Universidade Federal do Piauí
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação

Hybrid and Decoy Moving Target Defense in Cloud Computing: A Performance Modeling Approach

Lucas Vinícius Silva dos Santos

Número de Ordem PPGCC:

Picos-PI, Outubro de 2025

Lucas Vinícius Silva dos Santos

Hybrid and Decoy Moving Target Defense in Cloud Computing: A Performance Modeling Approach

Defesa de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Universidade Federal do Piauí – UFPI

Centro de Ciências da Natureza

Programa de Pós-Graduação em Ciência da Computação

Supervisor: Prof. Dr. Francisco Airton Pereira da Silva

Co-supervisor: Prof. Dr. Matheus D'Eça Torquato de Melo

Picos-PI

Outubro de 2025

FICHA CATALOGRÁFICA
Universidade Federal do Piauí
Biblioteca Comunitária Jornalista Carlos Castello Branco
Serviço de Processos Técnicos

S237h Santos, Lucas Vinícius Silva dos.
Hybrid and Decoy Moving Target Defense in Cloud Computing : A Performance Modeling Approach / Lucas Vinícius Silva dos Santos. – 2025.
75 f. : il.

Dissertação (Mestrado) – Universidade Federal do Piauí, Centro de Ciências da Natureza, Programa de Pós-Graduação em Ciências da Computação, Picos, 2025.
"Supervisor: Prof. Dr. Francisco Airton Pereira da Silva."
"Co-supervisor: Prof. Dr. Matheus D'Eça Torquato de Melo."

1. Segurança da Informação. 2. Moving Target Defense. 3. Migração. 4. Desempenho. 5. Petri Nets. 6. Isca. 7. IDS. I. Silva, Francisco Airton Pereira da. II. Melo, Matheus D'Eça Torquato de. III. Título.

CDD 005.8

Bibliotecário: Gésio dos Santos Barros - CRB3/1469

Lucas Vinícius Silva dos Santos

Hybrid and Decoy Moving Target Defense in Cloud Computing: A Performance Modeling Approach

Defesa de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Documento assinado digitalmente
gov.br
FRANCISCO AIRTON PEREIRA DA SILVA
Data: 22/10/2025 10:04:26-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Francisco Airton Pereira da Silva
Orientador

Documento assinado digitalmente
gov.br
MATHEUS D ECA TORQUATO DE MELO
Data: 22/10/2025 22:15:18-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Matheus D'Eça Torquato de Melo
Co-orientador

Documento assinado digitalmente
gov.br
GLAUBER DIAS GONCALVES
Data: 22/10/2025 17:38:27-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Glauber Dias Gonçalves
Membro da Banca

Documento assinado digitalmente
gov.br
ERMESON CARNEIRO DE ANDRADE
Data: 22/10/2025 23:31:21-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Ermesson Carneiro de Andrade
Membro da Banca

Documento assinado digitalmente
gov.br
RAYNER GOMES SOUSA
Data: 23/10/2025 16:13:12-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. rayner gomes sousa
Membro da Banca

Picos-PI
Outubro de 2025

Resumo

A segurança da informação enfrenta desafios cada vez mais complexos, exigindo estratégias inovadoras para aumentar a resiliência dos sistemas contra ataques. O conceito de *Moving Target Defense* (MTD) surge como uma abordagem promissora, pois dificulta a identificação e exploração de vulnerabilidades ao modificar dinamicamente a configuração do sistema. Diferentemente dos mecanismos de defesa tradicionais, que permanecem estáticos e previsíveis, o MTD busca reduzir a vantagem do atacante pela constante mudança do ambiente alvo. Entre as estratégias possíveis, destacam-se a migração de máquinas virtuais e a realocação de endereços IP, bem como o uso de servidores de *decoy*, que atuam como iscas para desorientar invasores e atrasar sua progressão. Este estudo propõe uma avaliação comparativa entre duas metodologias distintas de MTD, ambas modeladas por meio de redes de Petri. A primeira abordagem baseia-se na migração como estratégia defensiva, considerando três políticas de ativação: baseada em tempo, baseada em eventos e uma abordagem híbrida que combina ambas. Para fortalecer a resposta a ataques, foi incorporado um sistema de detecção de intrusão (IDS), responsável por identificar atividades maliciosas e acionar a migração do sistema conforme necessário. A segunda abordagem analisa o impacto do uso de servidores de *decoy*, combinados com a migração dinâmica de endereços IP, com o objetivo de dificultar o mapeamento dos servidores reais e aumentar a incerteza do atacante quanto à topologia do sistema. Os resultados das simulações indicam que a eficácia das políticas de detecção baseadas em eventos depende fortemente da precisão do IDS, sendo mais vantajosa quando esta ultrapassa 50%. Além disso, a abordagem híbrida demonstrou ser a mais eficiente para retardar o progresso de invasores, equilibrando segurança e desempenho. No contexto dos servidores de decoy, os experimentos revelaram que a introdução de um maior número de servidores falsos aumenta significativamente o tempo necessário para uma intrusão bem-sucedida, elevando-o de 8 para até 19 dias. Dessa forma, este trabalho contribui para a compreensão da aplicabilidade das diferentes estratégias de MTD, fornecendo análises sobre como políticas de migração e o uso de servidores de decoy podem ser otimizados para melhorar a segurança de infraestruturas computacionais contra ataques cibernéticos.

Palavras-chaves: Moving Target Defense, IDS, Migração, Segurança da Informação, Desempenho, Petri Nets, Isca;

Abstract

Information security faces increasingly complex challenges, requiring innovative strategies to enhance system resilience against attacks. The concept of Moving Target Defense (MTD) emerges as a promising approach, as it hinders the identification and exploitation of vulnerabilities by dynamically modifying the system's configuration. Unlike traditional defense mechanisms, which remain static and predictable, MTD seeks to reduce the attacker's advantage through constant changes in the target environment. Among the possible strategies, notable examples include virtual machine migration and IP address relocation, as well as the use of decoy servers, which act as traps to mislead intruders and delay their progression. This study proposes a comparative evaluation between two distinct MTD methodologies, both modeled through Petri nets. The first approach relies on migration as a defensive strategy, considering three activation policies: time-based, event-based, and a hybrid approach combining both. To strengthen the response to attacks, an intrusion detection system (IDS) was incorporated to identify malicious activities and trigger system migration as needed. The second approach analyzes the impact of using decoy servers combined with dynamic IP address migration, aiming to hinder the mapping of real servers and increase the attacker's uncertainty regarding the system topology. Simulation results indicate that the effectiveness of event-based detection policies strongly depends on IDS accuracy, being more advantageous when accuracy exceeds 50%. Furthermore, the hybrid approach proved to be the most effective in slowing the progress of intruders, balancing security and performance. In the context of decoy servers, experiments revealed that introducing a larger number of fake servers significantly increases the time required for a successful intrusion, raising it from 8 to as many as 19 days. Thus, this work contributes to understanding the applicability of different MTD strategies, providing insights into how migration policies and the use of decoy servers can be optimized to enhance the security of computational infrastructures against cyberattacks.

Keywords: Moving Target Defense, IDS, Migration, Cybersecurity, Performance, Petri Nets, Decoy.

List of Figures

List of Tables

List of abbreviations and acronyms

DoE	Design of Experiments
DSPN	Deterministic and Stochastic Petri Net
PN	Petri Net
MTD	Moving Target Defense
IDS	Intrusion Detection System
MRT	Mean Response Time
VM	Virtual Machine
MTTA	Mean Time to Absorption
CDF	Cumulative Distribution Function

Contents

1 Introduction

Today's society has become deeply dependent on interconnected digital systems that permeate various domains of daily life, including finance, data storage, and healthcare, as emphasized in previous research (HASSAN et al., 2020). These systems underpin critical operations and provide the foundation for economic and social stability. However, as reliance on such technology increases, the complexity and scale of cybersecurity challenges grow proportionally. Cyber threats continue to evolve, leveraging sophisticated methods to exploit weaknesses in existing infrastructures. Recent studies have highlighted that current computer systems still exhibit significant vulnerabilities (TUFAIL et al., 2021), which, if exploited, can lead to catastrophic financial losses, operational disruptions, and breaches of sensitive data (NAFEA; ALMAIAH, 2021). These risks demand a proactive and adaptive approach to securing digital infrastructures.

The implementation of robust cybersecurity measures remains critical to safeguarding systems against ever-evolving threats. Traditional security practices have been instrumental in hardening digital environments. Firewalls, for instance, play a pivotal role in monitoring and regulating network traffic to prevent unauthorized access (LIANG; KIM, 2022). Similarly, data encryption ensures confidentiality by converting sensitive information into unreadable formats for unauthorized users (CHINNASAMY et al., 2021). IDS add another layer of defense by identifying and alerting administrators to potential breaches (THAKKAR; LOHIYA, 2022). While these measures are foundational, their static nature introduces predictability, which attackers can exploit. By gathering intelligence on system configurations, attackers can leverage this predictability to tailor their attacks with high precision.

To address the limitations of static defenses, innovative approaches like MTD have emerged as game-changers in the field of cybersecurity. Unlike traditional measures, MTD employs dynamic strategies to continuously change the available attack surface, effectively disrupting attackers' reconnaissance efforts. This unpredictability forces adversaries to adapt continuously, significantly increasing the difficulty of launching successful attacks (TAN et al., 2023). MTD can be implemented through various mechanisms, such as dynamically changing IP addresses, migrating critical processes or services to different locations, and deploying deceptive traps to mislead attackers. These tactics not only protect crucial system components but also impose additional costs on attackers, reducing the likelihood of attack success.

The adoption of MTD represents a paradigm shift in cybersecurity, emphasizing adaptability and unpredictability over static resilience. By dynamically altering the attack

surface, MTD undermines the traditional strategies employed by attackers, who often rely on prolonged observation and system familiarity. As research in this area continues to expand, organizations are encouraged to integrate MTD techniques alongside traditional measures to create a layered, multi-faceted defense strategy. This combined approach ensures a more robust security posture, safeguarding critical systems from the ever-growing spectrum of cyber threats.

In MTD there are three principal types of activation policies: Time-Based Policies, which schedule MTD activations at regular, predefined intervals to ensure a consistent defense strategy over time; Event-Based Policies, which are triggered by specific incidents, such as the detection of anomalies or attacks, thereby responding directly to identified threats; and Hybrid Policies, which combine the strengths of both time-based and event-based approaches to create a more adaptive and comprehensive defense mechanism (CHO et al., 2020). A key challenge lies in assessing the impact of these different activation policies on both system security and operational performance.

Among the related works surveyed, the one most closely aligned with this dissertation is the study by Nguyen et al. (NGUYEN et al., 2023), which conducted an exhaustive evaluation of MTD strategies within Software Defined Networking (SDN) environments, employing stochastic reward nets (SRN) to analyze their impact. That study provided a comprehensive examination of time-based MTD strategies, particularly focusing on IP shuffling techniques and their effects on system availability and performance metrics. Their analysis explored various switch-over strategies, measuring their influence on performability aspects such as availability, service throughput, response times, server utilization, job loss, and operational costs. The results demonstrated that while increasing the frequency of MTD executions significantly enhances security, it does so at the cost of system performance, especially in policies that terminate jobs during MTD implementation. Conversely, strategies that delay the switch-over until job completion showed a less detrimental impact on performance, highlighting a critical trade-off between security and system efficiency. The foundation of this work served as a basis for the approach proposed in this dissertation, which extends the analysis by employing Deterministic and Stochastic Petri Nets (DSPNs) to model and evaluate additional MTD strategies.

One potent analytical tool is the use of DSPNs. DSPNs extend the traditional Petri net by incorporating stochastic elements, such as timed transitions, deterministic delays, and probabilistic branching, making them ideal for modeling the random, time-dependent behaviors of complex systems. These models are beneficial for analyzing performance metrics such as throughput, resource utilization, and system reliability, as they provide a detailed representation of both the operational dynamics and the timing uncertainties inherent in processes like network protocols or cloud computing.(REQUENO; MERSEGUER; BERNARDI, 2017; BRITO et al., 2021; CARVALHO et al., 2020; FÉ et al., 2022;

SILVA; FÉ; GONÇALVES, 2021; GORRY; IRELAND; KING, 2007).

DSPNs generate a reachability set of possible system states, where probabilistic transitions between states can be evaluated to predict the system's behavior under various scenarios. Furthermore, because DSPNs can be mapped directly to Markov processes, they enable the application of numerical methods for detailed performance evaluations. This feature allows researchers to derive quantitative performance measures, optimize system parameters, and anticipate changes in system behavior under varying conditions, making SPNs a powerful tool in the design and analysis of MTD strategies.

The research of (TORQUATO; MACIEL; VIEIRA, 2021) introduced a DSPN model to evaluate the effectiveness of virtual machine (VM) migration as an MTD tactic specifically targeting insider threats in cloud computing environments. Their study emphasized the balance between enhancing security and preserving system availability. The findings revealed that adopting VM migration within larger system architectures substantially reduces the likelihood of successful insider attacks. However, this comes at the expense of increased downtime, as frequent migrations introduce additional delays. These results underscore the importance of carefully planning VM migration schedules to achieve a balance between improved security and system availability.

1.1 Problem

The implementation of MTD requires the precise definition of activation policies, which are essentially rules that dictate when these dynamic defensive techniques should be deployed. These policies serve as a framework to optimize the application of MTD strategies, ensuring their efficiency while minimizing potential drawbacks such as performance degradation or unnecessary resource usage (CHO et al., 2020).

Integrating IDS with MTD can further enhance the effectiveness of activation policies. In this approach, the IDS operates in conjunction with time-based and hybrid MTD strategies, providing real-time threat detection that can trigger additional migrations when a genuine risk is identified, thereby complementing the proactive nature of MTD. This integration not only improves the system's responsiveness to threats but also optimizes the allocation of computational resources, reducing unnecessary system overhead and minimizing downtime. However, a critical **research problem** arises in the complexity of planning these strategies, particularly when considering the trade-offs between security, system availability, and performance. Factors such as potential downtime during MTD activations or the impact on critical workflows must be carefully evaluated. Previous studies highlighted that increasing the frequency of MTD executions significantly enhances security, albeit at the expense of system performance (NGUYEN et al., 2022), emphasizing the necessity for carefully planning VM migration schedules to balance improved security

and system availability (TORQUATO; MACIEL; VIEIRA, 2021). Furthermore, analyzing how migration techniques affect task completion times is crucial in environments under adversarial actions (CHANG et al., 2020). Analytical models, therefore, become powerful tools, allowing researchers to simulate and assess various strategies under controlled conditions without risking live systems.

Although a migration-based MTD policy is well established, errors can still occur in the detection of intruders, particularly due to limitations in the accuracy of the monitoring system. False positives and false negatives can compromise the effectiveness of the strategy, resulting in unnecessary migrations or the persistence of undetected threats in the system. Therefore, considering other MTD approaches may be essential to enhance the resilience of the defense.

The use of decoy servers emerges as a viable alternative to complement security in scenarios where the effectiveness of detection relies on variable probabilities. By misleading attackers and diverting their attacks to fake targets, decoy servers increase uncertainty and hinder the progression of the attack, reducing the exclusive dependence on the accuracy of the intrusion detection system.

Thus, the central question of this study is: **How to apply different MTD strategies based on migration techniques and decoy servers to optimize the balance between security and probability of attack success?** By leveraging the strengths of DSPNs, IDS, and decoy servers, this research aims to provide actionable insights into designing MTD approaches that effectively harmonize security and effectiveness.

1.2 Objectives

To develop and evaluate hybrid MTD strategies in cloud environments, combining VM migration, IP shuffling, and decoy servers, with the goal of improving security while minimizing performance and availability trade-offs.. The specific objectives of this study are described:

- **Proposal of a Petri net model to represent the event-based MTD policy:** Introducing a model based on an event-driven approach to enhance intrusion detection and threat identification in cloud security. Building on this foundation, the model is then extended to integrate time-based mechanisms, creating a hybrid approach that dynamically adjusts defenses in response to intrusion probabilities and evolving threats, ensuring a more adaptive and proactive security stance.

- **Comparison of MTD migration policies:** To evaluate strategies for VM migration in MTD, analyze time-based, event-based, and hybrid policies. Time-based policies

ensure consistent actions, event-based responses to threats, and a hybrid approach combines both for balanced defense. The goal is to optimize MTD for better security and resilience in cloud computing.

- **Evaluation of MTD focused on IP Shuffling:** This objective broadens the scope of the research by exploring a variation of the previously developed MTD model, focusing on IP Shuffling, which rotates IP addresses to disrupt attacks and obscure system configuration. It evaluates effectiveness against other MTD strategies, analyzing the probability of attack success and operational impact to optimize practical implementation.

1.3 Methodology

The methodology employed in this study follows a step-by-step approach to develop models for evaluating the time to absorption and the probability of attack success in an MTD architecture. As summarized in Figure 1, the research process consists of sequential stages designed to ensure accurate and reliable results. The proposed methodology was applied to generate both performance and availability models, encompassing data collection, model formulation, and analytical evaluation. Each stage builds upon the previous one, incorporating key assumptions and techniques to refine the models.

- **Application Understanding:** Understanding how the application operates is fundamental, and it is necessary to establish the number and type of components involved, in addition to carrying out a survey of the literature about the context to obtain a better understanding.

- **Metrics Definition:** It is important to identify which metrics are relevant, keeping in mind the information that the model can offer to evaluate both the performance and availability of the system. In this work, the metrics selected were: mean time to absorption; probability of attack success; and the rate of migrations per machine, which will be important to evaluate the availability aspect.

- **Parameter Definition:** The parameters that will be inserted into the model are defined. These parameters define the behavior and resource capabilities for the components. In this work, the parameters selected were: the activation time of components and the probability of some components being activated.

- **Analytical Model Generation:** Models are developed at this stage. Each model is built considering the metrics and parameters that were defined in the previous phases. Resulting in an absorption model. The choice of Stochastic Petri Nets occurs because

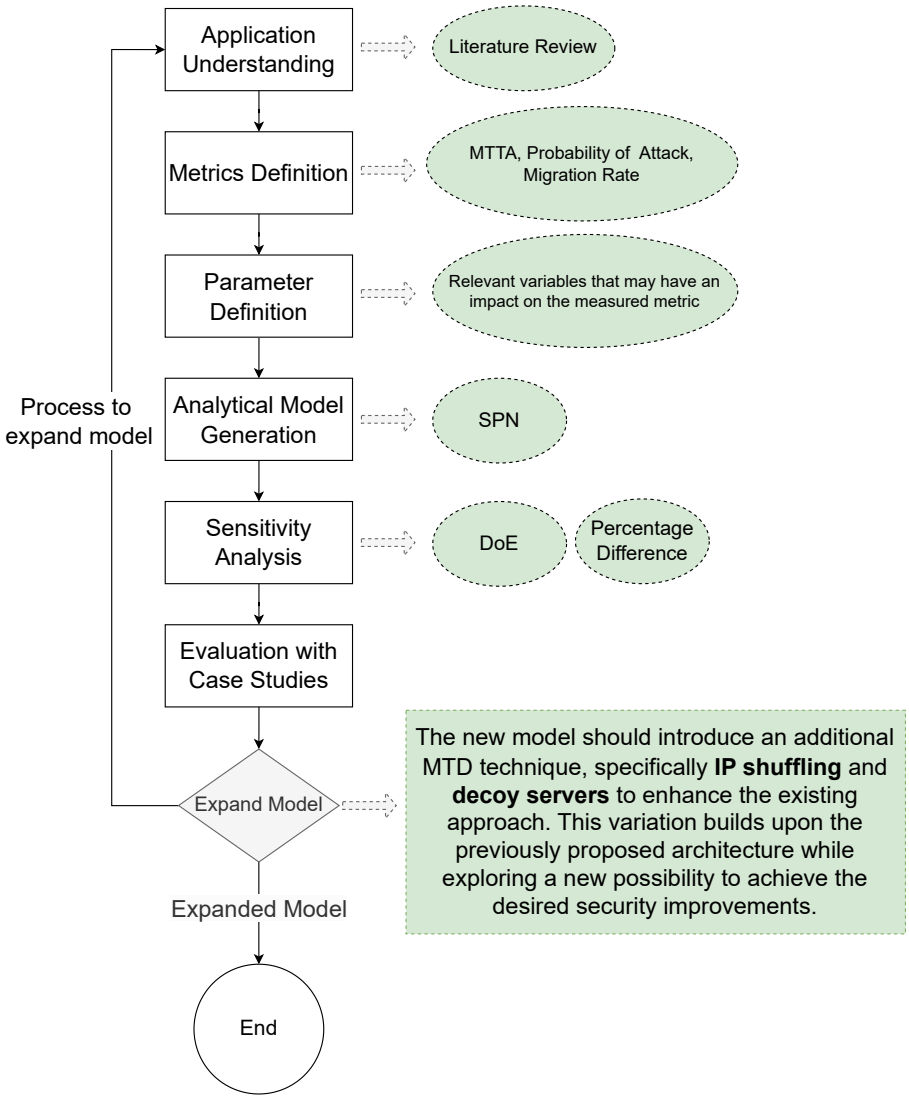


Figure 1 – Work development methodology.

they offer the necessary and sufficient tooling to describe the various behaviors and specifications of the evaluated system.

- **Sensitivity Analysis:** Using sensitivity analysis techniques allows the system designer to understand which components are most critical. The analyses used here were the DoE and the percentage difference analysis. The percentage difference was used in the context of availability, and the DoE was used in the context of performance.
- **Evaluation with Case Studies** Case studies were created to verify the models' ability to generate results. The case studies were designed to also serve as a practical guide on how the user of the model can use it in various analyses.
- **Expand Model:** For the expanding process, the new model should introduce a new

MTD technique that builds on the previous model but at the same time represents a variation of the initially proposed architecture. In this way, the expansion not only maintains coherence with the foundations of the original model but also contributes to the improvement of the strategy, offering a viable and potentially more efficient alternative.

1.4 Work Structure

The remainder of this dissertation is organized as follows: Chapter 2 presents concepts necessary to understand this dissertation; Chapter 4 presents a study based on the methodology with multiple migration policies. Chapter 5 describes the MTD study using multiple decoy servers and IP address migration; Finally, Chapter 6 presents the conclusion of this dissertation.

2 Background

This chapter introduces fundamental notions necessary for comprehending this study. The purpose of this chapter is to provide the reader with an understanding of the essential concepts and methodologies that underpin the study. It serves as a foundation for the subsequent sections, ensuring that all terms and approaches used in the analysis are clear and well-defined. The chapter addresses theoretical frameworks and mathematical tools that are crucial for evaluating the model proposed in this work.

The chapter begins by exploring the **MTD** strategy and its applicability in cybersecurity. Furthermore, **DSPNs** are explained in their structure, stochastic elements, and key components. It presents equations for analyzing performance metrics like throughput and reliability, detailing their derivation and application to assess the proposed model's effectiveness. The chapter also discusses principles of the **Design of Experiments (DoE)**, covering experimental designs and explaining statistical analysis methods.

2.1 Moving Target Defense

The Moving Target Defense is a cybersecurity strategy that enhances system resilience by introducing unpredictability and continuous changes to system configurations (TAN et al., 2023). The core idea behind MTD is to **disrupt an attacker's progress** by dynamically altering the system's attack surface, making it significantly harder for adversaries to exploit vulnerabilities. MTD techniques can be applied in various ways (TORQUATO; VIEIRA, 2020), including:

- **Migration:** Moving processes, services, or VMs across different execution environments to prevent attackers from establishing persistent access.
- **IP Shuffling:** Dynamically changing the IP addresses of network components, such as servers or routers, to prevent attackers from reliably targeting them.
- **Deception:** Deploying decoy systems or providing misleading information to misguide attackers and divert their attention from critical assets.

To illustrate how MTD works, consider a **cloud-based web application** that relies on VMs for hosting services. Suppose an attacker is attempting to exploit a known vulnerability in the system:

1. **Without MTD:** The attacker scans the network, identifies the server's IP address, and targets the vulnerability. Since the server remains in the exact location with the same configuration, the attacker has enough time to craft and execute an attack.

2. **With MTD:** The system periodically *migrates* the application to a different VM instance and dynamically *changes its IP address*. By the time the attacker gathers information and prepares an attack, the target has already shifted, rendering their efforts useless.

To deploy MTD effectively, a central controller can be used to manage system variability and adaptability. This controller is responsible for:

1. **Defining Rotation Rules:** Establishing parameters such as update frequency and resources to be rotated (e.g., IPs, ports, virtual machines).

2. **Automating Modifications:** Using orchestration platforms, preconfigured scripts, or software-defined networking (SDN) to implement changes dynamically.

3. **Ensuring Redundancy:** Maintaining system stability by replicating services across multiple locations, ensuring uninterrupted operation during transitions.

4. **Monitoring and Adjusting:** Continuously observing the environment to optimize MTD strategies based on threat intelligence and system performance.

A key aspect of MTD is determining when the system should apply changes. The three main activation policies (CHO et al., 2020) include:

- **Time-Based Activation:** Changes occur at fixed time intervals, regardless of system conditions.

- **Event-Based Activation:** Modifications are triggered in response to specific events, such as an ongoing attack.

- **Hybrid Activation:** A combination of time-based and event-based approaches, allowing for greater flexibility and adaptability.

2.2 Deterministic and Stochastic Petri Nets

Petri Nets (PNs) are mathematical and graphical tools designed to model systems characterized by asynchrony, simultaneity, distribution, parallelism, stochastic processes, and/or non-determinism (CHEN; HA, 2018). Their graphical and formal nature provides a clear means of representing system dynamics, facilitating the analysis of interactions

between components over time. A PN can be viewed as a directed bipartite graph composed of three primary elements (RODRIGUES et al., 2020): **places**, **transitions**, and **arcs**.

- **Places** (circles) represent conditions or states of the system.
- **Transitions** (rectangles) denote events or activities that may alter the system's state.
- **Arcs** connect places and transitions, defining the direction of token flow.

Tokens located in places represent the system's current marking or state. When a transition fires, tokens are consumed from its input places and produced in its output places, thereby updating the system configuration. Special elements, such as inhibitor arcs, can regulate transition firing by preventing or allowing specific actions according to predefined conditions. The ability to graphically represent such interactions makes Petri Nets valuable for the structural and behavioral analysis of concurrent and distributed systems.

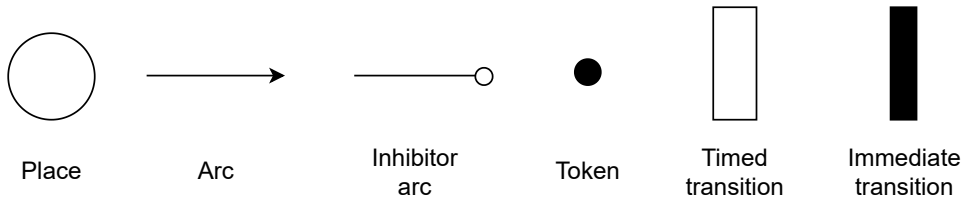


Figure 2 – Main components of a DSPN.

Over time, Petri Nets have been extended to incorporate additional features such as timing and probabilistic behavior, which are essential for modeling real-world systems where randomness and delay are present. A prominent extension is the SPN, which associates firing delays of transitions with probability distributions (MARSAN; CHIOLA, 1986; GERMAN, 2000; MARSAN et al., 1994; MACIEL, 2023b; MACIEL, 2023a). This stochastic nature enables the representation of systems in which events occur unpredictably, allowing for performance and dependability evaluation through formal analysis.

A further development of SPNs is the **DSPN**, which extends the modeling framework by allowing both stochastic and deterministic firing times within the same network. In DSPNs, transitions can be classified into two main types (MURATA, 1989):

- **Timed transitions**, whose firing times follow deterministic or stochastic probability distributions.
- **Immediate transitions**, which fire instantaneously once enabled, representing instantaneous events.

Immediate transitions can be associated with guard functions — Boolean expressions that determine whether a transition may fire based on the system state — and can be assigned *priorities* to control the execution order when multiple transitions are enabled simultaneously (MARSAN et al., 1998). These features provide flexibility in representing complex systems with mixed timing behaviors and conditional logic.

DSPNs combine formal correctness, graphical expressiveness, and quantitative analysis within a single framework (SILVA et al., 2018). This integration allows both qualitative verification such as reachability or deadlock analysis and quantitative performance evaluation such as throughput, utilization, and response time. They also benefit from a wide range of dedicated software tools that automate modeling, simulation, and analysis tasks (GIRAULT; VALK, 2013).

The motivation for adopting DSPNs lies in their ability to overcome the limitations of alternative modeling approaches. Traditional Markov models offer strong mathematical rigor but lack the expressiveness required to represent concurrency and event-driven interactions. Similarly, queuing models are effective for performance evaluation but cannot easily capture complex dependencies or structural relationships among system components. By combining stochastic timing, deterministic control, and structural modeling within a unified formalism, DSPNs provide a comprehensive and flexible approach for analyzing distributed and concurrent systems, making them suitable for studies involving reliability, availability, and performance optimization.

2.3 Sensitivity Analysis

Sensitivity analysis evaluates the influence of specific input data on output data to identify potential vulnerabilities in computer systems. It is a crucial technique for understanding how variations in input parameters or environmental factors affect system performance. By examining the relationship between inputs and outputs, sensitivity analysis helps highlight areas of the system that may be particularly sensitive to changes, which could lead to vulnerabilities or inefficiencies. This allows designers and administrators to focus their efforts on improving these areas to enhance overall system resilience.

Following this evaluation, various techniques are applied to enhance system performance in different scenarios (CAMPOLONGO; TARANTOLA; SALTELLI, 1999). Sensitivity analysis is not just about identifying vulnerabilities; it is also a tool to optimize system performance. Once the critical factors affecting the system are identified, designers can use different techniques to adjust these factors and improve system efficiency under various conditions. This proactive approach ensures that the system is robust and adaptable to a range of potential challenges.

System designers frequently utilize sensitivity analysis to determine how a particular

metric responds to changes in the model (SANTOS et al., 2021). Sensitivity analysis allows system designers to test how variations in input parameters (such as resource allocation, load, or configuration) will affect specific metrics or outputs (such as performance, response time, or security). This process is beneficial when designing complex systems where multiple interacting components are at play, as it helps in predicting how the system will behave under different configurations.

There are several approaches to conducting sensitivity analysis, including regression analysis, perturbation analysis, one-by-one variation, Monte Carlo simulation, parametric differential analysis, correlation analysis, and percentage difference. Each of these methods has its strengths and weaknesses, depending on the context in which they are applied. For instance:

- **Regression analysis** involves modeling the relationship between input variables and outputs, allowing designers to quantify the sensitivity of outputs to input changes.
- **Perturbation analysis** involves systematically changing input values and observing the resulting changes in outputs, often used in scenarios where inputs are uncertain or vary over time.
- **One-by-one variation** tests the impact of changing one input at a time while holding others constant, a simple and effective method for isolating individual variables.
- **Monte Carlo simulation** employs random sampling to simulate a wide range of possible input combinations and assess how variability in inputs affects outputs, commonly used when the input space is large or complex.
- **Parametric differential analysis** examines how small changes in parameters affect system behavior, providing insights into the sensitivity of a model to specific parameters.
- **Correlation analysis** explores the relationship between input variables and outputs to determine the strength and direction of their interdependence.
- **Percentage difference** is a straightforward approach that compares the relative difference between outputs as a percentage of the initial value, often used to quantify the impact of specific changes on system performance.

Choosing the appropriate method can be challenging, as it requires assessing the available computational resources and the characteristics of the problems being addressed (CAMPOLONGO et al., 2004; PIANOSI et al., 2016). Some methods, like Monte Carlo simulations, may require significant computational resources. In contrast, others, such as

regression analysis, can be less resource-intensive but might not be suitable for all types of problems. The decision about which method to use depends on the specific objectives of the analysis, the complexity of the system being modeled, and the available computational power.

Sensitivity analysis, in a way, provides essential security and helps guide results within the framework established by system administrators. By identifying which variables most influence the output, system administrators can focus on securing those aspects of the system that are most vulnerable or prone to error. This targeted approach ensures that resources are allocated effectively to mitigate risks and enhance system security.

In this study, sensitivity analysis is performed using the **DoE** and **Percentage Difference** methods. **DoE** is used to systematically explore the impact of multiple factors on the outcomes, enabling a comprehensive understanding of how different parameters interact and contribute to the overall system performance. The **Percentage Difference** method, on the other hand, provides a straightforward way to quantify the effect of individual changes on system performance, making it easier to compare the impacts of different variables. By combining these methods, the study aims to provide a robust framework for assessing system performance and identifying critical factors for improvement.

2.3.1 Design of Experiments

The DoE encompasses a set of statistical methods aimed at enhancing the understanding of the product or process being analyzed (KLEIJNEN, 1995). DoE provides a structured framework for planning, conducting, and analyzing experiments to explore relationships between factors and outcomes. By systematically manipulating input variables (factors) and observing the corresponding outputs, DoE allows researchers to identify significant factors, their interactions, and optimal conditions for performance improvement.

DoE is a robust tool used to investigate new processes, deepen knowledge of existing ones, and subsequently optimize them to achieve exceptional performance standards (ANTONY, 2014). In fields ranging from manufacturing to software development, DoE is widely used to improve system design, troubleshoot performance issues, and enhance product quality. By focusing on controlled experimentation, DoE ensures that conclusions drawn from experiments are valid and reliable, helping to avoid potential biases that could lead to inaccurate assessments.

In the specialized literature (FEITOSA et al., 2021; COSTA et al., 2016; SANTOS et al., 2021), several types of graphs commonly appear in experiments utilizing the DoE approach. These visual aids help to summarize and interpret the results of the experiments effectively, making it easier to communicate findings and derive insights. Among these graphs, some are specifically designed to highlight the relationships between factors and

their effects on the outcome of the experiment.

One such graph is the **factor effect graph**, which uses descending bars to illustrate the relative influence of each factor. The graph presents the factors on the horizontal axis and the magnitude of their impact on the outcome on the vertical axis. Each factor is represented by a bar, and the height of the bar indicates the strength of the effect. The taller the bar, the greater its impact on the output, offering a clear representation of how each factor contributes to the outcome. This visual format makes it easy to identify which factors have the most influence and which have less, aiding in decision-making for process optimization.

For instance, **Figure 3** depicts a factor effect graph with three factors: A, B, and C. Among them, factor C demonstrates the most significant impact, indicated by the tallest bar. This means that factor C has the most significant effect on the outcome of the experiment, and adjustments to it would likely yield the most substantial improvements in performance. The other factors, A and B, have a negligible influence, and modifying them may lead to less dramatic changes in the results.

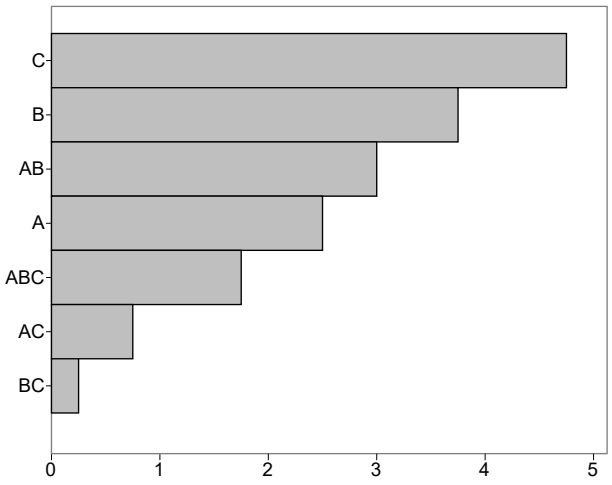


Figure 3 – Generic example of factor effect graph.

The **factor effect graph** helps determine which factor interactions exert the most significant influence on the optimization process or study design, highlighting where efforts should be concentrated. By analyzing the magnitude of interactions between factors, it becomes possible to identify the optimal combination of measures and recognize patterns of cumulative or detrimental effects between factors. This process is crucial for refining experimental designs and focusing resources on the most impactful variables.

The factor effect graph is particularly valuable in experiments with multiple factors, as it provides a clear visual representation of how each factor and its interactions contribute to the overall outcome. When interpreting the graph, researchers can identify not only the most influential factors but also how combinations of factors may enhance or interfere with

each other's effects. For instance, some factors may have a synergistic relationship, where their combined effect is greater than the sum of their individual effects. In contrast, others may interact in a way that diminishes their impact, leading to a detrimental overall effect.

The interaction between factors A and B is calculated using equation (2.1), where $E_{A,B(+1)}$ represents the effect of factor A at the high level of factor B, and $E_{A,B(-1)}$ indicates the effect of factor A at the low level of factor B. This equation captures how the outcome of factor A changes depending on the level of factor B, helping to quantify the strength and direction of their interaction. By calculating these effects for all factor combinations, researchers can construct a comprehensive model of the system's behavior and optimize the design accordingly.

$$I_{A,B} = \frac{1}{2}(E_{A,B(+1)} - E_{A,B(-1)}) \tag{2.1}$$

Interaction graphs are used to detect interactions between factors in an experiment. An interaction occurs when the effect of one factor on the outcome is modified—either amplified or diminished—by changes in the levels of another factor. Understanding these interactions is critical because they reveal how different factors work together, potentially enhancing or mitigating each other's effects.

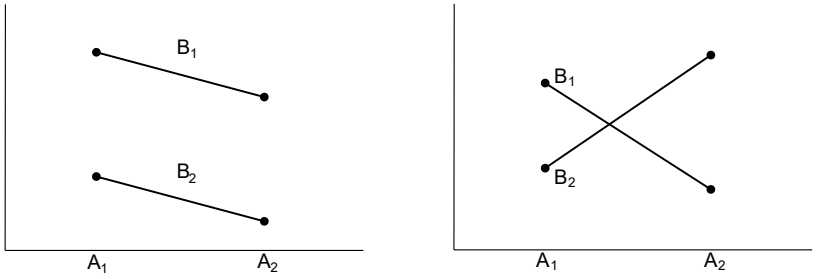
In an interaction graph, the relationship between factors is visually represented, typically with lines. Parallel lines on the graph indicate the absence of interaction between the factors. This suggests that the effect of one factor is consistent across the different levels of the other factor, implying that the two factors do not influence each other's impact on the outcome.

On the other hand, non-parallel lines in the graph suggest a significant interaction between the factors. This means that the level of the other factor influences the effect of one factor. For example, if the lines cross or diverge, it indicates that the effect of factor A on the outcome is not constant but varies depending on the level of factor B.

For instance, Figure 4a illustrates a scenario where no interaction exists between the factors, as the lines remain parallel. In contrast, Figure 4b demonstrates a case where interaction is present, as the lines intersect. In this specific example, the effect on a particular metric for factor A at level A1 is greater than at level A2, showing that the impact of changes in factor A depends on the levels of factor B. This interaction suggests that a more nuanced approach to optimization, considering the interaction between factors, is necessary for achieving optimal results.

2.3.2 Percentage Difference

The **percentage difference method** was selected to conduct the sensitivity analysis in this study, as it operates without relying on a continuous domain for the input



(a) Interaction graphs with low interaction between factors.

(b) Interaction graphs with high interaction between factors.

Figure 4 – Generic interaction graphs.

parameter values. This is particularly useful in situations where input parameters may not follow a continuous distribution, allowing flexibility in assessing how variations in different parameters impact the system. This method involves calculating the percentage difference when an input parameter is varied from its minimum to maximum value. By comparing the values of the system output at the extremes of the parameter range, this method provides a measure of sensitivity to input variations. To determine parameter sensitivities, it is essential to consider the entire range of possible values for each parameter (HOFFMAN; GARDNER, 1983). This ensures a comprehensive assessment of the parameter's influence on the output, avoiding biased results that could arise from focusing on only a limited subset of values.

$$S_\theta\{Y\} = \frac{max\ \{Y(\theta)\} - min\ \{Y(\theta)\}}{max\ \{Y(\theta)\}} \tag{2.2}$$

Equation 2.2 illustrates how the sensitivity index is calculated using the percentage difference method. The expressions $max\ Y(\theta)$ and $min\ Y(\theta)$ represent, respectively, the maximum and minimum output values measured by varying the θ parameter over its n possible values of interest. These values are used to determine how much the system output changes as the parameter θ is varied across its range. If $Y(\theta)$ is known to vary monotonically (i.e., consistently increasing or decreasing), only the extreme values of θ (i.e., θ_1 and θ_n) are needed to calculate $max\ Y(\theta)$ and $min\ Y(\theta)$. In this case, the sensitivity index S_θY can be determined more easily using just these extreme values, simplifying the calculation and interpretation of the sensitivity analysis.

3 Related Works

This chapter reviews previous studies that have utilized **stochastic models** within the context of **MTD**. Research that did not incorporate stochastic models, despite being set within the MTD framework. Table 1 compares the related work using seven critical aspects: MTD strategy, metrics, Attacker Knowledge Accumulation and Progression, IDS, Two-Layer Security, and Decoy Servers.

Table 1 – Related works

Work	**MTD Strategy**	**Metrics**	**Attacker Knowledge Accumulation and Progression**	**IDS**	**Two-Layer Security**	**Decoy Servers**
(BABADI; DOUSTMOHAMMADI, 2022)	Attack Deception	Migration Rate	×	×	×	×
(MENDONÇA et al., 2023)	IP shuffling	Response Time, Throughput, Utilization, Job Completion	×	×	×	×
(TORQUATO; MACIEL; VIEIRA, 2021)	Migration-Based	MTTA	✓	×	×	✓
(CHANG et al., 2020)	MTTA, Availability	MTTA	×	×	×	×
(MENDONÇA et al., 2022)	IP Shuffling	Probability of attack success	×	×	×	×
(CROUSE; PROSSER; FULP, 2015)	Decoy, Address Shuffling	Rate of scans attempted by the attack, efficiency of the number of honeypots	×	×	✓	✓
(WANG et al., 2024)	Decoy, Address Shuffling	Probability of Success, Probability of Failure, Scan Rate	×	×	✓	✓
(HU; ZHU; LIU, 2020)	Dynamic Change of States and Decisions	Aggregate Utility, Transition Probability, Effectiveness of Reinforcement Learning	×	×	×	×
(GE et al., 2020)	Network Topology Shuffling (NTS-MTD)	Number of attack paths, MTTSF, Defense Cost, Package Delivery Rate	×	×	✓	✓
(PAGNOTTA et al., 2023)	Fake Service Randomization	Time to Compromise, Attack Success Rate, Traffic Volume Generated	×	×	✓	✓
(MOODY; HU; APON, 2014)	Decoy	Transition rate between states	×	×	✓	×
This Work	Decoy, Address Shuffling, Migration Based	MTTA, Intrusion Probability, Probability of attack success, Migration Rate	✓	✓	✓	✓

MTD strategy: One of the key criteria for comparing related works is the **MTD strategy**, which focuses on the specific **Moving Target Defense** approaches adopted by each study. This criterion evaluates the techniques used to enhance system security by dynamically altering the attack surface, making it more challenging for adversaries to exploit vulnerabilities successfully. By analyzing the MTD strategies employed in related works, it is possible to assess the effectiveness and adaptability of the proposed solutions in addressing various attack scenarios. Each strategy offers a unique approach to defending against cyber threats, and understanding their application helps identify their strengths, limitations, and suitability for different types of systems and threat landscapes.

Metrics: The metrics employed in a study are essential for defining its primary objectives and assessing its outcomes. Research focused on evaluating the effectiveness of an architecture often employs metrics such as detection rate and probability of attack success. Some studies go beyond basic evaluation by incorporating advanced metrics like mean time to absorption, which provides better comprehension of the system's resilience and behavior over time.

Intrusion Detection System: The IDS is a crucial criterion for comparing related works, as it evaluates whether the probability of intrusion detection was integrated with the MTD strategy and how effectively this combination enhances security. An IDS plays a vital role in identifying threats and malicious activities, enabling MTD systems to respond dynamically by altering the attack surface. Additionally, the analysis includes the IDS's ability to minimize false positives and negatives, maintain consistent performance across diverse scenarios, and adapt to different activation policies. Metrics such as detection rate, probability of attack success, and system resilience further demonstrate the effectiveness of these integrations.

The study conducted by (BABADI; DOUSTMOHAMMADI, 2022) presented an approach to detecting cyberattacks by integrating MTD strategies with advanced detection mechanisms. The proposed method focuses on creating a defense mechanism designed to mislead adversaries, making their actions more detectable from the perspective of the defender. This approach leverages the core principle of MTD, which is to introduce dynamic and unpredictable changes in the system's configuration, thereby complicating the attacker's task of gathering information and executing a successful attack. By deliberately altering the attack surface and embedding deceptive elements, the method not only reduces the likelihood of a successful breach but also enhances the defender's ability to identify malicious activities in real time. The study highlights the effectiveness of combining deception with detection, demonstrating its potential to improve overall system resilience significantly.

In another contribution, (MENDONÇA et al., 2023) utilized an SPN model to analyze the impact of periodic MTD operations on various system performance metrics.

This research primarily focused on evaluating how frequent and planned changes in the system, as dictated by MTD strategies, influence response time, overall performance degradation, and Quality of Service (QoS). By employing an SPN model, the study was able to capture the inherent stochasticity and dynamic nature of MTD operations, offering a detailed understanding of the trade-offs between enhanced security and system performance. The findings revealed critical insights into how periodic changes could be optimized to minimize negative impacts on performance while maintaining robust security.

The work of (CHANG et al., 2020) explored the interplay between MTD strategies and system task management, specifically analyzing how migration techniques affect task completion times in an environment under attack. Using analytical modeling, this study investigated the dynamics of task execution when the system is subjected to adversarial actions and frequent migrations. The migration process, a core component of many MTD strategies, involves moving tasks or virtual resources from one physical or virtual location to another to disrupt potential attacks. The research highlighted the challenges and benefits of this approach, illustrating how migrations can serve as both a defensive measure and a factor that influences task execution times. By providing a detailed analysis of these dynamics, the study offers valuable insights into optimizing migration techniques to balance security and operational efficiency.

Focusing specifically on the role of migration as a defensive tactic, (TORQUATO; MACIEL; VIEIRA, 2021) developed a Petri net model to evaluate security within an MTD framework tailored for VM migration. This study aimed to assess the effectiveness of VM migration in reducing the likelihood of a successful attack by dynamically altering the system's configuration. The model incorporated up to four physical machines, enabling a detailed analysis of how system reconfiguration impacts security metrics. One of the key aspects examined was the probability of attack success under varying migration scenarios. Additionally, the study considered the potential impact of system downtime caused by VM migrations, offering a balanced view of the trade-offs between increased security and temporary interruptions in system availability. The findings underscored the importance of optimizing migration strategies to minimize downtime while maximizing the security benefits of MTD.

(CROUSE; PROSSER; FULP, 2015)'s work analyzes the effectiveness of two MTD strategies: network address shuffling, which consists of periodically remapping network addresses, making reconnaissance information useless to the attacker; and the combination with honeypots: Honeypots are used to deceive attackers, directing them to fake systems and collecting information about their actions. (WANG et al., 2024) implemented two main defense strategies: address mutation, which periodically changes the IP addresses of network edge nodes, making it difficult for attackers to track and attack; fingerprint decoy, which deploys honeypots with fake fingerprints to deceive attackers and reduce the

effectiveness of reconnaissance. The work of (HU; ZHU; LIU, 2020) focuses on adaptive defense against multistage attacks, that is, the network is protected dynamically and continuously, making the attack more difficult by changing the system conditions based on the attacker's behavior. The MTD strategy does not rely on a fixed defense, but continuously adapts the network conditions to make the attacker's job more difficult and reduce the chances of attack success.

The work proposed by (GE et al., 2020) presents a proactive defense for IoT networks. The MTD strategy used is Network Topology Shuffling (NTS-MTD), which consists of dynamically changing the topology of the IoT network to increase the complexity of attacks. In addition, the work combines decoy systems (fake nodes) to divert attackers from real targets and collect information about their behavior. The evaluation carried out by (MOODY; HU; APON, 2014) uses a deceptive defense (Deceptive Defense) as an MTD strategy. This tactic involves using decoy nodes that mimic operational nodes to lure and detect attackers, thereby increasing the environment's complexity. The nodes act as honeypots, helping to protect the actual operational nodes by increasing the difficulty of identifying legitimate targets. In this context, Moody's work uses Stochastic Petri Nets to model the defensive maneuver platform.

Finally, (MENDONÇA et al., 2022) extended the use of SPN models to evaluate security metrics in networks that integrate software-defined networking (SDN) with MTD strategies. This research compared the performance and security of systems with and without MTD, emphasizing the advantages of a Time-based MTD approach. By periodically introducing changes to the network's configuration, the Time-based strategy disrupts an attacker's ability to establish a foothold or maintain access to the system. The study demonstrated that this approach significantly impedes an attacker's progress, thereby reducing the probability of a successful breach. The findings highlighted the superior performance of systems employing MTD strategies in terms of thwarting attacks, while also shedding light on potential areas for optimization to enhance both security and system performance.

In summary, these studies collectively contribute to the body of knowledge on MTD strategies by exploring various dimensions, including deception, migration, performance trade-offs, and the integration of advanced models such as SPNs and Petri nets. Each work emphasizes the critical role of dynamic and adaptive defenses in mitigating threats and provides valuable insights into the practical implementation and optimization of MTD approaches in different contexts. While the reviewed studies have explored different approaches to MTD, none of them integrate an IDS with a hybrid time-and-event-based strategy and a decoy approach. Our approach is unique in combining these key aspects:

1. **Integration with an IDS**: Unlike previous works that evaluate MTD without a

dedicated intrusion detection system, our model employs an IDS to identify suspicious activities and dynamically adjust the defensive strategy.

2. **Event-Based Activation**: While many studies rely solely on time-based activation, our approach incorporates event-based activation, enabling a more responsive and context-aware defense mechanism.

3. **Hybrid Strategy**: Our model combines both time-based and event-based activation, optimizing defense efficiency by balancing proactive and reactive responses.

4. **Decoy approach**: An SPN model that incorporates decoy-based architectures and evaluates their effectiveness in reducing the probability of an attacker completing an attack.

4 Migration Analysis

This chapter presents a study based on the methodology of multiple migration policies, including time-based, event-based, and hybrid approaches. The objective is to evaluate how each policy impacts system security, analyzing its effectiveness in mitigating attacks and increasing system resilience. The hybrid approach, which combines both time and event-based triggers, is particularly examined for its potential to balance security and operational efficiency. Part of the content presented in this chapter is based on a previously published article by the author.

4.1 Environment

This section offers a comprehensive overview of the environment presented in the study by (TORQUATO; MACIEL; VIEIRA, 2021), as well as an extension introduced in this research, which further refines the understanding of the attacker's process during an intrusion attempt and elaborates on the model's approach to managing migration. VM migration involves transferring the VM's entire state, including all data, configuration, and execution context, from the current physical host to a designated new host machine. This process is executed in a manner that aims to disorient and delay potential attackers, as the target environment for exploitation is constantly changing, thus making it harder for adversaries to sustain their attacks.

It is essential to recognize that using VM migration as an MTD strategy has limitations, and its effectiveness is partial at best. The migration process does not inherently solve all security issues. Specifically, attacks directed at the VM itself are not nullified by migration. This is because the VM retains its entire state and data during the migration process, meaning that any attack that compromises the VM will continue to persist after it has been moved to a new physical host. For instance, if a malicious actor successfully exploits vulnerabilities within the VM's software or configuration, this attack will follow the VM to its new location, as the system state is unchanged post-migration.

On the other hand, attacks that are aimed at the physical host machine or infrastructure, such as hardware-based attacks or attacks exploiting vulnerabilities in the host OS, are effectively mitigated. This is because, during migration, the VM is moved to a completely different physical host, thus severing any ongoing exploitations targeting the original host machine. In this regard, VM migration provides a tactical defense mechanism by interrupting adversarial activities that rely on host-level vulnerabilities, such as

firmware-level malware or hypervisor escape exploits. However, it does not address attacks targeting the VM itself.

The extension introduced in this research aims to bridge this gap by considering additional factors and improving the way migration is managed during an attack. Specifically, the model attempts to account for both the persistence of VM-targeted attacks and the mitigation of host-targeted attacks. This nuanced understanding of how migration interacts with various attack vectors allows for more effective MTD strategies, as the model now adapts to a broader range of threat scenarios, taking into account the specific characteristics of each attack and tailoring the defense mechanism accordingly.

Thus, while VM migration is a valuable tool within an MTD strategy, it is not a comprehensive solution in isolation. The study emphasizes the importance of combining migration with other defense mechanisms, such as IDS, which are capable of detecting and mitigating attacks targeting the VM directly. By integrating such systems with the migration process, the effectiveness of MTD can be significantly enhanced, providing a more robust defense against a broader spectrum of cyber threats.

Figure 5 illustrates the three stages of an attack processed by the system: Reconnaissance, Attack in Progress, and Successful Attack. **Reconnaissance**: In this initial phase, the attacker begins by infiltrating the system and performing a detailed analysis of the environment. The goal is to identify potential vulnerabilities and weaknesses that can be exploited later. This stage involves gathering information about the system's architecture, configurations, and access points, which can include network topologies, exposed services, and other critical components. The attacker aims to develop a strategy for the upcoming attack phase based on the data collected during reconnaissance. The reconnaissance phase is usually stealthy, and the attacker avoids triggering alarms to remain undetected by security mechanisms like IDS.

Attack in Progress: Once the attacker has collected sufficient information, they transition into the Attack in Progress phase. This stage is characterized by active attempts to breach the system's defenses. The attacker tries to exploit the vulnerabilities identified during reconnaissance, launching specific exploits such as phishing, malware delivery, or denial-of-service attacks. The attacker often gains partial access to the system at this point, further analyzing the environment to escalate privileges and broaden their control. The attacker's actions might remain undetected if the system's defenses are not adequately monitoring suspicious activities or the system lacks effective MTD strategies. During this phase, the MTD approach is vital in making the attacker's efforts more difficult and unpredictable, as it dynamically changes the system's attack surface.

Successful Attack: The final phase is marked by a breach in the system's defenses, which allows the attacker to achieve their ultimate objective—complete control over the system. In this phase, the attacker might gain administrator-level access, allowing them to

manipulate data, exfiltrate sensitive information, or disrupt the normal operations of the system. The system has effectively been compromised, resulting in the failure of the security mechanisms in place, including those meant to thwart the attack (e.g., through MTD, IDS). This stage represents a significant security breach, as the attacker has overcome the system's defenses and achieved their malicious goal.

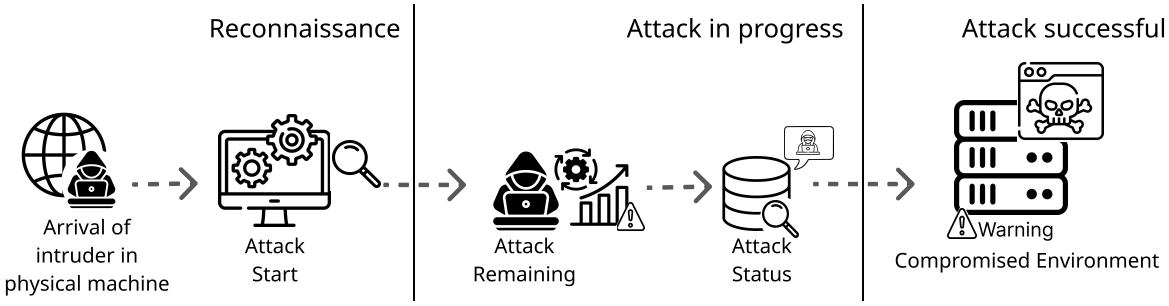


Figure 5 – Intrusion stages

Figure 6 illustrates the process of migrating a system environment from Machine 01 to Machine 02, with a particular focus on how this process impacts both the system and any potential intruder. The migration unfolds over three distinct stages, each playing a crucial role in disrupting the attacker's actions and altering the operational environment.

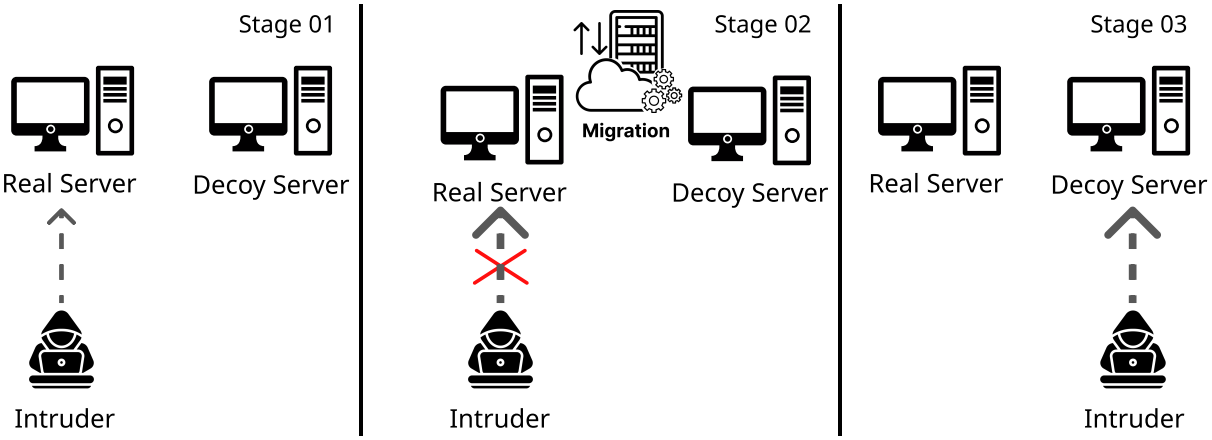


Figure 6 – Migration stages

- Stage 1: Attack on Machine 01: At this stage, the attacker is actively engaged in an attack on the environment hosted by Machine 01. They are systematically identifying and exploiting vulnerabilities within the system. This phase is crucial for the attacker as they gather information, escalate privileges, and attempt to compromise the system. The attacker's primary goal is to breach the defenses and gain control over the environment. During this stage, the system's defenses may be actively engaged in

detecting the intrusion, depending on the robustness of the MTD and IDS strategies employed.

- Stage 2: Migration Process: Once the migration process begins, there is an immediate disruption in the attacker's activities. The migration involves transferring both the operating system and the attacker's current session to Machine 02. This transfer interrupts the ongoing attack by effectively resetting the system's operational context. The attacker is forced to halt their actions because their direct access to the system on Machine 01 is interrupted. From a defensive standpoint, this disruption creates a significant setback for the attacker, as their current attack vector is no longer applicable in the new environment. The migration essentially forces the attacker to start over, and the system benefits from the temporary protection provided by the shift.

- Stage 3: Attack on Machine 02: Once the migration is complete, the attacker is now located on Machine 02. However, this new environment differs from the previous one, requiring the attacker to restart the process of exploring the system to find vulnerabilities. They have to repeat the reconnaissance activities they undertook on Machine 01, scanning for weaknesses and gathering information about the new system configuration. This restart significantly slows down the attacker's progress, buying valuable time for the defenders to implement additional security measures or initiate countermeasures. If the attacker later returns to Machine 01, the progress achieved in the previous round is preserved, allowing them to resume the attack from the point it was interrupted.

Figure 7 presents a detailed flowchart that outlines the sequence of actions within the proposed architecture. This flowchart highlights the integration of an **IDS** into the original model introduced by (TORQUATO; MACIEL; VIEIRA, 2021), with the new additions marked in blue for distinction. The diagram provides a structured and methodical approach to defending against potential attackers, illustrating the various stages and responses that occur during an attack attempt.

The process begins when an attacker gains access to the network of **Machine 01**. At this stage, the attacker initiates a preliminary investigation, aiming to identify any weaknesses or vulnerabilities within the system. This phase is critical for the attacker, as it enables them to gather valuable information about the system's operations and potential points of entry. During this reconnaissance stage, the attacker typically explores network services, system configurations, and open ports to identify exploitable weaknesses.

As the attacker continues their investigation, they gain deeper insights into the system's functionality. This knowledge can eventually enable them to successfully breach the system's defenses, leading to the complete compromise of **Machine 01**. However, if

Figure 7 – FlowChart for the defensive strategy system

the attacker's efforts to penetrate the system are unsuccessful or stalled, the architecture can activate an **MTD** strategy as a countermeasure. The **MTD** strategy, which could involve dynamic actions such as migration, aims to disrupt the attacker's progress by altering the system's configuration or environment. This proactive defense mechanism significantly reduces the chances of the attacker succeeding in their intrusion, ultimately preserving the integrity and security of the system.

The previous work of (TORQUATO; MACIEL; VIEIRA, 2021) implemented migration as a time-based **MTD** strategy. In our approach, migration occurs at regular

intervals, and the data is transferred systematically between machines according to a predefined schedule. The goal of the time-based migration is to alter the system's environment periodically, making it more difficult for an attacker to achieve a successful breach.

Building on this foundation, the current research introduces an extension by incorporating two additional strategies for initiating migration: an event-based approach and a hybrid strategy. The event-based migration strategy integrates with an **IDS** to monitor the system for unusual activities that may indicate a potential intrusion. Upon detecting a security threat, the **IDS** triggers the migration process, relocating the system to another machine and forcing the attacker to restart their reconnaissance and attack attempts from scratch. This strategy is designed to respond dynamically to emerging threats, thereby minimizing the risk of successful attacks.

The hybrid approach combines both time-based and event-based triggers, offering a more flexible and adaptive defense strategy. Under this hybrid model, migration can be activated either at regular intervals (time-based) or in response to detected threats (event-based). This approach provides a balance between proactive, scheduled defense measures and responsive actions based on real-time threat assessments.

In both time-based and event-based scenarios, the IDS supports the MTD strategy by monitoring for malicious activity. In event-based and hybrid approaches, it can directly trigger migrations when a threat is detected. In contrast, in time-based policies, migrations occur proactively at predefined intervals, with the IDS serving primarily to detect ongoing attacks and inform subsequent defensive actions. This prevents unnecessary resource consumption and ensures that migrations are effectively utilized, optimizing the overall defense strategy. By integrating the **MTD** strategy with the **IDS**, the system is able to respond to attacks more efficiently and strategically, minimizing the chances of an attack's success while maximizing the use of available resources.

In conclusion, by offering both time-based and event-based strategies, as well as a hybrid approach, the proposed architecture is better equipped to adapt to a variety of attack scenarios and effectively disrupt the attacker's progress, ultimately safeguarding the system's integrity, aiming to enhance the overall security posture of the system.

4.2 Model Overview

This section presents the **DSPN** model, offering a detailed representation of the environment discussed in Chapter 4.1. It further elucidates the foundational model, performs a sensitivity analysis of the baseline structure, and describes the maintenance models associated with the previously discussed strategies.

4.2.1 MTD Model

This study expands upon the research presented in (TORQUATO; MACIEL; VIEIRA, 2021) by integrating an **IDS** and introducing different policies for executing the migration process. The **DSPN** model represents a scenario involving two physical machines that utilize migration as a defensive mechanism against potential security threats. The primary objective of this model is to assess the effectiveness of **MTD** strategies in safeguarding systems, analyzing how different migration policies can mitigate cyberattacks. Additionally, the model serves as a tool for formulating strategies for system configuration, allowing for a thorough evaluation of the necessity of migrations and the probability of threat detection through the **IDS**.

One of the central aspects addressed in the modeling process is the analysis of the time required for an attacker to compromise a system under different migration policies. The goal is to determine which approach provides greater resilience against attacks by increasing the time needed for a successful breach. By extending the period required for an attacker to gain control over the system, the architecture significantly strengthens the security of the computational environment.

Figure 8 presents the **DSPN** model developed for this purpose, illustrating the structure and operational flows governing the transitions between system states. The model incorporates different mechanisms for decision-making regarding migration execution, whether based on a fixed time interval (*Time-Based*), on events detected by the **IDS** (*Event-Based*), or on a hybrid approach combining both strategies. This enables the system to dynamically respond to various threat scenarios, enhancing its defensive capabilities against cyberattacks.

To ensure a detailed description of the model's components and the rules governing its operation, the key parameters and the logic of the guard expressions that regulate system transitions are presented in Tables 2 and 3. All values were defined based on the original model proposed in (TORQUATO; MACIEL; VIEIRA, 2021), serving as a foundation for the construction and validation of the new approach implemented in this study.

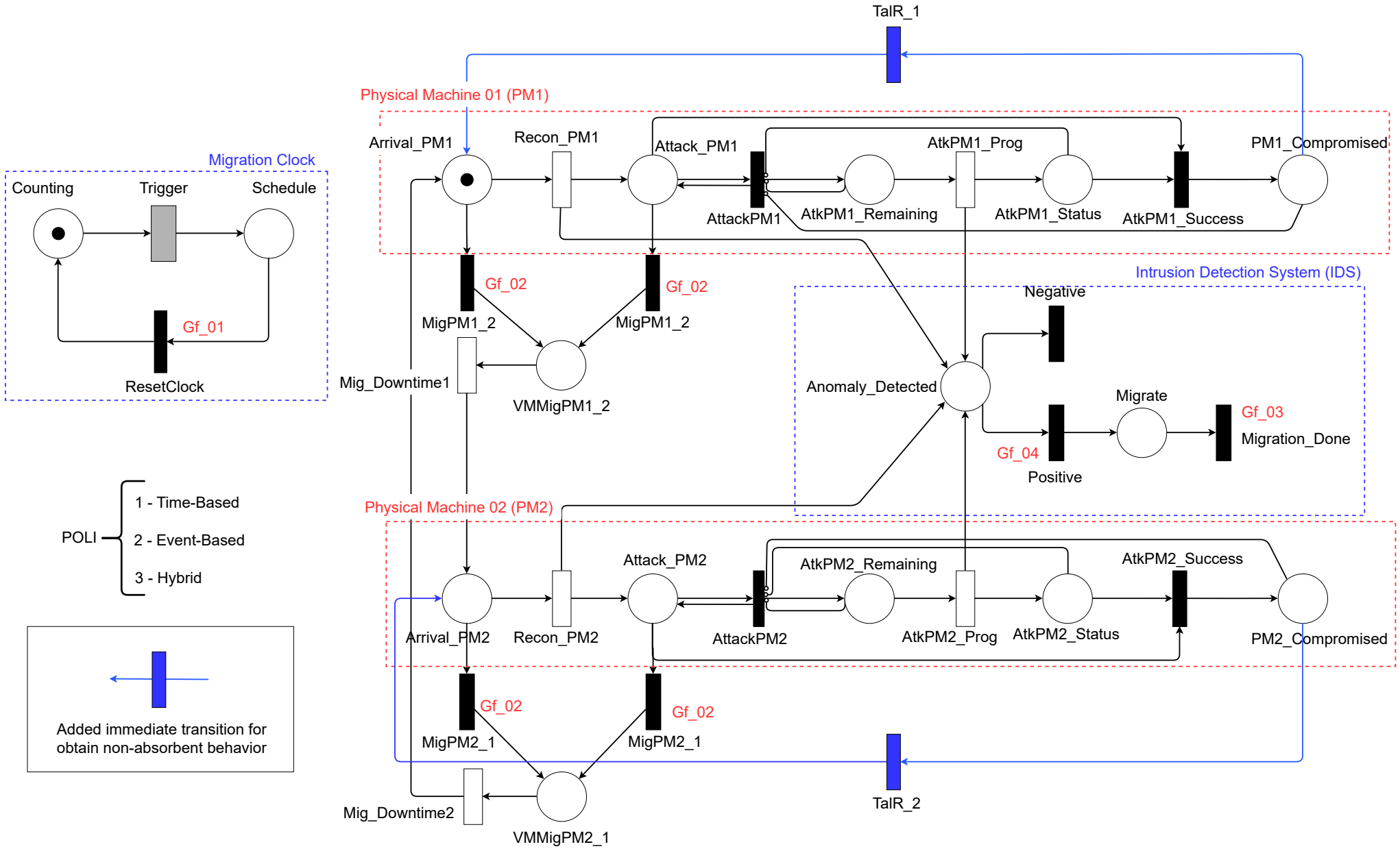


Figure 8 – DSPN model for evaluating intrusion success probability.

Table 2 – Description of the main transitions of the model

Type	**Element**	**Description**	**Value (min)**
Timed transactions	Trigger	Waiting time between the next migration	50 to 300
	Recon_PM1, Recon_PM2	Time taken by the intruder to recognize	30
	AtkPM1_Prog, AtkPM2_Prog	Time the attacker progresses in the attack	360
	Mig_Downtime1, Mig_Downtime2	Time that the system will be offline while the migration is carried out	0.0666667
Immediate transitions	ResetClock	Restart the count for the next migration	-
	Attack_PM1, Attack_PM2	Start of the attack	-
	AtkPM1_Success, AtkPM2_Success	Attack process completed successfully	-
	MigPM1_2, MigPM2_1	Start migration from Machine 01 to Machine 02, and vice versa	-
	Negative	Rule out previously detected behavior as suspicious	-
	Positive	Confirms behavior previously detected as suspicious	-
Places	Counting	Count for the next migration	-
	Schedule	Next migration has started	-
	Arrival_PM1, Arrival_PM2	Intruder's arrival	-
	VMMigPM1_2, VMMigPM2_1	Migration from Machine 01 to Machine 02, and vice versa has started	-
	Attack_PM1, Attack_PM2	Intruder has started attack	-
	AtkPM1_Remaining, AtkPM2_Remaining	Remaining attack progress	-
	AtkPM1_Status, AtkPM2_Status	Current attack progress status	-
	PM1_Compromised, PM2_Compromised	Attack successfully finished	-
	Anomaly_Detected	Anomaly detected by IDS	-
	Migrate	IDS has started new migration	-

The DSPN model provides a detailed representation of a system that operates across two physical machines, employing migration as a defensive mechanism against security threats. This model is specifically designed to evaluate the effectiveness of MTD strategies in enhancing the overall security of the system. Additionally, it plays a crucial role in optimizing system configurations by estimating the required number of migrations and assessing the likelihood of detecting potential security threats. One of the key aspects

Table 3 – Guard expressions whose indexes are highlighted in red

Transition	Code	Expression
ResetClock	Gf_01	(((POLI=1)OR(POLI=3))AND(#VMMigPM1_2>0) OR (#VM-MigPM2_1>0))
MigPM1_2, MigPM2_1	Gf_02	((#Migration>0)OR(#Schedule>0))
Migration_Done	Gf_03	((#VMMigPM1_2>0)OR(#VMMigPM2_1>0))
Positive	Gf_04	((POLI=2)OR(POLI=3))

of the model is its ability to determine which policy is most effective in delaying an attacker's ability to compromise a machine, thereby strengthening the system's defense.

The model incorporates a numerical variable, **POLI**, which determines the migration activation policy. This variable can assume three distinct values: **1, 2, or 3**. When **POLI = 1**, the **Time-Based** strategy is active, meaning migration occurs at predefined intervals. When **POLI = 2**, the **Event-Based** strategy is enabled, triggering migration in response to specific system events. Finally, when **POLI = 3**, a **hybrid strategy** is applied, combining both time-based and event-driven activations. The **Migration Clock** is responsible for initiating migration when the Time-Based strategy is employed, ensuring that migrations occur at regular intervals. The deterministic transition, **Trigger**, plays a crucial role in activating the migration process by moving a token from the **Counting** place to the **Schedule** place, effectively preparing the system for migration. Additionally, the Migration Clock incorporates an immediate transition, **ResetClock**, which resets the token back to the Counting state, allowing the migration cycle to restart. The activation of **ResetClock** is controlled by the guard expression $Gf_0 1$, ensuring that the transition is triggered only after an ongoing migration is completed and when either the **Time-Based** (**POLI = 1**) or hybrid policy (**POLI = 3**) is in effect.

The *IDS* plays an integral role in detecting anomalies that indicate a potential security threat, thereby triggering migration under the *Event-Based* strategy. When a token reaches the *Anomaly_Detected* state, it signifies that an anomaly has been identified, requiring further analysis. Depending on the assessment, if the detected anomaly is deemed non-threatening, the IDS disregards the alert, allowing the token to transition immediately via *Negative*, indicating legitimate user behavior. However, if the anomaly is classified as an intrusion attempt, the *Positive* transition is activated, moving the token to the *Migrate* state. This transition is controlled by the guard expression *Gf_04*, ensuring that migration is triggered only when either the *Event-Based* or hybrid policy is active. The presence of a token in the *Migrate* state indicates that the system is prepared for migration, transferring both the system and the intruder to a new machine. Once migration is completed, the

Migration_Done transition is triggered, governed by the guard expression *Gf_03*, marking the conclusion of the migration process.

The model accurately represents the operational behavior of both *Physical Machine 01* and *Physical Machine 02*. The attacker's initial entry into Machine 01 is signified by a token in the *Arrival_PM1* state. Following this, the attacker undergoes a reconnaissance phase, indicated by the transition *Recon_PM1*, during which vulnerabilities in the system are explored. Once reconnaissance is complete, the attacker progresses to the *Attack_PM1* state, marking the commencement of the intrusion attempt. At this stage, three potential outcomes are possible: the attacker may continue the attack uninterrupted, migration may be triggered by the *Migration Clock*, or the IDS may detect suspicious activity and respond accordingly. Migration, whether triggered before or after reconnaissance, is governed by the transitions *MigPM1_2*, ensuring that migration can only proceed when there is at least one token in either the *Schedule* or *Migrate* states. Upon migration, the *Mig_Downtime1* transition signifies a temporary system unavailability period, after which the token is relocated to the *Arrival_PM2* state in Machine 02, forcing the attacker to repeat the reconnaissance phase from the beginning.

If the attacker completes the reconnaissance phase without interference, the transition *Attack_PM1* is activated, signifying the initiation of the attack process. The attack is modeled through an Erlang-based structure divided into four progressive phases, as proposed by (TORQUATO; MACIEL; VIEIRA, 2020). This structure comprises an immediate transition *AtkPM1*, a transition *AtkPM1_Prog*, and two places, *AtkPM1_Remaining* and *AtkPM1_Status*. The primary objective of this structure is to preserve the progression of the attack while representing an increasing invasion rate, meaning that the probability of a successful attack escalates as the intruder continues their intrusion attempts. This approach effectively captures the attacker's gradual acquisition of knowledge about the system and the subsequent advancement of the attack.

The transition *AttackPM1* signifies the initiation of the attack process, allocating tokens to the *Atk_Remaining* state, representing the effort required to compromise the system entirely. As the attack progresses, the transition *AtkPM1_Prog* gradually moves tokens toward *Atk_Status* and *Anomaly_Detected*, where the IDS assesses the nature of the activity. An accumulation of tokens in *Atk_Status* signifies the successful completion of the attack, triggering the transition *AtkPM1_Success*, which places a token in the *PM1_Compromised* state, officially marking the system as compromised. The transitions marked in blue, *TaIR_1* and *TaIR_2*, are utilized to calculate the migration rate, illustrating the model's execution flow from initiation to a potential system compromise.

The migration rate is determined using Equation 4.1. In this equation, the representation of $E\#VMMigPM1\_2$ means the expected number of migrations from Physical Machine 1 (PM1) to Physical Machine 2 (PM2) within a specified period. The notation

"E" stands for "*Expected*," indicating an estimate or expected average value. Similarly, the query $E\#VMMigPM2\_1$ denotes the expected number of migrations in the reverse direction, from PM2 back to PM1, within the same timeframe.

$$\begin{aligned} MigrationRate = (E\#VMMigPM1\_2/Mig\_Downtime) \\ + \\ (E\#VMMigPM2\_1/Mig\_Downtime) \end{aligned} \tag{4.1}$$

The parameter $Mig\_Downtime$ represents the system downtime associated with each migration event. This downtime accounts for the period during which the system remains idle while the migration process is being executed between hosts. The equation calculates the total migration rate by dividing the sum of the expected number of migrations in both directions by the total downtime incurred due to these migrations. A higher migration rate signifies that the system performs migrations more frequently. Consequently, this leads to an increase in overall downtime, as each migration introduces a temporary period of unavailability.

4.3 Case Studies

This section details the outcomes of simulation experiments conducted on the model we proposed. The goal here is to numerically analyze and compare the impact of various migration policies on a system, particularly focusing on which policy optimally sustains system resilience while minimizing adverse effects. The case study utilized the same set of migration policies as described in Chapter 3.

Figure 9 pertains to the model's **Mean Absorption Time (MTTA)**, which is computed considering the system as an **absorbing process**. The model includes absorbing states that ensure the calculation of MTTA is valid. In contrast, Figure 10 addresses the system's **migration rate**, which is obtained by enabling the transitions *TaIR_1* and *TaIR_2*. This metric is essential because it reflects how frequently migrations are occurring, directly influencing both the level of disruption experienced by attackers and the operational overhead imposed on the system. These transitions are exclusively used for determining migration rates and are deactivated when computing the MTTA to preserve the absorbing nature of the system. In both diagrams, the probability of detecting an intrusion (denoted as the *Positive transition*) was adjusted between 10% and 90%, with the time allocated for executing a migration (*Trigger*) held constant at 320 minutes. Following the discussions in prior sections, three distinct migration policies were evaluated: *Time-based*, *Event-based*, and hybrid.

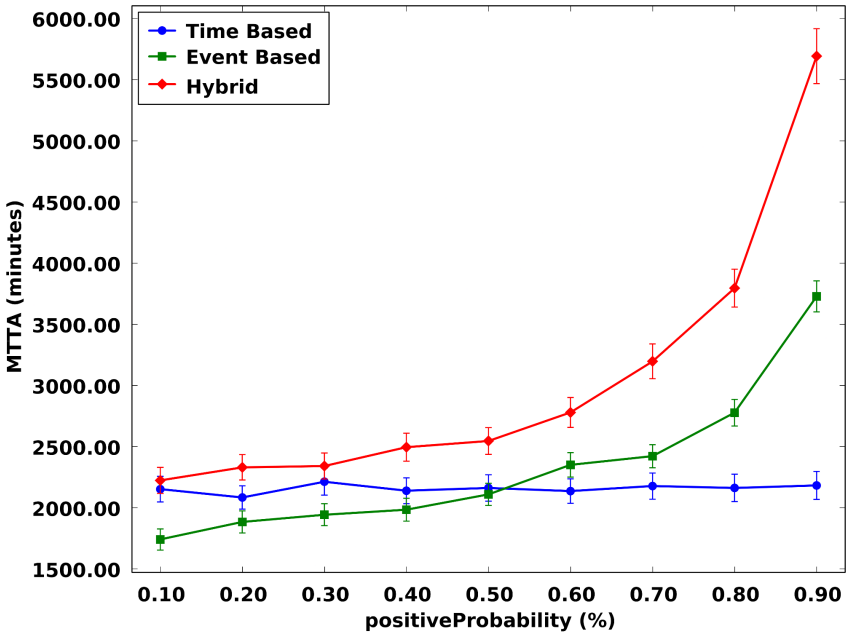


Figure 9 – MTTA varying detection probability.

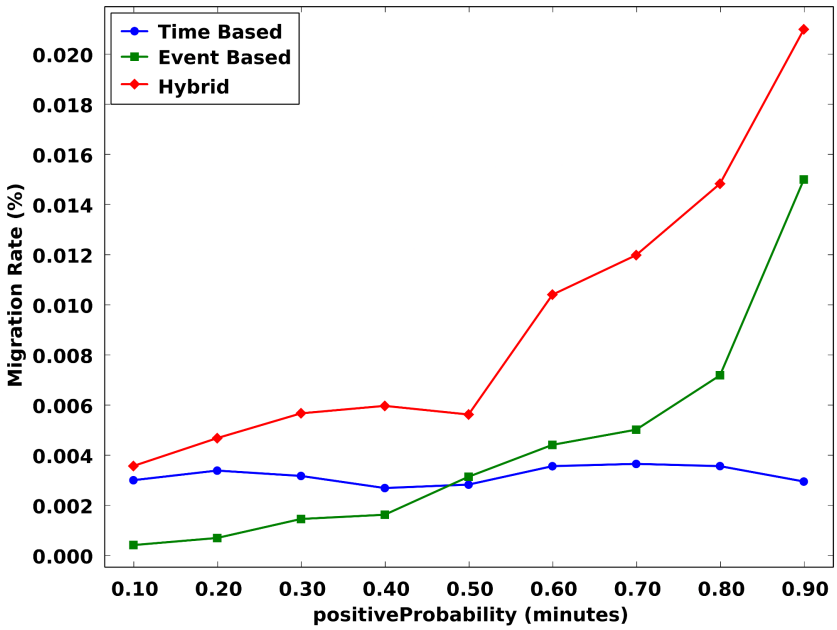


Figure 10 – Migration rate varying detection probability.

The hybrid policy exhibited the most extended MTTA duration, initiating at approximately 2000 minutes with a 10% detection probability and peaking close to 5500 minutes at a 90% probability. Conversely, the *Event-based* policy started at an MTTA of approximately 1500 minutes at a 10% detection likelihood, ascending to a peak of approximately 3500 minutes at a 90% probability. The *Time-based* policy remained stable throughout the analysis, preserving an MTTA of approximately 2000 minutes.

When examining migration rates, the hybrid policy emerged prominently, achieving

a migration rate of approximately 20% at the peak detection probability of 90%. The *Event-based* policy peaked at a 14% migration rate at the same detection probability, while the *Time-based* policy consistently maintained a migration rate of approximately 2%.

The analysis indicates that with an increase in detection probability, both hybrid and *Event-based* policies effectively delay the attacker's progress, thereby necessitating more migrations. It also highlights that the *Time-based* strategy's effectiveness is not influenced by variations in detection probability, as this metric is inherently linked to detecting intrusion attempts. When the IDS's detection confidence is low (below 50%), opting for *Time-based* strategies appears to be more effective.

Figure 11 illustrates the MTTA, which is computed by analyzing the time required for the system to reach an absorbing state representing a successful attack. This metric is derived from the expected absorption time in the DSPN model, considering different migration policies.

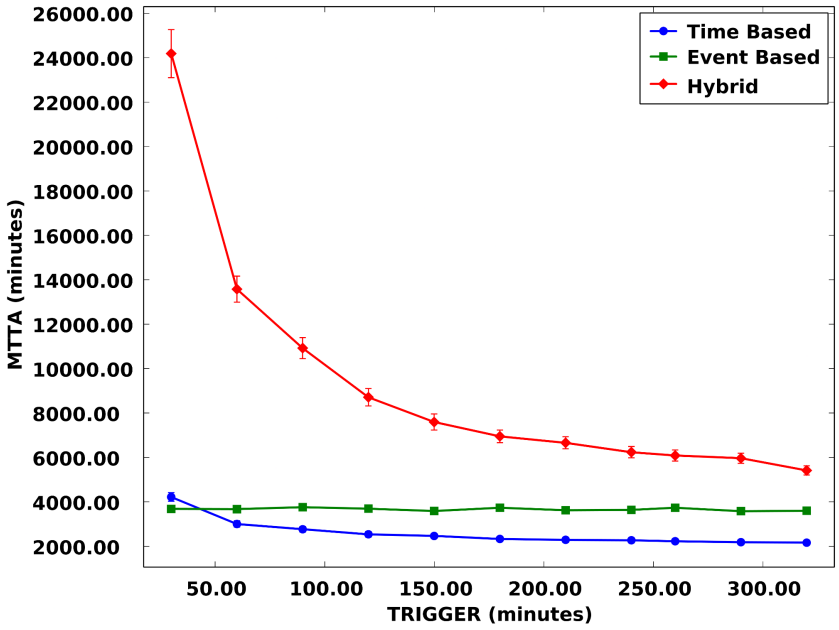


Figure 11 – MTTA varying the average trigger time

Similarly, Figure 12 focuses on the system's migration rate, which is obtained by enabling the transitions *TaIR_1* and *TaIR_2* within the model. These transitions track the number of migrations over time, allowing for the computation of migration frequency under various configurations. In these experiments, the interval for initiating a migration, referred to as the "trigger"was adjusted across a range of values from 50 to 300 minutes. The probability of detecting intrusions remained fixed at 90% throughout the experiments, ensuring that the results reflect the effects of migration policies rather than variability in detection accuracy. The chosen range for the trigger interval provides a comprehensive view of how different policies behave under various conditions of migration frequency.

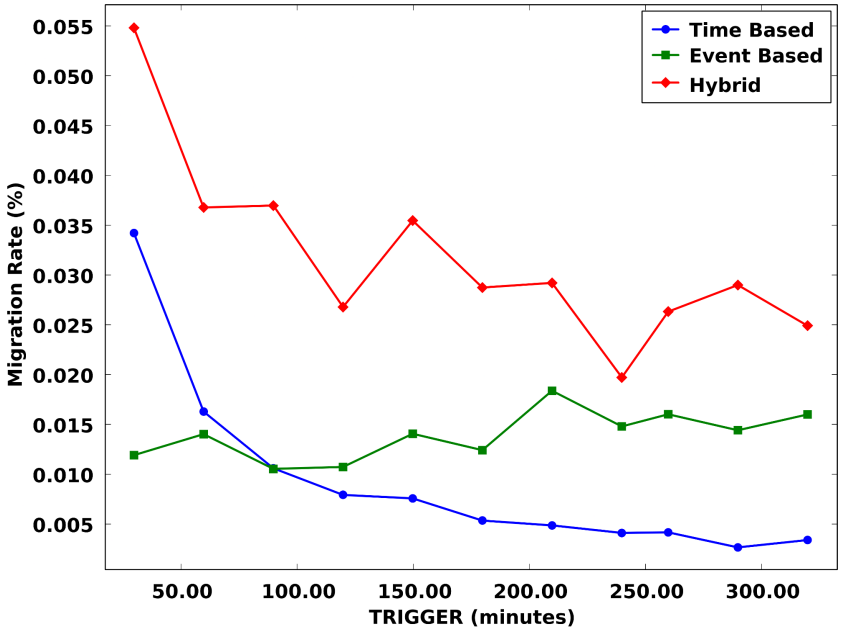


Figure 12 – Migration rate varying the average trigger time

The hybrid policy, which combines aspects of both time-based and event-based migration strategies, initially exhibited the highest MTTA, with a value of approximately 24,000 minutes when the Trigger interval was set to 50 minutes. This suggests that, at shorter intervals, the system was more resilient in terms of detection and response times, possibly due to more frequent migrations or better alignment with the timing of intrusion events. However, as the Trigger interval was gradually extended to 300 minutes, there was a marked decrease in MTTA, falling to approximately 6,000 minutes. This reduction indicates that, over longer intervals, the system became more efficient at mitigating attacks, likely due to the adaptive nature of the hybrid policy, which adjusts its approach to match the prevailing conditions.

In contrast, the *time-based* policy, which relies solely on fixed migration intervals, showed a less pronounced drop in MTTA. Initially, the MTTA was approximately 4,000 minutes when the Trigger was set to 50 minutes. As the Trigger interval increased to 300 minutes, the MTTA decreased steadily to roughly 2,000 minutes. This indicates that, while time-based migrations provide a predictable and regular pattern of activity, the system's ability to reduce MTTA improves slightly as the migration intervals lengthen. However, the time-based policy was less flexible than the hybrid approach, as it could not adapt to the specific timing of intrusion events.

On the other hand, the *Event-based* policy, which triggers migrations based on the detection of specific anomalies rather than on a fixed schedule, demonstrated a notably consistent MTTA throughout the entire experiment. Both the initial and final MTTA values were approximately 4,000 minutes, indicating that the performance of the *Event-based*

policy remained stable regardless of the Trigger interval. This stability can be attributed to the nature of the policy itself, which responds dynamically to detected anomalies rather than relying on a pre-set schedule. As a result, the *Event-based* strategy was unaffected by changes in the migration interval, highlighting its robustness in environments where intrusion patterns are irregular or unpredictable.

These findings collectively suggest that the hybrid policy, while starting with a longer MTTA, benefits from the increased flexibility provided by longer migration intervals, which ultimately reduces the overall attack duration. The *Time-based* policy, while less adaptable, still achieves a reduction in MTTA as the migration intervals increase, but to a lesser extent than the hybrid approach. Finally, the *Event-based* strategy demonstrates a steady and reliable performance, independent of the trigger interval, making it a practical choice in environments where a timely response to specific anomalies is crucial.

Figure 13 presents a CDF graph that evaluates the effectiveness of Time-based, Event-based, and hybrid migration policies in terms of the probability of a successful intrusion. The key metric analyzed here is the absorption time, which refers to the duration required for an attacker to breach the system successfully. A longer absorption time means that the attacker needs more time and resources to complete a successful intrusion, effectively increasing the system's resilience against potential breaches. This time is crucial for system defenses, as it offers the opportunity for countermeasures to be deployed and potentially thwart the attack.

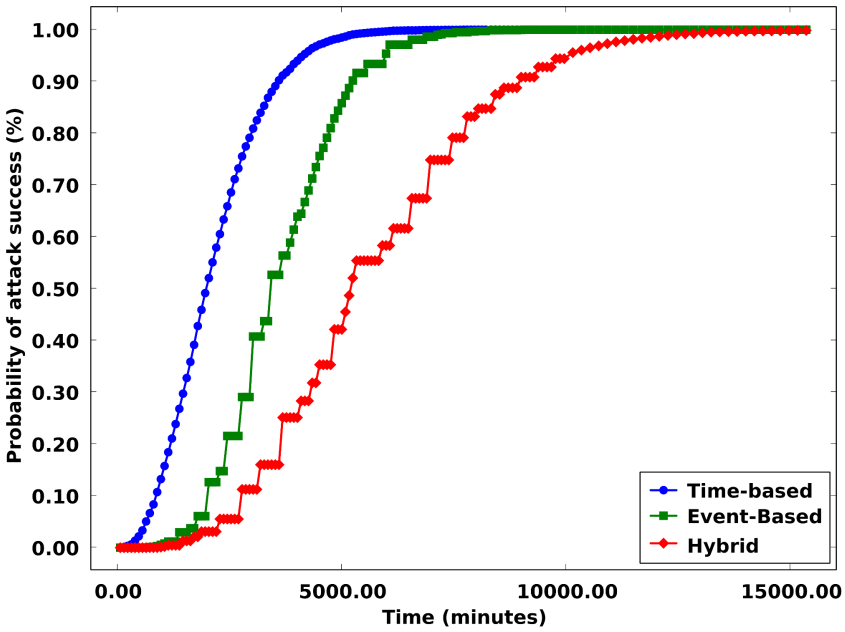


Figure 13 – Probability of intrusion success varying migration policies.

In this particular study, the probability of detecting an intrusion was consistently set at 90%, ensuring that the results reflect the impact of the migration policies rather

than variations in detection efficiency. Additionally, the interval for initiating scheduled migrations (referred to as Trigger) was fixed at 320 minutes, providing a consistent basis for comparing the performance of the different policies under controlled conditions.

The analysis begins with the *time-based* policy, which demonstrates the quickest absorption time. This indicates that the policy allows an intrusion to succeed in approximately 7,000 minutes. The relatively short absorption time suggests that the time-based migrations, while predictable and regular, may not be as effective in delaying an attack when compared to more dynamic strategies. This could be due to the fixed nature of the scheduled migrations, which may not align optimally with the timing of an attack.

Next, the *Event-based* policy shows an absorption time of approximately 9,000 minutes, which is longer than the *time-based* policy, indicating that it is more effective at delaying an attacker. The Event-based policy's reliance on real-time anomaly detection allows for more dynamic adjustments to the system's defenses, making it more resilient to attacks that are unpredictable or timed irregularly. By responding to specific anomalies rather than fixed intervals, this policy provides a better defense mechanism in cases where attack patterns are not easily anticipated.

The hybrid policy, which combines elements of both time-based and event-based strategies, demonstrates the longest absorption time, taking approximately 12,000 minutes. This suggests that the hybrid policy is the most effective in this particular scenario for delaying an attacker's efforts to compromise the system. The extended absorption time can be attributed to the hybrid policy's ability to leverage both the scheduled migrations for predictable resilience and event-based migrations for more responsive, context-sensitive protection. This dual approach provides enhanced flexibility, allowing the system to adapt to a broader range of attack patterns.

The findings from the CDF graph clearly indicate that the hybrid policy provides the longest absorption time, thus serving as the most effective strategy in this scenario for delaying an attacker's attempts to breach the system. These results highlight the advantages of using a more adaptive, combined approach, where both scheduled and event-driven migrations contribute to overall system resilience.

4.4 Sensitivity Analysis

This section presents a sensitivity analysis aimed at identifying the critical factors within the proposed DSPN model, employing a DoE methodology. DoE allows for the systematic manipulation of input variables to discern the underlying causes of variations in the output response, providing a structured approach to experiments (FUKUDA et al., 2018). As a statistical methodology, DoE is highly effective in evaluating the impact of different system components on the overall performance, enabling the identification of the

most influential factors.

By applying DoE to the DSPN model, this analysis offers valuable insights into potential optimizations by examining various scenarios and understanding how alterations in the parameters of system components influence functionality. This comprehensive methodology helps uncover opportunities for improvements, particularly in optimizing system performance, with a strong focus on enhancing the migration rate. This key metric directly impacts the efficiency of the system.

The DoE analysis conducted in this study incorporates two key types of graphs: *Main Interaction* and *Effects*. The *Main Interaction* graphs display the relationship between two system factors, highlighting whether one factor has a significant impact on the other. In these graphs, parallel lines suggest the absence of interaction, while intersecting lines indicate a meaningful interaction between the factors. This allows for the identification of critical relationships that might influence the system's behavior.

On the other hand, *Effects* graphs utilize horizontal and vertical bars to rank the system components based on their impact on the output, ordered in descending importance. The length of each bar directly correlates with the significance of that component to the system's overall performance, facilitating the identification of the most critical factors. These graphs provide a clear visual representation of which components are most responsible for variations in the system's behavior.

The experiment is based on the architecture described previously, with a particular focus on conducting the sensitivity analysis in relation to its effect on the migration rate metric. The migration rate was chosen as the focal point due to its direct influence on system downtime, a factor that significantly impacts the overall user experience. To ensure a thorough analysis, five components of the system were selected for this study: (i) *trigger*, (ii) *positiveProbability*, (iii) *recom_pm*, (iv) *atkpm_prog*, and (v) *mig_downtime*. Each of these components was evaluated using low and high settings, providing a comprehensive range of conditions for the DoE.

Table 4 outlines the specific settings applied to each of the five components in the DoE, providing an apparent reference for the experimental setup. Additionally, Table 5 presents the outcomes of the experiments based on the various combinations of these variables. This detailed analysis allows for a deeper understanding of how each factor contributes to the overall performance of the DSPN model, offering clear guidance for future optimizations and system improvements.

4.4.1 Numerical Analysis

This section details the outcomes of the sensitivity analysis conducted through the DoE methodology. The analysis focuses on identifying the most critical system components

Table 4 – Design configuration

Factor name	**Low Setting**	**High Setting**
trigger (min)	50.0	300.0
positive_prob (%)	0.1	0.9
recon_pm (min)	30.0	130.0
atkpm_prog (min)	100.0	360.0
mig_downtime (min)	0.0666667	0.1

Table 5 – Combination of factors and levels

trigger (min)	**positive_prob (%)**	**recon_pm (min)**	**atkpm_prog (min)**	**mig_downtime (min)**	**mig_rate (%)**
50.00	0.10	30.00	100.00	0.06	0.02
50.00	0.10	30.00	100.00	0.10	0.02
50.00	0.10	30.00	360.00	0.06	0.02
50.00	0.10	30.00	360.00	0.10	0.02
50.00	0.10	130.00	100.00	0.06	0.02
50.00	0.10	130.00	100.00	0.10	0.02
50.00	0.10	130.00	360.00	0.06	0.02
50.00	0.10	130.00	360.00	0.10	0.02
50.00	0.90	30.00	100.00	0.06	0.04
50.00	0.90	30.00	100.00	0.10	0.04
50.00	0.90	30.00	360.00	0.06	0.04
50.00	0.90	30.00	360.00	0.10	0.05
50.00	0.90	130.00	100.00	0.06	0.02
50.00	0.90	130.00	100.00	0.10	0.02
50.00	0.90	130.00	360.00	0.06	0.02
50.00	0.90	130.00	360.00	0.10	0.02
300.00	0.10	30.00	100.00	0.06	0.00
300.00	0.10	30.00	100.00	0.10	0.00
300.00	0.10	30.00	360.00	0.06	0.00
300.00	0.10	30.00	360.00	0.10	0.00
300.00	0.10	130.00	100.00	0.06	0.00
300.00	0.10	130.00	100.00	0.10	0.00
300.00	0.10	130.00	360.00	0.06	0.00
300.00	0.10	130.00	360.00	0.10	0.00
300.00	0.90	30.00	100.00	0.06	0.03
300.00	0.90	30.00	100.00	0.10	0.02
300.00	0.90	30.00	360.00	0.06	0.02
300.00	0.90	30.00	360.00	0.10	0.02
300.00	0.90	130.00	100.00	0.06	0.01
300.00	0.90	130.00	100.00	0.10	0.01
300.00	0.90	130.00	360.00	0.06	0.00
300.00	0.90	130.00	360.00	0.10	0.00

and understanding their impact on the overall performance of the system. Figure 14 presents the *Effects* graph, which visually represents the significance and relative impact

of various factors in the system. The graph serves as a valuable tool for assessing the contribution of each factor to the system's overall behavior, providing insights into the components that play the most pivotal roles in its functionality.

Among the factors analyzed, the *trigger* component, which determines the timing for initiating scheduled migrations, emerged as the most influential factor in the system's performance. This finding highlights the importance of carefully setting the migration interval to optimize system resilience and reduce downtime. The *trigger* setting directly governs how frequently migrations are initiated, influencing how well the system adapts to changing conditions and potential threats.

In close proximity to *trigger*, the *positiveProbability* factor was identified as the second most impactful element. This factor refers to the likelihood of the IDS recognizing an attack within the system. The effectiveness of the IDS is crucial for the timely identification and mitigation of intrusions, and *positiveProbability* plays a central role in determining how quickly the system can react to a potential threat. A higher probability of detection allows the IDS to respond faster, thus influencing the migration strategy and enhancing system resilience.

The third significant contributor identified in the analysis was the *recon_pm* factor, which represents the time required for an attacker to familiarize themselves with the system environment once they have successfully infiltrated the system. This factor indirectly influences the migration rate by impacting the time window available for detecting and mitigating an attack before it can cause significant damage. A longer *recon_pm* duration provides more time for the IDS to recognize suspicious activities and initiate defensive actions.

A noteworthy interaction was observed between the *positiveProbability* and *recon_pm* factors. This interaction emphasizes that the combined effect of these two components plays a substantial role in determining the effectiveness of the IDS in detecting intrusions. When both factors align optimally, the IDS can more accurately identify attacks, which, in turn, affects the system's migration rate. This interaction reveals that the detection capabilities of the IDS and the attacker's ability to learn the system's behavior are intricately linked, making it critical to consider these components together when optimizing the system's performance.

The sensitivity analysis also reveals that while other factors and their interactions contribute to the system's overall behavior, they have a comparatively lesser impact on determining the migration rate. This underscores the prominence of *trigger*, *positiveProbability*, and *recon_pm* in shaping the system's ability to mitigate intrusions effectively. The findings highlight the need for a targeted approach to optimizing these key components to achieve the desired level of system resilience and performance.

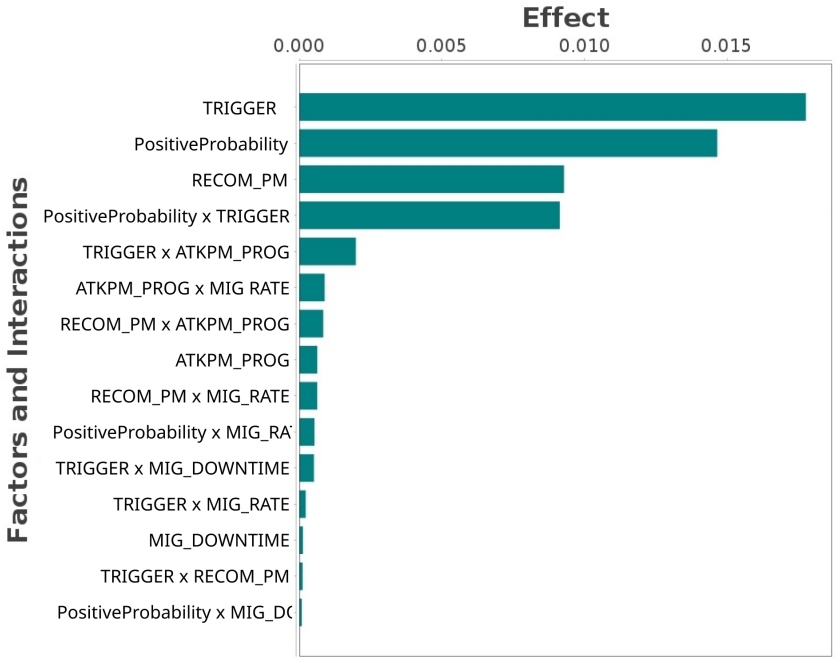


Figure 14 – Impact of factors and interactions

Figures 15, 16, 17, and 18 present the *Main Interaction* graphs generated from the DoE analysis. These graphs visually depict the impact of various factor combinations on the system's migration rate, offering a detailed understanding of how different elements interact to influence system performance.

Figure 15 examines explicitly the relationship between the probability of intrusion detection and the progression of an attack. The graph demonstrates a clear trend: as the probability of intrusion detection increases, the migration rate also rises. This finding highlights the crucial role that detection probability plays in enhancing the system's ability to respond to threats. A higher detection probability allows the system to identify potential intrusions more quickly, triggering a more frequent and effective migration strategy. Consequently, the system becomes better equipped to mitigate attacks, emphasizing the need for a robust detection mechanism to optimize system performance. In Figure 16, the interaction between detection probability and migration downtime is explored. This graph investigates how changes in system unavailability, particularly during migration, correlate with the migration rate. It reveals that variations in the downtime due to migration are closely tied to the detection probability. As the migration rate increases, so does the system's downtime, but the level of detection probability influences how significant this downtime becomes. Essentially, the higher the detection probability, the more proactive the system becomes in initiating migrations, leading to a trade-off between minimizing system downtime and ensuring timely threat mitigation. This interaction underscores the complex balance between system availability and migration frequency, where improvements in detection lead to more frequent migrations, but potentially increase downtime during

these processes.

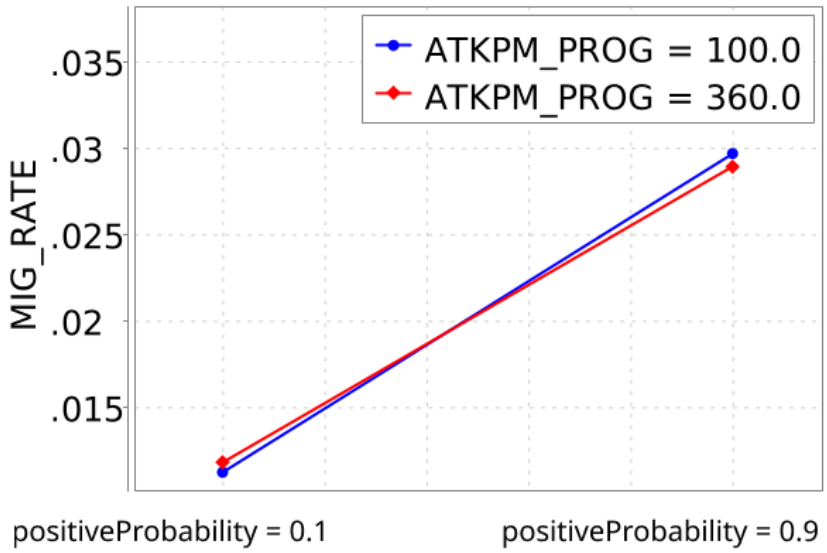


Figure 15 – Probability of detection x Downtime.

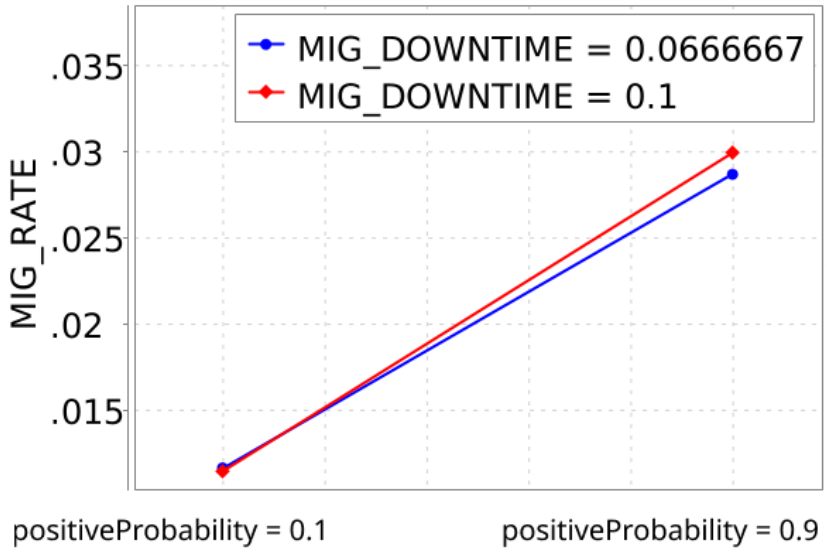


Figure 16 – Probability of detection x Downtime.

In Figure 17, the analysis shifts to exploring the interaction between the initiation time of scheduled migrations and the progression of attacks. The graph reveals a significant relationship between the timing of migration triggers and the rate at which an attack advances. Specifically, it shows that reducing the interval before initiating a migration leads to a noticeable increase in the migration rate. This outcome is directly linked to the *trigger* mechanism, which governs the timing between consecutive migrations. As the *trigger* interval shortens, the system initiates more frequent migrations, thereby enhancing the overall responsiveness to attacks. This finding underscores the importance of carefully managing the *trigger* setting to ensure that migrations are timely and effective in mitigating

attacks. The ability to fine-tune the *trigger* value, therefore, has a significant impact on the system's ability to respond to threats in real time, making it a key factor in optimizing system performance. Figure 18 takes the analysis a step further by examining how the timing of migration initiation influences system downtime during the migration process. The graph illustrates a direct correlation between the *trigger* interval and the system's downtime, revealing that longer *trigger* intervals result in slower migration rates. As the interval increases, the system migrates less frequently, which in turn leads to more extended periods of unavailability during migration. This relationship highlights a crucial trade-off between the frequency of migrations and system availability. While longer *trigger* intervals may reduce the overall number of migrations, they also increase the downtime associated with each migration event, potentially affecting the user experience and system performance.

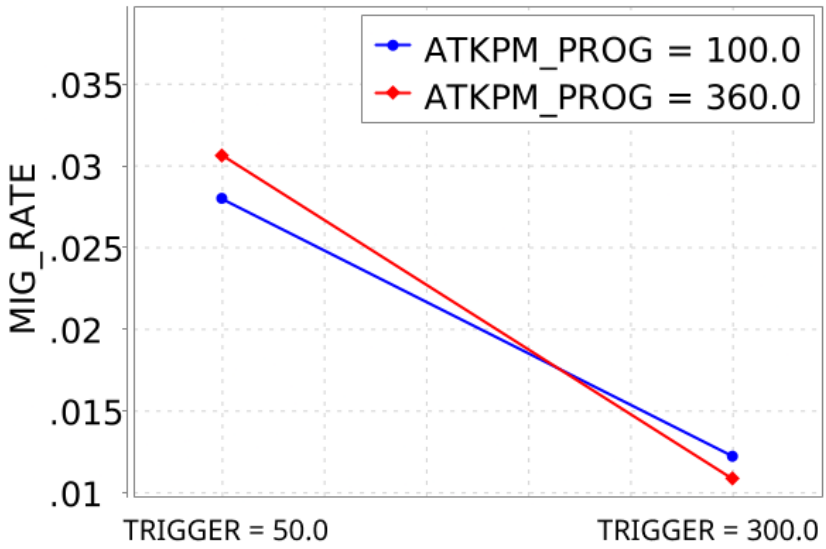


Figure 17 – Trigger x Attack progress.

Figure 19 presents a 3D plot that visualizes the impact of two critical factors, the IDS detection probability (*positiveProbability*) and the interval before initiating a scheduled migration (*trigger*), on the system's migration rate. The analysis, which is grounded in the results of the DoE methodology, reveals how variations in these factors influence system performance. The experiment systematically adjusted the detection probability to 20%, 40%, 60%, and 90%, while the *trigger* interval for triggering a migration was varied across three settings: 100, 200, and 300 minutes. The 3D plot provides a clear view of the combined effects of these two factors, showcasing how different levels of detection probability and migration trigger intervals interact to affect the rate of migration. By varying the detection probability, the analysis examines how an increased likelihood of intrusion detection influences the system's ability to respond to threats. As detection probability increases, the system's ability to identify and react to intrusions improves, leading to a higher migration rate. However, the *trigger* interval also plays a significant

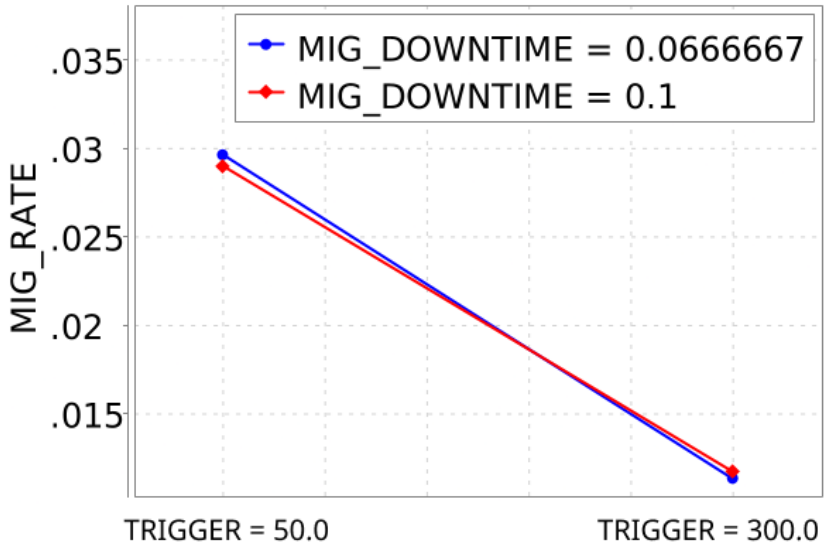


Figure 18 – Trigger x Downtime.

role, with shorter intervals leading to more frequent migrations, while longer intervals result in fewer migrations.

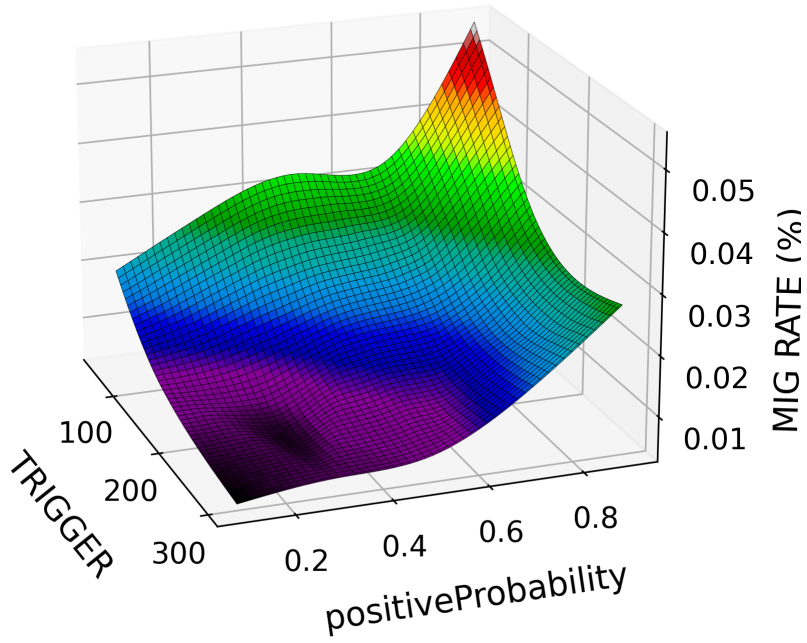


Figure 19 – Simultaneous variation in trigger and detection probability.

The graph clearly demonstrates a positive correlation between the migration rate and the IDS detection probability, with the most notable effects observed when the interval before initiating a migration (*trigger*) is reduced. When the *trigger* period is set at 300

minutes, the migration rate stabilizes at approximately 1%, and this rate remains constant until the detection probability surpasses 40%. Beyond this threshold, the migration rate begins to increase, reaching around 2% when the detection probability is raised to 80%. This indicates that higher detection probabilities lead to more frequent migrations, particularly when the time to trigger these migrations is longer, thus providing more opportunities for the system to respond to potential threats.

In contrast, when the *trigger* interval is set to 200 minutes, the migration rate rises significantly, approaching 3% when the detection probability is 80%. This increase reflects the combined influence of both the detection probability and the reduced trigger time, suggesting that a shorter *trigger* period facilitates a more responsive system. The most pronounced effect occurs when the *trigger* is set to 100 minutes, where the migration rate fluctuates between approximately 4% and 5%, depending on whether the detection probability is 80% or 90%, respectively. This variation indicates that at shorter *trigger* intervals, the system becomes even more responsive to changes in detection probability, further enhancing its migration frequency as the likelihood of detecting intrusions increases.

These findings illustrate the interaction between the *trigger* duration and the IDS detection probability, demonstrating how each factor independently and synergistically influences the system's migration rate. The analysis reveals that when the *trigger* period is longer, any increase in detection probability above 40% triggers a noticeable rise in the migration rate. In contrast, shorter *trigger* intervals amplify this effect. Additionally, the results underscore the importance of fine-tuning both variables to optimize the system's migration behavior, balancing between migration frequency and system availability to improve the system's overall resilience and responsiveness to intrusions.

4.4.2 Discussions

- MTD Strategies: In this dissertation, we primarily focus on the event-based VM migration approach for Moving Target Defense, as it represents the core innovation of our work. The integration of intrusion detection systems that trigger migrations provides a dynamic and proactive defense mechanism, significantly enhancing the adaptability of security measures against sophisticated cyber threats. While our study also evaluates time-based and hybrid migration policies, these are included to offer a comprehensive comparative analysis. The event-based method, being a novel contribution, is emphasized in the title to reflect the primary innovation of our research accurately. The hybrid approach, which merges both time-based and event-based triggers, is treated secondarily as it does not represent a distinct category of innovation but rather an integration of the two primary methods discussed. This prioritization in our focus and presentation underscores the significance of event-based strategies in advancing security technologies.

- Modeling approach: In this work, we have incorporated a formal description of the Deterministic and Stochastic Petri Nets (DSPN) model to elucidate its application and effectiveness in modeling dynamic Moving Target Defense (MTD) strategies within computational environments. DSPNs are chosen for their capability to handle complex behaviors and interactions that are pivotal in assessing the probabilistic nature of system states and transitions. This contrasts with Stochastic Reward Nets (SRN), which are typically utilized for reward-based analysis where system states are weighted to reflect their importance or cost. Our manuscript extends the existing DSPN framework to include adaptations specific to detecting and responding to insider attacks through VM migrations. This adaptation is essential to reflect the real-world complexities and adaptive strategies necessary for enhancing security. By detailing these modifications, we aim to provide a comprehensive and clear insight into the effectiveness of various MTD strategies, thereby contributing novel perspectives to the field of cybersecurity.

- Impact of VM Migration Techniques: In this work, while our primary focus has been on the technical evaluation of VM migration strategies as moving target defense tactics within computing environments, we recognize the significance of these strategies on end-user experience and service quality. Live VM migration techniques such as pre-copy, post-copy, and hybrid play a crucial role in resource management in cloud data centers, facilitating tasks like load balancing, fault management, and system maintenance with minimal downtime. However, these techniques must carefully manage the security of transient data, and balance CPU, memory, and network states to minimize service interruptions, which are critical for maintaining QoS and user experience. Furthermore, VM consolidation strategies, which aim to optimize resource usage and improve energy efficiency, also need to consider the latency and potential service disruptions they introduce. To mitigate these effects, advanced machine learning and heuristic approaches are employed to predict resource needs and adjust VM allocations accordingly, striving to minimize energy consumption while maintaining service quality. This delicate balance underscores the complex interplay between ensuring efficient, secure cloud operations and enhancing user satisfaction through reliable and responsive services.(HIJJI et al., 2022; AHMAD et al., 2015; CHOUDHARY et al., 2017)

4.4.3 Findings

The **findings** of this study are summarized in the following points, which provide a comprehensive overview of the key insights derived from the research:

- **Efficacy of event-detection policies:** The study concludes that event-detection

policies play a pivotal role in significantly improving system security, particularly when the accuracy of the IDS exceeds 50%. This finding underscores the critical importance of implementing an efficient and precise IDS as a foundational component of any MTD strategy. Accurate intrusion detection allows the system to dynamically adapt its defenses to real-time threats, enabling timely activation of event-based policies to counteract potential intrusions. Furthermore, this emphasizes the necessity of continuous refinement and calibration of IDS to maintain its accuracy and reliability, as even marginal improvements in detection rates can substantially enhance the system's security posture.

- **Superiority of hybrid policy:** Among the evaluated strategies, the hybrid policy emerges as the most effective in delaying an attacker's progress. By combining the strengths of time-based and event-based triggers, the hybrid approach introduces a balanced methodology that not only enhances system security but also maintains optimal performance. Time-based triggers ensure periodic defensive actions, while event-based triggers provide targeted responses to detected threats. Together, these elements create a dynamic and adaptive defense mechanism that is both proactive and reactive. The findings suggest that the hybrid policy is particularly well-suited for computing environments, where the dynamic nature of resources and workloads requires a flexible yet robust security framework. This strategy achieves a critical balance, minimizing the trade-offs between security enhancements and system performance degradation, making it a promising solution for modern cybersecurity challenges.

- **Impact on system migration rate and mean time to absorption (MTTA):** The study further reveals that the choice of migration initiation policy has a direct impact on critical performance metrics such as the system's migration rate and the mean time to absorption (MTTA). The migration rate refers to the frequency at which VMs are relocated as part of the MTD strategy, while MTTA represents the average time required for the system to transition into a stable state after a defensive action. Adjusting these policies allows for a significant reduction in the system's vulnerability to attacks, as well as an increase in the time required for attackers to achieve their objectives. The findings highlight the importance of fine-tuning migration initiation policies to strike an optimal balance between security and performance. For instance, increasing the migration rate can enhance security by frequently disrupting an attacker's reconnaissance efforts, but it may also lead to higher system downtime. Conversely, reducing the migration rate might preserve system performance but could leave the system more exposed to threats. Therefore, the study underscores the necessity of a carefully calibrated approach to policy adjustment, ensuring that both security and operational efficiency are maximized.

5 Decoy Analysis

This chapter presents a study based on the methodology of multiple decoy servers, designed to mislead attackers and prolong their intrusion attempts. In addition, an IP migration strategy is employed to prevent adversaries from mapping and tracking real servers. By combining these techniques, the study aims to enhance system resilience, significantly extending the time required for a successful system compromise and increasing the complexity and cost of attacks.

5.1 Environment

This section presents an overview of the environment proposed in this work. The proposed architecture is based on the works of (TORQUATO; MACIEL; VIEIRA, 2021) and (SUN; SUN, 2016), acting as an extension and integration of both. The model illustrates the steps taken by an attacker during the system intrusion process, as well as the mechanisms employed by the model to execute the defense, using decoy servers as the primary strategy. It is important to highlight that the use of decoy servers as a defense technique in the context of MTD does not guarantee 100% effective protection against intrusions. This approach seeks, primarily, to delay the progress of attackers, hindering their ability to compromise the servers and system environments. Although effective in delaying the attacker's advance, this strategy is not entirely immune to intrusions, functioning as an additional layer of security rather than a definitive solution.

Figure 20 illustrates the proposed architecture, which involves two server farms. The Real Server Farm has legitimate servers that run the application or store data. The Decoy Server Farm has fake servers that mimic the real ones, but without posing any risk to the system. The Decoy Server Farm serves as bait to attract attackers. When attempting to invade, the attacker must choose which set of servers to attack, but does not know which are real or fake. If he chooses a decoy server, he will need to restart the attack after taking over the server, since the decoys do not pose a real threat.

The Randomization Controller component acts as an additional layer of defense, forcing the periodic change of IP addresses of all machines, whether real or decoy. This strategy prevents the attacker from continuously tracking his target machine, forcing they to restart the attack if he chooses a server from the Real Server Farm. In addition, the controller prevents the attacker from creating a list of addresses, differentiating real and decoy servers, disorienting him, and making his choices more difficult. Even if the attacker accumulates some knowledge and progress throughout the attack, the techniques based on the Decoy Server Farm and the Randomization Controller seek to slow the attacker's

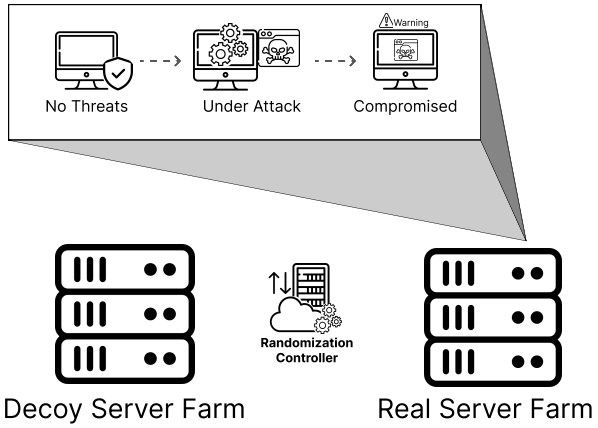


Figure 20 – Stages for real server compromise

progress, making him unable to control the environment effectively.

Figure 21 illustrates the process of changing IP addresses during an intrusion attempt, with emphasis on the Randomization Controller's performance. This component forces the change of IP addresses of all machines, causing the attacker to lose track of his target machine's location. The attacker, even if he has accumulated knowledge and saved progress, will not be able to identify the server that was previously attacked. This forces him to restart the process, randomly choosing a new machine. The attacker may end up selecting a real server again or falling for a decoy server, which, like bait, diverts his attention and hinders his progress, depending on the system's defense strategy.

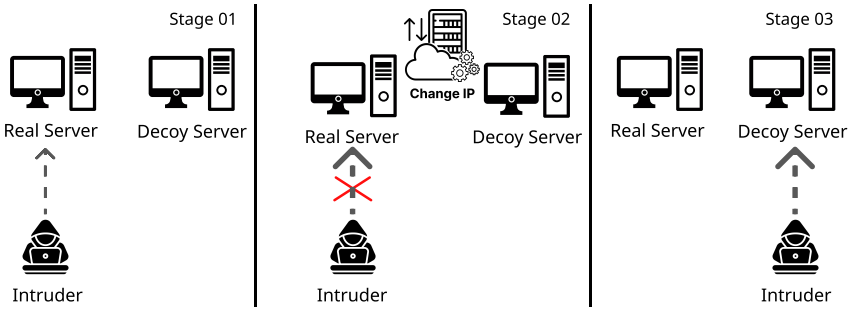


Figure 21 – Stages of the IP address change process

5.2 Model Overview

This section presents an SPN model that represents the environment described in the previous Section. Figure 22 illustrates the proposed SPN model, while Table 6 summarizes the main parameters used in the transitions. All values were obtained based on the model of (TORQUATO; MACIEL; VIEIRA, 2021). The objective of the model is to implement the strategy of multiple decoy servers in an MTD approach, delaying

the advancement of intrusions and strengthening the resilience of the system. The model considers a system with two physical machines and decoy servers used as defense elements to deceive attackers and divert them from the real servers, which are the critical targets. It is worth mentioning that the number of physical and virtual machines represented in the model can be expanded. The intrusion is considered successful when a real server is compromised. The model seeks to demonstrate how the use of decoys increases the time required for an attacker to compromise a real server, in addition to assisting in system planning and configuration. The model also allows for the assessment of the probability of threat detection and the impact of decoys on resilience against attacks. In particular, the model analyzes the ideal number of decoys required to extend the time it takes to compromise a machine significantly.

The *Randomization Controller* represents the system component responsible for triggering IP address changes on both real and decoy machines. This mechanism operates independently of the attack phase, meaning that IP switching can occur at any moment — whether before, during, or after an intrusion attempt — and this behavior is explicitly captured in the model's structure and transitions. The deterministic transition *Trigger* triggers the IP change by moving a token from the *Established_IP* location to the *Change_IP* location, indicating that the system is ready to start the address change. The *Randomization Controller* component also has an immediate transition called *Establish_IP*, designed to reset the token back to the *Established_IP* location, restarting the countdown for the subsequent IP migration. The triggering of the *Establish_IP* transition is controlled by the guard expression *Gf_02*, which serves to ensure that the transition is only triggered when the system is not compromised, avoiding unnecessary address changes.

The *Server Farm Entrance* is the entrance to the system, where the attacker must choose which server to attack: real or decoy. This choice is represented by Atk_Choice. If the attacker chooses a real server, the Real_Farm transition is triggered, and the token arrives at the Arr_S_Farm location, indicating that he has chosen a real server. The attack on the real server is initiated with the atk_PM transition. If the attacker chooses a decoy server, the Decoy_Farm transition is triggered, the token arrives at Arr_D_Farm, and the attack on the decoy server is initiated with atk_Decoy. This model ensures that the attacker cannot distinguish between real and decoy servers.

The *Real Server Farm* and *Decoy Server Farm* components represent sets of machines that can be real servers or decoy servers. When an attacker reaches one of these machines, his presence is registered by the arrival of a token at the Arr_PM (for real servers) or Arr_DECOY (for decoy servers) locations. After a time interval, the recon_PM or recon_DECOY timed transition is triggered, representing the period necessary for the attacker to perform the initial recognition of the target. If the transition is successful, the token is moved to the atk_PM or atk_DECOY locations, indicating that the attacker

has initiated the intrusion process. From this point on, the model offers two distinct possibilities for the evolution of the intrusion scenario. The first possibility is that the attacker continues his intrusion attempts without being interrupted, persisting in his actions. The second possibility involves IP address swapping, which is triggered by the *Randomization Controller* component. Regardless of whether the attacker is attempting to break into a real machine or a decoy server, the change_IP immediate transition will be triggered as soon as the randomization controller starts changing the IP addresses of all machines in the system. When this transition occurs, the attacker is removed from the Arr_PM or Arr_DECOY slots and moved to the atk_choice slot, symbolizing that the machines' IP addresses have changed and the attacker has lost track of the targets. In this scenario, the attacker will need to reselect a machine to resume the attack.

If the attacker completes the reconnaissance phase without interruptions, the atk_PM or atk_DECOY transitions are triggered, marking the beginning of the attack phase. This phase follows the approach described in (TORQUATO; MACIEL; VIEIRA, 2021), structured as a four-stage Erlang chain. This modeling allows representing both the gradual advancement of the attacker's knowledge about the target machine and the preservation of the intrusion progress, even if the attacker temporarily loses the machine's address and needs to resume the attack. When triggering the atk_PM or atk_DECOY transitions, four tokens are deposited in the atk_remaining place, representing the effort required to compromise the machine. The timed atk_prog transition gradually removes these tokens, transferring them to atk_status. The attack phase is completed when the four tokens are moved to atk_status. If the attack targets a real machine, the immediate transition atk_success is triggered, moving a token to the PM_compromised slot, which symbolizes the complete compromise of the system and the attacker's success. In contrast, if the attack targets a decoy machine, the token is moved to DECOY_Honeypot. In this case, the decoy does not compromise the real system, and the attacker is forced to restart their actions. The restart is represented by the immediate transition restart, which returns a token to the atk_choice slot, forcing the attacker to select a new machine to continue his attack. This dynamic demonstrates the resilience of the system by diverting the attacker's attention to decoy servers, delaying the progress of the attack, and protecting critical assets.

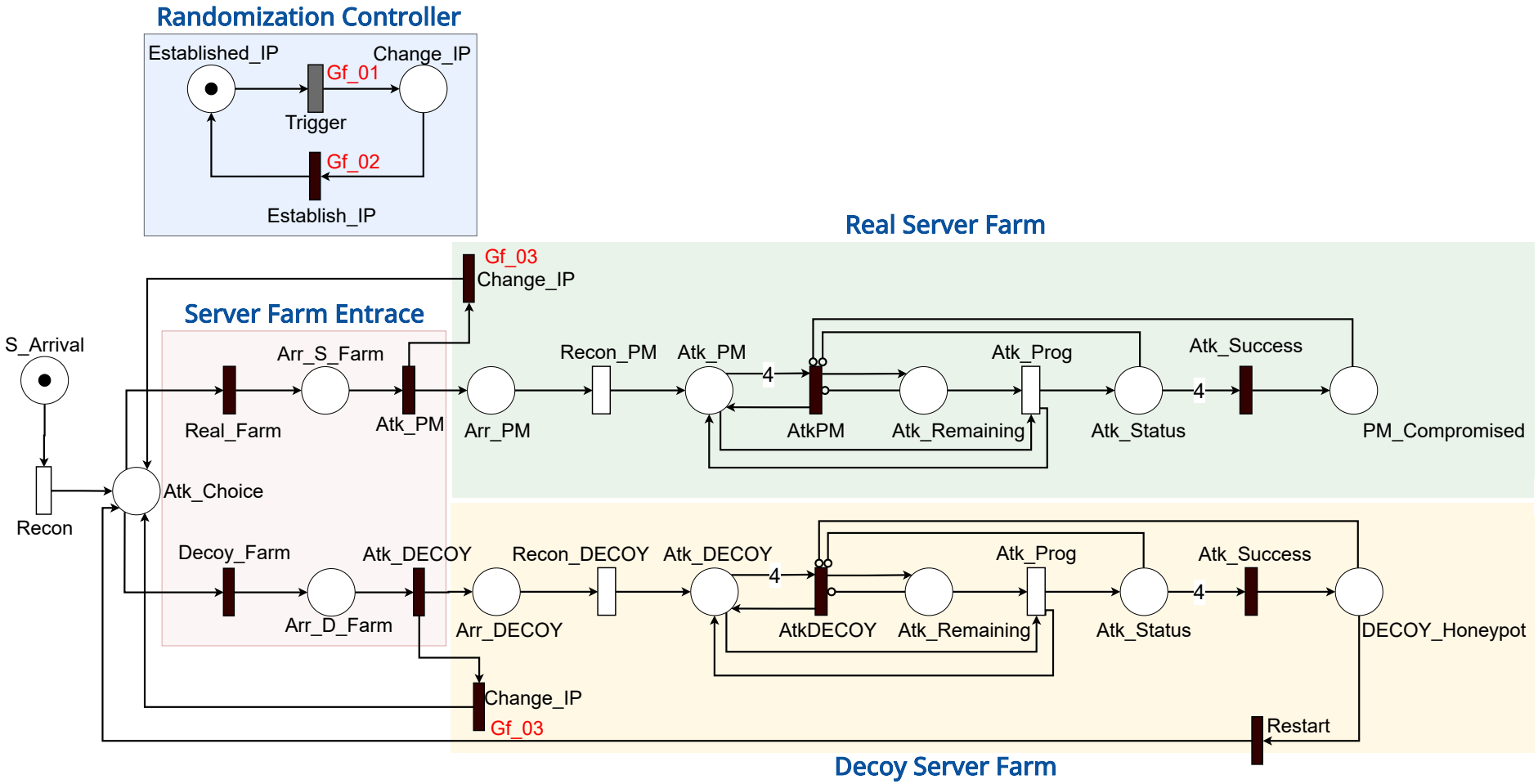


Figure 22 – SPN model for evaluating the probability of intrusion success with decoy servers

Table 6 – Main transitions of the SPN model

Transactions Type	Elements	Description	Value (min)
Timed	Trigger	Waiting time between IP change	50
	Recon, Recon_PM, Recon_DECOY	Time it takes for the intruder to make the recognition	30
	atkpm_prog	Time the attacker progresses in the attack	360
Immediate	Establish_IP	Reset the countdown for the next IP change	-
	Atk_PM, Atk_DECOY	Start of attack	-
	Real_Farm, Decoy_Farm	Choosing the target server set	-
	AtkPM, AtkDECOY	Start of attack on target server	-
	Atk_Success	Attack process completed successfully	-
	Change_IP	Start changing the servers' IP address	-
	Restart	Restart the intruder attack	-

Table 7 – Guard expressions of the SPN model

Transactions	Code	Expression
Trigger	Gf_01	(#PM1_Compromised=0)AND(#DECOY_Honeypot=0)
Establish_IP	Gf_02	(#start_attack=1)
Change_IP	Gf_03	(#change_IP=1)

5.3 Case Studies

This section presents the results of the simulations performed on the proposed model, focusing on analyzing the impact of variations in the number of real servers and decoys in the system. The objective is to evaluate how these variations affect the resilience of the system, seeking to minimize costs and side effects. The model and numerical evaluation were performed with the Mercury tool (MACIEL et al., 2017). The main metrics analyzed were the *Mean Time to Absorption* (MTTA) and the attacker's attack probability. The MTTA represents the average time until the system is not compromised, and it should be higher, indicating a greater capacity to delay the attack. The MTTA is obtained at the moment a token is placed in the PM_Compromised place. The attack probability was used to test the effectiveness of the defensive strategy in different scenarios.

Figures 23 and 24 show the analysis of MTTA as a function of the attack probability, considering scenarios with 1, 2, and 3 decoy servers. The probability of the attacker choosing a server was varied from 10% to 90%, with the graphs addressing the choice of real servers and decoys, respectively. In Figure 23, it can be seen that MTTA decreases as the probability of the attacker choosing a real server increases. Scenarios with more decoy servers present longer absorption times for low probabilities, but converge to similar

values when the probability of choosing a real server is high (90%). For example, MTTA varies from approximately 12,000 minutes to 2,000 minutes with 1 decoy and from 28,000 minutes to 2,000 minutes with 3 decoys. In Figure 24, which evaluates the MTTA as a function of the choice of decoy servers, the opposite behavior is observed: the MTTA increases as the probability of the attacker choosing a decoy increases. Scenarios with more decoys amplify this effect, with the MTTA reaching values of up to 28,000 minutes for 90% probability and 3 decoys, while scenarios with fewer decoys present shorter times, such as 10,000 minutes in the case of 1 decoy.

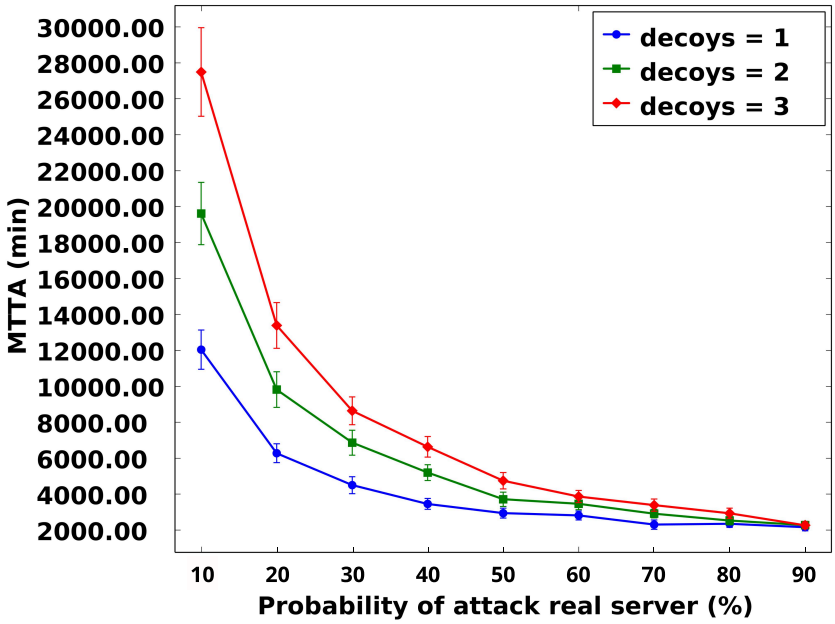


Figure 23 – MTTA varying the attack probability on a real server.

Figure 25 shows the graph of the CDF, which calculates the probability of success of an intrusion, considering different numbers of servers in the system. The absorption time, which represents the period required for the attacker to complete an intrusion, was used as the main metric: the longer the absorption time, the greater the effort required from the attacker. In this study, the probability of intrusion identification was kept constant at 90%, and the average interval to initiate a change of IP address of the servers (Trigger) was fixed at 360 minutes. It was observed that the strategy with only one server presented the shortest absorption time. In contrast, the variation with three decoy servers resulted in the longest absorption time, significantly prolonging the attacker's efforts. Comparatively, the absorption time gradually increased with the inclusion of more decoys: approximately 480 minutes (8 hours) with one server, 511 minutes (8.5 hours) with two servers, and 557 minutes (9.3 hours) with three. These results show that the inclusion of decoys improves the resilience of the system, increasing the difficulty for the attacker to complete the intrusion.

The results show that the choice between real and decoy servers influences the

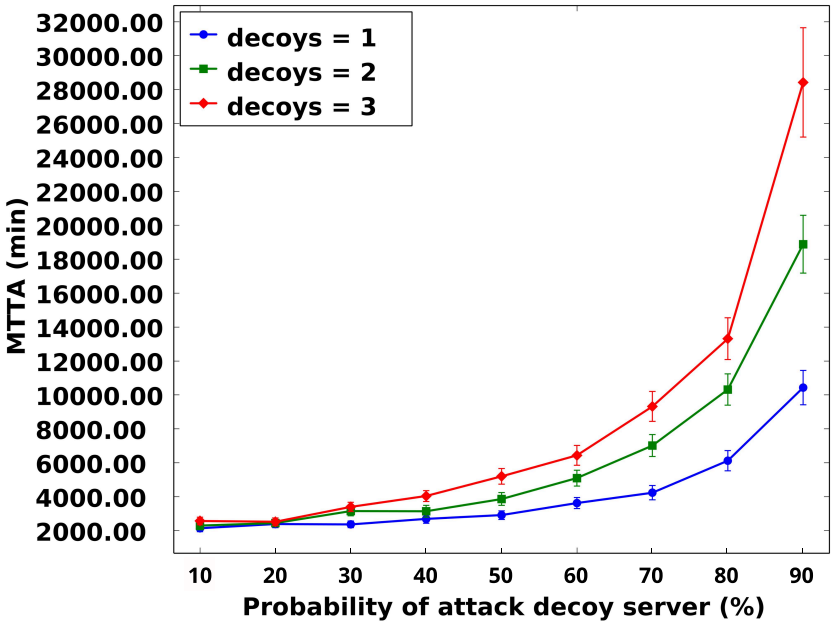


Figure 24 – MTTA varying the attack probability on a decoy server.

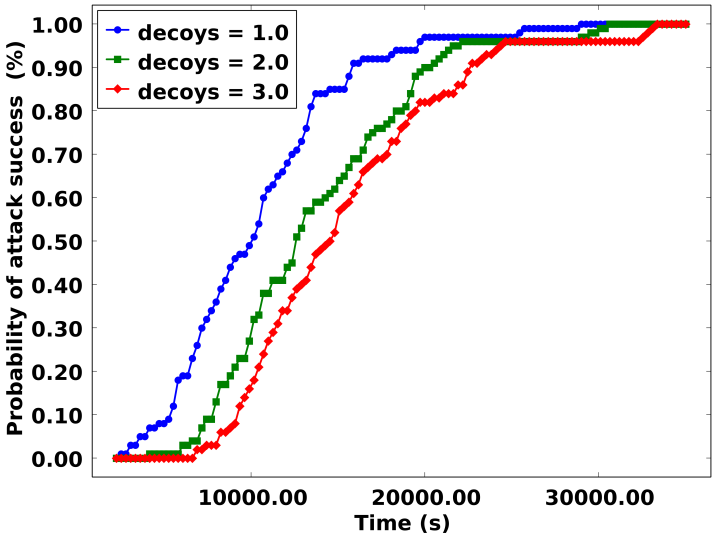


Figure 25 – Probability of successful intrusion varying number of decoys.

system absorption time. When the probability of the attacker choosing a real server increases, the MTTA decreases, indicating a faster compromise of the system. On the other hand, when choosing a decoy server, the absorption time increases, since this environment does not represent a real threat, confusing the attacker. Thus, the MTTA decreases as the probability of choosing a real server increases and increases with the probability of choosing a decoy server, delaying the compromise of the system and strengthening the defense with decoys.

5.4 Sensitivity Analysis

This section presents a sensitivity analysis to identify the most relevant factors in the proposed SPN model, using DoE. DoE is an efficient statistical technique to examine experiments and adjust input variables, evaluating their impact on output responses (FUKUDA et al., 2018). This approach enables understanding of system components, both individually and collectively, allowing for optimizations through scenario exploration and parameter performance evaluation, with a focus on enhancing the IP address change rate.

The DoE analysis in this work compares graphs of main factors (Figure 26) and interactions between factors (Figure 4b). The main factors graph uses horizontal bars to show the components with the most significant impact, ordered in descending order. The larger the bar, the greater the influence on system performance, helping to identify critical elements for optimization. The interaction graphs illustrate how two factors relate to each other, with parallel lines indicating no interaction and non-parallel lines suggesting interaction. The variation of the most impactful factors directly affects the IP address migration rate.

The experiment was conducted based on the architecture described in the previous sections, focusing on sensitivity analysis and its impact on the IP address change rate. This metric was chosen due to its direct relationship with the system absorption time, which affects an attacker's experience. Five factors were considered: (i) trigger, (ii) atkpm_prog, (iii) attack_probability, (iv) decoys, and (v) servers, evaluated at two levels: low and high. Table 8 presents the levels established for each factor considered in the experiment. Based on these values, a list containing 32 combinations was created, each representing a unique configuration of the factors analyzed. This list details the results obtained for each specific combination, allowing a clear view of the effects generated by the variations in the levels of each factor. Each line of this list corresponds to a result that would be achieved with the architecture of that specific configuration, providing an analysis of the impact that these combinations have on IP change. This approach allows us to evaluate how different combinations influence the behavior of the system.

Table 8 – Simulation factors and levels

Factor	**base value + 50%**	**base value - 50%**
trigger	50.0	500.0
atkpm_prog	180.0	540.0
attack_probability	0.1	0.9
decoys	1.0	3.0
servers	1.0	2.0

5.4.1 Numerical Analysis

Figure 26 shows the impact of the factors, highlighting the importance of each one in the system. The attack_probability factor, related to the activation time for scheduled migration, had the most significant impact identified. Next, atkpm_prog, which defines the time needed for an attacker to complete each phase of they attack, was the second most influential factor. The third relevant factor is decoys, which represent the number of fake servers intended to confuse attackers and increase the time of the attacks. The interaction between attack_probability and decoys also proved to be important, affecting the choice of the target server and influencing the IP address change rate. The graph also reveals other components and interactions that impact the system's IP change rate metric.

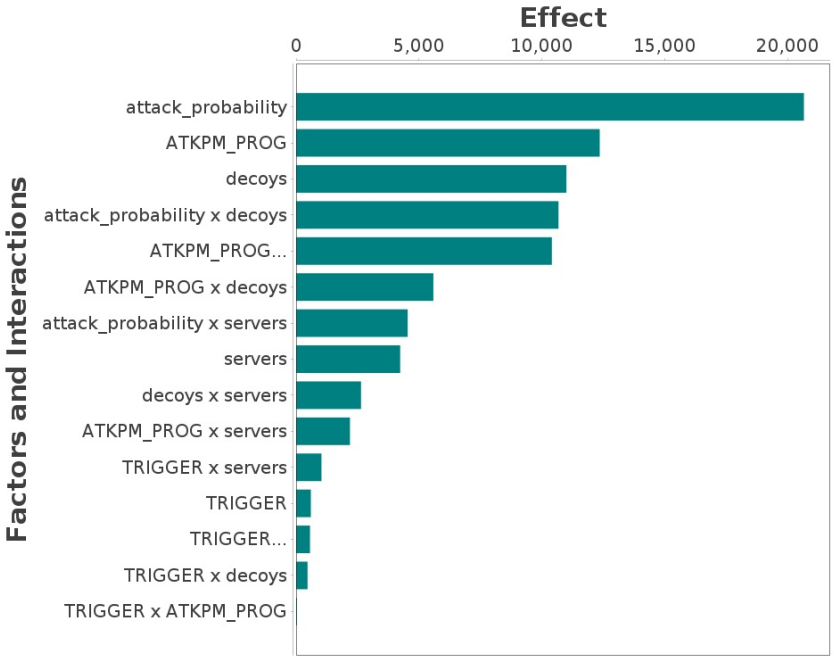


Figure 26 – Impact of factors and interactions.

Figures 27, 28, 29, and 30 illustrate how interactions between factors influence the system's IP address migration. Figure 27 shows the relationship between the attack progress time and the number of decoy servers. As the progress time increases, the IP migration rate also increases, highlighting the impact of this factor on system performance. Figure 28 analyzes the interaction between the number of real servers and decoys, indicating that a greater number of decoy servers directly impacts the IP migration rate, influenced by server availability. Figure 29 explores the interaction between attack probability and the number of decoy servers, showing that an increase in attack probability, combined with more decoy servers, results in more IP swapping, which increases the absorption time and makes the attack more difficult. Finally, Figure 30 examines the interaction between attack probability and the number of real servers, indicating that with higher probability and more real servers, the absorption time decreases, providing more time for the attacker

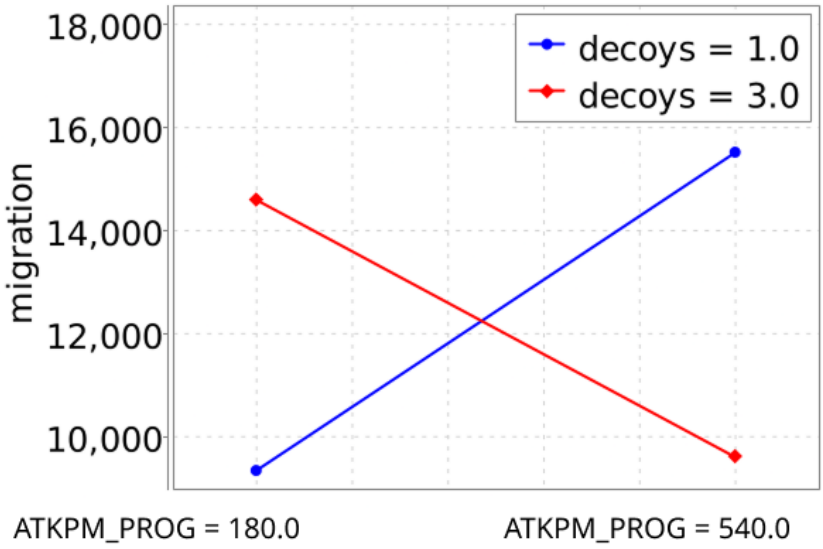


Figure 27 – Attack Progress X Decoy Servers.

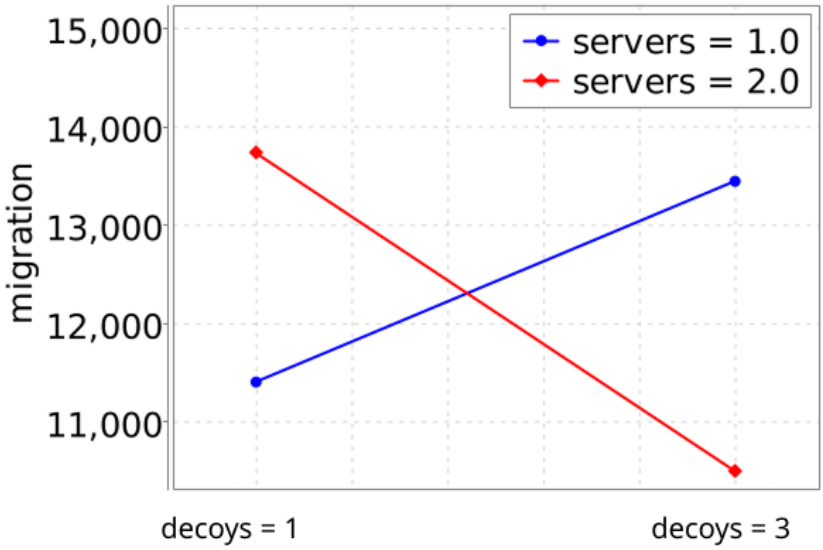


Figure 28 – Real Servers vs Decoy Servers.

to compromise the system and reducing the opportunities for IP swapping. These results highlight the complexity of the interactions between factors when absorption time is a critical parameter.

5.4.2 Discussions

This work proposed an MTD model focused on IP address swapping and the use of fake servers as the primary defense mechanisms. The study varied the number of decoy servers to evaluate their impact on delaying the advancement of invasions, identifying that this factor was the most determining factor in the effectiveness of the strategy. The comparison between different configurations indicated that increasing the number of decoy

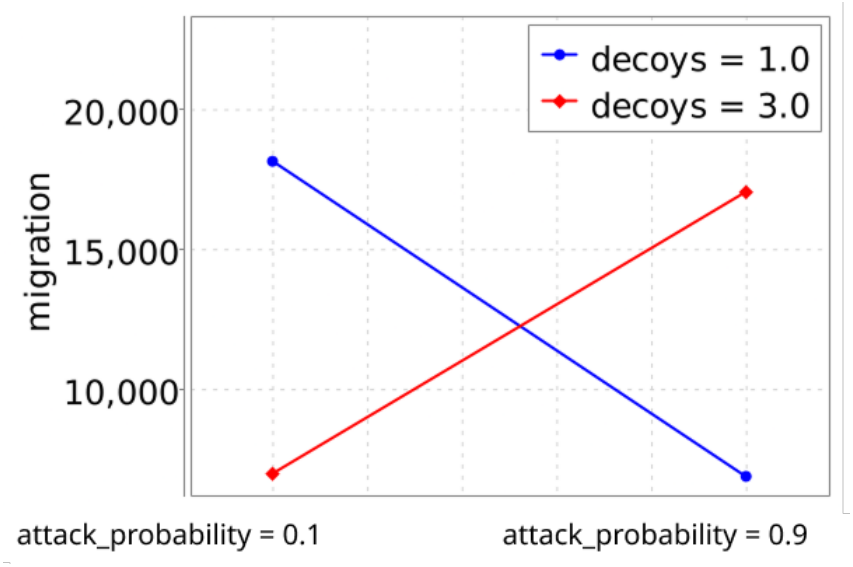


Figure 29 – Attack Probability X Decoy Servers.

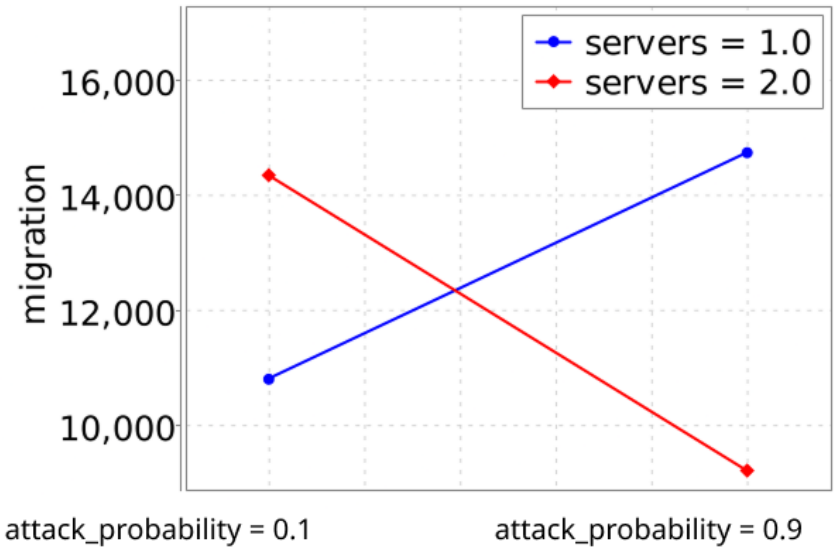


Figure 30 – Attack Probability X Real Servers.

servers significantly reduces the attacker's ability to progress, reinforcing the resilience of the system. In addition, it was observed that the balance between computational cost and system availability should be considered when sizing the number of decoy servers. As future work, we intend to expand the presented model, incorporating new defensive strategies in addition to IP swapping and the use of decoys. This includes the application of advanced monitoring and behavior analysis techniques to detect and respond to malicious activities quickly.

5.4.3 Findings

The findings of this study are summarized in the following points, which provide a comprehensive overview of the key insights derived from the research:

- **Factors with the greatest impact :** The results presented reveal that different factors play significant roles in the behavior of the MTD-based migration system. Figure 26 demonstrates the influence of the main parameters, highlighting the impact of the attack_probability factor, which regulates the activation time for the scheduled migration. This factor was identified as the most decisive in the model, evidencing its importance for the defense strategy. The second most relevant factor was atkpm_prog, which defines the time required for an attacker to complete each stage of his intrusion. This factor directly influences the time available for the system's defensive response. The third most important factor was the **decoys** variable, which represents the number of fake servers designed to deceive attackers and prolong the time of their attacks.

- **Interaction between factors:** The interaction between attack_probability and decoys significantly affects the choice of the attack target and the IP address change rate. This relationship reinforces the effectiveness of using decoy servers in increasing system resilience. Figure 27 illustrates the relationship between the attack progress time and the number of decoy servers, showing that a longer attack time is associated with a higher IP migration rate. Figure 28 highlights the relationship between the number of real servers and decoys, suggesting that the presence of more decoy servers contributes to an increase in the IP migration rate due to the distribution of the attack load.

- **Attack probability:** Figure 29 shows that the simultaneous increase in the attack probability and the number of decoy servers results in a greater exchange of IPs, making it more difficult for attackers to progress and extending the absorption time. Figure 30 suggests that a greater attack probability combined with a greater number of real servers reduces the absorption time, facilitating the action of attackers and reducing the opportunities for IP exchanges.

6 Conclusion

This dissertation proposed a comparative evaluation between two distinct MTD methodologies, each motivated by specific challenges in defending against cyberattacks. The first study was based on the need to improve the existing migration strategy, incorporating an IDS into the system to enable more dynamic and effective responses against attacks. In addition, a hybrid policy was introduced, combining time-based and event-based triggers, aiming to balance the security and performance of the system. This approach allowed for a more refined adaptation to threats, optimizing the timing of the migration and reducing the system's exposure to possible compromises. The second study emerged as a natural extension of the first, inspired by the results obtained and the search for new ways to strengthen the system's resilience. The central idea was to explore the use of decoy servers to deceive attackers, diverting their actions from legitimate targets and increasing the time needed to compromise the real infrastructure. This strategy was combined with the migration of IP addresses, making it difficult for attackers to map legitimate servers. In this way, the decoy-based approach complements migration by introducing an element of confusion, making the attack process more uncertain and less predictable.

In both studies, a Petri net-based model was developed to evaluate the system behavior and the effectiveness of the proposed approaches, allowing a detailed analysis of the impacts of the strategies on the security and performance of the defensive architecture. The findings of the first study demonstrate that the effectiveness of event-based detection policies is a crucial factor in increasing system security, especially when the IDS accuracy exceeds 50%. In addition, the hybrid policy, which combines time-based and event-based triggers, proved to be the most efficient in slowing down the progress of attackers, providing a defense balanced between security and performance. The influence of these policies on the system migration rate and MTTA reinforces the need for fine-tuning to optimize performance without compromising protection.

The second study highlighted that factors such as the probability of an attack, the attacker's progress time, and the number of decoy servers play a determining role in the behavior of IP migration. The interaction between these factors directly affects the resilience of the system, demonstrating that an increase in the number of decoy servers hinders the progress of attackers and prolongs the absorption time. On the other hand, a high number of real servers can reduce the opportunities for IP changes, making the system more vulnerable.

A case study was conducted to analyze the behavior of the architectures under different scenarios. In the IDS migration-based study, the first experiment evaluated the

MTTA and migration rate by varying the IDS detection probability. The second experiment analyzed the impact of the activation policy trigger time, while the third evaluated the probability of intrusion success. In the decoy server study, the first experiment analyzed the MTTA by varying the attack probability on a real server. The second experiment verified the MTTA by changing the attack probability on a decoy server. Finally, the third study evaluated the probability of intrusion success by varying the number of decoys. As future work, we plan to integrate event-based and hybrid policies into the decoy architecture for a full comparative assessment against migration-based strategies. Furthermore, we will investigate more complex VM migration scheduling approaches and analyze the MTD performance under attacks of varying durations to enhance system resilience.

References

AHMAD, R. W. et al. A survey on virtual machine migration and server consolidation techniques for cloud data centers. *Journal of Network and Computer Applications*, v. 52, p. 11–25, jun 2015. ISSN 10848045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804515000284>. 52

ANTONY, J. *Design of experiments for engineers and scientists*. [S.l.]: Elsevier, 2014. 16

BABADI, N.; DOUSTMOHAMMADI, A. A moving target defence approach for detecting deception attacks on cyber-physical systems. *Computers and Electrical Engineering*, Elsevier, v. 100, p. 107931, 2022. 21, 22

BRITO, C. et al. Stochastic Model Driven Performance and Availability Planning for a Mobile Edge Computing System. *Applied Sciences*, v. 11, n. 9, p. 4088, apr 2021. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/11/9/4088>. 4, 5

CAMPOLONGO, F.; TARANTOLA, S.; SALTELLI, A. Tackling quantitatively large dimensionality problems. *Computer Physics Communication*, Institute Jbr Systems, Informatics and Safety. Joint Research Centre of the European Commission, v. 117, n. 1, p. 75–85, 1999. 14

CAMPOLONGO, F. et al. *Sensitivity analysis in practice: a guide to assessing scientific models*. [S.l.]: John Wiley and Sons, 2004. 15

CARVALHO, D. et al. Mobile Edge Computing Performance Evaluation using Stochastic Petri Nets. In: . [s.n.], 2020. v. 2020-July, p. 1–6. ISSN 15301346. Disponível em: <https://ieeexplore.ieee.org/document/9219650/>. 4, 5

CHANG, X. et al. Job completion time under migration-based dynamic platform technique. *IEEE Transactions on Services Computing*, IEEE, v. 15, n. 3, p. 1345–1357, 2020. 6, 21, 23

CHEN, L.; HA, W. Reliability prediction and qos selection for web service composition. *International Journal of Computational Science and Engineering*, Inderscience Publishers (IEL), v. 16, n. 2, p. 202–211, 2018. 12

CHINNASAMY, P. et al. Efficient data security using hybrid cryptography on cloud computing. In: SPRINGER. *Inventive Communication and Computational Technologies: Proceedings of ICICCT 2020*. [S.l.], 2021. p. 537–547. 3

CHO, J.-H. et al. Toward proactive, adaptive defense: A survey on moving target defense. *IEEE Communications Surveys & Tutorials*, IEEE, v. 22, n. 1, p. 709–745, 2020. 4, 5, 12

CHOUDHARY, A. et al. A critical survey of live virtual machine migration techniques. *Journal of Cloud Computing*, v. 6, n. 1, p. 23, dec 2017. ISSN 2192-113X. Disponível em: <https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-017-0092-1>. 52

COSTA, I. et al. Availability evaluation and sensitivity analysis of a mobile backend-as-a-service platform. *Quality and Reliability Engineering International*, Wiley Online Library, v. 32, n. 7, p. 2191–2205, 2016. 16

CROUSE, M.; PROSSER, B.; FULP, E. W. Probabilistic performance analysis of moving target and deception reconnaissance defenses. In: *Proceedings of the Second ACM Workshop on Moving Target Defense*. [S.l.: s.n.], 2015. p. 21–29. 21, 23

FÉ, I. et al. Performance-Cost Trade-Off in Auto-Scaling Mechanisms for Cloud Computing. *Sensors*, v. 22, n. 3, p. 1221, feb 2022. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/22/3/1221>. 4, 5

FEITOSA, L. et al. Performance evaluation of message routing strategies in the internet of robotic things using the d/m/c/k/fcfs queuing network. *Electronics*, MDPI, v. 10, n. 21, p. 2626, 2021. 16

FUKUDA, I. M. et al. Design of experiments (doe) applied to pharmaceutical and analytical quality by design (qbd). *Brazilian journal of pharmaceutical sciences*, SciELO Brasil, v. 54, 2018. 43, 63

GE, M. et al. Proactive defense for internet-of-things: Integrating moving target defense with cyberdeception. *arXiv preprint arXiv:2005.04220*, 2020. 21, 24

GERMAN, R. *Performance analysis of communication systems - modelling with non-Markovian stochastic Petri nets*. [S.l.]: Wiley, 2000. (Wiley-Interscience series in systems and optimization). ISBN 978-0-471-49258-0. 13

GIRAULT, C.; VALK, R. *Petri nets for systems engineering: a guide to modeling, verification, and applications*. [S.l.]: Springer Science & Business Media, 2013. 14

GORRY, B.; IRELAND, A.; KING, P. PARTES: Performance Analysis of Real-Time Embedded Systems. In: *Fourth International Conference on the Quantitative Evaluation of Systems (QEST 2007)*. IEEE, 2007. p. 271–272. ISBN 0-7695-2883-X. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4338267>. 4, 5

HASSAN, W. et al. Cloud computing survey on services, enhancements and challenges in the era of machine learning and data science. *International Journal of Informatics and Communication Technology (IJ-ICT)*, v. 9, n. 2, p. 117–139, 2020. 3

HIJJI, M. et al. Cloud Servers: Resource Optimization Using Different Energy Saving Techniques. *Sensors*, v. 22, n. 21, p. 8384, nov 2022. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/22/21/8384>. 52

HOFFMAN, F.; GARDNER, R. *Evaluation of Uncertainties in Environmental Radiological Assessment Models*. [S.l.]: Radiological Assessments, 1983. 19

HU, Z.; ZHU, M.; LIU, P. Adaptive cyber defense against multi-stage attacks using learning-based pomdp. *ACM Transactions on Privacy and Security (TOPS)*, ACM New York, NY, USA, v. 24, n. 1, p. 1–25, 2020. 21, 24

KLEIJNEN, J. P. Sensitivity analysis and optimization in simulation: design of experiments and case studies. In: IEEE. *Winter Simulation Conference Proceedings, 1995.* [S.l.], 1995. p. 133–140. 16

LIANG, J.; KIM, Y. Evolution of firewalls: Toward securer network using next generation firewall. In: IEEE. *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. [S.l.], 2022. p. 0752–0759. 3

MACIEL, P. et al. Mercury: Performance and dependability evaluation of systems with exponential, expolynomial, and general distributions. In: IEEE. *2017 IEEE 22nd Pacific Rim international symposium on dependable computing (PRDC)*. [S.l.], 2017. p. 50–57. 60

MACIEL, P. R. M. *Performance, reliability, and availability evaluation of computational systems, Volume 2: Reliability, availability modeling, measuring, and data analysis*. [S.l.]: CRC Press, 2023. 13

MACIEL, P. R. M. *Performance, reliability, and availability evaluation of computational systems, volume I: performance and background*. [S.l.]: CRC Press, 2023. 13

MARSAN, M. A. et al. *Modelling with Generalized Stochastic Petri Nets*. USA: John Wiley & Sons, Inc., 1994. ISBN 0471930598. 13

MARSAN, M. A. et al. Modelling with generalized stochastic petri nets. *ACM SIGMETRICS performance evaluation review*, ACM New York, NY, USA, v. 26, n. 2, p. 2, 1998. 14

MARSAN, M. A.; CHIOLA, G. On petri nets with deterministic and exponentially distributed firing times. In: SPRINGER. *European Workshop on Applications and Theory in Petri Nets*. [S.l.], 1986. p. 132–145. 13

MENDONÇA, J. et al. Performance impact analysis of services under a time-based moving target defense mechanism. *The Journal of Defense Modeling and Simulation*, Sage Publications Sage UK: London, England, v. 20, n. 1, p. 41–56, 2023. 21, 22

MENDONÇA, J. et al. Security modeling and analysis of moving target defense in software defined networks. In: IEEE. *2022 IEEE 27th Pacific Rim International Symposium on Dependable Computing (PRDC)*. [S.l.], 2022. p. 141–151. 21, 24

MOODY, W. C.; HU, H.; APON, A. Defensive maneuver cyber platform modeling with stochastic petri nets. In: IEEE. *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*. [S.l.], 2014. p. 531–538. 21, 24

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, IEEE, v. 77, n. 4, p. 541–580, 1989. 13

NAFEA, R. A.; ALMAIAH, M. A. Cyber security threats in cloud: Literature review. In: IEEE. *2021 international conference on information technology (ICIT)*. [S.l.], 2021. p. 779–786. 3

NGUYEN, T. A. et al. Blockchain empowered federated learning with edge computing for digital twin systems in urban air mobility. In: LEE, S. et al. (Ed.). *The Proceedings of the 2021 Asia-Pacific International Symposium on Aerospace Technology (APISAT 2021), Volume 2*. Singapore: Springer Nature Singapore, 2023. p. 935–950. ISBN 978-981-19-2635-8. 4

NGUYEN, T. A. et al. Performability evaluation of switch-over moving target defence mechanisms in a software defined networking using stochastic reward nets. *Journal of Network and Computer Applications*, v. 199, p. 103267, 2022. Embora o ano de publicação na revista seja 2022, a sua dissertação cita como (NGUYEN et al., 2023), mantendo o rótulo para consistência no contexto do seu trabalho. 5

PAGNOTTA, G. et al. Dolos: A novel architecture for moving target defense. *IEEE Transactions on Information Forensics and Security*, Institute of Electrical and Electronics Engineers (IEEE), v. 18, p. 5890–5905, 2023. ISSN 1556-6021. Disponível em: <http://dx.doi.org/10.1109/TIFS.2023.3318964>. 21

PIANOSI, F. et al. Sensitivity analysis of environmental models: A systematic review with practical workflow. *Environmental Modelling and Software*, v. 79, p. 214–232, 2016. 15

REQUENO, J. I.; MERSEGUER, J.; BERNARDI, S. Performance analysis of Apache Storm applications using stochastic petri nets. In: *Proceedings - 2017 IEEE International Conference on Information Reuse and Integration, IRI 2017*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2017. v. 2017-Janua, p. 411–418. ISBN 9781538615621. 4, 5

RODRIGUES, L. et al. Modelo estocástico para avaliaçao de disponibilidade de hospitais inteligentes. In: SBC. *Anais do XIX Workshop em Desempenho de Sistemas Computacionais e de Comunicação*. [S.l.], 2020. p. 145–156. 13

SANTOS, L. et al. Data processing on edge and cloud: a performability evaluation and sensitivity analysis. *Journal of Network and Systems Management*, Springer, v. 29, n. 3, p. 27, 2021. 15, 16

SILVA, F. A.; FÉ, I.; GONÇALVES, G. Stochastic models for performance and cost analysis of a hybrid cloud and fog architecture. *Journal of Supercomputing*, Springer, v. 77, n. 2, p. 1537–1561, may 2021. ISSN 15730484. Disponível em: <https://link.springer.com/article/10.1007/s11227-020-03310-1>. 4, 5

SILVA, F. A. et al. Mobile cloud performance evaluation using stochastic models. *IEEE Transactions on Mobile Computing*, v. 17, n. 5, p. 1134–1147, 2018. 14

SUN, J.; SUN, K. Desir: Decoy-enhanced seamless ip randomization. In: IEEE. *IEEE INFOCOM 2016-the 35th annual IEEE international conference on computer communications*. [S.l.], 2016. p. 1–9. 55

TAN, J. et al. A survey: When moving target defense meets game theory. *Computer Science Review*, Elsevier, v. 48, p. 100544, 2023. 3, 11

THAKKAR, A.; LOHIYA, R. A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions. *Artificial Intelligence Review*, Springer, v. 55, n. 1, p. 453–563, 2022. 3

TORQUATO, M.; MACIEL, P.; VIEIRA, M. Security and availability modeling of vm migration as moving target defense. In: IEEE. *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*. [S.l.], 2020. p. 50–59. 37

TORQUATO, M.; MACIEL, P.; VIEIRA, M. Analysis of vm migration scheduling as moving target defense against insider attacks. In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. [S.l.: s.n.], 2021. p. 194–202. 5, 6, 21, 23, 27, 30, 31, 33, 55, 56, 58

TORQUATO, M.; VIEIRA, M. Moving target defense in cloud computing: A systematic mapping study. *Computers & Security*, Elsevier, v. 92, p. 101742, 2020. 11

TUFAIL, S. et al. A survey on cybersecurity challenges, detection, and mitigation techniques for the smart grid. *Energies*, MDPI, v. 14, n. 18, p. 5894, 2021. 3

WANG, S. et al. Probabilistic models for evaluating network edge's resistance against scan and foothold attack. *IET Communications*, Wiley Online Library, 2024. 21, 23