University of Western Australia

MASTERS THESIS

Evaluating Multiple Moving Target Defence in the Time Domain

Wenxiao Zhang

School of Engineering

Supervised by

Dr. Jin HONG

Department of Computer Science and Software Engineering

Word Count: 9375

21 May 2023

Contents

List of Figures

List of Tables

Abstract

Moving Target Defense (MTD) has been proposed as a method to reduce the impact of cyberattacks and confuse attackers. Instead of adopting an approach that involves the complete elimination of all vulnerabilities in a system, MTD aims to increase the dynamicity of systems by continuously reconfiguring their components to alter the attack surface. While numerous studies have investigated MTD techniques, there remains a gap in comprehensively analysing the effectiveness of combined MTD strategies, particularly in relation to different security aspects like time, cost, and effort. To address this gap, our research focuses on evaluating and comparing the effectiveness of various MTD strategies in the time domain. We have developed a simulation framework called *MTDSimTime* to assess the effectiveness of MTD strategies through multiple simulations based on time.

Our evaluation primarily relies on Mean Time to Compromise (MTTC) as the key metric. Our findings indicate that the effectiveness of MTD strategies varies depending on the employed techniques, combinations, and execution schemes. Among the techniques examined, network shuffle-based MTD techniques exhibit the highest effectiveness, resulting in significantly higher MTTC scores compared to diversity-based techniques. In fact, network shuffle-based techniques consistently outperform diversity-based techniques by more than 130% on average. However, this dominance is not consistently observed when network shuffle-based techniques are combined with other MTD techniques. Conversely, our analysis demonstrates a notable trend wherein the combination of MTD techniques from distinct categories, such as shuffle + diversity, tends to yield more favourable outcomes. This combination achieves higher MTTC scores in a majority of scenarios, resulting in an average improvement of approximately 20% compared to using shuffle techniques alone. Furthermore, our study reveals that the effectiveness of MTD strategies is influenced by various factors within the simulated network environment, which play a crucial role in determining overall effectiveness.

1 Introduction

With the development of network scale, cybersecurity has become a growing issue worldwide. Various cyberattacks have emerged and caused a huge amount of damage to both individuals and society as a whole [1]. In order to mitigate these attacks and also to continuously change the system and network attack surface to confuse the attackers, Moving Target Defense (MTD) has been proposed [2]. Instead of eliminating all the vulnerabilities in the system, the core idea of it is to make the system become dynamic by continuously reconfiguring the system components in order to change the attack surface [4]. From this perspective, Introducing MTD to the system can help deal with the common issues of static nature that traditional security mechanisms have. For instance, some traditional Intrusion Detection Systems (IDS) and firewalls use static defence mechanisms that can have a higher risk of being exploited by Advanced Persistent Threat (APT), which can perform long-term vulnerability analysis and penetration testing on the target and is good at breaking static defences [3, 4]. Such attack strategies can be effectively thwarted by applying MTD as it can dynamically move the components, which can effectively change the existing vulnerabilities of the system. Thus limiting the attack cycle and making it difficult for attackers to analyse vulnerabilities and exploit the system. Furthermore, In comparison with conventional defence mechanisms, MTD provides affordable defence opportunities as in many cases it can be deployed by utilising the existing system components and technologies instead of using a huge amount of time and resources to create a new defence solution [4].

The motivation behind this project is to address the limitations of previous research in the field of MTD, particularly regarding the consideration of the time domain when modelling and evaluating specific MTD techniques deployed. While several approaches have been proposed for designing, developing, and evaluating MTD systems, very few have considered the temporal aspects of these systems. This is a critical gap, as the effectiveness of MTD techniques can depend on the time duration of events in the network system. For example, previous research has proposed analytic modelling approaches to evaluate the performance impact of MTD [18, 19, 20]. However, these models were limited by their assumption

that the time duration of all events in the network system is exponentially distributed within a range of values referring to empirical observations, and their lack of investigation of specific events that occur when deploying MTD techniques.

Accordingly, this project evaluated the effectiveness of MTD techniques in the time domain by modelling and analysing discrete events that occurred using simulations. This was done through *MTDSimTime*, a simulator developed by extending the previously implemented MTD simulator *MTDSim* by Brown and Lee [17]. Examining combined MTD techniques in the time domain has the potential to significantly enhance the cybersecurity of the IT industry. Given that the time taken for events in the network can have a profound impact on the evaluation of MTD performance, it is essential to explore how MTD techniques can secure systems in the time domain to ensure reliable simulation results. The outcomes of this research can provide valuable guidance to IT companies in selecting effective MTD strategies to safeguard their network systems. Specifically, the primary contributions of this work include:

- Provided a brief overview of MTD techniques and adversary tactics and techniques.
- Reviewed previous works on MTD techniques, discrete event simulation, and evaluation metrics and synthesised them into our research.
- Presented our proposed methods in detail, including network modelling, MTD techniques, and adversary profiles, in a time-based simulation environment.
- Conducted evaluations of different MTD combination strategies and execution schemes across various scenarios using our proposed simulation framework.

2 Background

2.1 MTD Classifications

Different MTD techniques can be categorised into different classifications [5]. The key principle of classifying MTD techniques revolve around these three questions [4]: what to move, when to move, and how to move.

2.1.1 What to Move

Multiple components of system that can be moved or reconfigured to change the attack surface. Cho et al. [4] summarised dozens of elements in the system that can be moved based on different MTD techniques. As it is shown in Table 1, a large number of these components exist in the Application Layer and OS-Host/VM-Instance Layer.

2.1.2 When to Move

Timeliness-based MTD classification categorised MTD techniques into three different types to determine *when to move* [4]. Specifically, three categories are detailed as follows:

- **Time-based MTD**: This strategy triggers MTD operations periodically. All MTD operations are deployed based on a certain time interval, which is defined as *MTD interval*. *MTD interval* can be either a fixed time interval or an interval with a certain range of variation [6].
- **Event-based MTD**: This approach triggers MTD operations only when a certain event happens in the network. Events such as security alerts raised by Intrusion Detection System (IDS) can be a trigger to deploy MTD operation [26].
- **Hybrid**: Some MTD approaches take both periodically and event-driven strategies. These approaches deploy MTD operations adaptively based on current situations in the network.

Apart from these three categories, deploying MTD randomly without any trig-

Table 1: Moving elements of MTD techniques at different layers [4]

Layers	MTD Techniques		
	Shuffling	Diversity	Redundancy
Application	TCP/UDP ports	Web: Appache, IIS, GWS etc.	Web service replica
		App: .Net Framework, Java, PHP etc.	Application replica
		Database: SQL server, MySQL, Oracle etc.	Database backup and replica
		Others: Mail-server, Proxy-server etc.	Other service replica
OS-Host	IP address	Windows: Windows server 2003/2008, Windows 9.x, 8, 10 etc.	Host OS and VM replica
		Linux: Redhat, Debian, Caldera etc.	
		Solaris	
		Others: Unix, HP-UX etc.	
VM-Instance	Virtual IP address	Same as OS	
Virtual Machine Manager	Failover, Switchover	Xen	Hypervisor's replica
		Vmware	
		ESXi	
		Others: Kernel-based VM (KVM), Virtual-box (Vbox), IBM vSphere	
Hardware	Hardware replacement	Intel	Hardware backups and replica
		HP	
		Sun Solaris	
		Others: ARM, Atmega	

gering mechanism could be another possible approach. However, it is difficult to control and assess the effectiveness of a random defense mechanism as it is highly unstable.

2.1.3 How to Move

Operation-based MTD classifies MTD techniques into three different types to determine 'how to move' [4]. Specifically, each type of the MTD technique is detailed as follows:

- **Shuffling**: Operations aim to increase confusion and uncertainty for attackers by rearranging or randomizing system configurations.
- **Diversity**: Operations aim to improve system resilience by deploying system components with different implementations that can serve the same functionalities.
- **Redundancy**: Operations aim to increase system availability and reliability by providing multiple replicas of system components.

2.2 Adversary Tactics and Techniques

2.2.1 Cyber Kill Chain

Cyber Kill Chain (CKC) is a series of steps that the adversaries must complete in order to achieve their objective [14], which is always breaching the Confidentiality, Integrity, and Availability (CIA) of a system. The CKC model has been widely adopted in the cybersecurity industry to model the attack profile and evaluate the security level based on it. The steps of CKC are as follows [4]:

- **Reconnaissance**: The practice of discovering and collecting information about a system. The target of the attacker will be identified and selected during this step.
- **Weaponisation**: The practice of embedding malware into the deliverable payload. The malware contains the virus or/and the backdoor that can help exploit the target.
- **Delivery**: After the payload is created, the intruders can then deliver the above payload to the target.
- **Exploitation**: After the payload is delivered, the hacker can start exploiting the target by using the malware embedded in the payload.
- **Installation**: The practice of installing malware on the target.
- **Command and Control**: After the malware is installed, the intruders can take control of the target host by establishing a C2 channel.
- **Action on Objectives**: With the above six steps finished, the intruders can achieve their goal by breaching the CIA.

2.2.2 MITRE ATT&CK

MITRE ATT&CK is a new standard framework of adversary tactics and techniques based on real-world observations [15]. Unlike CKC, the tactics of MITRE ATT&CK are unordered and may not all take place in a single intrusion as the tactical objectives of the adversary can alter within an attack operation. The MITRE ATT&CK framework now has three different iterations [16]:

- **Enterprise**: The components that are present in traditional Information Technology (IT) threats and scenarios. It can also be divided based on cloud computing (XaaS) and operating systems (Windows, Linux, etc.).
- **Mobile**: The unique adversarial behaviour found when attacking the operating systems (iOS, Android, etc) of the mobile device.
- **Industrial Control Systems (ICS)**: The features found in Operational Technology (OT) attacks and scenarios. The convergent nature of IT and OT will cause the features to overlap so that it is separate from Enterprise's ATT&CK framework.

MITRE ATT&CK aids in modelling attackers in simulations by selecting relevant techniques from the framework, mapping them to the simulated environment, choosing adversary profiles, defining their behaviour, implementing ATT&CK techniques, and evaluating security measures. By leveraging ATT&CK, simulations replicate real-world attack scenarios, identify vulnerabilities, and improve detection and response capabilities.

3 Literature Review

In this section, we conduct a comprehensive review and analysis of prior works relevant to our study, with the goal of assessing their accomplishments, identifying limitations, and evaluating the suitability of their methodologies. To achieve this objective, we review various combination strategies of multiple MTD techniques and corresponding evaluation methods in Section 3.1. We also investigate existing models for simulating discrete events of MTD in Section 3.2. Additionally, we cover different types of evaluation methods, including prior work utilising the *MTDSim*, in Section 3.3. Finally, in Section 3.4, we explore metrics specifically designed for evaluating MTD in the time domain. By conducting a thorough examination of the literature, we aim to gain valuable insights and establish a strong foundation for our research.

3.1 Combined MTD Techniques

As described in Section 2.1.3, MTD techniques can be categorised into shuffling, diversity, and redundancy. Each technique employs a distinct approach to safeguard the system from a distinct perspective. For example, shuffle-based MTD techniques are often developed to enhance system security, while redundancy techniques generally strive to improve reliability and availability [8]. Studying how multiple MTD techniques can be combined and evaluated can aid in improving the MTD scheduling strategy, ultimately enhancing the system's overall security and performance.

Alavizadeh et al. [12] combined Shuffling (S), Diversity (D), and Redundancy (R). Similar to their previous works on combining S+R [8], and S+D [10], the researchers carry out their investigations of the effectiveness of the combined MTD techniques based on HARM for the cloud environment. And in this time, they used four security metrics, which are R, RoA, AC, and system availability (SA). They found that all security metrics are improved after deploying S+D+R in comparison with deploying a single MTD technique. However, one of the limitations of their work is the lack of consideration of the system performance. As deploying multiple MTD techniques can lead to more resource consumption, the overall performance

of the system could be affected, which may also have a negative impact in terms of Quality of Service (QoS).

While previous work has proposed a variety of approaches to evaluate the performance of different combinations of multiple MTD technologies, few have treated the time domain as a major consideration. For instance, literature [9] and [10] applied some time-based metrics on the attacker's side to evaluate their proposed MTD model on the cloud-based environment. However, the evaluation methods in the time domain are insufficient as they did not fully consider the issues caused by the time duration of specific events that occurred in the network. Time-related issues such as resource occupation conflicts that occurred during system reconfiguration can affect the overall MTD evaluation [20]. Thus, this work focused on tackling such issues by further integrating the time domain in the simulation. It introduced a further evaluation of the performance of the combination of multiple MTD techniques and an analysis of the pattern of time duration between the attack and the specific MTD operations in order to get more constructive outcomes.

3.2 MTD Discrete Event Simulation

Multiple events will occur in the network when the MTD approach is introduced. modelling discrete events is an essential process for evaluating the effectiveness of combining multiple MTD approaches in the time domain. For example, as two different MTD techniques can move the same resources [4], conflicts will happen if such two MTD techniques are deployed at the same time, or if one MTD operation is deployed while another MTD operation is unfinished. In addition, other events that occur in the network such as attacks, and job requests, can also affect the result of the simulation. To fully investigate the impact of different events occurring in the network and to find a solution to deal with conflicts, modelling discrete event simulation is needed [13].

Petri Nets, introduced by Carl Adam Petri in 1962, are widely used for simulating discrete event systems and analysing their behaviour and structure [25]. In a study by Cai et al. [26], Generalized Stochastic Petri Nets (GSPN) were employed to evaluate three shuffle-based Moving Target Defense (MTD) techniques. The model

classified MTD event state transitions into two categories: immediate transitions and timed transitions. Immediate transitions can occur randomly without time delay, while timed transitions involve a random delay between enabling and firing. The researchers applied their GSPN model to a web server system and found it suitable for the three MTD techniques and their combination. Although the study is more comprehensive than many existing approaches, it has certain limitations. Since the proposed MTD techniques in the model were only deployed on a web server system, some of the specific approaches may not be applicable for modelling generic networks.

Markov Chain is an effective approach for modelling state transitions in a system, representing events as stochastic processes with discrete states. The future state depends on the present state and time, making these parameters crucial for analysing state transitions [25]. By applying Markov chains, one can capture Moving Target Defense (MTD) behaviors in the time domain, enabling performance analysis based on the data of state transitions for each MTD event. This analysis facilitates the development of adaptive deployment strategies to enhance overall MTD performance. Chen et al. [18] proposed two Markov process-based models for MTD: time-based and event-based models. These models evaluate the effectiveness of MTD by investigating its impact on system performance while ensuring system protection. The time-based MTD is deployed periodically, while the event-based MTD is deployed during system idle periods, without any ongoing job processing on the server. Three metrics were used to evaluate MTD: mean number of jobs waiting in the system, mean job sojourn time, and ASP. Analysis of the Markov process-based models revealed that event-based MTD has a less detrimental effect on server performance compared to time-based MTD. However, the designed job request stream in their study was accompanied by the attack stream, which does not align with real-world cyberattacks. Consequently, the ASP in the event-based MTD scenario could not be calculated due to the omission of modelling the attack profile. Additionally, Connell et al. [19] proposed a quantitative analytic model for evaluating resource availability and MTD performance. They employed Continuous Time Markov Chains (CTMC) to compute the probability distribution of the reconfigured resource count, allowing determination of resource availabil-

ity and other performance metrics such as response time. The authors validated their approach through a simulation implemented using SimPy [21]. Simulation results demonstrated the trade-off between security and performance during MTD deployment.

Although Petri Net and Markov Chain are effective techniques for modelling and analysing state transitions in discrete events, previous approaches have been primarily focused on analytical modelling, abstracting the MTD techniques and attackers. Thus, the discrete event implementation in this research was not based on specific modelling strategies in the simulation. However, this study utilised the details of the design of state transitions for specific discrete events presented in the literature, contributing to a better understanding of the application of Petri Net and Markov Chain in MTD simulations. Further research may investigate the optimization of modelling strategies to enhance the accuracy of MTD simulations.

3.3 MTD Simulations

Cho et al. [4] conducted a survey on evaluation methods used to assess the performance of MTD techniques. The survey identified four types of methods: analytical models, emulation models, real test-bed environments, and simulation models. While analytical models provide insights at a low cost, they lack the ability to capture real-world deviations. Emulation models and real test-bed environments offer higher validity but struggle with large-scale networks. Simulation models, although less valid, provide flexibility in modelling attacks, system components, network scales, and MTD techniques without significant restrictions. Despite inherent uncertainties, simulation models are selected as the preferred modelling technique in this work due to their scalability and comprehensive nature, allowing for the evaluation of MTD techniques in various scenarios.

Xiong et al. [23] introduced a simulation environment aimed at evaluating the effectiveness and performance of MTD techniques. The simulation environment utilises SimPy to model attackers, defenders, services, and users in various scenarios. Through the simulation, different MTD strategies can be assessed using metrics such as average attacking time and request processing. The results in-

dicate that the simulation environment efficiently evaluates MTD and assists in the selection of appropriate strategies. However, a limitation of the simulation environment is its lack of explicit incorporation of dynamic network conditions or changes over time. By not considering network dynamics, the realism of the simulation may be compromised, and the applicability of the results to real-world dynamic environments could be limited.

Torquato et al. [22] developed a tool called *PyMTDEvaluator* to assess the effectiveness of time-based MTD techniques against availability attacks such as Denial of Service (DoS) and Resource Starvation (RS) attacks. *PyMTDEvaluator* was designed based on the Stochastic Petri Net (SPN) model, which draws its assumptions from empirical observations. The SPN model proposed by Torquato et al. represents main events such as MTD actions, attack progress, and system downtime. The tool utilises three metrics, namely Security Assessment (SA), Attack Surface Probability (ASP), and System Capacity (SC), to evaluate MTD techniques. Consequently, this tool provides an inspiring idea for designing time-based MTD state transitions and modelling discrete events occurring in the network. However, the MTD movement they designed was generic and did not consider different types of MTD techniques.

Brown et al. [17] assessed the effectiveness of various combinations of multiple MTD techniques. They utilised *MTDSim*, a Python-based MTD simulator. However, due to time constraints, only shuffle-based MTD techniques were implemented in his work. To address this limitation, Lee implemented new attack profiles and diversity-based MTD techniques with additional security metrics to improve the overall functionalities and performance of the simulator [17]. However, a significant drawback of the simulator was its limited simulation of time. It solely modelled time in the context of the MTD interval (Section 2.1.2) while disregarding the execution time of MTD operations. This omission can pose problems, as the deployment and execution of MTD operations inherently require a certain amount of time [18, 19]. On the other hand, the simulator comprehensively designed the attacker's movements, with adaptively changing actions based on simulation time and specific events. Regrettably, due to the lack of considering time delays in MTD events and the absence of a realistic attack profile against

security strategies, the overall experimental results of MTD fell short of expectations. Consequently, it is crucial to integrate the time domain into the simulator to enhance its performance and yield more meaningful outcomes. In this regard, the developed simulator serves as the foundation for , implemented in this work, which incorporates time-related aspects.

3.4 MTD Evaluation Metrics

To effectively evaluate the effectiveness of MTD techniques, it is crucial to carefully consider the choice of evaluation metrics. This evaluation involves two types of metrics: static metrics, which provide a holistic measure of security resilience, and dynamic metrics, which assess the adaptability and effectiveness of MTD techniques in the context of evolving threats. By selecting the most appropriate evaluation metrics, researchers can accurately assess the effectiveness of MTD techniques and their ability to enhance network security.

One widely adopted static metric is Attack cost (AC), which measures the cost incurred by attackers when exploiting vulnerabilities on a host machine. AC, in combination with Return on Attack (RoA) and system risk, has been utilized by Alavizadeh et al. [11, 10, 12] to evaluate their HARM-based MTD approach from both attackers' and defenders' viewpoints. Similarly, AC, RoA, and system risk have been employed in the *MTDSim* to assess the effectiveness of various combinations of shuffle-based MTD techniques [17]. However, these metrics had limitations in the previous simulator due to insufficient consideration of the time domain.

Another static metric widely used is Mean Time to Compromise (MTTC), which measures the time it takes for an attacker to compromise a target host on the network [4]. From the defender's perspective, MTTC can also be interpreted as Mean Time to Failure (MTTF). Employing MTTC or MTTF as evaluation metrics allows researchers to compare the effectiveness of different MTD techniques and choose the most suitable ones for specific scenarios. In this study, MTTC is selected as the primary evaluation metric in the simulator due to its focus on modelling the time domain and providing valuable insights into the performance of MTD

techniques.

In addition to static metrics, dynamic metrics have been introduced to address network changes. Hong et al. [24] introduced dynamic security metrics based on the Temporal Hierarchical Attack Representation Model (T-HARM), which captures security changes in the network and evaluates the effectiveness of MTD techniques. Building on this work, Lee [17] applied dynamic metrics such as Attack Path Variation (APV) and Attack Path Exposure (APE) to measure the effectiveness of MTD techniques by assessing the shift in security posture within the *MTDSim*. The authors also proposed a dynamic metric called Attack Compromise Duration (ACD), which calculates the MTTC of an attack on the target host in a given network state. ACD considers changes in network states (S) and the time required to exploit each vulnerability in the attack path ($t\,(ap_i)$), influenced by MTD operations. These dynamic metrics provide valuable insights into the adaptability and effectiveness of MTD techniques in dynamic network environments.

4 Methodology

This section presents the approach taken to design and evaluate the implemented time-based simulation framework *MTDSimTime*. We introduce the general structure of the simulation framework in Section 4.1, and discuss the network model used to simulate the network environment in Section 4.2. We also describe the MTD techniques implemented in the simulator in Section 4.3, and present the adversary profile in Section 4.4. Finally, we detail the methodology adopted to model simulation time in Section 4.5.

4.1 Overview of the Simulation Framework

The proposed simulation framework comprises three key modules: Simulated Network, MTD Techniques, and Adversary, as illustrated in Figure 1. The Simulated Network is generated as a graph of nodes and edges, which represent hosts and their connections, respectively. Each node runs a list of exploitable services that can be targeted by the Adversary module. The MTD techniques can conduct the MTD Operation, which performs a set of actions to interact with the Simulated Network and prevent attack actions in various ways. On the other hand, the Adversary module can launch the Attack Operation, which executes a list of attack actions on the hosts in the Simulated Network. To get a better understanding of the workings of each of these modules, more details can be found in the following sections.

4.2 Modelling the Network

In this section, we will provide a detailed account of how the Simulated Network is generated within the simulator. The process involves several key steps, including the generation of the network graph, assignment of host IP addresses, creation of operating systems, services, vulnerabilities, and establishment of user access credentials. Each of these steps is crucial for constructing a generic network environment that closely mimics the behaviour of a real-world network. By providing a faithful representation of network conditions, our simulator enables accurate and effective evaluation of MTD techniques.

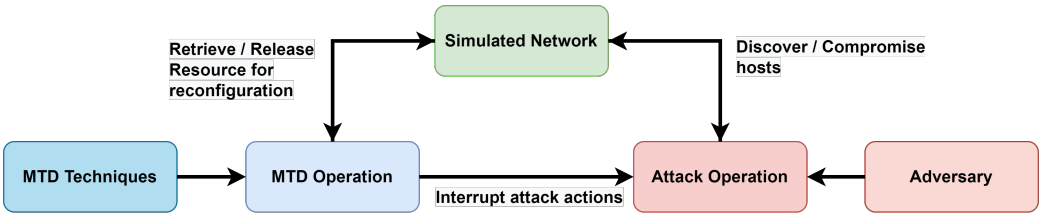


Figure 1: The Structure of *MTDSimTime*

4.2.1 Network Generation

To conduct a thorough evaluation of MTD techniques, it is essential to construct a well-designed simulated network. The primary objective of MTD techniques is to hinder adversaries from exploiting vulnerabilities in the services running on each host within the network. In this context, the network model should incorporate critical attributes, including a graph that represents the interconnections between hosts, where each host operates different services, each with its own vulnerabilities. To effectively represent these attributes in a structured manner, we adopt the widely utilized 3-layer HARM model [7]. This model encompasses three distinct layers that collectively capture the aforementioned attributes and facilitate a comprehensive evaluation of MTD techniques within the simulated network environment.

The 3-layer HARM model consists of three main components: the Host Attack Graph (AG_n), the Services on Host Attack Graph (AG_h), and the Service Attack Tree (AT_s), as shown in Figure 2. Firstly, the AG_n depicts the network topology structure of the model. It is generated using the Barabási-Albert graph model [40], with nodes representing hosts and edges representing their interconnections. Secondly, the AG_h illustrates the service connection status at the host level. The graph is generated using the Watts-Strogatz graph model [40] to reflect the common pattern of service connections on each host. Lastly, the AT_s provides a visual representation of the specific prerequisites an adversary must fulfil to compromise a given service.

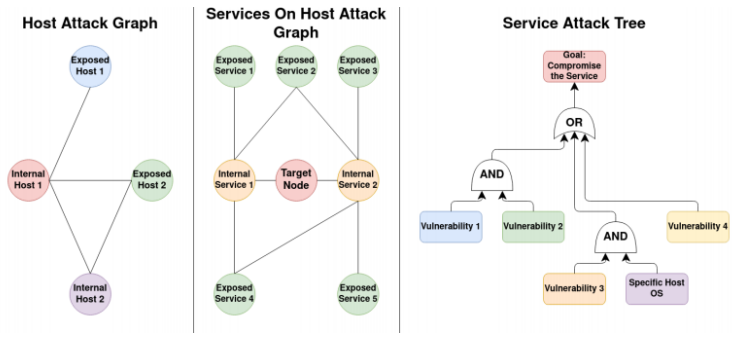


Figure 2: 3-Layer HARM Model [17]

4.2.2 Host Generation

When generating hosts in the simulation, we consider the basic attributes that can be realistically simulated. Therefore, during the network setup phase, each host is assigned a unique IP address, an operating system, a set of services, and a set of user access credentials. The IP address allows adversaries to locate and pivot to the host, and the choice of operating system affects the compatibility of services and potential vulnerabilities. Four types of operating systems (Ubuntu, CentOS, Windows, and FreeBSD) are available, each with its own set of services in the simulation. Additionally, each host is randomly assigned five users with their associated access credentials. However, some users may reuse their credentials across the network, increasing the risk of credential-stuffing attacks on other hosts.

4.2.3 Service Generation

As discussed in Section 4.2.2, services in the simulation are generated with varying compatibility for different operating systems. Therefore, the simulator generates services randomly for each of the four possible operating systems in the network, giving each service a unique name and version number between 1 and 99. Services with the same name may share vulnerabilities, with older versions being more susceptible to known vulnerabilities that attackers can exploit. To replicate the possibility of vulnerabilities being triggered on specific operating systems, half of the newly created services are designed with cross-platform compatibility. This enables them to run on any operating system within the network.

4.3 Modelling the MTD

This section aims to provide a comprehensive description of implementing MTD techniques in the proposed simulation environment. It starts by introducing the purpose and functionality of each MTD technique used, providing a detailed description of their operational flow and the discrete event modelling employed in the simulation. Moreover, the section outlines different execution schemes that facilitate the optimal execution of MTD techniques. Furthermore, it elaborates on the integration of MTD techniques into the time-based simulation, providing valuable insights into the application.

4.3.1 Modelling MTD Techniques

In the previous work [17], seven MTD techniques were implemented. For this study, we selected four techniques, including two shuffle-based methods (*CompleteTopologyShuffle* and *IPShuffle*) and two diversity-based methods (*ServiceDiversity* and *OSDiversity*), and made several adjustments to facilitate the simulation environment using *MTDSimTime*. Additionally, we introduce a newly implemented MTD technique in Section 4.3.1.5.

4.3.1.1 Complete Topology Shuffle *CompleteTopologyShuffle* is a generalised implementation of topology shuffling techniques that have been proposed in the previous research on Software Defined Network (SDN) [32, 33, 34]. The idea of it is to confuse the adversary by continuously changing the form of the network topology. The deployment of this MTD technique can entirely reconfigure the network's topology in the simulation, resulting in a change in the connection status of each host involved in the operation with other hosts.

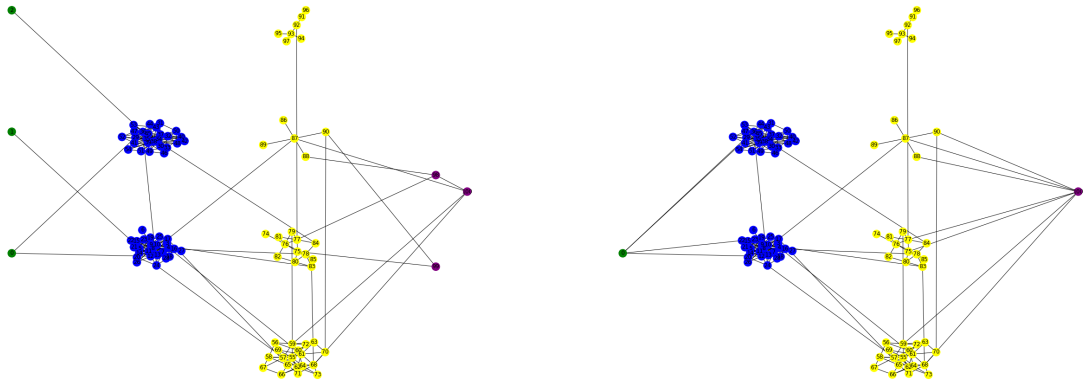
4.3.1.2 IP Shuffle *IPShuffle* is a commonly used MTD technique that has been extensively documented with various implementation approaches in previous literature [12, 30, 31]. The goal of this MTD technique is to impede the progress of the adversary when they try to find a target host or pivot their way into the network. It is implemented by randomly changing the virtual IP addresses of each involved host in the simulated network.

4.3.1.3 Service Diversity

ServiceDiversity re-configures services running on the host to thwart the attack process of exploiting the vulnerabilities on a certain service [36, 37]. As mentioned in Section 4.2.3, each service has 99 versions available for re-configurations and each version has a different number of vulnerabilities. Therefore, the implementation of this MTD technique randomly re-configures the services with different versions running on each involved host.

4.3.1.4 OS Diversity

OSDiversity diversifies the types of operating systems running on each host in the network. The purpose is to enhance the network's resilience against adversaries that exploit vulnerabilities on the OS level [38, 39]. Similarly, the deployment of this MTD technique re-assigns the operating system for each involved host, the new operating system is randomly selected from four different types as described in Section 4.2.2.



(a) Network Graph (b) Network Graph for Solving DAP

Figure 3: Comparison between the original network graph and the network graph for solving DAP

4.3.1.5 Optimising OS Diversity with DAP

Randomly assigning operating systems to devices on a network may result in suboptimal performance if the assignment turns out to be poor. To overcome this issue, the newly implemented MTD technique seeks to improve *OSDiversity* by solving the Diversity Assignment Problem (DAP) [38]. Unlike the previous random approach, this MTD technique is specifically designed to optimise the assignment of operating systems to hosts,

ensuring better overall network performance. The key idea is to maximise the connectivity of the network through the optimal assignment of each OS variant in the system. As illustrated in Figure 3, in order to simplify the analysis, a single endpoint host and a single database host are used to represent all endpoint hosts and database hosts, with the intermediate hosts treated as routing nodes to resolve client-to-client connectivity issues. This abstraction allows us to calculate connectivity in the context of DAP. The optimal assignment is retrieved by calculating the DAP objective function with a list of constraint functions demonstrated in the literature.

4.3.2 Modelling MTD Execution Scheme

In order to get the most out of the MTD simulation and evaluation in the time domain, the design of MTD execution schemes needs to be thoroughly considered. As different ways to execute MTDs may have different outcomes, it is worth exploring the most effective execution scheme that can maximise the effectiveness of MTD. Since previous works [17] did not focus on modelling how the implemented MTD techniques be triggered and executed, three specific MTD execution schemes are introduced in this work: random execution, alternative execution, and simultaneous execution.

MTDSimTime utilises time-based proactive MTD execution schemes with various execution strategies. Since these MTD techniques are triggered periodically and do not have reactive concerns, they are all implemented as time-based proactive schemes. Figure 4 illustrates the general logic flow of all MTD execution schemes. When a periodical triggering signal is received, the system will attempt to register a new MTD instance into a queue or trigger an existing MTD instance from a queue. Each registered MTD instance is then placed into the MTD queue, and the MTD instance with the highest priority is popped out of the queue and subsequently triggered and deployed in the system, provided there are no suspension issues (section 4.3.3). If there are suspension issues, the newly registered MTD instance is placed into the suspension queue, where MTD instances have higher priority to be popped and executed than the MTD queue. The primary distinction among each execution scheme resides in the register and trigger actions. To provide a visual

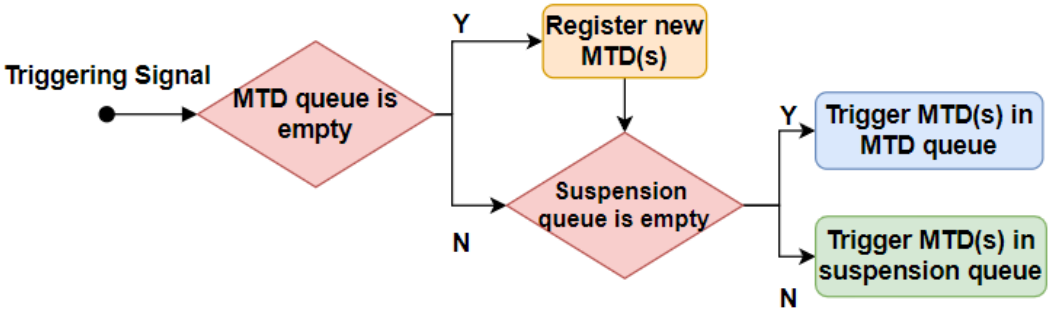


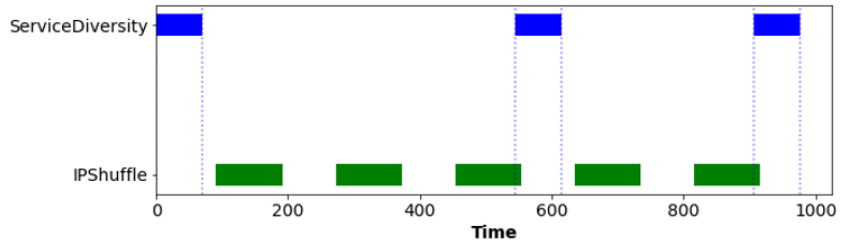
Figure 4: The logic flow of the MTD execution scheme

representation of these differences, Figure 5 presents an illustrative visualisation of these three execution schemes.

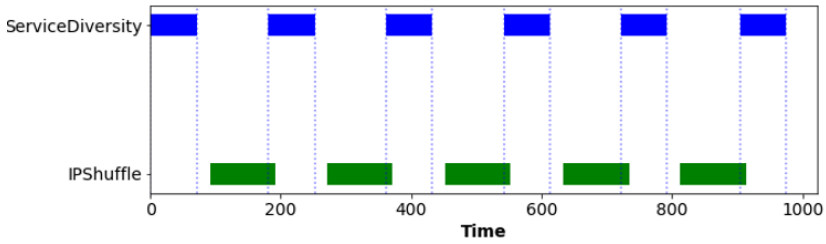
4.3.2.1 Random Execution The random execution scheme was used in previous work [17]. As shown in Figure 5a, the system registers an MTD that is randomly selected from all MTD techniques when the triggering signal is received. Then, a single MTD instance is triggered.

4.3.2.2 Alternative Execution The alternative execution scheme also registers and triggers one MTD technique at a time, but the MTD technique is not randomly selected at each triggering signal. Instead, it is selected alternatively to explore whether executing MTD techniques with a rational decision can affect effectiveness. Due to limited time and many factors to consider in the simulation, the alternative execution scheme implemented in *MTDSimTime* registers MTD based on the previously registered MTD instance only. For instance, as shown in Figure 5b, if *IPShuffle* is set to be registered after *ServiceDiversity*, then *ServiceDiversity* will be registered into the system the next time after textitIPShuffle is registered.

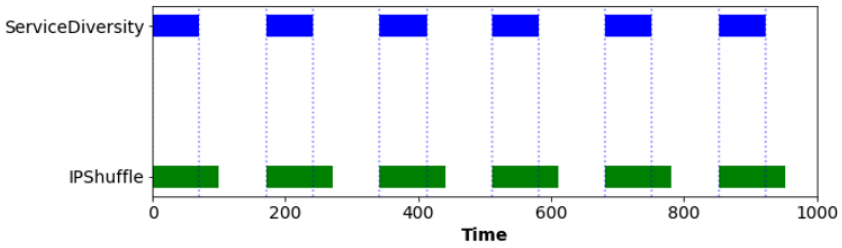
4.3.2.3 Simultaneous Execution The simultaneous execution scheme triggers all MTD techniques at each signal, as shown in Figure 5c. As all MTD techniques are triggered at the same time, this scheme can increase resource consumption compared to other schemes with the same MTD interval. Additionally, the highest priority MTD is deployed first among those occupying the same re-



(a) Random Execution Scheme



(b) Alternative Execution Scheme



(c) Simultaneous Execution Scheme

Figure 5: Visualisations of MTD Execution Schemes

sources due to the resource occupation issue (section 4.3.3). Lower-priority MTD techniques are put into the suspension queue and executed in the next cycle.

4.3.3 Modelling MTD Event

Figure 6 is the action flow diagram of the MTD operation. As described in section 4.3.2, MTD is registered and triggered periodically with a certain time interval. The time interval between the triggering of two MTD operations is denoted as *MTD interval*. After an MTD is triggered, it needs to get the resource it serves. As multiple MTD techniques may serve the same resource in the system, the first thing for the triggered MTD to do is to identify whether the resource they serve is available or not. If the resource is available, the triggered MTD will be deployed

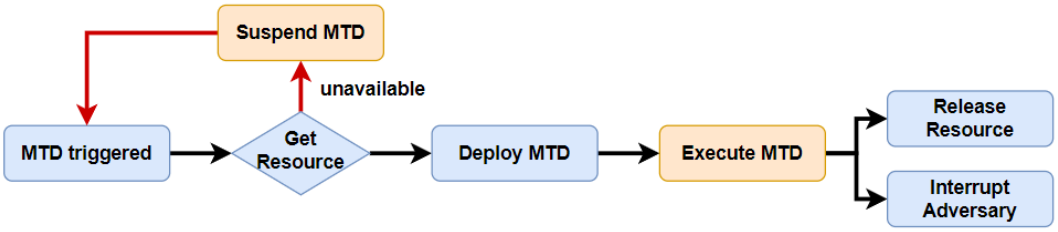


Figure 6: The Action Flow of the MTD Operation

and occupy that resource until its execution completes. If the resource is not available, the triggered MTD will be suspended until the resource is available.

For instance, there are three MTD techniques: *ServiceDiversity*, *CompleteTopologyShuffle*, and *IPShuffle*. The first two are designed to be deployed on the application layer while *IPShuffle* is deployed on the network layer. If *IPShuffle* is triggered first and no other MTD techniques occupy the network layer, then *IPShuffle* will be deployed and occupy the network layer. However, if a *CompleteTopologyShuffle* is triggered before the *IPShuffle* finished its deployment, the *IPShuffle* will be suspended since the network layer is still occupied by the *CompleteTopologyShuffle* and vice versa. However, *ServiceDiversity* can be deployed at any time in this case as it is deployed on the application layer.

4.3.4 Simulating Time for MTD

The previous simulator used a uniform distribution to model the *MTD Interval*. However, an exponential distribution may better reflect real-world scenarios and improve simulation accuracy. Therefore, this work proposes using an exponential distribution to simulate the *MTD Interval* in order to enhance the accuracy of the simulation.

When modelling the combination of multiple MTD techniques in the time domain, it is important to consider the varying time required to execute different MTD techniques. However, there is currently a lack of research measuring the time spent by specific MTD techniques. For example, Chen et al. [18] used a default value for the time duration of their generic MTD technique in their testbed environment. To address this, the proposed solution selects various time values for each MTD

technique within a reasonable range based on existing empirical data and conducts sensitivity analysis to determine the appropriate value to perform the evaluation.

4.4 Modelling the Adversary

This section provides a comprehensive overview of how the adversary is modelled and how their objectives are defined, in addition to detailing the techniques they use to execute their attacks. Furthermore, the section goes into detail about how the attack event is modelled and how the adversary's simulation of time is conducted, providing an in-depth explanation of how their behaviour is simulated in the proposed simulation framework.

4.4.1 Modelling Adversary Profiles

To accurately model the adversary, it is important to identify a set of key features. In this study, we have incorporated four primary features into the adversary profile to construct a comprehensive adversary model. These features are listed below.

4.4.1.1 Attack Objectives In previous works, there were two attack scenarios for the adversary model that have different objectives. The first attack scenario is to compromise as many hosts as possible giving a certain time period in the simulated network. In the second attack scenario, however, the adversary aims to compromise a specific "target" host located in the target layer of the network [17]. Due to the limited amount of time in developing the simulator on a new simulation framework, only the first attack scenario is selected to be refactored and simulated in *MTDSimTime*.

4.4.1.2 Exploiting Vulnerabilities The adversary model in the simulation always follows a sequence of attack actions to identify and compromise hosts within the simulated network. The adversary exploits vulnerabilities in the services running on each host to gain access to sensitive information. When enough services on a host have been compromised, it is assumed that the adversary has obtained enough information to consider the host compromised by them.

4.4.1.3 Command and Control Capabilities

The adversary has command and control capabilities on compromised hosts, which allows them to pivot themselves on each of the compromised hosts and form an attack path of compromised hosts to target new hosts in the network. It is assumed that the compromised hosts will always stay compromised and can be recognised by the adversary immediately upon regaining access, regardless of any network changes made by MTD techniques.

4.4.1.4 Exploiting User Credential

After compromising a host, the adversary gains access to all of its users, which they use to launch credential-stuffing attacks on other hosts in the network. If the attacks are successful, the adversary can compromise the target host without the need to exploit its vulnerabilities.

4.4.2 Modelling Attack Event

The attack operation, introduced in Section 4.4, follows the action flow of the Cyber Kill Chain [14]. Figure 7 shows the action flow diagram of the attack operation, which starts with Reconnaissance actions, including "Scan Host" and "Enum Host", to discover a list of hosts and select a target host for subsequent attack actions. These actions, "Scan Port & Exploit User Credential", "Exploit Vulnerabilities", and "Brute Force", are denoted as three phases that can cause the attack event if successful. After a successful attack event, the adversary performs "Scan Neighbour" to discover the reachable neighbour hosts of the compromised host.

The first attack action, referred to as Phase 1, is called "Scan Port & Exploit User Credential." During this phase, the adversary will perform port scanning on the host to look for services with exploitable vulnerabilities. They will also perform a user credential attack by checking usernames and passwords retrieved from other compromised hosts. If the user credential attack fails, the adversary will move on to Phase 2, denoted as "Exploit Vulnerabilities." In this phase, the adversary will attempt to exploit vulnerabilities in the services discovered in Phase 1. If these vulnerabilities cannot be exploited, the adversary will move on to Phase 3, "Brute Force." Here, the adversary will attempt to brute force a user login. If any of these

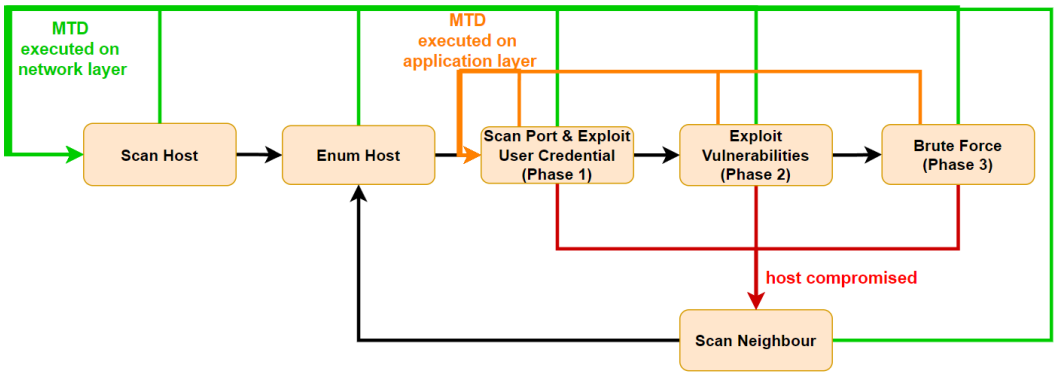


Figure 7: The Action Flow of the Attack Operation

actions succeed, the target host will be compromised.

However, the application of different MTD techniques can lead to the interruption or termination of ongoing actions. If network layer MTD techniques such as IP Shuffle and Complete Topology Shuffle are deployed, the attack event will fail immediately upon the adversary's discovery of a host. These techniques can alter the IP addresses or connection paths of each host, which can block the adversary's action and prevent them from connecting to the target host. On the other hand, application layer MTD techniques like OS Diversity and Service Diversity cannot block attack actions as they do not affect the network topology. However, they can still interrupt attack actions by reconfiguring or deploying services on another port of the target host. In this case, the adversary must restart the attack action from Phase 1, regardless of their progress before. Additionally, a limited number of attempts will be allowed for each attack action. If the number of interruptions caused by application layer MTD techniques reaches a certain threshold, the attack event will fail and the adversary will shift their focus to other hosts in the network

4.4.3 Simulating Time for Adversary

As mentioned in 4.4.2, there are three action phases that may compromise a host in the whole action flow of the attack operation. The time consumption of Phase 1 and Phase 3 is assumed to be no external interference. For Phase 1, the speed of the port scanning is usually constant based on the same scan strategy while the brute force is set to have a specific time limit in Phase 3. The time needed for

Phase 2, however, varies based on several factors, which are either on the server side or the adversary side.

In terms of the server side, the time duration of Phase 2 can be affected by the attack complexity (AC_v) of the service running on the target host. The AC_v would be produced randomly from a range of [0, 1] to reflect the likelihood that the attack would succeed, with 0 denoting the unexploitable vulnerability and 1 denoting the easiest exploitable vulnerability.

For the adversary side, Phase 2 can be affected by the learning capabilities and confusion of the adversary. In terms of the learning capabilities, the attack model is designed to halve the exploitation time spent on the vulnerabilities that were exploited in the previous attack operations conducted on previous hosts. The adversary in the simulator is designed to take a time penalty to simulate the confusion caused by MTD. The time penalty is assigned each time the attack event is interrupted or stopped by MTD.

In *MTDSimTime*, the time duration value of Phase 2 is modeled by associating an exponential time value $T_{A_{exploit}}$ with AC_v. Specifically, for a running service that has a list of vulnerabilities V, where $V_{exploited}$ vulnerabilities were exploited and $V_{unexploited}$ vulnerabilities have not been exploited yet (Equation 1). The formula to calculate the time duration of Phase 2 $T_{A_{phase2}}$ is shown in Equation 2.

$$V = V_{unexploited} + V_{exploited} \tag{1}$$

$$T_{A_{phase2}} = [\sum_{v_i \in V_{unexploited}} (1 - AC_{v_i}) + \sum_{v_j \in V_{exploited}} (1 - AC_{v_j})/2] * T_{A_{exploit}} \tag{2}$$

4.5 Modelling Simulation Time

Discrete Event Simulation (DES) involves simulating time for every time-consuming action in an event to integrate the time domain into the simulator. Moving simulated time from one value to the next is a crucial element of the process. To achieve this, *MTDSimTime* utilises the SimPy framework to control time passing

and event processing [29].

Given the high degree of randomness present in real-world systems, probability and statistics concepts are indispensable in simulation studies. It is critical to identify the probability distribution functions that accurately represent the input data. To this end, the exponential distribution is widely used in simulating the distribution of elapsed times for actions and has therefore been selected as the primary probability distribution method [28]. Equation 3 shows the probability density function (PDF) for the exponential distribution.

$$f(x) = 1/\mu \mathrm{e}^{-\frac{1}{\mu}x} \tag{3}$$

Typically, the random variable x can represent either (a) the time elapsed between two consecutive events, such as the *MTD interval*, or (b) the duration of an action. The parameter μ is the historical average elapsed time, which can be determined via empirical study and sensitivity analysis. Using the probability density function presented earlier, we can derive the corresponding cumulative density function (CDF), which is shown in Equation 4.

$$F(x) = \int_0^{\infty} \left[\frac{1}{\mu} e^{-\frac{x}{\mu}} \right] = 1 - e^{-\frac{x}{\mu}} \tag{4}$$

With the CDF, the probability that a time-consuming action finishes execution before a certain time $x = a$ can be calculated as $P(x < a) = 1 - e^{-\frac{a}{\mu}}$, which allows for the generation of the probabilistic value of the simulation time for each time-consuming action.

5 Evaluation

The evaluation methodology for this project comprises two main parts. The first part evaluates each of the five MTD techniques introduced in section 4.3.1 individually, which represents a baseline for assessing the combination strategies. In the second part, we assess the effectiveness of pairwise MTD combination strategies, along with different execution schemes introduced in section 4.3.2. To explore the factors that may affect the effectiveness of MTDs, we test the impact of four *MTD Intervals*, ranging from 50 to 200 seconds, and four networks with different sizes, ranging from 25 to 100 nodes. The number of nodes in the network is denoted as *Network Size*.

In addition, we use Network Compromise Ratio (NCR) as the simulation checkpoint to evaluate MTD techniques and their combinations. NCR is the ratio of compromised hosts to the total number of hosts in the network. It helps us measure the effectiveness of our MTD techniques by measuring the time it takes for the adversary to compromise a certain proportion of hosts in the network. In this perspective, we set the terminating condition for the simulation as the NCR reaching 0.8 and run the simulation 100 times for each variable set to obtain statistically significant results. This approach enables us to assess the impact of different conditions on the performance of the MTD techniques and their combinations comprehensively, offering valuable insights for future research in this area.

Table 2: Network Properties

Network	**Nodes**	**Density**	**Endpoints**	**Layers**
1	25	0.093	3	4
2	50	0.052	5	4
3	75	0.040	7	4
4	100	0.043	10	4

Table 3: MTD Execution Time

MTD technique	Average Duration (s)	Standard Deviation
CompleteTopologyShuffle	110	0.5
IPShuffle	100	0.5
OSDiversity	80	0.5
DAP_ OSDiversity	80	0.5
ServiceDiversity	70	0.5

5.1 Single MTD Evaluation

Figure 8 presents the average time for the adversary to compromise 80% nodes of the target network for each MTD technique under different scenarios, with each subplot depicting a unique scenario. Specifically, different *MTD Intervals* are shown in the vertical perspective, while networks with varying numbers of nodes are shown in the horizontal perspective.

5.1.1 Effectiveness of MTD techniques

The figure clearly illustrates that employing MTD techniques can significantly increase the time it takes for an adversary to achieve their objective, as compared to the case where no MTDs are employed. In addition, The figure provides clear evidence that MTD techniques based on network shuffling, including *IPShuffle* and *CompleteTopologyShuffle*, consistently outperform other MTD techniques. In the majority of the depicted scenarios, these network shuffle-based MTD techniques demonstrate a significant improvement of over 130% (Figure 9) compared to the other techniques on average. The results emphasize the effectiveness of network shuffling in enhancing security measures and highlight its superiority in the evaluated scenarios. The observed superiority may be attributed to the disruption of connections caused by these MTD techniques, as discussed in Section 4.4.2. These re-configurations require the adversary to spend extra time relocating their targets, impeding their ability to continue exploiting their previous targets. In contrast, diversity-based MTD techniques only reconfigure components within each host, allowing the adversary to resume the attack after interruption. Nevertheless, it can be inferred that MTD techniques, especially network shuffle-based ones, can

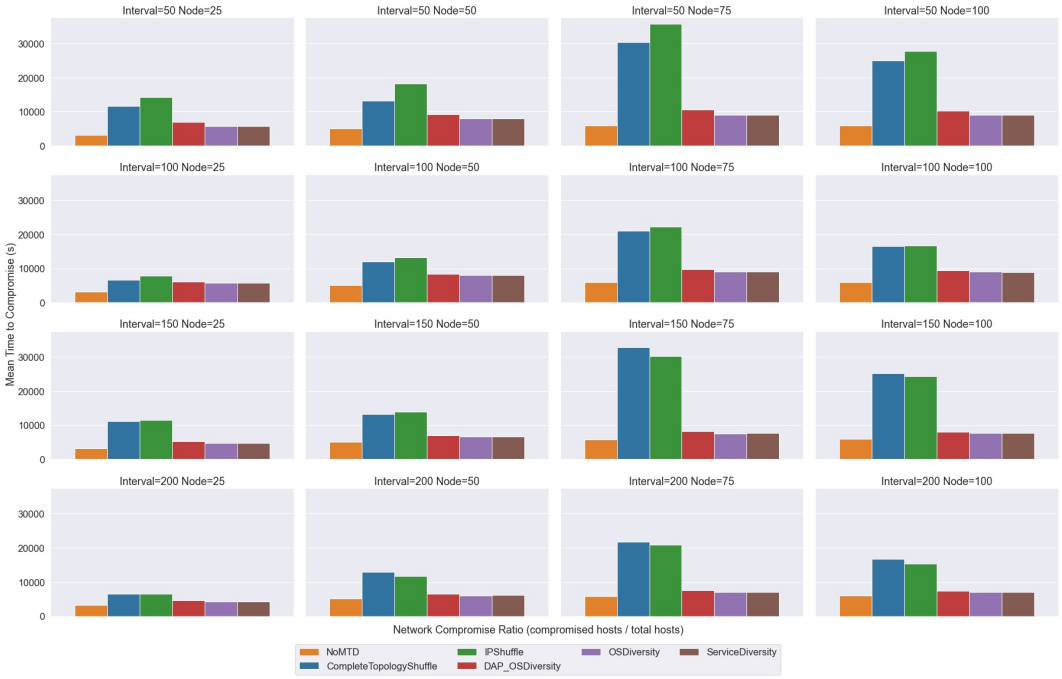


Figure 8: Single MTD Evaluation - Mean Time to Compromise on 0.8 NCR

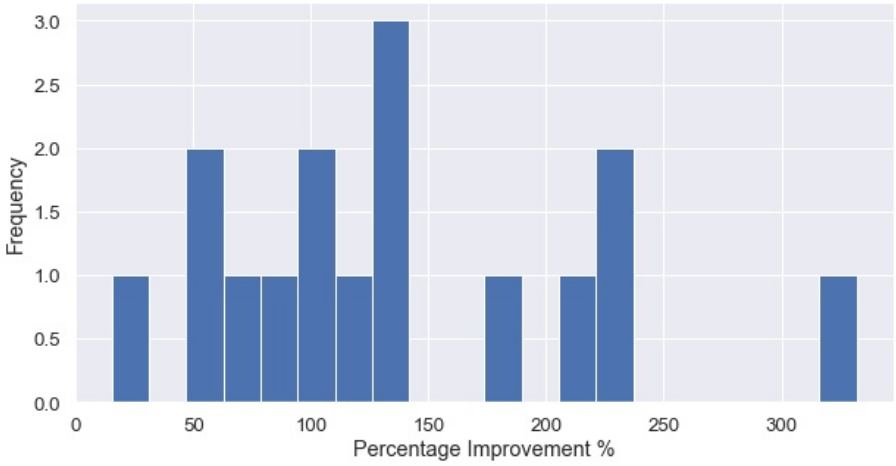


Figure 9: Distribution of Percentage Improvement: Network Shuffle MTD Outperforms Diversity MTD

effectively enhance a network's security against attacks.

5.1.2 Impact of MTD Interval

Regarding the *MTD Interval*, it is evident that MTD techniques are more effective on shorter intervals in most scenarios. However, when comparing the results for interval=100 and interval=150, diversity-based MTD techniques were generally more effective for interval=100, whereas the results for two shuffle-based MTD techniques exhibited a different trend. It is assumed that this difference in performance may be attributed to the longer execution time required by the shuffle-based MTD techniques. As shown in Table 3, the mean execution times for these MTDs are longer than the triggering interval, which may cause issues related to suspension actions and ultimately result in less effective outcomes.

5.1.3 Impact of Network Size

In terms of *Network Size*, MTD techniques are generally more effective on networks with more nodes. However, when comparing the results for node=75 and node=100, a different trend was observed. The differing outcomes may be attributed to the density of the networks, as shown in Table 2. The network with 75 nodes has a lower density than the network with 100 nodes. By associating results with network densities in other scenarios, it is assumed that MTD techniques are more effective on networks with more nodes and lower densities. However, to establish this relationship with greater certainty, further research and evidence may be necessary.

5.2 Multiple MTD Evaluation

5.2.1 Effectiveness of MTD pairwise combination strategies

Figure 10 illustrates the results of each pairwise MTD combination strategy. Like the single MTD evaluation, we conducted the multiple MTD evaluation under 16 scenarios. However, this time we considered the execution scheme as a variable in the experiment, as shown on the x-axis of each subplot. The figure reveals that the combination results followed a similar trend to the single MTD results for different

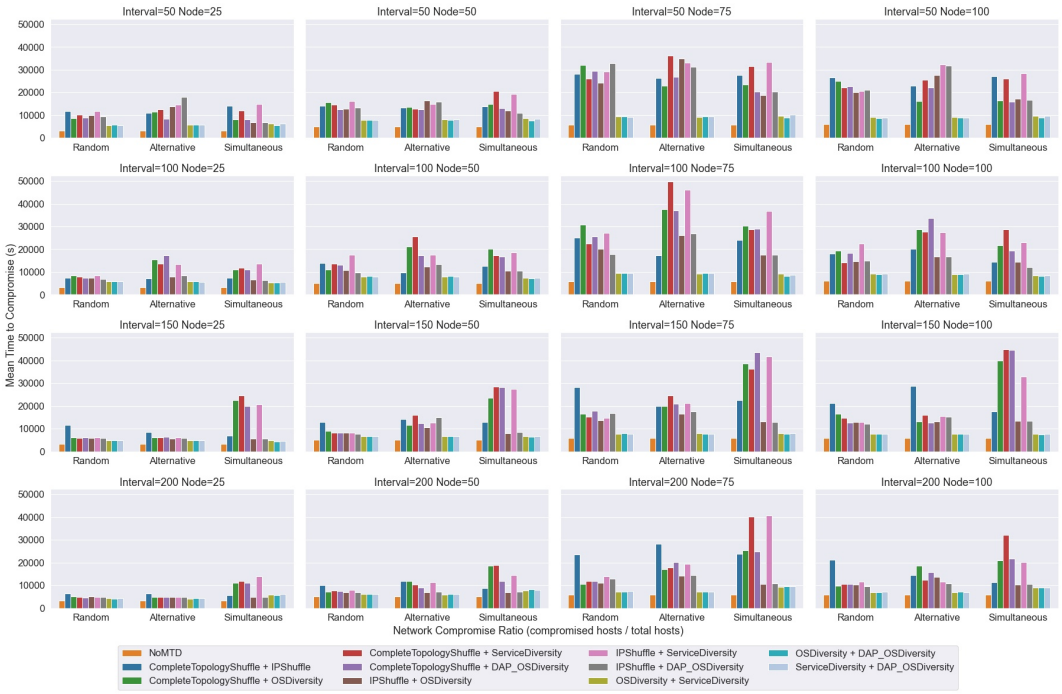


Figure 10: Multiple MTD Evaluation - Mean Time to Compromise on 0.8 NCR

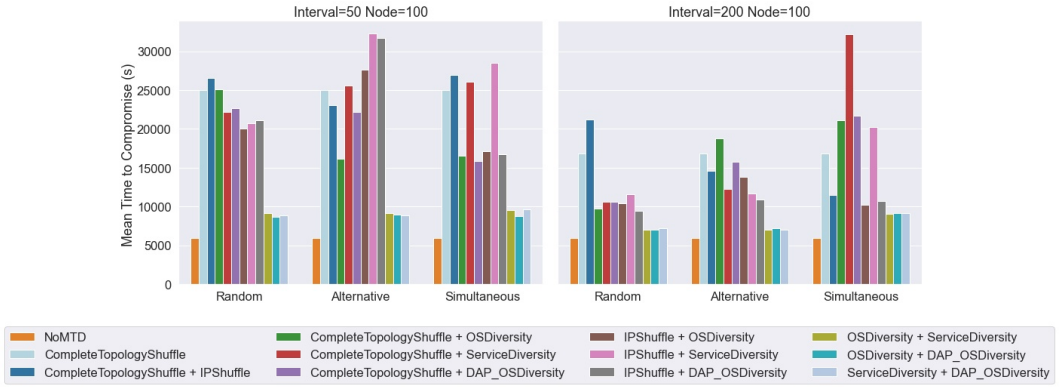


Figure 11: Multiple MTD Evaluation - Comparing scenarios with Interval=50 and Interval=200 on Node=100

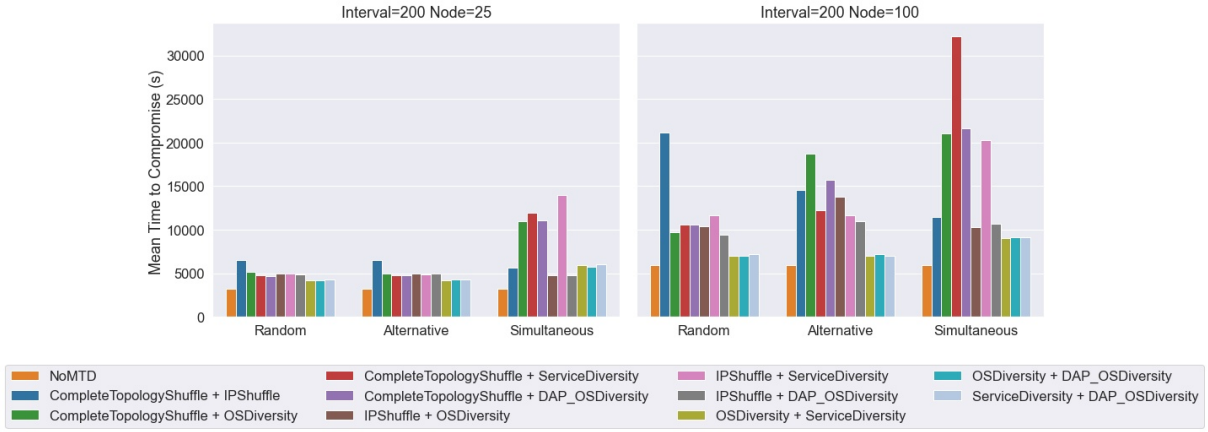


Figure 12: Multiple MTD Evaluation - Comparing scenarios with Node=50 and Node=200 on Interval=100

MTD Intervals and *Network Sizes*. To provide a closer examination, Figure 11 and Figure 12 offer enlarged views of the subplots in Figure 10, enabling clear comparisons between scenarios with the same node but different intervals (Figure 11), as well as scenarios with the same intervals but different nodes (Figure 12).

However, the top 2 MTD techniques, namely *CompleteTopologyShuffle* and *IPShuffle*, which were most effective when used alone, did not have the same dominant position when they were used in combination. This may be because both of these MTD techniques occupy resources in the network layer, which means they cannot be deployed concurrently. Moreover, the configuration of resources in the application layer remains static, which can make it easier for adversaries to exploit. To address this, combining MTD techniques from different categories, such as *CompleteTopologyShuffle* + *OSDiversity*, *IPShuffle* + *ServiceDiversity*, and *CompleteTopologyShuffle* + *ServiceDiversity*, demonstrated better performance with an average improvement of approximately 20% (Figure 13). Refer to Section 4.3.3 for more details on the resource occupation issue.

Figure 11 also includes the results of a single MTD technique (*CompleteTopologyShuffle*) that achieved the best result in the single MTD evaluation. The figure suggests that network shuffle-based MTD techniques exhibit similar effectiveness when used in combination compared to when used individually. Therefore,

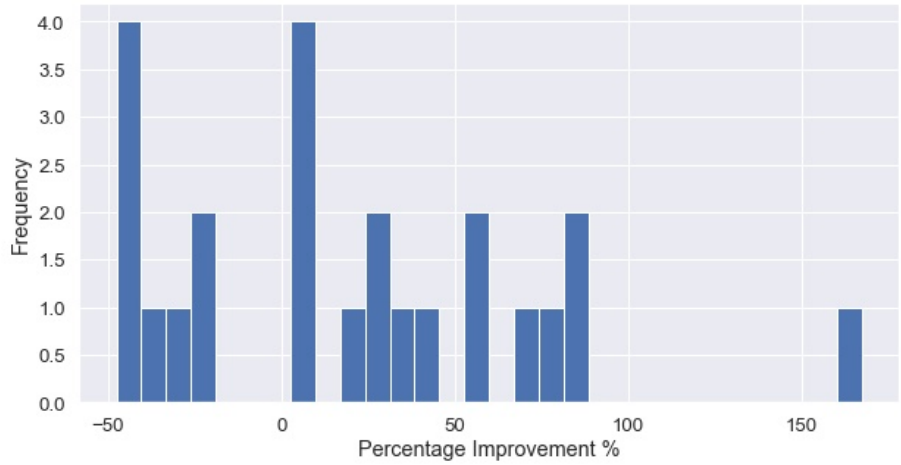


Figure 13: Distribution of Percentage Improvement: Combining Multiple Category MTDs over Combining Shuffle MTDs

in certain scenarios, it may be advantageous to utilise a single shuffle-based MTD technique instead of combining them. This approach offers minimal performance difference while simplifying the deployment process. However, it is important to note that the two network shuffle-based MTD techniques employed in this experiment assume the same resource utilisation (specifically, network layer resources) within the simulation environment, while not considering the computational resource consumption associated with executing MTD techniques. In real-world scenarios, these techniques may differ in their computational resource consumption, which becomes particularly significant when considering the trade-off between resource usage and security. Thus, further research is needed to optimise the selection of MTD techniques in practical scenarios, taking into account both their performance and resource requirements.

5.2.2 Effectiveness of MTD Execution Scheme

Figure 14 illustrates the maximum count of each MTD execution scheme for different pairwise combinations of MTD techniques, as presented in Figure 10. The maximum count is determined by comparing the performance of each MTD combination strategy across various scenarios (node and interval).

To gain a comprehensive understanding of the performance of different schemes in

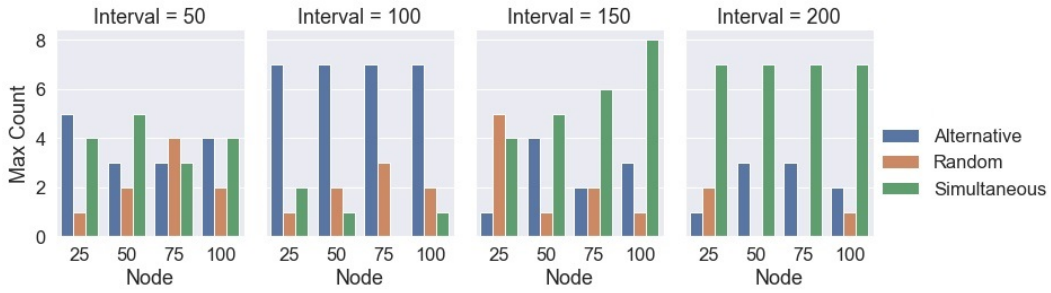


Figure 14: Maximum Count of Each MTD Execution Scheme

specific scenarios, we will start by analysing the leftmost subplot of Figure 14 as a representative sample, providing insights into the overall figure. For instance, when considering the Alternative scheme with node=25 and interval=50, a Max Count value of 5 indicates that there are five MTD combinations achieving the best results. This means a higher Max Count value suggests a more favourable performance for the scheme in that particular scenario. Overall, the figure demonstrates consistent superiority of the Alternative and Simultaneous schemes over the Random scheme in most scenarios, highlighting their effectiveness in enhancing MTD performance.

Regarding the relationship with the factors, no clear correlation is observed between the number of nodes and the effectiveness of the execution schemes. However, in terms of intervals, it can be inferred from the figure that the Alternative scheme is more effective for shorter intervals (50, 100), while the Simultaneous scheme performs better for longer intervals (150, 200). This discrepancy may arise because the Simultaneous scheme registers all MTD techniques simultaneously, providing an advantage over schemes that register one MTD at a time when the triggering interval is longer. Nonetheless, it is important to note that the Simultaneous scheme is more resource-intensive, and the Alternative scheme exhibits better performance under shorter intervals compared to the Simultaneous scheme. Therefore, when selecting an MTD execution scheme, careful consideration of the specific scenario and available resources is crucial to ensure optimal outcomes.

6 Scope for Future Work

This research aimed to evaluate the effectiveness of MTD strategies in different scenarios by analysing their performance in the time domain under various intervals and networks. The findings indicate that the effectiveness of MTD techniques varies depending on the specific scenario. Combining multiple categories of MTD techniques tends to be more effective than using fewer categories, as demonstrated by comparing the combined approaches with individual MTD results. While the study has yielded important findings, it is important to acknowledge that certain limitations may have impacted the results in some aspects. In order to ensure the robustness of future research, this section aims to outline these limitations and suggest potential directions for further investigation.

6.1 Factors Impacting MTD Technique Performance

This work examined the impact of two key factors, namely *MTD Interval* and *Network Size*, on the effectiveness of MTD techniques. Specifically, we investigated how different intervals and networks with varying numbers of nodes can influence the effectiveness of MTD techniques.

In terms of the *MTD Interval*, it is expected that the shorter interval will result in a higher frequency of deployment for MTD techniques. However, due to resource occupation and suspension mechanisms, the frequency of deployment for MTD techniques does not necessarily increase with shorter triggering intervals. Notably, it is hypothesized that a higher frequency of deployment could lead to better MTD technique performance, but this hypothesis could not be confirmed in this work due to the time limitation. Therefore, it is recommended that future research should focus on optimising the suspension mechanism and investigating the frequency of deployment for MTD techniques.

Similarly, the challenges encountered in examining the impact of the *Network Size* were akin to those of the *MTD Interval*. Given the multitude of factors influencing a network, it is impossible to maintain uniformity while exploring how varying node numbers can affect MTD performance across different networks. As mentioned

in Section 5, the *Network Density* may also play a pivotal role, but conclusive evidence is required to validate this hypothesis. Furthermore, the complexity of vulnerabilities within each host of a network is a potential factor that could affect the performance of MTD techniques. However, as the vulnerabilities in each host of the simulated network were randomly generated and assigned, it was challenging to measure and control their impact during experimentation. Unfortunately, this work did not consider these factors, nor did it examine the extent to which they impact the effectiveness of MTD techniques. In light of these challenges, future research could explore how other network factors, such as network density and the complexity of vulnerabilities within each host, affect the performance of MTD techniques. Investigating these factors could also aid in generating appropriate time values for different time-consuming actions based on the specific network environment.

6.2 Limited Set of MTD Techniques

Despite the valuable insights gained from this research, the experimental evaluation was limited to a narrow set of MTD techniques. Specifically, only one to two defenses were explored per layer, with just two shuffle and three diversity techniques examined. As a result, the findings may not be generalisable to other MTD techniques or combinations that were not tested in this study. Therefore, it is important to acknowledge the limitations of the current research and to exercise caution when applying the results to real-world scenarios where a broader range of MTD techniques may be employed. Future research could expand upon the current study by incorporating more diverse MTD techniques and examining their performance under different network configurations, including variations in the number of nodes and MTD intervals, to enhance the generalisability of the findings.

6.3 Adversary Profile

The current attack operation in this work consists of a fixed list of actions that can be optimised in various ways. For example, in the target selection phase,

the adversary's current decision is based on the distance to each discovered host, which may not be the most efficient approach as different hosts can have varying levels of vulnerability complexities. To improve this, the adversary could perform a port scan to analyse the exploitation difficulty of each discovered host before making a decision on subsequent attack actions. Also, other factors could be considered when selecting a target, and optimising the attack profile accordingly can significantly increase its intelligence. In this case, introducing a more intelligent attack profile will make it more comprehensive and sophisticated, enabling future works to evaluate the effectiveness of MTD techniques against a more advanced adversary. This will provide valuable insights into improving security measures and defending against sophisticated attacks.

6.4 Reconfiguration Limit

In this work, MTD techniques were modelled to execute on all nodes in the simulated network for the experimental evaluation. However, in a real-world scenario, deploying MTD techniques on all nodes can severely strain resources and negatively impact services. To mitigate this, it is crucial to identify critical nodes or subnets in the network and only reconfigure those when deploying MTD techniques, minimising service interruptions. This selective approach can be achieved by using various techniques such as network analysis, traffic profiling, or vulnerability assessments. However, we did not incorporate this mechanism in our simulation due to time constraints and a lack of data on user profiles and network interactions, making it difficult to measure how MTD techniques affect the quality of service (QoS). Nonetheless, incorporating this feature in the simulator can help us extend our topic on balancing system security and QoS while deploying MTD techniques.

7 Conclusion

This research has contributed to the growing body of knowledge on MTD strategies and their effectiveness in securing networks against attackers. The development of the simulation framework, *MTDSimTime*, has enabled a more efficient and accurate evaluation of MTD techniques in the time domain, which can aid in the development of more robust and effective MTD strategies.

By systematically evaluating and comparing the effectiveness of various MTD strategies in the time domain, this study has provided valuable insights into the performance of different MTD strategies, considering their techniques, combinations, and execution schemes. The findings reveal that network shuffle-based MTD techniques consistently outperform diversity-based MTD techniques. However, it is noteworthy that the combination of both approaches, specifically Shuffle + Diversity, emerged as the most effective strategy for enhancing network security. These results contribute to a deeper understanding of the effectiveness of MTD techniques and provide valuable guidance for the development of robust defensive strategies.

However, there is a need for further research to evaluate the performance of a broader range of MTD techniques and combinations under various network configurations. A more intelligent attack profile can also help to assess the effectiveness of MTD techniques against sophisticated attacks and provide insights into improving security measures. Furthermore, a selective approach to deploying MTD techniques can help minimise resource occupation and service interruptions in a real-world scenario. These areas require further investigation to improve the effectiveness of MTD strategies in securing networks against attackers.

8 References

[1] C. Ruhl, D. Hollis, W. Hoffman, and T. Maurer, "Cyberspace and geopolitics: Assessing global cybersecurity norm processes at a crossroads." Carnegie Endowment for International Peace., 2020.

[2] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Moving target defense: creating asymmetric uncertainty for cyber threats. Springer Science and Business Media, 2011.

[3] A. Alshamrani, S. Myneni, A. Chowdhary and D. Huang, "A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities," in IEEE Communications Surveys and Tutorials, vol. 21, no. 2, pp. 1851-1877, Secondquarter 2019, doi: 10.1109/COMST.2019.2891891.

[4] J. -H. Cho et al., "Toward Proactive, Adaptive Defense: A Survey on Moving Target Defense," in IEEE Communications Surveys and Tutorials, vol. 22, no. 1, pp. 709-745, Firstquarter 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," in IEEE Communications Surveys and Tutorials, vol. 22, no. 3, pp. 1909-1941, thirdquarter 2020, doi: 10.1109/COMST.2020.2982955.

[6] G.-L. Cai, B.-S. Wang, W. Hu, and T.-Z. Wang, “Moving target defense: State of the art and characteristics,” Front. Inf. Technol. Electron. Eng., vol. 17, no. 11, pp. 1122–1153, Nov. 2016

[7] J. Hong and D. S. Kim, “HARMs: Hierarchical attack representation models for network security analysis,” Proceedings of the 10th Australian Information Security Management Conference, AISM 2012, pp.74–81, 2012.

[8] 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 International Conference on Information Security Practice and Experience, 2017: Springer, pp. 539-548.

[9] H. Alavizadeh, J. B. Hong, D. S. Kim, and J. Jang-Jaccard, "Evaluating the effectiveness of shuffle and redundancy mtd techniques in the cloud," Computers and Security, vol. 102, p. 102091, 2021.

[10] 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.

[11] H. Alavizadeh, D. S. Kim, and J. Jang-Jaccard, "Model-based evaluation of combinations of shuffle and diversity MTD techniques on the cloud," Future Generation Computer Systems, vol. 111, pp. 507-522, 2020.

[12] H. Alavizadeh, J. B. Hong, J. Jang-Jaccard, and D. S. Kim, "Comprehensive security assessment of combined MTD techniques for the cloud," in Proc. 5th ACM Workshop Moving Target Defense (MTD), 2018, pp. 11–20

[13] "Discrete-event simulation - Wikipedia", En.wikipedia.org, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Discrete-event_simulation. [Accessed: 14- Aug- 2022]

[14] L. Martin. (2017). Cyber Kill Chain (CKC). URL: https://www.lockheedmartin.com/enus/capabilities/cyber/cyber-kill-chain.html

[15] MITRE ATT&CK Framework: Adversary Tactics and Techniques. 2015. URL: https://attack.mitre.org/

[16] J. Livingston. (2022). "What is MITRE ATT&CK? The Definitive Guide." 2022. URL: https://verveindustrial.com/resources/blog/what-is-mitre-attack-framework/

[17] A. Brown, T. Lee, and J. Hong, "Evaluating Moving Target Defenses against Realistic Attack Scenarios," in Proc. 4th Int. Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS), May 2023.

[18] Z. Chen, X. Chang, J. Mišić, V. B. Mišić, Y. Yang and Z. Han, "Model-based Performance Evaluation of a Moving Target Defense System," GLOBECOM

2020 - 2020 IEEE Global Communications Conference, 2020, pp. 1-6, doi: 10.1109/GLOBECOM42002.2020.9322609.

[19] W. Connell, D. A. Menascé, and M. Albanese, "Performance modeling of moving target defenses," in Proceedings of the 2017 Workshop on Moving Target Defense, 2017, pp. 53-63.

[20] W. Connell, D. A. Menascé and M. Albanese, "Performance Modeling of Moving Target Defenses with Reconfiguration Limits," in IEEE Transactions on Dependable and Secure Computing, vol. 18, no. 1, pp. 205-219, 1 Jan.-Feb. 2021, doi: 10.1109/TDSC.2018.2882825.

[21] SimPy (2002-2020). SimPy: Simulation framework in Python. URL: https://simpy.readthedocs.io/en/latest/index.html

[22] M. Torquato, P. Maciel and M. Vieira, "PyMTDEvaluator: A Tool for Time-Based Moving Target Defense Evaluation: Tool description paper," 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), 2021, pp. 357-366, doi: 10.1109/ISSRE52982.2021.00045.

[23] X. Xiong, L. Ma, and C. Cui, "Simulation Environment of Evaluation and Optimization for Moving Target Defense: A SimPy Approach," in Proceedings of the 2019 the 9th International Conference on Communication and Network Security, 2019, pp. 114-117.

[24] 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," Computers & Security, vol. 79, pp. 33-52, 2018.

[25] C. L. Belusso, S. Sawicki, F. Roos-Frantz, and R. Z. Frantz, "A study of Petri Nets, Markov chains and queueing theory as mathematical modeling languages aiming at the simulation of enterprise application integration solutions: a first step," Procedia Computer Science, vol. 100, pp. 229-236, 2016.

[26] G. Cai, B. Wang, Y. Luo, and W. Hu, "A model for evaluating and comparing moving target defense techniques based on generalized stochastic Petri net,"

in Conference on Advanced Computer Architecture, 2016: Springer, pp. 184-197.

[27] D. Weitz, "Introduction to Simulation with SimPy Part 1: The Basics", Towards Data Science. [Online]. Available: https://towardsdatascience.com/introduction-to-simulation-with-simpy-b04c2ddf1900. [Accessed: 22- Aug- 2022].

[28] D. Weitz, "Introduction to Simulation with SimPy Part 2: Measures of Performance for Queuing Systems", Towards Data Science. [Online]. Available: https://towardsdatascience.com/introduction-to-simulation-with-simpy-322606d4ba0c. [Accessed: 22- Aug- 2022].

[29] SimPy (2002-2020). SimPy: Time and Scheduling. [Online]. Available: https://simpy.readthedocs.io/en/latest/topical_guides

[30] J. H. Jafarian, E. Al-Shaer, and Q. Duan. “Openflow Random Host Mutation: Transparent Moving Target Defense Using Software Defined Networking”. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks. HotSDN ’12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 127–132. ISBN: 9781450314770. DOI: 10.1145/2342441.2342467. URL: https://doi.org/10.1145/2342441.2342467.

[31] D. Kewley et al. “Dynamic approaches to thwart adversary intelligence gathering”. In: Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX’01. Vol. 1. 2001, pp. 176–185.

[32] J. B. Hong et al. “Optimal network reconfiguration for software defined networks using shuffle-based online MTD”. In: Proceedings of the IEEE Symposium on Reliable Distributed Systems 2017-Septe (2017), pp. 234–243. ISSN: 10609857. DOI: 10.1109/SRDS.2017.32.

[33] L. Wang. “Shoal: A Network Level Moving Target Defense Engine with Software Defined Networking”. In: EAI Endorsed Transactions on Security and Safety 7.25 (2021), e5.

[34] S. Achleitner et al. “Deceiving network reconnaissance using SDN-based virtual topologies”. In: IEEE Transactions on Network and Service Management 14.4 (2017), pp. 1098–1112.

[35] Y. B. Luo, B. S. Wang, and G. L. Cai. “Effectiveness of port hopping as a moving target defense”. In: Proceedings - 7th International Conference on Security Technology, SecTech 2014 (2015), pp. 7–10. DOI: 10.1109/SecTech.2014.9.

[36] S. Vikram, C. Yang, and G. Gu. “Nomad: Towards non-intrusive moving-target defense against web bots”. In: 2013 IEEE Conference on Communications and Network Security (CNS). 2013, pp. 55–63.

[37] V. Casola, A. D. Benedictis, and M. Albanese. “A moving target defense approach for protecting resource-constrained distributed devices”. In: 2013 IEEE 14th International Conference on Information Reuse & Integration (IRI). 2013, pp. 22–29.

[38] A. Newell, D. Obenshain, T. Tantillo, C. Nita-Rotaru, and Y. Amir, "Increasing network resiliency by optimally assigning diverse variants to routing nodes," IEEE Transactions on Dependable and Secure Computing, vol. 12, no. 6, pp. 602-614, 2014.

[39] M. Thompson, N. Evans, and V. Kisekka, “Multiple OS rotational environment an implemented moving target defense,” in Proc. 7th Int. Symp. Resilient Control Syst. (ISRCS), 2014, pp. 1–6.

[40] A. B. Downey, "Think Complexity, 2nd ed.," O’Reilly Media, Inc., 2021