption keys through repeated attempts.

1) Attack Path Exposure (APE): A measure of the exposure of a network's shortest attack path to the target node. The score evaluates each node along the path based on the percentage of newly discovered vulnerabilities, calculated as:

$$APE = \frac{\sum_{i=1}^{n} V_{new}(h_i)}{n} \tag{5}$$

where n is the number of hosts on the shortest attack path.
New Vulnerability Percent for a host h is calculated by:

$$V_{new}(h) = \frac{V_{total} - V_{not\_unique}}{V_{total}} \tag{6}$$

If no new vulnerabilities are found for a host h, $V_{new}(h)$ will be zero.

2) Risk (R): The risk metric represents the risk of compromising a host in a network at every interval where the model takes input data from the network configuration. The change of risk per interval(before and after moving target defense deployment) is measured as an indicator of the effectiveness of the MTD deployment. A quantitative definition of the risk of compromising a host, h_i at time t_k can be defined as:

$$R(h_i)_{t_k} = CSP(h_i)_{t_k} \times AI(h_i)_{t_k} \tag{7}$$

where $CSP(h_i)_{t_k}$ and $AI(h_i)_{t_k}$ are the compromise success probability and attack impact of a host h_i at time t_k.

3) Return on Attack Cost (RoA): This metric quantifies the ratio of the benefits of an attack to the costs incurred. A higher value of Return on Attack (RoA) indicates a greater likelihood that an attacker will exploit the vulnerabilities present. The Return on Attack Cost (RoAC) for a single host h_i is calculated as the ratio of the risk associated with the host to the attack cost as shown in Equation (3). The overall RoA of a system is given in Equation (4).

$$RoA_{h_i} = \frac{R(h_i)_{t_k}}{AC(h_i)} \tag{8}$$

$$RoA_{\text{sys}} = \sum_{ap \in AP} \left(\sum_{h_i \in ap} RoA_{\text{sys}_{h_i}} \right) \tag{9}$$

with the attack cost defined as the time to exploit.

4) Host Compromised Ratio (HCR): The ratio of hosts that are compromised in the system. The ratio is calculated by:

$$HCR = \frac{C_t}{T_{host}} \tag{10}$$

where C_t is the number of compromised hosts at time t and T_{host} is the total number of hosts in the network.

5) Attack Stage: The current attack stage of the attacker. Integers are used to represent each stage of the attack. The attack stages ranges from `SCAN_PORT` to `BRUTE_FORCE`. If currently there is no attack the default number will be used.

6) Attack Success Rate (ASR): The attack success rate measures how effective the attacks by adversaries are. It is calculated by the percentage of attacks by adversaries that successfully compromised hosts compared to the overall attempted attacks.

$$ASR = \frac{C_t}{A_t} \tag{11}$$

where C_t is the number of successfully compromised hosts at time t and A_t is the number of attempted attacks at time t, including actions like `SCAN_PORT`, `EXPLOIT_VULN`, and `BRUTE_FORCE`.

If no compromised hosts are recorded, the attack success rate is zero.

7) MTD Execution Frequency (MEF): The frequency with which Moving Target Defense (MTD) actions are executed. It is calculated by:

$$\text{MEF} = \frac{N_{MTD}}{\text{Finish time}_{\text{last}} - \text{Start time}_{\text{first}}}$$

Where $\text{Finish time}_{\text{last}}$ is the finish time of the last MTD action, $\text{Start time}_{\text{first}}$ is the start time of the first MTD action and N_{MTD} is the number of executed MTDs.

If there are no MTD executions, the frequency will be zero.

8) Overall Mean Time to Compromise (MTTC): It represents the mean time taken for the adversary to successfully compromise the hosts. It average of the durations of attack events of `SCAN_PORT`, `EXPLOIT_VULN`, and `BRUTE_FORCE` for all relevant hosts. The calculation is as follows:

$$\text{MTTC} = \frac{\sum_{\text{events}} \text{Duration}(\text{SCAN\_PORT}, \text{EXPLOIT\_VULN}, \text{BRUTE\_FORCE})}{N}$$

Where N is the total number of relevant attack events for the specified hosts. If no attack events are present, the MTTC is set to zero

9) Time Since Last MTD (TSLM): The time between now and the last MTD execution time.

$$\text{Time Since Last MTD (TSLM)} = \text{Time}_{\text{now}} - \text{Time}_{\text{LME}}$$

where $Time_now$ is the current time of the simulator and $Time_{LME}$ is the time since last MTD is executed in the simulator.

10) Shortest Attack Path Variability (SAPV): The shortest path represents the length of the shortest attack paths from an attacker's initial state to the goal state. The SAPV metric estimates the changes in shortest attack paths over time. A set of the shortest attack paths at t_k estimated by:

$$SAP_{t_k} = \{ap_{t_k} \mid |ap_{t_k}| \leq |ap| \forall ap \in AP_{t_k}\} \tag{12}$$

where ap is the attack path and $\mid ap \mid$ is the length of the attack path.

The SAPV at t_k is estimated by:

$$SAPV_{t_k} = \frac{|SAP_{t_k} - SAP_{t_{k-1}}|}{|SAP_{t_k}|} \tag{13}$$

The overall SAPV for $[t_1, t_m]$ is calculated as:

$$SAPV = \frac{1}{m-1} \sum_{k=2}^{m} SAPV_{t_k} \tag{14}$$

11) Network Address Variability (NAV) : This measures the change in the network IP addresses over time. This changes of the IP addresses can be captured by the network states (i.e., NStk) at a different time point tk. The variability of the two consecutive network states is computed by:

$$NAV_{t_k} = \frac{|N_{S_{t_k}} - N_{S_{t_{k-1}}}|}{|N_{S_{t_k}}|} \tag{15}$$

where "$-$" represents the set difference operation and "$|\cdot|$" denotes the cardinality of the set. This equation measures the number of address changes in $N_{S_{t_k}}$ normalized by its size. NAV_{t_k} is estimated based on how the network state changes over time and takes values in the range $[0, 1]$ [29].

3.3.3 MTD Selection

This experiment utilizes four distinct Moving Target Defense (MTD) techniques, along with one overarching MTD technique that can deploy any of the four strategies. The techniques are as follows:

- **Complete Topology Shuffle:** Completely regenerates the network while preserving the hosts from the previous network.
- **IP Shuffle:** Assigns existing hosts a new random IP address.
- **OS Diversity:** Switches between different Operating System types and versions across the network.
- **Service Diversity:** Updates the services running on the nodes of hosts within a network.

3.3.4 Overview

Table 4: Combinations of Models

Metric/MTD	Any MTD	Complete Topology Shuffle	IP Shuffle	OS Diversity	Service Diversity
All Features					
Hybrid					
Host Compromise Ratio					
Attack Path Exposure					
Attack Success Rate					
Return on Attack					
Risk					
MTD Frequency					
Mean Time to Compromise					
Time Since Last MTD					

Table 5: Comparison of Metrics in Hybrid and All Features Optimization

Metric	Hybrid	All Features
Host Compromise Ratio	Yes	Yes
Attack Path Exposure	Yes	Yes
Attack Success Rate	Yes	Yes
Return on Attack	Yes	Yes
Risk	Yes	Yes
MTD Frequency	Yes	Yes
Mean Time to Compromise	Yes	Yes
Attack Stage	Yes	Yes
Time Since Last MTD	No	Yes
Shortest Path Variability	No	Yes
Host IP Variability	No	Yes

Table 6: Parameters for testing on different MTD Intervals

MTD Interval	Network Size	Number of Nodes
50	150	150
100	150	150
200	150	150

Table 7: Parameters for testing on different Network Sizes

MTD Interval	Network Size	Number of Nodes
50	100	150
50	150	150
50	200	150

Table 8: Comparison with Previous Schemes

Models	Random	Alternative	Simultaneous	MTD AI
MTD Interval	200	200	200	200
Network Size	150	150	150	150
Number of Nodes	150	150	150	150

In total 50 models will be trained for each intersection between MTD techniques and metrics optimization. Each model will try to optimize on the specific metric(s) in the reward and will only deploy the specific type of MTD (unless it is all MTD which means the model is able to deploy any of the 4 MTD types). For example, there is one model in the table that will only optimise the metric return on attack in its reward and will only deploy the MTD technique IP Shuffle. These 50 models will in addition be tested across different MTD intervals and network sizes as shown in Table 8. The reason why different network sizes are tested in low MTD interval is because previous work indicates MTD deployments works better in lower interval [16].

Static metrics such as host compromise ratio, risk and roa can capture the inherent vulnerabilities of the network. Time metrics such as MTD freuqncy and time since last MTD can reflect the immediate changes in the network, including potential attacks. The hybrid metric optimization include metrics host compromise ratio, attack stage, attack path exposure, attack success rate, return on attack, risk, MTD frequency and mean time to compromise. The all features optimization include host compromise ratio, attack stage, attack path exposure, attack success rate, return on attack, risk, MTD frequency, mean time to compromise and in addition time since last mtd, shortest path variability and host IP variability. The all features metrics optimization is used to test whether path and network path based variation will improve the performance.

3.4 Evaluation Method

3.4.1 Results collection pipeline

In order to ensure consistency, five checkpoints are used to collect the evaluation metrics for each trial. The mean of the checkpoints will be taken as the representative of that trial and the overall results for each models will be taken by the median of the trials.

3.4.2 Evaluation metrics calculation

A final evaluation score will be calculated using four evaluation metrics. Each evaluation metrics will be of equal weighting when contributing to the sum. The four evaluation metrics are Attack Success Rate (ASR), Return on Attack (ROA), Attack Path Exposure (APE), Risk (R). Trials with no MTD deployment will be run so that these metrics can be obtained for the baseline scheme. The metrics results collected by the trained models will be scaled accordingly. These valuation metrics should be minimised since they represents disadvantages or vulnerabilities of the network. The purpose of these scaling is so that the final evaluation score(sum) will be consistent. A larger score will indicate a better performance and vice versa. The calculations for scaling (whether maximise or minimise) is calculated by:

$$\text{Maximize: } = \frac{V}{N}$$

$$\text{Minimize: } = \frac{1}{\left(\frac{V}{N}\right)} = \frac{N}{V}$$

where V is the raw metric value obtained by the trained models and N is the baseline (no MTD) value. Since there are no existing similar AI models to compare to, the MTD AI scheme will be compare to three previous schemes - alternative, random and simultaneous. The evaluation score for other schemes will be calculated similar to the trained MTD AI models for consistency.

4 Results and Discussion

4.1 Impact of MTD Interval

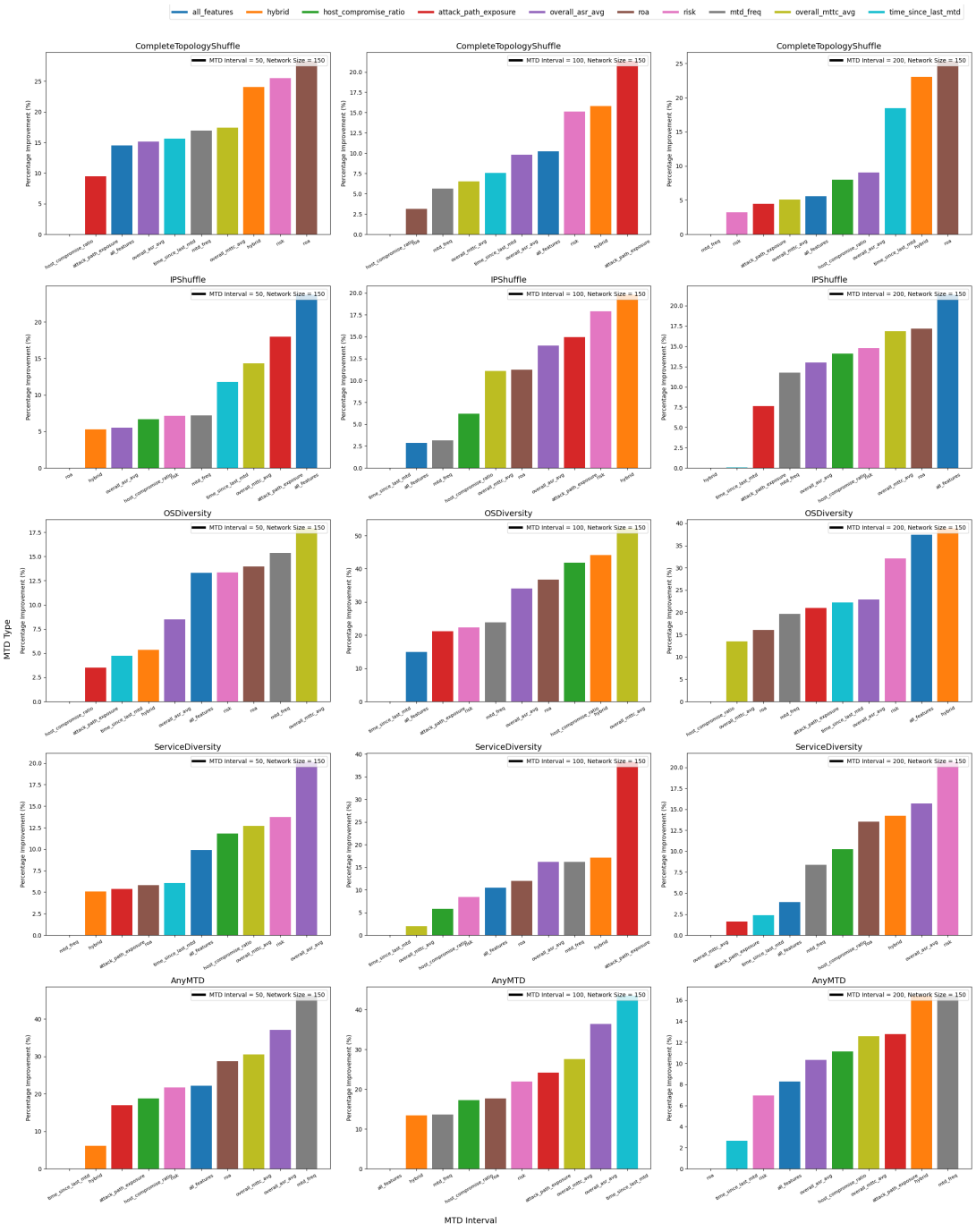

Figure 5: Performance between different metrics optimization across different MTD intervals and MTD types (Normalized)

4.1.1 Comparing different metric optimizations across different MTD intervals

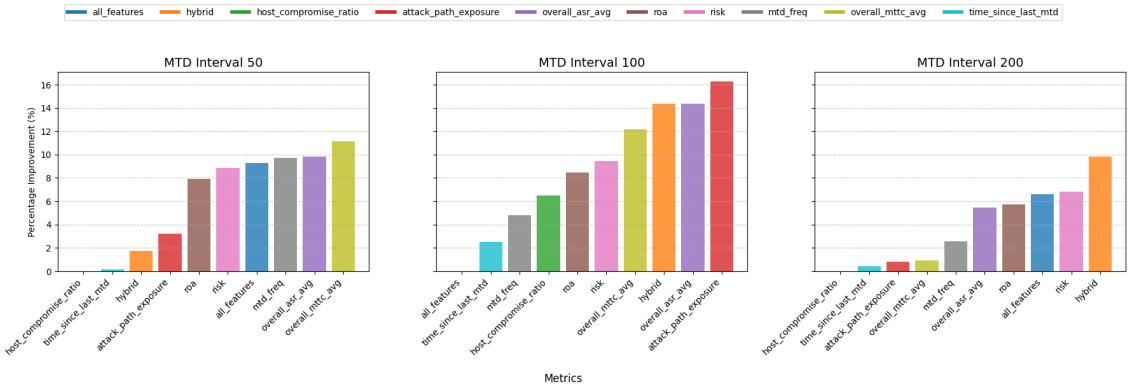

Figure 6: Average percentage improvement between different metric optimization over different MTD intervals (Normalized)

Figure 6 displayed the average percentage performance of each metric across all MTD techniques which uses that metric as optimization. Each score is normalized against the worst performing score in each graph. The range between the best and worst metrics optimization average performance are 11.1%, 16.2%, and 10% respectively for MTD interval = 50, 100, and 200.

There are only few individual metrics optimization that consistently performed well as shown in Figure 5. The overall mean time to compromised optimization showed an advantage consistently over other metrics in the MTD interval = 50 across all MTD techniques in particular when coupling with OS diversity technique. While under performing with shuffling technique in general, the overall ASR optimization performed decently across lower intervals when it can deployed any MTD and it coupled well with Service diversity technique in MTD interval = 50 as well. However, TSLM and HCR performed the worst in comparison to other metrics, especially when TSLM is used with IP shuffling technique or in MTD interval = 100 as shown in Figure 5 and 6.

Overall the results show that besides the performance of MTD interval = 50, the hybrid combination of metrics generally outperformed other metric(s) optimization as shown in Figure 5. It also performed consistently well when used with Complete Topology shuffling technique across all MTD intervals. All features optimization performed well with IP shuffling technique in extreme intervals.

However, it is noticed that the performance of the all features combination did not perform as well as expected in general since even as it incorporate more dynamic features compare to the hybrid counterpart. Moreover, even though it included the SPV it did not seem to perform better than other metrics when using complete topology shuffle (which directly affects the shortest path). Interestingly, it is able to out-performed all other metrics in extreme intervals when used with IP shuffle.

4.1.2 Overall impact of MTD Interval

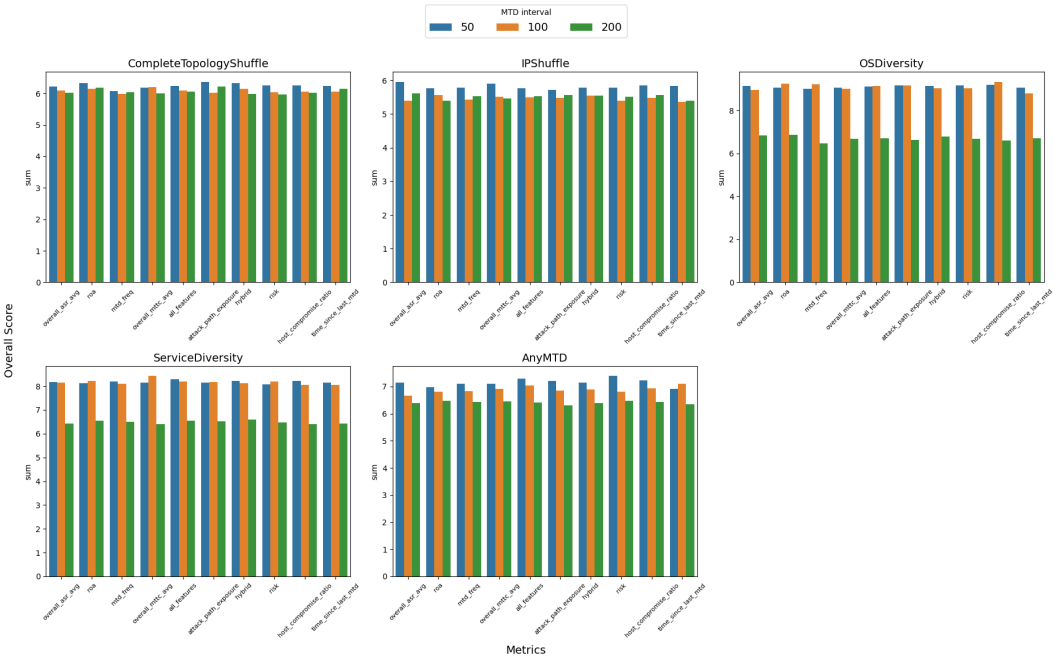

Figure 7: The effect of MTD Interval and MTD type on metrics optimization

Figure 7 shows that both diversity MTD techniques underperform in MTD interval = 200 compare to the lower intervals. All models under-performed in MTD interval = 200 especially with models that deployed diversity techniques. The IP shuffling technique works well with all different metrics optimization in MTD interval = 50, reaching up to around 50% performance differences in different intervals (i.e.overall ASR average). When models are able to deploy any types of MTD, there was a clear invert correlation between the size of the MTD interval and the performance of the models (i.e. the smaller interval the better).

4.1.3 Discussion

MTTC can performed better at around 12% to 17% compare to the worst model in MTD interval = 50 consistently across all MTD techniques because under shorter intervals gernerally more MTD can be deployed. MTTC better reflects the individual attack exploit time at lower intervals because it captures the specific disruptions caused by rapid MTD deployments. In contrast, when MTTC is measured over longer intervals, it reflects a broader period that includes more attack phases, making it less responsive to dynamic changes. The high frequency of OS changes ensures that attackers face a moving target, significantly increasing the time needed to find vulnerabilities and execute successful attacks. This aligns perfectly with the goal of MTTC which is to measure the average time it takes for attacker to compromise the network.

On the other hand, while under performing with shuffling technique in general, the overall ASR optimization performed decently across lower intervals when using diversity technique or when it can deployed any MTDs compare to other metrics which is shown in both Figure 5 and 6. This might be because diversity technique can be deployed more frequently compare to shuffling technique since they are less resource intensive.

When used with ASR (which reflect how effective the attacks are) in low interval, the model can detect any ongoing attack and immediately cut off the attacks.

The poor performance of HCR and TSLM might be because these HCR is less sensitive to frequent changes in system configurations and TSLM may not reflect significant changes in security posture with respect to certain MTD techniques. For example, TSLM tracks time since the last MTD action and is able to capture dynamic changes in the network, but these changes might not be useful or reflective of the network security posture or possible vulnerabilities.

The advanced performance of the hybrid optimization models is because optimizing for multiple aspects of security (e.g., MTTC, ASR, SPV, etc.) creates a more balanced defense strategy that covers different types of vulnerabilities or attack methods. The all features combination under-performed probably because it is diluted by the poor performance of TSLM, although it did not explain why it performed so well when coupling with IP shuffle (except in medium intervals). As the performance of TSLM got better in MTD interval = 50, the performance of all features optimization was shown to be better than hybrid in Figure 5 and 6.

4.2 Impact of Network Size

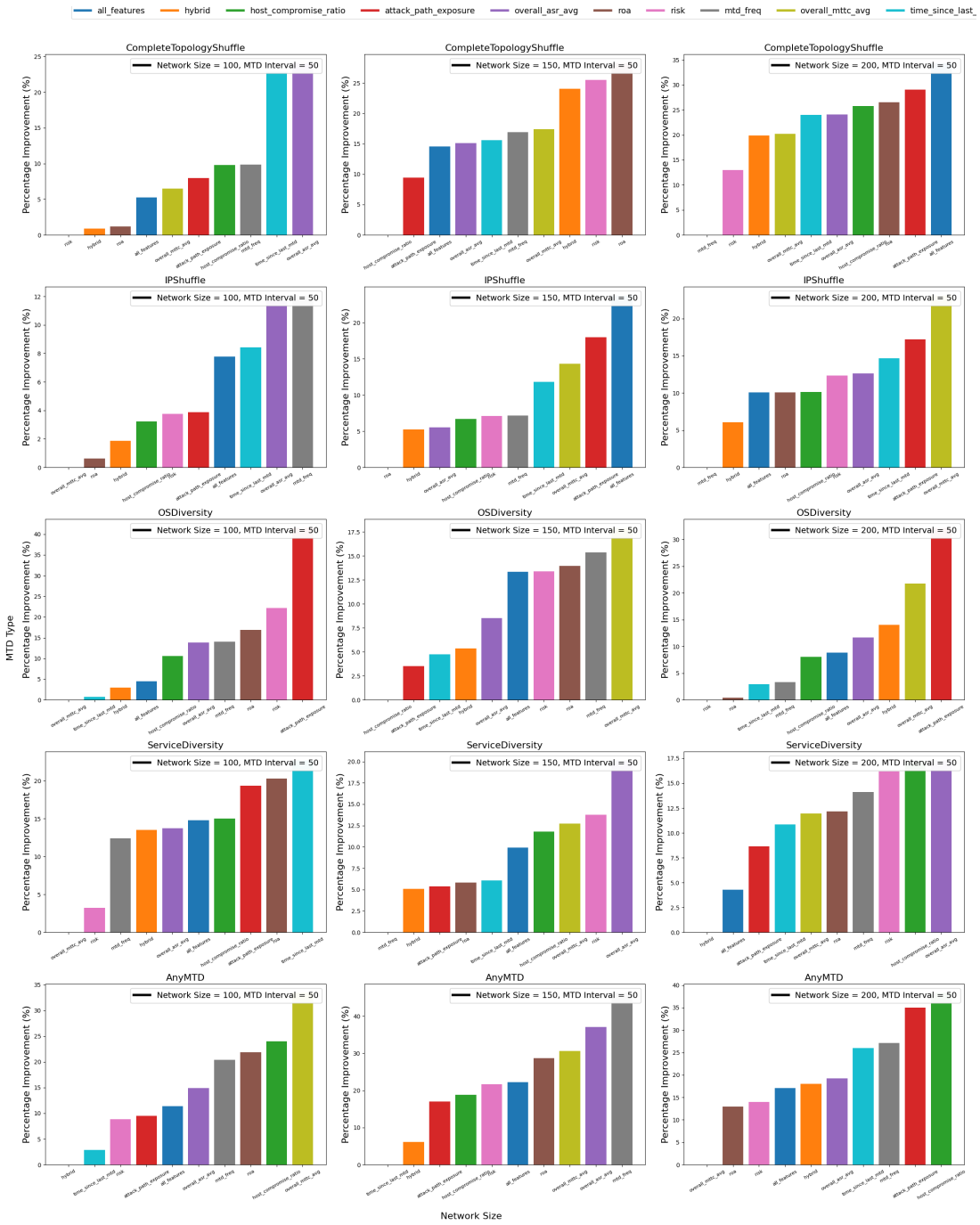

Figure 8: Performance between metrics optimization across different Network Sizes and MTD types (Normalized)

4.2.1 Comparing different metric optimizations under various Network Sizes

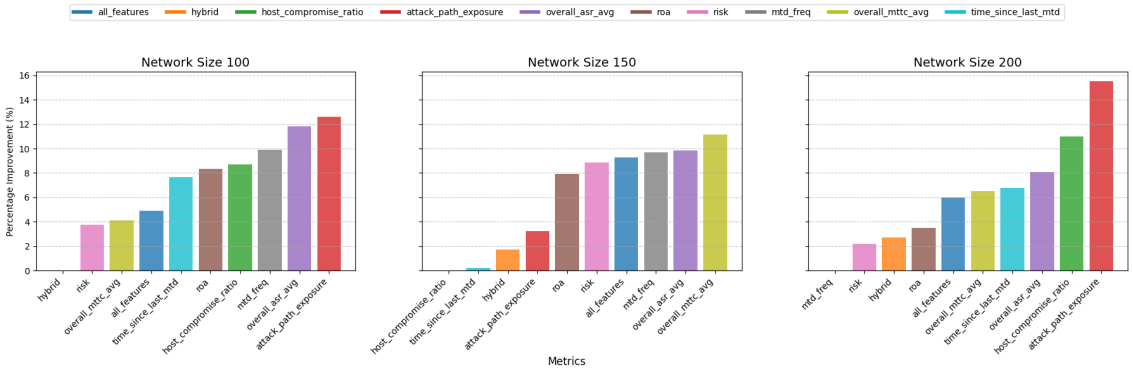

Figure 9: Average performance for each metric optimization across different network sizes (Normalized)

Similar to the differences in performance in MTD intervals, there were only very mild differences across the metrics as shown in Figure 9. It showed a maximum of 12.6% , 11.1% and difference in range on average for network size = 100, 150 and 200 respectively. In Figure 9 and 10, it is demonstrated that APE performed well in Network size = 200. Although on average APE performed well in network size = 100, the performance varies largely for individual MTD techniques. Previous MTTC is observed to perform well in MTD interval = 50 in Figure 5, but when tested against other network sizes it did not maintain the performance. It achieved poor results specifically when deployed with diversity technique and IP shuffle in network size = 100 and is only 4% better than the worst metric on average. Time metrics such as MTD frequency and TSLM along with overall ASR optimization performed well in comparison to other metrics when deployed with shuffling technique in small network, ranking the top 3 metrics optimization in both shuffling technique. ASR managed to do well across all tested network sizes compare to other metrics, consistently ranking second or third. However, the hybrid combination optimization still maintain its poorly performance in MTD interval = 50 across all network sizes.

4.2.2 Overall impact of Network Size

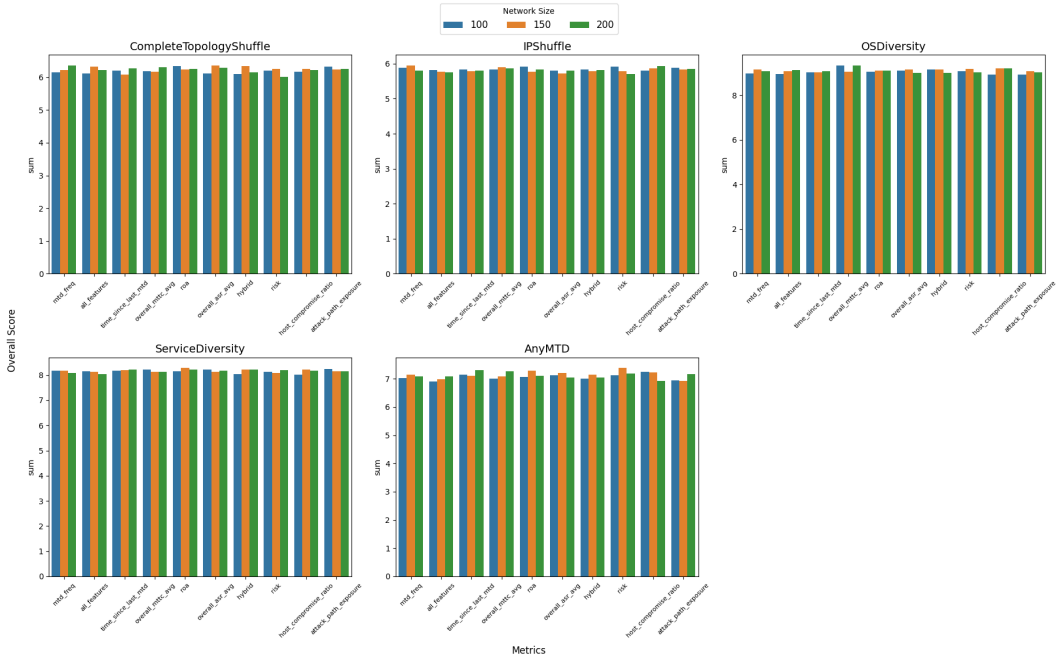

Figure 10: The effect of Network Sizes and MTD types on metrics optimization (normalized scores)

As shown in both Figure 11 , different network sizes did not appear to have any significant overall impacts on models across different metrics optimization or MTD techniques. Some more noticeable impact can be observed when TSLM is deployed in extreme network sizes(small and large) with OS diversity, it managed to improve compare to network size = 150.

4.2.3 Discussion

APE measures the percentage of new vulnerabilities in nodes along attack paths. In larger networks, the lower connection density per node (since the number of nodes in the setting did not change) results in fewer connections per node. This sparser distribution decreases the probability that attackers can easily reach critical nodes or compromise the network. Attackers must traverse longer paths with more intermediate nodes to reach high-value targets, increasing the difficulty of compromising the network and creating more opportunities for detection. When combined with low MTD intervals, APE can alert the model vulnerable paths more frequently and strategically deploy MTD to thwart attackers before they reach critical nodes.

MTTC optimizations did not perform ideally in small network compare to when in regular network size might be because in small networks shorter attack paths generally correlates to short compromise time and little variation in compromise time between attacks. However, it did managed to do well in large networks when deployed with OS diversity or IP shuffle.

Time metrics such as MTD frequency and TSLM can capture dynamic temporal changes in the network and works well when optimized with shuffling techniques since the higher connectivity and shorter attack paths in small networks make these techniques more effective.

Finally, overall ASR performed well across all network sizes with low MTD intervals because these intervals result in frequent configuration changes that the ability of attackers to exploit vulnerabilities which leads to a consistently lower ASR, allowing models to effectively evaluate the impact of the deployed MTD strategies in real time. The adaptability of this approach ensures robust security across various network environments.

4.3 Impact of MTD Type

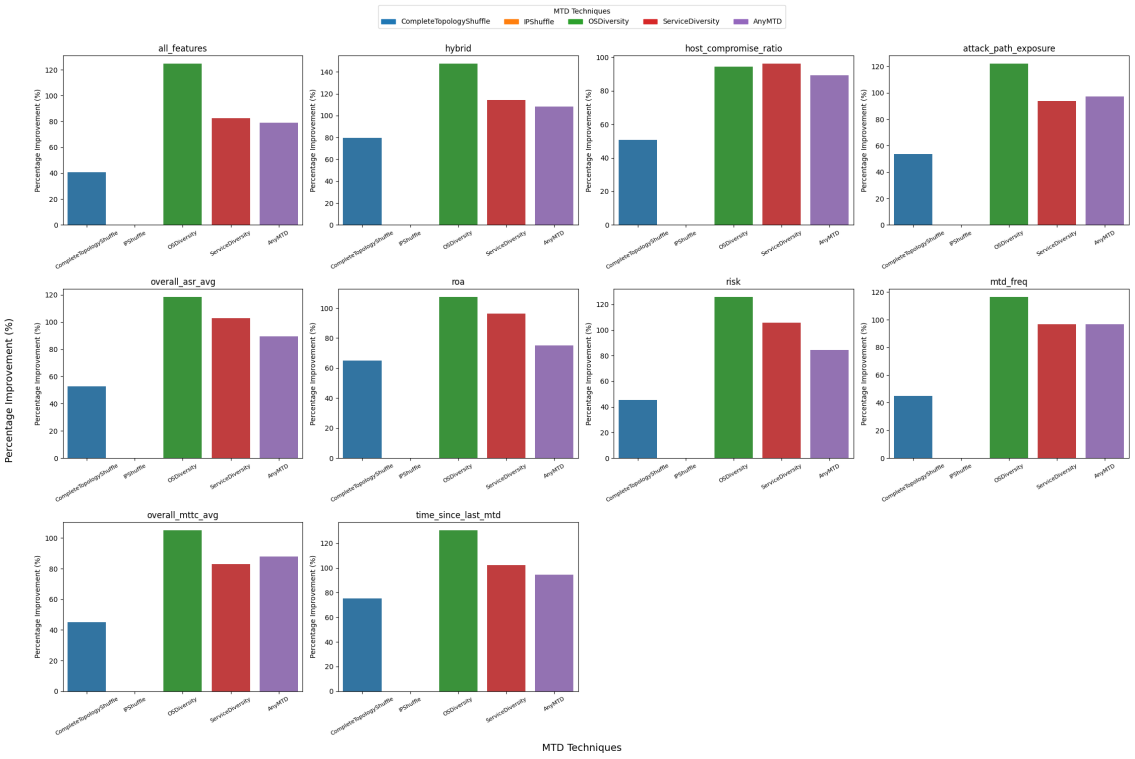

Figure 11: Performance of different MTD techniques across different metrics optimization in MTD interval = 200 and Network size = 150 (normalized scores)

As observed in Figure 11, the general difference in using different MTD techniques across all metrics were very similar in MTD interval = 200 and network size = 150. Diversity techniques outperformed shuffling technique in every single metric optimization. The worst performing MTD technique is IP shuffling and the best is OS diversity, which can outperform IP shuffling by about 140% (i.e. hybrid). When compare to the effect of MTD interval or network size, the choice of MTD technique place appears to be the most defining factor that determine the performance of models.

This discrepancy might be explained by the difference in resource intensity and deployment frequency. As seen before in Figure 7, there is a huge impact by the MTD interval on diversity technique. Diversity technique is able to perform a lot better in shorter interval while this did not seem to affect shuffling technique as much. This is because shuffling technique is more resource intensive and requires more time to deploy and reconfigure. However, further research is needed to clarify this speculation.

4.4 Comparison with previous MTD schemes

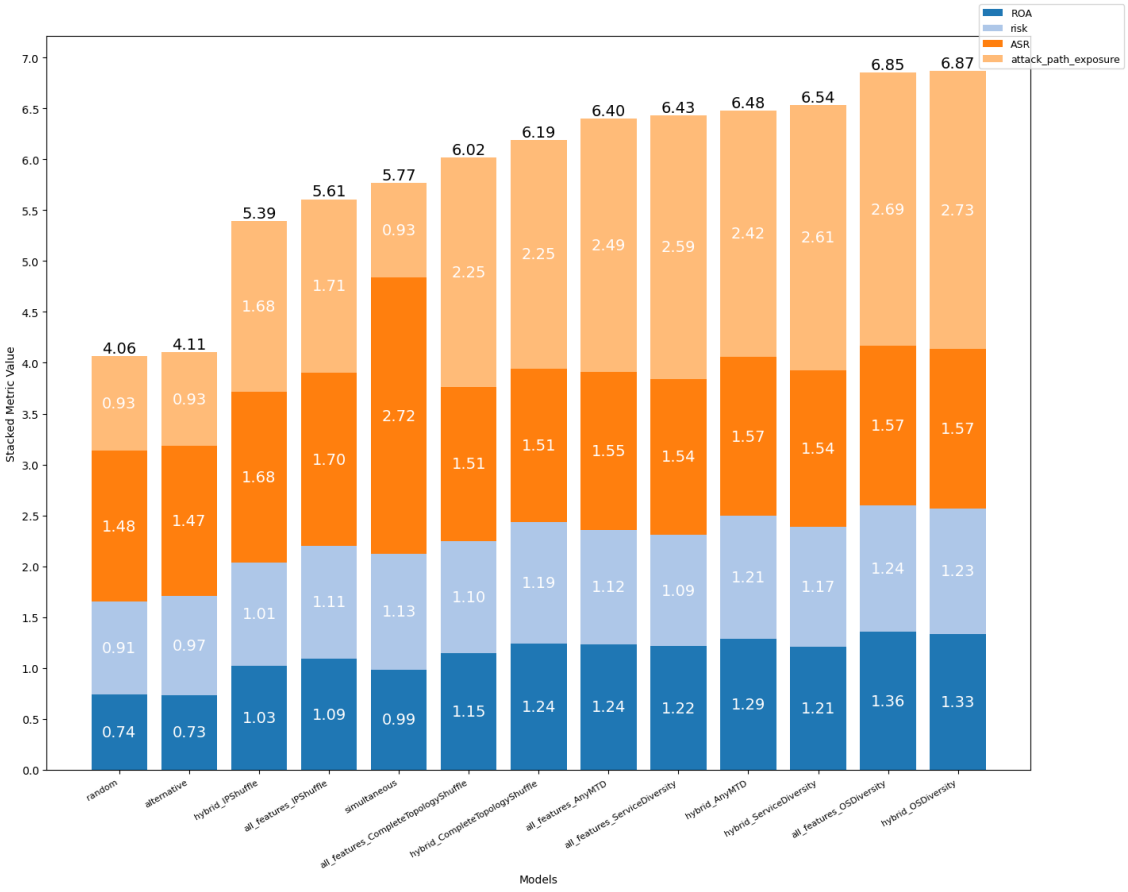

Figure 12: Comparing current models to existing schemes (actual scores)

According to Figure 7, the hybrid combination on average performed the best in MTD interval = 200. Therefore, it was compared to the random, alternative, and simultaneous schemes from previous work. At first glance, most of our new models significantly outperformed the simultaneous scheme by at least 30% overall, and none of the proposed models under performed compared to the random and alternative schemes. However, models that used IP shuffling exhibited lower performance compared to both the simultaneous scheme and other diversity techniques. Notably, the simultaneous scheme excelled in the ASR evaluation metric, outperforming all other models by a wide margin. This result reflected the ability of simultaneous scheme to deploy multiple MTD strategies at the same time, cutting off attackers from multiple aspects of the network. However, this approach is resource-intensive. In other metrics, such as ROA and R, our models performed 10% to 20% better. Additionally, the new models excelled in the APE metric, which measures the percentage of new vulnerabilities at each network node, providing evidence of their effectiveness. Interestingly, the general performance of these models seems to be more influenced by the architecture of the reinforcement learning algorithm and MTD type rather than the specific metrics used for optimization. As demonstrated in Figure 7 and 9, the difference between the best and worst models based on metric selection was relatively mild. Nevertheless, both proposed hybrid models managed to outperform traditional MTD schemes significantly. This comparison advances the state of the art by demonstrating that mixed models, which combine

diverse MTD strategies, offer superior security outcomes. These findings highlight the importance of adaptive models that incorporate MTD techniques, emphasizing that a flexible approach outperforms rigid, static MTD schemes in defending against evolving threats.

5 Future Works and Conclusion

5.1 The need for more complex adversaries

In the current simulator system there is only one type of adversary. Although certain previous gaps in researches have been addressed, this research did not incorporate for more advanced adversaries. This restrict our understanding of the limits of existing or future models and more complex adversaries need to be developed to test the performance of the models in different dynamics.

5.2 Limited evaluation metrics

There have not been evaluation in terms of the performance of network in this report. The evaluation metrics should be more diverse and holistic to reflect how well the models perform in both security while maintaining the network efficiency. There is also a need for more diverse evaluation metrics to evaluate different aspects of security in the network to better evaluate the models.

5.3 Limited types of MTD techniques

In this research, only four distinct types of MTD techniques were tested within the simulator. Utilizing more advanced or dynamic MTD techniques could enhance effectiveness in thwarting various types of attackers and defending against complex scenarios within the network.

5.4 More reactive reinforcement model

Although the proposed model have incorporated attack stage as a feature to make the model more aware of the attacks, it is far too simplistic and there needs to be more complex reactive components being incorporate into future models.

5.5 Lack of real life scenario testing

The current MTDSimTime simulator lacks the reliability needed to determine whether high-performing models can sustain their effectiveness in real-world scenarios. Testing in real-life conditions is essential, as failures in cybersecurity can lead to significant and detrimental consequences.

5.6 Conclusion

This report enhances the understanding of integrating AI and MTD by analyzing the effects of various metrics optimization on a proposed double Deep Q-Learning reinforcement learning model across different MTD intervals and network sizes. The findings demonstrate that incorporating multiple metrics into reward optimization generally leads to favorable outcomes. However, the inclusion of metrics that do not directly

represent the network's security posture can negatively impact overall performance. Notably, the hybrid model, which combines both static and time-based features, outperformed other individual metric approaches, though its effectiveness declined at shorter intervals. Additionally, ASR metric optimization consistently demonstrated strong performance across different network sizes during short intervals. Models that deploy diversity techniques significantly outperform models that deploy shuffling technique by up to 140% across all metrics optimization. These results offer valuable insights into how various metrics affects the performance of machine learning models under diverse network settings.

References

[1] Sandro Carvalho, João Carvalho, João Silva, Gilberto Santos, and Gonçalo Bandeira. Concerns about cybersecurity: The implications of the use of ict for citizens and companies. *Journal of Information Systems Engineering and Management*, 8:20713, 04 2023.

[2] Adel Alshamrani, Sowmya Myneni, Ankur Chowdhary, and Dijiang Huang. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys Tutorials*, PP:1–1, 01 2019.

[3] Jin-Hee Cho, Dilli P. Sharma, Hooman Alavizadeh, Seunghyun Yoon, Noam Ben-Asher, Terrence J. Moore, Dong Seong Kim, Hyuk Lim, and Frederica F. Nelson. Toward proactive, adaptive defense: A survey on moving target defense. *IEEE Communications Surveys and Tutorials*, 22(1):709–745, 2020.

[4] Irshaad Jada and Thembekile O. Mayayise. The impact of artificial intelligence on organisational cyber security: An outcome of a systematic literature review. *Data and Information Management*, 8(2):100063, 2024. Systematic Review and Meta-analysis in Information Management Research - Part II.

[5] Minjune Kim, Jin-Hee Cho, Hyuk Lim, Terrence Moore, Frederica Nelson, and Dan Kim. Performance and security evaluation of a moving target defense based on a software-defined networking environment. pages 119–129, 11 2022.

[6] Hooman Alavizadeh, Dong Seong Kim, Jin B. Hong, and Julian Jang-Jaccard. Effective security analysis for combinations of moving target defense techniques in cloud computing (short paper). In *Information Security Practice and Experience - 13th International Conference, ISPEC 2017, Proceedings*, Lecture Notes in Computer Science, pages 539–548, Germany, January 2017. Springer-Verlag London Ltd. 13th International Conference on Information Security Practice and Experience, ISPEC 2017; December 13-15, 2017.

[7] Hooman Alavizadeh, Jin Hong, Dong Seong Kim, and Julian Jang-Jaccard. Evaluating the effectiveness of shuffle and redundancy moving target defense techniques in the cloud. *Computers & Security*, 102, March 2021.

[8] Gayathri Rajakumaran and Neelanarayanan Venkataraman. Performance assessment of hybrid moving target defense for dos mitigation in public cloud. *International Journal of Intelligent Networks*, 2:140–147, 09 2021.

[9] Alex Brown, Tze-Wen Lee, and Jin Hong. Evaluating moving target defenses against realistic attack scenarios. pages 1–8, 05 2023.

[10] Sarah Alhozaimy and Daniel A. Menascé. A formal analysis of performance-security tradeoffs under frequent task reconfigurations. *Future Generation Computer Systems*, 127:252–262, 2022.

[11] Tuan Anh Nguyen, Minjune Kim, Jangse Lee, Dugki Min, Jae Woo Lee, and Dan Kim. Performability evaluation of switch-over moving target defense mechanisms in a software defined networking using stochastic reward nets. *Journal of Network and Computer Applications*, 199:103267, November 2021.

[12] Júlio Mendonça, Jin-Hee Cho, Terrence Moore, Frederica Nelson, Hyuk Lim, and Dan Kim. Performance impact analysis of services under a time-based moving target defense mechanism. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 20:154851292110369, 08 2021.

[13] Taha Eghtesad, Yevgeniy Vorobeychik, and Aron Laszka. *Adversarial Deep Reinforcement Learning Based Adaptive Moving Target Defense*, pages 58–79. 12 2020.

[14] Sailik Sengupta, Tathagata Chakraborti, and Subbarao Kambhampati. *MTDeep: Boosting the Security of Deep Neural Nets Against Adversarial Attacks with Moving Target Defense*, pages 479–491. 10 2019.

[15] Rongbo Sun, Yuefei Zhu, Jinlong Fei, and Xingyu Chen. A survey on moving target defense: Intelligently affordable, optimized and self-adaptive. *Applied Sciences*, 13:5367, 04 2023.

[16] Wenxiao Zhang. Evaluating multiple moving target defense in the time domain. 2023.

[17] A. Brown. Mtdsim, 2021.

[18] Ankur Chowdhary, Dijiang Huang, Abdulhakim Sabur, Neha Vadnere, Myong Kang, and Bruce Montrose. Sdn-based moving target defense using multi-agent reinforcement learning. In *Proceedings of the Conference Name*, 03 2021.

[19] Dilli P. Sharma, Simon Yusuf Enoch, Jin-Hee Cho, Terrence J. Moore, Frederica F. Nelson, Hyuk Lim, and Dong Seong Kim. Dynamic security metrics for software-defined network-based moving target defense. *Journal of Network and Computer Applications*, 170:102805, 2020.

[20] Eric M. Hutchins, Michael J. Cloppert, and Rohan M. Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. In *Leading Issues in Information Warfare & Security Research*, volume 1, pages 80–106. Academic Publishing Limited, Reading, UK, 2011.

[21] MITRE. Mitre ATT&CK framework: Adversary tactics and techniques, 2021.

[22] María del Carmen Lucas-Estañ, Baldomero Coll-Perales, and J. Gozalvez. Redundancy and diversity in wireless networks to support mobile industrial applications in industry 4.0. *IEEE Transactions on Industrial Informatics*, 17:311–320, 03 2020.

[23] H. Alavizadeh, S. Aref, D. Kim, and J. Jang-Jaccard. Evaluating the security and economic effects of moving target defense techniques on the cloud. *IEEE Transactions on Emerging Topics in Computing*, 10(04):1772–1788, Oct 2022.

[24] Qisheng Zhang, Jin-Hee Cho, Terrence Moore, Dan Kim, Hyuk Lim, and Frederica Nelson. *EVADE: Efficient Moving Target Defense for Autonomous Network Topology Shuffling Using Deep Reinforcement Learning*, pages 555–582. 05 2023.

[25] Tao Zhang, Changqiao Xu, Jiahao Shen, Xiaohui Kuang, and Luigi Grieco. How to disturb network reconnaissance: A moving target defense approach based on deep reinforcement learning. *IEEE Transactions on Information Forensics and Security*, PP:1–1, 01 2023.

[26] Sailik Sengupta and Subbarao Kambhampati. Multi-agent reinforcement learning in bayesian stackelberg markov games for adaptive moving target defense, 07 2020.

[27] Qian Yao, Yongjie Wang, Xinli Xiong, Peng Wang, and Yang Li. Adversarial decision-making for moving target defense: A multi-agent markov game and reinforcement learning approach. *Entropy*, 25(4):605, 2023.

[28] Tina Moghaddam, Minjune Kim, Jin-Hee Cho, Hyuk Lim, Terrence Moore, Frederica Nelson, and Dan Kim. A practical security evaluation of a moving target defence against multi-phase cyberattacks. pages 103–110, 06 2022.

[29] Jin Hong, Simon Enoch, Dan Kim, Armstrong Nhlabatsi, Noora Fetais, and Khaled Khan. Dynamic security metrics for measuring the effectiveness of moving target defense techniques. *Computers & Security*, 79, 08 2018.

[30] Hooman Alavizadeh, Julian Jang-Jaccard, and Dong Seong Kim. Evaluation for combination of shuffle and diversity on moving target defense strategy for cloud computing. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 573–578, 2018.

[31] Hooman Alavizadeh, Dong Seong Kim, and Julian Jang-Jaccard. Model-based evaluation of combinations of shuffle and diversity MTD techniques on the cloud. *Future Generation Computer Systems*, 111:507–522, 2020.

[32] Hooman Alavizadeh, Jin B. Hong, Julian Jang-Jaccard, and Dong Seong Kim. Comprehensive security assessment of combined MTD techniques for the cloud. In *Proceedings of the 5th ACM Workshop on Moving Target Defense*, MTD '18, pages 11–20, New York, NY, USA, 2018. Association for Computing Machinery.

[33] Simon Enoch, Mengmeng Ge, Jin Hong, Hani Alzaid, and Dan Kim. Evaluating the effectiveness of security metrics for dynamic networks. pages 277–284, 08 2017.

[34] Maxim Lapan. *Deep Reinforcement Learning Hands-On*. Packt Publishing, Birmingham, UK, 2018.

[35] Hado Van Hasselt. Double q-learning. pages 2613–2621, 01 2010.

[36] Naveen Venkat. The curse of dimensionality: Inside out, September 2018.

[37] Balázs Varga, Balázs Kulcsár, and Morteza Haghir Chehreghani. Deep q-learning: A robust control approach. *International Journal of Robust and Nonlinear Control*, 33(1):526–544, 2023.

[38] Muddasar Naeem, Syed Rizvi, and Antonio Coronato. A gentle introduction to reinforcement learning and its application in different fields. *IEEE Access*, 8:209320–209344, 01 2020.

6 Appendices

6.1 MTD Classification

Moving Target Defense can be categorized in different ways. Those categorization mainly revolve around three questions [3]: What to Move, When to Move and How to Move.

6.1.1 What to Move

What to Move is about the configurations in the network that can be moved by MTD techniques in order to change the attack surface. Majority of components that can be moved can be found in the Application Layer and OS-Host/VM- Instance Layer [3].

6.1.2 When to Move

When to Move refers to the decision-making process of determining the optimal time to change the network's state [3]. It can be classified into three categories: event-based, time-based, and hybrid.

- **Event-based:** Adaptation is triggered by an event or alert in response to detected suspicious activity. This approach aims to counter an attacker's actions under the assumption that those actions have been detected.
- **Time-based:** Adaptations are triggered at fixed or random intervals on a proactive schedule, regardless of detected threats. This ensures that any information gained by an attacker becomes obsolete over time.
- **Hybrid:** This approach combines both event-based and time-based strategies. Adaptations are triggered by events or security alerts but also occur within bounded time intervals to address potential undetected threats.

6.1.3 How to Move

How to Move defines the method for changing the configurations in the network to increase unpredictability and uncertainty [3], leading to greater confusion for attackers. This is related to the three operation-based MTD classification components: shuffling, diversity, and redundancy.

- **Shuffling**: Rearranges system attributes (e.g. IP addresses, ports) to make it harder for attackers to predict or target specific components and to dynamically change the attack surface.

- **Diversity**: Involves using varied implementations or configurations (e.g. different software versions or operating systems) to reduce vulnerabilities shared across systems.
- **Redundancy**: Introduces multiple equivalent resources (i.e replicated services) to ensure availability and resilience against attacks.

6.2 Reinforcement Learning

Reinforcement Learning (RL) is an unsupervised machine learning algorithm guided by a specific objective. The agent learns by interacting with an unknown environment, typically in a try-and-error way [38]. Its objective is to maximize a defined reward by taking actions that lead to favorable outcomes over time. Through this process, the agent can alter its strategy to achieve better long-term results.

6.2.1 Markov Decision Process

Reinforcement learning uses the Markov Decision Process(MDP). Table 1 documents the key properties involved in the Markov Decision Process.

Table 9: Markov Decision Process components

Symbol	Description
$\mathbb{S}$	Set of environment states.
$\mathbb{A}$	Set of actions available to the agent.
$\boldsymbol{E}_\pi$	Expectation operator with respect to the policy π.
γ	Discount factor.
τ	Time step relative to the current time t.
$r(t)$	Immediate reward obtained at time t.
$P(s' \mid s, a)$	Transition probability from state s to state s' under action a.
R	Accumulated reward
$\pi(a \mid s)$	Policy representing the agent's decision-making strategy, specifying the probability of taking action a in state s.
π^*	Maximum expected reward from all states
T	Total length of the experiment process

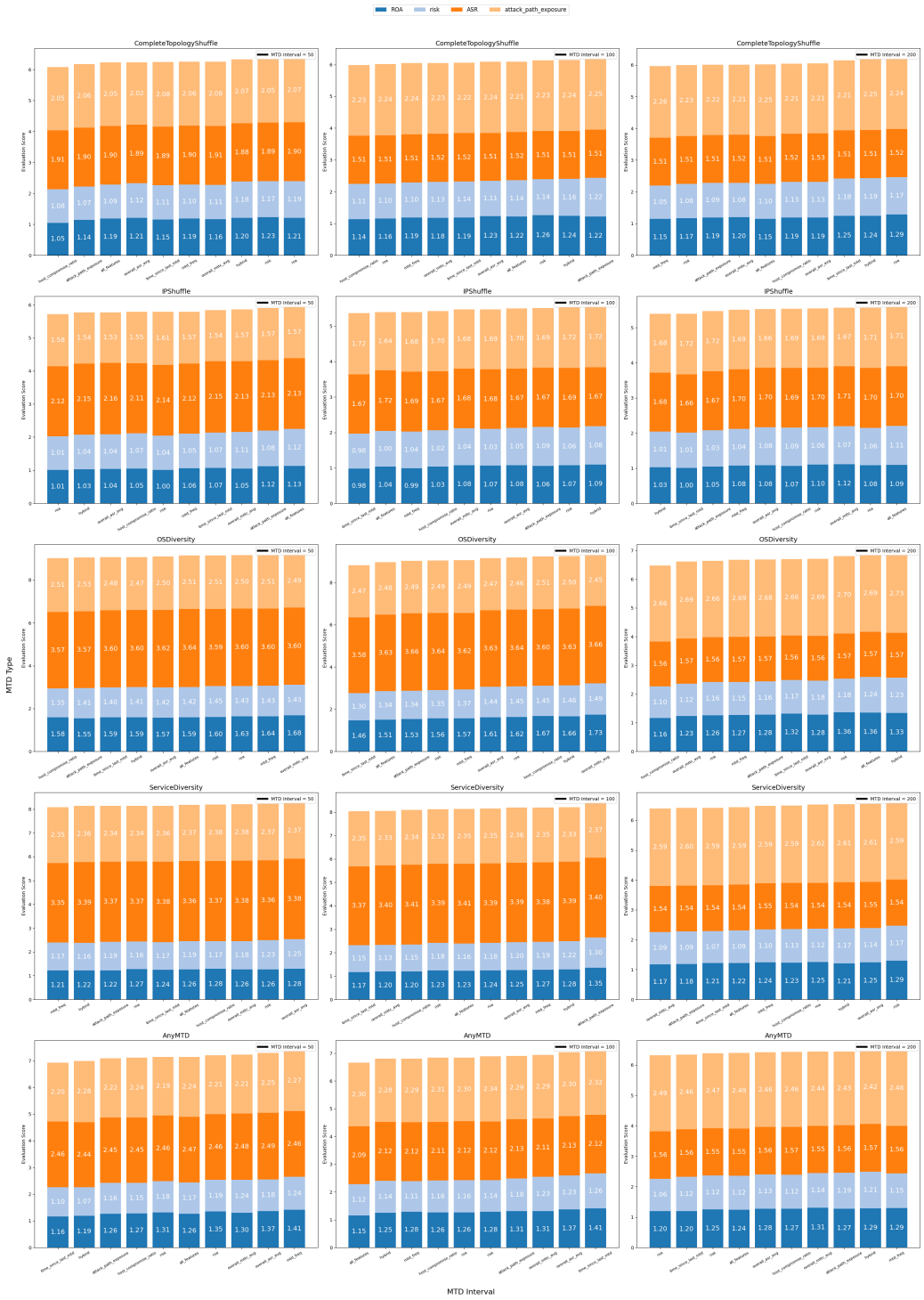

Figure 13: The performance of different metrics optimization across different MTD Intervals and MTD Types (actual scores)

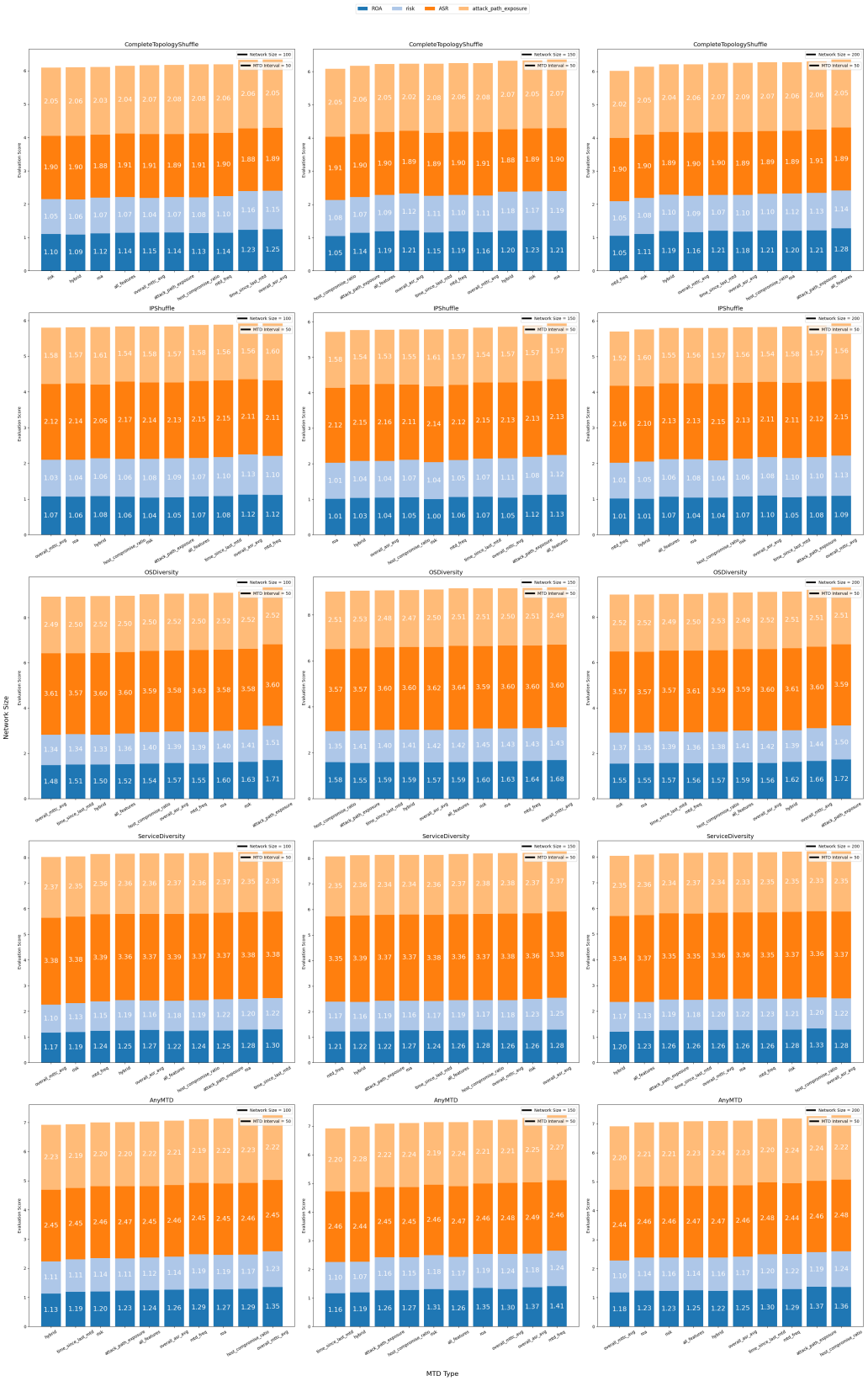

Figure 14: Performance of different metrics optimization across different Network Sizes and MTD Types (actual scores)

Figure 15: Performance improvement of metrics optimization across different MTD Intervals and MTD Types

Figure 16: Performance improvement of metrics optimization across different Network Sizes and MTD Types