Effective Security Analysis for Combinations of MTD Techniques on Cloud Computing

Hooman Alavizadeh

A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy (Ph.D.) in Computer Science,
Massey University, 2019.

Abstract

Moving Target Defense (MTD) is an emerging security mechanism that can introduce a dynamic defensive layer for a given system by changing the attack surface. MTD techniques are useful to address security issues in cloud computing.

MTD techniques are classified into three main categories: Shuffle, Diversity, and Redundancy. Shuffle MTD techniques can rearrange the system's components (e.g., IP mutation). They confuse the attackers by hardening the reconnaissance process and wasting the information collected by the attackers. Diversity MTD techniques change the variants of a system's component (e.g., operating systems), which makes an attack more difficult and costly because the attackers encounter a new set of vulnerabilities. Redundancy MTD techniques increase the system components' replicas. They can be used to increase system dependability (e.g., reliability or availability) by providing redundant ways of providing the same services when some system components are compromised. Since deploying each MTD technique may affect the others and also have different effects on the system (e.g., one can enhance the security and another can provide service's availability), it is important to combine MTD techniques in such a way that they can support each other directly or indirectly.

This research first conducts an extensive survey of MTD literature to realize and summarize the key limitations of the current MTD studies. We reveal that (i) there is a lack of investigation on the combination of MTD techniques, (ii) relatively less effort has been made to evaluate the effectiveness of MTD techniques using security analysis, and (iii) there is a shortcoming in the validation of MTD techniques on more realistic cloud testbeds.

We focus on the theoretical aspects of combining MTD techniques and provide formalization to combine MTD techniques in order to address those limitations. First, we achieve this by combining Shuffle and Redundancy to find a trade-off between System Risk and Reliability. Then, we provide a formal mathematical definition to combine Shuffle and Diversity to increase security while narrowing the scope for potential attacks. We evaluate the effectiveness of the proposed combined techniques using Graphical Security Models (GSMs) and incorporating various security metrics.

We extend the combination of MTD techniques by including Redundancy besides Shuffle and Diversity. We perform an in-depth analysis on combining those MTD techniques to find out a trade-off between security alongside the reliability of the cloud. We show that if those MTD techniques are combined properly, it not only improves the cloud's security posture but also it increases the reliability of the cloud. Moreover, we study the economic metrics to show how MTD techniques can be deployed in a cost effective way. We also propose an Optimal Diversity Assignment Problem (*O-DAP*) to find the optimal solution for deploying Diversity over cloud.

Finally, we design and develop an automated cloud security framework to evaluate the cloud security posture and adapt MTD techniques on the real cloud platform. We demonstrate the feasibility, adaptability, and usability of implementing MTD techniques on UniteCloud which is a real private cloud platform.

In memory of my father

Acknowledgements

I would like to express my great appreciation to everyone who have helped me throughout the completion of my Doctoral thesis, writing publications, and anyone who has accompanied me during my PhD journey – my apologies to anyone whom I failed to mention.

I specially owe gratitude to my research supervisors Associate Professor Julian Jang-Jaccard and Associate Professor Dongseong Kim, who have guided me through my Doctoral research. I am quite sure that this PhD dissertation would have not been possible and fruitful without their continued support, extensive expertise and knowledge. To Julian, you have been more than a supervisor for me; you have showed me compassion during difficult times of my PhD journey and helped me to overcome the challenges. Thank you for sharing your knowledge, invaluable guidance and relentless support, you have been a permanent source of enthusiasm for my research progress, I will forever remain thankful for that. To Dongseong, I owe a very special gratitude to you for giving me the chance to work with you and learn from you, thank you for sharing your knowledge, showing me the way to start, grow and learn, and conduct a fruitful research – thank you for being a friend. I will always remain indebted to you for indelible impact you have made on my growth and success. To both of you, I was immensely lucky to have you as my PhD supervisors.

I would like to extend my sincere thanks to my co-supervisor Professor Hans Guesgen for his kind support and helpful advice during the time when I was studying in Palmerston North and afterward.

I am tremendously grateful to Dr Samin Aref not only for great collaboration, but also for being a role model, a friend, a person who has helped me during the tough times in my life, given me the encouragement for making right decisions in my critical moments of my PhD journey. I am also very grateful to my previous supervisor Dr Mark Wilson for his utmost supports, sharing knowledge, and all he has taught me.

A very special thank you to Dr Jin B. Hong for his collaboration, sharing knowledge, comments and advice, and helpful contributions. Also, I would like to express my gratitude to Dr Simon Yusuf and Dr Mengmeng Ge for helping me to address my concerns regarding challenges and questions I encountered during my research.

I am also grateful to my previous supervisor Dr Kuda Dube for his valuable and constructive suggestions when I was studying in Palmerston North, and also for all his help and support before I move to New Zealand and start my PhD.

I wish to express my appreciation to Associate Professor Winston Sweatman and, my friend, Mohsen Hashemi for their insightful advice for my mathematical and notation related concerns. I am also thankful to Tony Shi for his help and support to address the problems and concerns regarding the UniteCloud.

I also would like to gratefully acknowledge the School of Natural and Computational Sciences at Massey University for supporting my research project and providing me with three years' financial support.

I would like to express my great appreciation to the people whom I have made collaboration and friendship in the School of Natural and Computational Sciences and cyber security lab in the University of Canterbury: Sibghat, Dilli, Yuanyuan, Mehmood, Azadeh, Niloofar, Rahila, Jinting.

Last but not the least, I would like to give a big thanks to my family, specially my mother for her enduring love and faithful support in all my pursuits, I am forever grateful to her. I owe a very special thanks to my wife – someone with whom I shared the rest of my life – thanks for her continued love and support – this thesis would have not be completed without her patient and sacrifices. Also, a gratitude to my brother, whom I have shared so many points of interests in computer science and so many similar objectives about what a perfect work should be about.

Publications Arising from this Thesis

The core of this thesis was based on the following peer-reviewed journals and conferences.

- Alavizadeh, H., Kim, D. S., Hong, J. B., and Jang-Jaccard, J. Effective security analysis for combinations of MTD techniques on cloud computing (short paper). In *International Conference on Information Security Practice and Experience (ISPEC)* (2017), Springer, pp. 539-548.
- Alavizadeh, H., Jang-Jaccard, J., and Kim, D. S. Evaluation for combination of Shuffle and Diversity on Moving Target Defense strategy for cloud computing. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* (2018), IEEE, pp. 573-578.
- Alavizadeh, H., Hong, J. B., Jang-Jaccard, J., and Kim, D. S. Comprehensive security assessment of combined MTD techniques for the cloud. *In Proceedings of the 5th ACM Workshop on Moving Target Defense* (2018), ACM, pp. 11-20.
- Alavizadeh, H., Alavizadeh, H., Kim, D. S., Jang-Jaccard, J., and Niazi Torshiz, M. An automated security analysis framework and implementation for MTD in cloud. In *International Conference on Information Security and Cryptology (ICISC)* (2019), Springer.
- Alavizadeh, H., Hong, J. B., Kim, D. S., and Jang-Jaccard, J. Evaluating the Effectiveness of Shuffle and Redundancy MTD Techniques in the Cloud. Submitted to *IEEE Transactions on Dependable and Secure Computing (IEEE TDSC) (Under review).*
- Alavizadeh, H., Kim, D. S., and Jang-Jaccard, J. Model-based Evaluation of Combinations of Shuffle and Diversity MTD Techniques on the Cloud. *Future Generation Computing Systems (FGCS).*
- Alavizadeh, H., Aref, S., Kim, D. S., and Jang-Jaccard, J. Security and Economic Modeling and Analysis of MTD techniques for Cloud Computing. Manuscript is ready for submission to *IEEE Transactions on Emerging Topics in Computing (IEEE TETC).*

- Jin-Hee, C, Sharma, D. p., Alavizadeh, H., Ben-Asher, N, Yoon, S, Moore, T. J., Kim, D. S., Lim, H., Free-Nelson, F. Toward Proactive, Adaptive Defense: A Survey on Moving Target Defense. Revised version submitted to *IEEE Communications Surveys and Tutorials (Under review).*

Contents

List of Figures

List of Tables

List of Acronyms

AC	Attack Cost
ACL	Access Control List
AG	Attack Graph
ALE	Annual Loss Expectancy
AP	Attack Path
ARO	Annualized Rate of Occurrence
ASP	Attack Success Probability
AT	Attack Tree
AV	Asset Value
BS	Benefits of Security
BVS	Betweenness VM Selection
CS	Cost of Security
CVE	Common Vulnerability and Exposures
CVS	Closeness VM Selection
CVSS	Common Vulnerability and Scoring System
DDoS	Distributed Denial of Service
DMZ	Demilitarized Zone
EF	Exposure Factor
ENB	Expected Net Benefit
ES	Exhaustive Search
FDD	Firewall Derision Diagram
GSM	Graphical Security Model

HARM	Hierarchical Attack Representation Model
IC	Informative Call
IDS	Intrusion Detection System
IM	Important Measure
MAPL	Mean of Attack Path Length
ML	Machine Learning
MoPL	Model of Attack Path Length
MTD	Moving Target Defense
NCM	Network Centrality Measure
OC	Operational call
O-DAP	Optimal Diversity Assignment Problem
OS	Operating System
PHC	Personal Health Cloud
PHI	Personal Health Information
PN	Protected Network
RoA	Return on Attack
RoSI	Return on Security Investment
RVS	Random VM Selection
RZ	Resource Zone
SAP	Shortest Attack Path Length
SDN	Software Defined Networking
SDPL	Standard Deviation of Attack Path Length
SLE	Single Loss Expectancy
VM	Virtual Machine
VM-LM	Virtual Machine Live Migration
VMM	Virtual Machine Monitoring

Chapter 1

Preface

Cloud computing is an on-demand network paradigm that facilitates access to a large pool of computing resources [105]. Cloud computing has attracted widespread attention in the domain of information technology (IT) and industries due to economic benefits such as reduction of capital and operational expenditure [125, 142]. Despite these multifaceted benefits, security remains one of the fundamental concerns related to cloud computing [15, 163]. For this reason, the majority of cloud's clients are unable to trust this technology [17]. The lack of sufficient trust and loss of certain security controls engaging with cloud computing make this transition difficult. Although many reactive security mechanisms have been proposed and used to protect the cloud against cyber-attacks such as virtual machine isolation, firewalls, Intrusion Detection Systems (IDS), anti-malware and so forth, cyber criminals are still able to utilize novel techniques in order to exploit the vulnerabilities and compromise security and privacy of the cloud.

Recently, Moving Target Defense (MTD) techniques have been proposed in extant literature and used extensively in the realm of cyber security. MTD can introduce a dynamic defensive layer to a system aiming to protect the system by changing the attack surface. The underlying idea behind the MTD strategies is that it is not always possible to secure a given system perfectly. To this end, the MTD strategies try to utilize normal functioning of the system even in the presence of attackers trying to penetrate the system. Since the perfect prevention of attacks is impossible, the main endeavor of MTD is to introduce ambiguity and difficulties for the attacker to mitigate the attack chance rate, thereby thwarting the potential attacks. In practice, MTD can be introduced by modifying and controlling the attack surface through manipulation of the system's configurations so that the changes transpiring in the system's components and configuration confound the attackers targeting the system. Deploying successful MTD strategies may exacerbate uncertainty and complexity for the attackers, thus making it increasingly difficult for the attacker to identify targets (e.g., vulnerable system components). It also may introduce higher costs in launching attacks in terms of time, resource, and monetary costs. As a consequence, the attacker will waste considerable time and effort without being able to gain useful/actionable intelligence about the system [66,114]. MTD techniques are mainly divided into three categories: Shuffle (rearranging a system's component), Diversity (using various variants for a system's component), and Redundancy (increasing a system's component replicas) [66].

Although MTD techniques such as Shuffle, Diversity, and Redundancy have been widely studied in the literature, there is still a lack in the current studies to combine those techniques. It is of paramount importance to combine MTD techniques in such a way that they can support each other directly or indirectly. Firstly, Shuffle MTD technique can change and rearrange the system's components. This introduces confusion to the attackers by hardening the reconnaissance process and making the collected information of the attackers obsolete and useless. Secondly, Diversity are usually deployed to enhance security by changing the system's components (i.e., software, OS, services, *etc.* [51, 54, 146]). This makes a system more robust and resilience in the occurrence of the attacks as the attackers need to encounter a new set of vulnerabilities. Thus, Shuffle MTD technique can support Diversity because the way in which system's components are rearranged may affect the degree of Diversity and vise versa. However, in order to combine those techniques to benefit from the advantages of each, it is important to know how much Shuffle can enhance security, and similarly, how Diversity can increase attacker's difficulties to deal with new system in terms of time, effort, costs, and so forth. Unlike Shuffle and Diversity MTD techniques, Redundancy is a non-trivial technique to increase service reliability and availability for users. This technique is often measured by some concept of system dependability (e.g., reliability). Redundancy MTD can be well-mingled with Shuffle and Diversity aiming to increasing both security and service reliability.

One can benefit from using a MTD technique solely, but the problem arises when a trade-off between security and dependability such as service availability or reliability is required. Deploying each MTD technique can affect the others. For instance, the service reliability or availability of a system can be easily interrupted by performing an unreasonable Shuffle or Diversity leading to a service interruption or even a deterioration of overall security. Moreover, deploying Redundancy may incur extra cost for the system as it creates additional replicas of the system components (i.e., servers or paths). Furthermore, if Redundancy is not properly deployed, it gives the attacker more chance to attack the system due to larger attack surface. It is important to extensively investigate the way in which MTD techniques can be combined to optimally meet these multiple objectives such as increasing benefits and reducing the undesirable effects. However, security modeling and analysis such as Graphical Security Models (GSMs) have been used in the literature to evaluate the effectiveness of MTD techniques. Many security metrics can be incorporated with the proposed GSMs to analyze the security [157].

MTD can be used to secure various application domains such as Internet of Things (IoT), software defined networks (SDNs), and so forth. To the best of our knowledge, there is no public or private cloud that has adapted moving target defenses on their platform as a defensive mechanism. Based on the aforementioned problems, we provide the research questions we aim to address throughout this thesis.

- Q1: How MTD techniques can be combined to enhance the effectiveness of the proposed technique on cloud?
- Q2: Which security metrics should be used to evaluate the effectiveness of combined MTD techniques?
- Q3: How we can adapt the effective MTD technique on a real cloud environment?

The main goals of this research is to evaluate the combinations of MTD techniques and adapt/implement the MTD techniques on cloud computing. We define following three sub-goals corresponding to the research questions:

- G1: To focus on the theoretical aspect of combined MTD techniques on cloud: The outcomes of this goal is an extensive survey on MTD techniques focusing on the way in which MTD techniques can be combined alongside providing distinctive formalism to model three combination of MTD techniques including Shuffle+Redundancy, Shuffle+Diversity, Shuffle+Diversity+Redundancy for cloud.
- G2: To evaluate different proposed combinations of MTD techniques for cloud through simulation: The outcome of this goal is using GSMs and various security metrics to evaluate the effectiveness of the combined MTD techniques find more promising strategies in terms of trade-offs among security, dependability, and economic metrics.
- G3: To investigate on the practical side of the research to demonstrate the feasibility, adaptability, and usability of those MTD techniques for a real cloud testbed: The outcome of achieving this goal consist of developing a cloud security framework which is able to 1) automatically assess the security posture of a private cloud using GSMs and security metrics, 2) finding the most promising defensive MTD techniques based on the security level required by the cloud provider, 3) adapt and deploy the MTD techniques on UniteCloud [1] which is a real private cloud platform.

The organization of this thesis is as follows. Chapter 2 in this thesis presents a comprehensive survey on MTD studies. We review the proposed MTD studies to understand what aspects lack in the MTD literature. The scope of survey covers three main MTD categories and implementation layers which have been used in this thesis.

In Chapter 3, we formulate and evaluate the combination of Shuffle and Redundancy MTD techniques to find a trade-off between System Risk (Risk) and Reliability (The earlier version of this work has been published in [13]). We use a scalable GSM named

HARM to evaluate the security metrics and use Network Centrality Measures (NCMs) to effectively combine MTD techniques. We conduct a correlation analysis to find out a relation between the deploying Shuffle and Diversity and the used metrics.

In Chapter 4, we present an extensive evaluation on combinations of Shuffle and Diversity MTD techniques to harden the attack success and reduce the system Risk (The earlier version of this work has been published in [12]). We formulate the combination of Shuffle and Diversity in which it can be able to capture different combinations strategies for those techniques to provide the promising results. We utilize eight security metrics to assess the effectiveness of the combination of Shuffle and Diversity (The extended version of this work has been published in [14]).

In Chapter 5, we present a comprehensive combination of MTD techniques including Shuffle, Diversity, and Redundancy together to analyze the effectiveness of MTD techniques based on both attacker and cloud provider's perspective (The work has been published in [11]). We reformulate our Diversity formalism so that it can capture deploying Diversity on multiple VMs to obtain better results. We utilize two Attack Cost (AC) and Return on Attack (RoA) metrics to assess security from the attacker point of view, and also used *Risk* and *Reliability* to show the security and dependability of the system from the cloud provider's perspective. We extend the work by including a more specific context by modeling an E-health cloud to investigate on economic-based evaluation for MTD techniques and Diversity Assignment Problem through optimization We formulate Diversity as a mathematical model considering graph coloring to harden the attack. We use a binary linear optimization model to solve the problem and find the optimized result.

In Chapter 6, We demonstrate the practicality of implementation and adaptation of MTD techniques on a read cloud environment. We investigate on feasibility of adaptation of MTD techniques on a real cloud platform. We evaluate the implemented MTD techniques using real cloud measurements. We utilize API programming allowing us to automate the cloud security framework and MTD deployment (The work has been published in [10]).

In Chapter 7, we conclude this thesis and introduce some future research directions.

Chapter 2

A Survey on MTD techniques

2.1 Introduction

Moving Target Defense (MTD) is an emerging proactive approach which provides an asynchronous defensive mechanism over a system aiming to increase the attack difficulties by changing the attack surface. The basic assumption of MTD is that it very difficult (even impossible) to provide complete and perfect security for a system. Since the perfect security of a given system cannot be always provided, the main idea behind MTD is to provide an adequate level of dynamicity for a system so that it can introduce uncertainty and unpredictability for attackers. MTD can decrease the attacker's chance to identify targets such as vulnerable system components or impose higher costs in launching attacks. The desired goal of MTD techniques is that the attacker spend more time and effort, while having minimum chance to penetrate to a system [114].

The concept of MTD technique goes back to the 1970's, when those concepts introduced under different names in the literature such as reconfigurable computing [20, 40], fault tolerance [18, 34] which introduced the n-version programming (NVP) concept, or bio-inspired cybersecurity [86, 94]), and so forth. Various defensive techniques have been proposed under the name of MTD and rapidly developed over the recent years due to its promising capability and merit.

However, there has been a lack on current MTD literature in terms of classification and deployment layers of MTD techniques, evaluation and effectiveness of the proposed techniques (e.g., using security modeling and analysis), combining MTD techniques, and implementing MTD techniques on real testbeds such as cloud computing.

In this chapter, we define a comprehensive survey for the existing state-of-art MTD techniques with an aim to find the important limitations of the current MTD proposals. At the end of this chapter, we highlight the main shortcoming of the current MTD research.

Table 2.1: MTD Techniques and Deployment Layers

Deployment Layer	**Techniques**		
	Diversity (D)	**Redundancy (R)**	**Shuffle (S)**
Application	Web [36,71,145] App [19,36,71] Database [36,145] Other [70]	Web [56,154] App Database Other	TCP/UDP Port [16,82,100] HTTP/HTML [78,151] Apps Migration [116]
Host (OS level) **VM (Instance)**	Windows Linux [71] Solaris [71] Other	Host/VM [98,120]	IP Address [78] Virtual IP [16,75,82] VM migration [55,120,164]
Hypervisor (VMM)	Xen VMware ESXi Other	Hypervisor Replica	Failover Switchover
Hardware	Intel HP Sun Solaris Other	Backup/Replica	Hardware Replace

2.2 Defining MTD Framework

We survey the existing literature in terms of (1) classification of MTD techniques, (2) MTD deployment Layers, (3) combination of MTD Techniques at different Layers, as shown in Table 2.1. Moreover, we survey the current MTD studies extensively in terms of application domain which includes Cloud computing, Internet of Things (IoT), Software Defined Networking (SDN), Enterprise Networks. Furthermore, we survey the MTD techniques based on the implementation, validation, and evaluation.

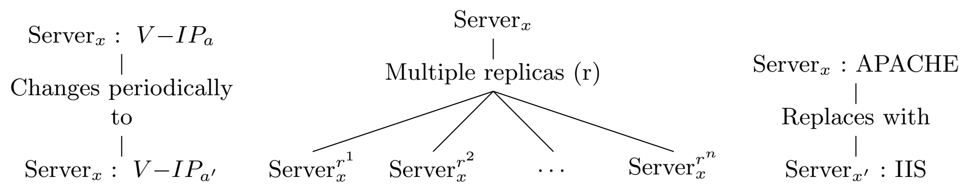
Generally, Hong *et al.* [66] classified the MTD strategies in three comprehensive operational categories as follows:

- Shuffle: This technique are defined as any method which can re-arrange the system's configuration such as IP mutations os changing, Host randomization, network's topology rearrangement, Virtual Machine (VM) migration, and so on [45,75,116]. The main idea behind Shuffle techniques is to thwart potential attacks by increasing the uncertainties and confusion for cyber attackers. This may waste the information collected by the attackers, and also can harden the reconnaissance and identification process for the attackers.

- Diversity: This technique can be considered as the replacing the system's component with the alternative variant or implementation (e.g, a server, programming language, Operating System, hardware, *etc.*), while the system provides equivalent functionality with the previous state (before changing variant) [74,110,126]. Diversity MTD technique has a twofold advantage, increasing the system's resiliency and fault-tolerance in presence of attack on a system, and making difficulties for the attackers to deal with a new variant in terms of spending time and effort to learn the new system and penetration ways (e.g., exploiting vulnerabilities).

- Redundancy: This technique are defined as the redundant ways of providing services in a system, such as increasing the number of components which includes replicas for servers, hardware, OS, software, services, and so forth [85, 154]. This includes providing multiple replicas for a systems' component such as having multiple VMs in a cloud, or multiple servers in a network. Redundancy can be used on a system to enhance service reliability and availability so that if a single or multiple components of a system are compromised, redundant variants can preserve the system reliability and availability in an acceptable threshold.

Figure 2.1 demonstrates three examples of MTD deployment categories Shuffle, Diversity, and Redundancy on a web server (denoted as Server$_x$ in Figure 2.1). Moreover, the MTD techniques can be applied on different systems and contexts (such as IoT, Cloud, Enterprises) and various components (e.g, servers, VMs, hosts, programming languages, *etc.*) and also different layers (like Hardware, Application, Network, and so forth).



(a) Shuffle (b) Redundancy (c) Diversity

Figure 2.1: An example of deploying MTD technique on a web server

2.2.1 Shuffle

In this section, we review existing MTD techniques categorized as Shuffle. Later on, we discuss the limitation of Shuffle technique found from the literature and the way that those limitations can be addressed.

Network, Port, and IP Shuffling: Shuffle can be deployed on various network using Port Hopping techniques [100]. Typically, these techniques can dynamically map a specific port to another unused port which is randomly selected. Luo *et al.* [100] evaluated the effectiveness of the port hopping techniques using hiding service identities aiming to increasing reconnaissance effort by the attacker.They evaluated their method using attack success probability with a large size of a port pool.

Deploying Shuffle technique through changing IP address and host randomization approaches has been proposed in many studies. Jafarian *et al.* [75] implemented an IP shuffle technique by mutating the IP addresses unpredictably while minimizing overhead of the operation. They used OpenFlow to frequently reassign the Virtual IP addresses which were translated from and to a real host's IP address. Their method is effective in terms of low operational overhead and can effectively defends against malicious scanning tools and worms. Sharma *et al.* [136] proposed an MTD technique using an IP shuffling approach which used IP multiplexing (or demultiplexing) for an SDN environment. In [101], the authors proposed a host IP mutation technique to defend against potential attacks in large networks using SDN controller. Another IP randomization technique is proposed by Antonatos *et al.* in [16]. The authors used IP randomization as an MTD technique aiming to thwart Hit-List worms attacks. The proposed technique increases the difficulties of information gathering process by the attackers aiming to find the vulnerable targets. However, they only assessed the performance and did not carry out the security assessment and analysis.

Carroll et al. [30] proposed a network address shuffling technique as an MTD technique. They evaluated the performance of their method using probabilistic models in terms of the number of possible IPs scanned by the attackers, the size of a network, the number of vulnerabilities and the IP shuffle technique frequency.

Changing the network topology has been proposed in many studies as a Shuffle MTD technique. Changing routes in a network can be a useful defensive technique aiming to invalidate the attacker's information about the attack paths in the network. For instance, Achleitner *et al.* [3,4] proposed a network routes shuffling technique using virtual topology generation which can defend against scanning attack. In [68], the authors proposed an optimal network reconfiguration technique using shuffle assignment problem for SDN environments. They showed their method can enhance the network security through changing the routes.

VM Migration: Virtual Machine migration can be used as Shuffle MTD technique to secure the virtualization environment. VM migration is a useful technique for security cloud computing [15,144]. VM migration can be used as Shuffle technique to address the multi-tenancy problems in a virtualized environment such as side-channel attack [164]. This wastes attacker's time and effort to gain information about a target VM and placing a malicious VM in the same physical host with the victim. Another advantage of using VM migration is that it can change the network topology in a cloud environment (e.g., Virtual Cloud Network (VCN)) and add the benefits of changing network topology by a Shuffle technique.

Danev *et al* [45] proposed an approach to securely migrating VMs in a private cloud as a shuffle MTD technique. The main approach they used is to utilize an extra physical Trusted Platform Module (TPM), and trusted parties for migration process. They also used public key infrastructure to secure their proposed protocol. Zhang *et al.* [164] proposed a shuffle method based on a VM-live migration in cloud environment to defend against the threats resulting from side-channel attack. They used random VM placement as the Shuffle technique. They showed a balance between the level of security against the cost incurred by VM migrations.

Penner and Guirguis in [120] proposed a set of MTD technologies to vary VM's location in the cloud to thwart Multi-Armed Bandit attacks (MAB) which was resulted from the a lack of adequate VM isolation in the cloud. Indeed, they evaluate their method based on the attacker's perspective, and also the VM migration time. They argued that their method can thwart MAB attack designed to find critical information (e.g. datasets and credit card information). In [119], the authors proposed a Shuffle MTD technique which utilizes the VM migration technique in the cloud. They investigated the way that VM can be implemented more effectively to be able to defend against intelligent attackers using dynamic attack surfaces. They also proposed MTD service strategy using probability models. Their results showed that the enhancement in effectiveness of MTD can be provided in the presence of the defender's awareness about dynamic attack surface.

Jia *et al.* [78] proposed an MTD technique named *client-to-proxy shuffling* which is able to continuously move the secret proxies. Their proposed method aimed to deal with Distributed Denial-of-Service (DDoS) attacks through securing data transmission between authenticated (legitimate) clients and a protected server. They evaluated their method using assessing the resistance and overhead of the proposed MTD technique using brute-force attacks.

Software and Platform Shuffling: Shuffle MTD techniques have been utilized in software, platform, and application layers as well. In [32], authors proposed an MTD technique for an IoT environment based on a reconfiguration approach for devices in a way that their firmware and cryptosystems can be shuffled. Vikram *et al.* [151] proposed a shuffle technique on the application layer to secure websites by randomizing the HTML elements. The static parts of websites such as HTML elements in the HTTP content/form page can be easily attacked to gain information by bots. Ultimately, randomizing those HTML elements/parameters are useful technique to defend against the bot attacks. The authors utilized a machine learning technique to enhance the effectiveness of their defensive strategy. They evaluated their method by measuring the page loading time overhead. In [115], the author developed a framework named Trusted Dynamic Logical Heterogeneity System (TALENT) for migrating the critical application running on the infrastructure through heterogeneous platforms which allows live changing of OS and Hardware. The TALENT can provide a virtualized environment in the OS layer through some containers having checkpoint compiler to enable live migration of different platforms running application.

The advantage of the Shuffle MTD technique is that it works with existing technologies (such as migrating a VM from one host to another host) without using another expensive security mechanism. Thus, Shuffle is a comparatively cheaper defensive option than other defensive mechanisms and also can be adapted immediately. Although Shuffle can only change the attack surface, it will not able to mitigate the current vulnerabilities existing on the system. However, additional service interruptions may be introduced to the system if Shuffle technique is not executed properly.

2.2.2 Diversity

In this section, we review the MTD studies based on changing the systems' components with different variants in various domains in the literature.

Software and Programming Language: Software diversity can be implemented on code blocks, loops, instructions, and different levels of systems and programs which can include linking, compilation, and/or installation [95]. For instance, in [19], the authors developed an approach to change a running program's variants erratically through which a large program can be divided into smaller components (e.g., cells or tasks). They used a recovery mechanism to enhance the system resilience. The main idea behind this technique is that using a different variant at runtime can make the attacker more confused. In this approach, only the variant that is affected by the attacker can be instantly changed by other variants during recovery time. However, dividing a running application to smaller chunks to be able to be detected and recovered can be a complex process.

In [70, 71], the authors introduced a Diversity MTD technique aiming to increase the network service resiliency. They deployed diversity on the virtual servers (VS) such as OS, virtualization components, Web Servers (WS), and application software. Then, they evaluated the proposed method through computing the probability of attack success. An example of a Diversity MTD technique in the programming language level is presented in [145]. The authors proposed a method which diversifies the programming language in different layers of the web application aiming to avoid code and SQL injection attacks. They showed that the proposed method can replace programming languages variants without imposing any disruption in the system functionality.

Network and Virtualization Diversity: In [170], the authors investigated on the relationship between deploying Diversity on the network configurations against the probability of attack success. They used a logical mission model to evaluate the proposed MTD technique. They utilized many factors to show the effectiveness of MTD techniques such as examining the network size, testing the frequency of shuffling and adaptations, and also the number of vulnerable computers. To do this, they used a network security simulator and conducted the experimental results.

Diversity MTD techniques leverages the existing technologies (e.g., changing an OS in a VM with another one). Although Diversity changes the system's variants and causes difficulties for attacker to deal with a new set of vulnerabilities, this technique cannot be effective enough if the new variants have high vulnerabilities. Moreover, the number of available variants for deploying Diversity is also important. If the number of those usable variants are not adequate, this would be inherent limitations of Diversity.

2.2.3 Redundancy

In this section, we discuss the existing studies proposing Redundancy-based MTD techniques in various implementation domains.

Redundancy of Software Components: Yuan *et al.* [154] developed a Redundancy approach for web servers which aims to defend against malicious code injection attacks on a web server using a self-protection model. The proposed method includes architectural adaptation threat detection and mitigation using so called agreement-based redundancy. The proposed method provides software component's replicas during runtime. However, they didn't study the effectiveness of their proposed technique.

Gorbenko *et al.* [56] proposed an MTD technique which provides redundant web services aiming to maximize system dependability. They evaluated the proposed method through the evaluation of system availability, reliability, and response time. However, they didn't assess the security of the proposed method (e.g., using security metrics).

Redundancy of Network Sessions: In [96], the authors adopted a redundancy technique for Cyber-Physical System (CPS) environments using the traffic morphing mechanism called *CPSMorph*. This mechanism can maintain a redundant number of network sessions which include the indistinguishable distribution of inter-packet delays comparing with other normal network sessions. Within a time constraint, a CPS message can be distributed through one of the redundant sessions. Using this technique, they can adjust the morphing process dynamically; consequently, it minimizes the overhead. They showed that their proposed method has low complexity while having high adaptation over CPS environments.

Although Redundancy MTD techniques are used to enhance the service availability and reliability on a system, adding additional replicas for any contents of a system such as servers, VMs, service paths may either incur additional costs to the system or affect the security of whole system. For instance, if Redundancy is not properly deployed, it makes an attack even easier for the attackers due to the larger attack surface caused by deploying a Redundancy on system's components (e.g., a server in a cloud). In this sense, evaluating the effectiveness of Redundancy technique is a non-trivial task based on the domains in which Redundancy is deployed and also based on the effects it may have on the attack surface.

2.2.4 Discussion and Limitations of Existing MTD Techniques

In this section, we discuss the main shortcomings of the current MTD techniques in literature based on the classification presented in table 2.1. The main limitations of MTD techniques we found in current studies are summarized as follows:

Limited investigations on combining MTD techniques: A large volume of Shuffle MTD techniques have been proposed in different application layers, as shown in Figure 2.1. Shuffle technique works with the existing technology to change the attack surface on a system (e.g., move a VM from one physical host to another host). Thus, if those existing technologies are not robust enough or deployed effectively against attacks, the effectiveness of Shuffle can be significantly limited by those vulnerabilities of the used system. For example, well-known vulnerabilities to attackers exist on a VM which is migrated to another host; in this case, although this migration makes the attackers confused in terms of changing VM's location, but those vulnerabilities are still available on that VM for the attackers to exploit. On the other hand, Diversity technique can also leverage the existing technologies, but it is able to vary the vulnerabilities of used system (such as different OS variants with different vulnerabilities). Thus, one can combine Shuffle and Diversity counterparts to cover the disadvantages of each technique and maximize the effectiveness of a proposed MTD.

We noticed that the Redundancy MTD techniques are mostly used to increase reliability and availability of a system (e.g., defense against Distributed Denial of Service (DDOS) attacks) and does not play a significant role as proactive MTD defense compared to the other two techniques. Redundancy can be easily excluded when Shuffle and Diversity MTD techniques are solely used without consideration of the critical trade-off between enhanced security and reliability. However, there is a lack in the literature for combining of Redundancy with Shuffle and Diversity.

Evaluation of the effectiveness of MTD techniques: Many MTD techniques are proposed in various application/operation domains. Most of the proposed methods rely on the evaluation of the performance and the overheads of the techniques and mostly measure the overhead. However, there still is a lack on comprehensive evaluation of the effectiveness of those proposed MTD techniques in terms of security metrics using security models. Moreover, the investigation on MTD techniques against economic impacts such as economic-related metrics have not been thoroughly investigated. To summarize, it is also critical to 1) evaluate MTD techniques using security metrics to show how the overall seurity of the system is improved after deploying MTD technique, 2) find trade-offs between enhanced security (e.g., *Risk*, *AC*, *etc.*) and performance (e.g., system utilization and service availability).

Adaptability and validation of MTD techniques on real testbeds: Proposed MTD techniques are mostly verified using analytical models, simulation, and emulation models. However, there is a lack on the studies to adapt and verify the MTD techniques on the real testbeds such as cloud computing.

2.3 MTD Techniques Evaluation

Security analysis plays an important role in evaluating a given system's security with different perspectives such as attacker's and defender's points of view [157]. Security analysis can be also incorporated into the MTD techniques to evaluate the effectiveness of MTD techniques. In this section, we provide a study on the methods, tools, and metrics which can be used to evaluate the effectiveness of MTD techniques using security analysis and modeling (such as Graphical Security Models (GSMs)).

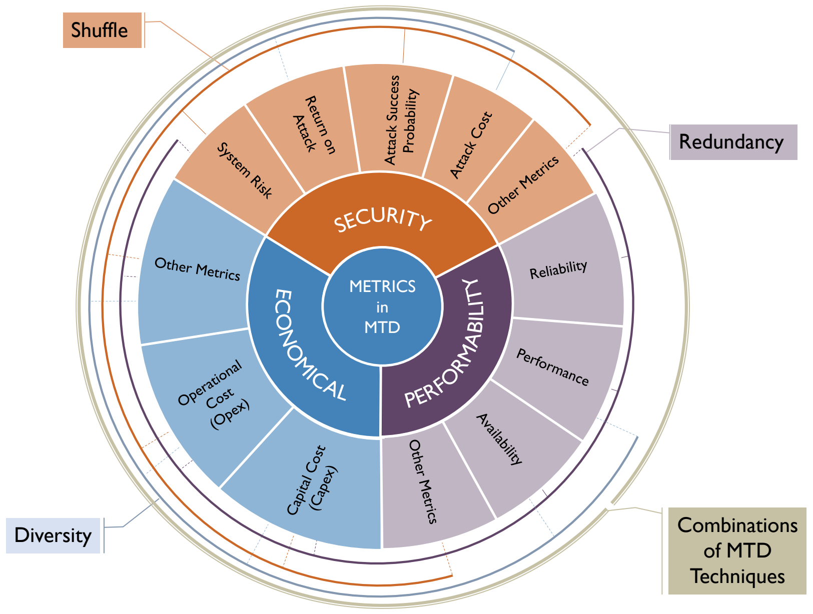


Figure 2.2: Metrics used for MTD techniques

2.3.1 Metrics for MTD Techniques

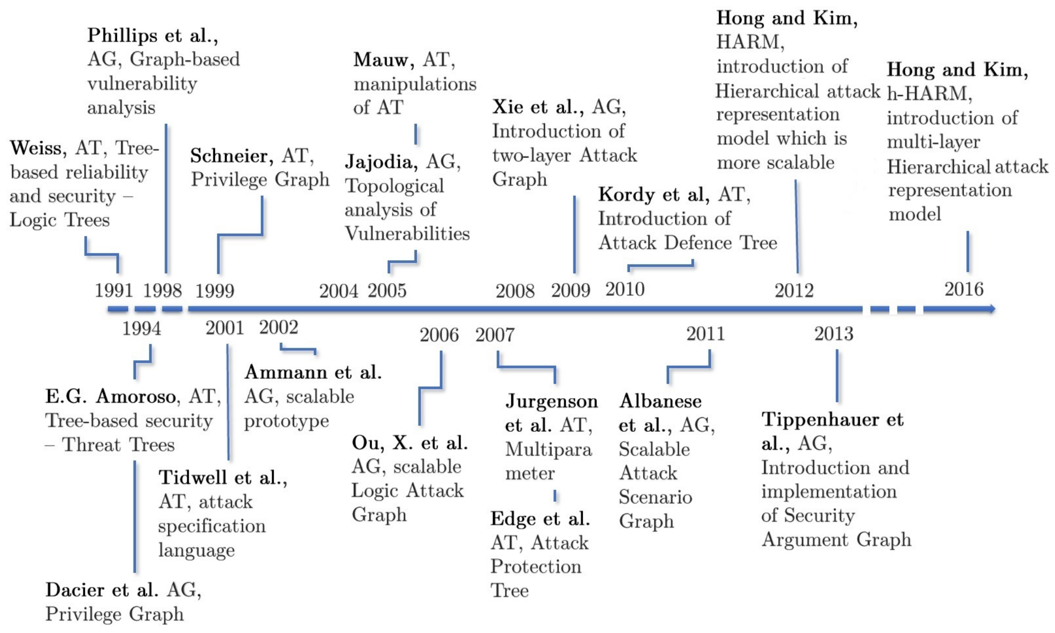
Figure 2.2 shows various metrics which can be utilized to evaluate the effectiveness of MTD techniques. They can be categorized into three main metrics including Security, Performability, and Economic Metrics. Security metrics can evaluate a given system in terms of both attacker's perspective (such as Attack Cost, and Return on Attack [157]) and defender's perspective (such as System Risk, Attack Success Probability). Likewise, performability metrics can evaluate proposed MTD techniques for the given system in terms of performance alongside the availability which includes the metrics such as Reliability, Availability, Overhead, and so forth. The third group are economic metrics which mainly evaluate the proposed defensive techniques against incurred costs such as Return on Security Investment metric. GSMs are useful tools to model the given system (such as enterprise networks [156], IoT [53], Cloud [13], *etc.* [67]) and accordingly evaluate the related security metrics. GSMs have also used to evaluate the effectiveness of MTD techniques by computing various metrics [12].

2.3.2 GSM Overview

In this section, we provide a review on GSMs techniques in the literature. Two main GSMs including Attack Trees (ATs) [131] and Attack Graphs(AGs) [121] have been proposed in order to model the complex and sophisticated attack scenarios in the formal ways. All potential and possible attack ways in a given system (such as a network) through which the attacker can compromise a target in that system (e.g., compromising a server) can be captured and described by ATs and AGs. Each path in an AT shows how an attacker can access to any crucial parts of a network leading to undesirable states. In [46], the authors defined four main phases which can model and represent of attacks by using GSMs which includes Information Gathering, Construction of GSM, Visualization and Analysis. Evaluating a system using GSMs usually undergoes the following steps [46]:

- Information Gathering: This phase is the preliminary step for security modeling in which the required information about the system need to be gathered. This information may include configurations, security rules, vulnerabilities, network topologies, hosts and virtual machines, connectivities of the network's components, *etc.*
- GSM Construction: The gathered information obtained from the previous phase can be used to construct GSMs. In this phase, possible attack scenarios can be captured based on hosts and vulnerabilities. However, most of the proposed AGs and ATs suffer from scalability problem for large networks.
- Security Analysis and Visualization: The final step is the evaluation of the given system using the constructed GSM. The security analysis process can incorporate many security metrics into the GSMs and evaluate the given system based on the various metrics of different aspects, as shown in Figure 2.2.

Proposed GSMs and Development: We have summarized and highlighted relevant GSM approaches in Figure 2.3. In [104], the authors proposed a method to model system's behavior and construct an AG through a simpler rank-based automaton (NFA). In [91], the authors proposed a measurement using AG for ranking and evaluating the security levels of a given system. They used both quantitative and qualitative security analysis approaches and utilized Bayesian Networks (BN) for their evaluation. The work presented in [84] used logical expressions and conditions to do statistical analysis for AG. In this method, a countermeasure solution is selected based on the cost-related criteria. However, they didn't explain how they generated the proposed AG in their methodology. Moreover, there have been numerous studies incorporating artificial intelligent (AI) approaches for security analysis of the given networks such as machine learning (ML) techniques (General ML and Classification-based ML [38, 39], and Reinforcement learning [147, 166]), generic algorithms, and so on [47]. In the study conducted by [152], the authors introduced AG-based security metrics. They used a combination of hosts' vulnerabilities and attacker's activity status to provide a security measurement using vulnerability information and probabilities techniques. However, the proposed methods are not scalable for large networks. In [69], the authors proposed a method to address the scalability issue of AGs by reducing the size of big graphs to smaller, but critical, ones. They used linear Depth First Search (DFS) and Boolean Clauses methods for this reduction. They could reduce the attack graph size by 15 percent. However, their proposed method cannot address the scalability issues appropriately in a very large context. In [122] the authors used



Weiss, AT, Tree-based reliability and security – Logic Trees
Phillips et al., AG, Graph-based vulnerability analysis
Schneier, AT, Privilege Graph
Mauw, AT, manipulations of AT
Jajodia, AG, Topological analysis of Vulnerabilities
Xie et al., AG, Introduction of two-layer Attack Graph
Kordy et al, AT, Introduction of Attack Defence Tree
Hong and Kim, HARM, introduction of Hierarchical attack representation model which is more scalable
Hong and Kim, h-HARM, introduction of multi-layer Hierarchical attack representation model
1991
1998
1999
2004
2005
2008
2009
2010
2012
2016
1994
2001
2002
2006
2007
2011
2013
E.G. Amoroso, AT, Tree-based security – Threat Trees
Dacier et al. AG, Privilege Graph
Tidwell et al., AT, attack specification language
Ammann et al. AG, scalable prototype
Ou, X. et al. AG, scalable Logic Attack Graph
Jurgenson et al. AT, Multipara meter
Edge et al. AT, Attack Protection Tree
Albanese et al., AG, Scalable Attack Scenario Graph
Tippenhauer et al., AG, Introduction and implementation of Security Argument Graph

Figure 2.3: Development of different GSMs approaches

Bayesian AGs to introduce a method to measure vulnerability as a metric called vulnerability scoring system. However, most of the proposed approaches and methods focusing on AG or AT suffer from scalability problem, especially, in larger networks. To address the aforementioned issues, Hong *et al.* [61], proposed a formal model based approach named Hierarchical Attack Representation Model (HARM) consisting of two hierarchical layers for capturing network reachabilities (as an AG), and network vulnerabilities (as an AT) in separate layers. HARM is more scalable and adoptable than other formal GSMs [62]. In the proposed method, topological information can be captured at the higher layer and vulnerability information remains in the lower layer of the model. However, this two-layered approach has been further developed and used in many studies [52, 64, 155] due to ease of uses and its merits to address the scalability problem.

2.4 Insights and Directions for this Research

Based on the limitations discussed for MTD techniques in this chapter such as Shuffle, Diversity, and Redundancy, we conduct an extensive investigation on the methods and practice for followings: 1) Combining different MTD techniques, 2) Evaluating the effectiveness of each combined MTD technique using various security metrics. 3) Implementing and adaptation of the MTD techniques in a realistic cloud platform. Each of the pointed contributions are extensively investigated and discussed in the following chapters of this study (Chapters 3 – 6).

Chapter 3

Evaluating the Effectiveness of Shuffle and Redundancy MTD Techniques on the Cloud

Summary

Moving Target Defense (MTD) is a defensive strategy to thwart adversaries by continuously shifting the attack surface. The MTD techniques can be applied to the cloud computing to make the cloud more unpredictable, hence more difficult to exploit. There are many MTD techniques proposed, and various metrics are used to measure their effectiveness. However, it is difficult to assess the effectiveness of MTD techniques when they are used in combinations. In this chapter, we propose a formal security assessment approach to evaluate the effectiveness of Shuffle and Redundancy techniques using security modeling. We use security metrics, such as system risk (*Risk*) and system reliability (*Reliability*), to evaluate those MTD techniques. In particular, we investigate how the security of the cloud changes when two categories of MTD techniques, Shuffle and Redundancy, are used in combination. We also present approaches to find important components in the cloud using Network Centrality Measures and the size of the cloud and evaluate the trade-off between security and performance in terms of the *Risk* and *Reliability*, respectively. Our experimental analysis shows that combining the Shuffle and Redundancy MTD techniques could enhance the security of the cloud with the trade-off between the *Risk* and *Reliability*, which can be managed using the proposed security assessment approach.

3.1 Introduction

Cloud computing (cloud) offers scalability, on-demand service, cost reduction, resilience, and available network access services to its customers. Many studies have explored implementing reactive and proactive security mechanisms to improve the security of the cloud computing resources and services [79, 167], but there is a lack of knowledge to evaluate how the security posture of the cloud changes when multiple security mechanisms are used in combination. As a result, there may be new vulnerabilities arising from unexpected dependencies or conflicts created when using multiple security mechanisms together [6]. Since cloud customers rely on the service provider for various services including data storage, infrastructure and services, security issues must be addressed for the customers to fully trust on cloud services offered by public and private cloud providers (e.g. Microsoft Azure, Amazon Web Service (AWS), Google Cloud, and so forth). Moving Target Defense (MTD) is a proactive defensive security solution in which it makes attack surface dynamic, unpredictable, and intricate for attackers [119,160,164,169]. Unlike the traditional security approaches that rely on detecting and removing vulnerabilities, MTD techniques aim to increase the attack efforts so that the attackers need to spend more resource (e.g., times and costs) to exploit the target system. However, most proposed MTD techniques evaluate their effectiveness using different evaluation methods. Thus, it is difficult to understand how the effectiveness of techniques changes when multiple MTD techniques are used together.

MTD techniques can be adopted in different cloud's deployment layers such as application, virtualization, hardware layers. For instance, Virtual Machine Live Migration (VM-LM) is a Shuffle technique in the virtualization layer. VM-LM is a feature on the clouds which can be enabled by the cloud provider and a VM can migrate from one physical host to another one [107]. The main goal of deploying VM-LM is to increase uncertainty for the attackers (e.g. to make information gathered by the attacker useless and obsolete). Moreover, VM replication is an example of Redundancy technique in the virtualization layer of the cloud through which different replicas of a VM can be created so that each replica has the same feature as the main VM. VM replication increases the reliability of the cloud by creating redundant VMs (e.g. crucial servers) so that if any crucial VM is compromised, others can provide the same services leading to high service quality for the users. OS diversification is a Diversity technique in the virtualization layer which can replace a VM's instance by another instance or image [9].

MTD techniques can be used individually or in combinations. Combining different MTD techniques would potentially benefit by grasping the strong points of each technique and can introduce additional benefits (such as enhancing security and decreasing the system interruption) which may not be possible under deployment of a single MTD technique. For instance, Redundancy techniques can be used to increase the service availability (fewer service interruption) and can be measured with system Reliability, while other MTD techniques like Shuffle are used to increase the security of a system. However, if Redundancy is not properly deployed, it may cost the cloud providers and also it may increase the attack surface. Thus, MTD techniques can be combined aiming to increase both security and reliability. However, combining MTD techniques may not necessarily improve the security of the system, because the operations of deployed MTD techniques may conflict. For example, application diversity to variate attack paths may become ineffective if a host shuffling technique changes the attack path. Therefore, it is of paramount importance to evaluate the combinations of the MTD techniques [11].

When it comes to the real cloud, deploying MTD techniques depends on the constraints defined by the cloud providers. For instance, for deploying OS diversification (as a Diversity technique) on the cloud, the cloud providers need to purchase the new OS, distributions, licenses, *etc.* which may cost the cloud providers. The economic impact of deploying Diversity techniques on the cloud providers does not mean that Diversity should not be deployed or combined with the other MTD techniques at all. However, it needs more investigation to find a trade-off between the economical impacts against security achievements. Considering and evaluating all combinations of all MTD techniques over the system's layers makes the scope too large. In this chapter, we only focus on combining the two Shuffle and Redundancy MTD techniques. The investigation and evaluation of other combinations including Diversity techniques is out of scope of this chapter and is presented in Chapters 4 and 5.

Graphical Security Models (GSM), such as Attack Graphs (AGs) [139] or Attack Trees (ATs) [132], can be used to formally assess the security of systems and networks [67,89]. They utilize various security metrics to represent the security posture of the system (e.g. System risk, reliability, Attack Cost, *etc.*), which can provide different aspects of a system's security [157]. Consequently, MTD techniques can be incorporated and modeled using GSMs to assess their effectiveness [66]. This helps to find better strategies and formulate the possible optimal MTD solutions before deploying them. However, there is a lack of studies on how MTD techniques can be combined, especially in the context of the cloud. Many studies proposed new MTD techniques at different layers. However, a few of them focused on evaluating the effectiveness of combined MTD method on the infrastructure layer of the cloud. Moreover, most of the proposed literature on MTD have not investigated on comprehensive security analysis using GSMs (like AG, AT, HARM, *etc.*) to compute important security metrics.

To address the aforementioned problems, we propose methods to evaluate the combinations of MTD techniques, taking into account the Shuffle and Redundancy, using a GSM named HARM [64].

The preliminary results and earlier version of this chapter is given in [13]. In addition to the investigations we have already made in the earlier version, we make new contributions which, to the best of our knowledge, has not been researched by other works. The new contributions of this chapter are summarized as follows.

- We provide the formal definitions for the MTD techniques Shuffle (S) and Redundancy (R) and the combinations of these two (S+R) explicitly as the functions applied on HARM and provide the pseudocodes for each MTD technique with regards to the defined formalism;
- We present and formalize the system unattackability metric as an additional component to investigate the effect of deploying S+R. We estimate system Unattackability under various attack rates to show how resistant the system is against varying degrees of attack rates. We demonstrate that the combination of two MTD techniques, such as S+R, contributes the likelihood to increase the system Unattackability;
- We conduct a regression analysis between two Network Centrality Measures (NCMs) Betweenness and Closeness against Shuffle, Redundancy, and S+R. We conclude that there is a strong relationship between the two NCMs and the Redundancy technique alone but no other MTD techniques;
- We include a small scale cloud system to help for understand the formalization and calculation steps. Besides the small scale cloud example, we used a larger cloud-band model, as the cloud example in a larger scales for more in-depth analysis. We utilize Firewall Decision Diagram (FDD) to concretely demonstrate the way the connectivity among network components are captured under this system. We show how this insight is used for the HARM construction.

This chapter is organized as follows. A review of the related work is summarized in Section 3.2. In Section 3.3, we define the necessary concepts, definitions, mathematical notations, and propose formalisms for MTD techniques. In Section 3.4, we analyze the MTD techniques using HARM. Discussion and limitations of the current study are presented in Section 3.5. Lastly, we conclude this chapter in Section 3.6.

3.2 Related Work

Most of the existing works only focused on the novelty on the proposed strategies and layers of implementation, like defensive method after detecting an attack, low-level shuffling techniques, finding a suitable time-period for applying the IP mutation frequently, dealing with worms and web bots through MTD [31, 151], and so on. However, there are very few works evaluating the MTD techniques especially for cloud computing needing precise and scalable security analysis. Carroll *et al.* [31] proposed a port number shuffling technique, and evaluated the effectiveness using a theoretical analysis in terms of practicability using the proposed probabilistic models. The attack success probability versus connection lost rates are evaluated. Luo *et al.* [100] also proposed a port number-based shuffle MTD technique, where they compared the attack success rate for evaluation. Another shuffle-based MTD technique, particularly for virtualized systems, is to use VM migration. Danev *et* *al.* [45] proposed a VM Migration in the cloud in a secure way by using an extra physical Trusted Platform Module, and trusted parties for the migration process. Okhravi *et al.* [116] proposed an application migration technique in the virtualized system to avoid malicious attackers exploiting application vulnerabilities (e.g. critical and vulnerable running application). They used live migration technique for critical applications in different platforms so that attacker's reconnaissance process to find a specific target or platform becomes ineffective. Nguyen and Sood [111] proposed a system architecture for MTD named "SCIT-MTD" aiming to secure web applications on the cloud through minimizing the attack surface for the servers compromising the system. They argued that the velocity of moving, diversity of various configurations, and redundancy of active servers (VMs) are important factors to enhance the security availability of the entire system. Bardas *et al.* in [21] presented an MTD platform for Cloud-Based IT Systems named MTD CBITS using an Automated eNterprise network COmpileR (ANCOR) [149] (which is a framework for creating and managing the cloud-based IT systems). They developed an automated platform for IT system in which any component can be replaced with a new version aiming to increase the attack difficulty. Indeed, they adapted a MTD strategy on a real cloud (e.g. OpenStack platform) using live instance replacement technique. They showed the practicality of their approach and analyzed the security benefits of their proposed method. They also evaluated the effectiveness of their MTD technique by analyzing the cost and security benefits of the proposed MTD platform. They showed that the deploying MTD technique based on live instance replacements has a low impact (negligible performance overhead) on the normal operation of the system. However, this technique can be categorized as a Diversity technique in which the systems' components can be replaced with another variant serving the same functionality. Kampanakis *et al.* in [82] investigated the advantage and disadvantages of using MTD technique through SDN and the methods for implementing it. They used a shuffle method for the host randomization and mutation. The evaluation was based on the overhead against the performance of using the MTD through SDN. However, they did not use a formal security models to perform in-depth security analysis and evaluation.

Even though many MTD techniques have been proposed in the literature, there is still a lack in combining MTD techniques, also the effectiveness of the proposed techniques needs to be evaluated in order to ensure that security requirements are met besides performance.

3.3 Preliminaries

In this section, we define and recall the required notations and mathematical formalisms for security analysis models, metrics and calculation method, MTD techniques, and pseudocodes through a cloud example. The defined notations will further be used in the other sections.

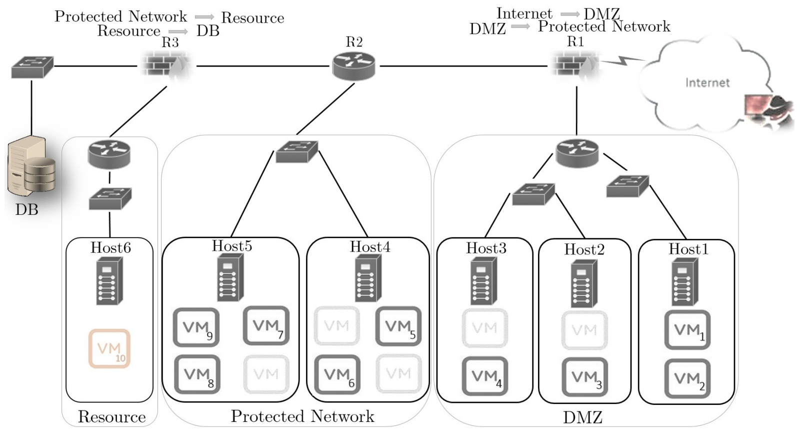


Figure 3.1: A cloud system example

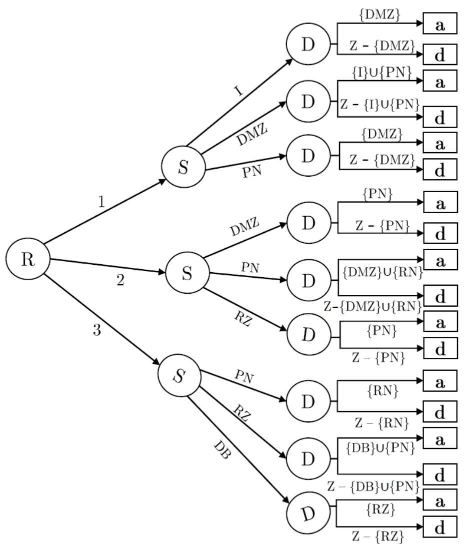


Figure 3.2: FDD diagram for cloud zones.

3.3.1 System Setting and Configuration

We set up a cloud system consisting of three subnets; Demilitarized Zone (DMZ), Protected Network (PN), and Resource Zone (RZ), as shown in Figure 3.1. Each subnet contains hosts, and each host holds up to four Virtual Machines (VM). VMs hosted in the DMZ are installed with Windows 10 operating system (OS), and VMs in PN and RZ are installed with Enterprise Linux OS. The entry point of the cloud from the Internet is the interface 0 of the gateway router (R1) connecting the router to the Internet, and interface 1 and 2 connect the gateway router to inside cloud, see in Figure 3.1. We assume that an attacker resides outside of the network and might exploit the vulnerabilities of the operating systems to gain access to the network. We assume that the attacker plans to compromise the Database (DB) in the RZ. The system's configurations, connectivi-

Table 3.1: OS Vulnerabilities

OS_{ID}	CVE ID	CVE BS	Impact	Exploitability
$W10_{v0}$	CVE-2017-8530	5.8	4.9	8.6
$W10_{v1}$	CVE-2017-8495	6.0	6.4	6.8
$W10_{v2}$	CVE-2016-7247	7.5	3.6	3.9
$W10_{v3}$	CVE-2016-3209	5.0	2.9	10
$W10_{v4}$	CVE-2016-0019	9.3	10	8.6
$Linux_{v0}$	CVE-2016-7034	6.8	6.4	8.6
$Linux_{v1}$	CVE-2016-4278	5.0	2.9	10
$Linux_{v2}$	CVE-2016-10309	7.5	6.4	10
$Linux_{v3}$	CVE-2016-10307	10	10	10
$Linux_{v4}$	CVE-2016-10066	4.3	2.9	8.6

Table 3.2: Cloud firewall traffic and access control rules

VM	Default Details		Access permit only from
	OS	Zone	
VM_1	W10	DMZ	Internet, VM_3
VM_2	W10	DMZ	Internet, VM_3
VM_3	W10	DMZ	$Host_1$, $Host_4$
VM_4	W10	DMZ	VM_2, VM_6
VM_5	Linux	PN	$Host_2$, $Host_4$, VM_7
VM_6	Linux	PN	$Host_1$, $Host_3$, VM_9
VM_7	Linux	PN	$Host_5$, $Host_6$, VM_5
VM_8	Linux	PN	$Host_5$, $Host_6$
VM_9	Linux	PN	$Host_5$, $Host_6$, VM_6
VM_{10}	Linux	RZ	$Host_5$
DB	Linux	Database	$Host_6$

ties, and constraints are assumed as follows (we use the following assumptions for the simplicity in description and models and these assumptions could be relaxed):

- All VMs are active and never suspended
- A VM can migrate to another host, and the downtime for that is neglectable
- A VM cannot migrate to other subnets
- Only VM_1 and VM_2 are connected to the Internet
- VM_{10} on $Host6$ cannot be migrated due to system constraints
- All VMs in $Host2$, $Host4$, and $Host5$ are interconnected, and VM_1, VM_4 are always connected to a VM in $Host2$ (if any)
- VM_7 and VM_8 are always connected
- Only VM_{10} can access to the Database (DB)

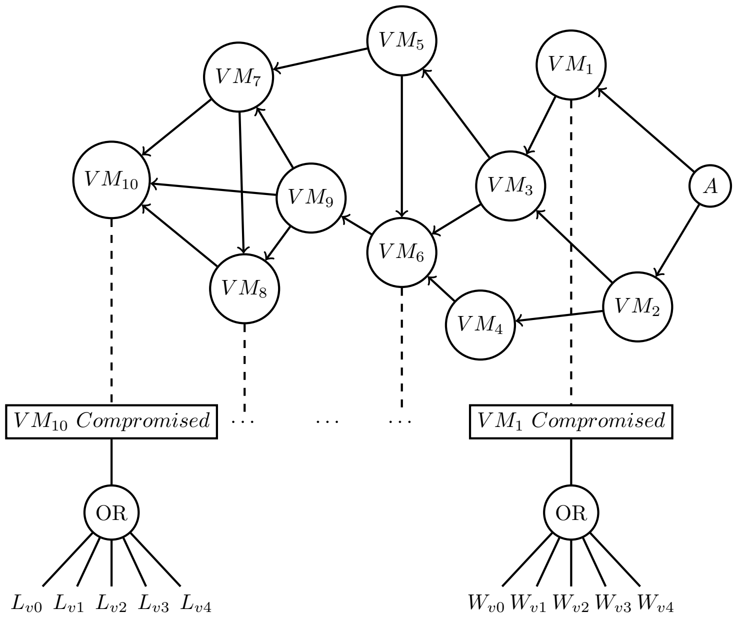


Figure 3.3: Two-layer HARM of the cloud example

Table 3.1 shows the vulnerabilities for Windows 10 (W) and Linux (L) OS. Here, we only modeled the vulnerabilities that can bypass firewalls and authentications. There are five of those vulnerabilities for the Windows OS, which are denoted as $W10_{v0}$ up to $W10_{v4}$, and similarly for the Linux OS from $Linux_{v0}$ up to $Linux_{v4}$. Further information regarding these vulnerabilities and measures can be found in the National Vulnerability Database (NVD) [106].

In our cloud example, we assume that firewall rules defined in routers (R) control the traffic passing between zones. We define Firewall Decision Diagram (FDD) [97] mapping every packet between zones and the Internet (I). We assume that the firewall rules are applied to three routers $R = \{1, 2, 3\}$. As we defined before, we have five zones including the Internet $Z = \{I, DMZ, PN, RZ, DM\}$. We define source zone S such that $S \in Z$, and destination zone D such that $D \in Z$. We donate a to traffic acceptance and d to traffic discard. Figure 3.2 demonstrates the FDD diagram for controlling traffic between zones in the cloud system. Also, Table 3.2 shows the VMs access control through an access list inside the hosts.

3.3.2 HARM Construction

We construct a two-layered HARM [66] to model and evaluate the MTD techniques. Using the HARM, we calculate the Risk as the seminal metric for assessing the overall security of the network. We use an AG and an AT in the upper and lower layers of the HARM respectively. The former can capture the reachability between VMs, and the latter captures the vulnerabilities existing on a VM. Further details of conducting the Risk analysis using the HARM is given in Section 3.3.8. HARM can be formulated as below [66].

Definition 1. HARM can be defined as a 3-tuple $H = (U, L, C)$ where U refers to the upper layer corresponding to an Attack Graph (AG), and L represents the lower layer in which an Attack Tree (AT) is constructed. We define $C = U \rightarrow L$ as a one-by-one mapping of the upper layer to the lower layer.

Definition 2. The upper layer of HARM U can be defined as a bidirectional graph $U = (VM, E)$, where $VM = \{vm_1, vm_2, \ldots, vm_n\}$ is a set of VMs in the cloud, with $|VM| = n$, and $E \subseteq VM^2 = \{(i,j)|i,j \in VM\}$ is a set of reachability of VMs. Then, vm_i shows a specific VM in the network ($i \in \{1, \ldots, n\}$). We can show the bidirectional graph of the VMs through a non-symmetric adjacency matrix a, in Equation (3.1). We define $e_{i,j} \in E$ as a bi-directed edge showing connectivity between vm_i and vm_j.

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{if } (i,j) \notin E \end{cases} \tag{3.1}$$

Definition 3. The lower layer L uses an AT, which is defined as a 3-tuple $L = (V, G, root)$. Then, $V = \{v_1, v_2, \ldots, v_m\}$ is a set of vulnerabilities in a VM denoting leaves of the tree. We denote the number of vulnerabilities in each VM as $|V| = m$, and G is a set of logical gates $G = \{AND\text{-gate}, OR\text{-gate}\}$ constructing the inner nodes of tree, and $root$ is the corresponding node in U.

Figure 3.3 shows the constructed HARM for the example cloud system.

Example 1. We formulate the HARM for cloud system (cs) shown in Figure 3.1 as $H_{cs} = (U_{cs}, L_{cs}^{VM_{cs}}, C_{cs})$, where U_{cs} is the AG of the cloud system represented in the upper layer of HARM and $L_{cs}^{VM_{cs}}$ denotes an AT in the lower layer corresponding to a VM in the upper layer, and $C_{cs} = VM_{cs} \rightarrow L_{cs}^{VM_{cs}}$ connects a VM in the upper layer to the corresponding lower layer L_{cs}.

Then, we can define the upper layer of HARM for cloud system as $U_{cs} = (VM_{cs}, E_{cs})$, where $VM_{cs} = \{A, vm_1, vm_2, vm_3, vm_4, vm_5, vm_6, vm_7, vm_8, vm_9, vm_{10}\}$, and the connectivity of VMs is as $E_{cs} = \{(A, vm_1)$, (A, vm_2), (vm_1, vm_3), (vm_2, vm_3), (vm_2, vm_4), (vm_3, vm_5), (vm_3, vm_6), (vm_4, vm_6), (vm_5, vm_6), (vm_5, vm_7), (vm_6, vm_9), (vm_7, vm_8), (vm_7, vm_{10}), (vm_8, vm_{10}), (vm_9, vm_7), (vm_9, vm_8), $(vm_9, vm_{10})\}$.

The lower layer of vm_3 in the cloud system can be defined as $L_{cs}^{vm_3} = (V_{W10}, G, vm_3)$, where $V_{W10} = \{W10_{v_0}, W10_{v_1}, W10_{v_2}, W10_{v_3}, W10_{v_4}\}$ is a set of OS vulnerabilities in vm_3 (obtained from Table 3.1), and G=OR-gate.

3.3.3 Network Centrality Measures on HARM

We compute NCMs in order to find a set of network components serving a crucial role in an event of an attack without exhausting all possible attack paths and named them as Importance Measures (IMs). We compute NCMs in the upper layer of the HARM to obtain the IMs [63]. In here, we consider two main NCMs; Betweenness Centrality (C_b) and Closeness Centrality (C_c) [27]. However, there are other NCMs measures which can be used on MTD techniques (e.g., Harmonic Closeness, PageRank, Eigenvector, *etc.*), but they are out of the scope of this study. NCMs can be formulated based on the upper layer of HARM defined in definition 2. Then, we can rank the VMs based on their NCM values as follows.

Definition 4. Let d be a function calculating geodesic distance of two VMs in U, then we can calculate the C_c value of a specific VM in the network as Equation (3.2), and C_b value of a VM can be computed as Equation (3.3).

$$C_c(vm_i) = \left(\frac{1}{n-1} \sum_{j \neq i \in VM} d(vm_i, vm_j) \right)^{-1} \tag{3.2}$$

$$C_b(vm_i) = \sum_{s,t \in VM \setminus \{vm_i\}} \frac{\delta_{st}(vm_i)}{\delta_{st}}, \tag{3.3}$$

where δ_{st} is a function calculating the total number of the shortest path between each pair of VMs $(s, t) \in VM$, and $\delta_{st}(vm_i)$ denotes the number of those paths passing through the specific VM (vm_i).

3.3.4 Selection Criteria

We consider some criteria to select VMs on the cloud in order to deploy MTD techniques on them. We defined three Selection Criteria (SC). (i) The VMs can be ranked and selected based on their C_b values , (ii) the SC can be based on C_c value , or (iii) none of them (i.e. random VM is selected using random selection function denoted by f), as in Equation (3.4).

$$k = \begin{cases} \arg \max\limits_{vm_i \in VM} C_b(vm_i), & \text{if } SC = C_b \\ \arg \max\limits_{vm_i \in VM} C_c(vm_i), & \text{if } SC = C_c \\ f(1, |VM|), & \text{otherwise} \end{cases} \tag{3.4}$$

Then, the value of k determines the argument of the VM that needs to be selected for deploying MTD technique on. For instance, $k = 3$ shows that MTD technique should be applied on vm_3.

3.3.5 Shuffle Formalism

We formulate the Shuffle technique in which the shuffle function is applied to HARM as follows.

Definition 5. Let $S(H, k)$ be a shuffle function on HARM where $1 \leq k \leq n$, and k denotes a specific VM that should be selected for shuffling. Then the result of shuffle function is as $S(H, k) = H^{\mathrm{s}}$. We define $H^{\mathrm{s}} = (U_k^{\mathrm{s}}, L, C)$ where U_k^{s} is the transformed AG resulted from shuffle on vm_k in the upper layer of the HARM and can be represented as $U_k^{\mathrm{s}} = (VM, E')$, where $E' \subseteq VM \times VM$.

Note that shuffle function $S(H, k)$ preserves L and C because it only changes reachability of VMs in the upper layer of HARM.

3.3.6 Redundancy Formalism

We formulate the Redundancy technique in which the redundancy function is applied to HARM as follows.

Definition 6. Let $R(H, k, r)$ be a redundancy function on HARM where k denotes the VM that should be replicated for r times, then the result of redundancy function is as $R(H, k, r) = H^{\mathrm{r}}$, where $1 \leq k \leq n$ and $r \leq l$, and l is a limit for replication of a VM. Thus, vm_k^{r} shows the replicated VM in the upper layer. We define $H^{\mathrm{r}} = (U_k^{\mathrm{r}}, L_k^{\mathrm{r}}, C)$ where U_k^{r} is a transformed AG resulted from replication of vm_k for r times in the upper layer of HARM and can be represented as $U_k^{\mathrm{r}} = (VM', E')$, where VM' can be shown as:

$$VM' = VM \cup \left(\bigcup_{r=1}^{l} VM_k^{\mathrm{r}} \right)$$

and $|VM'| = n + r$, and $E' \subseteq VM' \times VM'$.

Replication of VMs in the upper layer results in adding vulnerabilities in the lower layer of HARM, then we can define the lower layer as follows.

Definition 7. Let V^{r} be a set of vulnerabilities caused by replication of a VM in the upper layer of HARM, then the lower layer of HARM can be updated as $L_k^{\mathrm{r}} = (V', G, root)$, where $V' = V \cup V^r$.

Unlike shuffle function, $R(H, k, r)$ changes L and C with new vulnerabilities resulting from VM replication.

3.3.7 Combination of S+R Formalism

We formulate the combination of Shuffle and Redundancy (S+R) as a function on HARM as follows.

Definition 8. Let S+R(H, k_s, k_r, r) be a S+R function on HARM where k_r shows the VM that is selected to be replicated for r times, and k_s is the VM selected to be shuffled. Then the result of S+R function is as S+R$(H, k_s, k_r, r) = H^{\mathrm{s+r}}$, where $1 \leq k_s, k_r \leq n$ and $0 < r \leq l$. Thus, $vm^{\mathrm{r}}_{k_r}$ shows the replicated VM and $vm_{k_s} \in VM$ is the VM that is shuffled in the upper layer . We define $H^{\mathrm{s+r}} = (U^{\mathrm{s+r}}_{k_r,k_s}, L^{\mathrm{s+r}}_{k_r}, C)$ where $U^{\mathrm{s+r}}_{k_r,k_s}$ is a transformed AG in the upper layer in which S+R is deployed on and $L^{\mathrm{s+r}}_{k_r}$ is the corresponding transferred AT in the lower layer. Then, the former can be represented as $U^{\mathrm{s+r}}_{k_r,k_s} = (VM', E')$, where VM' can be shown as:

$$VM' = VM \cup \left(\bigcup_{r=1}^{l} VM^{\mathrm{r}}_{k_r} \right)$$

and $|VM'| = n + r$, and $E' \subseteq (VM + r) \times (VM + r)$. Next, the latter can be shown as $L^{\mathrm{s+r}}_{k_r} = (V', G, root)$, where $V' = V \cup V^r$, and V^{r} is a set of vulnerabilities caused by S+R in HARM.

3.3.8 System Risk Analysis

System Risk (Risk) is used to evaluate the effects of MTD techniques, which can be calculated as the product of exploitability of a VM (attack success probability for a VM) and the total impact of the attack on that specific VM [128]. Then, a sum of the all risks associated with the all VMs in the all attack paths in a system defines a total risk of a system. We first construct the HARM for our example network using the connectivity information of VMs obtained from system constraints (e.g. Firewall rules) and vulnerabilities information listed in Table 3.1. Then, we show how HARM can compute the Risk value. As we defined in Definition 2, we assume that the upper layer of HARM contains a set of VMs, where n is the number of VMs, and each $vm_i \in VM$ has up to a m number of vulnerabilities. Then, there exists a vulnerability $v_i \in vm_j \mid 0 \leq i \leq m, 0 \leq j \leq n$.

Lower layer of HARM includes two logical gates, AND-gate $\in G$, OR-gate $\in G$ connecting the vulnerabilities exist in this layer. Let AND_x represents a set of vulnerabilities and other logical gates connected by the AND-gate$_x$, and the OR_x shows a set of vulnerabilities and other logical gates connected by the OR-gate$_x$. Then, let $p(vm_j)$ be the probability of compromising the vm_j, and $p(v_i)$ is the probability of attack success when exploiting the vulnerability v_i. Also, we let $p(AND_x)$ be the probability of attack success for exploiting all vulnerabilities grouped for that AND-gate$_x$, and $p(OR_x)$ be the probability of attack success for exploiting any vulnerability for that OR-gate$_x$, respectively. Then, we can calculate the probability of attack success for vulnerabilities grouped by AND_x or OR_x gates as shown in Equations (3.5) and (3.6), respectively.

$$p(AND_x) = \prod_{v_j \in AND_x} p(v_j) \tag{3.5}$$

$$p(OR_x) = 1 - \prod_{v_j \in OR_x} \Big(1 - p(v_j)\Big) \tag{3.6}$$

Using Equations (3.5) and (3.6), we can calculate the probability of an attack success to compromise vm_i as shown in Equation (3.7) denoted by the top-gate, TOP.

$$p(vm_i) = p(TOP) \mid TOP \in \{AND_x, OR_x\} \tag{3.7}$$

We define the impact of an attack exploiting a vulnerability v_i as I_{v_i}. Then, we define the impact of an attack exploiting vm_i as denoted as I_{vm_i}, which is shown in Equation (3.8).

$$I_{vm_i} = \max_{v_j \in vm_i} (I_{v_j}) \tag{3.8}$$

Then, we denote the risk associated with vm_i as $Risk_{vm_i}$, which is calculated by the product of the probability of an attack success and the impact of an attack to vm_i as shown in Equation (3.9).

$$Risk_{vm_i} = p(vm_i) \times I_{vm_i}, \tag{3.9}$$

Here, we assume that each attack path is independent to other attack paths in the system. Then, we can define the Risk as a cumulative sum of all the risk associated with VMs in all possible attack paths, $paths$, where $path = (vm_1, vm_2, \ldots, vm_n) \in VM \times VM \times \ldots \times VM \mid path \in paths$ defines a series of VMs that form an attack path such that vm_i is adjacent to vm_{i+1} for $1 \leq i < n$. Finally, the total Risk for cloud ($Risk_c$) value can be calculated as shown in Equation (3.10).

$$Risk_c = \sum_{vm_i \in path \in paths} Risk_{vm_i} \tag{3.10}$$

3.3.9 Reliability Analysis

System Reliability (Reliability) is calculated using SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator) [130], which uses a reliability graph to quantify the chance of the existence of a path between an entry point of the network and the target (e.g. a Database). We assume that time to failure for each VM are exponentially distributed. We can estimate a VM failure probability $F(t)$ at the time t using a cumulative exponential distribution [159]. Hence, we define the Reliability of a VM as the probability that the failure on a VM has not occurred at time t, $R(t) = 1 - F(t)$. Finally, we can use SHARPE to evaluate the Reliability of the whole system (cloud) based on a given time and failure rate. To this purpose, the upper layer of the HARM can be fed into SHARPE as a reliability graph. Then, SHARPE can evaluate the robustness of the cloud over the time by considering various given failure rates.

Algorithm 1: Shuffle technique on HARM

```
Data: $H = (U, L, C)$, SC                                  /* Selection Criteria */
Result: $H^s = (U^s_k, L, C)$
1  begin
2      foreach $vm_i \in \{vm_1 \ldots vm_i\}$ do
3          if $SC == C_b$ then
4              $\beta = C_b(vm_i)$
5              Add $\beta$ into $Z_{vm_i}$
6          else if $SC == C_c$ then
7              $\zeta = C_c(vm_i)$
8              Add $\zeta$ into $Z_{vm_i}$
9          end
10     end
11     $k \leftarrow [\text{Max}(Z_{vm_i})]$
12     foreach $e_{i,j} \in E$ do
13         if $i{==}k$ or $j{==}k$ then
14             Remove $e_{i,j}$ form $U$
15         end
16         Create new edge $e_{x,j}$ in $U$
17     end
18     return $U^s_k$                                   /* Only $U$ in $H$ is changed */
19 end
```

3.4 Deploying MTD Techniques

MTD techniques can be applied to different layers of a network, e.g. Application Layer, Virtual Machine, Host, Hypervisor, and Hardware [5, 151, 164]. In this chapter, we consider deploying Shuffle, Redundancy, and Combination of both techniques in the Virtual Machine layer.

To deploy Shuffle technique, we utilize VM-LM in the cloud. In this technique, a VM can migrate from one physical host in the cloud to another host (subject to system constraints). Any changes in the cloud with regards to the reachability of VMs and vulnerabilities of OS can be captured by HARM. We present the algorithms for Shuffle technique based on the Shuffle formalism (3.3.5) in Algorithm 1. Then, we apply VM replication to deploy Redundancy technique. We denote k number of replications of a VM as k-R. We present the algorithms for Redundancy technique based on the Redundancy formalism defined in 3.3.6 in Algorithm 2. Finally, we combine both VM-LM and VM replication to deploy the S+R technique.

3.4.1 MTD Technique Analysis

In the previous section, we defined the formalisms for Shuffle, Redundancy, and S+R techniques in conjunction with HARM. In this section, we analyze the effectiveness of the MTD techniques through simulation. We simulated a large Cloud-band model as

Algorithm 2: Redundancy technique on HARM

```
Data: $H = (U, L, C)$, SC                                  /* Selection Criteria */
Data: R                                        /* R is the number of replicas */
Result: $H^r = (U^r_k, L^r_k, C)$
1  begin
2      foreach $vm_i \in \{vm_1 \dots vm_i\}$ do
3          if $NCM == C_b$ then
4              $\beta = C_b(vm_i)$
5              Add $\beta$ into $Z_{vm_i}$
6          else if $NCM == C_c$ then
7              $\zeta = C_c(vm_i)$
8              Add $\zeta$ into $Z_{vm_i}$
9          end
10     end
11     $k \leftarrow [\mathrm{Max}(Z_{vm_i})]$
12     foreach $vm_i \in \{vm_1 \dots vm_i\}$ do
13         if $vm_i == k$ then
14             for $r = 1$ to $R$ do
15                 Create $vm_{n+r}$
16                 $V^r \leftarrow$ set of new vulnerabilities
17                 Create $L^r_k(V^r, G, vm_{n+r})$
18                 Add $vm_{n+r}$ into $VM$
19                 foreach $e_{i,j} \in E$ do
20                     if $i{=}{=}k$ then
21                         Create new edge $e_{n+r,j}$ in $U$;
22                     end
23                     if $j{=}{=}k$ then
24                         Create new edge $e_{i,n+r}$ in $U$;
25                     end
26                 end
27             end
28         end
29     end
30     return $U^r_k, L^r_k, C$ /* $U, L$ in $H$ are changed */
31 end
```

used in [13]. This model includes two cloud-band nodes that can hold up to 450 VMs. Only a few VMs in the Cloud-band are connected to the Internet (i.e., front-end servers). We assume there is an attacker outside the cloud, and the attack goal is to compromise the resource node by compromising VMs in the attack paths. We assume that VMs can migrate between the Cloud-band nodes if there is an available space, which rearranges the logical connections between the VMs. Moreover, it is assumed that all VMs in the Cloud-bands are using the same OS.

In this chapter, we analyze the effectiveness of our deployed MTD techniques including Shuffle, Diversity, and combination of both. We use HARM and SHARPE to analyze

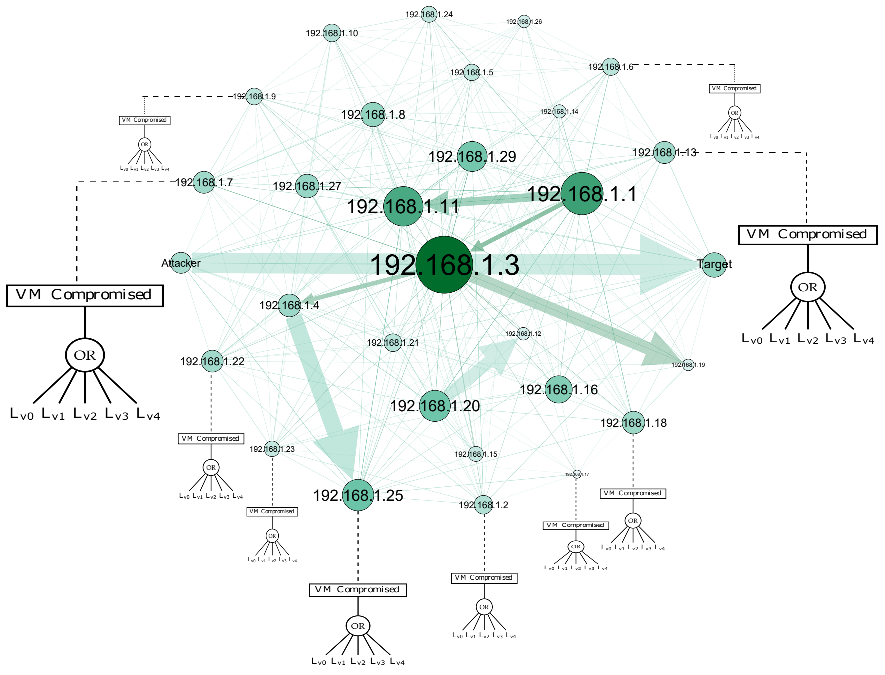


Figure 3.4: Generated HARM of cloud-band with 50 VMs

system Risk and the Reliability respectively. We measure the changes in Risk and Reliability to evaluate the effectiveness of MTD techniques. In the following sections, we show the effects of the MTD techniques on the Risk and Reliability metrics. Figure 3.4 shows an example of upper and lower layers of HARM generated for the Cloud-band model including 50 VMs.

The Risk and Reliability of the current system have been evaluated based on different number of VMs in each cloud-band node and illustrated in Figures 3.6a and 3.6b for further comparison with the results of deploying MTD techniques.

In the following section, we will show how deploying MTD techniques would affect the overall Risk and Reliability of the current system.

Table 3.3: Correlation of metrics against NCMs

Metrics correlations	Closeness		Betweenness	
	Trend	Value	Trend	value
Risk	Linear	70%	Exponential	92%
Reliablity	None	24%	None	28%

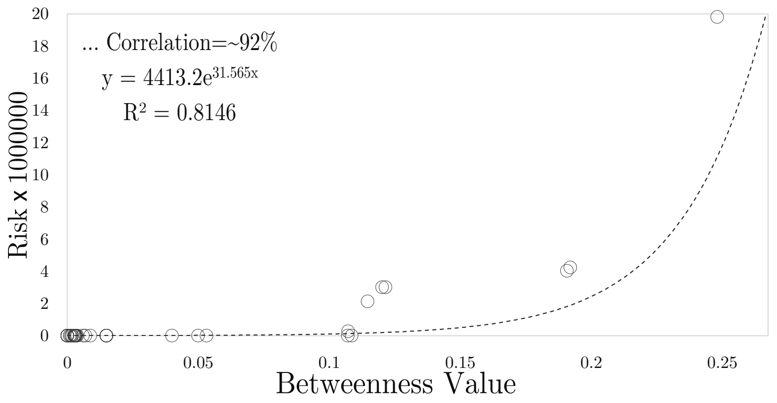
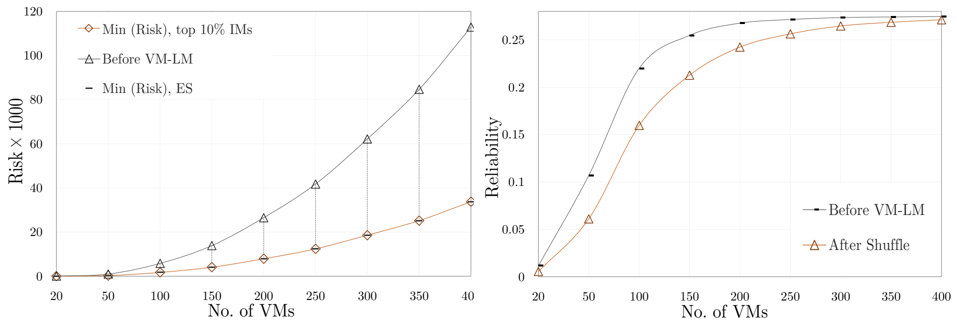
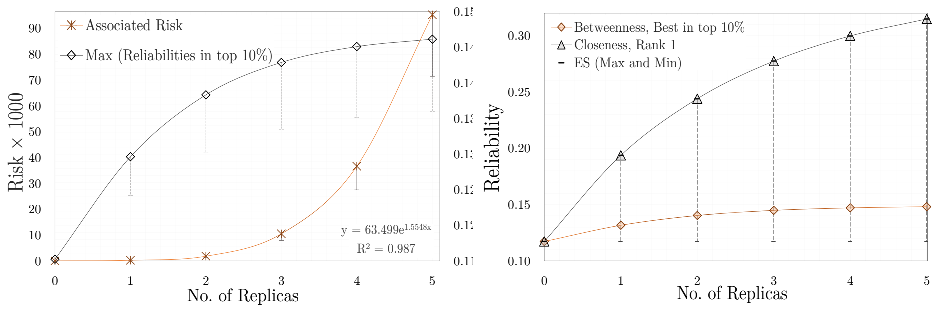


Figure 3.5: Exponential line fitted among Betweenness and Risk.



(a) (b)

Figure 3.6: Comparison of the Risk and Reliability after deploying Shuffle in Cloud-band. (a) the Risk resulting from deploying Shuffle based on the top 10% of Betweenness and ES. (b) Reliability after deploying Shuffle.



(a) (b)

Figure 3.7: Comparison of the Risk and Reliability based on Betweenness and Closeness after deploying Redundancy (from 0-R to 5-R). (a) the result of deploying Redundancy on the top 10% of Betweenness nodes. (b) the result of deploying Redundancy on the top 10% of Betweenness and Closeness nodes.

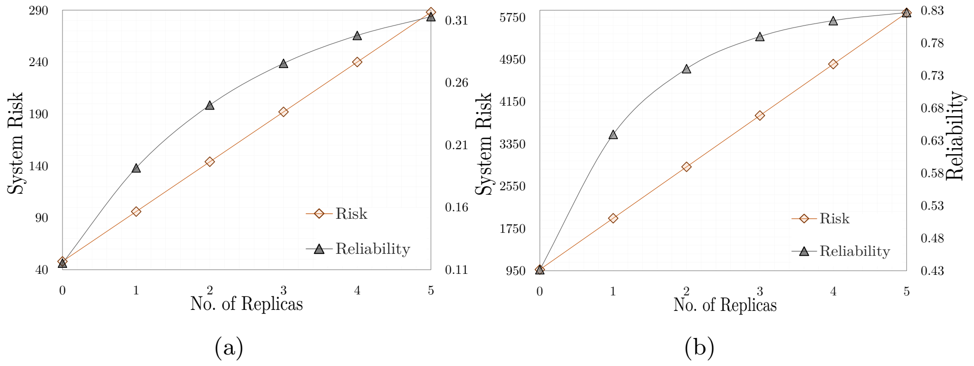


Figure 3.8: The Result of Deploying Redundancy on the Top 10% of Closeness Nodes (with 0-R to 5-R) on two Cloud-bands with (a) 20 VMs, (b) 50 VMs.

3.4.2 Shuffle Technique Analysis

We use VM-LM as the base technique of deploying Shuffle in our simulation. Migration of each VM from a host to another one may affect the overall security of the system. Thus, we consider the effects of deploying Shuffle technique on both Risk and Reliability metrics. We used HARM and SHARPE to compute Risk and Reliability respectively. Migration scenarios only affect the upper layer of HARM which is responsible for capturing connectivities. Thus, we only process the upper layer for Risk analysis and fed the upper layer into SHARPE in order to compute the Reliability values. It is obvious that if we consider all possible VM-LM scenarios and analyze the effectiveness of each migration through an ES method, we can obtain the best migration scenario. However, this evaluation is very time consuming and is not applicable for the large-sized networks. To address this problem, we use IMs to find the most important nodes in the network in terms of centrality. We analyze the correlation of each IMs, Betweenness and Closeness, against Shuffle technique. We then compare the results obtained from ES with those found through IMs.

We first deployed Shuffle technique on each VM and consequently evaluate the Risk and Reliability corresponding to that deployment using ES. Then, we repeated the evaluation by deploying Shuffle technique on those VMs having higher values of IM (Betweenness and Closeness). The results are demonstrated in Figure 3.6a showing (*i*) how deploying Shuffle can reduce Risk, (*ii*) whether the best scenario for deployed MTD technique can be obtained through IMs.

The results show that the best Shuffle deployment scenario minimizing Risk can be found through analyzing only the top 10% of the most important nodes based on Betweenness, but this method does not guarantee the best Reliability value. As it can be seen in the Figure 3.6a, the result of IMs analysis is equivalent with ES to find the optimal Shuffle deployment. However, deploying Shuffle so that it minimizes the Risk leads to a mild decrement on the Reliability which is negligible. Figure 3.6b demonstrates the Reliability values before and after deploying Shuffle.

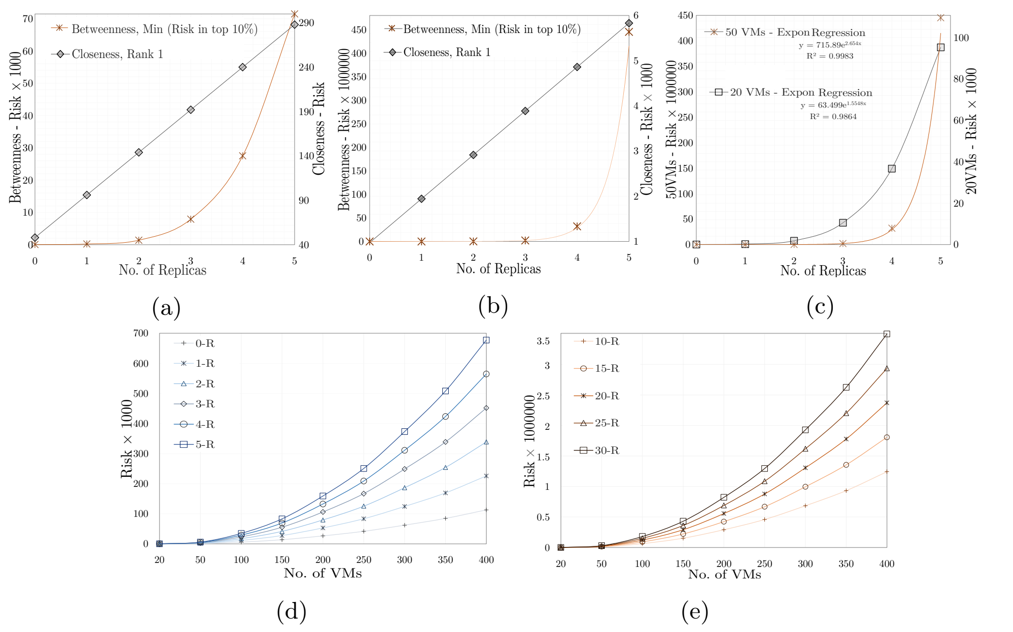


Figure 3.9: Comparing the result of deploying Redundancy technique and Risk

3.4.3 Redundancy Technique Analysis

We denote Redundancy technique with k replicas on a VM as k-R. Note that Redundancy technique affects both upper and lower layer of HARM, as it introduces new connections between VMs and new set of vulnerabilities for each created VM. Thus, both layer of HARM should be updated.

To analyze the effectiveness of Redundancy technique, (i) we perform a regression analysis to compare Risk and Reliability against the IMs (Closeness and Betweenness). We first calculate the values of Risk and Reliability after deploying Redundancy technique (with 3-R) for each VM through the ES. The upper layer of HARM is fed into SHARPE to obtain Reliability, then perform a regression analysis to show the correlation of each IMs with the corresponding Risk and Reliability values. Furthermore, for evaluation of deploying Redundancy, (ii) we investigate on whether the best values of either Risk or Reliability can be found through analysis of a portion of IMs to avoid using the Exhaustive Search (ES) methods. Finally, (iii) to what extent deploying Redundancy on those IMs can affect both Risk and Reliability metrics.

The results of regression analysis on deploying Redundancy in HARM are considered by comparing the correlation of Risk and Reliability against Betweenness and Closeness. We construct the HARM consisting of overall 50 VMs based on the Cloud-band model.

Then, behavior of the system is monitored after three hours has passed in order to calculate Reliability. We deploy three replicas (3-R) for each VM in the top layer of HARM in order to perform regression analysis. However, other VM sizes and different replicas are tested to compare the effects of Redundancy on both Risk and Reliability separately. Figure 3.5 reveals the correlation between Risk and Betweenness, an exponential line fitted through the data shows a high correlation (92%) between the values. Moreover, other correlations are reported in Table 3.3. We found a moderate linear correlation (70%) between Closeness with Risk. Other regression results among the IMs (Closeness, Betweenness) with Reliability are 0.24% and 0.28% respectively, that show no correlation between IMs and Reliability. Next, we deploy Redundancy up to 5 replicas on the VMs with top Betweenness and Closeness values separately to analyze the effects of these deployments in term of Risk and Reliability. This analysis is based on two scenarios, (i) considering the trend of both Risk and Reliability against the number of replicas based on Betweenness and Closeness, Figures 3.7a and 3.8 respectively, (ii) to investigate whether we can find the optimal Reliability value through IMs compared with ES, see Figure 3.7b. Then, we perform other analysis comparing the trend of the only Risk when deploying Redundancy on a VM with the highest Closeness and Betweenness values with various number of VMs and replicas, Figure 3.9.

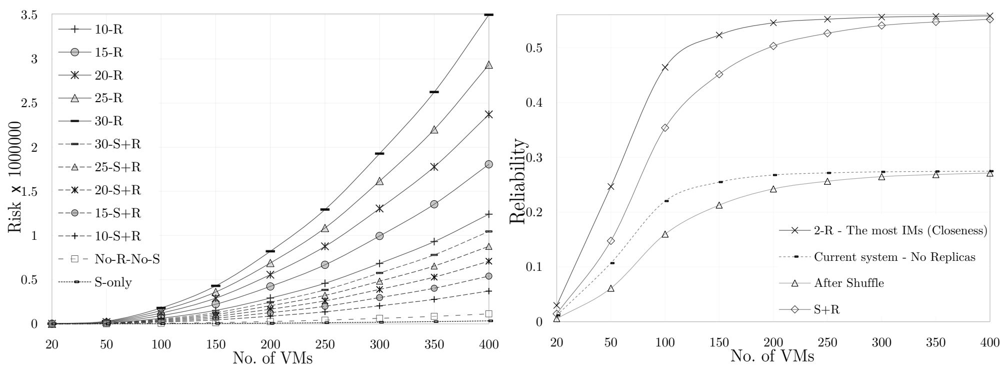
Scenario (i). Figure 3.5 depicts the correlation of Betweenness with Risk obtained based on 3-R, as it can be obviously seen, there is a very high exponential correlation about 82% between those values. Thus, we can conclude that deploying Redundancy on the nodes with high Betweenness causes very high Risk value.

Scenario (ii). Figure 3.7a compares the growth rate of Risk against Reliability with a various number of replicas on the cloud-band consisting 50 VMs. We deploy the Redundancy to the top 10% of VMs with the highest Betweenness values. The trend of Risk value is highly exponential, but Reliability grows logarithmic. However, we will later show that the best value of Reliability cannot be found through deploying Redundancy on a VM or a portion of them with the highest Betweenness values.

Scenario (iii). Figure 3.8 compares the same factors as Figure 3.7a through two charts showing cloud-bands with 20 and 50 VMs, but it only considers the growth rate of Risk and Reliability resulted from deploying Redundancy on only a VM with the highest Closeness value. We observe that by increasing replicas Risk grows linearly (this rate for Betweenness is exponential) and Reliability goes logarithmic.

Scenario (iv). Figure 3.7b compares the results obtained through an ES with two groups of IMs, first, top 10% of VMs having higher Betweenness values and, second, a VM with the highest Closeness value. As it can be clearly seen, the best values obtained through ES correspond with the latter group. Thus, the best Reliability value can be obtained via deploying Redundancy on the first rank of Closeness.

Scenario (v). Figure 3.9 compares only Risk values under different conditions. Figures 3.9a and 3.9b reveal the trend of increasing Risk through both Betweenness and Closeness



(a) (b)

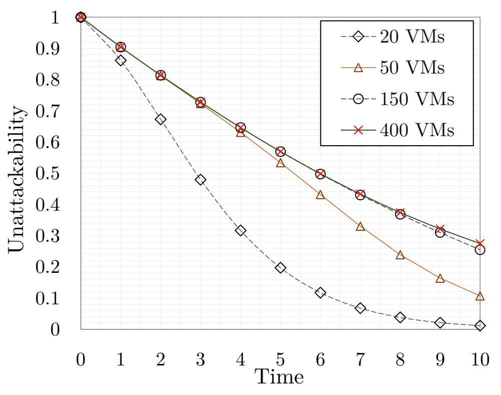
Figure 3.10: Comparing different combinations of MTD techniques with regard to Risk and Reliability. (a) comparing the combinations of MTD techniques based on different Cloud-band sizes and replicas over the Risk. (b) comparing the effects of deploying Shuffle, Redundancy (2-R), and S+R techniques on the Reliability.

in two different cloud-band sizes including 20 and 50 VMs respectively. Figure 3.9c shows a deeper analysis to show how fast Risk grows on those two cloud-bands when we deploy Redundancy based on Betweenness, these rates are 98% and 99% for 20VMs and 50VMs respectively. Nevertheless, in Figures (3.9d, and 3.9e) we see that the number of replicas are increased up to 30-R and the number of VMs up to 400 VMs, and observe that trend of increasing Risk remains linear when we choose a VM with the highest Closeness value for deploying Redundancy.

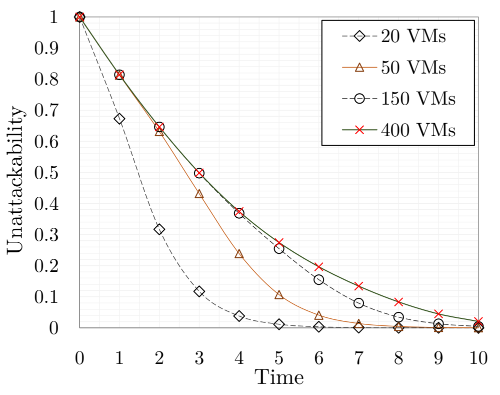
To conclude, deploying Redundancy technique increases the Reliability of a system. The best deployment scenario can be found by analyzing Closeness. The noticeable point is that Reliability obtained through this deployment grows logarithmic while in the worst case (if we use Betweenness as IM) it causes an exponential growth in Risk and in the best case (using Closeness) we have a linear increment in Risk. Hence, Redundancy technique should be deployed precisely based on the network's size and specifications. For instance, in our cloud-band example, the best number of replicas to enhance Reliability with a reasonable Risk value is 5-R based on Figure 3.8, and more replicas on that particular VM do not improve Reliability while it increases the system Risk.

3.4.4 Analysis of S+R MTD Techniques

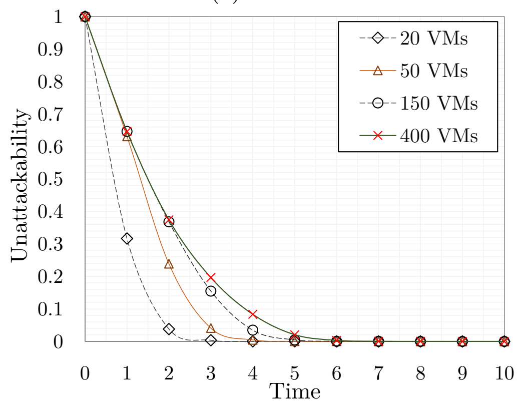
According to results of the previous sections, Shuffle can decrease the Risk of a network while Redundancy enhances the Reliability. Thus, satisfying those metrics are valuable in a network, especially in large sized networks or cloud environments. In this section, we consider the combination of both Shuffle and Redundancy denoted as S+R, and then evaluate the effectiveness of this technique. Based on the experimental results obtained from the previous sections, we develop the S+R together with IMs so that we deploy



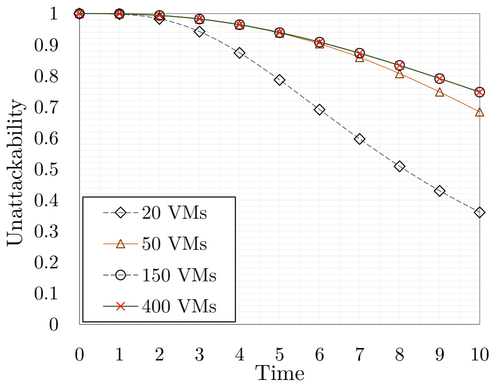
(a) λ=0.1



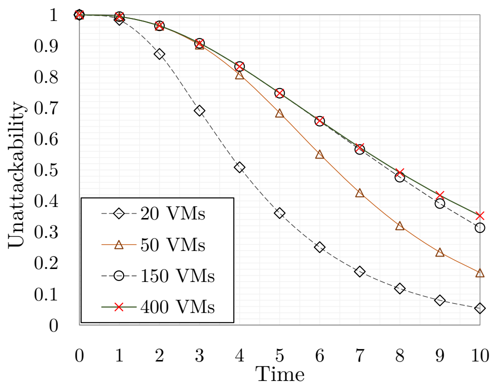
(b) λ=0.2



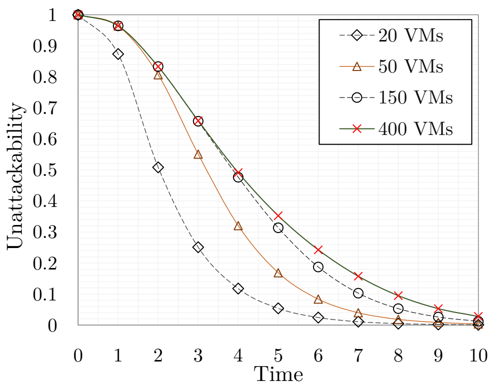
(c) λ=0.4



(d) λ=0.1



(e) λ=0.2



(f) λ=0.4

Figure 3.11: Comparing the result of deploying S+R on the Unattackability based on three attack rate values: 0.1, 0.2, and 0.4. (a), (b), and (c) the Unattackability values of the cloud before deploying S+R under the given attack rates. (d), (e), and (f) changes in Unattackability after deploying S+R.

Shuffle among the top 10% of VMs having the highest Betweenness values, also we deploy Redundancy on the most important VM ranked by the Closeness value. Then, we show that $S{+}R$ can provide a trade-off between the Risk and Reliability. We assess the effectiveness of the system after deploying $S{+}R$ with a various number of replicas and different number of VMs in each cloud-band. The obtained results of $S{+}R$ are compared with other deployments like Shuffle only, Redundancy only, and No-R No-S (current system). Figure 3.10a compares the growth trend in the Risk against different cloud-band sizes and replicas by deploying all combinations of foregoing MTD strategies. In order to analyze the effects of deploying $S{+}R$ on Reliability, and compare it with other deployment scenarios, we replicate the most important VM using Closeness and find the best Shuffle scenario using Betweenness (as in Subsection 3.4.2), see Figure 3.10b.

As shown in 3.10a, deploying Shuffle only (S-only) can decrease the Risk (compare S-only with No-R-No-S in the chart). Next, deploying Redundancy only (R-only) increases the system Risk. Nevertheless, deploying $S{+}R$ causes a gentle increment on the Risk which is not comparable with the same values caused by deploying Redundancy only. In Figure 3.10b, comparing the current system with the results of deploying both Redundancy and $S{+}R$, we obviously observe that both of these techniques enhance the Reliability, while Shuffle decreases Reliability. Then, we conclude that the two important system Risk and Reliability metrics have a negative correlation toward MTD techniques. Although increasing the Reliability through deploying Redundancy may also increase Risk and vice versa, one can benefit from a combination strategy to find a reliable threshold between those two metrics based on the particular system and cloud environment.

System Unattackability. System Unattackability metric measures how tolerant the cloud is against various attack rates. Let X be a random variable that represents the time to launch a successful attack on a single VM. Then, *Attackability* of a VM at time t is the probability that the VM $vm_i \in VM$ be attackable (a successful attack has occurred) in the interval $[0, t)$, and can be represented as $A_{vm_i}(t){=}P(X < t)$. We assume that the times to a successful attack on each VM follows an exponential distribution. Then, the Attackability of the whole cloud system (A_{cs}) depends on the Attackability on each single VM in the upper layer of HARM. Let $A_{cs}(U_{cs}, \lambda, t)$ be a function on the upper layer of HARM which determines the probability that the cloud system be attackable under the given attack rate and time parameters, where U_{cs} is the upper layer of HARM and λ is the attack rate. Then, the upper layer can be represented as $U_{cs} = (VM, E, \sigma)$, where σ is a mapping function assigning the λ value to each edge $\sigma : E \rightarrow \{\lambda\}$. An attack path of length ℓ in U_{cs} is a sequence of VMs $path = (vm_0, vm_1, ..., vm_{\ell-1}, vm_\ell)$ such that for each $i = 1, 2, .., \ell$ there is an edge with the assigned λ value between vm_{i-1} and vm_i. Then, a system is attackable if every single VM $vm_i \in path$ is compromised which shows the probability of the existence of an attack from a source to a target (DB). Then, we define the system Unattackability as the probability that the cloud system be unattackable under a given attack rate over a period as $1 - A_{cs}(H_{cs})$. In order to compute the overall Unattackability of the cloud, we used SHARPE and fed the U_{cs} to the SHARPE and set the λ value together with a source and a target to evaluate the overall Unattackability of

U_{cs} by considering all possible attack paths.

In practice, the actual attack rate observed can be near impossible to predict, our model provides a method to evaluate various cloud systems with different attack types of varying attack rates. Because our model is generalized, we show how the security posture of the cloud may change with respects to the varying attack rates.

We measured Unattackability over with different attack rates 0.1, 0.2, and 0.4, respectively. Figure 3.11 plots the Unattackability values for before (as shown in Figures 3.11d, 3.11e, and 3.11f) and after deploying S+R techniques (as shown in Figures 3.11a, 3.11b, and 3.11c). We evaluate the results of deploying S+R techniques on the cloud based on two scenarios, (*i*) increasing the number of VMs and evaluating the system Unattackability before and after MTD deployment, (*ii*) increasing the attack rates and evaluate the system Unattackability.

Scenario (i). Figure 3.11 shows the system Unattackability postures of the cloud system before and after deploying MTD techniques over different attack rates with different cloud sizes. First, the results show that the system Unattackability follows a descending trend for all scenarios over time. The results show that the clouds having lower number of VMs tend to be more attackable than the clouds including higher number of VMs for both before and after deploying S+R. Moreover, the descending trends on Unattackability values after deploying S+R are lower than the values before deploying S+R. For instance, the system Unattackability for a cloud having 20 VMs is too low in time 5 which is about 0.2, see Figure 3.11a, while this value is more than 0.7 in time 5 after deploying S+R, see Figure 3.11d.

Scenario (ii). Increasing the attack rates reduces the system Unattackability over time. As it can be seen in Figures 3.11a, 3.11b, and 3.11c, the values of Unattackability for the clouds having 400 VMs under various attack rates λ=0.1, λ=0.2, and λ=0.4 in time 5 are about 0.6, 0.3, 0, respectively. However, these values after deploying S+R have higher rates and are about 0.9, 0.8, and 0.4 respectively, see Figures 3.11d, 3.11e, and 3.11f.

Finally, the results demonstrate that deploying S+R technique can secure the cloud by increasing the Unattackability threshold over time. Moreover, the results show that increasing the number of VMs in cloud-band can increase the Unattackability values over time.

3.5 Discussion and Limitations

MTD techniques have been proposed to prevent adversaries to penetrate in a cyber environment. In this chapter, we evaluated three different MTD techniques: Shuffle, Redundancy, and the combination of both. However, deploying each technique resulted in various changes to the system Risk and Reliability. To compare their effectiveness, we adopted a scalable security model, named HARM, together with using two NCMs, Closeness and Betweenness. We simulated a large-sized cloud-band model and compared the changes in the system Risk and Reliability of each deployed MTD technique.

The experimental analysis in Section 3.4 showed that the best Shuffle technique (that minimize the Risk) can be found using the IMs with only the top 10% of VMs. Although deploying Shuffle decreased the Reliability, this decrement is neglectable (especially in the larger cloud-bands) as shown in Figures 3.6b and 3.10b. When deploying the Redundancy technique, Betweenness measure has a strong exponential correlation with the Risk. It shows that deploying Redundancy technique on the nodes with higher Betweenness values increases Risk exponentially. Although Betweenness works well for finding the best Shuffle deployment, replication of the nodes with higher Betweenness leads a heavy increament on the Risk. As the Redundancy technique aims to improve the system Reliability, we observed a trade-off between the Risk and the Reliability when using the Redundancy technique. The second finding is that Betweenness has no correlation with Reliability; thus, replication of a VM with highest Betweenness centrality does not guarantee the best Reliability. On the other hand, one can deploy Redundancy on a VM with the highest Closeness rate to achieve the best Reliability value while Risk grows linearly. Finally, combining S+R can help to find an appropriate threshold between Reliability and Risk metrics based on the security levels required by cloud providers. Moreover, we estimated system Unattackability, our proposed model can suggest the potential attack rate the system could withstand by testing against a range of attack rates, and security hardening suggestions to improving the security posture to withstand higher attack rates.

The limitations of this study are as follows. The observed results are valid based on our cloud-band model and may vary on the different type of networks and topologies, and cloud models [108, 129]. Other MTD combinations including Diversity technique should also be considered with more metric analysis. We only focused on three Risk, Reliability, and Unattackability metrics for evaluation, while there are many other security metrics, like Attack cost, Attack impact, Return-on-attacks, Normalized value, *etc* [157]. Moreover, we only analyzed the OS level vulnerabilities in the VM layer, but different vulnerabilities from other layers of the system (e.g. application vulnerabilities) should also be considered. The cost of deploying MTD techniques is another perspective that needs to be considered to show the economic impact against the achieved security. In this study, we assumed that the attacker is outside of the system, but considering attacks from inside the cloud is also important (like attacks resulted from VM co-residency, *etc.*) [72]. Finally, this model should be implemented in a real cloud testbed to be evaluated based on the real environment.

We will incorporate other MTD techniques and combinations to evaluate the effectiveness of each combination scenarios in Chapters 4 – 5. Further, we will conduct experiments using a real cloud testbed. We will use a private cloud named UniteCloud [1], in Chapter 6, to implement our work and evaluate the techniques in a real cloud infrastructure.

3.6 Conclusion

MTD techniques have been proposed to enhance the cybersecurity by making changes on the attack surface to make it unpredictable, and consequently confuse the attackers. However, deploying MTD techniques need to be evaluated as it may affect the system Risk and Reliability. Thus, it is important to evaluate the effectiveness of the MTD techniques prior to deploying the strategies. Moreover, by computing security metrics, we can quantify the effectiveness of each MTD technique separately or in combinations. To conduct the study, we first formalized MTD techniques based on the scalable HARM to combines Shuffle, Redundancy, and the combination of both. Then, we evaluated the effectiveness of MTD techniques with regards to analyzing security metrics. We considered three metrics for evaluation: Risk, Reliability, and Unattackability. To improve the security analysis through HARM, we utilized NCMs to find IMs: Closeness and Betweenness to rank the most crucial VMs in the cloud. Then, we showed how the IMs can improve the security analysis for evaluating MTD techniques. Finally, the experimental results revealed that we can combine Shuffle and Redundancy techniques to minimize the Risk while we increase the Reliability metric and hold Unattackability in an acceptable threshold. However, we showed that a single MTD technique like Shuffle only or Redundancy only cannot satisfy all metrics in a desirable level.

Chapter 4

Model-based Evaluation of Combinations of Shuffle and Diversity MTD Techniques on the Cloud

Summary

Regardless of cloud computing capabilities, security is still one of the biggest threats in the cloud. In this chapter, we propose a combination of two MTD techniques: Shuffle and Diversity which we believe further attributes to reduce the cyber attack surface. We first provide the formal definitions of the combination to design and implement our proposal. Then, we investigate a number of approaches in which Shuffle and Diversity can be combined in order to provide the most effective defense. Towards, we utilize Network Centrality Measures (NCMs) to find out the most critical component in the cloud. Then, we evaluate the proposed MTD techniques through formal Graphical Security Models (GSM) and quantify the cloud security level through eight useful security metrics before and after deploying the MTD techniques. Our experimental evaluation shows that the combination of Shuffle and Diversity techniques can increase the security posture of the cloud.

"This chapter is the peer reviewed version of the following article: "Alavizadeh, H., Kim, D. S., and Jang-Jaccard, J. Model-based evaluation of combinations of Shuffle and Diversity MTD techniques on the cloud. Future Generation Computer Systems (2019)", which has been published in final form at `https://doi.org/10.1016/j.future.2019.10.009`. This article may be used for non-commercial purposes in accordance with Elsevier Terms and Conditions for Use of Self-Archived Versions."

4.1 Introduction

Cloud computing security has become a huge challenge for the cloud providers as the cloud's customers cannot trust the security of this new paradigm while the cloud provides comprehensive services to their customers. According to the International Data Corporation (IDC) survey on the cloud computing challenges, the cloud security with 87.5% was ranked first as the greatest concern for the enterprise cloud customers [123]. Conventional security mechanisms are used to address the security issues by eliminating the vulnerabilities and risks. However, it is difficult to perfectly remove or patch all possible vulnerabilities on a system. Hence, it is crucial to have effective security mechanism to improve the cloud security from different defensive aspects [79, 167]. As an emerging proactive approach, Moving Target Defense (MTD) has been proposed which can provide another perspective of defensive strategies against cyber attacks. MTD makes a system more unpredictable for the attackers by continuously changing the attack surface. MTD can utilize the existing system components and technologies providing more affordable defense solutions.

As stated earlier, Hong *et al.* [66] categorized MTD techniques into three comprehensive categories including [66]: Shuffle [75, 151], Redundancy [154] and Diversity [110, 127]. In general, Shuffle MTD techniques can reconfigure the system's components in order to change the attack surface and consequently increase uncertainty and confusion for the attacker. Redundancy MTD techniques deal with replication of any system's component aiming to enhance the system and service reliability or availability for the customers. Thus, if a system's component fails due to attack, there would be alternative ways to provide the same service. The advent of the Internet of Things (IoT) makes more viable attack sources for attackers so that they can launch various attacks through the botnets these days. Botnets are good starting points for attackers to launch a wider attack range to the cloud using vulnerable IoT-based botnets [83]. For example, a Distributed Denial-of-Service (DDoS) attack utilizes botnets (leveraging compromised IoT devices) to attack a cloud by flooding traffic messages from various sources aiming to deny services to users. In this case, Redundancy contributes to increase cloud resiliency, and can battle these kinds of attacks. However, the investigation of Redundancy technique is out of the scope of this chapter and is presented in Chapters 3 and 5. Diversity MTD techniques may increase the difficulties of attacks by changing the system's component variants. Changing a component in the system may introduce different and new set of vulnerabilities and invalidate the vulnerability information collected by the attackers. Ultimately, the attacker may spend more time, effort, and money to learn new techniques to exploit the newly introduced vulnerabilities.

Deploying an MTD technique for a specific reason may vary the security posture of a system. Most of proposed MTD techniques do not offer convincing evidence if they would be effective as claimed. Therefore, it is important to assess the effectiveness of MTD techniques through security metrics such as Attack Cost (AC) and Return on Attack

(RoA) which evaluate the security from the attackers' perspective and other metrics like System Risk (Risk) and Attack Success Probability (ASP) which may be desirable metrics for cloud providers' perspective. Security analysis plays an inevitable role in evaluating the overall security-related perspectives of a system.

Formal graphical attack models like Graphical Security Models (GSMs) are useful tools to model and evaluate the security of the systems such as IoT and enterprises [53] or clouds [112]. GSMs can be used to evaluate the effectiveness of MTD techniques [13,66]. However, analyzing the security through most of GSM suffers from exponential computational complexity issue, especially, in the large networks [61]. To overcome this problem, the Hierarchical Attack Representation Model (HARM) is proposed which is a formal hierarchical graph-based model including two layers [61]. HARM is more scalable and adaptable than other formal GSMs [62]. In this chapter, we use HARM to evaluate the effectiveness of the MTD techniques and compute the security metrics.

MTD techniques can be either used independently or combined together to obtain more effective results. Many MTD strategies have been proposed [103, 160], but it is still difficult to evaluate the effectiveness of combined MTD techniques. Combining MTD techniques can introduce additional benefits of enhancing security, which may not be possible under a single technique based MTD solutions; for instance, as Redundancy is mostly used to increase service reliability, it can be measured with the concepts of system dependability (e.g. reliability), while other MTD techniques like Shuffle and Diversity are used to increase the security of a system and need to be evaluated using security metrics. Thus, MTD techniques can be well-mingled together aiming to increase both security and reliability. However, those techniques should be evaluated using adequate security metrics as deploying each MTD technique may affect others in different ways. A combination of MTD techniques including Shuffle, Diversity, and Redundancy is presented in [11] which is mainly limited to a single deployment strategy and four security metrics such as Risk, AC, RoA, and Reliability. However, in this Chapter, we conduct extensive analysis on Shuffle and Diversity MTD technique which considers different sets of combination strategies. Moreover, we capture the effects of different MTD deployment strategies using eight important security metrics.

The earlier published version of this chapter is presented in [12]. In this chapter, we extend the earlier version mainly focusing on formalism and definitions of MTD techniques and combination strategies on the cloud together with more effective security metrics to show the different perspective of the cloud security posture affected by deploying MTD techniques. Moreover, we have revised the previous model with new vulnerabilities and metrics. The new contributions of this Chapter, which to the best of our knowledge have not already been proposed by other works, are listed as follows:

- We provide the formal mathematical definitions for the combination of Shuffle (S) and Diversity (D) MTD techniques to unambiguously design and implement it in the cloud.

Our formal method is written based on Hierarchical Attack Representation Model (HARM).

- We propose a new approach that combines Shuffle (S) and Diversity (D) techniques. We also provide a set of strategies for the way Shuffle (S) and Diversity (D) can be combined differently. By computing Important Measures (IMs), the effects of the different combination strategies are calculated and explained.

- We provide simulation and calculation results for the deployed MTD techniques using four security metrics to assist in extensive understanding of the trades between MTD techniques and Metrics involved in the combined defense and the attack. The security metrics we use include: Risk, Attack Cost (AC), Return on Attack (RoA), and Attack Success Probability (ASP).

- We also include the path-based security metrics and evaluate them against each MTD strategy to evaluate how difficulty is required for the attacker can reach a target. We also conduct regression analysis between path-based metrics and security metrics (Risk and ASP) to investigate the correlation between those metrics.

- We perform comparative analysis and evaluation of "before" and "after" deployment of MTD techniques to be able to quantify and compare the cloud security posture.

The rest of this chapter is organized as follows. We present the related work in Section 4.2. We define the concepts, definitions, formalism, and the security metrics used throughout this chapter in Section 4.3. In Section 4.4, we provide definitions and formalism for MTD techniques including Shuffle and Diversity and evaluate the deployment of each MTD technique. Definition, deployment, and analysis of different strategies for combining MTD techniques are given in Section 4.5. Then further discussion and limitations are given in Section 4.6. Finally, we conclude the chapter in Section 4.7.

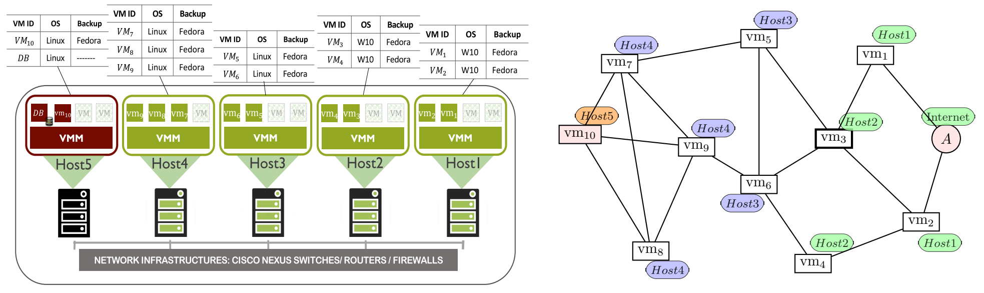
4.2 Related Work

Definition of MTD is not restricted to a portion or a specific part of a system. Any static or dynamic component of a system can be changed using MTD techniques to make a system more unpredictable for attackers. MTD can be deployed through different layers of a network. Numerous research have been proposed either to introduce new techniques or improve an MTD model [75, 138]. Many researchers have focused on MTD frameworks [164,169], applications [33,150], strategies and techniques [8,75,162]. However, we summarized the proposed MTD techniques based on different contexts.

Redundancy can be deployed on the cloud using replication of the cloud resources (such as VM replication). This technique can increase the reliability of the cloud by creating redundant VMs (e.g. crucial servers). The extension of cloud resources to increase the service reliability and battle against DDoS attacks have been investigated in [154]. However, deploying redundant cloud resources to avoid attacks may affect cloud's performance. To address this problem, Al-Haidari *et al.* [7] studied the impact of cloud scaling size factors against the CPU cost and performance. They used the optimization problem to find an optimal number of cloud resources against QoS requirements. In [29], the authors proposed a resource allocation scheme for the virtual desktop cloud (VDC) and developed a tool named VDC-Analyst aiming to satisfy quality of experiments (QoE) for users by increasing net utility and service response time. The concept of Random Route Mutation (RRM) has been introduced by Al-Shaer [8] to find an optimal randomized path between source and target. However, this technique can be categorized as a Shuffle technique deployed on the service paths in a network. The application of MTD techniques to mitigate DDoS attacks on the cloud using Software-defined networking (SDN) has been studied in many research [75, 141]. Moreover, Bawany *et al.* [22] conducted a substantial survey to study and classify the proposed SDN-based DDoS attack detection and mitigation techniques. They also proposed an SDN-based proactive DDoS Framework (ProDefence) for detection and mitigation of DDoS attacks in a large-scale network. However, these types of MTD techniques can be combined with other techniques to double the effectiveness of the proposed methods in terms of both service availability, reliability and security.

Machine Learning-based MTD (ML-based MTD) techniques have been proposed by researchers in various studies [43, 44, 81, 148, 151]. For instance, Vikram *et al.* [151] proposed a Shuffle MTD technique on the application layer for securing the web by randomizing the HTML elements using Machine Learning (ML). ML-based MTD enables a defensive system to capture evolving attack patterns with high scalability and applicability. The ML-based techniques can be affected by the lack of a large amount of data for training reasons to provide an acceptable prediction accuracy level. Data-driven incident prediction methods play a crucial role in addressing such ML-based techniques. In [143], the authors surveyed the emerging research for cybersecurity incident prediction focusing on data-driven methodologies. They categorized the related research into six data types category such as the organization's report, dataset, network dataset, synthetic dataset, web page data, social media data, and mixed-type dataset. Anomaly detection techniques and Intrusion Detection System (IDS) can be incorporated with MTD techniques [134]. Network traffic classification is useful methods which can be used for various purposes such as anomaly detection on the networks. In [161] the authors proposed a framework for network traffic classification which can be adapted using very few training samples but high performance. However, this technique can be implemented and used by trigger-based MTD techniques.

Most of the previous works only focused on the novelty on the proposed strategies and layers of implementation, like defensive method after detecting an attack, low-level shuffling techniques, finding a suitable time-period for applying the IP mutation frequently,



(a) (b)

Figure 4.1: A Cloud system example: (a) The infrastructure layer of a private cloud. (b) The Cloud example Model showing the connection of the VMs.

dealing with worms and web bots through MTD [151], and so on. However, there are very few works effectively analyzing the MTD techniques for the large networked system needing precise and scalable security analysis, like cloud-based environments. Peng *et al.* [119] investigated the effectiveness of MTD techniques for securing cloud-based services with a heterogeneous or dynamic attack surface. However, they did not utilize a rational or formal security model and analysis tool to evaluate the effectiveness of the deployed strategy. Hong *et al.* [66] introduced Shuffle, Redundancy, and Diversity MTD techniques separately and analyzed the varies on security when MTD techniques are deployed. However, other combinations of MTD techniques should be also considered like deploying a combination of Shuffle and Diversity, and analyzing more security metrics aiming to cover more security requirements of the cloud.

4.3 Preliminaries

In this section, we explain and recall some required notations, concepts and definitions used throughout this chapter, such as system setting, configuration and constraints, security metrics, and other related assumptions through a running example in a cloud system.

4.3.1 System and Threat Model

As a running example for the rest of the chapter, we assume a private cloud consisting of five main hosts (servers) each of which can hold up to four active Virtual Machines (VMs). We assume that only two VMs on the first host are allowed to connect to the Internet and only the last host (Host5) is connected to the critical Database (DB), as shown in Figure 4.1. Each VM includes a default operating system (OS) and a backup one shown on the top of each server in Figure 4.1. VMs hosted in the Host1 and Host2 are installed with Windows 10, and VMs in the other hosts are installed with Enterprise Linux OS. We assume that an attacker is outside the private cloud and can exploit the

Table 4.1: OS Vulnerabilities (V) including Base-Score (BS), Impact (I), Exploitability (E), and Attack Cost (AC) 4.2

OS (θ)	V	CVE-ID	BS	I	E	AC
Win10	$\nu_{1,\mathrm{W}}$	CVE-2018-8490	8.4	6	0.17	1.6
	$\nu_{2,\mathrm{W}}$	CVE-2018-8484	7.8	5.9	0.18	2.2
	$\nu_{3,\mathrm{W}}$	CVE-2016-3209	8.8	5.9	0.28	1.2
Linux	$\nu_{1,\mathrm{L}}$	CVE-2018-14678	7.8	5.9	0.18	2.2
	$\nu_{2,\mathrm{L}}$	CVE-2018-14633	7	4.7	0.22	3
	$\nu_{3,\mathrm{L}}$	CVE-2017-15126	8.1	5.9	0.22	1.9
Fedora	$\nu_{1,\mathrm{F}}$	CVE-2014-1859	5.5	3.6	0.18	4.5

vulnerabilities of those operating systems to gain access. In addition, we also assume that the hypervisors provide enough isolation for VMs hosted on each server. The goal of the attacker is to compromise the Database (DB) in the Host5. The system's configurations, connectivities and constraints are assumed as follows[1]):

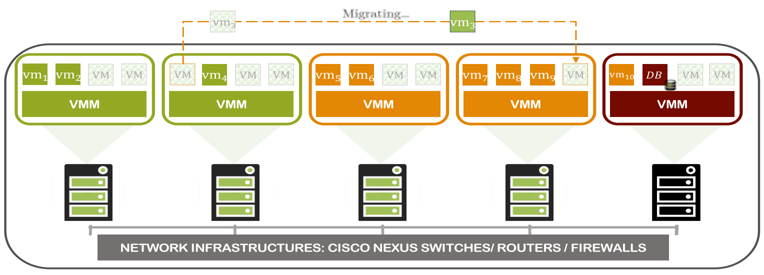
- All VMs are active and never suspended
- A VM either can migrate to another host or be launched with its backup OS; the downtime for those processes are negligible
- Two vm_1 and vm_2 are connected to the Internet (entry points of the cloud)
- vm_{10} on $Host5$ cannot be migrated due to system constraints
- Only vm_{10} can access the Database (DB) which makes the vm_{10} a target to the attackers

Table 4.1 shows the vulnerabilities for different OS: Win10 (W), Linux (L), and Fedora (F). Here, we only modeled the vulnerabilities that can bypass firewalls and authentications. There are three of those vulnerabilities for both Windows OS and Linux OS, together with a single vulnerability for Fedora OS. Further information regarding these vulnerabilities and measures can be found in the National Vulnerability Database (NVD) [106].

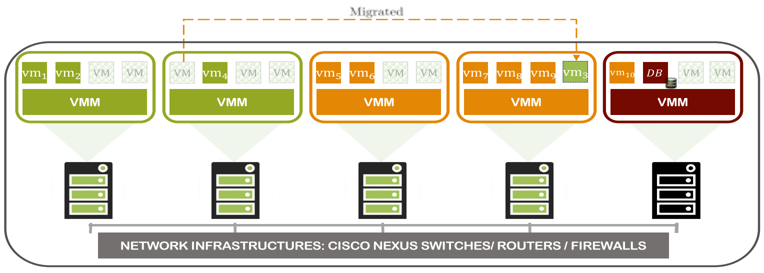
4.3.2 Defensive MTD Model

Generation of a defensive model is important for providing appropriate security strategies. The defensive model can analyze the current system and make the decision for deploying the best MTD techniques based on the system requirements, constraints, and limitations. We define a defensive model which can follow the following steps: (1) Model creation (in here, the model we create is HARM for computing the security metrics), (2) Evaluation

[1]These assumptions are used for simplicity in model description and could be released.


Migrating...
vm3
vm1
vm2
VM
vm4
vm5
vm6
vm7
vm8
vm9
vm10
DB
VMM
NETWORK INFRASTRUCTURES: CISCO NEXUS SWITCHES/ ROUTERS / FIREWALLS

(a) Migrating


Migrated
vm1
vm2
VM
vm4
vm5
vm6
vm7
vm8
vm9
vm3
vm10
DB
VMM
NETWORK INFRASTRUCTURES: CISCO NEXUS SWITCHES/ ROUTERS / FIREWALLS

(b) Migrated

Figure 4.2: VM-Live Migration on the Cloud. (a) Migrating Phase. (b) Migrated.

of current cloud security posture, (3) MTD analyzer to find the best strategies, (4) MTD Deployment on the cloud. We assume that the following operations are permitted by the cloud provider. We then use these operations for deploying MTD techniques.

Virtual Machine Live Migration: Virtual Machine Live Migration (VM-LM) can be enabled by the cloud provider, and a VM can migrate from one physical host to another one if there is enough space in the new host. Figure 4.2 shows an example of VM-LM technique on the cloud example. We use VM-LM as the main approach of deploying Shuffle technique in the cloud. However, other approaches can be used as Shuffle techniques like Virtual IP mutation, Port Hopping , *etc.* [137] which are out of the scope of this chapter.

OS Diversification: Changing the operating system is the process of using a backup OS instead of the default OS on each VM in the cloud. OS Diversification can be used as a Diversity technique [11]. We assume that the probability of failure in launching a new OS is negligible. Changing the OS may introduce a new set of vulnerabilities to the attacker. Other Diversity methods can be utilized such as changing applications and services running on the VMs, changing programming languages, etc. However, in this study we only consider OS Diversification to deploy Diversity.

We deploy VM-LM and OS Diversification through simulation. In this study, the focus is not on implementation on the real cloud but the theoretical appraisal and formalism, and only considered theoretical analysis and evaluation through simulation to assess the effectiveness of combining MTD techniques. However, the feasibility, adaptability, and, usability of those MTD techniques on a private cloud, named UniteCloud [1], are practically considered in [9].

4.3.3 GSM Models

HARM is proposed by Hong *et al.* which is a scalable GSM for analyzing the security of the systems through two layers. HARM has been used in many studies needing scalable security analysis like enterprise security analysis [49, 156], cloud computing, IoT, and MTD [53, 66]. In this study, we use HARM for security analysis and evaluate the MTD techniques. HARM separates the networks' connectivities and vulnerabilities into two layers so that reachability of the network's components is captured in the upper layer and the vulnerabilities existing on each component can be captured in the lower layer. We utilize HARM to capture the connectivities of VMs on the cloud in the upper layer and OS vulnerabilities excising on each VM in the lower layer of HARM. Constructing the HARM, we can perform the security analysis and compute the security metrics.

We recall HARM defined in Definition 1 presented in Chapter 3.

Definition 1. HARM can be defined as a 3-tuple $H = (U, L, C)$ where U refers to the upper layer which is an Attack Graph (AG), and L represents the lower layer in which an Attack Tree (AT) is constructed. We define $C = U \to L$ as a one-by-one mapping of the upper layer to the lower layer. Then, the upper layer of HARM is defined as a graph $U = (\mathrm{VM}, \mathrm{E})$, where $\mathrm{VM} = \{vm_1, vm_2, \ldots, vm_n\}$ is a set of VMs in the cloud, with $|VM| = n$, and $E \in VM \times VM$ is a set of connectivities between VMs.

We reformulate the lower layer of HARM in a way that it includes a set of OS for deploying Diversity as follows.

Definition 9. The lower layer L is a set of ATs corresponding to each VM vm_i in the upper layer and can be defined as $L = \{\ell_1, \ell_2, \ldots, \ell_n\}$, where $\ell_i = (V_{i,\theta}, G, root)$, and ℓ_i is an AT corresponding to the vm_i. Then, $V_{i,\theta} = \{\nu_{1,\theta}, \nu_{2,\theta}, \ldots, \nu_{m,\theta}\}$ is a set of vulnerabilities existing on each corresponding OS $\theta \in OS$ on each VM vm_i, where $OS = \{W, L, F\}$. We denote the number of vulnerabilities in each VM (specifically, on each OS) as $|V_{i,\theta}| = m$, and G is a set of logical gates $G = \{AND\text{-gate}, OR\text{-gate}\}$ constructing the inner nodes of the AT, and $root$ is the corresponding node in U.

Figure 4.3a demonstrates the generated HARM for the cloud system example presented in Figure 4.1. We can calculate the security metrics including Risk, AC, RoA using the generated HARM. We then can compute and evaluate the overall security of the cloud by considering both sides: the cloud provider and the attackers. In the upper layer of the HARM, an AG is used, and in the lower layer an AT is used, such as in [66]. Figure 4.3a shows the constructed HARM for the example cloud system.

Hence, the generated HARM for the cloud system illustrated in Figure 4.1 is presented as follows.

Example 2. The HARM for the Cloud system example (cs) is shown as $H_{cs} = (U_{cs}, L_{cs}, C_{cs})$. Then, $U_{cs} = (VM_{cs}, E_{cs})$, where $VM_{cs} = (A, vm_1, vm_2, \ldots, vm_n)$. The VMs are connected as $E_{cs} = \{(A, vm_1),$

Table 4.2: Some notations, explanations and supporting examples or references

Notations		**Description**	**Formula, Reference**
		General	
VM		The set of all Virtual Machines (VMs) in the cloud, $\|VM\| = n$	$VM = \{vm_1, vm_2, \ldots, vm_n\}$
$V_{i,\theta}$		The set of all vulnerabilities existing on i^{th} VM, i.e. vm_i, $\|V_{i,\theta}\| = m$	$V_{i,\theta} = \{\nu_{1,\theta}, \nu_{2,\theta}, \ldots, \nu_{m,\theta}\}$
$\nu_{j,\theta}$		The j^{th} vulnerability in a single VM, $\nu_{j,\theta} \in V_{i,\theta}$	Table 4.1
$I_{\nu_{j,\theta}}$		The impact of exploiting the j^{th} vulnerability in a single VM, $\nu_{j,\theta} \in V_{i,\theta}$.	Table 4.1
$E_{\nu_{j,\theta}}$		The exploitability of the j^{th} vulnerability in a single VM, $\nu_{j,\theta} \in V_{i,\theta}$.	Table 4.1
ap		An attack path in the cloud defined as a series of adjacent VMs in the cloud from the entry point to the target.	$ap = (vm_1, vm_2, \ldots, vm_n)$
AP		All possible attack paths in the cloud from an entry point to a target, $\|AP\| = p$	$AP = \{ap_1, ap_2, \ldots, ap_p\}$
$C_b(vm_i)$		A function returning the betweenness value of a VM in the cloud.	Equation 4.1
		Security Metrics	
Cloud Risk ($Risk_c$)	$Risk_{vm_i}$	The risk value associated with a VM in the cloud	Equation 4.3
	$Risk_{ap}$	The count of all VMs' risk values of an attack path ap in the cloud	Example 3
	$Risk_c$	The total risk of the cloud based on an entry point and the target	Equation 4.4
Attack Cost (AC_c)	AC_{vm_i}	The cost of exploiting a VM in the cloud for an attacker by considering attacker's knowledge and excising vulnerabilities in the VM.	Table 4.1
	AC_{ap}	The count of all VMs' AC values of an attack path ap in the cloud	Example 4.5
	AC_c	The total attack cost imposed on an attacker to successfully compromise the target in the cloud	Equation 4.5
Return on Attack (RoA_c)	RoA_{vm_i}	The total gain an attacker may achieve by compromising a VM in the cloud against the attacker's efforts	Equation 4.6
	RoA_{ap}	The count of all VMs' RoA values of an attack path ap in the cloud	Equation 4.7
	RoA_c	The total benefits an attacker gain by successfully compromising the target in the cloud by considering the attacker's effort	Equation 4.7

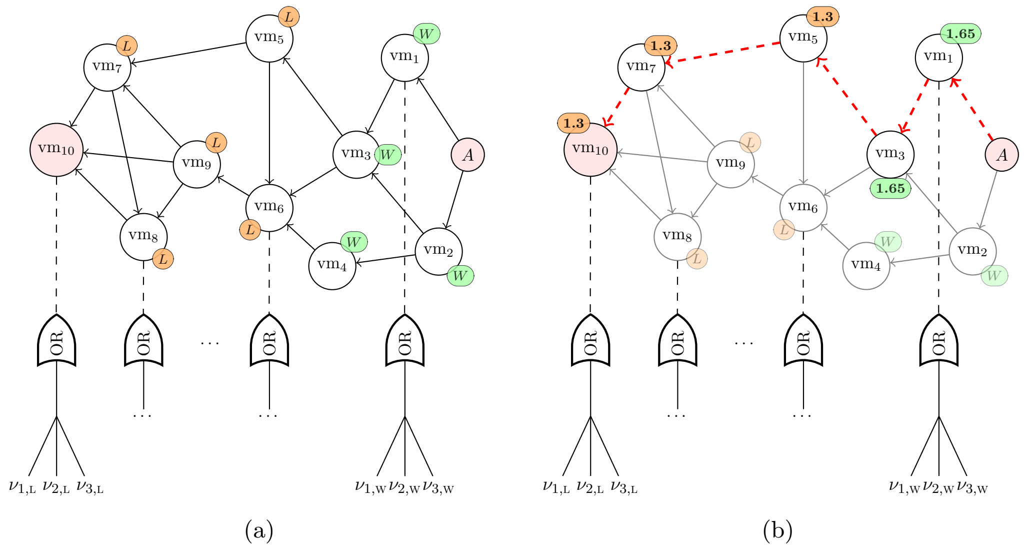


Figure 4.3: (a) Two-layer HARM of the Cloud example. (b) An attack path from the attacker to target in HARM (the risk values of exploiting each VM are depicted beside each VM).

$(A, vm_2), (vm_1, vm_3), (vm_2, vm_3), (vm_2, vm_4), (vm_3, vm_5), (vm_3, vm_6), (vm_4, vm_6), (vm_5, vm_6), (vm_5, vm_7), (vm_6, vm_9), (vm_7, vm_8), (vm_7, vm_{10}), (vm_8, vm_{10}), (vm_9, vm_7), (vm_9, vm_8), (vm_9, vm_{10})\}$. Then, $L_{cs} = \{\ell_1, \ell_2, \ldots, \ell_{10}\}$. For example, $\ell_1 = \{V_{1,W}, G_1, root_1\}$ shows the AT for vm_1, where $V_{1,W}$ is a set of OS vulnerabilities (Windows) existing on vm_1, and $G_1 = OR - gate$, and $root_1 = vm_1$.

4.3.3.1 Importance Measures (IM)

As stated earlier, security analysis through GSM suffers from scalability problems, especially, when we use Exhaustive Search (ES) to find the optimal solution. To address this shortfall, we utilize Betweenness Centrality Measures (C_b) [27] as the IM to discover more critical VMs in the cloud. Using IMs we can find the most important nodes in the network (VMs in here) in the upper layer of HARM. Then we deploy our MTD Technique on a set of crucial VMs without using ES. In Chapter 3, we showed that the best MTD scenario could be found by deploying MTD on the VMs having higher values of Betweenness in the cloud. We recall C_b in Equation (4.1).

$$C_b(vm_i) = \sum_{s,t \in VM \setminus \{vm_i\}} \frac{\delta_{st}(vm_i)}{\delta_{st}}, \tag{4.1}$$

where δ_{st} is a function calculating the total number of shortest path between each pair of VMs $(s,t) \in VM$, and $\delta_{st}(vm_i)$ denotes the number of those paths passing through the specific VM (vm_i). Then, after constructing the HARM, we can find the most important

VM in the cloud based on the HARM definition denoted as $\kappa = \Gamma(H)$, where $\Gamma(H)$ is a function returning the argument of a VM in the HARM (H) having the highest Betweenness values based on Equation (4.2).

$$\Gamma(H) = \arg \max_{vm_i \in VM} C_b(vm_i) \tag{4.2}$$

Then, the value of κ determines the argument of the VM that needs to be selected for deploying MTD technique on. For instance, $\kappa = 3$ shows that MTD technique should be applied on vm_3.

4.3.3.2 Cloud Risk

Constructing the HARM, we can compute the security metrics. The overall risk value associated with the cloud is called Cloud Risk ($Risk_c$). We can construct the HARM through obtaining information related to the VM connectivities on the cloud and considering the vulnerabilities of each VM through a vulnerability database (i.e. NVD) as listed in Table 4.1. In order to measure the $Risk_c$, we use both layers of HARM in which VMs connectivities are presented on the upper layer and vulnerabilities are captured on the lower layer. Let $E_{\nu_{j,\theta}}$ and $I_{\nu_{j,\theta}}$ be the exploitability and impact values of j^{th} vulnerability existing on the OS θ on the vm_i such that $\nu_{j,\theta} \in V_{i,\theta}$. Then, the risk of a VM can be computed as the product of those values returning the maximum value, as shown in Equation (4.3).

$$Risk_{vm_i} = \max_{\nu_{j,\theta} \in V_{i,\theta}} \big(E_{\nu_{j,\theta}} \times I_{\nu_{j,\theta}}\big). \tag{4.3}$$

Thus, the value of $Risk_{ap}$ can be computed as counting all risk values on any single VM in an attack path (ap) from the attacker to the target. Then, the sum of all risk values associated with all possible attack paths on the system provides the $Risk_c$ value, see Equation (4.4).

$$Risk_c = \sum_{ap \in AP} \Bigg(\sum_{vm_i \in ap} Risk_{vm_i} \Bigg) \tag{4.4}$$

Example 3. Figure 4.3b shows an attack path from the attacker to target and can be shown as $ap_1 = \{(A, vm_1), (vm_1, vm_3), (vm_3, vm_5), (vm_5, vm_7), (vm_7, vm_{10})\}$, then $Risk_{ap_1} = Risk_{vm_1} + Risk_{vm_3} + Risk_{vm_5} + Risk_{vm_7} + Risk_{vm_{10}} = 5.9 * 0.28 + 5.9 * 0.28 + 5.9 * 0.22 + 5.9 * 0.22 + 5.9 * 0.22 = \mathbf{7.2}$. Finally, the total risk of the system is as sum of all R$_{ap}$ values which is $Risk_c = \mathbf{211.692}$ for the cloud example.

4.3.3.3 Attack Cost

The cost of exploiting the vulnerabilities on a VM by an attacker is defined as Attack Cost (AC). We expand this metric to compute the overall AC of the cloud. We use the upper layer of HARM to compute AC. Table 4.1 lists the cost of exploiting a VM through vulnerabilities (AC_{vm}). Equation (4.5) shows the calculation formula for the overall AC value of the cloud.

$$AC_c = \sum_{ap \in AP} \left(\sum_{vm_i \in ap} AC_{vm_i} \right) \tag{4.5}$$

where ap is a single attack path in the system and AP_s is the list of all possible attack paths in the network.

4.3.3.4 Return on Attack

Return on Attack (RoA) is another security metrics from the attacker's perspective [42]. RoA quantifies the attack cost versus benefits of attack. The higher value of RoA indicates a higher probability that attacker exploit those vulnerabilities (higher tendency to attack). The ratio of risk value for a VM and attack cost determines the RoA value for a specific VM which are shown in Equation (4.6). Then, the overall RoA of a system can be computed through Equation (4.7).

$$RoA_{vm_i} = \frac{\max_{\nu_{j,\theta} \in V_{i,\theta}} \left(E_{\nu_{j,\theta}} \times I_{\nu_{j,\theta}} \right)}{AC_{vm_i}} \tag{4.6}$$

$$RoA_c = \sum_{ap \in AP} \left(\sum_{vm_i \in ap} RoA_{vm_i} \right) \tag{4.7}$$

4.3.3.5 Path-based Metrics

We consider path-based metrics as presented in [158] to quantify the network security level resulting from deploying MTD techniques on each VM. We use four path-based metrics in this chapter, Shortest Attack Path (SAP), Mean of Attack Path Lengths (MAPL), Standard Deviation of Path Lengths (SDPL), Mode of Path Lengths (MoPL), but other metrics such as Attack Resistance Metric, Network Compromise Percentage [158] can also be computed. Path-based metrics only considers the reachability of the network and the changes on the vulnerabilities in the VMs may not affect the path-based metrics. Thus, Path-based metrics may be changed if only the upper layer of HARM is changed, and the changes in the lower layer of HARM resulting from adding or removing the vulnerabilities have no effects on the path-based metrics. The SAP metric for the cloud example, as shown in Figure 4.3a can be computed as Equation (4.8).

$$SAP_c = \min_{ap \in AP} |ap| = 5 \tag{4.8}$$

The SAP_c value indicates the minimum number of VMs the attacker needs to exploit before compromising the target VM. The cloud provider may select the appropriate countermeasure to increase SAP_c to make the attack more difficult for the attacker. For

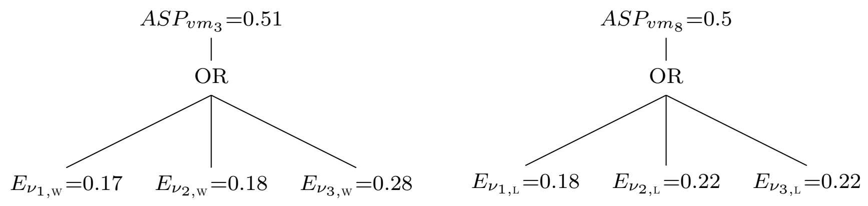


Figure 4.4: Generating Lower Layers AT to calculate ASP for vm_3 and vm_8

example, the cloud provider may choose a replacement strategy for Shuffle so that a VM can be injected to the shortest path to make the attack path more difficult to exploit.

The existence of more number of attack paths (p) in the network causes less security as the attacker can leverage different alternative paths to launch the attacks. Thus, the increment of p and higher values for MAPL indicates less security in the network. Equation (4.9) shows the calculation for MAPL for the cloud example. Moreover, the SDPL metric for the cloud example can be computed as Equation (4.10).

$$MAPL_c = \frac{\sum_{ap \in AP} |ap|}{p} = 6.25 \tag{4.9}$$

$$SDPL_c = \sqrt{\frac{\sum_{ap \in AP} (|ap| - \text{MAPL}_c)^2}{p}} = 0.89 \tag{4.10}$$

4.3.3.6 Attack Success Probability

We can generate the ATs in the lower layer of HARM based on the given vulnerabilities and compute Attack Success Probability (ASP) for each VM vm_i in the cloud denoted as ASP_{vm_i}. However, generating ATs from vulnerabilities needs a clear understanding of the vulnerabilities and the way in which they can be exploited. For instance, an attacker can exploit only a single vulnerability to penetrate into a VM, or the attacker may need to exploit a set of vulnerabilities to penetrate into a VM. In the former, a logical OR-gate can be used and for the latter, a combination of logical AND/OR-gates can be used as presented in [128]. For simplicity, we assume that the attacker can compromise a VM by exploiting any vulnerability existing in a VM. In this case, only OR-gate is used to generate AT. The ASP_{vm_i} can be obtained based on the exploitability values (E) of the vulnerabilities on the VM. Let $E_{v_{j,\theta}}$ be the exploitability value of j^{th} vulnerability existing on the OS θ on the vm_i such that $v_{j,\theta} \in V_{i,\theta}$. Then, we can compute the ASP_{vm_i} as Equation (4.11).

$$ASP_{vm_i} = 1 - \prod_{v_{j,\theta} \in V_{i,\theta}} \left(1 - E_{v_{j,\theta}}\right) \tag{4.11}$$

In this study, we only consider the OS vulnerabilities. For example, we assume that there are three vulnerabilities for Win10 as shown in Table 4.1 and vm_3 uses Win10.

Thus, based on the Example 2, the lower layer of HARM for vm_3 can be shown as $\ell_3 = \{V_{3,\mathrm{W}}, G_3, root_3\}$ shows the AT for vm_3, where $V_{3,\mathrm{W}}$ is a set of Win10 vulnerabilities existing on vm_3, and $G_3 = OR - gate$, and $root_3 = vm_3$. Then, the ASP_{vm_3} can be computed as $1 - (1 - 0.17) \times (1 - 0.18) \times (1 - 0.28) = 0.51$. The generated ATs for both vm_3 and vm_8 with the corresponding ASP values are shown in Figure 4.4. Finally, in order to compute the total ASP for the cloud ASP_c, we used SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator) [130], which uses a reliability graph to quantify the overall ASP. The upper layer of HARM including the ASP for each VM can be fed into the SHARPE as a reliability graph.

4.4 MTD Techniques Deployment

Deploying MTD techniques on the cloud depends on the constraints defined by the cloud providers. For instance, some operations may be restricted by the cloud providers such as VM-LM from a host to a specific host (which is protected). We assume that VM-LM and OS Diversification are allowed by the cloud provider. Then, we utilize VM-LM and OS Diversification techniques to develop Shuffle and Diversity techniques respectively. Later on in Section 4.5, we discuss the MTD combination deployment strategies. In order to evaluate the effects of MTD techniques on the cloud, we analyze the cloud security level before deploying MTD techniques by computing the security metrics: $Risk_c$, AC_c, and RoA_c as described in Table 4.2, and then analyze the effectiveness of each selected MTD technique by reconsidering those security metrics again, the notations and symbols are presented in Table 4.3. In this section, we define and formalize the MTD techniques we used in this chapter which are Shuffle, Diversity, and the combinations of both. Then, we consider the effects of deploying each MTD techniques on the security metrics.

4.4.1 Shuffle Technique Definition and Formalization

In general, Shuffle techniques work based on the rearrangement of the system's components, settings, and configuration such as mutating or shuffling the IP address, rearranging the network's topology, moving a VM from one host to another one, and so forth [45, 75, 137]. Shuffle strategies can be deployed in different layers like Application Layer, Virtualization, Host, Virtual Machine Monitoring (VMM), Hardware [5, 151, 164]. In this study, we deploy MTD techniques on the virtualization layer. We utilize VM-LM method to deploy Shuffle in our simulation, as depicted in Figure 4.2. Migration of each VM from a host to another machine may affect the overall security of the system. For this reason, before deploying Shuffle technique, we investigate the effects of deploying Shuffle technique on the cloud by analyzing the security metrics which are $Risk_c$, AC_c, and RoA_c through HARM.

Deploying Shuffle techniques falls into two scenarios: (1) Selecting the VM for deploying Shuffle which is usually named as 'what to move' in the literature [28]. (2) The place

Table 4.3: Notations w.r.t deploying MTD techniques and metrics

Symbol	**Descriptions**
$S{\otimes}D$	Combination of Shuffle (S) and Diversity (D) strategies
$S{+}D$	Strategy: deploying S first, then D
$S\Delta D$	Strategy: deploying S first, computing IM, then D
$Risk_c^S$	Total cloud's risk after deploying S only
$Risk_c^D$	Total cloud's risk after deploying D only
$Risk_c^{S+D}$	Total cloud's risk after deploying $S{+}D$
$Risk_c^{S\Delta D}$	Total cloud's risk after deploying $S\Delta D$
AC_c^S	Total attack cost value after deploying S only
AC_c^D	Total attack cost value after deploying D only
AC_c^{S+D}	Total attack cost value after deploying $S{+}D$
$AC_c^{S\Delta D}$	Total attack cost value after deploying $S\Delta D$
RoA_c^S	Total return on attack value after deploying S only
RoA_c^D	Total return on attack value after deploying D only
RoA_c^{S+D}	Total return on attack value after deploying $S{+}D$
$RoA_c^{S\Delta D}$	Total return on attack value after deploying $S\Delta D$
ASP_c^S	Attack Success Probability after deploying S only
ASP_c^D	Attack Success Probability after deploying D only
ASP_c^{S+D}	Attack Success Probability after deploying $S{+}D$
$ASP_c^{S\Delta D}$	Attack Success Probability after deploying $S\Delta D$

in which the VM should be moved into which is called as 'how to move' [28]. In this chapter, we utilized the both Exhaustive Search (ES) and Important Measures (IMs) for the former scenario and a replacement algorithm for the latter scenario which chooses the shortest path in the upper layer of the HARM and migrates the selected VM to that place. This replacement algorithm aims to increase the length of the shortest attack path that attacker can traverse from the entry point (the Internet) to the target (i.e. a DB).

As the Shuffle technique only changes the physical locations of VMs (i.e. from a physical server in the cloud to another server), it only affects the reachability of VMs in the upper layer of HARM.

We recall Shuffle technique formulated in Definition 5 presented in Chapter 3.

Definition 5. Let $S(H,\kappa)$ be a Shuffle function on the HARM where $1 \leq \kappa \leq n$, and κ denotes a specific VM $vm_\kappa \in VM$ chosen for the shuffling. Then the result of Shuffle function is as $S(H,\kappa) = H^{\mathrm{s}}$. We define $H^{\mathrm{s}} = (U^{\mathrm{s},\kappa}, L, C)$ where $U^{\mathrm{s},\kappa}$ is the transformed AG resulted from Shuffle on vm_k in the upper layer of the HARM and can be represented as $U^{\mathrm{s},\kappa} = (VM, E')$, where $E' \subseteq VM \times VM$.

4.4.2 Shuffle Technique Evaluation

According to the Shuffle definition 4.4.1, the result of deploying Shuffle on the cloud example (Figure4.3a) is as follows.

Table 4.4: The percentage of changes in the security metrics after deploying Shuffle on each VM in the cloud example

ID	Values			% of Changes		
	$Risk_c^S$	AC_c^S	RoA_c^S	$Risk_c^S$	AC_c^S	RoA_c^S
vm$_1$	164.49	201.8	102.81	-22.30%	-19.73%	-24.28%
vm$_2$	130.51	161.8	80.86	-38.35%	-35.64%	-40.44%
vm$_3$	**82.25**	100.9	**51.40**	**-61.15%**	-59.86%	**-62.14%**
vm$_4$	174.05	209.7	110.37	-17.78%	-16.59%	-18.70%
vm$_5$	116.94	139.5	74.73	-44.76%	-44.51%	-44.95%
vm$_6$	88.26	102.4	57.61	-58.31%	-59.27%	-57.56%
vm$_7$	122.48	146.4	78.16	-42.14%	-41.77%	-42.43%
vm$_8$	205.67	236.5	135.13	-2.84%	-5.93%	-0.46%
vm$_9$	93.1	110.7	59.65	-56.02%	-55.97%	-56.06%
Best	vm$_3$	vm$_8$	vm$_3$	vm$_3$	vm$_8$	vm$_3$

Example 4. The result of Shuffle function on the generated HARM for the cloud example (H_{cs}) in which the most crucial VM is selected to be shuffled (vm_κ) can be represented as $S(H_{cs}, \kappa) = H_{cs}^s$. Based on Equation (4.2), $\kappa = 3$ which means vm_3 is selected as a VM with the highest IM values on the cloud. Then, $H_{cs}^s = (U_{cs}^{s,3}, L, C)$ which $U_{cs}^{s,3} = (VM_{cs}, E'_{cs})$ is the upper layer of HARM for cloud system, where $VM_{cs} = \{A, vm_1, vm_2, vm_3, vm_4, vm_5, vm_6, vm_7, vm_8, vm_9, vm_{10}\}$, and E'_{cs} denotes the connectivity of VMs after deploying Shuffle technique on vm_3 and is represented as $E'_{cs} = \{(A, vm_1),(A, vm_2), (vm_2, vm_3), (vm_3, vm_4), (vm_4, vm_6), (vm_6, vm_9), (vm_5, vm_6), (vm_5, vm_7), (vm_7, vm_{10}), (vm_9, vm_8), \ldots\}$. Note that Shuffle technique only changes the upper layer of HARM and preserves L and C.

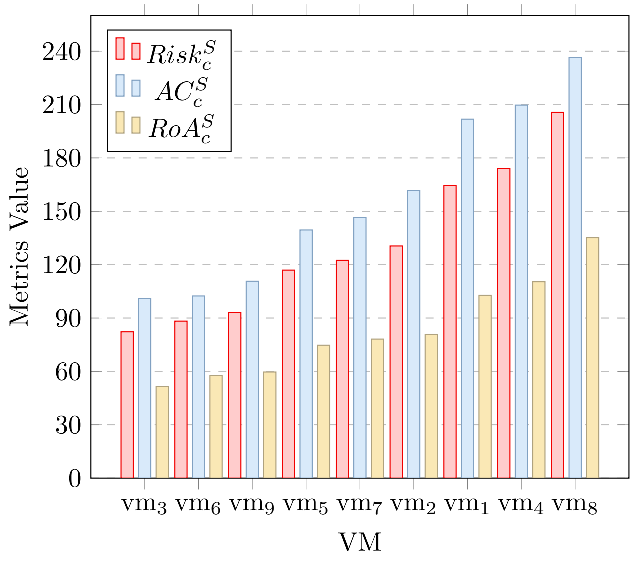
Based on the definition of the Shuffle technique 4.4.1, we deploy Shuffle on a VM and then recalculate the security metrics to observe the effectiveness of the deployed technique in the cloud. For Example, deploying $S(H_{cs}, vm_3)$ may add/remove edges in the upper layer of HARM. Let consider a new attack path resulting from deploying Shuffle $ap_1 = \{(A, vm_2), (vm_2, vm_3), (vm_3, vm_4), (vm_4, vm_6), (vm_6, vm_9), (vm_9, vm_8), (vm_8, vm_{10})\}$, then we can compute the $Risk_{ap_1} = Risk_{vm_2} + Risk_{vm_3} + Risk_{vm_4} + Risk_{vm_6} + Risk_{vm_9} + Risk_{vm_8} + Risk_{vm_{10}} = 1.65+1.65+1.65+1.3+1.3+1.3+1.3 = \mathbf{10.15}$. Likewise, we calculate the AC_{ap_1}, RoA_{ap_1} as follows. $AC_{ap_1} = AC_{vm_2} + AC_{vm_3} + AC_{vm_4} + AC_{vm_6} + AC_{vm_9} + AC_{vm_8} + AC_{vm_{10}} = 1.2 + 1.2 + 1.2 + 1.9 + 1.9 + 1.9 + 1.9 = \mathbf{11.2}$. $RoA_{ap_1} = RoA_{vm_2} + RoA_{vm_3} + RoA_{vm_4} + RoA_{vm_6} + RoA_{vm_9} + RoA_{vm_8} + RoA_{vm_{10}} = 1.38 + 1.38 + 1.38 + 0.68 + 0.68 + 0.68 + 0.68 = \mathbf{6.86}$.

We denote the overall security metrics after deploying Shuffle techniques on the cloud as $Risk_c^S$, AC_c^S, and RoA_c^S. Then, after deploying Shuffle on vm_3, $Risk_c^S = \mathbf{82.25}$, $AC_c^S = \mathbf{100.9}$, and $RoA_c^S = \mathbf{51.4}$. We then calculate the results of deploying Shuffle on other VMs and compare them. Figure 4.5a reveals the results of deploying Shuffle technique on each VM in the cloud example. We illustrate the VMs sorted in descending order based on their IMs values in the x-axis and the security metrics values for $Risk_c^S$, AC_c^S, RoA_c^S in the y-axis. It is obvious that the values obtained for VMs vm_3, vm_6, and vm_9 (which have higher ranks in terms of IMs) are lower than the others, while deploying Shuffle on the VMs vm_4 and vm_8 (which have lower IMs ranks) yields higher $Risk_c$ and RoA_c. Accordingly, we tabulated the percentage of the changes in the security metrics in Table 4.4. The results show that the $Risk_c^S$, and RoA_c^S are improved after deploying Shuffle techniques on the cloud and the best percentage of changes belongs to deploying Shuffle on vm_3 with **-61.15%** and **-62.14%** for $Risk_c^S$ and RoA_c^S, respectively. The lower values of RoA_c shows that the attacker has less orientation to launch the attacks again on that specific cloud's configuration, while the higher rate of the RoA_c shows the higher probability of the same attack is launched by the attacker.

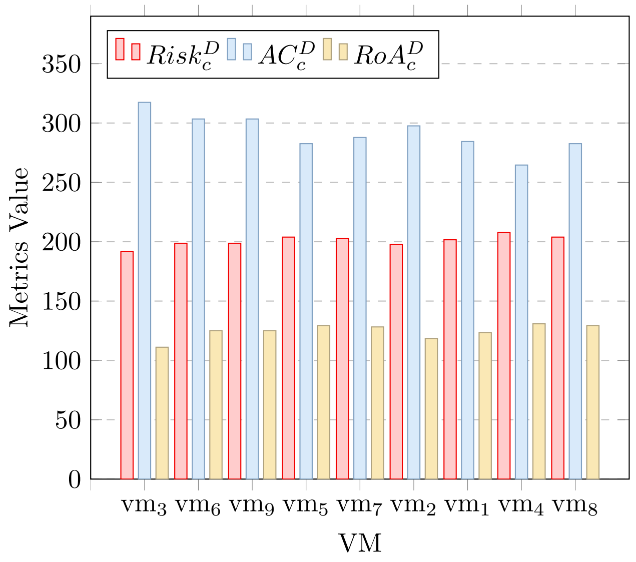
4.4.3 Diversity Technique Definition and Formalization

The main idea behind the Diversity techniques is to replace the systems' components (e.g. a VM, server, programming language, OS, hardware, and *etc.*) with different variants or implementations, while the system provides the same services for the customers [110, 126]. We change the OS instances (OS Diversification) in the cloud to deploy Diversity technique. Unlike Shuffle technique which may change the reachability of the VMs in the cloud, deploying Diversity has no effects on the reachability of the VMs and may vary the vulnerabilities existing on the VMs (only the lower layer of HARM may be affected and the upper layer remains the same). In this study, we only consider OS vulnerabilities though other vulnerabilities can also be investigated through HARM [65]. Deploying Diversity has two advantages. First, it invalidates the information collected by the attacker to exploit the vulnerabilities of the current system. Secondly, Diversity can introduce new vulnerabilities to the attacker, thus the adversary needs to deal with new vulnerabilities and spend more time and cost to gain information and exploit them. In here, we assume that the cloud provider can use a backup OS which imposes a higher cost to the attacker based on the vulnerabilities on the new OS, we consider AC as the difficulties to exploit vulnerabilities using NVD database. Diversity can be formulated based on the HARM definition as follows.

Definition 10. We formulate the Diversity technique in which the diversity function is applied on H as $D(H, \kappa) = H^d$, where κ denotes a specific VM $vm_\kappa \in VM$ selected for replacing with another OS variant. Then, $H^d = (U, L^{d,k}, C)$ is the result of deploying Diversity technique, where $L^{d,k} = \{\ell_1, \ldots, \ell_{\mathbf{k}}, \ldots, \ell_n\}$ denotes the ATs corresponding to each VM and $\ell_k = (V_{k,\theta}, G, root)$ is the transformed AT of vm_k which is replaced with another variant $\theta \in OS$. Diversity technique affects the lower layer and varies vulnerabilities $V_{k,\theta} = \{\nu_{1,\theta}, \nu_{2,\theta}, \ldots, \nu_{m,\theta}\}$, while $U = (VM, E)$ is preserved.



(a) S



(b) D

Figure 4.5: Comparison of security metrics after deploying two Shuffle MTD and Diversity MTD strategies separately on the VMs (VMs in x-axis are sorted in descending order based on their importance using IMs). (a) The results of deploying Shuffle technique on the cloud example. (b) The results of deploying Diversity technique on the cloud example.

4.4.4 Diversity Technique Evaluation

In order to analyze Diversity technique, we first create the HARM for the cloud example 2 and compute the security metrics to observe the current security posture of the cloud. Then, we deploy the Diversity technique on the VMs based on the Diversity def-

Table 4.5: The percentage of changes in the security metrics after deploying Diversity on each VM in the cloud example

ID	Values			% of Changes		
	$Risk_c^D$	AC_c^D	RoA_c^D	$Risk_c^D$	AC_c^D	RoA_c^D
vm$_1$	201.65	284.4	123.44	-4.74%	13.13%	-9.08%
vm$_2$	197.64	297.6	118.51	-6.64%	18.38%	-12.71%
vm$_3$	**191.61**	**317.4**	**111.11**	**-9.5%**	**26.25%**	**-18.16%**
vm$_4$	207.68	264.6	130.83	-1.9%	5.25%	-3.63%
vm$_5$	203.89	282.6	129.29	-3.68%	12.41%	-4.77%
vm$_6$	198.69	303.4	124.98	-6.14%	20.68%	-7.94%
vm$_7$	202.59	287.8	128.21	-4.3%	14.48%	-5.56%
vm$_8$	203.89	282.6	129.29	-3.68%	12.41%	-4.77%
vm$_9$	198.69	303.4	124.98	-6.14%	20.68%	-7.94%
Best	vm$_3$	vm$_3$	vm$_3$	vm$_3$	vm$_3$	vm$_3$

inition 10. Finally, we compare and analyze the security metrics after deploying each Diversity scenario. The following example shows the Diversity based on the given Diversity definition 10. We change the current OS for each VM with the backup one. To be specific, we use Fedora as the backup OS which includes different vulnerability values, see Table 4.1.

Example 5. The result of Diversity function on the HARM for the cloud example (H_{cs}) in which a VM is selected to be shuffled (vm_κ) is represented as $S(H_{cs}, \kappa) = H_{cs}^d$. For example, we select vm_3 having the highest IM value on the cloud for deploying Diversity. Then, $H_{cs}^d = (U_{cs}, L_{cs}^{d,3}, C)$ which $U_{cs} = (VM_{cs}, E_{cs})$ remains unchanged, and $\ell_3 = (V_{3,W}, G, root)$, where $\ell_3 \in L_{cs}^{d,3}$, and $V_{3,W} = \{\nu_{1,W}, \nu_{2,W}, \nu_{3,W}\}$, has been replaced with the backup OS (Fedora(F)). Then, $\ell_3 = (V_{3,F}, G, root)$ where, $V_{3,F} = \{\nu_{1,F}\}$ as in Table 4.1

Unlike Shuffle, deploying Diversity will not add/remove or change any attack path in the upper layer of the HARM, but it varies the lower layer of the HARM by introducing different vulnerabilities. Consequently, the security metrics regarding each attack path may change. For example, let consider an attack path existing on the cloud example (showed by red dashed lines in Figure 4.3b) which is defined as $ap_1 = \{(A, vm_1)\ , (vm_1, vm_3), (vm_3, vm_5), (vm_5, vm_7), (vm_7, vm_{10})\}$. Then, we deploy Diversity techniques on vm_3, and calculate the security metrics for ap_1 as follows. $Risk_{ap_1} = Risk_{vm_1} + Risk_{vm_3} + Risk_{vm_5} + Risk_{vm_7} + Risk_{vm_{10}} \approx \mathbf{6.2}$. Likewise, AC_{ap_1} and RoA_{ap_1} are around **11.4** and **3.56** respectively. Then, the overall values of the security metrics after deploying Diversity on vm_3 are as follows: $Risk_c^D = \mathbf{191.61}$, $AC_c^D = \mathbf{317.4}$, $RoA_c^D = \mathbf{111.11}$. To compare with the other deployment scenarios, we compute the Diversity technique on the other VMs in the cloud example and compare the results among them. Figure 4.5b shows the values of Diversity on each VM in the cloud example. The

VMs are sorted in descending order based on their IMs values on the x-axis and the security metrics on the y-axis. The results show that deploying Diversity highly increases the AC value. However, the highest AC values can be found through deploying Diversity on the VMs vm_3, vm_6, and vm_9 which have higher IMs values. Similarly, better results can be obtained for $Risk$ and RoA values for vm_3. Table 4.5 lists the percentage of changes in the security metrics after deploying Diversity showing the negative changes for $Risk$ and RoA values and the positive changes on AC values. Although changing OS is used for deploying Diversity technique, the variation on the security metrics such as AC highly depend on the vulnerability information introduced by the new OS.

4.5 MTD Combinations Definition and Formalization

In this section, we investigate the effects of deploying the combination of Shuffle and Diversity together using the security metrics. Based on the results reported in sections 4.4.2 and 4.4.4, deploying Shuffle technique decreases both $Risk_c^S$ and RoA_c^C, but it also reduces the AC_c^S values. However, deploying Diversity increases AC_c^S values, but the percentage of changes in $Risk_c^S$ and RoA_c^S are not very significant. Thus, the idea to combine both Shuffle and Diversity may be useful to improve all those metrics. We formulate and combine Shuffle and Diversity technique to understand the benefits of deploying these techniques together on the cloud. To combine both Shuffle and Diversity techniques, we utilize VM-LM as the Shuffle technique and OS diversification as the Diversity techniques; then we deploy both VM-LM and OS diversification at the same time for each VM in the cloud. The strategies for ways combining MTD techniques can be different. For example, we can deploy Shuffle and Diversity on a single VM, or different VMs. In this section, we investigate the combination strategies and the effectiveness of each. Finding the best deployment strategy through Exhaustive Search (ES) is time-consuming on the large clouds. To address this problem, we use IMs to find out the most important VMs on the cloud first before applying MTD techniques on them.

We combine MTD techniques through considering the following strategies and steps after the first computation of IMs:

$S{+}D$**:** (i) deploying Shuffle on the VM indicated by IMs, (ii) deploying Diversity on the same VM;

$D{+}S$**:** (i) deploying Diversity on the VM indicated by IMs, (ii) deploying Shuffle on the same VM;

$S\Delta D$**:** (i) deploying Shuffle on the VM indicated by IMs, (ii) computing IMs again, (iii) deploying Diversity on a VM indicated by IMs;

$D\Delta S$**:** (i) deploying Diversity on the VM indicated by IMs, (ii) computing IMs again, (iii) deploying Shuffle on a VM indicated by IMs;

We define $S{\otimes}D$ as the combination of MTD techniques including different combination strategies. Although there are four possible combinations for combining Shuffle and Diversity using IMs, some combinations are equivalent based on the definition of HARM. For example, as Diversity only affects the lower layer of HARM, thus deploying Diversity first does not vary the upper layer of HARM. In this case, the permutation of $S{+}D$ or $D{+}S$ are the same and both techniques yield the same results. Similarly, the computation of IMs cannot be affected by deploying Diversity first. Then, the permutations of $S\Delta D$ or $D\Delta S$ are equivalent and both lead to the same results. The formal definitions of $S{\otimes}D$ include $S{+}D$ and $S\Delta D$ based on HARM are given as follows.

Definition 11. Let $S{\otimes}D(H, \kappa_s, \kappa_d)$ be a combination of Shuffle and Diversity function on the HARM where $1 \leq \kappa_s, \kappa_d \leq n$, and κ_s,κ_d denote the specific VM vm_{κ_s}, vm_{κ_d} $\in VM$ chosen for the shuffling and OS diversification respectively. Then the result of $S{\otimes}D$ function is as $S{\otimes}D(H, \kappa_s, \kappa_d) = H^{\mathrm{s}\otimes\mathrm{d}}$. We define $H^{\mathrm{s}\otimes\mathrm{d}} = (U^{\mathrm{s},\kappa_s}, L^{\mathrm{d},\kappa_d}, C)$ where U^{s,κ_s} is equivalent to $U^{\mathrm{s},\kappa}$ in the Shuffle definition 4.4.1, where $\kappa = \kappa_s$, and L^{d,κ_d} is equivalent to $L^{\mathrm{d},\kappa}$ in the Diversity definition 10, where $\kappa = \kappa_d$.

The $S{+}D$ can be a specific strategy under the definition of $S{\otimes}D(H, \kappa_s, \kappa_d)$, where $\kappa_s = \kappa_d$ which means deploying Shuffle and deploying Diversity on the same VM vm_κ and named as $S{+}D(H, \kappa)$. The $S{+}D$ strategy includes two consecutive steps: $S{+}D(H, \kappa){=}S(H, \kappa){+}D(H^s, \kappa)$, which means first deploying Shuffle technique on the $vm_\kappa \in H$ where κ is selected by $\kappa = \Gamma(H)$ and then deploy Diversity on the same VM vm_κ in the transformed HARM $vm_\kappa \in H^s$.

Then, the $S\Delta D$ strategy can be defined as $S{\otimes}D(H, \kappa_s, \kappa_d)$, where κ_s and κ_d are not necessarily the same and is denoted as $S\Delta D(H, \kappa_s, \kappa_d)$. The $S\Delta D$ includes two consecutive steps: $S\Delta D(H, \kappa_s, \kappa_d) = S(H, \kappa_s) + D(H^s, \kappa_d)$, which specifically means first deploying Shuffle technique on the VM selected by $\kappa_s{=}\Gamma(H)$ which is vm_{κ_s}. Then deploy Diversity on the VM chosen by $\kappa_d = \Gamma(H^s)$ which is $vm_{\kappa_d}{\in}H^s$ which exists on the transformed HARM resulting from deploying Shuffle.

4.5.1 Evaluation of MTD Combinations

In this section, we evaluate the effectiveness of combinations of Shuffle and Diversity MTD techniques. This evaluation undergoes three steps: (1) creating the cloud model using HARM, (2) deploying different combination strategies (based on $S{+}D$ and $S{\otimes}D$), and finally (3) analyzing the changes in the cloud's security posture for each strategy.

We analyze the security metrics values after deploying each $S{\otimes}D$ strategies. The following example shows the deploying $S{+}D$ on the cloud based on the given Definition 11.

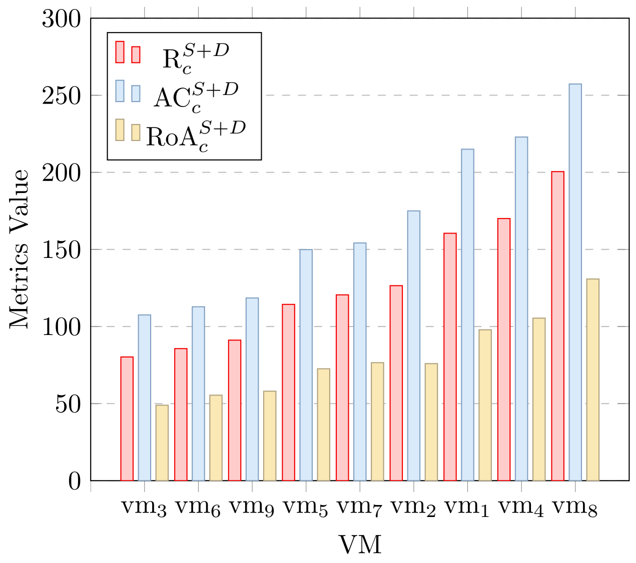
Example 6. The result of $S{+}D$ function on the generated HARM for the cloud example (H_{cs}) in which a VM is selected to be shuffled (vm_κ) can be represented as $S{+}D(H_{cs}, \kappa) = S(H_{cs}, \kappa) + D(H_{cs}^s, \kappa)$ including the following steps: (i) Finding the most important VM in the HARM using $\kappa = \Gamma(H)$, which returns $\kappa = 3$, (ii) Deploying $S(H_{cs}, 3)$, where vm_3 is shuffled and results in the changes on the upper layer of HARM (H^s). (iii) Deploying Diversity on the transformed HARM $D(H^s_{cs}, 3)$, where the selected VM for deploying Diversity is the same as the previous step which was vm_3.

We denote the security metrics resulting from deploying $S{+}D$ as $Risk_c^{S+D}$, AC_c^{S+D}, and RoA_c^{S+D}. Similarly, we can show the example of deploying $S\Delta D$ as follows.

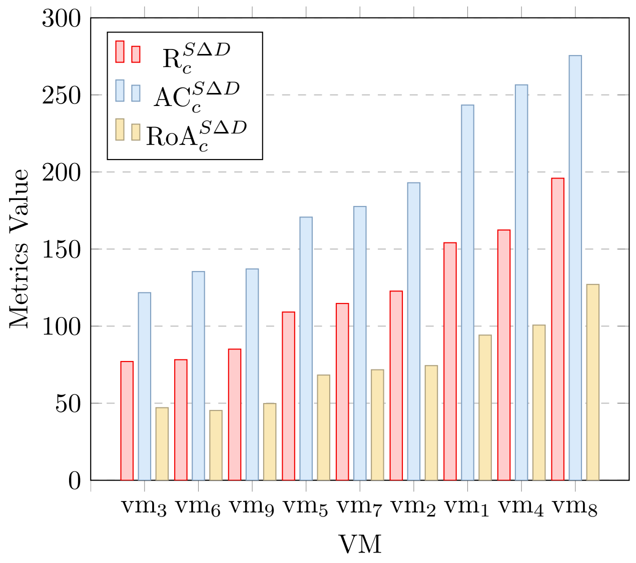
Example 7. The result of $S\Delta D$ function on the generated HARM for the cloud example (H_{cs}) in which two VMs are selected for deploying combined MTD can be represented as $S\Delta D(H_{cs}, \kappa_s, \kappa_d) = S(H_{cs}, \kappa_s) + D(H^s_{cs}, \kappa_d)$ consisting of the following steps: (i) Finding the most important VM in the HARM using $\kappa_s = \Gamma(H)$, which returns $\kappa_s = 3$ meaning vm_3 is selected, (ii) Deploying $S(H_{cs}, 3)$, where vm_3 is shuffled and results in the changes on the HARM (H^s). (iii) Re-calculating IMs again on the transformed HARM (H^s_{cs}) to find the most important VM for deploying Diversity using $\kappa_d = \Gamma(H^s_{cs})$, which returns $\kappa_d = 6$ meaning vm_6 is selected, (iv) Deploying Diversity on the transformed HARM $D(H^s_{cs}, 6)$, where the selected VM for deploying Diversity is vm_6.

We denote the security metrics resulting from deploying $S\Delta D$ as $Risk_c^{S\Delta D}$, $AC_c^{S\Delta D}$, and $RoA_c^{S\Delta D}$. We consider two combination strategies of MTD and evaluate the cloud security level before and after deploying the combined techniques. Then, we compare the combination strategies to find a more effective technique. Figure 4.6 compares the results of deploying two different combinations of MTD techniques which are $S{+}D$ and $S\Delta D$ on the VM in the cloud example through analysis of the security metrics. The metric values are plotted as the y-axis against the VMs sorted in the descending trend based on their IMs values in the x-axis. Figure 4.6a shows the security metrics after deploying $S{+}D$, and Figure 4.6b reveals the results of deploying $S\Delta D$ on the cloud example. Both graphs show the combinations of MTD techniques on VMs which have higher IM values (like vm_3, vm_6, and vm_9) cause lower rates for $Risk_c$ and RoA_c metrics. Although deploying the combined MTD decreases AC_c values, it also intensively reduces $Risk_c$, and RoA_c values. With these results, we can observe that it is important to choose the appropriate VM. For instance, deploying MTD techniques on the VMs with the lower values of IMs are shown to be less effective than the others. Moreover, we can observe that $S\Delta D$ deployment strategy is better than $S{+}D$ as the $Risk_c^{S\Delta D}$ and $RoA_c^{S\Delta D}$ are lower than $Risk_c^{S+D}$ and RoA_c^{S+D}, while $AC_c^{S\Delta D}$ rates are higher than AC_c^{S+D} values. However, the obtained security metrics values depend on the cloud's constraints and might be different in other contexts. Based on the results, the optimal results of deploying $S{+}D$ and $S\Delta D$ are the selection of vm_3 leading to the best values.

Comparing the percentage of the changes on metrics reported in Table 4.6 and 4.7 for combinations of MTD techniques with Table 4.5 for D only, we can notice a significant improvement on $Risk_c^D$ and RoA_c^D, where the percentage of the changes for those values are about **-9.49%** and **-18.16%**, respectively in the best scenario (deploying Diversity



(a) S+D



(b) $S\Delta D$

Figure 4.6: Comparison of security metrics after deploying two MTD combination strategies on the VMs (VMs in x-axis are sorted in descending order based on their importance using IMs)

on vm_3), while these rates for $Risk_c^{S+D}$ and RoA_c^{S+D} are about **-62.1%** and **-63.9%** for the best deployment scenario, and even better for $S\Delta D$ with the values of **-63.6%** and **-65.3%** for $Risk_c^{S\Delta D}$ and $RoA_c^{S\Delta D}$, respectively.

Figure 4.7 compares the results of deploying each MTD technique (S and D) and combination strategies ($S\Delta D$ and S+D) on the security metrics (RoA_c, $Risk_c$, AC_c, and

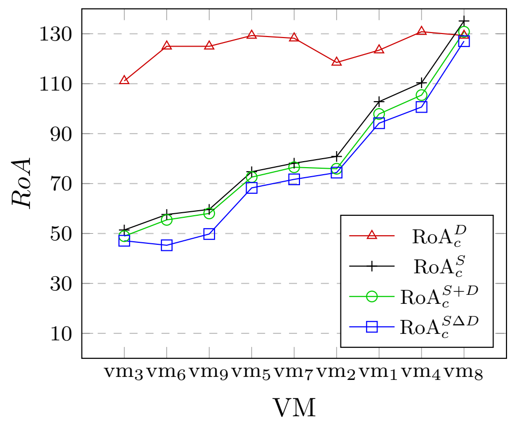
Table 4.6: The percentage of changes in the security metrics after deploying S+D on each VM in the cloud example

VM	Values			% of Changes		
	$Risk_c^{S+D}$	AC_c^{S+D}	RoA_c^{S+D}	$Risk_c^{S+D}$	AC_c^{S+D}	RoA_c^{S+D}
vm_1	160.48	215	97.87	-24.19%	-14.48%	-27.91%
vm_2	126.49	175	75.93	-40.25%	-30.39%	-44.07%
vm_3	**80.24**	107.5	**48.94**	**-62.10%**	-57.24%	**-63.95%**
vm_4	170.03	222.9	105.44	-19.68%	-11.34%	-22.33%
vm_5	114.34	149.9	72.58	-45.99%	-40.37%	-46.54%
vm_6	85.66	112.8	55.46	-59.53%	-55.13%	-59.15%
vm_7	120.53	154.2	76.54	-43.06%	-38.66%	-43.62%
vm_8	200.47	257.3	130.82	-5.30%	2.35%	-3.64%
vm_9	91.15	118.5	58.03	-56.94%	-52.86%	-57.25%
Best	vm_3	vm_8	vm_3	vm_3	vm_8	vm_3

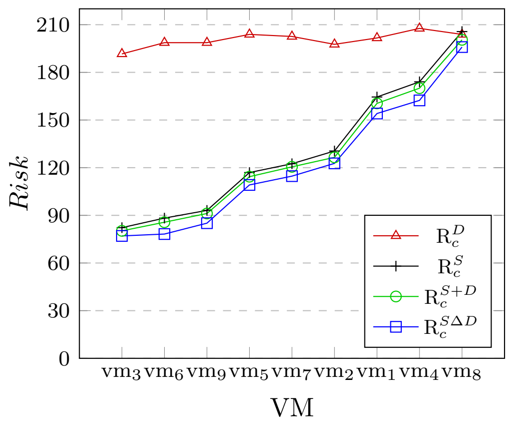
Table 4.7: The percentage of changes in the security metrics after deploying SΔD on each VM in the cloud example

VM	Values			% of Changes		
	$Risk_c^{S\Delta D}$	$AC_c^{S\Delta D}$	$RoA_c^{S\Delta D}$	$Risk_c^{S\Delta D}$	$AC_c^{S\Delta D}$	$RoA_c^{S\Delta D}$
vm_1	154.09	243.4	94.18	-27.21%	-3.18%	-30.63%
vm_2	122.71	193	74.39	-42.03%	-23.23%	-45.20%
vm_3	**77.05**	121.7	**47.09**	**-63.60%**	-51.59%	**-65.31%**
vm_4	162.35	256.5	100.67	-23.31%	2.03%	-25.85%
vm_5	109.14	170.7	68.26	-48.44%	-32.10%	-49.72%
vm_6	78.22	135.4	45.29	-63.05%	-46.14%	**-66.65%**
vm_7	114.68	177.6	71.69	-45.83%	-29.36%	-47.19%
vm_8	195.92	275.5	127.04	-7.45%	9.59%	-6.42%
vm_9	85.07	137.1	49.79	-59.81%	-45.47%	-63.32%
Best	vm_3	vm_8	vm_6	vm_3	vm_8	vm_6

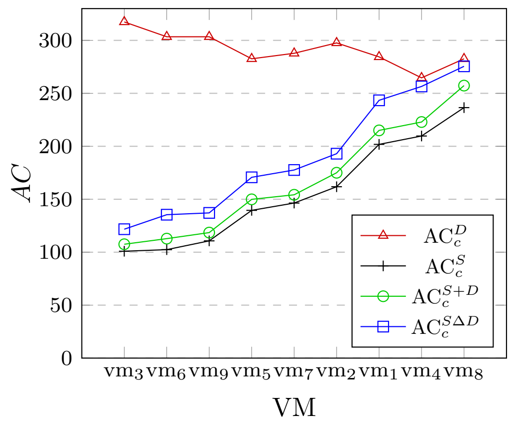
ASP_c). The results demonstrated in Figure 4.7a show that $RoA_c^{S\Delta D}$ has lower values than the other MTD techniques and combination strategies for all selected VMs. However, the best RoA_c results can be found through deploying $S\Delta D$ on the top three VMs vm_3, vm_6, and vm_9 having the higher IMs rates which yield $RoA_c^{S\Delta D}$ lower than 50. Nevertheless, the worst results for RoA_c can be obtained after deploying Diversity which yields the highest RoA_c as RoA_c^D values fall between **110** and **130** in the best and the worst deployment scenarios (shown in Figure 4.7a). The same trend can be seen in the results of deploying MTD strategies on $Risk_c$ values. The best $Risk_c$ results from deploying $S\Delta D$ as $Risk_c^{S\Delta D}$ for all VMs have lower values than the $Risk_c$ obtained from the other MTD strategies. In contrast, $Risk_c^D$ yield the worst results, see Figure 4.7b. The results of AC_c are shown in Figure 4.7c. Comparing the AC_c values obtained from deploying different MTD strategies, we observe that AC_c^D values are highest and AC_c^S values are the lowest among the other



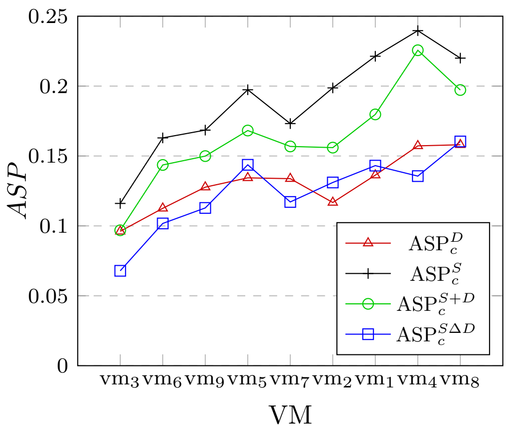
(a) RoA_c values



(b) $Risk_c$ values



(c) AC_c values



(d) ASP_c values

Figure 4.7: Comparison of security metrics resulting from deploying various MTD techniques (VMs in x-axis are sorted in descending order based on their importance using IMs)

techniques. Figure 4.7d compares all ASP_c values resulting from deploying different MTD strategies on the VMs. As it can be seen, the results of $ASP_c^{S\Delta D}$ yield the lower ASP_c values in comparison with the other MTD strategies. It means that after deploying $S\Delta D$ the attacker has a lower chance to compromise the target while this chance after deploying other MTD strategies are higher. The worst ASP_c values can be found after deploying Shuffle as the ASP_c^S are the highest. Furthermore, in all cases, deploying MTD strategies on the VMs having higher IMs values leads to better results in terms of the security metrics ($Risk_c$, RoA_c, AC_c, and ASP_c). Moreover, considering the overall results, we can conclude that combining MTD techniques leads to better results in terms of the security metrics. We also observe that the combination strategies are important to achieve a better security posture as the results of $S\Delta D$ surpasses $S+D$ values.

However, deploying Diversity has no effect on the reachability of the VMs, whilst

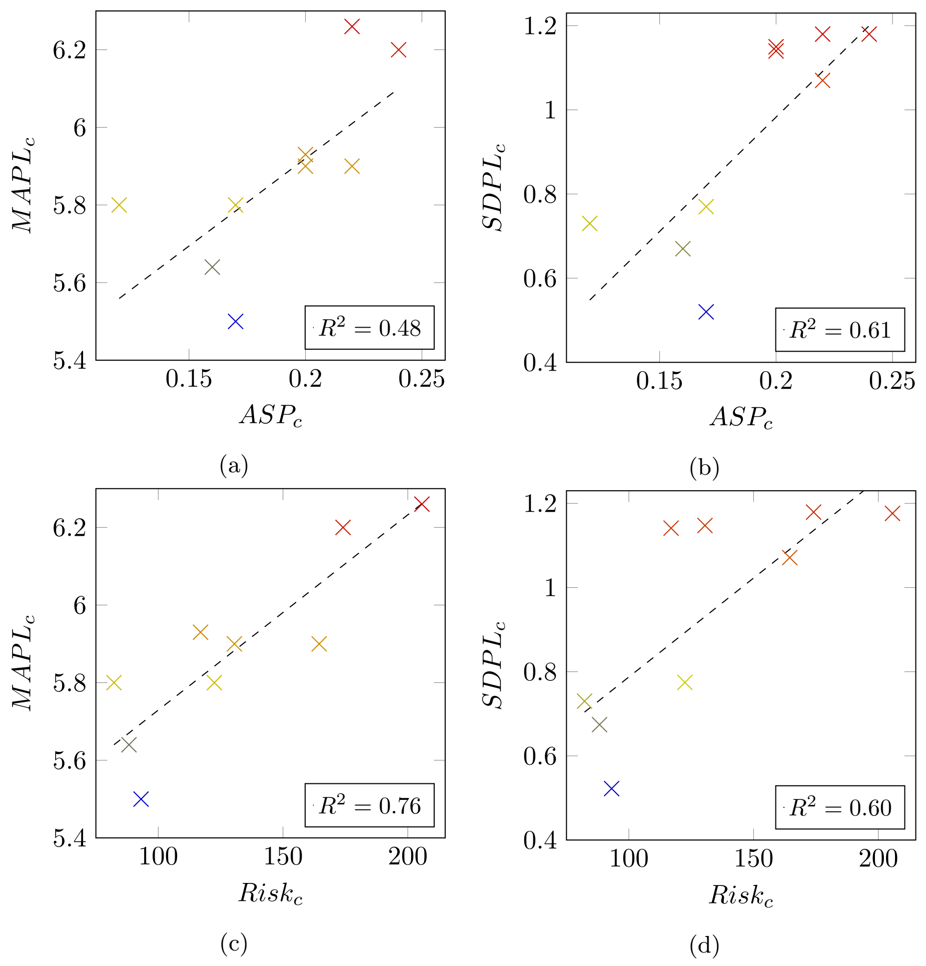


Figure 4.8: Comparing the correlation of path-based metrics against ASP_c and $Risk_c$ after deploying Shuffle on each VM. The correlation co-efficient between (a) ASP_c and $MAPL_c$ is about 70%, (b)ASP_c and $SDPL_c$ is about 78%, (c)$Risk_c$ and $MAPL_c$ is about 87%, (d)ASP_c and $SDPL_c$ is about 77% showing a positive relation.

deploying Shuffle varies the reachability of VMs and changes the upper layer of HARM. Thus, Shuffle may change the path-based metrics. Moreover, as Diversity technique cannot affect the upper layer of HARM, it cannot vary the path-based metrics. Consequently, the results of deploying those MTD techniques combined with Shuffle (S, $S{+}D$, and $S\Delta D$) on the path-based metrics are similar. Thus, we only evaluate the path-based metrics for Shuffle technique. We computed the path-based metrics after deploying Shuffle on each VM in the cloud. In Figure 4.8, we plot the ASP and $Risk$ values obtained after deploying Shuffle on each VM against the corresponding values for $MAPL$ and $SDPL$ and compute the correlation coefficient of those metrics. We found a positive linear correlation between them which means increment on path-based metrics like $MAPL$ and $SDPL$ may increase the risk and attack success probability. Figure 4.9 shows the normalized values of path-

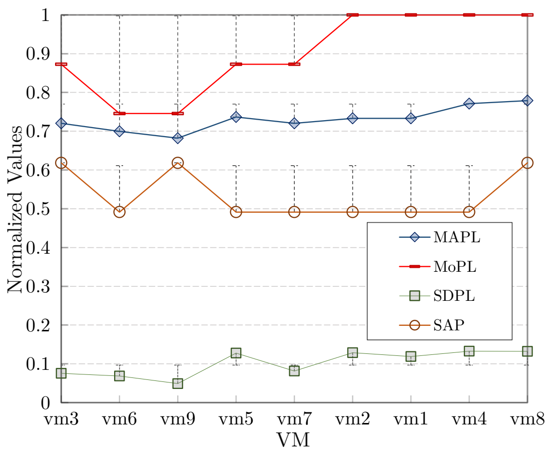
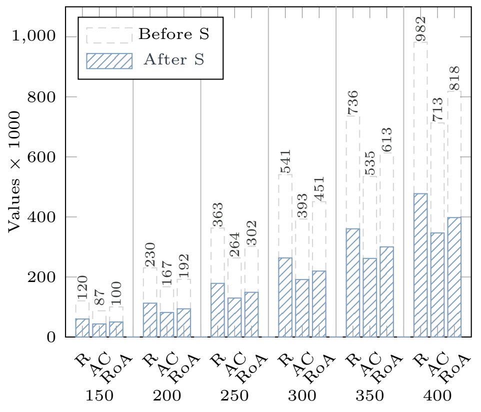


Figure 4.9: Changes on the Path-based Metrics after deploying Shuffle on each VM (VMs in x-axis are sorted in descending order based on their importance using IMs)

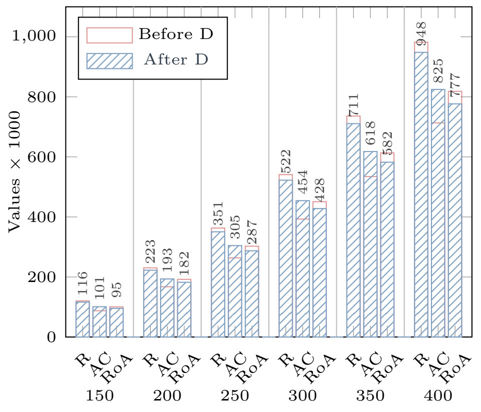
based metrics and the changes in those metrics after deploying Shuffle on each VM. The values of x-axis are the VMs sorted in descending order based on their importance using IMs (as Equation (4.1)). The results demonstrate a gentle increment on the both $MAPL$ and $SDPL$ values. As the higher values of $MAPL$ and $SDPL$ may increase the ASP and $Risk$ values, deploying Shuffle on the VMs vm_9, vm_6, and vm_3 which have higher betweenness values leads better $MAPL$ and $SDPL$ values. Moreover, although the SAP values remain steady after deploying Shuffle on each VM, deploying Shuffle on VMs vm_3, vm_9, and vm_8 have the higher SAP values showing the minimum number of VMs that the attacker needs to exploit to compromise the target are higher for those VMs.

4.5.2 Simulation and Evaluation in Large Cloud Model

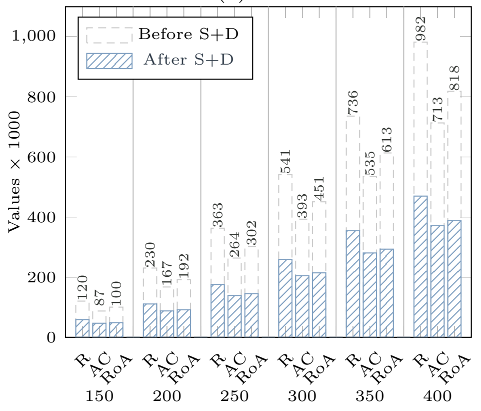
We assess the effectiveness of individual and combined MTD techniques in a larger context. The evaluation in the larger scale can show that (1) how scalable the proposed MTD techniques and strategies are, (2) how the cloud security posture varies as the size of the cloud changes (as the management of the larger clouds is more difficult than smaller clouds). We simulated a large Cloud-band model we used in the earlier version of this chapter presented in [11]. This cloud-band model includes up to 900 VMs divided into two cloud-band nodes each of which can accommodate up to 450 VMs. We assume that only a few VMs are the entry points of the system (i.e., front-end servers) and are connected to the Internet. Moreover, we assume that the attackers can only enter the cloud from the VMs connected to the internet. Attackers can exploit vulnerabilities on each VM, explore the attack paths, and finally get access to the resource node. We also assume that all VMs in the cloud-band nodes use the Linux as the main OS and Fedora as the backup OS. The vulnerability values for each OS are considered based on the values in Table 4.1. We also assume that the backup OS for each VM in the cloud can be launched



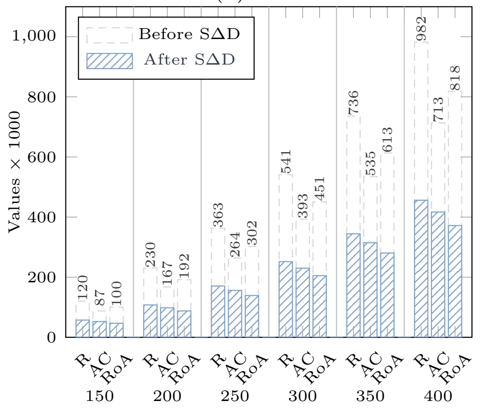
(a) S



(b) D



(c) S+D



(d) SΔD

Figure 4.10: Comparison of metrics before and after deploying the four types of MTD metrics

by the cloud provider with the neglectable machine downtime. Finally, the cloud provider allows VM-LM feature between cloud-band nodes as if there is available space on each node. Note that VM-LM may rearrange the logical connectivities between the VMs based on the cloud constraints. In Section 4.4, we showed that deploying MTD techniques on the VMs with higher IM values leads to better security posture in comparison with other VMs. To deploy MTD techniques effectively, we used IMs rather than ES. We deploy MTD techniques in the cloud-band example and consider the effectiveness of each MTD combination by comparing the security metrics. We first generate the two layered-HARM for the cloud-band model and analyze the security posture of the cloud before and after deploying each MTD technique.

To conduct the results, we simulate the various cloud-band sizes and deploy for deploying MTD techniques. Figure 4.10 compares the security posture of the cloud with various VM sizes before and after deploying the MTD techniques: S, D, $S{+}D$, and $S\Delta D$. Considering the graphs, we can obvious that deploying MTD decreases *Risk* and *RoA* values for all techniques. Moreover, AC increases in D, while this value decreases in the other techniques. However, in D technique, as in Figure 4.10b, the decrements in $Risk$ and RoA rates are less in comparison with the others which infer that the other techniques have better results than D based on the defined cloud and system constraints. Then, the results show that using S in MTD can be helpful for cloud providers as it highly decreases the values of $Risk$ and RoA, as in Figure 4.10a. We also can see that increasing the cloud-band sizes provides the linear increment on all metrics. This trend is almost similar in S and S+D, but different in D and $S\Delta D$. For instance, the overall values for the metrics are very high in D only. Although it increases the AC values, the values of RoA after deploying D are very high. For instance, this value is about 777 for the cloud-band with 400 VMs which shows the high orientation of the attacker to attack the cloud again, while the same RoA values are about halved for other techniques.

The best scenario. Although Diversity (D) provides the worst results in $Risk$ and RoA comparing with the other techniques, it increases the AC rates. Thus, combining D with S improves the overall security posture of the cloud. When the IMs are calculated between the techniques $S\Delta D$, it leads to the best combination scenario, see Figure 4.10d. Moreover, the security posture of S+D is better than S or D-only, see Figure 4.10c, but lower than $S\Delta D$. Comparing the results, we observe that the best MTD combination is $S\Delta D$ technique in which the values of $Risk$ and RoA are lower than the others, the RoA values in $S\Delta D$ are 47 and 372 for the cloud-band with 150 and 400 VMs respectively, which shows the optimal values. Moreover, the optimal values for $Risk$ are 57 and 456 for 150 and 400 VMs in the cloud-bands respectively. Furthermore, the values for AC when $S\Delta D$ are deployed are better than those of S and S+D techniques.

4.6 Discussion and Limitations

In this chapter, we proposed the novel combinations of MTD techniques which utilize different strategies to combine Shuffle and Diversity techniques. The four MTD deployment scenarios include Shuffle and Diversity as the single MTD techniques together with S+D and $S\Delta D$ as the combined strategies. The approaches we used to deploy the MTD techniques on the cloud are limited to VM-LM and OS diversification for Shuffle and Diversity respectively. We utilized four main security metrics $Risk$, AC, RoA, and ASP to evaluate the security posture of the cloud. We chose to use those metrics due to evaluating the cloud security posture based on two different perspectives. Two AC and RoA show the security level of the cloud based on the attacker's perspective, while $Risk$ and ASP are categorized as the metrics showing the cloud security based on the cloud providers' perspective. Then, we evaluated the results of deploying MTD techniques on the path-based metrics to observe the cloud security based on the reachability of the VMs. Path-based metrics can show the attacker's efforts needed to compromise the target in terms of exploiting the paths, the number of VMs on a path, minimum path length, and so on. We further discuss the limitations and extensions of this work.

Table 4.8: Comparison of the studied MTD techniques and evaluations

Reference	MTD Techniques	Combination Strategies	Security Metrics								
			Risk	ASP	Rel	AC	RoA	MAPL	MoPL	SDPL	ASP
This chapter	S, D	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
Chapter 5 [11]	S, R, D	✗	✓	✗	✓	✓	✓	✗	✗	✗	✗
Chapter 3 [13]	S, Risk	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗
Alavizadeh *et al.* [12]	S, D	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗

4.6.1 Combining MTD Techniques

A comparison of current MTD literature which proposed the combination MTD technique is presented in Table 4.8. This comparison is based on factors such as MTD techniques, combination strategies (as discussed in Section 4.5), and security metrics including path-based metrics. We investigated the combining MTD techniques as each technique can be effective on a particular sight. For instance, Diversity incurs more costs for the attacker and can be evaluated in terms of the attacker's point of view. Some security metrics such as AC and RoA are useful indicators to quantify the security of the cloud based on the attacker's perspective. We showed how Diversity increases AC (shown in Figure 4.10b). In fact, the main drawback of Diversity is that it may need extra resources (such as more Operating Systems (OS) variants) and lack of enough variants may affect the Diversity functionality. Alternatively, Shuffle may be deployed in such a way to improve the overall security of the system such as reducing the Risk and RoA values. The inherent benefit of Shuffle is that it can be easily applicable and usually no need extra resources. However, as Shuffle relies on the existing resources, it cannot eliminate or change the vulnerabilities of the current resources. Consequently, those MTD techniques can be well-mingled together by the cloud provider to benefit from the advantages of each technique.

4.6.2 Limitations and Extensions

In this chapter, we only considered Shuffle, Diversity and combination of those two techniques, while the other combinations of MTD techniques can also be formalized and investigated. Although analyzing more MTD techniques and combinations of them might increase the security analysis complexity, it might provide the cloud providers with better security solutions and defensive capabilities and the cloud provider can opt the desirable MTD techniques for combination based on the available resources.

Despite their importance, the security metrics we used in our experiments are only limited to $Risk$, AC, RoA, and ASP together with path-based metrics while there are other security metrics which can be used based on the cloud security level requirements, such as those discussed in [118]. We will further consider other sets of security metrics to capture different aspects of the cloud security requirements in our future work. Moreover, we assumed that the attacker is outside of the cloud. However, adapting an MTD technique, which is able to defend against the insider threats, is also important. The techniques to identify the propagation source of attacks in a network can be utilized to enhance the defensive techniques against the prospective attack from a single or a group of suspicious sources [80, 153]. Moreover, the techniques for detecting and preventing attacks launched from inside of a network are extensively discussed in [99]. We believe that our MTD techniques can be expanded based on intrusion alerts, then the MTD triggers when a threat is detected by the Intrusion Detection System (IDS).

We considered and evaluated Diversity technique with only a single backup VM. Thus, the security achieved by the Diversity technique can be significantly degraded by the limitation on the number of variants in such a way that the attacker may gradually learn the techniques to exploit and cover a set of limited vulnerabilities. To address this, we further investigate Diversity technique with more backup OS variants. Moreover, other Diversity approaches such as Applications, Code and Software Diversity [70] can be included.

Moreover, there is a lack of the economical evaluation of deploying MTD techniques on the cloud. For example, in order to deploy Diversity technique the cloud provider should purchase more OS or licenses. It is difficult for most of the private cloud providers to supply various OS variants for their customers due to economical-related reasons; vendors can provide the customers with more OS options, but customers may not afford to buy them. Thus, cost evaluation of deploying MTD techniques is a crucial problem which we plan to address in our future work.

4.7 Conclusion

MTD techniques can be applied to cloud computing to enhance the security of the cloud by making the cloud more unpredictable for the attackers. In this chapter, we introduced the combinations of MTD techniques including Shuffle and Diversity and evaluated the effectiveness of them by deploying different combination strategies for the cloud. Comparing the security posture of the cloud before and after deploying each combination scenario, we showed that combining MTD techniques is important, as it can introduce additional benefits of enhancing security, which may not be possible under a single MTD technique. For instance, Diversity technique can enhance some security metrics such as AC and RoA which are mainly used to show the cloud security level based on the attacker's perspective, while Shuffle technique may vary other aspects of security metric like $Risk$ and ASP which mainly capture the security from cloud providers' point of view. Thus, MTD techniques can be well-mingled together aiming to increase various aspects of security. We also wanted to show that, if those techniques are not properly deployed, it may cost the cloud providers and also it may increase the attack surface. Moreover, we simulated a large cloud-band model to conduct the results. Our experimental analysis showed the best combination strategies for Shuffle and Diversity providing better results in terms of improving the security posture of the cloud in comparison to single MTD techniques or other combinations.

Chapter 5

Security and Economic Modeling and Analysis of MTD Techniques for Cloud Computing

Summary

Cloud computing leverages MTD techniques to enhance the security posture of the cloud against cyber threats. While many MTD techniques have been applied to cloud computing, there is still uncertainty involving the effectiveness of MTD techniques with respect to security and economic metrics. In this chapter, we first introduce mathematical definitions on the combination of three main MTD techniques: *Shuffle*, *Diversity*, and *Redundancy*. Then, we utilize four security metrics including system risk, attack cost, return on attack, and reliability to assess the effectiveness of the combined MTD techniques applied to large-scale cloud models. We focus on a specific context based on a cloud model for E-health applications to evaluate the effectiveness of the MTD techniques using both security and economic metrics. We introduce (1) a strategy to effectively deploy Shuffle technique using virtual machine placement technique and (2) two strategies for deploying Diversity through OS diversification. Since deploying Diversity may involve a cost benefit analysis for the cloud providers, we propose the *Optimal Diversity Assignment Problem (O-DAP)* and formulate it as a binary linear programming model to find the optimal solution which maximizes the expected net benefit.

5.1 Introduction

Moving Target Defense (MTD) techniques have been proposed aiming to make a system dynamic, less deterministic, and unpredictable for the cyber attackers by continuously changing the attack surface [102]. The static nature of the systems makes a system more attack-prone since the attackers have enough time to learn the potential attack ways, exploiting vulnerabilities, and eventually penetrating into the system. Unlike the traditional defensive security solutions such as anti-malware, Intrusion Detection Systems (IDS), firewalls (which are usually expensive and are reactive methods against possible threats), MTD techniques are mostly proactive defensive techniques which adopt the existing technologies in a system (e.g., virtual machines, backup Operating Systems (OS), and so forth) to introduce adequate levels of unpredictability to the attackers such that they make the attacks difficult and complicated. Consequently, it may increase the attacker's effort, time, and cost while potentially decreasing the defensive costs. However, the proposed defensive MTD strategies need to be effective and efficient to prevent potential attacks while being affordable. In this chapter, we leverage MTD capabilities to secure cloud computing.

Furthermore, the effects of combining different MTD techniques are considered in this chapter by investigating various MTD properties when several techniques are deployed. We use the MTD classification proposed by Hong *et al.* [66], which are Shuffle, Diversity, and Redundancy. We investigate the effects of combining different MTD techniques from these three categories and evaluate them in terms of both security and economic metrics. These include the system risk, attacker cost and effort, return on attack, and reliability together with promising economic metrics such as return on security investment and expected net benefit of security. In general, Shuffle MTD methods may enhance the overall security of a system by reconfiguring the system to change the attack surface of the system, but it may have no effect on the system's reliability, or even deteriorate the reliability of the system. Redundancy MTD techniques may enhance the reliability or availability of the system while it may affect the overall security of a crucial system in a opposite way (e.g. increase the system risk), because it could potentially place the system in a more vulnerable state, as described in [13]. Moreover, Diversity MTD techniques may increase the difficulties of attacks (e.g., software vulnerability exploitation), but it may increase the cost of the defense and therefore have negative economic impacts. Due to the uncertainty of deploying individual or combined MTD techniques, the effectiveness of the proposed MTD techniques must be evaluated prior to deployment. One can benefit from using MTD techniques solely, but the problem arises when a trade-off between security and dependability (such as service availability or reliability) is required. Deploying each MTD technique can affect the others. However, there is still uncertainty involving these analytical issues [12, 13].

In this chapter, we aim to address the aforementioned problems by evaluating the effectiveness of different MTD techniques including Shuffle, Diversity, Redundancy, and combinations of them for cloud computing. Accordingly, we model and analyze MTD techniques using a graphical security model named Hierarchical Attack Representation Model (HARM) [2]. We identify applicable MTD techniques in the cloud computing environment and formally define them. In addition, we use important measures such as network centrality measures (NCMs) to improve the scalability of the evaluation process for large-sized cloud computing systems. First, we carry out experimental analysis to evaluate and compare how combined MTD techniques affect the security of the cloud systems. Then, we focus on more specific context studying the evaluation the effectiveness of MTD techniques on an E-health cloud model as a case study. We utilize both security and economic metrics to show the effectiveness of the proposed MTD techniques. We propose an effective Shuffle strategy to deploy this technique in an appropriate way aiming to reduce economic impacts while increase security level. Moreover, we extend our study by conducting in-depth investigations on Diversity technique considering the interplay between cost of security and benefit of security. Diversity technique may incur additional cost to the cloud providers due to purchasing the license, component's variants (such as VMs), *etc.*, while the cloud providers usually have to deploy defensive strategy with limited (allocated) budgets. Thus, the use of various system's components (such as backup OS variants) should be precisely prioritized and possibly optimized. To this end, we propose a Diversity strategy based on the globally optimal solution of an optimization model which maximizes the expected net benefit under all possible Diversity assignments.

The earlier published version of this chapter is presented in [12]. In this chapter, we extend the earlier version mainly focusing on formalism and definitions of combined MTD techniques together with evaluation the MTD techniques using the economic metrics alongside the security metrics. Moreover, we propose an optimization model which finds an optimal solution to diversity assignment by considering both the cost of security and the benefit of security. The main contributions of this chapter, which to the best of our knowledge have not been proposed by other works, are listed as follows:

- We provide the formal mathematical definitions for combining Shuffle, Diversity, and Redundancy MTD techniques;
- We evaluate the effectiveness of combined MTD techniques using security metrics including System Risk ($Risk$), Attack Cost (AC), and Return on Attack (RoA), and Reliability for a large cloud model through simulation. We evaluate the combined method by deploying Diversity on multiple VMs in the cloud using OS diversification technique.
- We model an E-health cloud example (also called Personal Health Cloud (PHC)), and evaluate the effectiveness of MTD techniques based on both security and economic metrics.

- We provide a set of strategies in which Shuffle and Diversity can be effectively deployed. We propose a VM placement strategy for Shuffle and two strategies for deploying Diversity based on deploying Diversity (OS diversification) (i) with only one backup OS and (ii) with multiple backup OS variants over the set of VMs.
- To solve the second case mentioned above, we propose the *Optimal Diversity Assignment Problem (O-DAP)* and formulate it as a binary linear programming model. This allows us to find an assignment of OS variants on multiple VMs in such a way that the expected net benefit is maximized.

The rest of the chapter is organized as follows. Section 5.2, we provide a comprehensive overview of the related work and define MTD technique frameworks. Section 5.3 presents the preliminaries of this chapter including formalism for combination of MTD techniques. Proposed MTD definition and evaluation criteria based on the security metrics are given in Section 5.4. In Section 5.5, we evaluate the effectiveness of MTD techniques on both economic and security metrics as well as defining optimization model to solve diversity assignment problem. In Section 5.6, we provide the discussion and limitation of this chapter. Finally, the conclusion is provided in Section 5.7.

5.2 Related Work

Several studies have been conducted on MTD theory, techniques, and evaluation [25, 28, 165]. Hobson *et al.* [60] introduced three main challenges of developing MTD as coverage, unpredictability, and timeline. Zhuang *et al.* [168] argue that an effective MTD technique should consider the following issues: (1) the right pieces to be moved, (2) sufficient space for movement, and (3) correct time for movement. Similarly, Cai *et al.* [28] defined three aspects for the movement of MTD techniques: (1) WHAT to move, (2) HOW to move and (3) WHEN to move. However, these studies have not discussed the cost or economic efficiency of the MTD movement. They merely explore common properties of the MTD (movement selection, movement strategy and movement time) that should be achieved when an MTD technique is adopted. In addition to the total cost of security, the relative cost of movement with respect to the achieved security should be taken into account.

Our analysis framework contributes to the literature by including (1) MTD techniques and categories, (2) MTD applicable layers, and (3) definition of the combination of MTD techniques at different cloud's layer (shown in Table 5.1). As those studies in MTD literature have been extensively discussed in Chapter 2, we avoid recurring the related work in this section. The readers who are interested in a specific research and MTD technique mentioned in Table 5.1 can jump to Chapter 2. However, due to the importance of understanding the three MTD categories, we recall those MTD categories as follows [66]:

- Shuffle: this technique refers to any technique which can re-arrange the system setting in different software, hardware, and network layers like changing or shuffling

Table 5.1: MTD techniques applicable in different cloud computing layers

Cloud Layer	Diversity	Redundancy	Shuffle
Application Layer	Web Services [36, 71, 145] Web Applications [19, 36, 71]	Web [56, 154]	Port/IP [16, 82, 100] Web App. [116] HTTP [78, 151]
Platform Layer	Application/ Web service Design [36, 71] Database [36, 145]	Web Server Replica	Web Service
Infrastructure Layer	Operating System (OS) Virtualization [13, 66, 71]	Virtualization [13, 66] SDN [75, 133]	SDN, VM migration [13, 55, 66, 120] Virtual IP [75, 76]

the IP address, re-arranging the network's topology, moving or migrating a VM, Host, Hardware to another location, *etc.* [45, 75, 116].

- Diversity: this technique can be considered as replacing the components' variant (which can be a server, programming language, operating system, hardware, and *etc.*), while the system provides equivalent functionality with the previous state (before changing variant) [74, 110, 126].
- Redundancy: through this technique, one can increase the number of components' replica in the system, like servers, hardware, OS, software, services, and *etc* [85,154].

However, the existing MTD techniques proposed in the literature mostly focused on proposing novel approaches and methods for operating MTD techniques, while evaluating their effectiveness are not adequately discussed. Moreover, there is still a shortcoming in the literature to effectively combine MTD techniques for cloud. Among the existing MTD methods proposed for different layers, a few of them investigated on securing infrastructure layer of cloud and there is no work to evaluate the effectiveness of the combination of all three MTD categories together to assess how effective the combined method is. Moreover, there is a gap in the MTD studies in evaluating the proposed MTD techniques based on security and economic metrics. Only a few studies such as [24,50] have proposed evaluating the economic impacts of defensive techniques using Graphical Security Models (GSMs) and economic metrics. Security metrics can be incorporated into GSMs to evaluate both the effectiveness of given network models and that of the MTD techniques.

5.3 Definitions and Formalization

5.3.1 A Cloud Model

Throughout this chapter, we use a running example based on the cloud-band model consisting of two main cloud band nodes each accommodating up to 450 VMs and one resource node connecting to a Database (DB). The system model can be used as an input to generate the HARM described in Subsection 5.3.2, which can be used for evaluating the

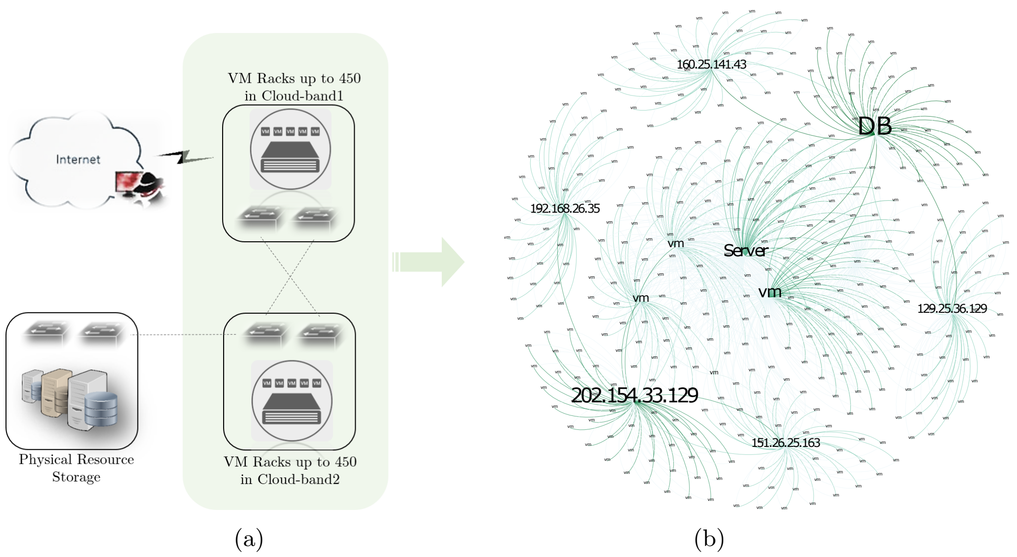


Figure 5.1: The Cloud-band model. (a) Cloud-band model including two cloud-band nodes and one resource node, (b) Generated upper layer of the HARM for the cloud-band model including 400 VMs.

security of the cloud. Figure 5.1 demonstrates the abstract cloud-band model and related upper layers of HARM. In this model, we assume that an attacker is outside of the cloud-band. The attacker can penetrate into the cloud by exploiting vulnerabilities of VMs in the first cloud band node. We assume there exist vulnerabilities that can be exploited by the attacker to give them the root privilege. We use the information from reported vulnerabilities and rankings which are populated from the vulnerability databases such as the National Vulnerability Database (NVD) [106]. Further assumptions are as follows: (1) cloud provider permits Virtual Machine-Live Migration (VM-LM) for cloud-band nodes, (2) VM-LM downtime is negligible, (3) cloud provider purchases enough licenses for backup OS, and (4) VMs' OS can be replaced with other variants if needed.

5.3.2 HARM Construction

In order to perform the security analysis of the cloud and further evaluate the effects of MTD techniques on the cloud, we construct a two-layered HARM [2] of the cloud-band model. Using the HARM, we can compute security metrics for comparison. We show the computations of the four security metrics of the running cloud-band example. Later in Section 5.4, we evaluate how they change.

In this section, we used HARM to compute security metrics and evaluate the MTD techniques. To recall the notations and definitions oh HARM, refer to Definitions (1) – (3) presented in Chapter 3.

5.3.3 Importance Measures

As stated earlier, the upper layer of HARM represents a comprehensive scheme of the connectivity of VMs in the cloud through a graph. To efficiently carry out the security analysis, it is important to identify the most important components (in here VMs) of the network. We use two important Network Centrality Measures (NCM), betweenness and closeness [27] for identifying the VMs playing a more crucial role in the cloud. The utilization of those NCMs has already shown in Chapters 3 – 4 and in [12,13]. In Section 5.4, we show how Important Measures (IMs) can be utilized to find out more effective combined MTD strategies including Shuffle, Diversity, and Redundancy.

NCMs can be incorporated into the upper layer of HARM. Then we can calculate the Closeness Centrality (C_c) of a specific VM in the network as in Equation (3.2) and Betweenness Centrality (C_b) of a VM as in Equation (3.3) which have already defined in Chapter 3.

We have defined the cloud setup for the running example, the HARM for modeling and analysis for the cloud security, and also the use of IMs for efficiently evaluating the security for large-sized clouds. We describe the analysis of MTD techniques in the next section.

5.3.4 Security Metrics

In this section, we utilize four security metrics to evaluate the security of the cloud after deploying MTD techniques and identify the most suitable technique deployment strategies. We analyze the security of the cloud from four security perspectives: (i) System Risk ($Risk$), (ii) Attack Cost (AC), (iii) Return on Attack (RoA), and (iv) System Reliability (*Reliability*). $Risk$ is based on the vulnerabilities of the network's components [66]. AC measures the difficulties of attackers to attack a system and can be quantified in terms of cost incurred by an attacker to exploit a network component or the whole system [156]. RoA shows the willingness of the attacker to use the same components, attack path, and vulnerabilities to penetrate the network. In fact, RoA quantifies the cost of an attack versus the benefit of the attack [42]. *Reliability* quantifies the reliability of the network's components (i.e. critical components) under certain attack circumstances. Calculation of system *Reliability* can be performed using the SHARPE tool (Symbolic Hierarchical Automated Reliability and Performance Evaluator) [130]. The details of computing these four security perspective metrics are recalled in Table 5.2.

5.3.5 MTD Formalism

We utilize VM-LM as the main technique for deploying Shuffle. Shuffle technique can be formulated based on the HARM definition. The formalism for Shuffle has already been given in Definition 4.4.1 presented in Chapter 4.

Table 5.2: Recalling some notations, formulas, and metrics

Notations	Metrics Description	Reference and Formula
AP	An attack path from attacker to target (DB)	-
AP_c	All possible attack paths in a cloud	-
$P(vm)$	Attack success probability for a single VM	lower layer of HARM
I_{vm}	The impact of a successful attack on a VM	NVD [106,113]
$Risk_{vm}$	The risk value associated with a VM	$p(vm) \times I_{vm}$
$Risk_p$	The risk of a single attack path	$\sum Risk_{vm_i}, vm_i \in AP$
$Risk_c$	The overall risk value of a cloud	$\sum Risk_p, p \in AP_c$
AC_{vm}	Attack cost of exploiting a single VM	NVD, [156]
AC_p	Attack cost of an attack path	$\sum AC_{vm_i}, vm_i \in AP$
AC_c	The overall attack cost for a cloud	$\sum AC_p, p \in AP_c$
RoA_{vm}	Return on attack value of a VM	$\frac{p(vm) \times I_{vm}}{AC_{vm}}$
RoA_p	Return on attack for an attack path	$\sum RoA_{vm_i}, vm_i \in AP$
RoA_c	The overall return on attack for a cloud	$\sum RoA_p, p \in AP_c$

In here, we reformulate Diversity definition in such a way that Diversity function is applied on a set of VMs as follows.

Definition 12. We formulate the Diversity technique in which the diversity function is applied on H as $D(H, K) = H^d$, where K denotes a set of VMs $K \subseteq VM$ selected for replacing with another OS variant. Then, for each $\kappa \in K$ we have $H^d = (U, L^{d,\kappa}, C)$ which is the result of deploying Diversity technique on a denoted VM, where $L^{d,\kappa} = \{\ell_1, \ldots, \ell_{\mathbf{k}}, \ldots, \ell_n\}$ denotes the ATs corresponding to each VM, and $\ell_\kappa = (V_{\kappa,\theta}, G, root)$ is the transformed AT of vm_κ which is replaced with another variant $\theta \in OS$. Diversity technique affects the lower layer and varies vulnerabilities $V_{\kappa,\theta}=\{\nu_{1,\theta}, \nu_{2,\theta}, \ldots, \nu_{m,\theta}\}$, while $U=(VM, E)$ is preserved.

We formulate the combination of shuffle, Diversity, and redundancy (S+D+R) as a function on HARM as follow:

Definition 13. Let S+D+R(H, k_s, k_d, k_r, r) be a S+D+R function on HARM where k_r shows the VM that is selected to be replicated for r times, and k_s is the VM selected to be shuffled, and k_d denotes the VM selected for Diversity. Then the result of S+D+R function is as S+D+R$(H, k_s, k_d, k_r, r) = H^{\text{s+d+r}}$, where $1 \leq k_s, k_d, k_r \leq n$ and $0 < r \leq l$. We define $H^{\text{s+d+r}} = (U^{\text{s+d+r}}_{k_r,k_s}, L^{\text{s+d+r}}_{k_r,k_d}, C)$ where $U^{\text{s+r}}_{k_r,k_s}$ is a transformed AG in the upper layer in which S+R is deployed on and $L^{\text{s+r}}_{k_r,k_d}$ is the corresponding transferred AT in the lower layer. Then, the former can be represented as $U^{\text{s+d+r}}_{k_r,k_s} = (VM', E')$, where

$$VM' = VM \cup \left(\bigcup_{r=1}^{l} VM^{\text{r}}_{k_r} \right)$$

and $|VM'| = n + r$, and $E' \subseteq (VM + r) \times (VM + r)$. Next, the latter can be shown as $L^{\text{s+d+r}}_{k_r,k_d} = (V', G, root)$, where $V' = V \cup V^r \cup V^d$, and V^{r} is a set of vulnerabilities caused by replication of a VM, and V^{d} is a set of new vulnerabilities introduced by replacing the OS variants.

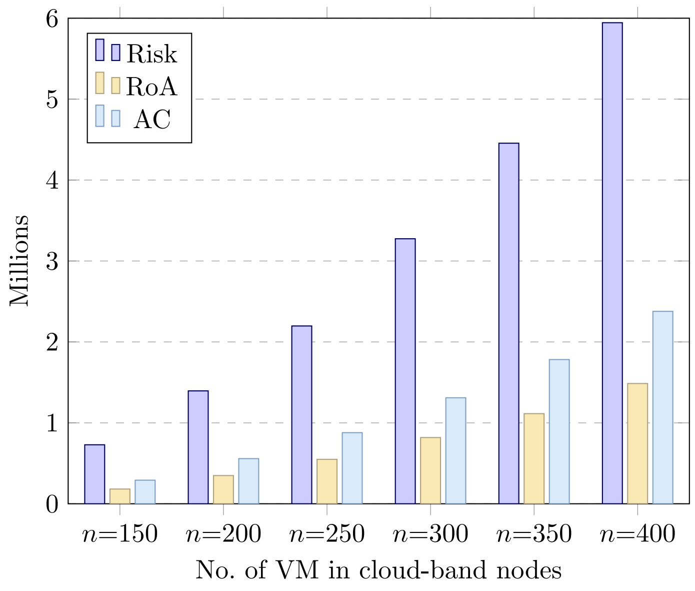


Figure 5.2: Security analysis results of the current cloud-band

5.4 MTD Analysis

5.4.1 Security Analysis of Current System

The HARM can be used to compute the security metrics such as *Risk*, *AC*, and *RoA*. For the running example considered here, we assume there are vulnerabilities which if exploited would grant the attacker the root privilege. For the comprehensive security overview, the computation will incorporate the analysis of all possible attack paths that are calculated using the upper layer of the HARM (the formulas are shown in Table 5.2). For the reliability analysis, we use the SHARPE software package [130]. We assume that the attack rate for cloud-band models follows an exponential function with an average value of 0.2 (i.e., one attack per every five hours). Also, we compute the *Reliability* values during a 10-hour period. We compute the defined security metrics of the current cloud-band system before deploying the MTD techniques on the cloud. We calculate these metrics for different cloud-band sizes ranging from 150 VMs up to 400 VMs in the cloud-band. Figures 5.2 and 5.3 illustrate the posture of the current security level of the cloud-band with various cloud-band sizes. Figure 5.2 compares the metrics obtained through analyzing HARM which shows that larger cloud-band sizes have more *Risk*, *AC*, and *RoA* values. However, Figure 5.3 shows that the variation of *Reliability* value is not significant for different cloud-band sizes. In the following sections, we compare the results obtained from deploying MTD techniques on the cloud-band against the current security posture of the cloud to investigate the effectiveness of the deployed MTD techniques.

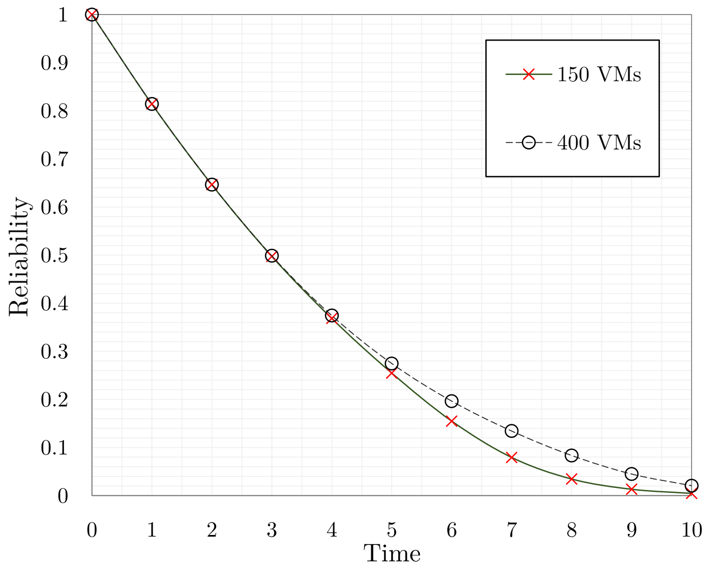


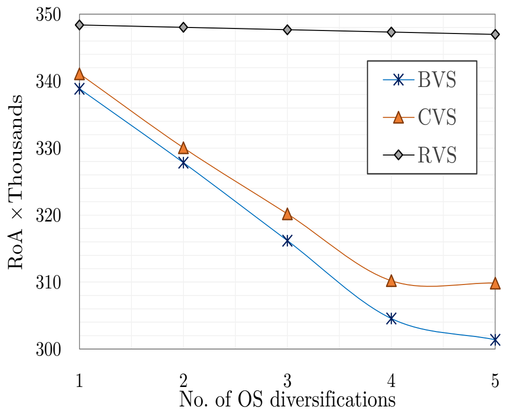
Figure 5.3: The values of *Reliability* for the current cloud-band

5.4.2 Diversity on Multiple VMs

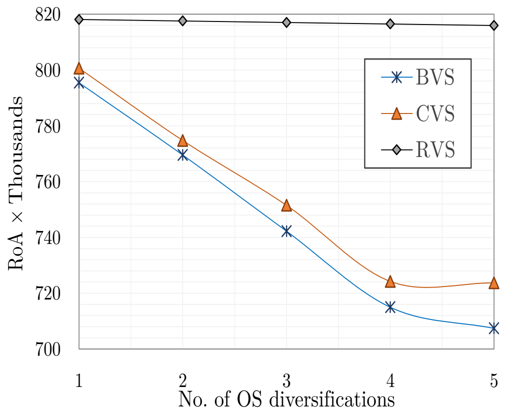
Diversity plays a crucial role in increasing the required efforts for the attackers. The attackers must spend time, money, and efforts to gain sufficient knowledge to exploit the vulnerabilities of a system (i.e. network's component, a VM, service, and OS). Any sudden change in those component confuses the attacker, which also increases the time and effort needed for carrying out the attack. We consider using the OS diversification method as the main technique for deploying Diversity. In this section, we consider deploying Diversity techniques on multiple VMs (i.e., for different subsets of VMs). To evaluate the Diversity technique, we deploy OS diversification on multiple VMs based on three selection criteria (SC): (i) Random VM Selection (RVS), (ii) Betweenness VM Selection (BVS), and (iii) Closeness VM Selection (CVS). The RVS method selects a set of random VMs in the cloud. The BVS method selects the set of VMs based on their higher Betweenness ranks, which is one of the NCM measures. Similarly, the CVS method uses the Closeness ranks of the VMs in the cloud-band.

Deploying Diversity preserves the upper layer of HARM. In here, we assume that the cloud provider has up to five OS variants backup. Note that increasing OS variants costs cloud provider (i.e. purchase OS licenses for OS variants). For simplicity, in this section, we do not consider the cloud provider cost.

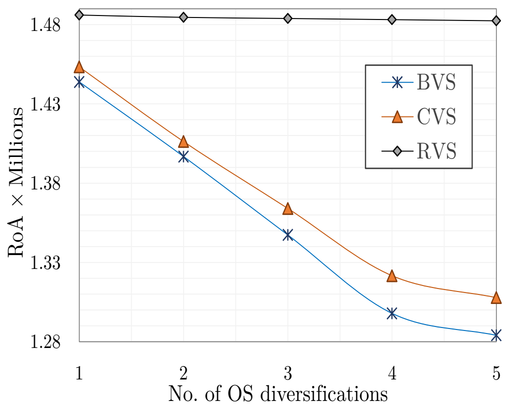
Our approaches, formulas, and references of calculating AC, $Risk$, and other security metrics are given in Table 5.2. Other approaches to calculating AC can be found in [69]. Figure 5.5 shows the AC and $Risk$ metrics after deploying Diversity on multiple VMs ranked based on higher Betweenness values. We denote xV as deploying new variants on x selected VMs respectively. Then, $5V$ represents deploying five OS variants on the five



(a) $n = 200$



(b) $n = 300$



(c) $n = 400$

Figure 5.4: Comparing the result of RoA metrics after deploying Diversity on multiple nodes selected based on three different criteria on the cloud-band with various number of VMs

selected VMs. We observe that deploying Diversity on the VMs with higher Betweenness ranks results in a larger increase in cost for the attacker and increases the AC value. This increase will be higher if we add to the the number of OS variants and deploy the Diversity technique on multiple nodes. Deploying OS diversification by assigning $5V$ in a cloud band with 400 VMs increases AC value from around 2.3 million to 3 million, while the increase for $1V$ does not go beyond 2.5 million.

We do not observe a substantial change in $Risk$ values after deploying Diversity techniques. The $Risk$ value remains almost steady after increasing OS variants, but it still increases exponentially by increasing the cloud-band nodes, see Figure 5.5.

Figure 5.4 compares the results of RoA metric through deploying Diversity on three VM selection groups: BVS, CVS, and RVS in different cloud-band sizes. These observations indicate that deploying Diversity on a set of VMs selected using BVS provides the best results in comparison to the other groups. The values of RoA for RVS groups

Algorithm 3: VM selection pseudocode based on RVS, BVS, and CVS in the HARM

```
   Data: $H = (AG, AT, M)$, SC                        /* SC: Selection Criteria */
   Data: $m$                                    /* m: Number of required VMs */
   Result: $K$                                     /* k: Set of selected VMs */
1  begin
2      if $SC = RVS$ then
3          $k \leftarrow$ Select(rnd $i, i \in VM$, $m$)        /* opt $m$ random VMs */
4      else
5          foreach $vm_i \in \{vm_1 \dots vm_i\}$ do
6              if $SC = BVS$ then
7                  $\beta = C_b(vm_i)$
8                  Add $\beta$ into $Z_i$
9              else if $SC = CVS$ then
10                 $\zeta = C_c(vm_i)$
11                 Add $\zeta$ into $Z_i$
12             end
13         end
14         $k \leftarrow$ Select$\big([\max_{i \in VM}\ Z_i]$, $m\big)$        /* opt $m$ top VMs */
15     end
16     return $k$
17 end
```

have very gentle decrements while OS diversification increases. However, both RoA values for BVS and CVS groups decrease sharply when increasing OS diversification numbers. Other results for different cloud-band nodes show the same trend, but as it is expected, RoA values for cloud-band including more VMs are higher (i.e. RoA values of cloud-band with 400 VMs are between 1 and 1.5 million, Figure 5.4c, while this rate is between 300 and 350 thousand for cloud-band including 200 VMs, Figure 5.4a).

5.4.3 Combining Shuffle, Diversity, and Redundancy

We combine the main three MTD techniques, Shuffle, Diversity, and Redundancy (S+D+R) to investigate the enhanced the security of the cloud. It is important to quantify the effects of the combined MTD techniques and compare them with the current security level of the system. In order to combine the three techniques, we set combination criteria based on the results obtained from previous sections. The results discussed in Subsection 5.4.2 showed that OS diversification on the VMs having higher betweenness (grouped in BVS) has a better effect on security metrics. They also revealed that increasing the numbers of OS variants increases AC and decreases RoA. Based on those results, we only consider BVS group for deploying Diversity in this section. Moreover, based on the results of previous studies reported in [13] for combining Shuffle and Redundancy techniques, we deploy Shuffle technique on the most suitable VM which can be found through analyzing only the top 10% of the VMs holding higher values of Betweenness.

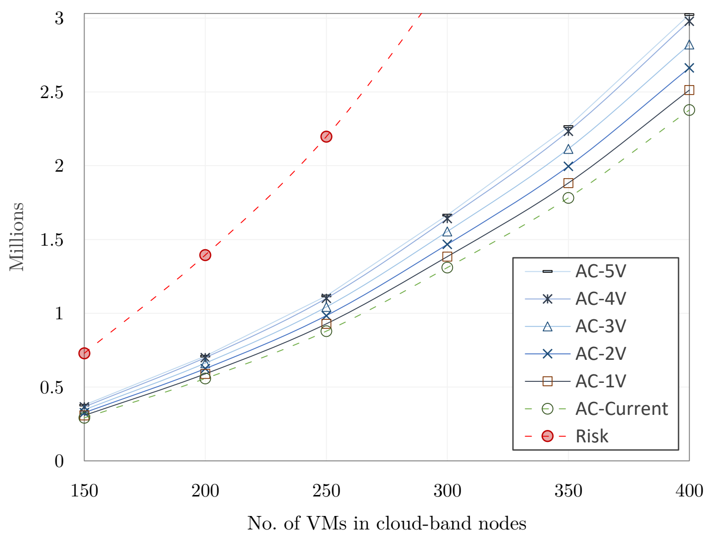


Figure 5.5: Comparison of AC and $Risk$ values obtained after deploying the Diversity technique on the multiple VMs having the highest Betweenness values for the cloud-band example with a various node sizes

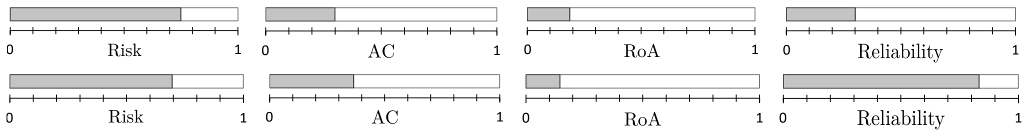
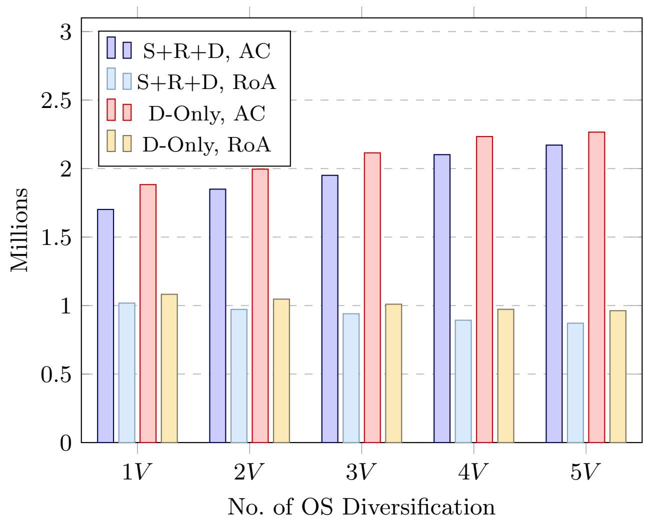


Figure 5.6: Line chart comparing normalized metrics of the cloud-band with $n = 350$ before and after deploying S+D+R: upper line charts show the current cloud-band and lower line charts show the metrics after deploying MTD techniques

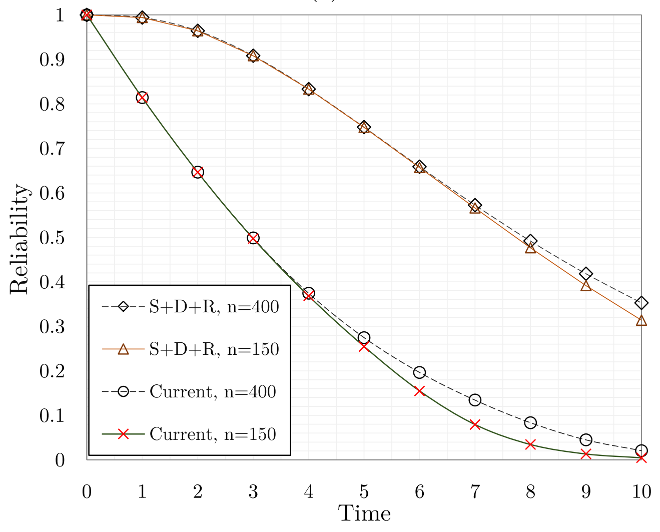
Finally, We deploy Redundancy on DB or VMs connected to DB (target) to increase the Reliability of the cloud and availability of DB against DDoS attacks.

We deployed the S+D+R technique with $5V$ on the cloud-band and evaluate the results. Figure 5.6 compares the results of deploying S+D+R on the cloud-band with 350 VMs. It is clear that all security metrics are improved after deployment.

Figure 5.7a compares AC and RoA metrics after deploying S+R+D with $5V$ against Diversity (D-Only). We observe that the values of AC in D-Only are lower than AC in S+D+R. However, the corresponding RoA values for S+D+R is also lower in D-Only which shows the attacker has less tendency to attack again. Figure 5.7b compares the $Reliability$ values of the cloud-band with 150 and 400 VMs before and after deploying S+D+R technique. We select the boundaries of $n = 150$ and $n = 200$ as the other values of n fall between these two boundaries. We can observe how $Reliability$ values increase after deploying S+D+R. After passing 10 hours of assumed attack with α=0.2, the cloud-bands which are secured with S+D+R are around 40% available, while this rate reaches almost 0% for current cloud-bands without deploying MTD techniques.



(a)



(b)

Figure 5.7: (a) Comparing the results of *AC* and *RoA* for D-Only with S+D+R. (b) Comparing the *Reliability* values after S+R+D against *Reliability* value for current cloud-band for $n = 150,\ 400$.

Nevertheless, combining MTD techniques can be quantified based on the security levels expected by cloud providers of network administrators. Deploying MTD techniques may be costly and it is important for cloud providers to find a trade-off between security and economic requirements.

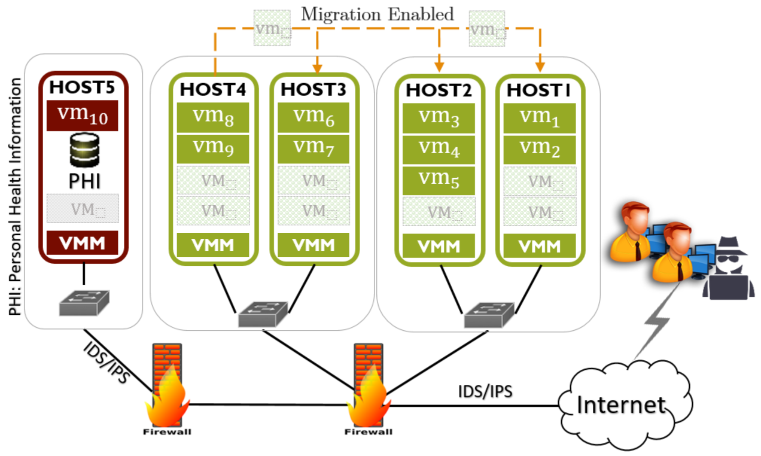


Figure 5.8: An E-Health cloud model including PHI records

5.5 Economic Metrics for MTD techniques

Although security metrics show different dimensions of a cloud's security posture, investigating economic aspects of deploying MTD techniques are also crucial. In here, we compute various economic metrics to show different perspectives of MTD deployment scenarios for a cloud example.

5.5.1 A Case Study on E-Health Cloud Model

We consider that the personal health information (PHI) of patients including medical histories are located in a private personal health cloud (PHC) as shown in Figure 5.8. We assume that the attacker is located out of the cloud and can use the vulnerabilities of the cloud's components (e.g., VMs) to get into the cloud and find a path to the PHI database. In this section, we aim to evaluate the effectiveness of MTD techniques in terms

Table 5.3: VM Assets and Vulnerabilities (Note that vm_{10} is the target VM and includes PHI records)

VMs	OS (θ)	Asset Value (AV) ($)	Vulnerabilities (V)					
			V-ID	CVE-ID	Threat	E	AC	EF (%)
vm_{1-5}	**W**in10	500	$\nu_{1,W}$	CVE-2018-8490	Remote	0.17	1.6	0.6
			$\nu_{2,W}$	CVE-2018-8484	Privilege Escalation	0.18	2.2	0.59
			$\nu_{3,W}$	CVE-2018-0784	Privilege Elevation	0.28	1.2	0.59
vm_{6-9}	**L**inux	480	$\nu_{1,L}$	CVE-2018-14678	DDoS	0.18	2.2	0.59
			$\nu_{2,L}$	CVE-2018-14633	DDoS & Remote	0.22	3	0.47
			$\nu_{3,L}$	CVE-2017-15126	Use After Free (UAF)	0.22	1.9	0.59
vm_{10}	**L**inux	10000	$\nu_{1,L}$	CVE-2018-14678	DDoS	0.18	2.2	0.59
			$\nu_{2,L}$	CVE-2018-14633	DDoS & Remote	0.22	3	0.47
Backup	**F**edora	450	$\nu_{1,F}$	CVE-2014-1859	Symlink attack	0.18	4.5	0.3

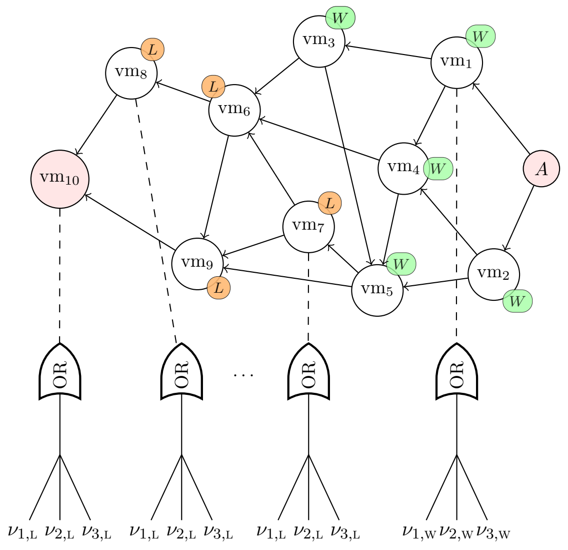

vm$_1$
vm$_2$
vm$_3$
vm$_4$
vm$_5$
vm$_6$
vm$_7$
vm$_8$
vm$_9$
vm$_{10}$
A
W
L
OR
$\nu_{1,L}$ $\nu_{2,L}$ $\nu_{3,L}$
$\nu_{1,W}$ $\nu_{2,W}$ $\nu_{3,W}$

Figure 5.9: Generated Two-layer HARM for the Cloud

of economic metrics. We first model the cloud using HARM demonstrated in Figure 5.9. Table 5.3 demonstrates the vulnerabilities existing on each VM [113]. Moreover, we assume that the cloud provider has one backup OS which can be used for Diversity. We assume that PHI records are stored in a DB connected to vm_{10} and any successful attack exploiting vm_{10} incurs significant damage of 10000$ to the organization (due to loss and/or disclosure of patients health information).

5.5.2 Single Loss Expectancy

The Single Loss Expectancy (SLE) measures an organization's loss from a single threat [93]. SLE can be determined for a cloud based on the Asset Values (AV) for each VM including costs of maintenance, running OS, services, DB record values, applications, and so forth. The estimated AV for each OS is shown in Table 5.3. We assume that SLE can be calculated for both VM and network (cloud) levels. The value of SLE for a VM can be obtained by multiplying asset value and the maximum percentage of loss for that asset caused by a treat which is called exposure factor (EF), see Equation 5.1.

$$SLE_{vm_i} = \left(1 - \prod_{v_{j,\theta} \in V_{i,\theta}} \left(1 - EF_{v_{j,\theta}}\right)\right) \times AV_{vm_i} \tag{5.1}$$

In Equation 5.1, the AV consists of the cost associated with running an active VM (i.e. purchasing a license for an OS, applications, values of DB, *etc.*)

Then, the SLE for the cloud (SLE_c) including all assets (in here VMs) can be calculated as Equation (5.2).

$$SLE_c = \sum_{ap \in AP} \left(\sum_{vm_i \in ap} SLE_{vm_i} \right) \tag{5.2}$$

5.5.3 Annual Loss Expectancy

Annual Loss Expectancy (ALE) can be defined as the expected financial loss due to an attack event and can be computed by the product of SLE and Annualized Rate of Occurrence (ARO) which represents the estimated number of occurrences of a threat event per year [93]. However, due to lack of real data to estimate ARO, we assume ARO value to be 1 (similar to [24]).

$$ALE_c = \sum_{ap \in AP} \left(\sum_{vm_i \in ap} SLE_{vm_i} \times ARO_{vm_i} \right) \tag{5.3}$$

5.5.4 Benefit of Security

Many defensive strategies can be adapted to the cloud to either avoid or mitigate the exploitation or damages to the cloud. In this chapter, we evaluate the benefits achieved by deploying MTD techniques. Benefits of Security (BS) [24] can be used to show the effects of deploying a single or combined defensive techniques. The benefit of security for a cloud BS_c can be computed based on Equation 5.4.

$$BS_c^{\mu} = ALE_c - (ALE_c^{\mu} \times MF^{\mu}) \tag{5.4}$$

In Equation 5.4, ALE_c^{μ} denotes ALE value of the cloud after deploying MTD techniques. $\mu \subseteq \{S, D, R\}$ denoted a set of MTD used as defensive technique. Mitigation Factor MF^{μ} shows the ability of the defensive MTD techniques to mitigate the ALE which can be mapped to $[0, 1]$ as in Equation 5.5.

$$MF^{\mu} = MF^{\mu} = \begin{cases} 1 - \frac{ALE_c^{\mu}}{ALE_c}, & ALE_c^{\mu} > \text{if } ALE_c \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

5.5.5 Cost of Security

Cost of Security (CS) consists of any expenses associated with the factors such as deployment, purchase, maintenance, patching costs, *ets.* for a defensive security mechanism [26]. In fact, cost of security may also include loss of productivity due to system downtime or other security-related activities such as training which causes indirect costs. However, for simplicity, we present the following assumptions for costs of security. We assume that the unit cost of deploying Shuffle technique for a given VM is 20$ per operation which includes the costs of experts and loss of productivity. We also assume that the unit cost of deploying Diversity on a VM such that a given VM is replaced with the backup OS (Fedora in Table 5.3) is 55$ per operation which includes the costs of experts, maintenance, and loss of productivity for a given VM for an operation per year [124, 140].

5.5.6 Return on Security Investment

The overall benefits of the selected defensive MTD strategies against the costs of implementation can be evaluated using Return on Security Investment (RoSI) metric. RoSI can be used to evaluate the profitability of a defensive investment against the costs as formulated in Equation (5.6).

$$RoSI_c^{\mu} = \frac{BS_c^{\mu} - CS^{\mu}}{CS^{\mu}}, \tag{5.6}$$

5.5.7 Shuffle Evaluation

In this section we evaluate the Shuffle technique on the PHC cloud example based on economic metrics. We propose a VM placement strategy based on the shortest path in the upper layer of HARM. We aim to enhance the migration scenarios for Shuffle technique by using the shortest path rather than random VM placement strategy. The idea behind selecting the shortest path strategy for VM placement is to increase the Shortest Attack Path (SAP) [157] alongside other benefits of Shuffle technique.

Migration Strategy. We propose the shortest path injection approach in which a selected VM can be moved and connected to the VMs located in the shortest path in the upper layer of HARM. Since there may be more than one shortest path we utilize a migration strategy to find the most critical shortest path as in Equation (5.7).

Let $SAP = \{sp_1, sp_2, sp_3, \dots, sp_q\}$ be a set of possible shortest paths existing in the upper layer of HARM. Then, we define the strategy T_{sp} as a selected shortest path having lower in-degree values as follows.

$$T_{sp} = \min_{sp \in SAP} \Big(\sum_{vm_i \in sp} \sum_{j \leq n} a_{ij} \Big) \tag{5.7}$$

We deploy Shuffle techniques on all VMs in the upper layer of HARM and evaluate the effectiveness of each migration scenario. We compare both security and economic metrics to find the best deployment scenario. Table 5.4 demonstrates the results of Shuffle on the VMs based on $Risk_c$, AC_c, and RoA_c security metrics, and ALE_c, BS_c, and $RoSI_c$ economic metrics. Table 5.4 shows that the most promising results for security metrics is deploying Shuffle on VM vm_5 which yields the lowest $Risk_v$ and RoA_c values. While, deploying Shuffle on VM vm_9 leads to the best results for economic metrics which yields the highest $RoSI_c$=2631 compared to the other deployment scenarios. However, in the case that VM vm_9 is selected for Shuffle, it still yields appropriate results for RoA which is about 87 (the second best RoA_c value). In fact, the cloud provider can incorporate both RoA and $RoSI$ in the decision making process and prioritize based on their values.

5.5.8 Diversity Evaluation

In Subsection 5.4.2, we evaluated Diversity using security metrics through a large cloud-band model. In here, we extend Diversity evaluation by considering economic metrics and

Table 5.4: The results of deploying Shuffle on each VM in the cloud

VM ID	Security Metrics			Economic Metrics		
	$Risk_c^S$	AC_c^S	RoA_c^S	ALE_c^S($)	BS_c^S ($)	$RoSI_c^S$
vm$_5$	**118.35**	136.7	**77.51**	113878	46315	2315
vm$_6$	126.14	134.7	87.19	114913	45281	2263
vm$_9$	117.65	124.7	81.7	**107563**	**52630**	**2631**
vm$_4$	128.97	142.5	87.15	115712	44482	2223
vm$_2$	135.58	147.3	92.66	116912	43282	2163
vm$_3$	149.98	163.5	102.27	131045	29149	1456
vm$_1$	140.54	150.9	96.79	117812	42382	2118
vm$_7$	145.73	158.5	99.52	130178	30015	1500
vm$_8$	182.07	198.3	124.23	159910	283	13
Best	vm$_5$	vm$_8$	vm$_5$	vm$_9$	vm$_9$	vm$_9$

Table 5.5: The results of deploying Diversity on each VM in the cloud

VM ID	Security Metrics			Economic Metrics		
	$Risk_c^D$	AC_c^D	RoA_c^D	ALE_c^D($)	BS_c^D($)	$RoSI_c^D$
vm$_5$	**167.31**	**253**	**105.19**	**157266**	**2928**	**52.24**
vm$_6$	174.27	236.6	117.37	157867	2327	41.31
vm$_9$	173.62	239.2	116.83	157701	2493	44.33
vm$_4$	171.32	239.8	110.12	157998	2196	38.93
vm$_2$	173.33	233.2	112.59	158364	1830	32.27
vm$_3$	177.35	220	117.52	159096	1098	18.96
vm$_1$	171.32	239.8	110.12	157998	2196	38.93
vm$_7$	175.57	231.4	118.44	158199	1994	35.26
vm$_8$	178.82	218.4	121.14	159030	1163	20.15
Best	vm$_5$	vm$_5$	vm$_5$	vm$_5$	vm$_5$	vm$_5$

optimization. We consider deploying Diversity based on two scenarios: (i) Diversity on single or multiple VMs using a single backup OS and (ii) Diversity on multiple VMs using multiple backup OS (using an optimization model).

Scenario 1. We deploy Diversity only on a single VM through Exhaustive Search (ES). We evaluate the effectiveness of Diversity technique through computing security and economic metrics. Table 5.5 shows the results of deploying Diversity on each VM. The experimental results show that deploying Diversity on VM vm_5 yields the best results in terms of $Risk_c$, AC_c and RoA_c for security metrics. It provides the best result for economic metrics as it yields the lowest ALE_c which is 157266. It also leads to the highest BS_c and $RoSI_c$ values which are 2928 and 52.24, respectively.

Scenario 2. We evaluate the Diversity deployment on various VMs with a single backup OS. To do this, we leverage two strategies to find a set of VMs for deploying Diversity (OS diversification): Random VM Selection (RVS) and Betweenness VM Selection (BVS) (as in Subsection 5.4.2). We compere the results of deploying those two strategies on the PHC cloud example focusing on return on security investment and cost of security.

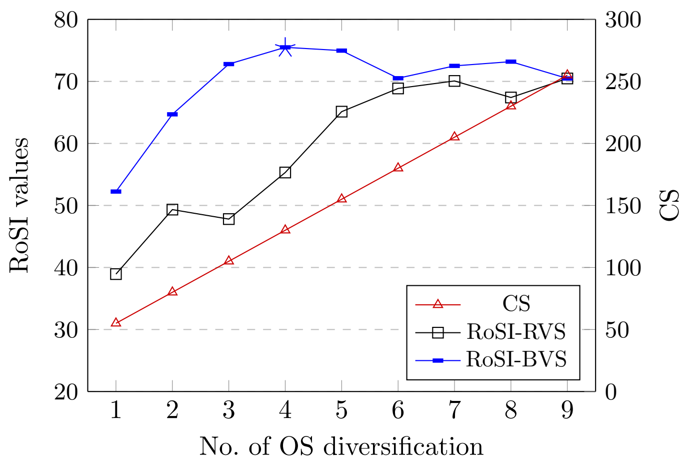


Figure 5.10: Comparing RoSI values obtained after deploying Diversity on various VMs against CS based on RVS and BVS (the asterisked point shows the optimal solution.)

We aim to find a trade-off between the number of OS diversification (using the same back up OS) on a set of VMs against the CS while we maximize the RoSI. Figure 5.10 compares the results of deploying Diversity on RoSI values based on two RVS and BVS strategies on various number of VMs (from 1 to 9 OS diversification). The results show that deploying Diversity on the four VMs having the highest Betweenness values reach the peak and yields the best RoSI values while the cost of security remains between 100$ and 150$. However, the results based on RVS strategies suggest that deploying nine backup OS to increase the RoSI, but it incurs the highest cost of security which is more than 250$. Moreover, the highest RoSI metric after deploying BVS is about 75, while the same metric resulting from RVS strategy reaches 70 in the best case.

5.5.9 Optimal Diversity Assignment

In this section, we analyze the Diversity technique with multiple OS variants on multiple VMs through an optimization model. We model the decision problem of maximizing expected net benefit by assigning backup operating systems to existing virtual machines as a mathematical programming model with binary decision variables. In our model, we consider a graph coloring scheme so that each backup OS variant is assigned with a color (c) in the graph such that no two adjacent VMs are assigned the same color (Backup OS). This aims to increase the difficulty for the attacker who would encounter a different back up OS in adjacent VMs through the attack path.

We use $\theta = \{1, 2, \ldots, k = |\theta|\}$ to represent the set of all potential backup operating systems from which to choose to implement Diversity on some virtual machines.

We use the binary decision variable d_{ic} $\forall i \in VM, c \in \theta$ for virtual machine i and backup operating system c such that d_{ic} takes value 1 if and only if backup operating system c is assigned to virtual machine i. We also use the binary decision variable e_i $\forall i \in VM$ for virtual machine i which takes value 1 if and only if virtual machine i is assigned a backup operating system.

In our mathematical programming model, we penalize diversity assignments in which the same operating system is assigned to adjacent nodes. Accordingly, we use the binary decision variable f_{ij} $\forall (i,j) \in E$ to penalize assigning of same backup operating system on endpoint i and endpoint j of the edge (i,j).

The maximization objective function represents the expected net benefit (ENB) which is calculated based on Equation (5.8) in which M represents a large enough value to be used as "big M" for penalizing same backup operating system being assigned to adjacent nodes.

$$\begin{aligned} ENB =& ALE_{\text{after}} - ALE_{\text{before}} \\ &- \text{cost of security} - \sum_{(i,j)\in E} M f_{ij} \\ =& \sum_{p\in\text{paths}} \sum_{i\in p} SLE_i^d ARO_i - \sum_{p\in\text{paths}} \sum_{i\in p} SLE_i ARO_i \\ &- \sum_{i\in VM} \sum_{c\in\theta} CS_c d_{ic} - \sum_{(i,j)\in E} M f_{ij} \end{aligned} \tag{5.8}$$

The objective function in (5.8) is formed first by subtracting the cost of security incurred from implementing Diversity technique from the benefit of security which was formulated in (5.4). Secondly, the penalty of assigning same backup operating system on adjacent nodes is applied to the objective function.

We formulate the term SLE_i^d using the binary decision variables d_{ic} and e_i in (5.9). According to this linear formulation, the SLE^d for virtual machine i remains unchanged if no backup operating system is assigned to it ($e_i = d_{ic} = 0$ $\forall c \in \theta$). However, if backup operating system c is assigned to virtual machine i ($e_i = d_{ic} = 1$), the value of SLE^d is updated according to the asset value and exploitability factor of the assigned backup operating system c.

$$SLE_i^d = SLE_i(1 - e_i) + \sum_{i\in VM} \sum_{c\in\theta} d_{ic} AV_c EF_c \tag{5.9}$$

The maximum expected net benefit under all possible assignments of $|\theta|$ potential backup operating systems on $|VM|$ virtual machines is obtained by solving the *Optimal Diversity Assignment Problem (O-DAP)* formulated in (5.10) as a binary linear optimization model.

$$
\begin{aligned}
& \max_{d_{ic}:i\in VM,c\in\theta,e_i:i\in VM,f_{ij}:(i,j)\in E} Z = ENB \\
\text{s.t.} \quad & \sum_c d_{ic} \geq e_i \quad \forall i \in VM \\
& f_{ij} \geq d_{ic} + d_{jc} - 1 \quad \forall (i,j) \in E, \forall c \in \theta \\
& d_{ic} \in \{0,1\} \quad \forall i \in VM, \forall c \in \theta \\
& x_i \in \{0,1\} \quad \forall i \in VM \\
& f_{ij} \in \{0,1\} \quad \forall (i,j) \in E
\end{aligned}
\tag{5.10}
$$

Given a network of virtual machines with certain asset values and exploitability factors before implementing any Diversity techniques (before solving the optimization problem), ALE_{before} can be computed by summing $SLE \times ARO$ over all virtual machines in all attack paths. Based on the binary decision variables d_{ic} and e_i, ALE_{after} can be computed after updating SLE^d values according to (5.9).

The dependencies between the d_{ic} and e_i values are taken into account using the first constraint in (5.8) (one linear constraint for each virtual machine) which also supports the natural constraint that each virtual machine gets at most one backup operating system. The second constraint in (5.8) is for obtaining the edges (i,j) for which the same backup operating system is assigned on adjacent nodes. Accordingly, *O-DAP* model has $(|\theta|+1)|VM| + |E|$ binary decision variables and $|VM| + |E|$ constraints.

Generally, the problem of assigning $|\theta|$ potential backup operating systems on $|VM|$ virtual machines has $(|\theta|+1)^{|VM|}$ solutions because each virtual machine can independent of others get either one of the backup operating systems or none. Therefore, the solution to *O-DAP* for instances with a large $|VM|$ and $|\theta|$ cannot be found by exhaustively going through all $(|\theta|+1)^{|VM|}$ possibilities and finding the solution with the maximum desired output. However, our formulation provided in (5.10) is a binary linear programming model and can be used to efficiently find the globally optimal solutions to large instances with thousands of virtual machines in a reasonable time.

5.5.10 Numerical Experiment of Optimization Model

In this section, we discuss a numerical example with seven potential backup operating systems to be implemented as Diversity technique on nine virtual machines in the upper layer of HARM shown in Figure 5.9 and solve it using *Gurobi* solver [57].

Table 5.6 shows a E-Health cloud equipped with various backup OS variants which can be used for Diversity techniques. The table represents the number of patched or mitigated vulnerabilities and the cost of security for each entry as well as the asset value for each VM. It is assumed that more secure backup variants have higher cost of security values, and accordingly have less impact of damage.

Table 5.6: Backup OS variants used for the optimization test case

No.	Backup OS (θ)	Vulnerabilities (V)		CS (\$)	AV (\$)
		$\lvert V\rvert$	EF		
1	HP-UX 11i	4	0.55	55	450
2	Windows (Win 8)	4	0.53	65	490
3	Solaris	3	0.51	80	550
4	Win XP	3	0.49	100	590
5	CentOS	2	0.47	120	620
6	OpenBSD	1	0.45	150	680
7	Win Server 2008	1	0.43	200	690

According to the values in Table 5.6, the seven backup systems have the following values for cost of security, exploitability factor, and asset value:

```
cost_of_security=[55,65,80,100,120,150,200]
exploitability_factor=[0.55,0.53,0.51,0.49,0.47,0.45,0.43]
asset_value=[450,490,550,590,620,680,690].
```

Based on the network shown in Figure 5.9, the values for ALE_{before}, SLE_i, and ARO_i are as follows:

```
ALE_before=160194
SLE=[300.0,300.0,300.0,300.0,283.2,283.2,283.2,283.2,283.2,5900.0]
ARO=[1,1,1,1,1,1,1,1,1,1].
```

The Gurobi [57] model for this instance of the *O-DAP* is provided in the Appendix in which the value of “big M” is considered to be 100000.

While this instance of the problem has $(|\theta| + 1)^{|VM|} = 8^9 = 134217728$ feasible solutions, Gurobi solver obtains the globally optimal solution (associated with the maximum value of the expected net benefit) in 0.02 seconds on an ordinary laptop with 8.00 GBs of RAM and Intel Core i5 6360U CPU @ 2.00 GHz.

The optimal value of the expected net benefit is 117.8 which is achieved by assigning backup operating system 6 on virtual machines 5 and 6 and backup operating system 5 on virtual machine 9 ($d_{5,6} = 1.0, d_{6,6} = 1.0, d_{9,5} = 1.0$). These optimal changes to the upper layer of HARM are represented in Figure 5.11.

5.6 Discussion and Limitations

In Subsection 5.4.2, we showed that deploying Diversity technique can enhance the security of the cloud by increasing the *AC* values (i.e,. the attacker must spend more efforts to penetrate into the cloud and exploit a target). We also observed that Diversity decreases

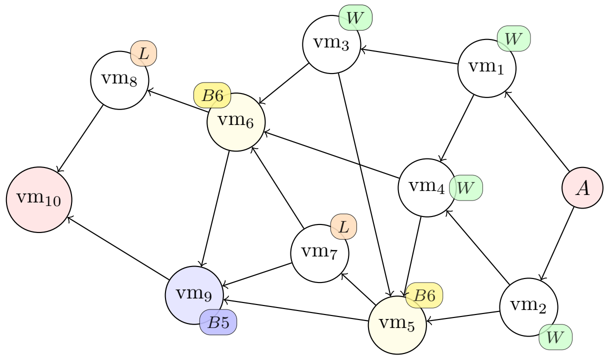


Figure 5.11: Optimal OS Diversity assignment satisfying the coloring requirement on adjacent nodes and maximizing the expected net benefit (note that the backup denoted by $B5$ and $B6$ are CentOS and OpenBSD, respectively.)

the RoA values, which shows that attacker would have less tendency to attack. Furthermore, we witnessed that increasing the number of variants of the OS diversification technique leads to higher AC values and provides lower values for the RoA metrics.

Comparing three strategies for deploying Diversity technique on multiple VMs, we show that the best OS diversification strategy is deploying Diversity technique on VMs grouped by the BVS group. However, Diversity does not have significant effects on *Risk* and *Reliability* values. In Section 5.4.3, we showed that combining all MTD techniques (S+D+R) can improve the security in all aspects with respect to the security metrics used (i.e., the *Risk*, AC, RoA, and *Reliability* values).

We compared D-Only to combining (S+D+R) and observed that the results of deploying S+D+R are better. The values of RoA metric in S+D+R is less than the same metric in D-Only. Moreover, combining S+D+R leads to more promising results on *Reliability*, see Figure 5.7. In fact, combining S+D+R technique will help the cloud providers to keep the security level of the cloud at a desirable level. Figure 5.6 demonstrated the overall results of combining MTD techniques. It compares all security metrics before and after deploying MTD techniques and shows the effectiveness of MTD deployment in one scope. Comparing metrics in Figure 5.6, we observe that deploying S+D+R decreases the RoA and the overall risk of the cloud while the values of AC, and *Reliability* increase. However, one can change the defined parameters to obtain the desired results based on the type of the cloud and the required security levels. For instance, we experimented with setting the attack rate to 0.2, and setting OS diversification variants between $1v$ to $5v$. In addition, we conducted our experiment based on modeling a PHC cloud example to evaluate the security and economic metrics for both Shuffle and Diversity MTD techniques. We leveraged a VM placement strategy for deploying Shuffle. We also utilized two strategies for deploying Diversity in which a single backup OS can be deployed either on one selected VM or a multiple set of selected VMs in the cloud. We observed that deploying Diversity using the same OS (one OS variants) on multiple VMs can enhance RoSI value. We found

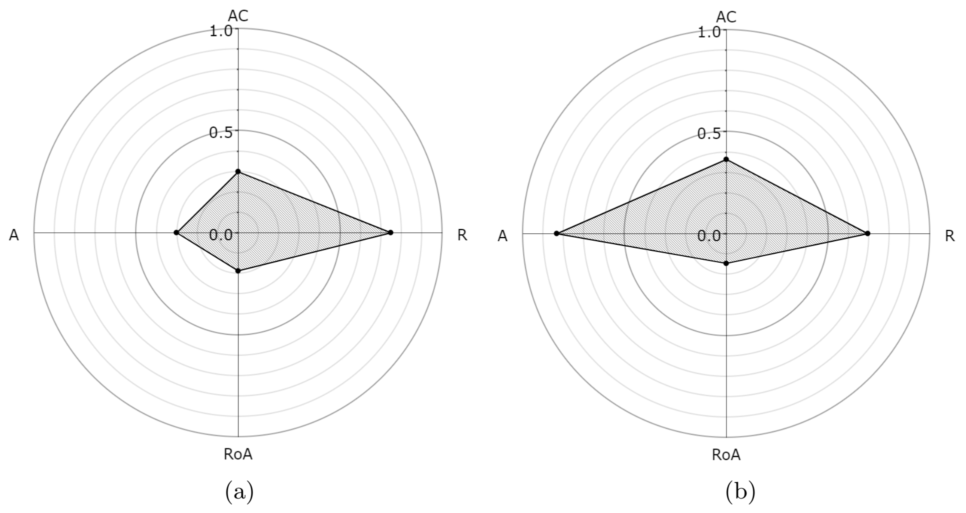


Figure 5.12: Radar charts comparing all *Risk*, *AC*, *RoA*, *Reliability* metrics after and before deploying S+D+R MTD technique. (a) security levels of current cloud, (b) security levels after deploying MTD technique.

that deploying four OS diversification among four VMs has the highest Betweenees values which leads to the highest RoSI values while incurring a reasonable CS values compared to other Diversity strategies.

Moreover, we solved an optimal diversity assignment problem to find the most promising results based on the given network and a set of various backup OS variants using theoretical and mathematical optimization model in Section 5.5.9. We showcased our proposed Diversity techniques by a test case of seven OS variants (Backup OS) and a cloud network with nine VMs based on the parameters shown in Table 5.6. We used Gurobi solver to find the optimal assignment of OS variants on virtual machines through solving a binary linear programming model *(O-DAP)*. According to the globally optimal solution for that instance, among over 134 million possibilities, assigning certain backup OS's on certain VMs yields the maximum expected net benefits 5.5.9.

Limitations: In this chapter, we did not consider the economic metrics for combining all three MTD technique. Since each MTD technique may vary the security and economic metrics in different ways. A multi-objective optimization is needed to solve the problem in a way that three MTD techniques can be combined effectively to satisfy the security and economic levels required by the cloud provider based on the constraints such as the given model and allocated budget.

Moreover, we only include OS-level vulnerabilities on each VM in order to perform the security analysis, while there are many other vulnerabilities existing on the cloud and each VMs such as network, application, service vulnerabilities, and so on. Hence, a more promising model should be proposed to cover the other vulnerabilities in the security analysis phase. We also assumed that the attacker is located outside of the cloud, but it is also important to propose a formal model to be able to capture internal threats like the multi-tenancy and co-residency threats. [72].

The system performance is known as another downside of most MTD techniques. In this chapter, we did not investigate performance analysis. For instance, any Shuffle technique may lead to system functionality degradation i.e. downtime, component failure, *etc.* which we plan to address in our future work.

Lastly, we deployed an optimization model for Diversity technique, while modeling and formulating optimization models for other MTD techniques such as Shuffle and Diversity can be considered. Moreover, multi-objective optimization can be utilized to find an optimal solution based on several objective functions such as RoSI and RoA. We plan to address an integrated optimization model in future work. Moreover, the obtained optimal values were based on a single operation only, while MTD techniques can be adapted and deployed on periodic basis. Deploying defensive MTD technique can be set periodically or erratically based on various factors such as the annual rates of attack (ARO), intrusion detection, and so forth which is out of the scope of this chapter and we aim to address them in our future works.

5.7 Conclusions

Many techniques have been proposed to enhance the security of the cloud. Among them, MTD strategies are the new paradigm having been examined systematically in the past couple of years aiming to mitigate the possible cyber-security threats on the cloud. However, the effectiveness of combining them for enhancing the security of the cloud has not been studied well. In this study, we reviewed the current state-of-the-art MTD techniques applicable in the cloud. Then, we adopted a formal security model to evaluate the effects of combined MTD techniques. In comprehensive experimental analyses, we showed the effectiveness of a combined approach compared to a single MTD technique. The results showed that our proposed approach can be used for evaluating the effectiveness of MTD techniques (individually or combined). Moreover, we evaluated the effectiveness of MTD techniques in terms of economic metrics alongside security metrics for a specific E-health cloud model to analyze the cost of security against security achievements. We proposed an optimization model for Diversity allocation and showed that our binary linear programming formulation handles large instances of the *O-DAP* with millions of feasible solutions in a fraction of a second on an ordinary computer.

5.8 Appendix*

Gurobi model for the instance of the OPTIMAL DIVERSITY ASSIGNMENT problem used as a test case in Subsection 5.5.10 is provided below. Properties of the seven potential backup operating systems were provided in Table 5.6. In this instance, diversity technique was to be implemented on nine virtual machines in the network shown in Figure 5.9.

Maximize
(-269.2
+ 2915.0 d1,1
+ 3051.4 d1,2
+ ...
+ 4250.5 d9,7
+ -3600.0 e1
+ -3000.0 e2
+ ...
+ -4248.0 e9
+ -100000.0 f1,3
+ -100000.0 f1,4
+ ...
+ -100000.0 f7,9)

Subject To
-1.0 d1,1 + -1.0 d3,1 + f1,3 $\geq$ -1.0
-1.0 d1,1 + -1.0 d4,1 + f1,4 $\geq$ -1.0
...
-1.0 d7,7 + -1.0 d9,7 + f7,9 $\geq$ -1.0
d1,1 + d1,2 + d1,3 + d1,4 + d1,5 + d1,6 + d1,7 $\leq$ e1
d2,1 + d2,2 + d2,3 + d2,4 + d2,5 + d2,6 + d2,7 $\leq$ e2
...
d9,1 + d9,2 + d9,3 + d9,4 + d9,5 + d9,6 + d9,7 $\leq$ e9

Binaries
d1,0, d1,1, ... , d9,7, e1, e2, ... , e9, f1,3, f1,4, ... , f7,9

Chapter 6

Usability and Adaptation of MTD Techniques on a Realistic Testbed:

Implementation of an Automated Security Analysis Framework using MTD Techniques on Cloud

Summary

Cloud service providers offer their customers with on-demand and cost-effective services, scalable computing, and network infrastructures. Enterprises migrate their services to the cloud to utilize the benefit of cloud computing such as eliminating the capital expense of their computing need. There are security vulnerabilities and threats in the cloud. Many studies have been proposed to analyze the cloud security using Graphical Security Models (GSMs) and security metrics. In addition, it has been widely researched in finding appropriate defensive strategies for the security of the cloud. Moving Target Defense (MTD) techniques can utilize the cloud elasticity features to change the attack surface and confuse attackers. Most of the previous work incorporating MTDs into the GSMs are theoretical and the performance was evaluated based on the simulation. In this chapter, we realize the previous framework and design, implement and test a cloud security assessment tool in a real cloud platform named UniteCloud. Our security solution can (1) monitor cloud computing in real-time, (2) automate the security modeling and analysis and visualize the GSMs using a Graphical User Interface via a web application, and (3) deploy three MTD techniques including Diversity, Redundancy, and Shuffle on the real cloud infrastructure. We analyze the automation process using the APIs and show the practicality and feasibility of automation of deploying all the three MTD techniques on the UniteCloud.

6.1 Introduction

The growth of the cloud computing as a powerful and affordable context for users has caused many business and commerce migrate to this on-demand, scalable, and cost-effective paradigm. The organizations outsource their network infrastructures, computing needs, software and services into the cloud in order to benefit from the cloud's utilities such as economical benefits (cutting off physical resources and damages). However, many organizations and enterprises find this migration undesirable due to cloud security issues [135, 171].

Many security mechanisms and defensive strategies have been proposed by researchers both theoretically and practically. In order to improve the security of cloud computing, it is important to evaluate the security posture of cloud. Graphical Security Models (GSMs) (such as Attack Graphs (AGs) [41], Attack Trees (ATs) [88], Attack-defense threes (ADTrees) [87], Hierarchical Attack Representation Model (HARM) [61]) are the widely adopted methods to analyze the security of enterprise networks [77, 156]; a GSM can be used to define attack surfaces and summarize the attack scenarios, and compute security metrics. Moreover, GSMs can be used to evaluate the cloud security posture. GSMs can also be used to evaluate the effectiveness of defensive techniques such as Moving Target Defense (MTD). MTD techniques are proactive defensive techniques and the primary idea is mainly changing the attack surface in order to introduce confusions to attackers carrying out cyber attackers. There are a few research in this line. Only a few studies have been proposed for the uses of GSM in evaluating MTD techniques for cloud computing. However, most of the previous research are theoretical and use simulation only [11–13, 66] to show the feasibility of their approaches. To the best of our knowledge, the incorporation of GSMs and MTD techniques together for security analysis and deployment of MTD techniques in the infrastructures of the real clouds has not been proposed. In this chapter, we tackle the aforementioned shortcomings by designing and development of a cloud security assessment framework which can automatically monitor, model, and analyze a private cloud security and deploy the MTD techniques on the cloud infrastructures. In this section, we focus on the practical side rather theoretical appraisal. We demonstrate the practicality of implementation, feasibility of automation, usability of the project using a real cloud platform named UniteCloud [1, 59, 117].

The main contributions of this chapter are summarized as follows:

- *Formalization*: We formalized a cloud-based security model hosting various independent enterprises.
- *Cloud monitoring*: We developed a cloud security framework which can automate the process of cloud vulnerability scanning in order to collect the information of the cloud's components together with the vulnerabilities of each component.

- *Cloud security evaluation:* Cloud security framework can create the HARM based on the collected information for security analysis and MTD evaluation purposes.
- *MTD deployment*: Cloud security framework automated the deployment of three MTD techniques such as Diversity, Redundancy, and Shuffle on the real cloud infrastructure.
- *Automation evaluation*: We investigated on a private cloud platform and uses of OpenStack Application Programming Interfaces (APIs) to analyze the automation process for implementation steps.
- *MTD visualization*: We developed a graphical user interface (GUI) as a web application for interaction between cloud security framework and security experts including both cloud provider view and HARM [66] visualization.
- *MTD measures*: We analyze the MTD deployment based on different API calls time measurements.

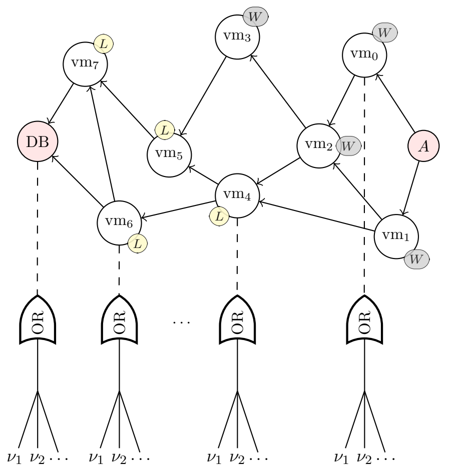
The rest of the chapter is organized as follows. Section 6.2 defines the proposed approach including a brief explanation on preliminaries, concepts, and definitions. Section 6.3 presents the design and implementation of the cloud security assessment framework. Discussion and limitations of this work are given in Section 6.4. Section 6.5 summarizes the related work. Finally, we conclude this chapter in Section 6.6.

6.2 Proposed Approach

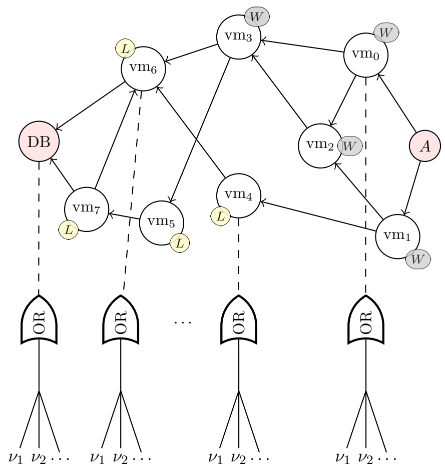
In this work, we implement a cloud security assessment framework which is able to monitor the cloud, analyze, and deploy the three MTD techniques including Shuffle, Diversity, and Redundancy on the real infrastructures of the cloud. The main part of this chapter is the automation of the cloud assessment framework in the real cloud. The uses of APIs in the implementation and automation of the project are nontrivial. Automation process needs a deep understanding of the infrastructures and platforms in which the private cloud uses. On the other hand, the security analysis framework should be able to handle the cloud constraints defined by both cloud provider and security experts such as uses of services, access to controllers, etc.. This work includes four main phases elucidated as follows. (1) Information Collection, (2) Cloud Security Modeling using HARM, (3) Security Analysis Engine, (4) Deployment Phase.

6.2.1 Preliminaries

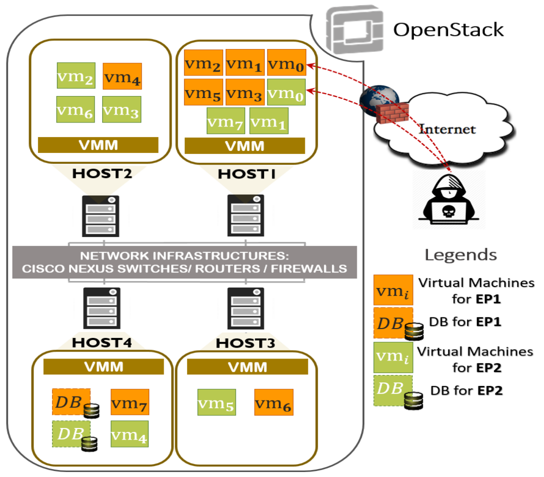
In this section, we describe the related concepts and definitions used throughout this chapter. We first define a running example as the main scenario for the migration of enterprises to the cloud. Later on, we deploy the proposed scenario together with the proposed approach on the UniteCloud.



(a)



(b)



(c)

Figure 6.1: Running Example and cloud model for two enterprises migrated into the cloud named EP1 and EP2. (a) Two-layer HARM of the EP1 in the Cloud, (b) Two-layer HARM of the EP1 in the Cloud, (c) a private cloud example including the various Hosts (servers) and Virtual Machines (VMs) hosting EP1 and EP2.

6.2.1.1 Running Example

Figure 6.1c shows the running example scenario on the migrations of two independent organizations entitled Enterprise-1 (EP1) and Enterprise-2 (EP2) to a private cloud. Those companies decide to cut off the physical equipment and use a private cloud for accommodating their computing needs. Each organization has launched 8 Virtual Machines (VMs) on the cloud together with a Database (DB) creating a virtual network. We assume that the first four VMs use Windows10 and the rest use Linux Ubuntu. Moreover, the VMs vm_0 and vm_1 for both organizations are connected to the Internet. Later on, we deploy the running example shown in Figure 6.1c in the real cloud infrastructures of UniteCloud.

6.2.1.2 System and Attack Models

System constraints are usually defined based on both cloud provider and security experts. For instance, the cloud provider can determine which cloud zones or physical hosts are available for the customers. Moreover, the cloud provider can set the limitations on the physical hosts such as defining the maximum VMs can be located on each host and so forth. The system constraints are defined due to different reasons like workload balance or energy saving, security purposes [58]. However, the security experts of enterprises migrated into the cloud may have their own security policies like defining firewalls rules and Access Control Lists (ACL). We assume that an attacker can launch the attacks from outside of the cloud using exploiting the software vulnerabilities of the VMs connected to the Internet. Then, the attacker can launch a series of other attacks in order to access the DB along the identified attack paths.

6.2.2 Security Model for Cloud

We define our model based on multiple organizations migrating to the cloud. We first define a cloud system as follows.

Definition 14. A cloud system can de defined as a Tuple $\mathcal{C} = (N_{\mathcal{C}}, S_{\mathcal{C}})$, where $N_{\mathcal{C}} = \{VM_{\mathcal{C}}, E_{\mathcal{C}}\}$ is the network of VMs in the cloud connected to each other and $S_{\mathcal{C}} = \{s_1, s_2, ..., s_z\}$ is a set of physical servers in the cloud hosting different VMs. Then, $VM_{\mathcal{C}} = \{vm_1, vm_2, \ldots, vm_n\}$ is a total set of VMs in the cloud, with $|VM| = n$, and $E_{\mathcal{C}} \subseteq \{VM_{\mathcal{C}} \times VM_{\mathcal{C}}\}$ denotes the connectivities of VMs.

Then, as a cloud can accommodate multiple organizations each of which has independent network in the cloud, we define a set of sub-clouds showing various enterprises which have their own network and infrastructures in the cloud.

Definition 15. Let define $SC = \{sc_1, sc_2, ..., sc_e\}$ as a set of sub-clouds where $|SC| = e$ denotes the number of enterprises migrating to the cloud. We define $sc_x = (N_x, S_x)$ as the cloud infrastructures assigned to the x^{th} enterprise, EPx, migrated into the cloud, where $0 \leq x \leq e$, and $S_x \in S_{\mathcal{C}}$, and $N_x = \{VM_x, E_x\}$ shows the network of EPx, such that $VM_x \in VM_{\mathcal{C}}$, and $E_x \subseteq \{VM_x \times VM_x\}$.

In this study, we use HARM [61, 66] for graphical security modeling, analysis and evaluation. Since multiple independent organization can reside in the same cloud, we redefine HARM based on sub-clouds for each independent organization migrated to the cloud as follows.

Definition 16. HARM can be modeled as a 3-tuple $H_{sc} = (U_{sc}, L_{sc}, M_{sc})$ where U_{sc} refers to an AG corresponding to a sub-cloud sc_x, and L_{sc} denotes an AT corresponding to a sub-cloud sc_x, and M_{sc} is a one-to-one mapping link from the AG to the corresponding

AT, $M_{sc} = U_{sc} \rightarrow L_{sc}$ (shown as dashed lines in Figure 6.1). The upper layer of HARM captures the connectivities of VMs, and the lower layer captures the vulnerabilities of each VMs $V_{vm_i} = \{\nu_1, \nu_2, \ldots, \nu_m\}$, such that the vulnerabilities make the leaves of three and the root is a logical gate.

Figure 6.1a and 6.1b represent two layers of HARM for EP1 and EP2 respectively. Figure 6.1c show a private cloud model hosting EP1 and EP2.

6.2.3 Security Analysis

Constructing the security model, we can leverage HARM to compute the security metrics and quantify the cloud security. As the focus of this chapter is implementation of cloud security framework using MTD, we only leverage three security metrics, which are Cloud Risk (R), Attack Cost (AC), and Return on Attack (RoA), to evaluate the cloud security posture before and after deploying MTD techniques to find out the most effective defensive strategy. The uses of those metrics for evaluation of cloud are theoretically investigated through simulation in Chapters 3−5. Other security metrics such as Attack Success Probability (ASP), Mean Time to Attack (MTTA), *etc.* [156,158] can also be computed. HARM uses the vulnerabilities information which can be obtained from National Vulnerability Database (NVD) [106] and generate the lower layer using the vulnerability values such as Impact, Exploitability, Base Score. The brief description, formalism and calculation for the security metrics are given in Table 4.2 presented in Chapter 4.

6.3 Design and Implementation

This section provides the design and development of a security analysis framework for cloud computing. We investigate the feasibility and practical requirements such as Software tools, packages, programming interfaces, libraries in order to implement and automate the security analysis tool and MTD techniques in the real-world cloud deployment. We develop a framework which can perform security modeling, evaluation, MTD deployment for enterprises migrated into the private clouds. The cloud security framework is able to automate information collection: cloud scanning, vulnerability scanning, HARM creation, security evaluation, and MTD deployment on a real cloud infrastructure. To implement the framework we utilize a private cloud named UniteCloud and develop our framework on UniteCloud as a case study. However, we believe that our developed framework can be adopted for other private clouds as well.

6.3.1 Case Study: UniteCloud Analysis

The UniteCloud uses the OpenStack cloud platform. For setting up the project, we can either use OpenStack horizon dashboard or utilize OpenStack APIs. The setup process

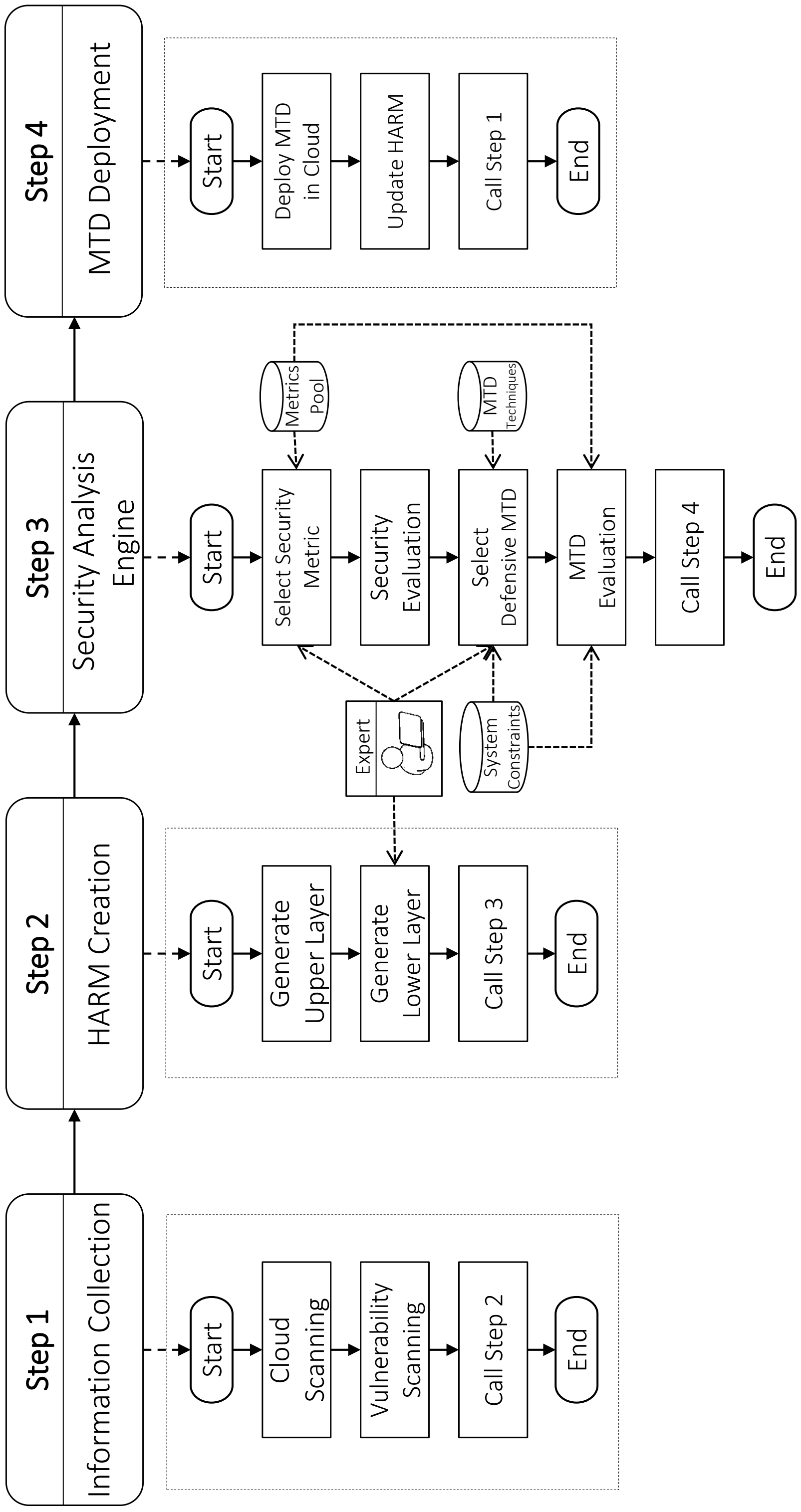


Figure 6.2: Security modeling, analysis, and deployment Phases

includes the creation of VMs with different flavors and OS, assigning internal and floating IP addresses, defining firewall rules and ACL, *etc.* However, we first create the cloud example VMs shown in Figure 6.1c into the UniteCloud infrastructures using the horizon dashboard. Further, we utilize the OpenStack APIs for automation. Table 6.1 shows the created VMs including the related information on each host. The cloud consisting of 16 physical Hosts (Compute Hosts) is distributed over three availability zones: IBMZone, HPZone, and Nova. However, for our implementation, we used four hosts each of which includes different VMs with different flavors. We assign two flavors for the VMs: m1.medium and m1.generic. The specification of the former VM is 2 VCPUs, 4 GB RAM, and 80 GB Disk, and that of the latter is 1 VCPUs, 1 GB RAM, and 20 GB Disk, respectively. We assign two floating IP addresses for both VMs vm_0 and vm_1 of two enterprises which are the entry points of the cloud. Moreover, VMs vm_6 and vm_7 of both enterprises EP1 and EP2 are connected to their own DB.

6.3.2 Cloud Security Framework

The security framework consists of a backend engine and user interface (UI). The backend engine is responsible for information collection, security modeling, analysis, and deployment phases which are demonstrated in Figure 6.2. The UI is used for interactions between

Table 6.1: Configuration and setup for VMs and hosts in the cloud. Note: floating IPs are denoted as asterisked

Host & Zone Name	VM Name	OS	IP Addresses	Flavor Size
h_0 IBMZone	vm_1-EP2	Win10	172.16.7.33 192.168.1.100*	m1.medium
	vm_2-EP2	Win10	172.16.7.32	m1.medium
	vm_5-EP2	Ubuntu14.04	172.16.7.39	m1.generic
	vm_3-EP2	Win10	172.16.7.36	m1.medium
	vm_7-EP1	Ubuntu14.04	172.16.19.16	m1.generic
	vm_0-EP1	Win10	172.16.19.14 192.168.1.239*	m1.medium
	vm_0-EP2	Win10	172.16.7.35 192.168.1.149*	m1.medium
	vm_1-EP1	Win10	172.16.19.12 192.168.1.63*	m1.medium
h_1 IBMZone	vm_4-EP2	Ubuntu14.04	172.16.7.37	m1.generic
	vm_6-EP1	Ubuntu14.04	172.16.19.18	m1.generic
	vm_2-EP1	Win10	172.16.19.15	m1.medium
	vm_3-EP1	Win10	172.16.19.11	m1.medium
h_2 HPZone	vm_6-EP2	Ubuntu14.04	172.16.7.40	m1.generic
	vm_5-EP1	Ubuntu14.04	172.16.19.19	m1.generic
h_3 HPZone	vm_7-EP2	Ubuntu14.04	172.16.7.38	m1.generic
	vm_4-EP1	Ubuntu14.04	172.16.19.17	m1.generic

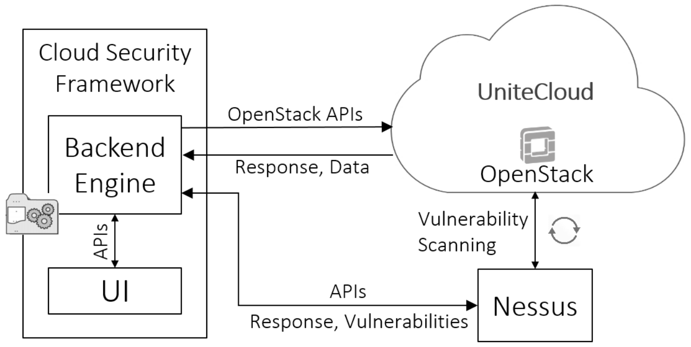


Figure 6.3: Security framework and communication overview.

security experts of enterprises and the backend engine for configuration and visualization purposes. The generated graphical security model can be visualized in the UI. Figure 6.3 shows an overview of the security framework prototype and related tools and communication. The cloud security framework utilizes the following programming languages, tools, and concepts: .NET Core, JSON, JavaScript, jQuery Ajax, Python, Nessus [23], Data-Driven Documents JavaScript (D3.js). In this section, we show the implementation of the cloud security framework. Security modeling is the first phase of the cloud security framework. Security modeling consists of two phases: (1) information collection, (2) HARM creation shown as steps 1 and 2 in Figure 6.2. First, the cloud infrastructure should be scanned in order to obtain Hosts, VMs, and reachability information. Then, the vulnerabilities existing on each VM should be obtained using the vulnerability scanning tools [23]. Information gathering is a crucial phase for the security modeling. Next, the HARM can be constructed using the obtained information. The reachability information can be used to generate the upper layer of HARM where an AG is used to show all the possible attack scenarios given system and attack model. Moreover, the vulnerability information can be used to construct the lower layer of HARM which uses the ATs. However, generating ATs from vulnerabilities needs a clear understanding of the vulnerabilities and the way in which they can be exploited. For instance, an attacker can exploit

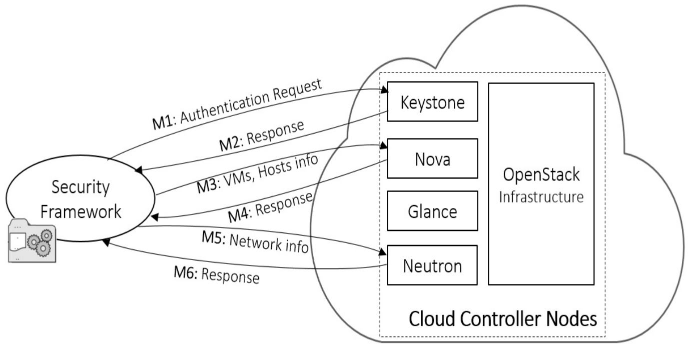


Figure 6.4: OpenStack API Calls for information collection phase.

only one vulnerability to penetrate into a VM, or the attacker may need to exploit a set of vulnerabilities to penetrate into a VM. In the former case a logical OR-gate can be used and for the latter, a combination of logical AND/OR-gates can be used [128]. The uses of logical AND/OR-Gates and computation approach are presented in [128]. However, a security expert can help to define the vulnerabilities relations. Entry points of the cloud are actually the VMs connected to the Internet. Those VMs are the entry points of the attacks as well. The target could be any VM which includes important information or runs crucial services. We assume that the DB is the attackers' target. Both entry points and target are captured in the upper layer of the HARM. The upper layer can be generated using reachability information obtained from analyzing the firewall rules.

6.3.2.1 Information Collection Automation

As stated earlier, cloud infrastructure information including VMs and hosts, and the reachability of VMs are required for constructing the upper layer of HARM, and vulnerabilities associated to each VM are required to create the lower layer of HARM. Cloud security framework needs to automatically fetch two information: (1) cloud information such as the number of VMs, the number of physical hosts, the host of each VM, the reachability between the VMs and (2) vulnerability information existing on each VM. We use .NET Core as the backend engine programming language and call APIs in order to access both OpenStack and Nessus automatically and fetch information. Accessing to the UniteCloud OpenStack consists of two parts: OpenStack authentication and fetching information. OpenStack uses Keystone feature for user authentication. Moreover, it uses nove-computes, neutron-networks, Glance-images features for different purposes such as accessing to compute nodes (VMs, Hosts, Zones, *etc*). In order to access to the OpenStack and retrieve the information, we first need to access keystone using APIs for authentication. The username, password, and domain name are sent to the Keystone controller using a JSON API call for authentication. Once the user is authenticated using the Keystone authentication method, OpenStack sends a response including the authentication token (X-Subject-Token), other OpenStack Controllers' address including nova, neutron, glance, cinder, *etc.* which can be used for further API calls. The received message should be first parsed to receive the authentication token together with the nova controller address. Then, the backend engine sends another API call using the authentication token and the nova controller to gather the list of VMs and Hosts. The received message contains unnecessary/irrelevant information including VM status, availability zone, created and updated time, *etc.*, the message should be parsed to fetch only the required information. Similarly, another API including the authentication token and neutron controller should be called to get network-related information. The received information should be again parsed to obtain VMs' IP addresses and the reachability of VMs. Figure 6.4 demonstrates the API calls and related responses between the cloud security framework and OpenStack in order to gain the information. Beside the VMs and reachability information, we need

Algorithm 4: Information collection procedure

```
                                  /* Input info. needed for OpenStack (Ops) */
   Data: Ops-user-credential, Keystone-Controller-Url
                                     /* Input Info. needed for Nessus (NS) */
   Data: NS-user-credential, Nessus-Session-Url
              /* Result: Dictionaries of VMs and Reachability, VMs and Vulnerabilities */
   Result: VMs_Links_Dic, VMs_Vuls_Dic
 1 begin
                                    /* Cloud Scanning: fetch Host & VM info. */
 2     Credential-Data←JSonConvert(Ops-user-credential)
 3     JResult←API_Call(Credential-Data, Keystone-Controller-Url)
 4     Auth_Token←Parse(JResult, Authentication)
 5     Controllers_List←Parse(JResult, Nova and Neutron Controllers)
 6     Host-VM-Info← API_Call(Auth_Token, Nova-Controller)
 7     Network-Info← API_Call(Auth_Token, Neutron-Controller)
                                       /* Parsing and saving the fetched data */
 8     Hosts_List←Parse(Host-VM-Info, Hosts)
 9     VMs_List←Parse(Host-VM-Info, VMs)
10     Reachability_List←Parse(Network-Info, Reachability)
11     VMs_Links_Dic = Create Dictionary[VM, VM]
                                  /* Nessus Scanning: fetch vulnerabilities */
12     Credential-Data←JSonConvert(NS-user-credential)
13     JResult←API_Call(Credential-Data, Nessus-Session-Url)
14     Auth_Token←Parse(JResult, token)
15     JResult←API_Call(Auth_Token, Nessus-vulnerabilities-Url)
16     VMs_Vuls_Dic = Create Dictionary[VM, Vulnerabilities-List]
                                    /* Get & save vulnerabilities on each VM */
17     foreach vm ∈ VMs_List do
18         Vuls-Info←Parse(JResult, vm)
19         Add vm and Vuls-Info into VMs_Vuls_Dic
20     end
                                    /* Return reachability of VMs           */
       Output: VMs_Links_Dic
                                    /* Return vulnerabilities on each VM    */
       Output: VMs_Vuls_Dic
21 end
```

vulnerabilities information for each VM on the cloud. We use Nessus [23] to scan the cloud and obtain vulnerabilities. Then, cloud security framework uses a backend engine to access to Nessus and retrieve the vulnerabilities' information. The first API called is used for authentication. Having obtained the response message, the backend engine sends other API calls using the authentication token in order to get the vulnerability information. the extracted information contains useful information related to Vulnerability, possible threats, Base Score [106], severity, *etc.*, CVE identifier (CVE-ID). However, cloud security framework only need CVE-ID for selected vulnerabilities so that it can obtain the other information such as vulnerability impact and exploitability through National Vulnerability Database (NVD) [106]. The pseudocode for the overall information collection is shown in Algorithm 4. Note that cloud scanning using Nessus is a time-consuming process and cannot be done frequently. Instead, it can be run once a while to keep the vulnerabilities updated, or run once a change catches on the VMs such as adding new VM, or changing OS, *etc.*

6.3.2.2 HARM Creation

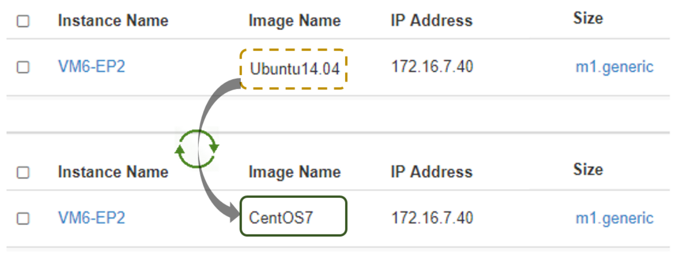
The upper layer of HARM can be generated using the VMs and reachability information obtained from the previous step. This information is saved as a key and value dictionary representing the VMs' links as a graph. Thus, the backend engine can generate the AG based on the dictionary. The second part of the information obtained from Nessus scanning is a dictionary of VMs and related vulnerabilities on each VM which can be used to generate the lower layer of HARM. The lower layer of HARM uses the ATs. The backend Engine uses Python programming language to generate HARM. However, other software and tools can also be used like Gephi which is a network analysis and visualization software package. Moreover, we use Python [77] as the security analysis engine to compute security metrics and evaluate MTD techniques.

6.3.2.3 Security Analysis Engine

Security analysis engine has two main phases: general security evaluation and MTD evaluation. HARM can be adopted to compute the security metrics in the pool. Security analysis engine is implemented on the backend engine using Python. It consists of security evaluation and MTD evaluation subroutine. Security analysis engine uses the generated HARM and the security metrics. In fact, security experts can choose or prioritize various security metrics and add them to the metric pools based on the security requirements such as System Risk, Attack Cost, MTTA, Attack Success Probability, and so forth. Once the security metrics are selected, security analysis engine uses HARM for security evaluation and computing the selected security metrics. Security analysis framework uses MTD techniques as the main defensive strategies for security the organizations on the cloud. However, deploying MTD techniques could be limited based on system constraints. For instance, VM-LM (Shuffle technique) might be restricted from one host to another one due to lack of space on the target host, or OS Diversification (a Diversity technique) could be limited to only a few OS instances due to the cost of the license for the cloud provider. Thus, the MTD techniques should be chosen based on the defined system constraints.

6.3.2.4 MTD Deployment Implementation

The final phase of the cloud security framework is the deployment of selected MTD techniques on the cloud infrastructure. It uses .NET Core and OpenStack APIs to deploy MTD techniques, it utilizes glance for creating and retrieving OS instance images, nova, and network controllers for accessing and manipulating VMs and Network purposes.



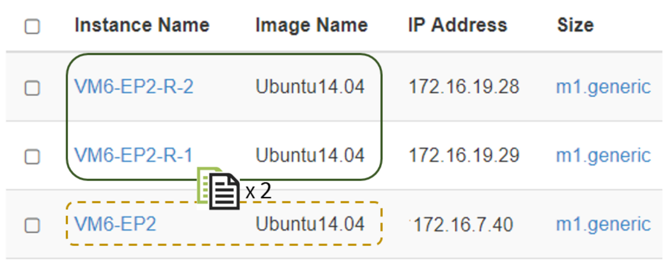
Instance Name	Image Name	IP Address	Size
VM6-EP2	Ubuntu14.04	172.16.7.40	m1.generic

Instance Name	Image Name	IP Address	Size
VM6-EP2	CentOS7	172.16.7.40	m1.generic

Figure 6.5: OS Diversification: Ubuntu14.04 replaces with CentOS7 for vm_6-EP2

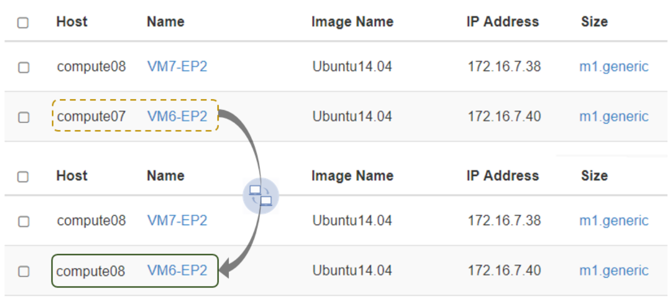
Diversity. Security analysis framework uses OS diversification technique for deploying Diversity. In order to deploy Diversity technique, backend engine uses nova to access the desired VM and update the VM instances with another OS image. Similar to the information collection phase, the user credential information should be sent to the Keystone controller using JSON API call for authentication. Backend engine omits this phase as the authentication token is already received in information collection phase; moreover, both nova and glance controllers are fetched from the response message. Before calling API to change the VM instance, we need to fetch the ImageRef by sending an API to glance. Once the response received, the ImageRef associated to the desired VM image can be obtained. Finally, an API should be called to pass the authentication token, VM ID, ImageRef to the nova in order to rebuild the VM with another OS variant. Figure 6.5 shows the results of calling APIs for replacing Ubuntu14.04 with CentOS7 for vm_6-EP2 on the cloud. Note that Diversity preserves the VM's physical host

Redundancy. Based on the Redundancy definition, different replicas of a VM should be created so that each replica has the same feature as the main VM. For instance, the replicated VMs should have the same OS, Flavor, inbound and outbound links from/to other VMs The only difference is the newly assigned IP addresses. Backend engine is responsible for deploying redundancy. However, the number of replicas for deploying redundancy is chosen by either MTD evaluation part or expert entry using UI. There is no feature on OpenStack to create replication for each VM. Thus, deploying redundancy on OpenStack needs creation r new VMs based on the similar existing instance or copied



Instance Name	Image Name	IP Address	Size
VM6-EP2-R-2	Ubuntu14.04	172.16.19.28	m1.generic
VM6-EP2-R-1	Ubuntu14.04	172.16.19.29	m1.generic
VM6-EP2	Ubuntu14.04	172.16.7.40	m1.generic

Figure 6.6: OS Replication: Create 2 replicas for vm_6-EP2



	Host	Name	Image Name	IP Address	Size
▢	compute08	VM7-EP2	Ubuntu14.04	172.16.7.38	m1.generic
▢	compute07	VM6-EP2	Ubuntu14.04	172.16.7.40	m1.generic

	Host	Name	Image Name	IP Address	Size
▢	compute08	VM7-EP2	Ubuntu14.04	172.16.7.38	m1.generic
▢	compute08	VM6-EP2	Ubuntu14.04	172.16.7.40	m1.generic

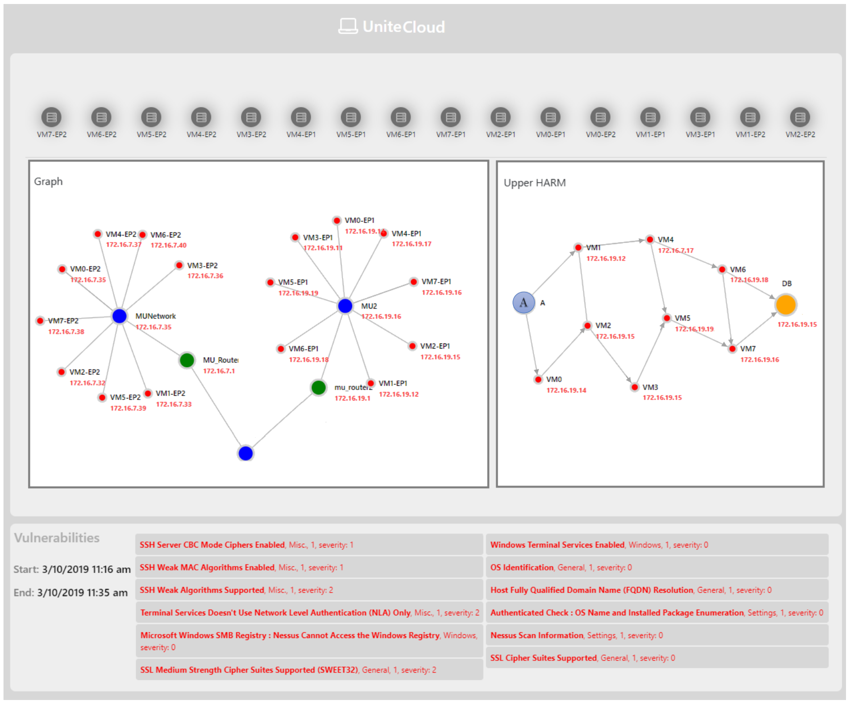
Figure 6.7: VM-LM: Migration of vm_6-EP2 from Compute07 node to Compute08 node

snapshot. Backend engine can use the same authentication token already obtained from the information collection phase and use nova controller. Thus, the backend engine sends an API to nova controller including the authentication token, ImageRef, FlavorRef, NetworkID together with a max_count which is the number of required replicas (r). Figure 6.6 demonstrates the results of calling APIs for the creation of two new replicas of vm_6-EP2 with the same OS, links, hosts, flavors, but different IP addresses.

Shuffle. In this framework, VM-LM is used as the Shuffle technique. VM-LM can be deployed on the OpenStack using nova controller. Similar to other MTD techniques, the backend engine omits the authentication API call because the authentication token and nove controllers have already been fetched in the information collection section. The target host can be selected either by MTD evaluation results or security experts. In order to deploy VM-LM, an API including authentication token together with the VM ID and Target Host ID is called. Figure 6.7 demonstrates the results of calling APIs for migration of vm_6-EP2 from Compute07 to Compute08.

6.3.3 User Interface (UI) Implementation

Cloud security framework uses a UI in order to interact between the security experts of enterprises and backend engine. Security experts can add update the security metrics pool, choose MTD techniques, analyze and monitor the cloud security using visualization panel. UI is implemented as a web application using JavaScript, JSON, jQuery Ajax, and D3.js interacting with the backend engine. UI web application includes two different perspectives for visualization. Cloud provider and security model previews. Cloud provider preview illustrates the internal connection of the VMs, routers, subnets, and etc. in the cloud, and security model preview visualizes the generated upper layer of HARM which captures the reachability of VMs based on the firewall rules and possible attack scenarios. UI also shows the vulnerabilities captured for each VM . UI uses internal APIs to communicate with backend engine and update and gain information. Figure 6.8 demonstrates the UI panel showing two different previous based on the UniteCloud network and HARM view.



Vulnerabilities

Start: 3/10/2019 11:16 am

End: 3/10/2019 11:35 am

SSH Server CBC Mode Ciphers Enabled, Misc., 1, severity: 1

SSH Weak MAC Algorithms Enabled, Misc., 1, severity: 1

SSH Weak Algorithms Supported, Misc., 1, severity: 2

Terminal Services Doesn't Use Network Level Authentication (NLA) Only, Misc., 1, severity: 2

Microsoft Windows SMB Registry : Nessus Cannot Access the Windows Registry, Windows, severity: 0

SSL Medium Strength Cipher Suites Supported (SWEET32), General, 1, severity: 2

Windows Terminal Services Enabled, Windows, 1, severity: 0

OS Identification, General, 1, severity: 0

Host Fully Qualified Domain Name (FQDN) Resolution, General, 1, severity: 0

Authenticated Check : OS Name and Installed Package Enumeration, Settings, 1, severity: 0

Nessus Scan Information, Settings, 1, severity: 0

SSL Cipher Suites Supported, General, 1, severity: 0

Figure 6.8: Cloud security framework UI panel: UniteCloud Graph view and HARM visualization

6.4 Results and Discussion

Backend engine is the base of the cloud security framework which use the OpenStack APIs to create security, perform security analysis, and deploy MTD techniques. The backend engine is responsible for automating the information collection APIs and MTD deployment APIs. The feasibility and practicability of implementing the backend using OpenStack API calls is important. We evaluated the usability of the backend engine by considering the API calls passing through the backend and two other parties: Nessus vulnerability scanning tool, and OpenStack controllers. The details of API calls like the type of APIs and elapsed times are elucidated in this section.

Cloud security framework uses two types of APIs which can be categorized as *informative calls (ICs)* and *operational calls (OCs)*. The first group can be only used to get the information like getting authentication tokens, list of hosts, VMs, *etc.*, these APIs will not make any changes on the cloud. Unlike the first group, *operational calls* can perform an operation and make the changes on the cloud such as migrating a VM from one host to another one, or changing the VM's instance, *etc.* We measure the informative calls with the response time (T_R). Particularly, T_R is the total time needed for sending a request to the cloud and receiving the required information, as shown in Figure 6.4. For instance, the T_R of a keystone authentication call is the time elapsed between calling API and receiving the response from keystone showing accepted status 202 together with

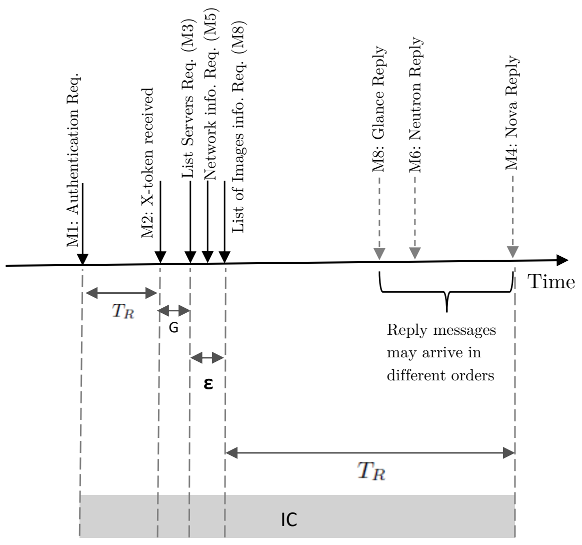


Figure 6.9: Cloud scanning API calls in Information Collection phase

Table 6.2: Benchmark analysis for MTD API calls

API Calls		VM Status	Request Numbers (time in Seconds)										Ave.	Std.
MTD	Type (T)		1	2	3	4	5	6	7	8	9	10		
S	T_γ	Up	0.65	0.77	0.42	0.43	0.42	0.55	0.45	0.45	0.71	0.43	0.53	0.13
	T_o	Down	10.00	11.00	13.00	12.00	9.00	18.00	11.00	13.00	17.00	11.00	12.50	2.77
	T_C	N/A	10.65	11.77	13.42	12.43	9.42	18.55	11.45	13.45	17.71	11.43	13.03	2.80
D	T_γ	Up	0.56	0.72	0.66	0.44	0.42	0.42	0.70	0.81	0.44	0.68	0.58	0.14
	T_o	Down	18.00	17.00	18.00	20.00	17.00	19.00	16.00	19.00	18.00	18.00	18.00	1.10
	T_C	N/A	18.56	17.72	18.66	20.44	17.42	19.42	16.70	19.81	18.44	18.68	18.58	1.06
R (3-r)	T_γ	Up	0.73	0.73	0.76	0.74	1.08	0.76	0.82	0.75	0.91	1.06	0.83	0.13
	T_o	Up	10.00	10.00	11.00	11.00	12.00	12.00	11.00	12.00	11.00	11.00	11.10	0.70
	T_C	Up	10.73	10.73	11.76	11.74	13.08	12.76	11.82	12.75	11.91	12.06	11.93	0.75

the required information in the body of message. However, *operational calls* consist of: (1) Reaction Time (T_γ) which is the time between calling API and the start time of an operation. Note that the response for an API call may include some acknowledge such as denied, abort, unauthorized, *etc.* which means the operational call was unsuccessful. In this case T_γ includes the response time. (2) Operational Time (T_o) which means the difference between the start of an operation using API calls and the time in which the task is fully done. (3) Completion Time (T_C) which is the total time for completion of an operational call: T_γ+T_o; for instance, the total time between sending a request for VM-LM process and the end of the process. Table 6.3 shows the details of API calls including the required fields, received messages together with the average elapsed time for each call and response. The total average T_R for cloud scanning including *informative calls* only is around 3121 milliseconds (*ms*). Nessus scanning APIs also includes as *informative calls* for authentication and vulnerability information fetches. The total measured T_R for Nessus scanning APIs is about 4509 *ms.* Note that, cloud scanning using Nessus servers

Table 6.3: API JSON calls and related information including the T_R and T_C. Note: the asterisked times are T_C

API Calls	Type	Content	Time (ms)
		Cloud Scanning APIs	
M1	Request	[User, Password, Domain], [Keystone Controller]	356
M2	Response	[Authentication Token, Controllers' URL]	
M3	Request	[Authentication Token], [Nova Controller]	1997
M4	Response	[List of Servers, Hosts, Zones, *etc.*]	
M5	Request	[Authentication Token], [Neutron Controller]	209
M6	Response	[Networks, Routers, Ports, etc]	
		Nessus Scanning APIs	
M1	Request	[User, Password], [Nessus Session URL]	471
M2	Response	[Token]	
M3	Request	[Token], [Nessus Session URL]	4038
M4	Response	[CVE-IDs, etc.]	
		Diversity APIs: OS Diversification	
M1	Request	[Authentication Token], [glance controller URL]	559
M2	Response	[List of images (OS instances)]	
M3	Request	[Authentication Token], [VM ID, ImageRef, Nova]	18081*
M4	Response	[Status: accepted or abort]	
		Redundancy APIs: VM Replicas	
M1	Request	[Token], [Image&FlavorRef, max_count, Nova]	12091*
M2	Response	[Status: accepted or abort]	
		Shuffle APIs: VM-LM	
M1	Request	[Auth. Token], [VM ID, Host ID, Nova controller]	7216*
M2	Response	[Migration Status]	

is a separate process and is not included in Nessus scanning APIs. As the T_C includes both T_γ and T_o we only tabulated T_C for deploying MTD technique (as asterisked in Table 6.3).

MTD Deployment Measures. We developed the experiments by performing a Benchmark analysis for deploying three Shuffle (S), Diversity (D), and Redundancy (R) MTD techniques on the OpenStack. We evaluated the operational API calls by measuring the T_γ, T_o, and T_C obtained based on a sequence of 10 API requests. We sent these request to the cloud for deploying MTD techniques on the VM-6-EP2 which uses Ubuntu14.04 and m1.generic flavor size. For analyzing Shuffle, we send operational API calls to the cloud for randomly migrating vm_6-EP2 to other Hosts and measured the operational times for each request. For Diversity, we repeated the experiments by changing

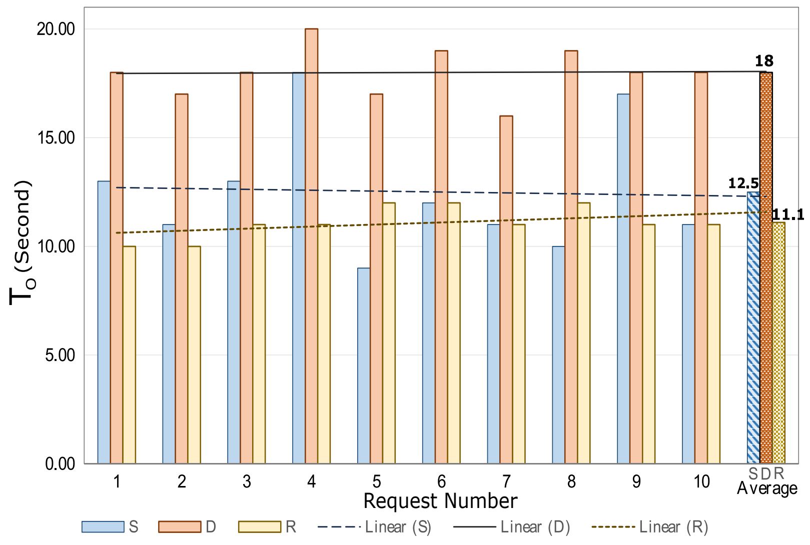


Figure 6.10: Comparing the operational time (T_o) for deployment of MTD in real cloud for 10 API requests denoted as (1-10)

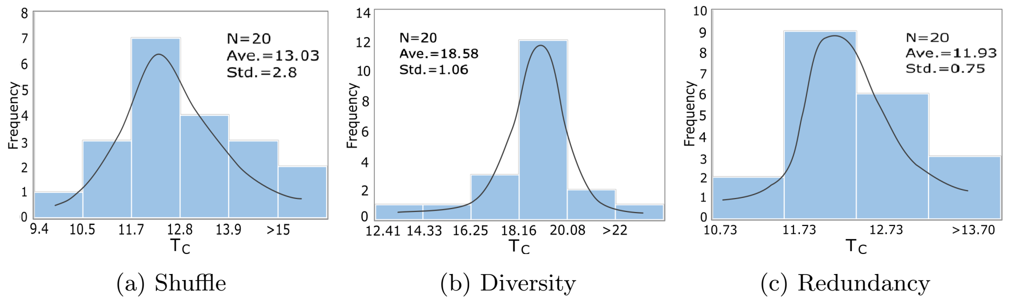


Figure 6.11: The histograms showing the T_C distributions for MTD deployments (Times in Seconds)

the vm_6-EP2 OS to CentOS and vice versa and measured the time. Finally, we analyze Redundancy by creating three replicas (3-R) for vm_6-EP2 named as vm_6-EP2-R-1, vm_6-EP2-R-2, and vm_6-EP2-R-3. We tabulated the measurements of operational times for all MTD techniques on each request together with the average, and standard deviation values in Table 6.2. The results show that the average T_C for S, D, and R (3-R) are 13.03, 18.58, and 11.93 seconds, respectively. Moreover, the results show that the VM is active (Up) during T_γ, while the VM is not accessible (Down) during T_o for both S and D. The VM status is N/A during T_C if there T_γ be in Up and T_o be in Down states. However, the VM status for R is always UP as there are always at least one replication of a VM which can work without the interruption or downtime. Figure 6.10 visualizes and compares the T_o measured based on each request together with the average values. It is obvious that 3 VMs (3-R) can be created in about the average of 11.1 Seconds which is faster than VM-LM (used for deploying Shuffle) with the average migration time 12.5

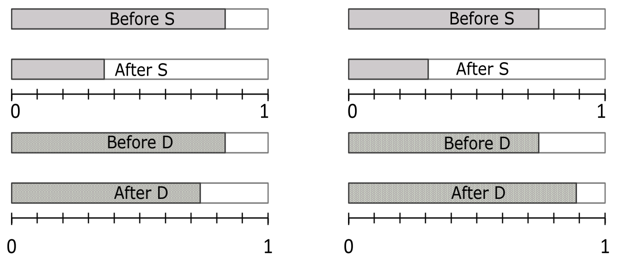
Seconds. Moreover, the highest average T_o value is for deploying Diversity which takes about 18 Seconds to change the current OS with another instance.

We extended our analysis by conducting experiments for a sequence of 20 API requests to measure the operational time for MTD techniques with more accuracy. We divided the T_C into different intervals and counted the number of occurrence for each group. Figure 6.11 demonstrates the histograms for MTD techniques based on the measured times for 20 requests (N=20). We observe that most of the Shuffle technique requests can be completed between 11.7 and 12.8 seconds. Moreover, Diversity can be fully deployed between 18.16 and 20.08 seconds in most of the cases. Finally, Redundancy API requests for creation of three replicas (3-R) can be fully served between 11.73 and 12.73 seconds in most of the cases.

Security Metrics Evaluation. We also evaluated the effectiveness of the MTD techniques in terms of system security. We adopted three security metrics Risk, AC, and RoA into the metrics pool and evaluate each MTD technique. Those metrics are useful for evaluation of Shuffle and Diversity and have already been investigated for evaluation of MTD techniques on cloud through simulation [12]. However, more security metrics can be similarly used to evaluate other security aspects of the cloud [156]. Table 6.4 shows the security metrics resulting from deploying MTD techniques on each VM on the cloud for both EP1 and EP2. Those results can be used by MTD Evaluation phase in the Security Analysis Engine (as shown in Figure 6.3) to find and deploy the most effective deployment. Comparing the results for deploying MTD techniques for EP1, we can observe that deploying Shuffle on vm_7 can lead to better result in terms of Risk and RoA metrics which yield 25.4 and 14.2, respectively. Similarly, deploying Diversity on vm_7 yields 45.5, 89, and 24.4 for Risk, AC, and RoA, respectively. Likewise, deploying Shuffle on vm_7 for EP2 cause lower Risk and RoA values and deploying Diversity on vm_6 provides the better results in terms of Risk and RoA which yields 62.9 and 34.6, respectively. However, the

Table 6.4: The results of three security metrics: Risk, AC, and RoA on the cloud resulting from deploying MTD techniques on EP1 and EP2

VM	Shuffle (EP1)			Diversity (EP1)			Shuffle (EP2)			Diversity (EP2)		
	Risk	AC	RoA	Risk	AC	RoA	Risk	AC	RoA	Risk	AC	RoA
vm_0	34.2	50.5	18.9	48.7	79.2	26.6	47.9	70.9	26.9	65.0	117.9	35.2
vm_1	31.1	45.7	16.9	47.3	87.9	25.1	53.5	79.2	30.3	65.5	115.1	35.8
vm_2	32.1	47.3	17.6	47.8	85.0	25.6	44.7	66.1	24.9	65.0	117.9	35.2
vm_3	45.4	67	25.1	47.8	85.0	25.6	53.3	78.8	29.7	63.7	126.6	33.6
vm_4	34.5	51.2	19.5	47.6	81.2	26.1	56.6	84	32.3	66.4	105.7	37.3
vm_5	30.7	45.5	17.5	45.5	89.0	24.4	63.9	94.8	36.4	65.7	108.3	36.8
vm_6	31.9	47.4	18.2	46.9	83.8	25.5	49.9	74.1	28.3	62.2	121.3	34.0
vm_7	25.4	37.5	14.2	45.5	89.0	24.4	41.4	61.4	23.6	62.9	118.7	34.6



(a) Risk (b) AC (c) RoA

Figure 6.12: Comparing normalized security merics before and after deploying MTD technique on cloud (EP_1)

best results for AC is deploying Diversity on vm_6 yields 121.3. The overall changes on the security metrics obtained from the best MTD deployment scenario for EP1 are shown in the Figure 6.12. The results shows that deploying Shuffle yields better results that Diversity in terms of Risk and RoA metrics. Deploying Diversity yields a gentle decrements for Risk and RoA in the best case while those metrics are almost halved after deploying best Shuffle scenario. However, Diversity yields better results for AC and increases AC.

Limitations and Extensions. The update phase has not been implemented in cloud security framework. This includes running of Nessus scanning and recreation of HARM based on any changes captured in the cloud, such as updating VMs or vulnerabilities. We will further consider the update phase in our future work. We deployed and measured the Redundancy technique on the real cloud. The main aim of Redundancy technique is to enhance the service availability in the cloud. Redundancy can be measured with the concepts of system dependability (e.g. reliability and availability) which is out of scope of this chapter. We will further consider dependability metrics for evaluating Redundancy technique on real cloud in our future work. Moreover, This Chapter presents the implementation and real measurements on a private cloud using OpenStack Platform. However, implementation, adaptation, and deployment of MTD techniques on other types of clouds such as private and/or public clouds (e.g., Microsoft Azure, Amazon AWS, *etc.*) also need to be investigated to show the robustness of the proposed method.

6.5 Related Work

The theoretical investigation and evaluation of the security modeling and analysis adopting based on the MTD techniques for cloud computing have been proposed in the work [13, 66]. However, most of the proposed frameworks have focused on the implementations of GSMs on the networks [48, 77, 90]. The security modeling and analysis tools on the literature can be categorized based on the context of implementation testbed such as cloud computing [37], networks and enterprises [92], or based on GSMs [61], ATs [48], AGs [73, 90], *etc.*, the automation approaches and levels [109], or based on the effectiveness of solution like response time and the probability of success [109]. Authors in [92] proposed and implemented a fast network security assessment prototype based on the real scenario. Moreover, the work [37] developed a framework named NICE in the virtual network systems which is able to detect possible attacks against the cloud infrastructure. To the best of our knowledge, there is no prior work developing the MTD techniques incorporated with the automated GSMs in a cloud environment. In this chapter, we developed an automated cloud security framework able to monitor and detect a private cloud and deploy MTD techniques on the infrastructures of the cloud.

6.6 Conclusions

We have investigated on practicability and usability of incorporating MTD techniques into GSMs as a framework on the real cloud. We have developed a cloud security framework which is able to run on a private cloud platform named UniteCloud. The developed framework can 1) automatically monitor the cloud and collect the information such as hosts, VMs, network, and vulnerabilities existing on each VM using OpenStack APIs, 2) model and evaluate the cloud's security and adopt defensive MTD techniques, 3) automate the deployment of three MTD techniques OS Diversification as the Diversity technique, VM replication as the Redundancy technique, and VM-LM as the shuffle technique on the infrastructures of the UniteCloud using API calls, and 4) use a web application UI for interaction between the security experts and the backend engine of the framework and also visualize the generated security model. Finally, we have evaluated MTD techniques based on real measurements and security metrics and showed that MTD techniques can be adopted in the real cloud infrastructure.

Chapter 7

Conclusions and Future Directions

The present thesis aimed to utilize the MTD techniques in a way that it enhances the cloud computing security. An extensive survey on MTD techniques was first conducted to find the gaps in the existing proposals. Then, combinations of MTD techniques were investigated, defined, and formulated for the cloud. Lastly, a cloud security framework was designed for a realistic testbed, which was able to adapt the MTD techniques on a real cloud testbed.

In Chapter 2, We conducted a survey of the studies on MTD techniques, their proposed techniques, key designs, principal concepts, domains of application, and implementation. We summarized the key limitations of the MTD techniques proposed in the literature and found relatively less effort in: 1) investigating the combinations of multiple MTD techniques, 2) evaluating the effectiveness of MTD techniques using security metrics in addition to performance and overhead assessment, and 3) adapting MTD technique on real cloud platforms, such as private, public, and hybrid clouds. Ultimately, based on the insights and lessons learned from the survey, we defined the research questions and the direction of this thesis. However, we realized other limitations, such as interplay between the MTD and other defensive mechanisms, investigation on optimal MTD techniques and combinations, which have important implications for future investigations.

In Chapter 3, we studied the combination of Shuffle and Redundancy MTD techniques and conducted the experimental results through simulation. We first formalized Shuffle and Redundancy MTD techniques based on the scalable HARM to combine Shuffle, Redundancy, and a combination of both. We then utilized Network Centrality Measures (NCMs) to improve the security analysis through HARM and to rank the most crucial VMs in the cloud. Our studies of these combinations revealed that leveraging Important Measures (IMs) can improve the security analysis and evaluation process. We also showed that we can combine Shuffle and Redundancy techniques to minimize the Risk while increasing the Reliability and holding system unattackability at an acceptable threshold, while deploying single MTD techniques cannot satisfy all desired metrics.

However, to investigate how the MTD techniques could enhance the security by hardening potential attacks on the cloud, it was recognized that the Shuffle and Diversity MTD techniques would need to be properly combined. In Chapter 4, we introduced the combinations of Shuffle and Diversity and evaluated their effectiveness. We formulated Shuffle and Diversity techniques by considering different combination scenarios and strategies. By incorporating eight different security metrics including the path-based metrics. In our combined model, we discovered that using IMs between deploying the Shuffle and Diversity can provide the most promising strategies to combine these MTD techniques, which could in tern yield better results (it increases the attack cost while reducing system risk and return on attack). However, because the Redundancy was not included in this combination, it is unable to increase the dependability (e.g., reliability) of the system.

The efficient combinations of MTD techniques we developed in Chapters 3 and 4 encouraged us to formulate and extend the combination of MTD techniques by including the Redundancy technique in addition to the Shuffle and Diversity. To this end, in Chapter 5, we investigated how those three techniques can be combined to efficiently secure the cloud from both the attacker's and the cloud provider's perspectives. The experimental results showed that combining Shuffle, Diversity, and Redundancy can contribute to provide security and reliability while making the attack more difficult for the attackers and increasing the attack cost. Moreover, we focused on more specific context using E-health cloud model to evaluate the effects of MTD techniques on economic metrics. We showed that deploying MTD techniques can increase the return on security investment values, while decreasing the return on attack metrics. To find the most promising MTD strategies, we proposed a migration strategy for deploying Shuffle which is able to deploy Shuffle on only a selected critical shortest paths in the network to avoid computational complexity of Exhaustive Search (ES) and also increase the attack path length. We also proposed an optimization model called *O-DAP* for Diversity and showed that our binary linear programming formulation handles large instances of Diversity allocation using various OS backup over multiple VMs with millions of feasible solutions in a fraction of a second on an ordinary computer. However, since deploying MTD techniques either solely or in combinations can introduce a trade-off between security, availability, and defensive costs, a multi-optimization problem for those conflicting goals can be further investigated in-depth such as using techniques seen in [35].

While Chapters 3–5 mainly have focused in the theoretical aspects of proposing and evaluating the combinations of MTD techniques, the importance of practical side of the research for either validation or usability of MTD techniques encouraged us to develop and adapt MTD techniques on a realistic cloud testbed. In Chapter 6, we developed and automated security analysis framework equipped with MTD defensive techniques in a real private cloud platform (that is UniteCloud). Our proposed cloud security framework showed that how API programming can help security analysis and MTD deployment phases to design and develop an automated cloud security framework on cloud. We also demonstrated that all MTD techniques Shuffle, Diversity, and Redundancy MTD are deployed on a real cloud platform with low effect in performance in terms of system downtime. However, performing MTD on real clouds to obtain real measurements is a difficult task since the cloud providers has their own policies and restrictions. To this end, the realistic results and measurements presented in Chapter 6 can be on an inspiration for further research.

Bibliography

[1] Unitecloud, http://www.unitecloud.net/.

[2] Towards scalable security analysis using multi-layered security models. *Journal of Network and Computer Applications 75* (2016), 156 – 168.

[3] Achleitner, S., La Porta, T., McDaniel, P., Sugrim, S., Krishnamurthy, S. V., and Chadha, R. Cyber deception: Virtual networks to defend insider reconnaissance. In *Proceedings of the 8th ACM CCS international workshop on managing insider security threats* (2016), pp. 57–68.

[4] Achleitner, S., Porta, T. L., McDaniel, P., Sugrim, S., Krishnamurthy, S. V., and Chadha, R. Deceiving network reconnaissance using SDN-based virtual topologies. *IEEE Transactions on Network and Service Management 14* (Dec. 2017), 1098–1112.

[5] Adili, M. T., Mohammadi, A., Manshaei, M. H., and Rahman, M. A. A cost-effective security management for clouds: A game-theoretic deception mechanism. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on* (2017), IEEE, pp. 98–106.

[6] Aikat, J., Akella, A., Chase, J. S., Juels, A., Reiter, M., Ristenpart, T., Sekar, V., and Swift, M. Rethinking security in the era of cloud computing. *IEEE Security & Privacy* (2017).

[7] Al-Haidari, F., Sqalli, M., and Salah, K. Impact of cpu utilization thresholds and scaling size on autoscaling cloud resources. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science* (2013), vol. 2, IEEE, pp. 256–261.

[8] Al-Shaer, E. Toward network configuration randomization for moving target defense. *Moving Target Defense* (2011), 153–159.

[9] Alavizadeh, H., Alavizadeh, H., Kim, D. S., Jang-Jaccard, J., and Niazi Torshiz, M. An automated security analysis framework and implementation for cloud. *arXiv preprint arXiv:1904.01758* (2019).

[10] Alavizadeh, H., Alavizadeh, H., Kim, D. S., Jang-Jaccard, J., and Niazi Torshiz, M. An automated security analysis framework and implementation for MTD techniques on cloud. In *International Conference on Information Security and Cryptology* (2019), Springer.

[11] Alavizadeh, H., Hong, J. B., Jang-Jaccard, J., and Kim, D. S. Comprehensive security assessment of combined mtd techniques for the cloud. In *Proceedings of the 5th ACM Workshop on Moving Target Defense* (2018), ACM, pp. 11–20.

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

[13] Alavizadeh, H., Kim, D. S., Hong, J. B., and Jang-Jaccard, J. Effective security analysis for combinations of mtd techniques on cloud computing (short paper). In *International Conference on Information Security Practice and Experience* (2017), Springer, pp. 539–548.

[14] Alavizadeh, H., Kim, D. S., and Jang-Jaccard, J. Model-based evaluation of combinations of shuffle and diversity MTD techniques on the cloud. *Future Generation Computer Systems* (2019).

[15] Ali, M., Khan, S. U., and Vasilakos, A. V. Security in cloud computing: Opportunities and challenges. *Information Sciences 305* (2015), 357–383.

[16] Antonatos, S., Akritidis, P., Markatos, E. P., and Anagnostakis, K. G. Defending against hitlist worms using network address space randomization. *Computer Networks 51*, 12 (2007), 3471–3490.

[17] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., et al. Above the clouds: A berkeley view of cloud computing. Tech. rep., Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[18] Avizienis, A. The n-version approach to fault-tolerant software. *IEEE Transactions on Software Engineering 11*, 12 (Dec. 1985), 1491–1501.

[19] Azab, M., Hassan, R., and Eltoweissy, M. Chameleonsoft: a moving target defense system. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on* (2011), IEEE, pp. 241–250.

[20] Baran, M. E., and Wu, F. F. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Transactions on Power Delivery 4*, 2 (Apr. 1989), 1401–1407.

[21] Bardas, A. G., Sundaramurthy, S. C., Ou, X., and DeLoach, S. A. Mtd cbits: Moving target defense for cloud-based it systems. In *European Symposium on Research in Computer Security* (2017), Springer, pp. 167–186.

[22] Bawany, N. Z., Shamsi, J. A., and Salah, K. Ddos attack detection and mitigation using sdn: methods, practices, and solutions. *Arabian Journal for Science and Engineering 42*, 2 (2017), 425–441.

[23] Beale, J., Deraison, R., Meer, H., Temmingh, R., and Walt, C. The NESSUS project. *Syngress Publishing* (2002).

[24] Bistarelli, S., Fioravanti, F., Peretti, P., and Santini, F. Evaluation of complex security scenarios using defense trees and economic indexes. *Journal of Experimental & Theoretical Artificial Intelligence 24*, 2 (2012), 161–192.

[25] Blakely, B., Horsthemke, W., Poczatec, A., Nowak, L., and Evans, N. Moving target, deception, and other adaptive defenses. In *Industrial Control Systems Security and Resiliency*. Springer, 2019, pp. 95–118.

[26] Böhme, R. Security metrics and security investment models. In *International Workshop on Security* (2010), Springer, pp. 10–24.

[27] Cadini, F., Zio, E., and Petrescu, C.-A. Using centrality measures to rank the importance of the components of a complex network infrastructure. In *CRITIS* (2008), Springer, pp. 155–167.

[28] Cai, G.-l., Wang, B.-s., Hu, W., and Wang, T.-z. Moving target defense: state of the art and characteristics. *Frontiers of Information Technology & Electronic Engineering 17*, 11 (2016), 1122–1153.

[29] Calyam, P., Rajagopalan, S., Seetharam, S., Selvadhurai, A., Salah, K., and Ramnath, R. Vdc-analyst: Design and verification of virtual desktop cloud resource allocations. *Computer Networks 68* (2014), 110–122.

[30] Carroll, T. E., Crouse, M., Fulp, E. W., and Berenhaut, K. S. Analysis of network address shuffling as a moving target defense. In *Proceedings of the IEEE International Conference on Communications (ICC)* (Jun. 2014), pp. 701–706.

[31] Carroll, T. E., Crouse, M., Fulp, E. W., and Berenhaut, K. S. Analysis of network address shuffling as a moving target defense. In *Communications (ICC), 2014 IEEE International Conference on* (2014), IEEE, pp. 701–706.

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

[33] Chatfield, B., and Haddad, R. J. Moving target defense intrusion detection system for ipv6 based smart grid advanced metering infrastructure. In *SoutheastCon, 2017* (2017), IEEE, pp. 1–7.

[34] Chen, L., and Avizienis, A. N-version programming: A fault-tolerance approach to reliability of software operation. In *Digest of Papers FTCS-8: Eight Annual International Conference on Fault-Tolerant Computing* (Toulouse, June 1978), pp. 3–9.

[35] Cho, J., Wang, Y., Chen, I., Chan, K. S., and Swami, A. A survey on modeling and optimizing multi-objective systems. *IEEE Communications Surveys Tutorials 19*, 3 (2017), 1867–1901.

[36] Christodorescu, M., Fredrikson, M., Jha, S., and Giffin, J. End-to-end software diversification of internet services. *Moving Target Defense* (2011), 117–130.

[37] Chung, C., Khatkar, P., Xing, T., Lee, J., and Huang, D. NICE: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems. *IEEE Transactions on Dependable and Secure Computing 10*, 4 (July 2013), 198–211.

[38] Colbaugh, R., and Glass, K. Predictability-oriented defense against adaptive adversaries. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (2012), IEEE, pp. 2721–2727.

[39] Colbaugh, R., and Glass, K. Moving target defense for adaptive adversaries. In *2013 IEEE International Conference on Intelligence and Security Informatics* (2013), IEEE, pp. 50–55.

[40] Compton, K., and Hauck, S. Reconfigurable computing: A survey of systems and software. *ACM Computing Surveys 34*, 2 (Jun. 2002), 171–210.

[41] Cook, K., Shaw, T., Hawrylak, P., and Hale, J. Scalable attack graph generation. In *Proceedings of the 11th Annual Cyber and Information Security Research Conference* (2016), ACM, p. 21.

[42] Cremonini, M., and Martini, P. Evaluating information security investments from attackers perspective: the return-on-attack (roa). In *WEIS* (2005).

[43] Crouse, M., and Fulp, E. W. A moving target environment for computer configurations using genetic algorithms. In *2011 4th Symposium on Configuration Analytics and Automation (SAFECONFIG)* (2011), IEEE, pp. 1–7.

[44] Crouse, M., Fulp, E. W., and Canas, D. Improving the diversity defense of genetic algorithm-based moving target approaches. In *Proceedings of the National Symposium on Moving Target Research* (2012).

[45] Danev, B., Masti, R., Karame, G., and Capkun, S. Enabling Secure VM-vTPM Migration in Private Clouds. In *Proc. of the 27th Annual Computer Security Applications Conference (ACSAC 2011)* (New York, NY, USA, 2011), ACM, pp. 187–196.

[46] Dawkins, J., Clark, K., Manes, G., and Papa, M. A framework for unified network security management: Identifying and tracking security threats on converged networks. *Journal of Network and Systems Management 13*, 3 (2005), 253–267.

[47] Dewri, R., Poolsappasit, N., Ray, I., and Whitley, D. Optimal Security Hardening using Multi-objective Optimization on Attack Tree Models of Networks. In *Proc. of ACM conference on Computer and communications security (CCS 2007)* (New York, NY, USA, 2007), ACM, pp. 204–213.

[48] Dewri, R., Ray, I., Poolsappasit, N., and Whitley, D. Optimal security hardening on attack tree models of networks: a cost-benefit analysis. *International Journal of Information Security 11*, 3 (2012), 167–188.

[49] Enoch, S. Y., Ge, M., Hong, J. B., Alzaid, H., and Kim, D. S. A systematic evaluation of cybersecurity metrics for dynamic networks. *Computer Networks 144* (2018), 216–229.

[50] Enoch, S. Y., Hong, J. B., Ge, M., Alzaid, H., and Kim, D. S. Automated security investment analysis of dynamic networks. In *Proceedings of the Australasian Computer Science Week Multiconference* (2018), ACM, p. 6.

[51] Franz, M. E unibus pluram: massive-scale software diversity as a defense mechanism. In *Proceedings of the New Security Paradigms Workshop* (2010), pp. 7–16.

[52] Ge, M., Hong, J. B., Yusuf, S. E., and Kim, D. S. Proactive defense mechanisms for the software-defined internet of things with non-patchable vulnerabilities. *Future Generation Computer Systems* (2017).

[53] Ge, M., Hong, J. B., Yusuf, S. E., and Kim, D. S. Proactive defense mechanisms for the software-defined internet of things with non-patchable vulnerabilities. *Future Generation Computer Systems 78* (2018), 568–582.

[54] Gherbi, A., and Charpentier, R. Diversity-based approaches to software systems security. In *Proceedings of the International Conference on Security Technology* (2011), pp. 228–237.

[55] Gillani, F., Al-Shaer, E., Lo, S., Duan, Q., Ammar, M., and Zegura, E. Agile virtualized infrastructure to proactively defend against cyber attacks. In *Computer Communications (INFOCOM), 2015 IEEE Conference on* (2015), IEEE, pp. 729–737.

[56] Gorbenko, A., Kharchenko, V., and Romanovsky, A. Using Inherent Service Redundancy and Diversity to Ensure Web Services Dependability. In *Methods, Models and Tools for Fault Tolerance*, M. Butler, C. Jones, A. Romanovsky, and E. Troubitsyna, Eds., vol. 5454 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 324–341.

[57] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2018. url: `www.gurobi.com/documentation/8.0/refman/index.html` date accessed 1 June 2018.

[58] Han, Y., Chan, J., Alpcan, T., and Leckie, C. Using virtual machine allocation policies to defend against co-resident attacks in cloud computing. *IEEE Transactions on Dependable and Secure Computing 14*, 1 (2017), 95–108.

[59] He, M., Pang, S., Lavrov, D., Lu, D., Zhang, Y., and Sarrafzadeh, A. Reverse replication of virtual machines (rrvm) for low latency and high availability services. In *Proceedings of the 9th International Conference on Utility and Cloud Computing* (2016), ACM, pp. 118–127.

[60] Hobson, T., Okhravi, H., Bigelow, D., Rudd, R., and Streilein, W. On the challenges of effective movement. In *Proceedings of the First ACM Workshop on Moving Target Defense* (2014), ACM, pp. 41–50.

[61] Hong, J., and Kim, D. HARMs: Hierarchical Attack Representation Models for Network Security Analysis. In *Proc. of the 10th Australian Information Security Management Conference on SECAU Security Congress (SECAU 2012)* (2012), pp. 1–8.

[62] Hong, J., and Kim, D. Performance Analysis of Scalable Attack Representation Models. In *Security and Privacy Protection in Information Processing Systems (SEC 2013)*, L. Janczewski, H. Wolfe, and S. Shenoi, Eds., vol. 405 of *IFIP Advances in Information and Communication Technology*. Springer Berlin Heidelberg, 2013, pp. 330–343.

[63] Hong, J., and Kim, D. Scalable Security Model Generation and Analysis Using k-importance Measures. In *Security and Privacy in Communication Networks (SecureComm 2013)*, T. Zia, A. Zomaya, V. Varadharajan, and M. Mao, Eds., vol. 127 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer International Publishing, 2013, pp. 270–287.

[64] Hong, J., and Kim, D. Towards Scalable Security Analysis Using Multi-layered Security Models. *Journal of Netwowrk and Computer Applications 75*, C (Nov. 2016), 156–168.

[65] Hong, J., Kim, D., and Haqiq, A. What Vulnerability Do We Need to Patch First? In *Proc. of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSNW 2014)* (2014), pp. 684–689.

[66] Hong, J. B., and Kim, D. S. Assessing the effectiveness of moving target defenses using security models. *IEEE Transactions on Dependable and Secure Computing 13*, 2 (2016), 163–177.

[67] Hong, J. B., Kim, D. S., Chung, C.-J., and Huang, D. A survey on the usability and practical applications of graphical security models. *Computer Science Review 26* (2017), 1–16.

[68] Hong, J. B., Yoon, S., Lim, H., and Kim, D. S. Optimal network reconfiguration for software defined networks using shuffle-based online MTD. In *IEEE Symposium on Reliable Distributed Systems (SRDS)* (2017).

[69] Huang, H., Zhang, S., Ou, X., Prakash, A., and Sakallah, K. Distilling critical attack graph surface iteratively through minimum-cost sat solving. In *Proceedings of the 27th Annual Computer Security Applications Conference* (2011), ACM, pp. 31–40.

[70] Huang, Y., and Ghosh, A. Introducing Diversity and Uncertainty to Create Moving Attack Surfaces for Web Services. In *Moving Target Defense*, S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Eds., vol. 54 of *Advances in Information Security*. Springer New York, 2011, pp. 131–151.

[71] Huang, Y., Ghosh, A. K., Bracewell, T., and Mastropietro, B. A security evaluation of a novel resilient web serving architecture: Lessons learned through industry/academia collaboration. In *Dependable Systems and Networks Workshops (DSN-W), 2010 International Conference on* (2010), IEEE, pp. 188–193.

[72] Hussain, S. A., Fatima, M., Saeed, A., Raza, I., and Shahzad, R. K. Multilevel classification of security concerns in cloud computing. *Applied Computing and Informatics 13*, 1 (2017), 57–65.

[73] Ingols, K., Chu, M., Lippmann, R., Webster, S., and Boyer, S. Modeling Modern Network Attacks and Countermeasures Using Attack Graphs. In *Proc. of the 25th Annual Computer Security Applications Conference (ACSAC 2009)* (2009), pp. 117–126.

[74] Jackson, T., Salamat, B., Homescu, A., Manivannan, K., Wagner, G., Gal, A., Brunthaler, S., Wimmer, C., and Franz, M. Compiler-Generated Software Diversity.

In *Moving Target Defense*, S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Eds., vol. 54 of *Advances in Information Security*. Springer New York, 2011, pp. 77–98.

[75] Jafarian, J., Al-Shaer, E., and Duan, Q. Openflow Random Host Mutation: Transparent Moving Target Defense Using Software Defined Networking. In *Proc. of the 1st Workshop on Hot Topics in Software Defined Networks (HotSDN 2012)* (New York, NY, USA, 2012), ACM, pp. 127–132.

[76] Jafarian, J. H., Al-Shaer, E., and Duan, Q. An effective address mutation approach for disrupting reconnaissance attacks. *IEEE Transactions on Information Forensics and Security 10*, 12 (Dec 2015), 2562–2577.

[77] Jia, F., Hong, J. B., and Kim, D. S. Towards automated generation and visualization of hierarchical attack representation models. In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on* (2015), IEEE, pp. 1689–1696.

[78] Jia, Q., Sun, K., and Stavrou, A. MOTAG: Moving Target Defense against Internet Denial of Service Attacks. In *Proc. of the 22nd International Conference on Computer Communications and Networks (ICCCN 2013)* (2013), pp. 1–9.

[79] Jia, Q., Wang, H., Fleck, D., Li, F., Stavrou, A., and Powell, W. Catch Me if You Can: A Cloud-Enabled DDoS Defense. In *Proc. of the the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014)* (Jun 2014).

[80] Jiang, J., Wen, S., Yu, S., Xiang, Y., and Zhou, W. Identifying propagation sources in networks: State-of-the-art and comparative studies. *IEEE Communications Surveys & Tutorials 19*, 1 (2016), 465–481.

[81] John, D. J., Smith, R. W., Turkett, W. H., Cañas, D. A., and Fulp, E. W. Evolutionary based moving target cyber defense. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation* (2014), ACM, pp. 1261–1268.

[82] Kampanakis, P., Perros, H., and Beyene, T. Sdn-based solutions for moving target defense network protection. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a* (2014), IEEE, pp. 1–6.

[83] Khan, M. A., and Salah, K. Iot security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems 82* (2018), 395–411.

[84] Kijsanayothin, P., and Hewett, R. Analytical approach to attack graph analysis for network security. In *Availability, Reliability, and Security, 2010. ARES'10 International Conference on* (2010), IEEE, pp. 25–32.

[85] Kirrmann, H., and Dzung, D. Selecting a Standard Redundancy Method for Highly Available Industrial Networks. In *Proc. of the 3rd IEEE International Workshop on Factory Communication Systems (WFCS 2006)* (2006), pp. 386–390.

[86] Knight, J., Davidson, J., Nguyen-Tuong, A., Hiser, J., and Co, M. Diversity in cybersecurity. *Computer 49*, 4 (Apr. 2016), 94–98.

[87] Kordy, B., Mauw, S., Radomirović, S., and Schweitzer, P. Attack–defense trees. *Journal of Logic and Computation 24*, 1 (2014), 55–87.

[88] Kordy, B., Pietre-Cambacedes, L., and Schweitzer, P. DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. *CoRR abs/1303.7397* (2013).

[89] Kordy, B., Piètre-Cambacédès, L., and Schweitzer, P. DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. *Computer Science Review 13* (2014), 1–38.

[90] Kotenko, I., and Chechulin, A. Computer attack modeling and security evaluation based on attack graphs. In *2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)* (2013), vol. 2, IEEE, pp. 614–619.

[91] Kotenko, I., and Stepashkin, M. Attack graph based evaluation of network security. *Lecture Notes in Computer Science 4237* (2006), 216–227.

[92] Kotenko, I. V., and Doynikova, E. Evaluation of computer network security based on attack graphs and security event processing. *JoWUA 5*, 3 (2014), 14–29.

[93] Krutz, R. L., Vines, R. D., and Stroz, E. M. *The CISSP prep Guide: Mastering the ten domains of Computer Security.* Wiley New York, 2001.

[94] Lala, J. H., and Schneider, F. B. It monoculture security risks and defenses. *IEEE Security Privacy 7*, 1 (Jan. 2009), 12–13.

[95] Larsen, P., Homescu, A., Brunthaler, S., and Franz, M. SoK: Automated software diversity. In *2014 IEEE Symposium on Security and Privacy* (2014), IEEE, pp. 276–291.

[96] Li, Y., Dai, R., and Zhang, J. Morphing communications of cyber-physical systems towards moving-target defense. In *2014 IEEE International Conference on Communications (ICC)* (June 2014), pp. 592–598.

[97] Liu, A. X., and Gouda, M. G. Diverse firewall design. *IEEE Transactions on Parallel and Distributed Systems 19*, 9 (2008), 1237–1251.

[98] Liu, H. A new form of dos attack in a cloud and its avoidance mechanism. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop* (2010), ACM, pp. 65–76.

[99] Liu, L., De Vel, O., Han, Q.-L., Zhang, J., and Xiang, Y. Detecting and preventing cyber insider threats: a survey. *IEEE Communications Surveys & Tutorials 20*, 2 (2018), 1397–1417.

[100] Luo, Y.-B., Wang, B.-S., and Cai, G.-L. Effectiveness of port hopping as a moving target defense. In *Security Technology (SecTech), 2014 7th International Conference on* (2014), IEEE, pp. 7–10.

[101] MacFarland, D. C., and Shue, C. A. The SDN shuffle: Creating a Moving-Target Defense using host-based Software-Defined Networking. In *in Proceedings of the Second ACM Workshop on Moving Target Defense (MTD)* (2015), pp. 37–41.

[102] Manadhata, P., and Wing, J. An Attack Surface Metric. *IEEE Transactions on Software Engineering 37*, 3 (2011), 371–386.

[103] Manadhata, P. K. Game theoretic approaches to attack surface shifting. In *Moving Target Defense II*. Springer, 2013, pp. 1–13.

[104] Mehta, V., Bartzis, C., Zhu, H., Clarke, E., and Wing, J. Ranking Attack Graphs. In *Proc. of the 9th international conference on Recent Advances in Intrusion Detection (RAID 2006)* (Berlin, Heidelberg, 2006), Springer-Verlag, pp. 127–144.

[105] Mell, P., Grance, T., et al. The nist definition of cloud computing.

[106] Mell, P., Scarfone, K., and Romanosky, S. Common vulnerability scoring system. *IEEE Security & Privacy 4*, 6 (2006).

[107] Moon, S.-J., Sekar, V., and Reiter, M. K. Nomad: Mitigating arbitrary cloud side channels via provider-assisted migration. In *Proceedings of the 22nd acm sigsac conference on computer and communications security* (2015), ACM, pp. 1595–1606.

[108] Moreno-Vozmediano, R., Montero, R. S., Huedo, E., and Llorente, I. M. Cross-site virtual network in cloud and fog computing. *IEEE Cloud Computing 4*, 2 (2017), 46–53.

[109] Nespoli, P., Papamartzivanos, D., Mármol, F. G., and Kambourakis, G. Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks. *IEEE Communications Surveys & Tutorials 20*, 2 (2018), 1361–1396.

[110] Newell, A., Obenshain, D., Tantillo, T., Nita-Rotaru, C., and Amir, Y. Increasing network resiliency by optimally assigning diverse variants to routing nodes. *IEEE Transactions on Dependable and Secure Computing 12*, 6 (2015), 602–614.

[111] Nguyen, Q. L., and Sood, A. Scalability of cloud based scit-mtd. In *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)* (2017), IEEE, pp. 581–582.

[112] Nhlabatsi, A. M., Hong, J. B., Kim, D. S. D., Fernandez, R., Hussein, A., Fetais, N., and Khan, K. M. Threat-specific security risk evaluation in the cloud. *IEEE Transactions on Cloud Computing* (2018).

[113] NIST. National vulnerability database (nvd).

[114] of Homeland Security, D. Moving target defense, 2018.

[115] Okhravi, H., Comella, A., Robinson, E., and Haines, J. Creating a cyber moving target for critical infrastructure applications using platform diversity. *International Journal of Critical Infrastructure Protection 5*, 1 (2012), 30–39.

[116] Okhravi, H., Comella, A., Robinson, E., Yannalfo, S., Michaleas, P., and Haines, J. Creating a cyber moving target for critical infrastructure applications. In *Critical Infrastructure Protection V*, J. Butts and S. Shenoi, Eds., vol. 367 of *IFIP Advances in Information and Communication Technology.* Springer Berlin Heidelberg, 2011, pp. 107–123.

[117] Pang, S., Shi, T., Zhang, R., and Lavrov, D. Cdmc task 2: Incident detection over unified threat management (utm) operation on unite-cloud. *Unitec Institute of Technology, Auckland, New Zealand* (2017).

[118] Pendleton, M., Garcia-Lebron, R., Cho, J.-H., and Xu, S. A survey on systems security metrics. *ACM Computing Surveys (CSUR) 49*, 4 (2017), 62.

[119] Peng, W., Li, F., Huang, C.-T., and Zou, X. A moving-target defense strategy for cloud-based services with heterogeneous and dynamic attack surfaces. In *Communications (ICC), 2014 IEEE International Conference on* (2014), IEEE, pp. 804–809.

[120] Penner, T., and Guirguis, M. Combating the bandits in the cloud: A moving target defense approach. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (2017), IEEE Press, pp. 411–420.

[121] Phillips, C., and Swiler, L. A Graph-based System for Network-vulnerability Analysis. In *Proc. of the 1998 Workshop on New Security Paradigms (NSPW 1998)* (New York, NY, USA, 1998), ACM, pp. 71–79.

[122] Poolsappasit, N., Dewri, R., and Ray, I. Dynamic Security Risk Management Using Bayesian Attack Graphs. *IEEE Transactions on Dependable and Secure Computing (TDSC 2012) 9*, 1 (2012), 61–74.

[123] Popović, K., and Hocenski, Ž. Cloud computing security issues and challenges. In *The 33rd International Convention MIPRO* (2010), IEEE, pp. 344–349.

[124] Rabai, L. B. A., Jouini, M., Aissa, A. B., and Mili, A. A cybersecurity model in cloud computing environments. *Journal of King Saud University-Computer and Information Sciences 25*, 1 (2013), 63–75.

[125] Rimal, B. P., Choi, E., and Lumb, I. A taxonomy and survey of cloud computing systems. *NCM 9* (2009), 44–51.

[126] Rohrer, J., Jabbar, A., and Sterbenz, J. Path Diversification for Future Internet End-to-End Resilience and Survivability. *Telecommunication Systems* (2013), 1–19.

[127] Rohrer, J. P., Jabbar, A., and Sterbenz, J. P. Path diversification for future internet end-to-end resilience and survivability. *Telecommunication Systems 56*, 1 (2014), 49–67.

[128] Roy, A., Kim, D., and Trivedi, K. Attack Countermeasure Trees (ACT): Towards Unifying the Constructs of Attack and Defense Trees. *Security and Communication Networks 5*, 8 (2012), 929–943.

[129] Sabi, H. M., Uzoka, F.-M. E., Langmia, K., Njeh, F. N., and Tsuma, C. K. A cross-country model of contextual factors impacting cloud computing adoption at universities in sub-saharan africa. *Information Systems Frontiers* (2017), 1–24.

[130] Sahner, R. A., Trivedi, K., and Puliafito, A. *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package.* Springer Publishing Company, Incorporated, 2012.

[131] Schneier, B. Attack trees. *Dr. Dobb's journal 24*, 12 (1999), 21–29.

[132] Schneier, B. *Secrets and Lies: Digital Security in a Networked World.* John Wiley and Sons Inc., 2000.

[133] Scott-Hayward, S., O'Callaghan, G., and Sezer, S. Sdn security: A survey. In *Future Networks and Services (SDN4FNS), 2013 IEEE SDN For* (2013), IEEE, pp. 1–7.

[134] Sengupta, S., Chowdhary, A., Huang, D., and Kambhampati, S. Moving target defense for the placement of intrusion detection systems in the cloud. In *International Conference on Decision and Game Theory for Security* (2018), Springer, pp. 326–345.

[135] Sgandurra, D., and Lupu, E. Evolution of attacks, threat models, and solutions for virtualized systems. *ACM Computing Surveys (CSUR) 48*, 3 (2016), 46.

[136] Sharma, D. P., Kim, D. S., Yoon, S., Lim, H., Cho, J., and Moore, T. J. FRVM: Flexible random virtual IP multiplexing in software-defined networks. In *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* (Aug, 2018), pp. 579–587.

[137] Sharma, D. P., Kim, D. S., Yoon, S., Lim, H., Cho, J.-H., and Moore, T. J. Frvm: Flexible random virtual ip multiplexing in software-defined networks. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* (2018), IEEE, pp. 579–587.

[138] Sheldon, F. T., and Vishik, C. Moving toward trustworthy systems: R&d essentials. *Computer 43*, 9 (2010), 31–40.

[139] Sheyner, O., Haines, J., Jha, S., Lippmann, R., and Wing, J. Automated Generation and Analysis of Attack Graphs. Tech. rep., CMU, 2002.

[140] Sonnenreich, W., Albanese, J., Stout, B., et al. Return on security investment (rosi)-a practical quantitative model. *Journal of Research and practice in Information Technology 38*, 1 (2006), 45.

[141] Steinberger, J., Kuhnert, B., Dietz, C., Ball, L., Sperotto, A., Baier, H., Pras, A., and Dreo, G. Ddos defense using mtd and sdn. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium* (2018), IEEE, pp. 1–9.

[142] Sultan, N. Cloud computing for education: A new dawn? *International Journal of Information Management 30*, 2 (2010), 109–116.

[143] Sun, N., Zhang, J., Rimba, P., Gao, S., Zhang, L. Y., and Xiang, Y. Data-driven cybersecurity incident prediction: A survey. *IEEE Communications Surveys & Tutorials 21*, 2 (2018), 1744–1772.

[144] Sun, X., Dai, J., Singhal, A., and Liu, P. Inferring the stealthy bridges between enterprise network islands in cloud using cross-layer bayesian networks. In *International Conference on Security and Privacy in Communication Systems* (2014), Springer, pp. 3–23.

[145] Taguinod, M., Doupé, A., Zhao, Z., and Ahn, G.-J. Toward a moving target defense for web applications. In *Information Reuse and Integration (IRI), 2015 IEEE International Conference on* (2015), IEEE, pp. 510–517.

[146] Taylor, C., and Alves-Foss, J. Diversity as a computer defense mechanism. In *Proceedings of the 2005 Workshop on New Security Paradigms* (2005), pp. 11–14.

[147] Tozer, B., Mazzuchi, T., and Sarkani, S. Optimizing attack surface and configuration diversity using multi-objective reinforcement learning. In *2015 ieee 14th international conference on machine learning and applications (icmla)* (2015), IEEE, pp. 144–149.

[148] Tozer, B., Mazzuchi, T., and Sarkani, S. Optimizing attack surface and configuration diversity using multi-objective reinforcement learning. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)* (Dec. 2015), pp. 144–149.

[149] Unruh, I., Bardas, A. G., Zhuang, R., Ou, X., and DeLoach, S. A. Compiling abstract specifications into concrete systems—bringing order to the cloud. In *28th Large Installation System Administration Conference (LISA14)* (2014), pp. 26–42.

[150] Venkatesan, S., Albanese, M., Amin, K., Jajodia, S., and Wright, M. A moving target defense approach to mitigate ddos attacks against proxy-based architectures. In *Communications and Network Security (CNS), 2016 IEEE Conference on* (2016), IEEE, pp. 198–206.

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

[152] Wang, L., Islam, T., Long, T., Singhal, A., and Jajodia, S. An attack graph-based probabilistic security metric. *Lecture Notes in Computer Science 5094* (2008), 283–296.

[153] Wen, S., Haghighi, M. S., Chen, C., Xiang, Y., Zhou, W., and Jia, W. A sword with two edges: Propagation studies on both positive and negative information in online social networks. *IEEE Transactions on Computers 64*, 3 (2014), 640–653.

[154] Yuan, E., Malek, S., Schmerl, B., Garlan, D., and Gennari, J. Architecture-Based Self-Protecting Software Systems. In *Proc. of the 9th International ACM Sigsoft Conference on the Quality of Software Architectures (QoSA 2013)* (2013), pp. 33–42.

[155] Yusuf, S. E., Ge, M., Hong, J. B., Alzaid, H., and Kim, D. S. Evaluating the effectiveness of security metrics for dynamic networks. In *2017 IEEE Trustcom* (2017), IEEE, pp. 277–284.

[156] Yusuf, S. E., Ge, M., Hong, J. B., Kim, H. K., Kim, P., and Kim, D. S. Security modelling and analysis of dynamic enterprise networks. In *Computer and Information Technology (CIT), 2016 IEEE International Conference on* (2016), IEEE, pp. 249–256.

[157] Yusuf, S. E., Hong, J. B., Ge, M., and Kim, D. S. Composite metrics for network security analysis. *Software Networking 2017*, 1 (2017), 137–160.

[158] Yusuf, S. E., Hong, J. B., Ge, M., and Kim, D. S. Composite metrics for network security analysis. *Software Networking 2018*, 1 (2018), 137–160.

[159] Zacks, S. *Introduction to reliability analysis: probability models and statistical methods*. Springer Science & Business Media, 2012.

[160] Zhang, H.-q., Lei, C., Chang, D.-x., and Yang, Y.-j. Network moving target defence technique based on collaborative mutation. *Computers & Security* (2017).

[161] Zhang, J., Xiang, Y., Wang, Y., Zhou, W., Xiang, Y., and Guan, Y. Network traffic classification using correlation information. *IEEE Transactions on Parallel and Distributed systems 24*, 1 (2012), 104–117.

[162] Zhang, L., Shetty, S., Liu, P., and Jing, J. Rootkitdet: Practical end-to-end defense against kernel rootkits in a cloud environment. In *European Symposium on Research in Computer Security* (2014), Springer, pp. 475–493.

[163] Zhang, Q., Cheng, L., and Boutaba, R. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications 1*, 1 (2010), 7–18.

[164] Zhang, Y., Li, M., Bai, K., Yu, M., and Zang, W. Incentive compatible moving target defense against vm-colocation attacks in clouds. In *SEC* (2012), Springer, pp. 388–399.

[165] Zheng, J., and Namin, A. S. A survey on the moving target defense strategies: An architectural perspective. *Journal of Computer Science and Technology 34*, 1 (Jan 2019), 207–233.

[166] Zhu, M., Hu, Z., and Liu, P. Reinforcement learning algorithms for adaptive cyber defense against heartbleed. In *Proceedings of the First ACM Workshop on Moving Target Defense* (2014), ACM, pp. 51–58.

[167] Zhu, Y., Hu, H., Ahn, G., Huang, D., and Wang, S. Towards temporal access control in cloud computing. In *Proc. of Annual IEEE International Conference on Computer Communications (INFOCOM 2012)* (2012), pp. 2576–2580.

[168] Zhuang, R., DeLoach, S. A., and Ou, X. Towards a theory of moving target defense. In *Proceedings of the First ACM Workshop on Moving Target Defense* (2014), ACM, pp. 31–40.

[169] Zhuang, R., Zhang, S., Bardas, A., DeLoach, S., Ou, X., and Singhal, A. Investigating the Application of Moving Target Defenses to Network Security. In *Proc. of the 6th International Symposium on Resilient Control Systems (ISRCS 2013)* (2013), pp. 162–169.

[170] Zhuang, R., Zhang, S., DeLoach, S., Ou, X., and Singhal, A. Simulation-based Approaches to Studying Effectiveness of Moving-Target Network Defense. In *Proc. of National Symposium on Moving Target Research* (2012).

[171] Zissis, D., and Lekkas, D. Addressing cloud computing security issues. *Future Generation computer systems 28*, 3 (2012), 583–592.

DRC 16

MASSEY UNIVERSITY
GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the candidate and the candidate's Primary Supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the candidate's contribution as indicated below in the *Statement of Originality*.

Name of candidate:	Hooman Alavizadeh
Name/title of Primary Supervisor:	Associate Professor Julian Jang-Jaccard
Name of Research Output and full reference:	
Alavizadeh, H., Kim, D. S., Hong, J. B., and Jang-Jaccard, J. Effective security analysis for combinations of MTD techniques on cloud computing (short paper). In International Conference on Information Security Practice and Experience (ISPEC) (2017), Springer, pp. 539-548.	
In which Chapter is the Manuscript /Published work:	Chapter 3
Please indicate:	
• The percentage of the manuscript/Published Work that was contributed by the candidate:	80
and	
• Describe the contribution that the candidate has made to the Manuscript/Published Work:	
Designing study, carrying out the results, writing manuscript, and responding the reviewers' comments.	
For manuscripts intended for publication please indicate target journal:	
N/A	
Candidate's Signature:	H. Alavizadeh
Date:	12/09/2019
Primary Supervisor's Signature:	Julian Jang-Jaccard
Date:	05/12/2019

(This form should appear at the end of each thesis chapter/section/appendix submitted as a manuscript/ publication or collected as an appendix at the end of the thesis)

DRC 16

MASSEY UNIVERSITY
GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the candidate and the candidate's Primary Supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the candidate's contribution as indicated below in the *Statement of Originality*.

Name of candidate:	Hooman Alavizadeh
Name/title of Primary Supervisor:	Associate Professor Julian Jang-Jaccard
Name of Research Output and full reference:	
Alavizadeh, H., Hong, J. B., Kim, D. S., and Jang-Jaccard, J. Evaluating the Effectiveness of Shuffle and Redundancy MTD Techniques in the Cloud. Submitted to IEEE Transactions on Dependable and Secure Computing (Under Review)	
In which Chapter is the Manuscript /Published work:	Chapter 3
Please indicate:	
• The percentage of the manuscript/Published Work that was contributed by the candidate:	85
and	
• Describe the contribution that the candidate has made to the Manuscript/Published Work:	
Designing study, carrying out the results, writing manuscript, and responding the reviewers' comments.	
For manuscripts intended for publication please indicate target journal:	
N/A	
Candidate's Signature:	H. Alavizadeh
Date:	12/09/2019
Primary Supervisor's Signature:	Julian Jang-Jaccard
Date:	05/12/2019

(This form should appear at the end of each thesis chapter/section/appendix submitted as a manuscript/ publication or collected as an appendix at the end of the thesis)

MASSEY UNIVERSITY
GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the candidate and the candidate's Primary Supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the candidate's contribution as indicated below in the *Statement of Originality*.

Name of candidate:	Hooman Alavizadeh
Name/title of Primary Supervisor:	Associate Professor Julian Jang-Jaccard
Name of Research Output and full reference:	
Alavizadeh, H., Jang-Jaccard, J., and Kim, D. S. Evaluation for combination of Shuffle and Diversity on Moving Target Defense strategy for cloud computing. In 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) (2018), IEEE, pp. 573-578.	
In which Chapter is the Manuscript /Published work:	Chapter 4
Please indicate:	
• The percentage of the manuscript/Published Work that was contributed by the candidate:	85
and	
• Describe the contribution that the candidate has made to the Manuscript/Published Work:	
Designing study, carrying out the results, writing manuscript.	
For manuscripts intended for publication please indicate target journal:	
Candidate's Signature:	H. Alavizadeh
Date:	12/09/2019
Primary Supervisor's Signature:	Julian Jang-Jaccard
Date:	05/12/2019

(This form should appear at the end of each thesis chapter/section/appendix submitted as a manuscript/ publication or collected as an appendix at the end of the thesis)

MASSEY UNIVERSITY
GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the candidate and the candidate's Primary Supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the candidate's contribution as indicated below in the *Statement of Originality*.

Name of candidate:	Hooman Alavizadeh
Name/title of Primary Supervisor:	Associate Professor Julian Jang-Jaccard
Name of Research Output and full reference:	
Alavizadeh, H., Kim, D. S., and Jang-Jaccard, J. Model-based Evaluation of Combinations of Shuffle and Diversity MTD Techniques on the Cloud. Future Generation Computing Systems (FGCS) (2019)	
In which Chapter is the Manuscript /Published work:	Chapter 4
Please indicate:	
• The percentage of the manuscript/Published Work that was contributed by the candidate:	90
and	
• Describe the contribution that the candidate has made to the Manuscript/Published Work:	
Designing study, carrying out the results, writing manuscript, and responding the reviewers' comments.	
For manuscripts intended for publication please indicate target journal:	
Candidate's Signature:	H. Alavizadeh
Date:	04/12/2019
Primary Supervisor's Signature:	Julian Jang-Jaccard
Date:	05/12/2019

(This form should appear at the end of each thesis chapter/section/appendix submitted as a manuscript/ publication or collected as an appendix at the end of the thesis)

MASSEY UNIVERSITY
GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the candidate and the candidate's Primary Supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the candidate's contribution as indicated below in the *Statement of Originality*.

Name of candidate:	Hooman Alavizadeh
Name/title of Primary Supervisor:	Associate Professor Julian Jang-Jaccard
Name of Research Output and full reference:	
Alavizadeh, H., Hong, J. B., Jang-Jaccard, J., and Kim, D. S. Comprehensive security assessment of combined MTD techniques for the cloud. In Proceedings of the 5th ACM Workshop on Moving Target Defense (2018), ACM, pp. 11-20.	
In which Chapter is the Manuscript /Published work:	Chapter 5
Please indicate:	
• The percentage of the manuscript/Published Work that was contributed by the candidate:	80
and	
• Describe the contribution that the candidate has made to the Manuscript/Published Work:	
Designing study, carrying out the results, writing manuscript.	
For manuscripts intended for publication please indicate target journal:	
Candidate's Signature:	H. Alavizadeh
Date:	12/09/2019
Primary Supervisor's Signature:	Julian Jang-Jaccard
Date:	05/12/2019

(This form should appear at the end of each thesis chapter/section/appendix submitted as a manuscript/ publication or collected as an appendix at the end of the thesis)

DRC 16

MASSEY UNIVERSITY
GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the candidate and the candidate's Primary Supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the candidate's contribution as indicated below in the *Statement of Originality*.

Name of candidate:	Hooman Alavizadeh
Name/title of Primary Supervisor:	Associate Professor Julian Jang-Jaccard
Name of Research Output and full reference:	
Alavizadeh, H., Aref, S, Kim, D. S. and Jang-Jaccard, J. Security and Economic Modeling and Analysis of MTD techniques for Cloud Computing. To be submitted to IEEE Transactions on Emerging Topics in Computing (IEEE ETC).	
In which Chapter is the Manuscript /Published work:	Chapter 5
Please indicate:	
• The percentage of the manuscript/Published Work that was contributed by the candidate:	80
and	
• Describe the contribution that the candidate has made to the Manuscript/Published Work:	
Designing study, carrying out the results, writing manuscript.	
For manuscripts intended for publication please indicate target journal:	
To be submitted to IEEE Transactions on Emerging Topics in Computing	
Candidate's Signature:	H. Alavizadeh
Date:	04/12/2019
Primary Supervisor's Signature:	Julian Jang-Jaccard
Date:	05/12/2019

(This form should appear at the end of each thesis chapter/section/appendix submitted as a manuscript/ publication or collected as an appendix at the end of the thesis)

DRC 16

MASSEY UNIVERSITY
GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the candidate and the candidate's Primary Supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the candidate's contribution as indicated below in the *Statement of Originality*.

Name of candidate:	Hooman Alavizadeh
Name/title of Primary Supervisor:	Associate Professor Julian Jang-Jaccard
Name of Research Output and full reference:	
Alavizadeh, H., Alavizadeh, H., Kim, D. S., Jang-Jaccard, J., and Niazi Torshiz, M. An automated security analysis framework and implementation for MTD techniques on cloud. In International Conference on Information Security and Cryptology (2019), Springer.	
In which Chapter is the Manuscript /Published work:	Chapter 6
Please indicate:	
• The percentage of the manuscript/Published Work that was contributed by the candidate:	70
and	
• Describe the contribution that the candidate has made to the Manuscript/Published Work:	
Designing study, carrying out the results, writing manuscript.	
For manuscripts intended for publication please indicate target journal:	
Candidate's Signature:	H. Alavizadeh
Date:	04/12/2019
Primary Supervisor's Signature:	Julian Jang-Jaccard
Date:	05/12/2019

(This form should appear at the end of each thesis chapter/section/appendix submitted as a manuscript/ publication or collected as an appendix at the end of the thesis)