Threat Modeling for Cloud and NFV Infrastructures

Nawaf Alhebaishi

A thesis
in
The Concordia Institute
for
Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Information and Systems Engineering) at
Concordia University
Montréal, Québec, Canada

September 2020

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Nawaf Alhebaishi**
Entitled: **Threat Modeling for Cloud and NFV Infrastructures**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. Xiupu Zhang	Chair
Dr. Xiaodong Lin	External Examiner
Dr. Dongyu Qiu	External to Program
Dr. Amr Youssef	Examiner
Dr. Walter Lucia	Examiner
Dr. Lingyu Wang	Thesis Supervisor

Approved by ______________________
Dr. Mohammad Mannan, Graduate Program Director

November 02, 2020 ______________________
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Threat Modeling for Cloud and NFV Infrastructures

Nawaf Alhebaishi, Ph.D.

Concordia University, 2020

Today's businesses are increasingly relying on the cloud as an alternative IT solution due to its flexibility and lower cost. Compared to traditional enterprise networks, a cloud infrastructure is typically much larger and more complex. Understanding the potential security threats in such infrastructures is naturally more challenging than in traditional networks. This is evidenced by the fact that there are limited efforts on threat modeling for cloud infrastructures. My doctoral research will tackle several issues related to this.

In the first topic, we have conducted comprehensive threat modeling exercises based on two representative cloud infrastructures using several popular threat modeling methods, including attack surface, attack trees, attack graphs, and security metrics based on attack trees and attack graphs, respectively. In addition, we show how hardening solution can be applied based on the threat models and security metrics through extended exercises. Such results may not only benefit the cloud provider but also embed more confidence in cloud tenants by providing them a clearer picture of the potential threats and mitigation solutions.

In the second topic, we take the first step towards understanding and mitigating the insider threats of remote administrators in clouds. First, we model the maintenance task assignments and their corresponding security impact due to privilege escalation. Second, we mitigate such impact through optimizing the task assignments with respect to given constraints. Finally, the simulation results demonstrate the effectiveness of our solution in various scenarios.

In the third topic, we focus on modeling and mitigating security threats unique to network functions virtualization (NFV), which provides virtualization of network functions based on cloud infrastructures. First, we model both cross-layer and co-residency attacks on the NFV stack. Second, we mitigate such threats through optimizing the virtual machine (VM) placement with respect to given constraints. The simulation results demonstrate the effectiveness of our solution.

Acknowledgments

I would like to express my deepest thanks to my supervisor, Prof. Lingyu Wang, for his heartily guidance, endless support and enduring patience during the whole process of this research. Without him, this thesis would not have been possible. I have been extremely lucky to have a supervisor who lightened my research all the time, and who responded to my questions and queries so promptly.

I would also like to thank my PhD committee members, Prof. Amr Youssef, Dr. Dongyu Qiu, and Dr. Walter Lucia from Concordia University, for their discussions, constructive comments, and time, which helped me to refine and improve my thesis. I also would like to thank Dr. Xiaodong Lin my external examiner from University of Guelph, for his time and dedicated work in examining my thesis.

I would like to offer my deepest gratitude and appreciation to my wonderful parents, Mrs. Sabah Helal and Mr. Jamal Alhebaishi, for their continuous support, immense love, ultimate kindness, and phenomenal faith in me, which have motivated me to strive toward my full potential. However, words cannot express my incredible appreciation and gratitude to them.

I would also like to offer my endless gratitude and appreciation to my soulmate, Mrs. Noran Alkhouly, for her endless love, support, care, and patience during my PhD journey. Noran is a wonderful and loving mother to our two children, Yousouf and Shahad. I am forever grateful for her.

I would like to thank my lovely children, Yousouf and Shahad, for being a source of happiness and motivation, which enormously helped me to overcome times of stress and frustration during this journey.

I would like to extend my appreciation to my sisters, Nouf, and Sama Alhebaishi; and my brother Samer, and Mohammed Alhebaishi for their support, encouragement, and belief in me.

I would also like to offer special thanks and appreciation to my mother in law, Mrs. Faiza Alkhoumaly, for her support, effort, and coming all the way from Saudi Arabia to Canada several times to help us during tough times.

I would also like to express my appreciation to my colleagues in Concordia University, for their insightful discussions and valuable remarks, especially my friends, Dr. Saed Alrabaee, Dr. Mengyuan Zhang, Dr. Suryadipta Majumdar, Paria Shirani, Daniel Borbor Cedeño, Sudershan Lakshmanan, and Mina Khalili

Finally, I would like to thank the Saudi Ministry of Education and King Abdulaziz University, for providing generous financial support during my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Today cloud computing is becoming one of the most trending IT solutions for saving costs and gaining various other benefits. Many businesses, government agencies and organizations move to the cloud due to its added flexibility and reduced cost. Also, the cloud providers are attracting customers by providing better services and less downtime. Despite all the benefits, common practices like sharing the cloud infrastructure between different tenants and having administrators to access the cloud remotely will inevitably lead to new security threats.

Since “you cannot improve what you cannot measure”, we need solutions for modeling and measuring the security threat related to cloud environments to help cloud providers to understand the threats and improve their services. The National Institute of Standards and Technology (NIST) highlighted the importance of having security models and metrics for securing the cloud [2]. There are two major types of attackers to be considered in a cloud system, which are external attackers[96] and attackers coming from the inside [51]. The external attackers can be considered as any end users who have access to the cloud tenant service, whereas the internal attackers can be cloud administrator, cloud provider’s onsite administrators, and third party administrators (who can perform maintenance tasks remotely).

Modeling and measuring security threats in cloud environments can be challenging due to the lack of publicly available information about cloud configurations. As a result, although there are many existing works related to risk analysis and cloud security including insider attacks and defence, most of those works either stay at a relatively high level of abstraction or are limited to very specific attacks and vulnerabilities. For instance, Saripalli and Walters focus on high-level frameworks for risk and impact assessment [119], and there are also many works that focus on general guidelines or frameworks for devising cloud security metrics [2, 88]. On the other hand, there are many works on very specific threats and vulnerabilities in the cloud [46, 124]. As to insider attacks, most existing works also either stay at a high level or focus on individual nodes instead of the overall infrastructure [29, 85, 133] (Chapter 2 will review related work in more details). Our observation is that there exist a gap between the high level frameworks or guidelines and detailed studies of specific vulnerabilities or attacks.

1.2 Objectives and Contributions

This research aims to present a series of efforts on threats modeling in clouds and network functions virtualization (NFV) in order to better understand the level of security in such systems. We study various threats on the cloud infrastructure and NFV from internal and external attackers, and suggest mitigation solutions to increase security. Throughout this work, we make the following contributions.

– The main contribution for the first research topic is twofold. First, to the best of our knowledge, this is the first comprehensive study of threat modeling that not only covers many well-established models, but also is based on concrete cloud infrastructures incorporating technologies used by major cloud providers. Second, our study can provide insights to many practical questions, such as, *What kind of information could be relevant to the security of cloud infrastructures? How can cloud providers model the security of a cloud data center at different abstraction levels? How can cloud providers measure the security of their cloud data center before and after applying a hardening option?* Such insights can not only benefit cloud providers in understanding and improving the security of their cloud infrastructures but may also embed more confidence in cloud tenants by providing them a clearer picture of potential threats to cloud infrastructures.

– The main contribution of the second research topic is as follows. To the best of our knowledge, this is the first study on the insider threat of remote administrators in cloud infrastructures. As cloud providers leverage third parties for better efficiency and cost saving, our study demonstrates the need to also consider the security impact, and our model provides a way for quantitatively reasoning about the tradeoff between such security impact and other related factors. By formulating the optimization problem of mitigating the insider threat of remote administrators through optimal task assignments, we provide a relatively effective solution, as evidenced by our simulation results, for achieving the optimal tradeoff between security and other constraints using standard optimization techniques.

– The main contribution for the third research topic is twofold. First, to the best of our knowledge, this is the first study on the modeling of security threats in an NFV stack. Our multi-layer resource graph model demonstrates the possibility of novel security threats in NFV, such as cross-layer and co-residency attacks, and also provides a systematic way to capture and quantify such threats. Second, by formulating the optimization problem of mitigating attacks on NFV through optimal VM placement and migration, we provide a low-cost solution to improve the security of NFV using readily available features of cloud and NFV.

1.3 Thesis Organization

The remainder of this thesis is structured as follows. Chapter 2 discusses related literature. In Chapter 3, we present threat modeling for cloud infrastructures. Chapter 4 shows mitigating the insider threat of remote administrators in clouds through maintenance task assignments and Chapter 5 presents modeling and mitigating security threats in network functions virtualization. Chapter 6 provides concluding remarks together with a discussion of future works.

Chapter 2

Literature Review

In this chapter, we provide a review of related literature.

2.1 Attack Graph

Security problems in modern systems are not only caused by single vulnerabilities but by combinations of vulnerabilities between multiple hosts [116]. A precondition for defining security metrics for a cloud environment is to understand how vulnerabilities from hosts may be combined as attack paths. The concept of attack graphs, originally proposed by Phillips and Swiler [110], is a well-established model for this purpose. They built attack graphs based on an attacker profile, a database of common attack templates broken into atomic steps, and a configuration file. This work clearly shows that a system administrator may still face the challenge of defending against system attacks. Many services are safer when they work alone; however, multiple services may help each other to execute exploitable vulnerabilities. Checking the security of each host individually is insufficient for evaluating a system's security; relationships between hosts and vulnerabilities should be considered when measuring security.

Ritchey et al. [116] proposed using model checking to analyze a network's vulnerabilities. The model checker is considered as an attack path–generating system with input information from the network environment. Four main descriptions are chosen as inputs to a model checker to determine whether an attackable path exists. By defining preconditions and goals, model checkers give concrete attack paths for examined network systems. This is the first work that systematically defined attack graphs, which opened up a new era of network security analysis.

Farmer and Spafford [53] described a tool, the Computerized Oracle and Password System COPs, which is used to reconfigure sets of programs and shell scripts to help administrators check for potential security holes in systems. The COPs is used to detect potential security problems and report them to system administrators. Phillips and Swiler [110] presented a graph-based method to analyze network vulnerabilities. This analysis system identifies highly probable attack paths and could be used to test a system when certain changes are applied, e.g., configuration changes or the deployment of an IDS. Zerkle and Levitt [149] presented a useful model checker to help to implement attack graphs and validate network security metrics. Swiler et al. [132] developed a robust attack graph tool with a graphical interface for linking vulnerabilities and configuring attack graphs. Furthermore, Sheyner et al. [125, 126] presented the first formal treatment of attack graphs. They represented the network's state as a collection of Boolean variables, each representing configuration parameters and the attacker's privileges, while the state-transition relations represent the attacker's actions. The security properties required for this network are then evaluated against the model, using a model checker. While this formal approach might prove to be less error-prone than custom-designed algorithms, its efficiency is highly dependent on the semantics chosen and the formal tool used for building the graph. With this particular model, every node contains the entire network state at an attack step, leading to an exponential number of possible states.

2.2 Security Metrics

Security metrics are helpful tools for security analysts in identifying the security risk level within an organization. This identification of security risk can also be used to distinguish the effectiveness of the security programs' components [107]. Metrics make it easier for analysts to answer hard questions raised from higher authorities (e.g., Are we secure enough? How do we compare to others in this regard? Are we more secure than before?).

Metrics can be qualitative or quantitative. Qualitative metrics can be defined as assessment processes that lead to non-numerical values, which are hard to analyze, while quantitative metrics are easy to understand. These are in numerical form and are often used for ranking purposes. Ortalo et al. [103] modeled the minimal effort for an attacker to exploit vulnerabilities as a metric to evaluate systems' security levels. Similar to Ortalo et al.'s work, Balzarotti et al. [21] proposed the minimum efforts required for executing each exploit as a metric. To consider the attack path as a security metric, Pamula et al. [104] proposed a metric to measure networks' security strength in terms of the strength of the weakest attacker (who has the weakest ability to compromise this system). With a customized algorithm, this metric computes minimal sets of required initial attributes for the weakest attacker to successfully compromise a system. These metrics are designed from the attackers' perspective.

The National Institute of Standards and Technology (NIST) underlined the importance of security measurements and metrics for cloud providers by providing high-level definitions and requirements [2]. Following security standards is insufficient to ensure the security of cloud infrastructures, and NFV stacks and security metrics may help to evaluate security levels [26]. Many works have investigated network security metrics in general [108, 140], with some focusing on extending attack trees and attack graphs to security metrics [141, 137]. Some other works have focused on modeling known vulnerabilities for network security [142, 9], and other works focus on modeling unknown vulnerabilities (zero-day attacks) [138, 144, 150, 139], which are usually considered unmeasurable due to the uncertainties involved [94]. Furthermore, Luna et al. proposed a framework with basic building blocks for cloud security metrics [88]. We loosely follow this framework in this work. Halabi and Bellqich used goal–question–metric to develop a quantitative evaluation metric with which to help cloud providers evaluate their cloud security service and determine the level of security [65]. Edge et al. presented a protection tree [52], which is similar to attack trees but contain information on how the system can be secured. Our work borrows from this work to apply the attack tree-based metric.

Wang et al. [137] proposed aggregating vulnerability probabilities from the Common Vulnerability Scoring System (CVSS). The probability of a vulnerability indicates the successful exploitation rates for a given vulnerability. However, this work focuses only on vulnerabilities while ignoring the interdependencies inside attack graphs. To address this limitation, a Bayesian network (BN)–based security metric applies attack graphs (resource graphs) to measure a network's security levels [57]; the metric converts the CVSS scores of vulnerabilities into attack probabilities and then obtains the overall attack likelihood for reaching critical assets. We apply this metric to measure cloud data center infrastructure, insider threats in the cloud, and threats in the NFV stack. Security metrics and measurements in clouds and NFV still face many challenges, as several authors have shown [33, 87, 25, 68, 98]. Branco and Santos stated that no regulatory bodies are in place to monitor cloud service providers and that cloud service providers mainly only guarantee quality of service and availability to SLAs, which is clearly not enough. It is clear that the contracts with cloud service providers must be more transparent and specific to clarify the security issues and to define the pertinent responsibilities in the providers' business relationship with their clients [33]. Bayuk introduced a methodology to help identify the required security metrics in the cloud. This methodology has three steps: first, identifying security features that require system-level functions; second, evaluating the extent to which security features protect systems from deliberate damage that would cause system failure; third, devising verification and validation metrics at the system level to show that security requirements are met [25].

Some works have focused on risk assessment for the cloud. Saripalli and Walters showed a framework with which to evaluate the security of clouds based on the security impacts on six security categories related to the cloud: confidentiality, integrity, availability, multi-party trust, mutual auditability, and usability (CIAMAU), according to three abstract levels of security impact: low, medium, and high [119]. Le and Hoang used Markov chains and the common vulnerability scoring system to compute the probability of security threats on cloud infrastructure [82]. However, the existing works are missing the security impacts of the NFV, which takes advantage of cloud-based infrastructure.

Security metrics are also developed for specific applications, such as information-theoretic metrics for measuring the effectiveness of IDSs [63, 84]. Gu et al. [63] built a formal framework using information theory for analyzing and quantifying the effectiveness of IDSs. This work presents a formal IDS model to analyze the information-theoretic approach. Similar works in other areas exist; for example, some design principles have been proposed for developing metrics of trust [114, 115]. Authenticating entities in large-scale systems always use authentication, which means each entity can authenticate the next path. This technology has been extended to multiple paths [114, 115]. Reiter et al. in [114, 115] proposed metrics to evaluate the confidence afforded by a set of paths. These two papers illustrate the general principles for designing security metrics.

2.3 Threat Modeling for Clouds

Cloud environments may usually be subject to more security threats compared to traditional enterprise networks, and many of such threats come from exploiting existing vulnerabilities in the cloud [61]. Security issues in the cloud data center are similar to those in the traditional data center, but there are unique issues related to the cloud [39]. Chen et al. discussed multi-party trust and mutual auditability unique to the cloud [39]. Threat modeling can help to understand issues like from where the attacker can start the attack, and what consequences an attack may cause to the cloud data center [119]. Saripalli and Walters present a framework called QUIRC for analyzing and assessing the risks and impacts to the security of cloud. They define risk impact based on six categories, which are confidentiality, integrity, availability, mutual, auditability and usability [119]. Ingalsbe et al. present a threat model that cloud tenants can use to evaluate the system [72]. The authors adopt an Enterprise Threat Modeling methodology, which classifies all components related to the cloud tenant under three categories (Actor, End Points, and Infrastructure). However, the authors do not provide concrete case studies detailing how such a threat model might be used. Gruschka and Jensen apply the attack surface concept to provide classifications for attacks in a cloud [62]. The authors identify three main entities (User, Cloud provider, and Service) and the attack surfaces between those entities. The authors provide high-level examples of attacks but do not mention specific services or vulnerabilities underlying each attack surface. We borrow this classification in applying attack surface. The original attack surface concept [91] is intended to measure the security of a software system focusing on identifying entry/exit points, communication channels, and untrusted data items from the source code. Like most existing work, our work applies those concepts of attack surface but at a higher abstraction level. An attack tree is a well-known threat model which can be used for many useful analyses, such as analyzing the relative cost of attacks and the impact of one or more attack vectors [123]. Attack trees can also be used in security hardening to determine the best options to increase security within a budget [48]. Using attack trees can help to understand what kind of attackers may follow an attack path [123, 113, 13]. Attack graphs can be automatically generated by modeling the network and vulnerabilities, and many useful analyses may be performed using attack graphs [125, 143, 142]. We borrow the concepts of attack trees and attack graphs but study their particular application to cloud data center infrastructures.

Network hardening is a process or procedure based on a vast range of techniques, strategies, and methodologies used to evade multiple attacks or threats. From the work undertaken on network-hardening models, the most common modeling techniques used to derive a solution for network hardening involve attack graphs [141, 99, 37] or attack trees [47]. Furthermore, network hardening is applied based on the theory of game-based models [77], in which a game is played between a defender and an attacker to effectively analyze each player's strengths and weaknesses. Wang et al. [141] used network hardening with attack graphs to measure the safety of given critical assets with the minimum overall costs. Chen et al. [37] converted attack graphs to reduced ordered binary decision diagram (ROBDD) to compute the lowest-cost network-hardening solutions. Some other works use game theory. Jiang et al. [77] used an attack–defense game model by focusing on the attacker's strategy from an attack graph. Dewri et al. [47] used attack trees with multiple objective optimization to measure risk and costs.

Finally, some works focus on risk assessment for the cloud. Cayirci et al. used risk assessment to help cloud tenants choose cloud providers to meet their security requirements [35]. This model is based on the background information collected from tenants and cloud providers. Risk assessment is also used to ensure that no violations of the service-level agreement (SLA) exist related to the provider and tenant during the run time [49, 109]. Maglaras et al. discussed the directive on the security of network information system (NIS) through a case study of Greece [90] in which it is mentioned that, in order to create an IT and security inventory, information needs to be collected about critical infrastructures including governmental clouds in order to reveal vulnerabilities and lack of security measures. This provides an interesting use case for applying the threat-modeling solutions discussed in this topic.

2.4 Insider Threat of Remote Administrators in Clouds

The insider threat is a challenging issue for both traditional networks and clouds. Ray and Poolsapassit proposed an alarm system to monitor the behavior of malicious insiders using attack trees [113]. Their approach provides the system administrator with an estimated attack probability, and the system enters an alert mode once the probability of an attack is determined to be sufficiently strong. Mathew et al. used capability-acquisition graphs (CAG) to monitor the abuse of privileges by malicious insiders [93]. Sarkar et al. proposed DASAI to analyze whether a process contains a step that meets the conditions for an insider attack [120]. Chinchani et al. proposed a graph-based model for insider attacks and to measure the threat [40]. They presented an assessment theory that models the user's view of the organization and of key information and capabilities. Althebyan and Panda proposed a model to predict and prevent insider threats. This model focuses on two components to develop their model, the insider's knowledge, and existing dependencies among objects in the system. Their model limits the possibilities for the insider to gain accesses to documents and get sensitive information from the organization [15]. Bishop et al. [28, 27] proposed a progressive notion of insiderness. They introduced a hierarchy of policy abstractions and argued that the discrepancies between the different layers of abstraction are useful for identifying insider threats. They also presented a methodology for analyzing the threat based upon these definitions. They introduced attribute-based group access control, which allows any attributes to define a group. The attributes that they studied included job function (e.g., system administrator) and building access (e.g., before 5 P.M.). They applied this model to insider threats by defining groups based on access capabilities and using those groups to identify users with a high level of threat with respect to high-risk resources. Roy et al. [117] used the RBAC model for a commercial information system and focused on the problem of employee assignment. The objective was to identify an employee-to-role assignment that would permit the maximum flexibility in assigning a task by using 0-1 integer linear programming. To determine the optimum assignment, the model assumed that the set of roles and employees in an organization was completely known.

Limited efforts have been made on studying insider attacks in the context of clouds. Our previous work focused on applying different threat-modeling techniques to cloud data center infrastructures, focusing on external attackers [10]. Gruschka and Jensen devised a high-level attack-surface framework to show where attacks can begin [62]. Resource graphs can be automatically generated by modeling the network and vulnerabilities, and many useful analyses can be performed using resource graphs [125, 143, 142, 16]; however, our work is the first to use resource graphs for modeling insider attacks.

The proactive mitigation of security threats can be performed through network hardening. Early works on network hardening focus on breaking all the attack paths that an attacker can follow to compromise a critical asset, either in the middle of the paths or at the beginning (disabling initial conditions) [141, 125, 136]. Network hardening using optimization is proposed by Gupta et al. in [64], refined with multiple objective optimization by Dewri et al. in [47] and with dynamic conditions by Poolsappasit et al. in [111], and extended as vulnerability analysis with cost/benefit assessment [48] and risk assessment [146]. More recent works [30, 31] focus on combining multiple hardening options through optimization, and improving the diversity of networks, respectively. We borrow the optimization-based hardening techniques [64, 30] to mitigate the insider threats in this topic. In the context of clouds, there are works on securing the cloud from insider attacks by limiting the trust on the compute node [133]. Li et al. focus on supporting users to configure privacy protection in compute nodes [85]. Closest to our work, Bleikertz et al. focus on securing the cloud during maintenance time by limiting the privileges granted to the remote administrators based on the tasks assigned to that administrator [29].

There are many works that focus on the service dependency. Some of those existing works focus on the mission impact which relies on the service dependency model to capture the threats that can impact a mission [76, 122]. Chen et al. [38] show how the service dependency can affect the system since compromising a service can affect other services hosted in same network. Natarajan et al. [97] develop a tool for automatically finding dependency between services in large networks. Another work develops different techniques to monitor service dependencies in distributed systems [71]. Closest to our work, Sun et al. [131] embed mission impact and service dependency in an attack graph to find how service dependency can impact different missions. We borrow their model of service dependency in evaluating the insider threats of remote administrators.

2.5 Threats in NFV

To the best of our knowledge, there only exist very limited effort on threat modeling specifically for NFV environments. As already mentioned earlier, most existing works focus on the underlying cloud infrastructures. In particular, our previous work applies different threat modeling techniques to cloud data center infrastructures for different types of attackers [12]. Gruschka and Jensen devise a high level attack surface framework to show from where the attack can start [62]. There exist other works focusing on insider threats in clouds [40, 11]. Chinchani et al. proposed a graph-based model for insider attacks and measure the threat [40]. There are many works that focus on the co-residency attacks by improving the placement policy. Han et al. introduce a new strategy to prevent attackers from achieving co-residency by modifying the placement policy on CloudSim [66]. Madi et al. propose a quantitative model and security metric for multi-tenancy in the cloud at different layers [89]. Atya et al. study co-residency in clouds and suggest solutions for the victim tenant to avoid co-residency with malicious users [18].

Unlike our work which focuses on the cross-layer and co-resident threats and the application of threat models and security metric, existing studies on NFV security [81, 106, 148, 55] mostly focus on issues related to virtualization. Lal et al. [81] propose to adapt several well-known best practices like VM separation, hypervisor introspection, and remote attestation to NFV. Pattaranantakul et al. [106] adapt best practices like access control to address virtualization-related threats in NFV. Our cross-layer resource graph model is partially inspired by existing works [130, 131] in which Sun et al. use a cross-layer Bayesian network to measure security threats for enterprise networks [130] and additionally they employ a multi-layer attack graph to measure security in clouds [131].

Many works exist on network-slicing security in general [102, 79]. Ordonez-Lucena et al. [102] provided a comprehensive study of the architectural frameworks of both SDN and NFV as key enablers to realize network slices as well as of their challenges. Some existing works focus on intra-slice security by using pseudo-random number generators to protect private information [32]. Sattar and Matrawy [121] addressed two challenges of network slicing, namely the isolation between slices and satisfying the end-to-end delay. They used intra-slice isolation to run VNFs on different physical machines to increase the reliability. However, those works did not cover interslice attacks, which is an important threat in multi-tenant environments. We cover both interslice and intraslice attacks in our work. Wang et al. [145] discussed the security challenges on the physical layer for 5G networks for IoT slices; however, we cover threats on all layers on the NFV stack in our work.

There also exist related works on other aspects of NFV security. Firoozjaei et al. [55] showed how multi-tenancy and live migration can affect the security on NFV by using a side-channel and shared resource misuse attack. Alnaim et al. [14] used architectural modeling to analyze security threats and possible mitigating solutions for NFV. However, their model is relatively abstract and only considers a malicious tenant to exploit vulnerabili ties when he/she co-resides with the target VM on the same physical machine. Tian et al. proposed a framework using a hierarchical attack-and-defense model that divides the 5G network into four layers (the physical layer, virtual layer, service layer, and application layer) [134]. Basile et al. [23] proposed adding a new policy-manager component to enforce security policies when deploying and configuring security functions. Coughlin et al. [44] integrated trusted computing solutions based on Intel SGX to enforce privacy with secure packet processing.

Chapter 3

Threat Modeling for Cloud Infrastructures

3.1 Introduction

Cloud computing has emerged as an alternative IT solution for many enterprises, governm-ent agencies, and organizations due to its flexibility and reduced costs. The shifting to this new paradigm, however, might still be impeded by various security and privacy concerns of the cloud tenants, especially considering the lack of transparency in the underlying cloud infrastructures. In contrast to traditional enterprise networks, the increased complexity of cloud infrastructures implies that security flaws may still be present and undetected despite all the security solutions deployed inside the cloud; moreover, the complexity may also lead to new challenges in systematically understanding the potential security threats. For instance, unlike traditional enterprise networks, cloud data centers usually exhibit unique characteristics including the presence of significant similarity in terms of hardware configurations (e.g., server blades inside a rack), and the co-existence of both physical and virtual components. Such unique characteristics may imply novel challenges and opportunities in applying existing threat modeling techniques to cloud infrastructures, which motivates our study.

On the other hand, modeling security threats for cloud infrastructures also faces a practical challenge, i.e., there lack public accesses to detailed information regarding hardware and software configurations deployed in real cloud data centers. Existing work mainly focuses on either high-level frameworks for risk and impact assessment [119] and general guidelines for cloud security metrics [2, 88], or very specific vulnerabilities or threats in the cloud [46, 124]. To the best of our knowledge, there lacks a concrete study on threat modeling for cloud data centers using realistic cloud infrastructures and well-established models. Although there already exist a number of threat modeling models, such as attack surface, attack tree, attack graph, and various security metrics, a systematic application of those models to concrete cloud infrastructures is yet to be seen.

In this topic, we present a comprehensive study on applying threat modeling techniques to cloud infrastructures. We first provide the basis of our study as two representative cloud infrastructures. Those infrastructures are devised based on fictitious but realistic cloud data centers by integrating established technologies of several major players in the cloud market, e.g., Amazon, Microsoft, Google, Cisco, VMware, and OpenStack. We provide details on the hardware and software components used in the data center to manage the cloud services, such that the infrastructures may facilitate our later application of threat models at different abstraction levels (e.g., while attack surface and trees focus on hardware and software components, attack graphs involve lower-level details including specific vulnerabilities in those components). We then apply several popular threat modeling methods on such cloud infrastructures, including attack surface, attack tree, attack graph, and security metrics based on attack trees and attack graphs. Furthermore, we discuss the application of network hardening solutions for improving the security based on the threat modeling results. During the application of those models, we discuss detailed results and challenges as well as general lessons that can be taken based on those exercises.

3.2 Background

This section briefly reviews several popular threat models and existing security metrics that will be applied in this chapter, including attack surface, attack tree, attack graph, attack tree-based metric (ATM), and Bayesian network (BN)-based metric.

– Attack surface: Originally proposed as a metric for software security, attack surface captures software components that may lead to potential vulnerabilities, including entry and exit points (i.e., methods in a software program that either take user inputs or generate outputs), communication channels (e.g., TCP or UDP), and untrusted data items (e.g., configuration files or registry keys read by the software) [91]. Since attack surface requires examining the source code of a software, due to the complexity of such a task, most existing work applies the concept in a high-level and intuitive manner. For example, six attack surfaces are said to exist between an end user, the cloud provider, and cloud services [62], although the exact meaning of such attack surface is not specified.

– Attack tree: While attack surface focuses on what may provide attackers initial privileges or accesses to a system, attack trees demonstrate the possible attack paths which may be followed by the attacker to further infiltrate the system [123]. Figure 1 shows an attack tree example in which the attackers goal is to get accesses to the database. In the example, there are two ways to reach the root node (the goal). First, the attacker can follow the left and middle paths at the same time (due to the *and* label), or the attacker can follow the right path for reaching the root node.

– Attack graph: As a more fine-grained model, an attack graph depicts all possible attack steps and their causal relationships [125]. In Figure 2, each triplet inside a rectangle indicates an exploit <service vulnerability, source host, destination host>, and each pair in plaintext indicates a pre- or post-condition <condition, host> of the

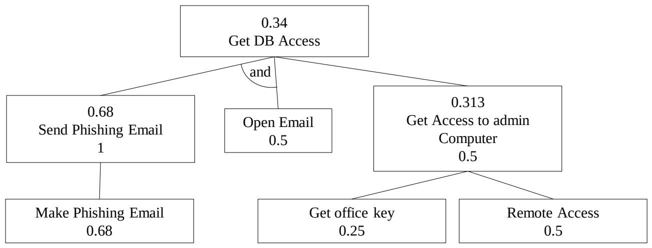


Figure 1: Examples of attack tree

exploits. The logic relationships between the nodes are represented as edges, where an exploit can be executed if and only if all of its pre-conditions are already satisfied (e.g., In Figure 2, the first exploit requires all three pre-conditions to be satisfied), whereas a condition may be satisfied by one exploit for which the former is a post-condition.

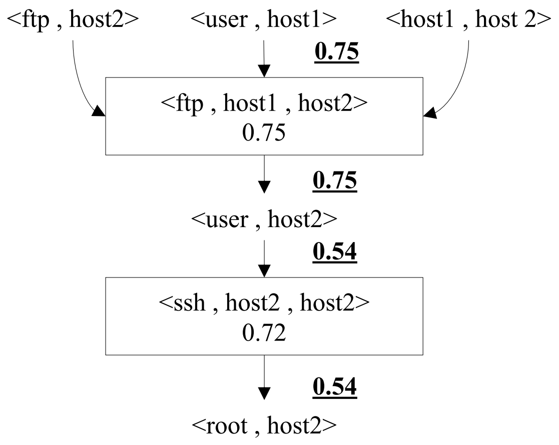


Figure 2: Examples of attack graph

– while the above threat models are all qualitative in nature, they may be extended to quantitatively measure the level of security. The attack tree-based metric (ATM) quantifies the threat in an attack tree using the concept of *probability of success* [52]. The probability of each node in the attack tree is typically determined based on historical data, expert opinions, or both. In Figure 1, a number above the label represents the overall probability of success, and a number below the label represents the probability of each node alone. The probability on the root node indicates the most risky path, which should be prioritized in security hardening. The BN-based metric [137, 57] can be applied to attack graphs to calculate the probability for an average attacker to compromise a critical asset. The conditional probabilities that an exploit can be executed given its pre-conditions are all satisfied can usually be estimated based on standard vulnerability scores (e.g., the CVSS scores [95]). In CVSS, each vulnerability is assigned a base score ranging from 0 to 10, based on two groups of totally six base metrics [95, 58]. The first group is the exploitability metric to measure access vector, access complexity and authentication. The second group is the impact metric to measure confidentiality, integrity and availability. In Figure 1, the probability inside a rectangle is the CVSS score divided by 10 (the domain size of those scores) [69], and each underlined number represents the probability for successfully executing that exploit. In this example, the attack goal has a probability of 0.54, and if we change the ftp service on host 2 and suppose the new probability becomes 0.4, then the new attack probability for the goal will become 0.228, indicating increased security.

The aforementioned threat models are mostly designed for traditional networks and not specific to cloud infrastructures. While a cloud data center can also be regarded as a large and complex computer network, the network may have some unique characteristics especially regarding threat modeling, such as the existence of both physical and virtual components, the existence of many different types of users (e.g., cloud users, cloud tenants, administrators of the cloud, administrators of the tenants, cloud operators, etc.), the existence of a large number of hardware components with similar configurations (e.g., server blades in a rack), and the multi-tenancy nature of cloud. To understand how those characteristics may affect the application of existing threat modeling techniques when applied to cloud infrastructures, we will apply them to two representative cloud infrastructure in the remainder of the chapter.

3.3 Devising Cloud Infrastructures

In this section, we devise two cloud data center infrastructures that will be used later for threat modeling.

3.3.1 Overview

As we have seen in Section 3.2, threat modeling usually requires detailed information regarding hardware and software components and their configurations, e.g., attack graphs contain information about specific vulnerabilities on each host and the causal relationships between such vulnerabilities. However, there lack public accesses to such detailed information for real cloud data centers, which is understandable since cloud providers would be reluctant to disclose details about their infrastructures and especially the vulnerabilities. To address this challenge, we devise fictitious but realistic cloud infrastructures based on concepts and ideas borrowed from major players on the market, including Cisco, VMware, and OpenStack. The following provides some examples.

- Cisco presents a cloud data center design for both public and private clouds [19], which is divided into multiple layers with suggested hardware for the physical network and software used to virtualize the resources. Our infrastructures borrow the multi-layer concept and some hardware components, e.g., Carrier Routing System (CRS), Nexus (7000,5000,2000), Catalyst 6500, and MDS 9000.
- VMware vSphere provides recommendations for the hardware and software components required to run a private cloud data center [67]. They also tag the port

numbers used to connect services together. Our infrastructures borrow the concepts of Authentication Server, Domain Name System(DNS), and Storage Area Network (SAN), which are synthesized to represent the main functionality of some hardware components in our cloud infrastructures.

– OpenStack is one of the most popular open source cloud operating systems [101]. Our infrastructures relies on OpenStack and particularly its following components: Dashboard, Nova, Neutron, Keystone, Cinder, Swift, Glance, and Ceilometer [101].

Table 1 relates some of the concepts used in our infrastructures to those found in the three major cloud providers[22, 128, 7] (some of those concepts will also be discussed later in this section). By incorporating those popular concepts and hardware/software components shared by major players in the market, we ensure our design is representative such that the threat modeling exercises later can bring out useful lessons for cloud providers even though their cloud infrastructures will certainly be different from ours. Also, we assume hardware and software components of specific versions, which are carefully designed in such a way that those components (and their specific versions) correspond to various real world vulnerabilities that will later be used in our threat modeling exercises. In the following, we discuss two different infrastructures since OpenStack components can either run centrally on a single server or be distributed to multiple servers [101].

Table 1: Concepts used by major cloud providers such as Amazon web services (AWS), Microsoft Azure (MA) and Google compute engine (GCE).

	AWS [22]	MA [128]	GCE [7]
Multiple layers	✓	✓	✓
Authentication Server	✓	✓	
Domain Name System	✓	✓	✓
One service in each cluster	✓	✓	✓
Multi-tier	✓	✓	✓

3.3.2 Infrastructure 1

Figure 3 illustrates our first infrastructure. The physical network provides accesses to both cloud users and cloud administrators, e.g., cloud administrators can connect to the data center through firewalls (node 17) and (node 19), an authentication server (node 18), and a Nexus 7000 (node 20), which is connected to the other part of the network. For cloud users, Cisco's multi-layer concept is used [19] as follows.

- In Layer 1, a CRS (node 1) is used to connect the cloud to the internet, which then connects to a firewall (node 2, ASA 5500-X Series) while simultaneously being connected to two different types of servers (authentication servers (node 3) as well as DNS and Neutron Servers (node 4)). Those servers provide services to the cloud tenants and end users. The servers then connect to Cisco Nexus 7000 with Catalyst 6500 (node 5) to route the requests to destination machines.

- In Layer 2, a firewall (node 6, ASA 5500-X Series) connects the first layer to this layer through Nexus 5000 (node 7). The Nexus 5000 is used to connect rack servers through Nexus 2000, which is used to connect servers inside each rack at the computing level (nodes 8,9,10,11, and 12). The Nexus 5000 (node 7) then connects to the next layer.

- In Layer 3, another Nexus 7000 (node 13) connects the previous layer to the storage. A firewall (node 14, ASA 5500-X Series) connects the Nexus 7000 (node 13) and MDS 9000 (node 16).

The following outlines how the cloud works. OpenStack components run on the authentication servers among which one (node 3) is designated for cloud tenants, and another (node 18) for cloud administrators. The first runs following components: Dashboard, Nova, Neutron, Keystone, Cinder, Swift, Glance, and MySql. The second runs the same components, but additionally runs Ceilometer for a billing system. The DNS server (node 4) runs a Neutron component that provides the address of the machine running a requested service. At the computing level (nodes 8,9,10,11, and 12), all physical servers run four components: Hypervisor, Nova to host and manage VMs, Neutron agent to connect VMs to the network, and Ceilometer agent to calculate the usage. At the computing level, each physical server cluster runs the same VMs service [127], e.g., all *http* VMs run on the *http* server cluster, and the same occurs for application VMs, *ftp* VMs, smtp VMs, and database VMs. Finally, all physical machines and VMs run *ssh* for maintenance.

3.3.3 Infrastructure 2

The second infrastructure is illustrated in Figure 4. This infrastructure has a similar physical network as the previous, with the addition of new machines that separate OpenStack components, which are installed on the authentication servers for cloud tenants in the previous infrastructure, into different machines. These new machines are Neutron servers (node 25), controller servers (node 36), and network nodes (node 34). In addition, the authentication server (node 23) for cloud tenants will run a Dashboard component to access and manage the VMs related to the tenant user. Moreover, Neutron server (node 25) controls the virtual network and connects to the controller node (node 36), which runs Nova API, Neutron API, Keystone, Glance, Swift, Cinder, MySql, and any other components needed to manage and control the cloud. Finally, a network node (node 34) translates between the virtual IPs and the physical IPs to grant accesses to services running on VMs. For example, if a cloud tenant wishes to access their VMs, they will first need to connect to the Dashboard. Next, the Neutron server will send the authentication request to the keystone service on the controller node. If the user possesses the privilege for accessing the VMs, the controller will send a request to the network node to obtain the address for the VMs, and will then send the address to the Neutron server to connect the user to their VMs.

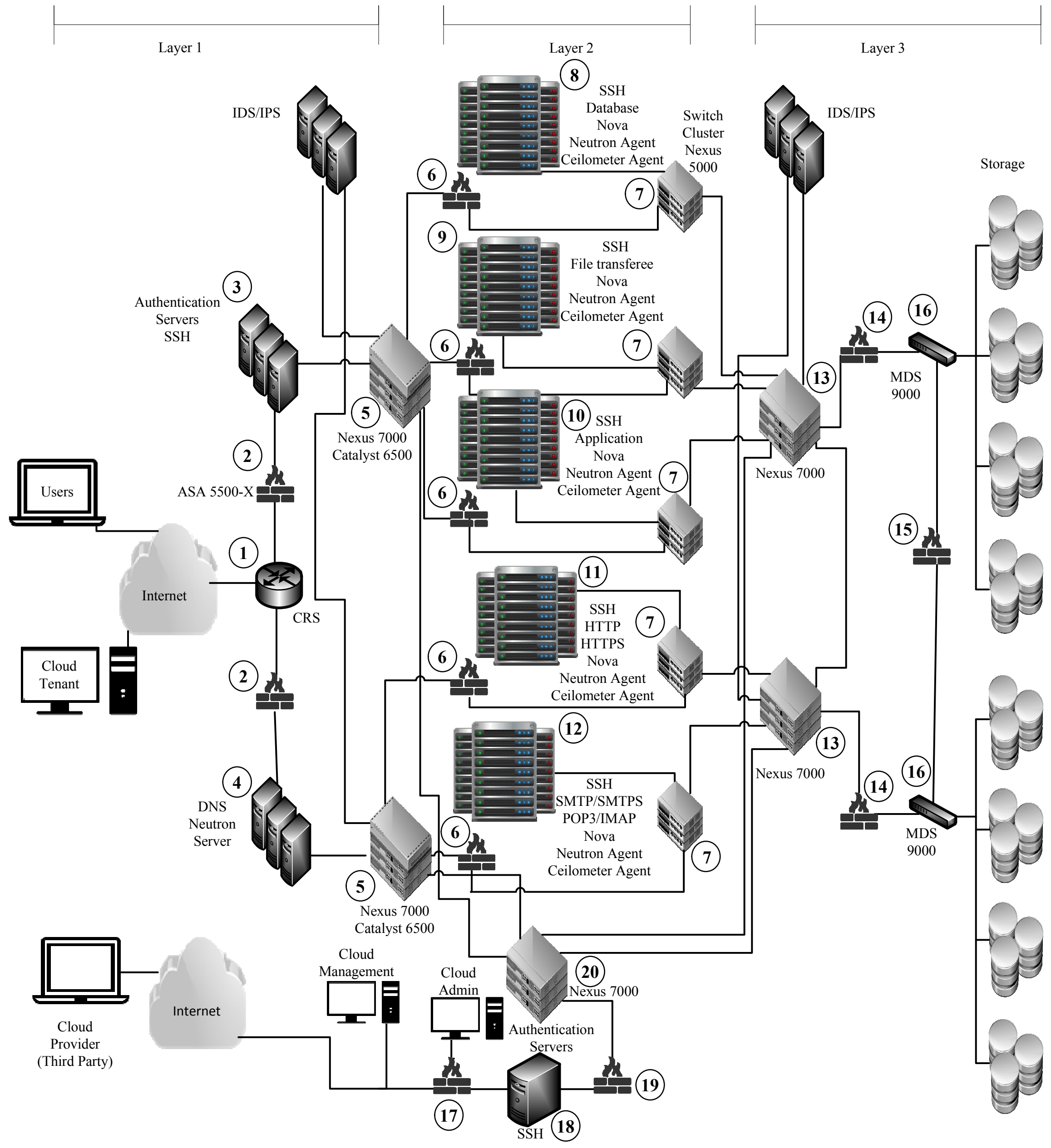


Figure 3: Cloud data center infrastructure 1

In the remainder of the chapter, we will apply several threat modeling techniques to those cloud infrastructures. Also, those threat models may apply by different clouds because the general way of applying those threat models as well as the unique challenges (such as lots of identical servers, and the co-existence of physical/virtual components) should still apply there. In addition to the details about the hardware/software components and configurations provided above, we will introduce additional assumptions, e.g., those about vulnerabilities, during the discussions of each model.

3.4 Threat Modeling

This section applies several popular threat models, including attack surface, attack tree, and attack graph, to the two cloud infrastructures introduced above.

3.4.1 Attack Surface

We apply the attack surface concept to our cloud infrastructures at the level of hardware and software resources. Gruschka and Jensen categorize attack surfaces into those between users, services, and the cloud provider [62]. The same classes are used in our discussions, with the addition of attack surfaces belonging to each class. Also, we consider the service class as the intermediate layer between users (either end users and cloud tenants) and the cloud provider (or cloud operators) in the sense that, if a user wishes to attack a cloud provider or another user, he/she must pass through an attack surface consisting of services. In addition, we focus on entry and exit points [91] which indicate the means through which the attack starts, and those through which data is leaked out, respectively.

In Figures 3 and 4, it can be observed that there are three types of attack surfaces in a cloud data center. First, there are attack surfaces related to the physical network, involving hardware and software components, such as switches, routers, servers, applications, and operating systems. Second, there are virtualization-related attack surfaces, such as hypervisors and virtual switches. Third, there are attack surfaces related to the cloud operating systems, such as OpenStack components (Glance, Neutron, Nova, Ceilometer, and Keystone). The first type of attack surface is similar to those in traditional networks

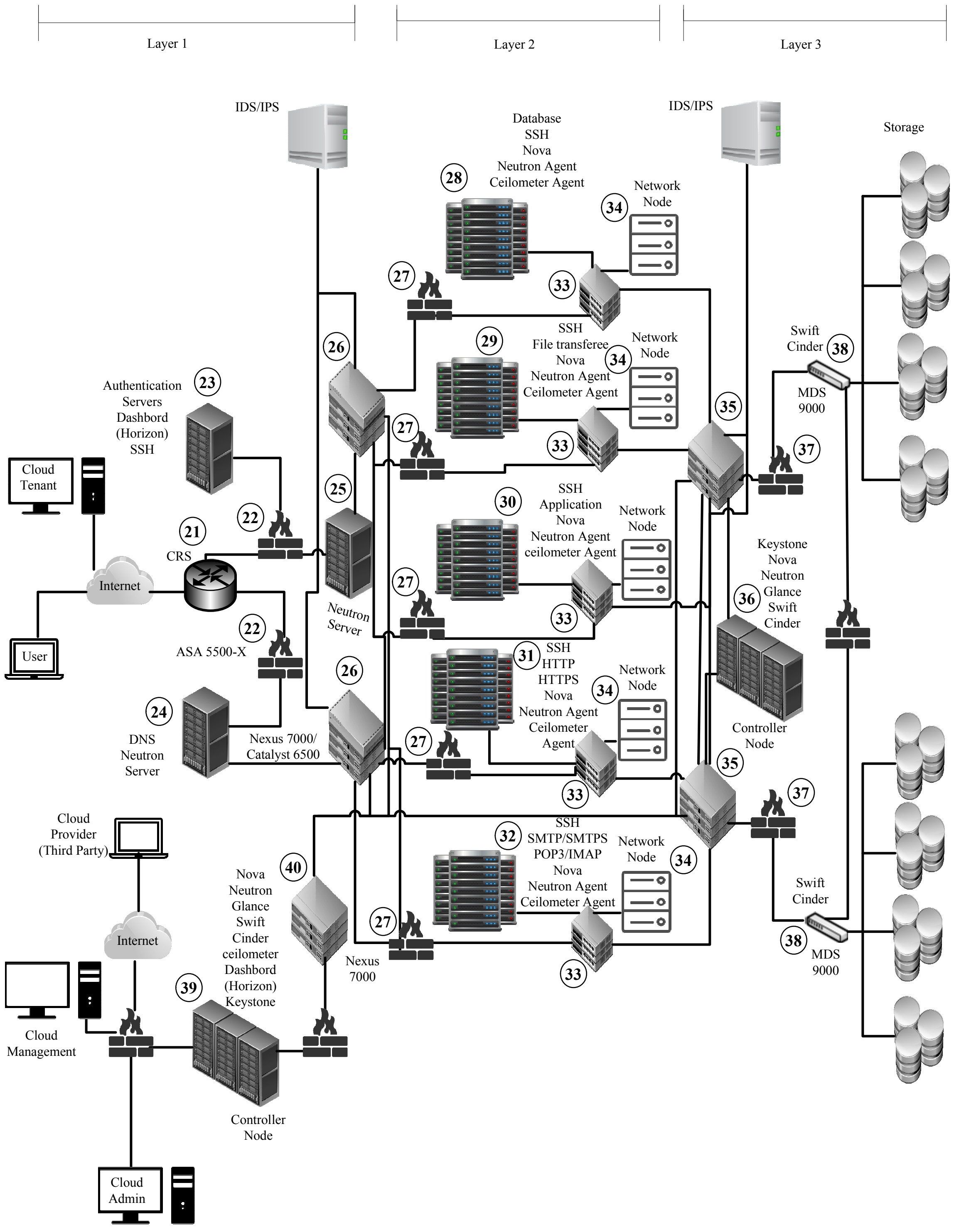


Figure 4: Cloud data center infrastructure 2

except that software components may exist both at the infrastructure level and in virtual machines or virtual networks. On the other hand, virtualization and cloud operating systems-related attack surfaces are unique to a cloud, and their analysis will pose new challenges. Figure 5 illustrates the entry points that can be used by end users, cloud tenants, and cloud operators, respectively.

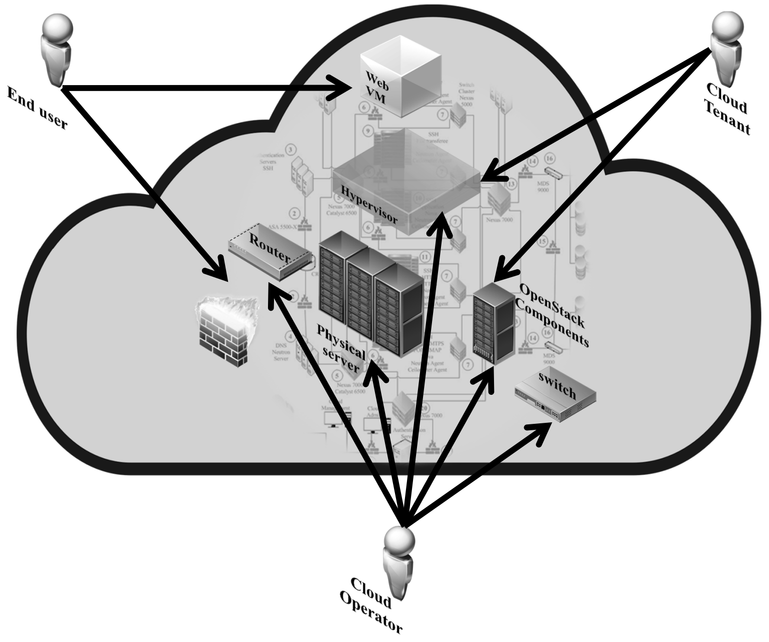


Figure 5: Entry point for attack surface

Attack Surface w.r.t. End Users

We consider an adversary taking the role of an end user who can only access some cloud services over the Internet, but is not part of any cloud tenant. Assume the malicious user wants to reach a database server and attack a hypervisor to control all VMs run on that machine. The following discusses two example scenarios to show which attack surfaces may be involved when the malicious user attempts to reach his/her goal.

Example 1. *An example entry point for the end user to start the attack is the* $http$ *VM (node 11) running in the* $http$ *tier which may have a vulnerability inside the services (*$http$ *or ssh) (note attack surface is not directly concerned with specific vulnerabilities). After he/she gets access to the* $http$ *VM, it becomes an exit point to attack the* app *VM (node 10) running in the* app *tier. By exploiting a vulnerability, e.g., in the Oracle application, the attacker can turn the* app *VM into an exit point to attack the database (node 8). By exploiting a vulnerability in the DB VM, he/she can make it an exit point to reach the database hypervisor. Finally, by exploiting a vulnerability in the hypervisor, e.g., CVE-2013-4344 [1], the attacker can potentially obtain control over all VMs running on top of this hypervisor and turn the hypervisor into an exit point to reach data belonging to all those VMs. This example shows how different hardware/software components may become part of the attack surface (entry points and exit points) along a path followed by the attacker, which also motivates us to better capture such a path using other threat models later, such as attack trees or attack graphs.*

Example 2. *This example shows a slightly different attack surfaces that can potentially be used by the malicious end user to reach the same goal. The entry point is the same as the previous example, the* $http$ *VM (node 11). After the end user gets access, he/she can use that VM as an exit point to attack the hypervisor running in this VM and make the hypervisor an exist point to attack other* $http$ *VMs running on the same hypervisor, which will be similar to the previous example, or to attack the physical machine. After getting access to the physical server (node 11), the attacker can turn it into an exit point to attack other physical machines in the same tier, or to attack the next tier, e.g., the* app *server (node 10) followed by attacks on the database server (node 8), and he/she can reaches his/her goal in a similar fashion as above. Comparing this example to the previous one, we can see that the co-existence of physical and virtual components enlarges the attack surface in cloud infrastructures, which potentially gives attackers more choices in reaching* *a goal.*

Attack Surface w.r.t. Cloud Tenants

We consider an adversary taking the role of a legitimate cloud tenant who can use his/her own VMs to attack another tenant who reside in the same physical machines or in the same cloud data center. We will discuss two examples related to such a cloud tenant adversary. The first example shows the attack surface used to attack other tenants co-residing on the same physical machines. The second example shows the attack surface for attacking other tenants in the same cloud data center.

Example 3. *Suppose a malicious cloud tenant wants to attack another tenant residing on the same physical machine. Unlike the end users, the malicious tenant dose not need to find an entry point among the cloud services to start his/her attack as he/she has access to VMs running inside the cloud. Assume the cloud tenant in this example has access to the* ftp *VM (node 9). The malicious tenant may use vulnerabilities related to the hypervisor as an entry point to gain access to the hypervisor. Once he/she gets such accesses, the hypervisor becomes an exit point to access any other tenants' VMs running on top of the same hypervisor. This example clearly shows the tenant privilege gives attackers an edge over the end users in the previous examples.*

Example 4. *Now consider a slightly different scenario where the malicious cloud tenant wants to attack another tenant not residing on the same hosts, but still inside the same cloud data center. Assume the malicious tenant has access to the* ftp *(node 9), so he/she can use that VM as an exit point to attack the hypervisor, and that compromised hypervisor then in turn becomes an exist entry point to attack the physical machine. Once in control of the physical machine, that machine becomes an exit point to attack the switches (e.g., node 7), which then becomes an exit point to attack other physical machine in the* ftp *tier (e.g., node 9) or the switches (node 13), and eventually leading to access to the storage (node*

16). In those two examples, we can see that, for malicious tenants, the hypervisors are almost always the foremost and also the most important attack surface during cross-tenant attacks.

Attack Surface w.r.t. Cloud Operators

A cloud operator here refers to an employee of the cloud provider who has limited privileges to access specific components (e.g., switches, firewall, and SAN) for maintenance and management purposes. An adversary taking the role of such a cloud operator may abuse his/her accesses to resources to attack the cloud data center. The cloud operators may further be divided into two categories, the local employees of the cloud provider, and those who are from a third party company under a contract with the cloud provider. We use two examples to show the attack surface corresponding to each category.

Example 5. *Suppose the malicious operator wants to steal data belonging to cloud tenants. Specifically, assume the operator has access to switch Nexus 5000 (node 7) to perform maintenance task and his/her goal is to steal data from storage (node 16). The malicious operator can use switch Nexus 5000 as an exit point to attack switch Nexus 7000 (node 13), which then becomes an exit point to reach the firewall (node 14). The firewall (node 14) then becomes an exit point to reach the MDS 9000 (node 16), which in turn becomes an exit point to access data stored in the cloud. Clearly, this example shows that a malicious operator would have a much larger attack surface than in all previous cases, which will enable him/her to simply bypass any cloud services or hypervisors and attack directly the critical hardware components.*

Example 6. *In this example, a third party operator is given remote access to perform maintenance tasks on the compute node (node 12), and the target is to get access to emails belonging to the tenants. The malicious operator can use his/her access to the compute node to attack its operating system, and hence he/she can make the compute node an exit* *point to attack the hypervisor and VMs running on the same machine, eventually leading him/her to access the email service. In contrast to the above example, although the third party operator in this case has slightly lower privileges (i.e., not directly accessing the hardware components), there is still a possibility he/she may abuse his/her initial privileges on an important attack surface (the compute node).*

Summary The attack surface we have applied above is a high level model that indicates the resources initially accessible (and thus can be attacked) to an attacker. The above examples show that, since cloud may have many different types of users with different initial privileges, a defender must consider many different attack surfaces as well. For instance, for a malicious end user, the initial attack surface generally includes the cloud services, and the once a cloud service is compromised, the attacker will gain access to the VM and becomes an adversary similar to cloud tenants. The increased privileges of cloud tenants and cloud operators give them a larger attack surface due to their legitimate access to VMs or hardware components. For cloud tenants, the hypervisor is generally the first attack surface, and also the isolation provided by hypervisors is the most important layer of defense. As to cloud operators, their attack surface include not only what are mentioned before but also important hardware components of the cloud infrastructure. In contrast to traditional enterprise networks, the attack surface of clouds is much more complex, involving physical components, software services, virtualization, cloud operating systems, etc., as demonstrated by our examples above.

3.4.2 Attack Tree

In the previous section, we have described each attack scenario through a series of attack steps involving different attack surfaces. To better capture what may happen once an attacker gains initial privileges, we now apply attack trees, which represent high level attack paths leading attackers to their goals. Figure 6 shows an attack tree for our cloud data center infrastructures. It is assumed that the root node, or goal node, is a storage device in the cloud that is susceptible to attacks by either a malicious user, a cloud tenant, or a cloud operator. Eight paths in Figure 6 represent the possible ways to reach such a target. Each path represents a different capability level of attackers who can follow the path so not all paths are accessible to every attacker. For example, some paths can only be followed by the cloud operator but cannot be accessed by end users or cloud tenants. In what follows, we explain those paths and corresponding attack scenarios in further details.

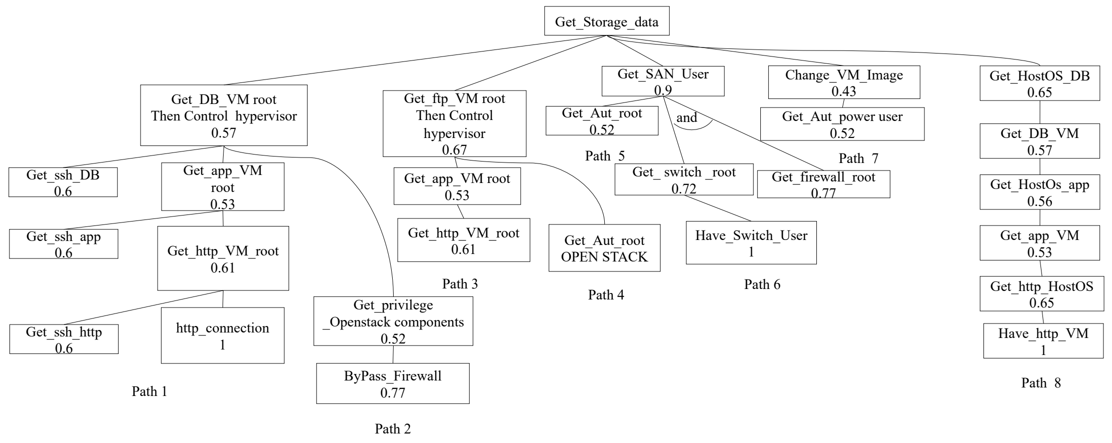


Figure 6: Attack tree

- *Path 1:* This attack can be executed by an end user to obtain data from the storage device (node 16). The user must first establish a connection to the $http$ VM server (node 11) and must then acquire the root privilege on this VM. The attacker can then connect to the application VM server (node 10) provided that they have obtained root privilege on that VM. After the user acquires access to the application VM, he/she may create a connection to the database VM server (node 8). From this point, the user can attack the database VM to obtain root privilege on that VM. Finally, the attacker can launch an attack on the hypervisor to gain access to other database VMs (node 8) running on the same physical machine and obtain data related to all database VMs stored on the storage device (node 16).

– *Path 2:* The end user can use this path to attack the cloud storage device (node 38). The attacker begins the attack by bypassing the firewall (node 22) to obtain privilege on OpenStack (node 36) in order to gain a direct connection to the database VM server (node 28). The remainder of this attack is similar to that of path 1, and serves to gain access to the hypervisor and the storage device.

– *Path 3:* This path can be used by a cloud tenant user who has user access to the $http$ VM server (node 11) and wishes to access ftp files stored on the storage device (node 16). First, the cloud tenant user must obtain root privilege on the $http$ VM server (node 11). Then, he/she will need to obtain root privilege on the application VM server (node 10) to start a connection to the ftp VM server (node 9). After this, the user will obtain root privilege to this VM and get the ftp files related to this VM. In addition, the user can attack the hypervisor to obtain the ftp files related to other VMs running on top of this hypervisor.

– *Path 4:* Cloud tenants who do not already possess ftp VM servers running on the cloud can use this path to obtain data from the storage device (node 16) through the ftp VM server (node 9). Cloud tenants on this path will use OpenStack components (node 3) to gain privileges to access the ftp VM (node 9) belonging to another cloud tenant. In this situation, the attacker can obtain all files belonging to this VM. Furthermore, the attacker may attack the hypervisor to gain access to other ftp VMs running on the same physical machine.

– *Path 5:* Cloud operators with access to the admin user authentication server (node 18) can use this path by obtaining root access to the authentication server. They can then use this device to obtain root access on the SAN device (node 16) to control the data stored on the storage device.

- *Path 6:* This path can be used by a cloud operator who has access to a physical machine (e.g., a switch, firewall, or other type of machines) to attack the storage device. Suppose the attacker has user access to a switch device (node 13) for maintaining this device. The attacker can obtain root access to this device followed by root access to a firewall device (node 14) between the switch device and the SAN (node 16). These two accesses may allow him/her to create a connection to the SAN device and subsequently attack the SAN in order to access the stored data.

- *Path 7:* This path may be used by a third party cloud operator who has access to the authentication server (node 18) of an administrator. The attacker must obtain root access to the authentication server and then gain privilege on the VM image storage (node 18) and (node 16). In this case, the attacker may use this privilege to modify or change the VM images stored on Glance such that the modified image will have a backdoor embedded which can later be used by the attacker to gain access to all VMs using this image.

- *Path 8:* This path can be used by either a cloud tenant or an end user. The goal for these attackers is to control the data belonging to other tenants in the cloud. The attacker must first have access to the $http$ VM server (node 31) and then gain access to the host operating system (node 31) and hence access to all VMs running on this machine. The attacker may then gain access to all the application VMs (node 30) that are connected to all $http$ VMs to which they have access. Subsequently, the attacker gains access to the application VMs which may be running on different physical machines and acquire access over their host OS and VMs (node 30). The attacker can then gain root access to the database VM server (node 28) in order to obtain the data stored on the storage device. The attacker may also decide to gain access to all the host OSs running database VMs (node 28).

Summary In contrast to attack surface, the attack tree model more clearly shows the big picture by depicting all the paths that can be followed by different kinds of attackers to compromise an important asset modeled as the attack goal (note although we have assumed each path is followed by one type of attackers, it can certainly be followed by more powerful attackers with a superset of the required privileges along the path). The structured representation of attack tree also makes it easier to spot interesting patterns across different attack scenarios. For instance, we can observe different paths sometimes share some common nodes (e.g., between paths 1 and 2) in the attack tree. This clearly depicts that common attack surfaces are usually required for the same type of attackers (e.g., end users) despite the difference in their specific attacks, which also implies an opportunity in defense, since removing such common nodes may help mitigate many attacks. Finally, we can also observe that more powerful attackers (e.g., cloud operators) tend to have shorter paths (e.g., path 5 and 7) since their increased initial privileges can usually simplify the attacks. On the other hand, such an observation is obviously qualitative in nature and not precise enough, and it does not take into consideration other important factors, such as the relative risks of different paths. This motivates us to discuss quantitative models, such as security metrics, in Section 3.5.1.

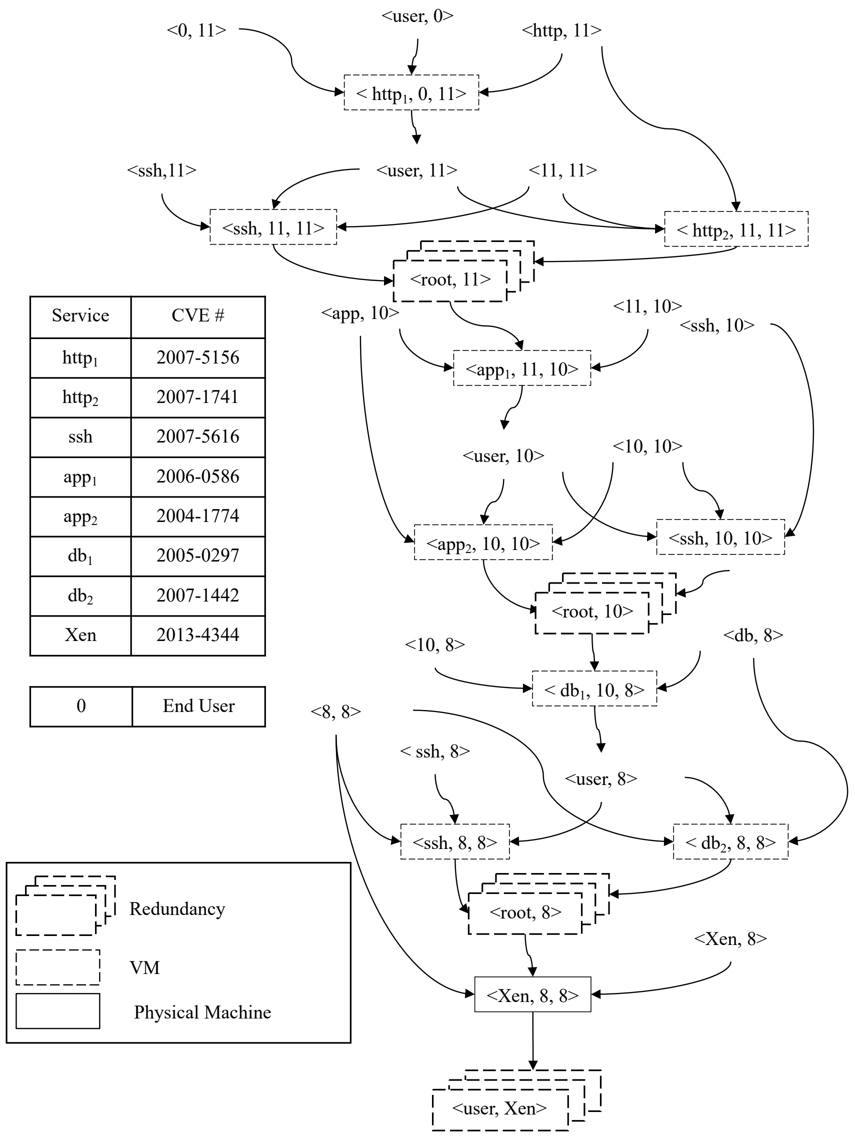
3.4.3 Attack Graph

The previous section shows how attack trees can capture the attack paths potentially followed by attackers to compromise critical assets. However, the attack tree is still a relatively high level concept, without details about specific ways for exploiting a resource. We now apply attack graphs to represent specific exploits of vulnerabilities that can be used to compromise critical assets along each path of the attack tree. Although we can apply the standard attack graph concept designed for traditional networks, special consideration needs to be given to the unique aspects of clouds, such as virtualization and redundancy. First, traditional attack graphs do not distinguish between physical and virtual resources, which can be important for human inspection or certain analysis performed on attack graphs. Second, a cloud data center usually have racks of machines with similar or identical configurations, and the traditional way of modeling every one of those machines in an attack is obviously redundant and not scalable. Therefore, in our application of attack graphs, we introduce two new graphical notations, i.e., dashed line for representing virtualization (e.g., exploits on VMs), and stacked rectangles as a simplified representation for a collection of similar exploits on multiple hosts with similar configurations. Finally, like in the case of attack surface and attack tree, we also need to construct attack graphs for different types of attackers. Also, we construct our attack scenarios based on real vulnerabilities related to hardware and software components used in our infrastructures as listed in the National Vulnerability Database (NVD) [1].

Attack Graphs for End Users

Figures 7 and 8 show two attack graphs for adversaries taking the role of end users. The Figure 7 is based on infrastructure 1 and the Figure 8 for infrastructure 2. In both cases it is assumed that the attacker has access to cloud services. The main goal for the attacker is to steal data from the storage. The attack graphs show how an attacker may gain access to the *http* VM, the application VM, and database VM, before reaching the goal due to the multi-tier infrastructure. The following services are assumed, i.e., Tectia Server version 5.2.3, for *ssh* running on all VMs, Apache *http* server running on the *http* VM, Oracle version 10.1.0.2 on the application VM, Oracle version 10.2.1 on the database VM, and Xen version 4.3.0 as a hypervisor to control VMs running on top of physical machines.

Example 7. *Figure 7 shows an attack graph corresponding to path 1 in the aforementioned*



Service	CVE #
http$_1$	2007-5156
http$_2$	2007-1741
ssh	2007-5616
app$_1$	2006-0586
app$_2$	2004-1774
db$_1$	2005-0297
db$_2$	2007-1442
Xen	2013-4344

0	End User

Figure 7: Attack graphs for end users in infrastructure 1

attack tree. Between five to seven vulnerabilities are required to reach the goal. Specifically, five vulnerabilities are required if we assume the ssh vulnerability will be the same in the $http$ server VM, application server VM, and database server VM, whereas seven vulnerabilities are required if the ssh vulnerability is not used to reach the goal. We divide the attack graph to four stages and in each stage the attacker will gain a different level of privileges.

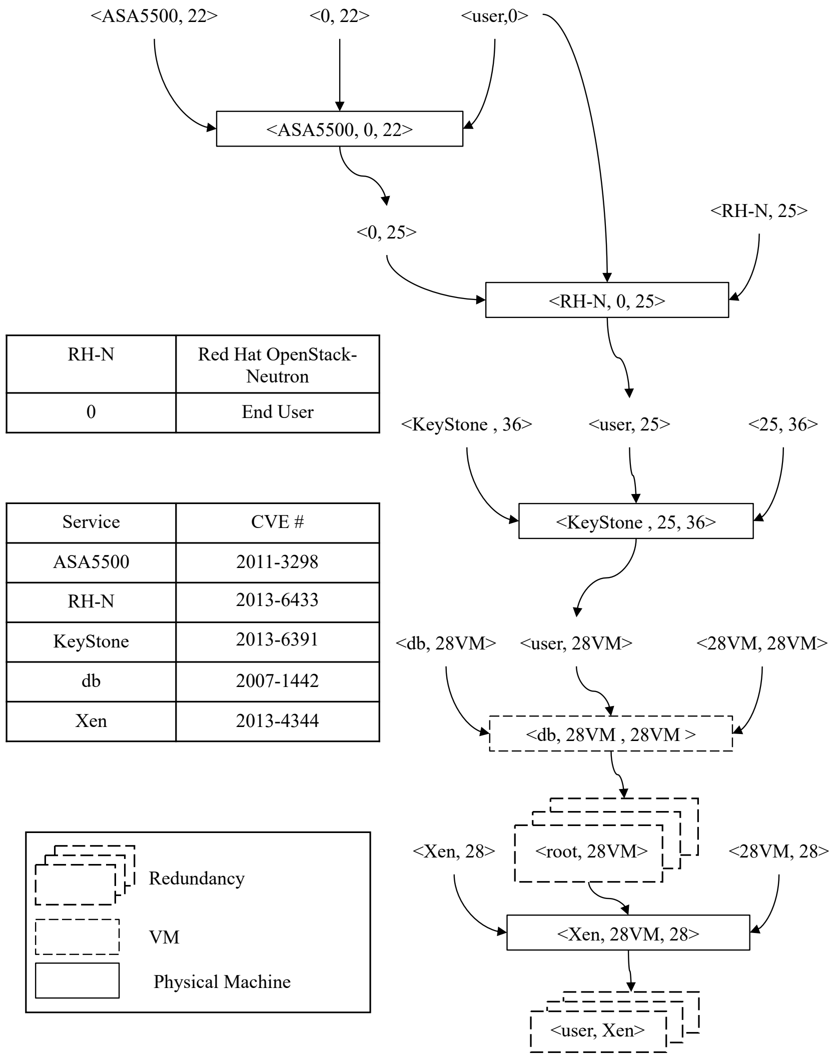
– Stage 1: *A vulnerability in the $http$ server VM (node 11) (CVE-2007-5156) is employed by the attacker to gain user access by uploading and executing arbitrary PHP code. Then, another vulnerability on the same VM (CVE-2007-1741) is used to gain root privilege by renaming the directory or performing symlink attacks. A ssh (node 11) vulnerability (CVE-2007-5156) can also be used to gain root privilege on the same VM. At this point, the attacker can use exploit VMs with similar configurations to compromise other copies of the VM to expand his/her attack and go through Stage 4 below to reach the hypervisor on each VM copy.*

– Stage 2: *The attacker now can connect to the application server (node 10). By exploiting a vulnerability related to the application server VM (CVE-2006-0586), the attacker can gain the user privilege by executing arbitrary SQL commands via multiple parameters. To gain root privilege on this VM, the attacker can apply this vulnerability (CVE-2004-1774) or by using an ssh (node 10) vulnerability (CVE-2007-5616), and at this point the attacker can establish a connection to the database server VM. Also, he/she can exploit the redundancy between VMs as mentioned in the previous stage.*

– Stage 3: *The attacker exploits a vulnerability related to the database server (node 8) VM (CVE-2005-0297) to gain user access. Then, he/she can gain root access to this VM by using vulnerability (CVE-2007-1442) or an ssh (node 8) vulnerability (CVE-2007-5616). The attacker can also exploit the redundancy here.*

– Stage 4: *The attacker can now obtain data related to the database VM (node 8), and he/she may also attempt to obtain more data from other VMs running on the same physical machine by attacking the hypervisor through a vulnerability such as the CVE-2013-4344 (buffer overflow in Xen).*

Example 8. *Figure 8 is related to the infrastructure 2, where OpenStack components run* *on different physical machines. The goal of this attack is to gain access to date storage in three stages. This attack graph corresponds to path 2 in the attack tree.*



RH-N	Red Hat OpenStack-Neutron
0	End User

Service	CVE #
ASA5500	2011-3298
RH-N	2013-6433
KeyStone	2013-6391
db	2007-1442
Xen	2013-4344

Figure 8: Attack graphs for end users in infrastructure 2

– Stage 1: *A vulnerability in the firewall (node 22) (CVE-2011-3298) (which allows attackers to bypass authentication via a crafted TACACS+ reply) is employed by the attacker to bypass the firewall in order to connect to the Neutron server (node 25). The attacker can then use the Neutron vulnerability (CVE-2013-6433) (which allows remote attackers to gain privileges via a crafted configuration file) to gain* *privileges with which he/she can use vulnerability (CVE-2013-6391) in Keystone to gain privileges and access a database VM (node 28).*

– Stage 2: *After the attacker obtains access to the database VM (node 28), he/she can exploit the vulnerability (CVE-2007-1442) (a vulnerability in Oracle to allow local users to gain privileges) to gain root privilege on the same VM. This privilege allows the attacker to obtain data related to this VM or to further exploit the redundancy to access other VMs run on the same physical machine.*

– Stage 3: *To obtain data from another database on the same physical machine, the attacker can exploit the aforementioned vulnerability (CVE-2013-4344) to gain access to the hypervisor running on this physical machine.*

Attack Graphs for Cloud Tenants

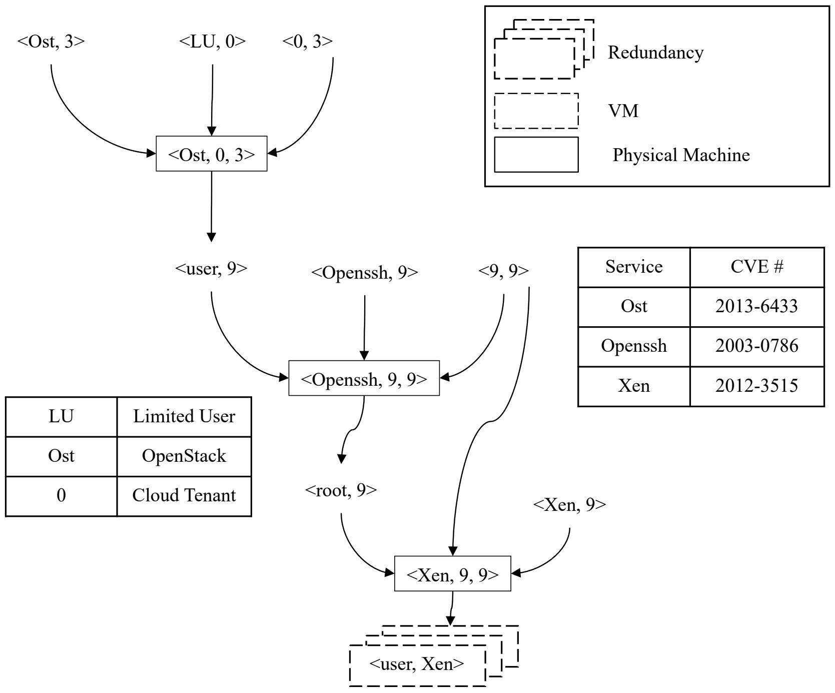
Figures 9 and 10 show two attack graphs for adversaries taking the role of cloud tenants. The Figure 9 is based on infrastructure 1 and the Figure 10 for infrastructure 2.

Example 9. *Figure 9 shows an attack that can be used by a cloud tenant with low privilege. Assume the attacker does not has access to a ftp server VM and his/her goal is to gain access to both the ftp server VM (node 9) and another tenant's VMs running on the same physical machine. This attack graph corresponds to path 3 in the attack tree. In this example, the following services are assumed to be used in the data centers, i.e., all components of OpenStack running on the authentication server, OpenSSH 7.7.1 on the ftp server VM, Xen version 4.1.0 as a hypervisor on the ftp server's physical machine. Three stages are required to reach the goal.*

– Stage 1: *A vulnerability in the ftp server VM (node 9) (CVE-2013-6433 as mentioned above) is employed by the cloud tenant attacker to gain user access on the ftp server VM.*

– Stage 2: *After the attacker obtains access to the ftp VM (node 9), he/she uses another ftp vulnerability (CVE-2003-0786, which allows remote attackers to gain privileges) to gain root privilege on the same VM.*

– Stage 3: *To obtain files belonging to another tenant on the same physical machine, the attacker exploits the vulnerability (CVE-2012-3515, which allows local OS guest users to gain privileges) to gain access to Xen (node 9) running on this physical machine such that he/she can access all VMs running on this machine and obtain files related to these VMs. Also, the attacker may exploit redundancy to expand his/her attack to other VMs copies.*



Service	CVE #
Ost	2013-6433
Openssh	2003-0786
Xen	2012-3515

LU	Limited User
Ost	OpenStack
0	Cloud Tenant

Figure 9: Attack graphs of cloud tenants for infrastructure 1

Example 10. *Figure 10 is based on infrastructure 2 where OpenStack components are distributed to multiple hosts. The goal for a malicious cloud tenant is to gain access on a host OS on the physical machine of the database VM server (node 28) to control all VMs* *running on that physical machine. In this example, the cloud tenant has a VM running on the $http$ server (node 31) but does not have any VM on the application (node 30) or database (node 28) servers. This attack graph corresponds to path 8 in the attack tree. Three stages are required in this example to reach the goal.*

– Stage 1: *A vulnerability in the $http$ VM (node 3) (CVE-2015-5154, a buffer overflow vulnerability that allows local guest users to execute arbitrary code on the host via unspecified ATAPI commands) is employed by the attacker, who then has access to the host OS of the physical machine and hence the control of all VMs running on this machine. The attacker can exploit the redundancy to attack other physical machines running a copy of the same VM and expand his/her attack.*

– Stage 2: *The attacker now can connect to the application server (node 30) by using one $http$ VM, which is connected to the application VM server (assuming this application VM does not connects to a database VM server (node 28)). Then, by using a vulnerability related to the application VM server (CVE-2015-3247, a race condition vulnerability that allows a remote attacker to execute arbitrary code on the host via unspecified vectors), the attacker is allowed to gain access to the host OS on the application server to control all its VMs. The redundancy can also be exploited here to control other VMs.*

– Stage 3: *The attacker can then use one of the new application VM servers to get access to the database VM server (node 28) and obtain data related to this database. Also, he/she may attempt to obtain more data from other VMs running on the same physical machine by attacking the host OS through exploiting (CVE-2015-3456, a vulnerability that allows local guest users to execute arbitrary code). In this stage, the redundancy can also be exploited to expand the attack.*

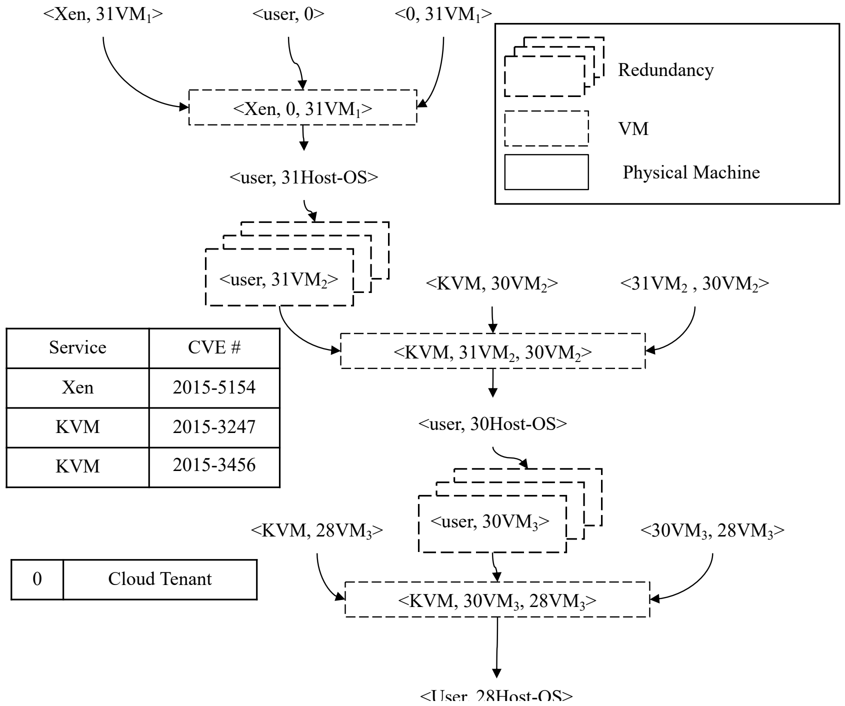
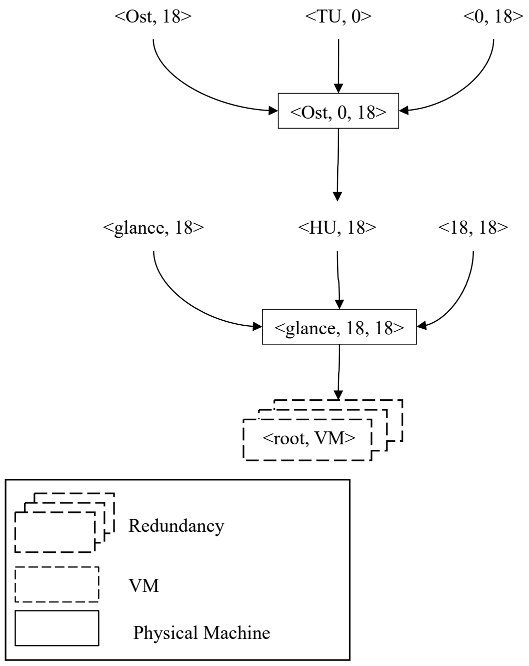


Figure 10: Attack graphs of cloud tenants for infrastructure 2

Attack Graphs for Cloud Operators

We first model third party cloud operators and then model internal cloud operators with more privileges. Figure 11 shows the attack graph of an adversary taking the role of a third party cloud operator who has the permission of changing VM images for cloud tenants. This attack graph is based on the infrastructure 1. The attack corresponds to path 7 in the attack tree, and is similar to real world cases where unauthorized third parties gain access to cloud customers' account information [20].

Example 11. *A vulnerability in OpenStack Keystone (node 18) (CVE-2014-3476) is employed by the attacker, which allows remote authenticated users to gain privileges they do not already possess. The attacker is then assumed to have access to the Glance storage from the previous exploit. The attacker now can exploit the vulnerability (CVE-2014-0162), which allows a user with permission to make changes (add, remove, and* *modify) to VM images. Subsequently, the attacker can attack any VM running the modify image to gain control or information. The attacker can also exploit redundancy to access another physical machine running a copy of the same VM or exploit one of the previous vulnerabilities to attack the hypervisor.*



Ost	OpenStack
TU	Third party user
HU	Third party user W/Higher Privilege
0	Third Party

Service	CVE #
Ost	2014-3476
glance	2014-0162

Figure 11: An attack graph of third party cloud operators for infrastructure 1

Figures 12 and 13 demonstrates two attack graphs for internal cloud operators with more privileges. The Figure 12 illustrates an attacker who has access to the physical device and gain access to storage. The Figure 13 illustrates a user who has access to the admin user authentication server to obtain data from the storage. These two attack graphs correspond to path 6 and path 5 in the attack tree, respectively. There also exist well known real world incidents similar to those attacks, e.g., the case of Google dismissing employees due to breaching customers' privacy [36].

Example 12. *In Figure 12, the attacker is a cloud operator who has access to the physical* *switch device (Nexus 5000) (node 7) for maintenance. The goal for this attacker is to gain access to the storage device. In this example, three vulnerabilities are required to reach the goal in four stages.*

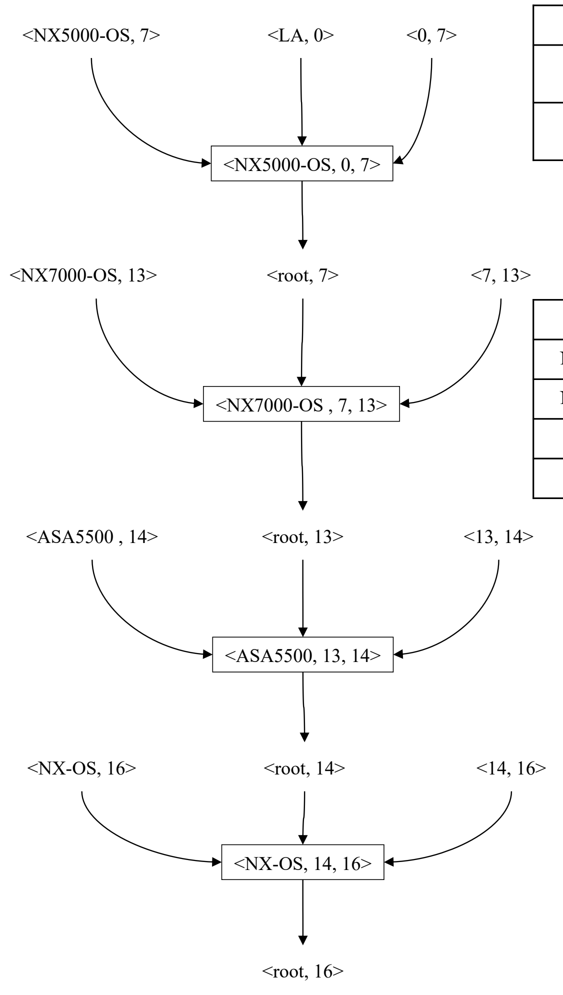
– Stage 1: *A vulnerability in the Nexus 5000 (node 7) (CVE-2013-1178, which are multiple buffer overflows in the Cisco Discovery Protocol (CDP) implementation that allow remote attackers to execute arbitrary code via malformed CDP packets) is employed by the attacker to gain root privilege on the Nexus 5000. Then, the attacker can establish connection to the Nexus 7000 (node 13).*

– Stage 2: *By exploiting the previous vulnerability again, the attacker will gain root access to Nexus 7000 (node 13).*

– Stage 3: *The attacker now can connect to the firewall (node 14). Then, by using a vulnerability related to the firewall (CVE-2007-0960), which allows the attacker to gain root privilege. Consequently, the attacker can change the firewall rules and allow connection to the MDS 9000 device (node 16).*

– Stage 4: *To obtain data from MDS 9000, The attacker can use the vulnerability (CVE-2013-1180, a buffer overflow in the SNMP implementation which allows remote authenticated users to execute arbitrary code) to gain root privilege and thereby gain access to storage.*

Example 13. *In Figure 13, the attacker has access to the billing system (node 18). The goal of the attacker is to obtain higher privileges on the cloud system and to access the cloud storage device (node 16). In this attack graph, Three vulnerabilities can be used to gain access to storage, and the attacker needs to exploit two vulnerabilities to reach his/her goal.*



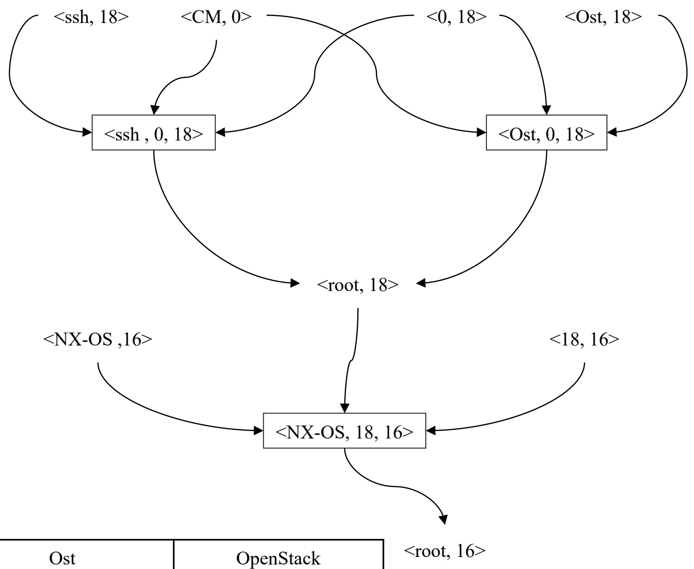
NX	Nexus Cisco
LA	Limited Admin
0	Cloud Operator

Service	CVE #
NX5000-OS	2013-1178
NX7000-OS	2013-1178
ASA5500	2007-0960
NX-OS	2013-1180

Figure 12: Attack graphs of cloud operators example 13

– Stage 1: *The attacker can exploit one of the two following vulnerabilities in the authentication server (node 18) to gain root access, i.e., a vulnerability related to ssh (CVE-2007-5616) which allows local user to gain root privilege, and a vulnerability related to OpenStack-Neutron (CVE-2014-3632) which allows attackers to gain root privilege.*

– Stage 2: *Then, the attacker can use one of the previous vulnerabilities to get root privileges to open a connection to the MDS 9000 (node 16), and he/she can then exploit the vulnerability in the (CVE-2013-1178) MDS 9000 to obtain root access,* *thereby obtaining data from the storage.*



Ost	OpenStack
CM	Cloud Management User
NX-OS	Cisco Nexus Operating System
0	Cloud Operator

Service	CVE #
Ost	2014-3632
ssh	2007-5616
NX-OS	2013-1178

Figure 13: Attack graphs of cloud operators example 13

Summary Unlike attack surface and attack tress, attack graphs provide more specific details about the vulnerabilities that may be exploited to compromise a critical asset. We have demonstrated how each path in an attack tree may be instantiated as an attack graph with concrete exploits of vulnerabilities. Our examples also demonstrate the new opportunities, in terms of concrete vulnerabilities, for attackers to exploit virtualization and redundancy in cloud infrastructures, and how such unique features of clouds may be easily handled in attack graphs through adding some simple graphical notations.

Finally, by constructing attack surface, attack trees, and attack graphs for our cloud infrastructures, we have demonstrated how each model may capture potential threats at a different abstraction level and how they could work together. Models at a higher level, such as attack surface or attack trees, may serve as a starting point to show the big picture and to guide further efforts spent on a more detailed, and certainly more expensive model, such as attack graphs. We have also focused on attack scenarios which are designed to employ the unique features of cloud infrastructures, such as the co-existence of different types of users, virtualization, and components with similar configurations, and those scenarios clearly show that cloud infrastructures may be subject to novel threats not present in traditional enterprise networks. Nonetheless, all those models are qualitative in nature, and we will apply security metrics to measure the threats in the coming section.

3.5 Security Metrics

In this section, we apply security metrics based on the attack tree and attack graphs to quantitatively model the threats discussed in the previous section.

3.5.1 Attack Tree Metric

We first apply an attack tree metric (ATM) based on the attack tree described in Section 3.4.2. In Figure 14, all nodes inside the same path are considered as having AND relationships, whereas an OR relationship is assumed between different paths unless if an AND relationship is explicitly stated. Following such assumptions, the probabilities may be calculated based on the corresponding logic relationships. The highest probability is assigned to the root node after applying the metric. In Figure 14, between the two

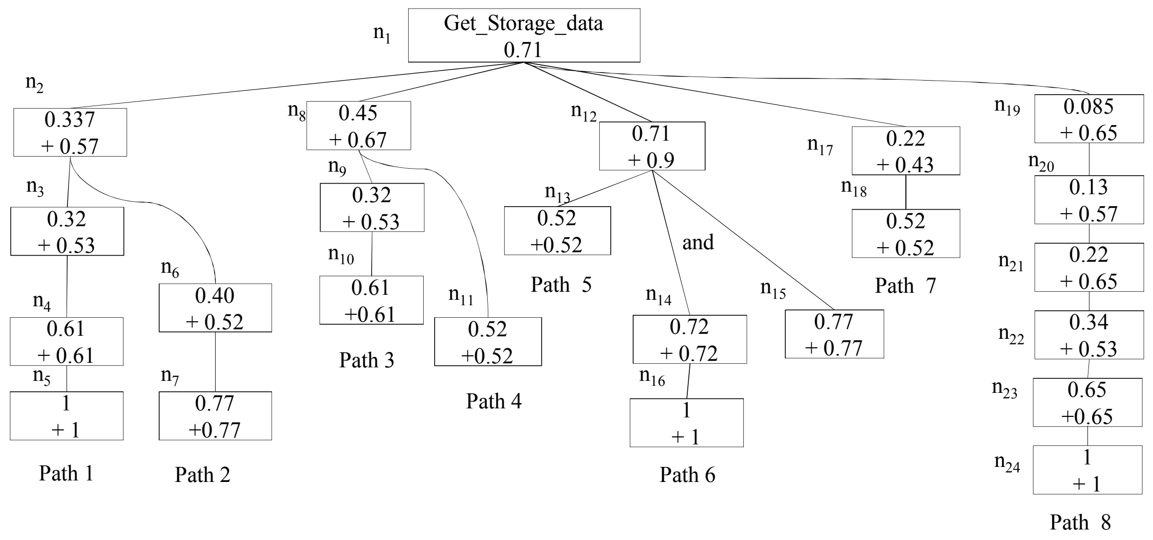


Figure 14: Attack tree metric

probabilities in each node, the probability with a preceding (+) symbol represents the average values of the Common Vulnerability Scoring System (CVSS) [95] scores divided by 10 (the domain size of CVSS scores), which represents the probability of realizing each individual node without considering the dependence on its parent node. The other probability represents the metric result calculated as above.

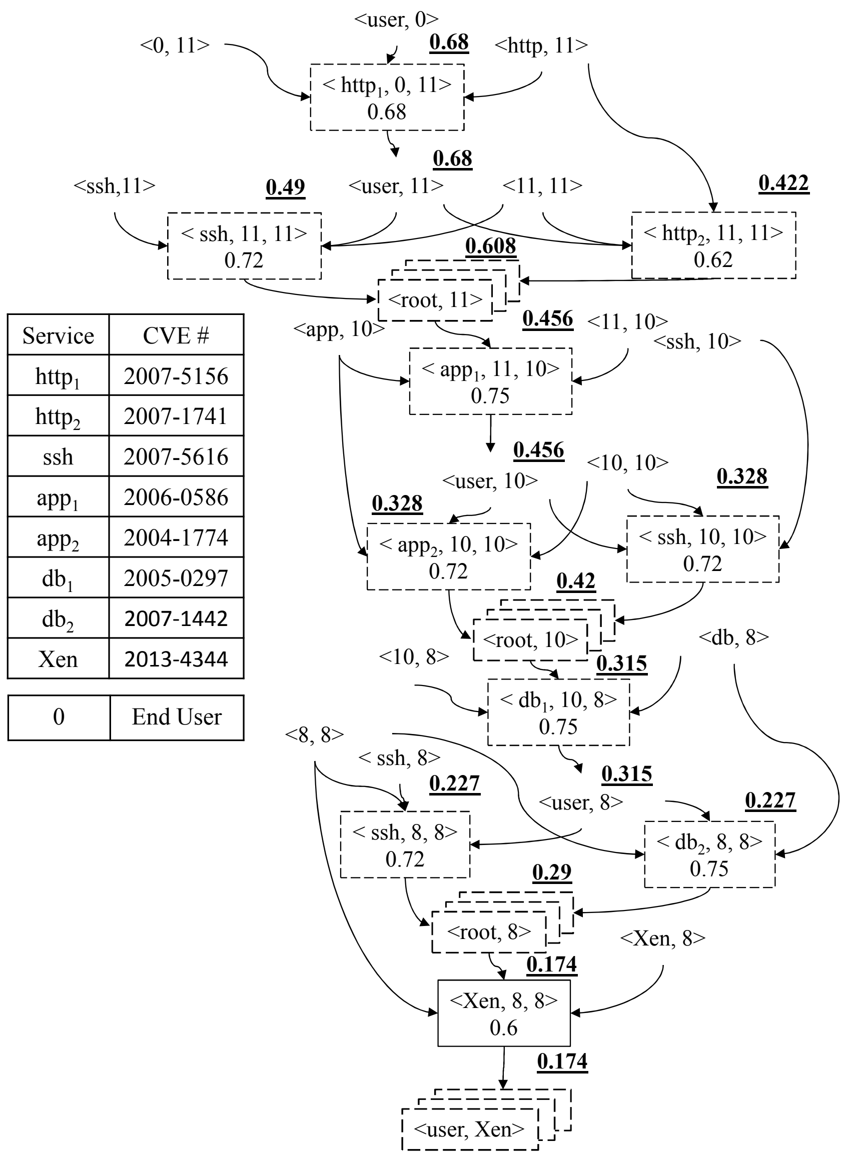
In Figure 14, it can be observed that path 5 and 6 are the least secure paths in the attack tree. This makes sense since those two paths represent the insider attacks launched by the most powerful attackers, i.e., cloud operators. In addition to weighing different paths, this metric can also be used to evaluate whether adding a new service or disabling existing services can increase security and by how much. As shown in Figure 14, the probability to reach n_8 is 0.45; as such, if the cloud operator wishes to decide whether to increase security levels in that node, he/she can use the metric before and after applying the desired changes. For example, suppose the cloud operator wishes to add new rules to a firewall to prevent attacks from n_9 and n_{11} to n_8. After re-applying the ATM metric, the probability on n_8 becomes 0.348, showing increased security. Applying the metric on other potential changes may help the cloud operator to make the right decisions in hardening the cloud, and we will discuss such changes in more details in the coming section.

3.5.2 Attack Graph Metric

In this section, the attack graph-based security metric [137, 57] will be applied to the Figure 7. By annotating the attack graph with probabilities derived from CVSS [95] scores (retrieved from the NVD) as depicted inside each node, we convert the attack graph into a Bayesian network shown in Figure 15. The goal is to quantitatively model the threat, and also to evaluate the effect of certain changes made to the cloud infrastructure. In particular, we show how the level of redundancy and diversity may affect the security of the cloud infrastructure. For redundancy, the *ssh* service running on some of the servers will be disabled to see the effect on security. As to diversity, we assume the *ssh* service may be diversified with other software, e.g., OpenSSH version 4.3, denoted as ssh_2, which has a vulnerability CVE-2009-290 with a CVSS score of 6.9 [1].

Table 2 shows how security is affected by reducing redundancy and increasing diversity through disabling or diversifying some of the *ssh* instances in the infrastructure. In the left-hand side table, the first row shows that the probability for an attacker to reach the goal is 0.174 in the original configuration, and the remaining rows show the same probability after disabling one or more *ssh* instances on the three servers, e.g., the probability after disabling *ssh* on the *http* server is reduced to 0.121, which corresponds to the most secure option by disabling one *ssh* instance, and the lowest probability after disabling two and three *ssh* instances is 0.094 and 0.074, respectively.

The middle and right-hand side of Table 2 show the effect of diversifying the *ssh* instances. In the middle figure, we can observe that, after we replace the *ssh* service on *app* and *DB* servers with ssh_2, the probability for reaching the goal decreases from 0.174 to 0.171, which indicates a slight improvement in security. The next three rows of the table show that the same effect remains when one of the *ssh* instances is disabled. The last three



Service	CVE #
http$_1$	2007-5156
http$_2$	2007-1741
ssh	2007-5616
app$_1$	2006-0586
app$_2$	2004-1774
db$_1$	2005-0297
db$_2$	2007-1442
Xen	2013-4344

0	End User

Figure 15: Attack graph metric (end user on infrastructure 1)

rows show the simple fact that, when there is only one *ssh* instance left, the diversification effort has not effect.

In the right-hand side of Table 2, we change the *ssh* instance on the *http* server instead of the *app* server, as in the above case, in order to see whether different diversification options make any difference to security. We can see the probability decreases in most cases (except the fourth row), which indicates a slightly more effective option than the previous one.

Overall, the best option in terms of diversification without disabling any service instance is given in the first row in the right-hand table, with a probability 0.17, and the best option for disabling one service instance is given in the fourth row of the middle table with a probability 0.119 (disabling two instances always yields 0.094). Obviously, considering more options may further harden the cloud infrastructure, which will be addressed in the coming section.

Table 2: The metric results of making changes to the cloud infrastructure

$\langle user, Xen\rangle$			
$http$	app	DB	T
ssh			T
T	T	T	0.174
T	F	T	0.136
T	T	F	0.136
F	T	T	0.121
T	F	F	0.106
F	F	T	0.094
F	T	F	0.094
F	F	F	0.074

$\langle user, Xen\rangle$			
$http$	app	DB	T
ssh_1	ssh_2	ssh_2	T
T	T	T	0.171
T	F	T	0.135
T	T	F	0.135
F	T	T	0.119
T	F	F	0.106
F	F	T	0.094
F	T	F	0.094
F	F	F	0.074

$\langle user, Xen\rangle$			
$http$	app	DB	T
ssh_2	ssh_1	ssh_2	T
T	T	T	0.17
T	F	T	0.133
T	T	F	0.134
F	T	T	0.12
T	F	F	0.105
F	F	T	0.094
F	T	F	0.094
F	F	F	0.074

Summary Our examples have shown how the attack tree and attack graph models can be enhanced with quantitative modeling power. The attack tree-based metric may allow cloud providers to prioritize further modeling effort among different paths or different nodes. The attack graph-based metric further illustrates the relative importance of individual vulnerabilities inside an attack scenario. More importantly, both models allow cloud providers to evaluate and compare the security effect of different hypothetic changes in order to identify the most effective hardening options to be actually deployed in the cloud infrastructure. Such a capability can significantly improve the effectiveness of cloud providers' security hardening practice while reducing the cost in terms of time and effort needed for the hardening.

3.6 Hardening the Cloud Data Center Infrastructure

In this section, we discuss different hardening options that can be used in a cloud infrastructure to improve the security. We will focus on the attack graph model (Section 3.4.3) and the BN-based security metric (Section 3.5.2) and apply them to examine the effectiveness of the hardening options applied to the cloud infrastructure. Specifically, we will evaluate the BN-based metric on the attack graph before and after applying hardening options to the infrastructure, and examine the difference in the metric results.

Based on our threat modeling results, we can observe many hardening options for improving the security of cloud infrastructures, as demonstrated in the following.

– Enforcing stricter access control to cloud services to make it harder for end user type of adversaries to access such services, and ensuring minimum privileges and sufficient accountability for cloud operators.

– Deploying firewalls to block non-essential connections inside the cloud infrastructure to prevent an attack from expanding its scope.

– Increasing diversity by deploying different hardware and software components in the cloud infrastructure such that a vulnerability will less like affect multiple components.

– Enforcing stronger isolation between VMs running on the same machine by improving hypervisor security to prevent attackers from escaping the VMs and compromise the host.

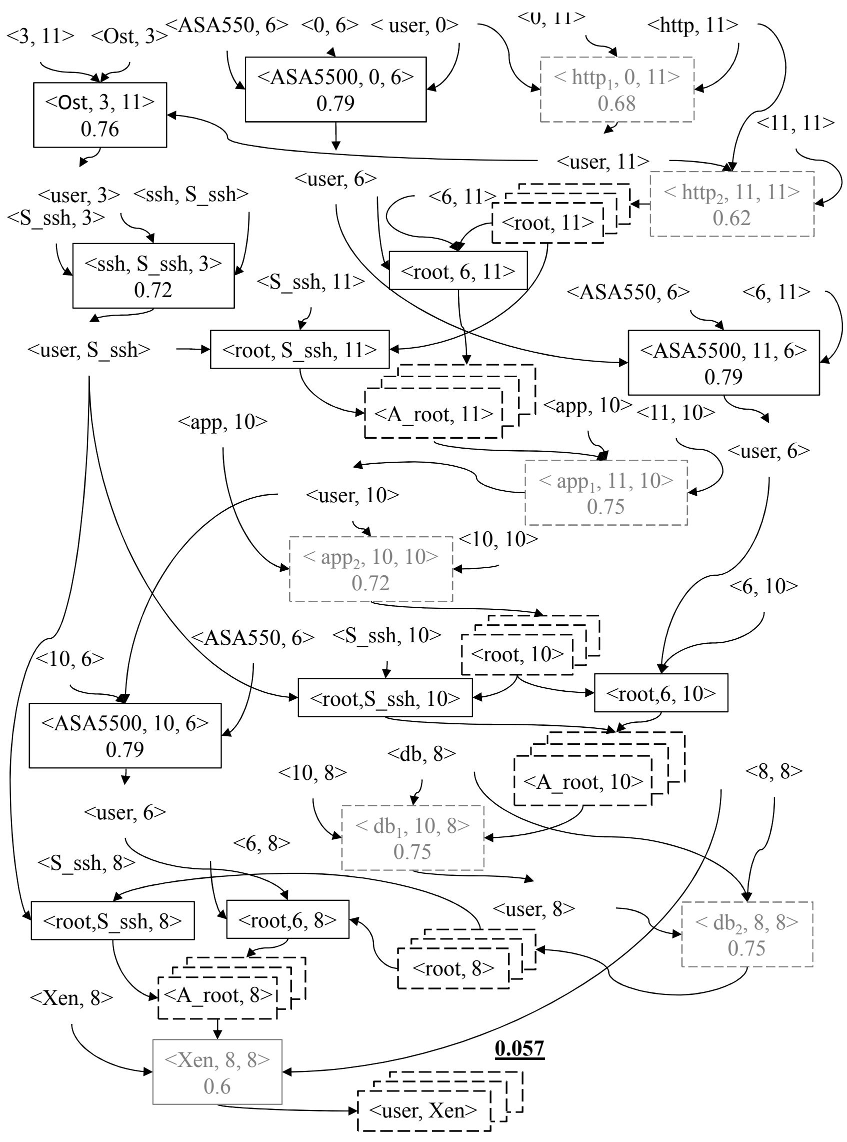
– Disabling non-essential services and removing unnecessary components from the cloud infrastructure to reduce the amount of attack surfaces available to an attacker.

– Patching known vulnerabilities in the cloud infrastructure to further reduce the attack surfaces.

We will use the cloud data center infrastructure 1 and attack graph examples in Section 3.4.3 to demonstrate how different hardening options may help to improve the security. The nodes in gray color in Figures 16 and 17 represent exploits and attack paths available to attackers before applying the hardening options, and those exploits will be removed after applying such options. We focus on two types of hardening options as follows. First, for enforcing stricter access control, we add an *ssh* authentication server such that any user who wants to use the *ssh* service to connect to his/her VMs must first get authenticated. Second, we will also add new firewall rules and new firewalls to block certain connections in the infrastructure. More specifically, we add new *ssh* servers which are connected to the authentication server (node 3) for cloud tenants in layer 1. We add new rules to the firewall (node 6) which allow only *ssh* connections coming from the new servers. Also, we add a new firewall between node 7 and node 13 which will drop all packets coming from node 7.

Hardening w.r.t. End Users Figure 16 shows the attack graphs for the end user type of attackers before and after the aforementioned hardening options are applied. The attack graph is similar to that in Figure 7 with the key difference that, once the attacker gains root privilege in each VM, he/she needs to exploit a firewall vulnerability (CVE-2011-0379) on node 6, or an *ssh* vulnerability and the VM service vulnerability (e.g., in $http$). After applying the BN-based metric, we can see that the security level has increased from 0.174 in Figure 7 to 0.057 in Figure 16. Thus, our hardening options have increased the level of security for end users by roughly 67%.

Hardening w.r.t. Cloud Tenant Figure 17 presents the case of cloud tenants. The key difference between this new attack graph and the attack graph in Figure 9 lies in the exploit of the *ssh* vulnerability (CVE-2007-5616) on the new *ssh* authentication server. By applying the BN-based metric to Figures 9 and 17, we can see the probability to reach the



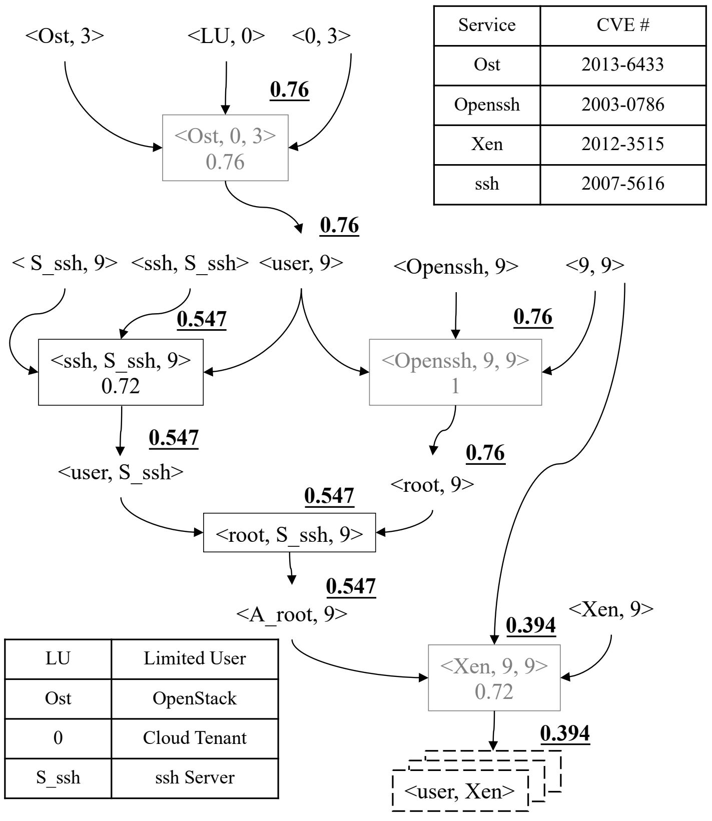
0	End User
Ost	OpenStack
S_ssh	ssh Server

Service	CVE #
$http_1$	2007-5156
$http_2$	2007-1741
Ost	2013-6433
ASA5500	2011-0379
ssh	2007-5616
app_1	2006-0586
app_2	2004-1774
db_1	2005-0297
db_2	2007-1442
Xen	2013-4344

Figure 16: Hardening Infrastructure 1 w.r.t. End Users

goal are 0.547 and 0.394, respectively. This means the level of security after the hardening effort has increased by about 28% for cloud tenants.

Hardening w.r.t. Cloud Operator We examine how much security can be added w.r.t. cloud operator type of attackers shown in Figure 12 by deploying a new firewall device. Figure 18 shows the new attack graph after we add the new firewall. By applying the BN-based metric, we find the leve of security has increased by 21% from 0.558 in Figure 12 to



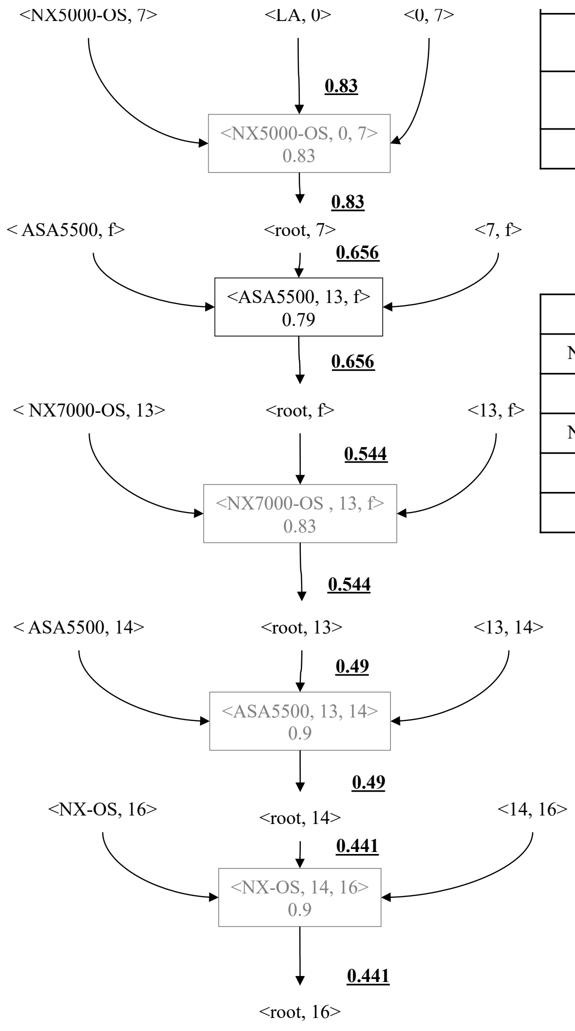
Service	CVE #
Ost	2013-6433
Openssh	2003-0786
Xen	2012-3515
ssh	2007-5616

LU	Limited User
Ost	OpenStack
0	Cloud Tenant
S_ssh	ssh Server

Figure 17: Hardening infrastructure 1 w.r.t. cloud tenants

0.441 in Figure 18.

Summary In addition to the hardening options of reducing redundancy and increasing diversity discussed in the previous section, we have demonstrated in this section two more hardening options, i.e., enforcing stricter access and adding new firewall (rules). In practice cloud administrators will need to consider not only such hardening options but also their corresponding monetary, operational, and administrative costs. A more systematic approach, such as the one proposed in [31], can be employed to automatically derive the most cost effective solution by combining multiple hardening options in an optimal way. Such a useful application clearly demonstrates the power of threat modeling when applied



LA	Limited Admin
0	Cloud Operator
f	Firewall

Service	CVE #
NX5000-OS	2013-1178
ASA5500	2011-0379
NX7000-OS	2013-1178
ASA5500	2007-0960
NX-OS	2013-1180

Figure 18: Hardening infrastructure 1 w.r.t. cloud operators

to cloud infrastructures.

3.7 Summary

In this chapter, we have studied the application of a series of threat modeling techniques to cloud data center infrastructures. First, we have devised two cloud data center infrastructures by integrating existing technologies adopted by major players in the cloud market. Three threat models were then applied to those infrastructures, namely, the attack surface, attack trees, and attack graphs, which model potential threats from different viewpoints and at different abstraction levels. We have also applied security metrics based on attack trees and attack graphs, respectively, to quantify the threats. Finally, we applied several hardening options to take the threat models into action by showing how the security level of cloud infrastructures may be improved in terms of a comparison between the metric results for the original infrastructure and the hardened infrastructure. Throughout our modeling exercises, we have focused on some unique aspects of cloud infrastructures, such as the existence of different types of users, virtualization, and configuration redundancy. We have demonstrated how such unique features may be handled in threat modeling and what additional security threats they may lead to. Such lessons may potentially benefit cloud providers in better understanding and mitigating the security threats facing their cloud infrastructures.

Chapter 4

Mitigating the Insider Threat of Remote Administrators in Clouds through Maintenance Task Assignments

4.1 Introduction

Cloud computing has become the cost-saving IT solution for 73% of organizations worldwide [50] and is predicted to grow to a $300 billion business by 2021 [59]. Cloud computing is also affecting our daily lives through its impact on politics (e.g., politicians are increasingly turning to social networks, which are mostly cloud-based), education (e.g., Massive Open Online Course (MOOC) is mostly delivered via cloud), healthcare, entertainment, etc. The success of cloud computing comes from the many benefits it brings to IT management, e.g., the pervasive access from anywhere with an Internet connection, the flexibility of scaling services up or down to fit changing needs, and the efficiency to deploy applications quickly without worrying about underlying costs or maintenance of the infrastructure.

On the other hand, the widespread adoption of cloud computing also attracts more attention to its unique security and privacy challenges [42, 129]. In particular, as the cloud service market becomes more and more competitive, cloud providers are striving to attract customers with better services and less downtime at a lower cost. Consequently, the search for an advantage in cost and efficiency will inevitably lead cloud providers to follow a similar path as what has been taken by their tenants, i.e., outsourcing cloud maintenance tasks to remote administrators including those from specialized third party maintenance providers [29]. Such an approach may also lead to many benefits due to resource sharing, e.g., the access to specialized and experienced domain experts, the flexibility (e.g., less need for full-time onsite staff), and the lower cost (due to the fact that such remote administrators are usually shared among many clients).

However, the benefits of outsourcing cloud maintenance tasks come at an apparent cost, i.e., the increased insider threats from remote administrators. Specifically, in order to complete their assigned maintenance tasks, the remote administrators must be provided with necessary privileges, which may involve accesses to physical and/or virtual resources of the underlying cloud infrastructure. Armed with such privileges, a dishonest remote administrator, or an attacker with the stolen credentials of such an administrator, can pose severe insider threats to both the cloud tenants (e.g., causing a large scale leak of confidential user data) and the provider (e.g., disrupting the cloud services or abusing the cloud infrastructure for illegal activities) [41]. On the other hand, cloud providers are under the obligation to prevent such security or privacy breaches caused by insiders [45], either as part of the service level agreements, or to ensure compliance with security standards (e.g., ISO 27017 [74]). Therefore, there is a pressing need to better understand and mitigate the insider threats of remote administrators in clouds.

Dealing with the insider threat of remote administrators in clouds faces unique challenges. First, as mentioned in the previous chapter, there is a lack of public access to the detailed information regarding cloud infrastructure configurations and typical maintenance tasks performed in clouds. Evidently, most existing works on insider attacks in clouds either stay at a high level or focus on individual nodes instead of the infrastructure [29, 85, 133]. Second, cloud infrastructures can be quite different from typical enterprise networks in terms of many aspects of security. For instance, multi-tenancy means there may co-exist different types of insiders with different privileges, such as administrators of a cloud tenant, those of the cloud provider, and third party remote administrators. Also, virtualization means a more complex attack surface consisting of not only physical nodes but also virtual or hypervisor layers. To the best of our knowledge, there is a lack of any concrete study in the literature on the insider attack of remote administrators in cloud data centers.

In this topic, we take the first step towards understanding and mitigating the insider threat of remote administrators in clouds. Specifically, we first model the maintenance tasks and their corresponding privileges based on industrial practices from major cloud vendors and providers. We then model the insider threats posed by remote administrators assigned to maintenance tasks by applying existing security metrics; remote administrators possess elevated privileges due to the assigned maintenance tasks, and those privileges correspond to initially satisfied security conditions, which are normally only accessible by external attackers after exploiting certain vulnerabilities. Such model allows us to formulate the mitigation of the insider threats of remote administrators as an optimization problem and solve it using standard optimization techniques. We evaluate our approach through simulations and the results demonstrate the effectiveness of our solution under various situations.

4.2 Preliminaries

This section gives a motivating example and discusses maintenance tasks and privileges.

4.2.1 Motivating Example

The insider threat of remote administrators depends on the underlying cloud infrastructures. Therefore, we will need the detailed configuration of cloud data centers in order to construct a concrete example of such insider threats. A key challenge here is the lack of public accesses to detailed information regarding hardware and software configurations deployed in real cloud data centers. Consequently, most existing works focus on either high level frameworks and guidelines for risk and impact assessment [2, 119, 88], or specific vulnerabilities or threats in clouds [46, 124], with a clear gap between the two. To overcome such a limitation, we choose to devise our own fictitious, but realistic cloud data center designs, by piecing together publicly available information gathered from various cloud vendors and providers [10], as shown in Figure 19.

To above configuration is based on existing concepts and common practices borrowed from major cloud vendors and providers to make our design more representative. For example, we borrow the multi-layer concept and some hardware components, e.g., Carrier Routing System (CRS), Nexus (7000,5000,2000), Catalyst 6500, and MDS 9000, from the cloud data center design of Cisco [19]. We synthesize various concepts of the VMware vSphere [67] for main functionality of hardware components in our cloud infrastructure (e.g., authentication servers, DNS, and SAN). We also assume the cloud employs OpenStack as its operating system [101]. The infrastructure provides accesses to both cloud users and remote administrators through the three layer design. Layer 1 connects the cloud to the internet and includes the authentication servers, DNS, and Neutron Server. Layer 2 includes the rack servers and compute nodes. Layer 3 includes the storage servers. OpenStack components run on the authentication servers, DNS server (a Neutron component provides address translation to machines running the requested services), and compute nodes (Nova to host and manage VMs, Neutron to connect VMs to the network, and Ceilometer to calculate the usage) to provide cloud services.

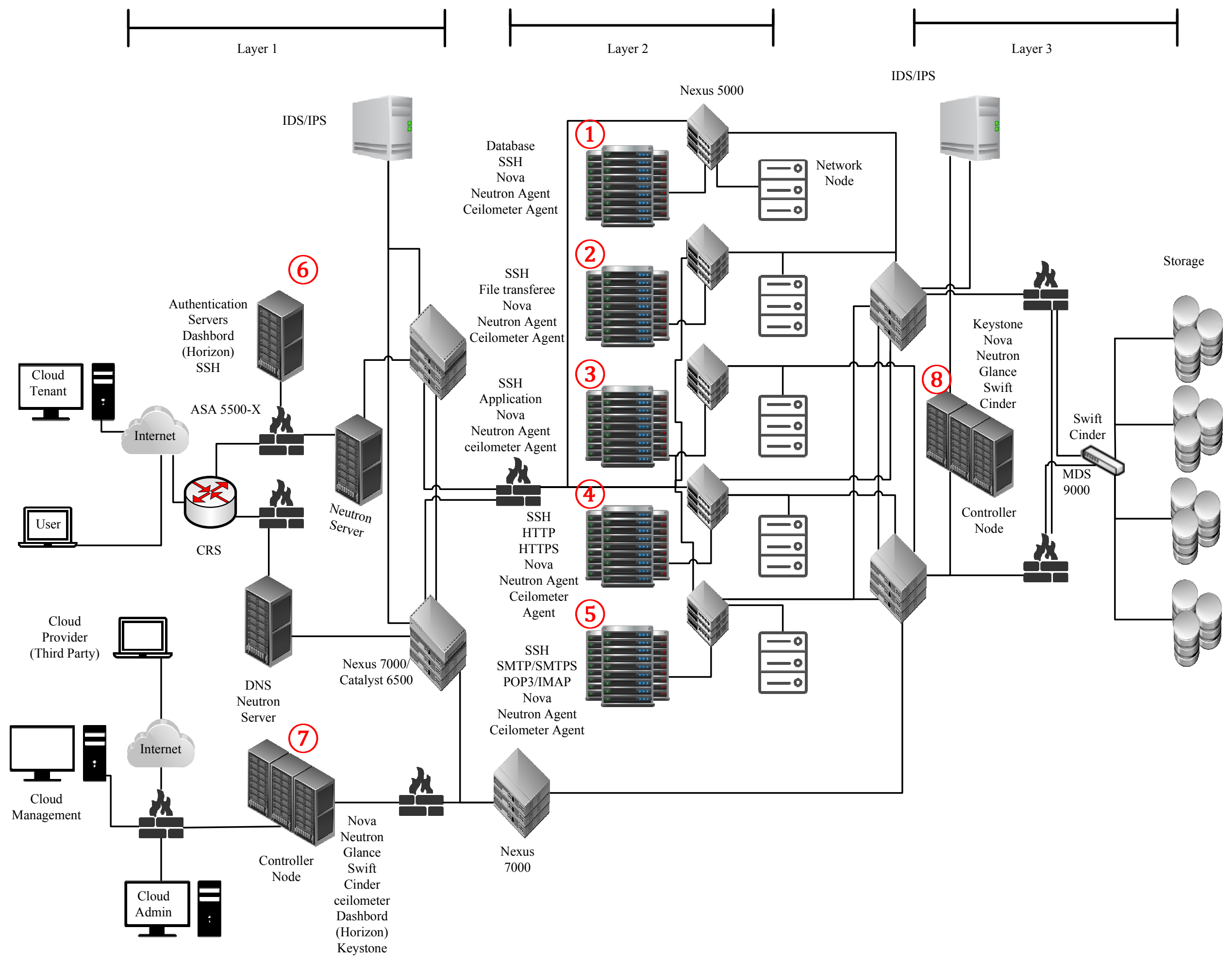


Figure 19: An example of cloud data center

Such a cloud data center may require many maintenance tasks to be routinely performed to ensure the normal operation of the hardware and software components. Such maintenance tasks may be performed by both internal staff working onsite, and remote administrators including those from specialized third party providers. In our example, assume the cloud provider decides to rely on third party remote administrators for the regular maintenance of the five compute nodes (nodes #1-5 in Figure 19), the authentication server (node #6), and the two controllers (nodes #7 and #8). As an example, Table 3 shows the maintenance tasks that need to be performed on those nodes. For simplicity, we only

Table 3: An example of required maintenance tasks

Node number (in Figure 19)	Maintenance tasks		
	Read log files	Modify configuration files	Install a new system
1	×	×	
2	×		×
3	×	×	×
4		×	×
5	×		×
6	×	×	
7	×		
8	×		

consider three types of tasks here (more discussions about maintenance tasks will be given in next section).

In such a scenario, the cloud provider naturally faces security challenges due to the fact that necessary privileges must be granted to allow the third party remote administrators to perform their assigned maintenance tasks. For instance, the task of reading log files needs certain read privilege to be granted, whereas modifying configuration files and installing a new system would demand much higher levels of privileges. Even though the cloud provider may (to some extent) trust the third party maintenance provider as an organization, the granted privileges may allow a dishonest remote administrator, or attackers with stolen credentials of a remote administrator, to launch an insider attack and cause significant damage to the cloud provider and its tenants. It is in the cloud provider's best interest to better understand and proactively mitigate such potential threats. However, this toy example is enough to demonstrate that there exist many challenges in modeling and mitigating such threats.

– First, as demonstrated in Table 3, there may exist complex relationships between maintenance tasks and corresponding privileges needed to fulfill such tasks, relationships between different privileges (e.g., a root privilege implies many other privileges), and dependency relationships between services or business functions and the underlying physical and virtual resources used to host such services or functions.

Those relationships will determine the extent of an insider threat.

– Second, the insider threat will also depend on which nodes in the cloud infrastructure are involved in the assigned tasks, e.g., an insider with privileges on the authentication servers (node #6 in Figure 19) or on the compute nodes (nodes #1-5) may have very different security implications.

– Third, the extent of the threat also depends on the configuration (e.g., the connectivity and firewalls), e.g., an insider having access to the controller node #8 would have a much better chance to compromise the storage servers than one with access to the other controller node #7).

– Finally, while an obvious way to mitigate the insider threat is through assigning less tasks to each remote administrator such as to limit his/her privileges, as our study will show, the effectiveness of such an approach depends on many other factors and constraints, e.g., the amount of tasks to be assigned, the number of available remote administrators, constraints like each administrator may only be assigned to a limited number of tasks due to availability, or a subset of tasks due to his/her skill set, etc.

Clearly, modeling and mitigating the insider threat of remote administrators may not be straightforward even for such a simplified example (the solution for this example scenario is given in Section 4.4.2), and the scenario is likely far more complex for real clouds than the one demonstrated here. The remainder of the chapter will present a systematic approach to tackle those challenges.

4.2.2 Remote Administrators, Maintenance Tasks, and Privileges

There exist different types of administrators in cloud data centers who perform maintenance tasks either onsite or through remote accesses [29]. For example, *hardware administrators* have physical access to the cloud data center to perform maintenance on the physical

Table 4: Maintenance tasks in popular cloud platforms

Maintenance Task	AWS [3]	GCP [4]	Azure [5]
Review Logs	×	×	×
Hard Disk Scan		×	×
Update Firmware	×	×	×
Patch Operating System	×	×	×
Update Operating System	×	×	×
System Backup	×	×	×
Upgrades System	×	×	×
Maintain Automated Snapshots	×		
Bug Fix	×	×	×
Update Kernel	×	×	

components. *Security team administrators* are responsible for maintaining the cloud security policies. *Remote administrators* (RAs) perform maintenance tasks on certain nodes of the infrastructure through network connections from remote sites. The first two types can be considered relatively more trustworthy due to their usually limited quantity and the fact they work onsite and directly for the cloud provider. The last type is usually considered riskier due to two facts, i.e., they work through remote accesses which are susceptible to attacks (e.g., via stolen credentials), and they may be subcontracted through third party companies, which means less control by the cloud provider. In this chapter, we focus on the security risk of such remote administrators (RAs), even though our models and mitigation solution may be adapted to deal with other types of administrators and users if necessary.

There exists only limited public information about the exact maintenance tasks performed by major cloud providers. We have collected such information from various sources, and our findings are summarized in Table 4, which shows sample maintenance tasks mentioned by Amazon Web Service [3], Google Cloud [4], and Microsoft Azure [5]. As to privileges required for typical maintenance tasks, Bleikertz et al. provided five sample privileges required for maintaining the compute nodes in clouds [29], which we will borrow for our further discussions, as shown in Table 5.

To simplify our discussions, our running example will be limited to ten maintenance tasks on three compute nodes with corresponding privileges on such nodes, as shown in

Table 5: Privileges used in this work

Privilege	Restriction
No privilege	No access
Read	Cannot read VM-related data
Write_L1	The restriction of read privilege applies, software modification restricted to trusted repository
Write_L2	Bootloader, kernel, policy enforcement, maintenance agent, file system snapshots, package manager transaction logs, and certain dangerous system parameters
Write_L3	No restriction

Table 6. Later in Section 4.4.2, we will expand the scope to discuss the solution for our motivating example which involves all the eight nodes.

Table 6: Maintenance tasks and privileges for the running example

Task Number	Node Number (in Figure 19)	Task Description	Privilege
1	4 ($http$)	Read log files for monitoring	Read
2	4 ($http$)	Modifying configuration files	Write_L1
3	4 ($http$)	Patching system files	Write_L3
4	3 (app)	Read log files for monitoring	Read
5	3 (app)	Modifying configuration files	Write_L1
6	3 (app)	Update kernel	Write_L3
7	1 (DB)	Read log files for monitoring	Read
8	1 (DB)	Modifying configuration files	Write_L1
9	1 (DB)	Update kernel	Write_L3
10	1 (DB)	Install new systems	Write_L2

4.3 Models

This section presents out threat model and the proposed models of the maintenance task assignment and insider threats.

4.3.1 The Threat Model and Maintenance Task Assignment Model

The in-scope threats we consider include insider attacks from dishonest remote administrators or attackers with stolen credentials of such administrators. Consequently, we assume the majority of remote administrators is trusted, and if there are multiple dishonest administrators (or attackers with their credentials), they do not collude (a straightforward extension of our models by considering each possible combination of administrators as one insider can accommodate such colluding administrators). The third party provider is considered trusted as an organization and it will collaborate with the cloud provider to implement the intended task assignment. The cloud provider is concerned about certain critical assets, such as physical or virtual resources and services or business functions, inside the cloud, and it is aware of the constraints about task assignments such as the number of remote administrators, their availability and skill set, etc. Finally, as a preventive solution, our mitigation approach is intended as a complementary solution to existing vulnerability scanners, intrusion detection systems, and other prevention or mitigation solutions.

The cloud provider assigns the maintenance tasks to remote administrators (RAs) based on given constraints (e.g., which tasks may be assigned to each RA), and consequently the RA will obtain privileges required by those tasks. This can be modeled as follows (which has a similar syntax as [118]).

Definition 1 (Maintenance Task Assignment Model)**.** *Given*

- *a set of remote administrators RA,*
- *a set of maintenance task T,*
- *a set of privileges P,*
- *the remote administrator task relation $RAT \subseteq RA \times T$ which indicates the maintenance tasks that are allowed to be assigned to each remote administrator, and*
- *the task privilege relation $TP \subseteq T \times P$ which indicates the privileges required for each task,*

a maintenance task assignment is given by function $ta(.) : RA \to 2^T$ *that satisfies* $(\forall ra \in RA)(ta(ra) \subseteq \{t \mid (ra,t) \in RAT\}$ *(meaning a remote administrator is only assigned with the tasks to which he/she is allowed), and the corresponding set of privileges given to the remote administrator is given by function* $pa(ra) = \bigcup_{t \in ta(ra)} \{p \mid (t,p) \in TP\}$.

4.3.2 The Insider Threat Model

To model various resources and their relationships in cloud infrastructures, we borrow the resource graph concept [150, 144] to represent hardware hosts (e.g., servers and networking devices), software resources (e.g., network services and applications) running on such hosts (only remotely accessible resources are considered), and the causal relationships between different resources (e.g., a zero day exploit on the Web server may lead to user privilege on that server which subsequently causes the application server to be accessible). This concept is more formally stated in Definition 2 and will be illustrated through an example.

Definition 2 (Resource Graph [150, 144])**.** *Given a network with the set of hosts* H*, the set of resources* R*, with the resource mapping* $res(.) : H \to 2^R$*, the set of zero day exploits* $E = \{\langle r, h_s, h_d \rangle \mid h_s \in H, h_d \in H, r \in res(h_d)\}$ *and their pre- and post-conditions* C*, a resource graph is a directed graph* $G(E \cup C, R_r \cup R_i)$ *where* $R_r \subseteq C \times E$ *and* $R_i \subseteq E \times C$ *are the pre- and post-condition relations, respectively.*

To quantify the insider threat of remote administrators based on resource graphs, we extend the *k*-zero day safety security metric [138, 150, 139]. Roughly speaking, the metric starts with the worst case assumption that the relative severity of unknown (zero day) vulnerabilities are not measurable; it then simply counts how many different resources must be compromised through such unknown vulnerabilities in order to compromise a given critical asset; a larger count will indicate a relatively more secure network, since the likelihood of having more unknown vulnerabilities all available at the same time, inside the same network, and exploitable by the same attacker, would be significantly lower. The following provides a simplified version of this concept, which will be illustrated through an example.

Definition 3 (Attack Path and *k*-Zero Day Safety [138, 150, 139])**.** *Given a resource graph $G(E \cup C, R_r \cup R_i)$, we call $C_I = \{c : c \in C, (\nexists e \in E)(\langle e,c \rangle \in R_i)\}$ the set of initial conditions; we call any sequence of zero day exploits $e_1, e_2, \ldots, e_n$ an attack path if all the pre-conditions of each e_i are either initial conditions, or post-conditions of some $e_j (j < i)$. For any given critical asset $c \in C$, we say the network is k-zero day safe if there does not exist any attack path which involves k or less distinct resources, and includes at least one exploit having c as its post-condition.*

Figure 20 shows an example resource graph for our running example (the dashed lines and shades represent our extension to the model, which can be ignored for now and will be discussed later in Section 4.4.2; also, only a small portion of the resource graph is shown here due to space limitations). Each triple inside an oval indicates a potential zero day or known exploit in the format <service or vulnerability, source host, destination host> (e.g. <Xen, RA, 4> indicates an exploit of Xen on host 4 from host RA), and the plaintext pairs indicate the pre- or post-conditions of those exploits in the format <condition, host> where a condition can be either a privilege on the host (e.g., <W1,4> means the level 1 write privilege and <R,4> means the read privilege which are both explained in Section 4.2.2), the existence of a service on the host (e.g., <Xen,4>), or a connectivity (e.g., <0,4>means attacker can connect to host 4 and <4,4> means a local exploit on host 4). The edges point from pre-conditions to an exploit and then to its post-conditions, which indicate that any exploit can be executed if and only if all of its pre-conditions are satisfied, whereas executing an exploit is enough to satisfy all its post-conditions.

In Figure 20, the left-hand side box indicates the normal resource graph which depicts what an external attacker may do to compromise the critical asset <user, Xen>. The right-hand side boxes depict the insider threats coming from RAs assigned to each of the three

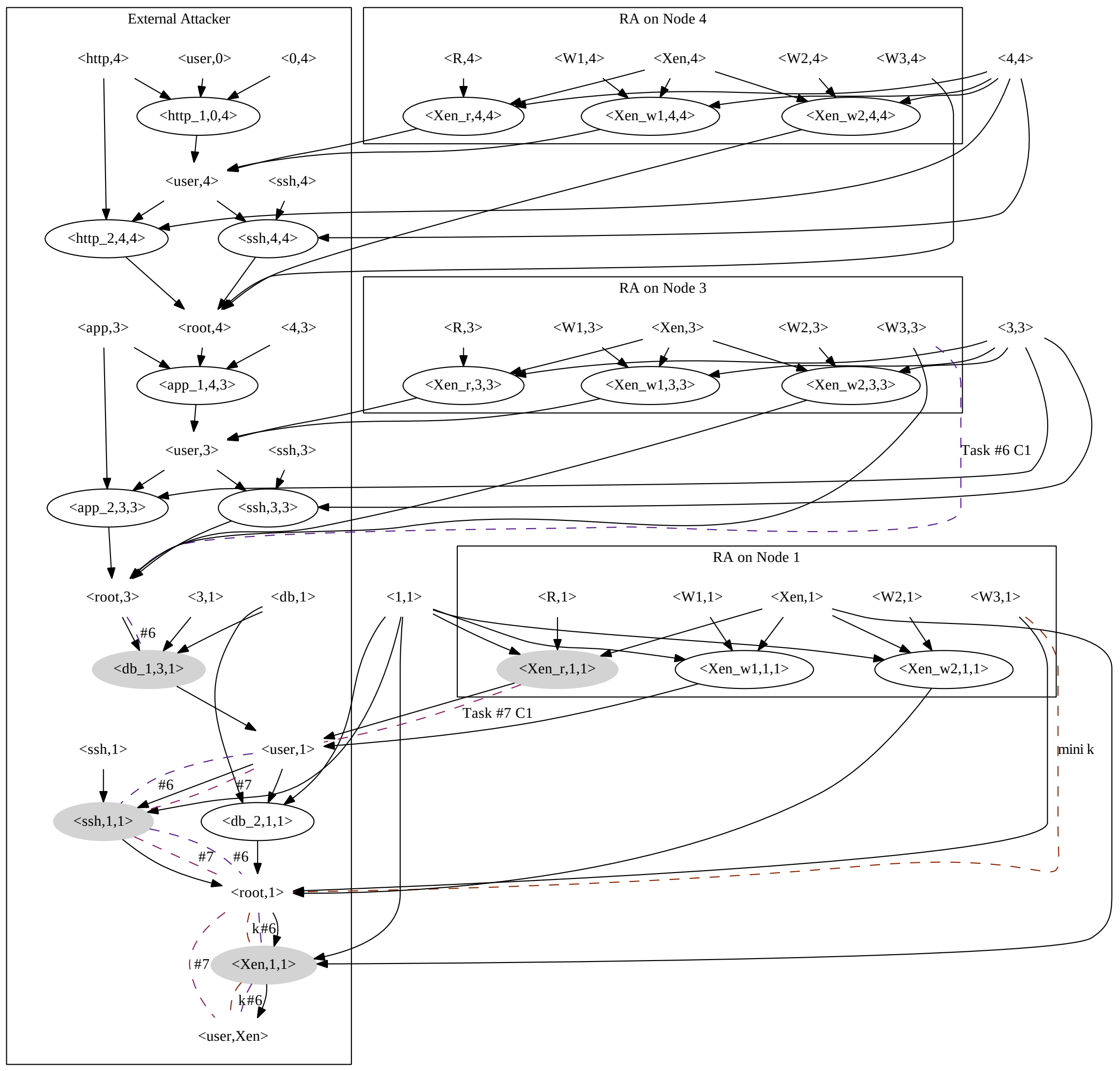


Figure 20: Modeling insider threat using the resource graph

compute nodes. The gray color exploits are what captures the consequences of granting privileges to remote administrators. For example, an RA with the level 1 write privilege <W1,4> can potentially exploit Xen (i.e., <Xen_w1,4,4>) to escalate his/her privilege to the user privilege on host 4 (i.e.., <user,4>), whereas a higher level privilege <W2,4> can potentially lead to the root privilege <root,4> through an exploit <Xen_w2,4,4>, and the highest privilege <W3,4> can even directly lead to that privilege. Those examples show how the model can capture the different levels of insider threats as results of different privileges obtained through maintenance task assignments.

Next, given the maintenance task assignment for each RA, we can obtain all the possible paths he/she may follow in the resource graph, starting from all the initially satisfied conditions (e.g., <Xen,4>) and those implied by the task assignment (e.g., <W1,4>) to the critical asset (i.e., <user,Xen>). To quantify the relative level of such threats, we apply the *k*-zero day safety metric (*k*0d) mentioned above. The metric value of each RA provides an estimation for the relative level of threat of each RA, since a larger number of distinct zero day exploits on the shortest path means reaching the critical asset is (exponentially, if those exploits are assumed to be independent) more difficult. For example, an RA with privilege <W3,1> would have a *k*0d value of 1 since only one zero day exploit <Xen,1,1> is needed to reach the critical asset, whereas an RA with <W2,1> would have a *k* value of 2 since an additional exploit <Xen_w2,1,1> is needed. Finally, once we have calculated the *k* values of all RAs based on their given maintenance task assignments, we take the average (and minimum) of those *k* values as the average (and worst) case indication of the overall insider threat of the given maintenance task assignments. The above discussions are formally defined as follows.

Definition 4 (Insider Threat Model)**.** *Given the maintenance task assignment (i.e., RA, T, P, RAT, TP, ta, and pa, as given in Definition 1) let* $C_r = \bigcup_{ra \in RA} pa(ra)$ *be the set of privileges implied by the assignment and* E_r *be the set of new exploits enabled by* C_r*. Denote by* $G(E \cup E_r \cup C \cup C_r, R)$ *the resource graph (where E and C denote the original set of exploits and conditions, respectively, and R denote the edges) and let* $k0d(.)$ *be the k zero day safety metric function. We say* $k0d(ra)$, $\frac{\sum_{ra \in RA} k0d(ra)}{|RA|}$, *and* $min(\{k0d(ra) : ra \in RA\})$ *represent the insider threat of ra, the average case insider threat of the maintenance task assignment, and the worst case insider threat of the maintenance task assignment, respectively.*

4.3.3 The Bayesian Network Model

The previous section has applied the *k*-zero day safety metric to model the insider threat of remote administrators. This is a conservative model since the *k* value is defined based on the shortest attack paths, which attacker may or may not be able to follow in practice. Moreover, the model only considers zero day exploits and known vulnerabilities do not contribute to the *k* value. In this section, we extend this model by applying the Bayesian network (BN)-based metric [57] instead of the *k*-zero day safety.

The BN-based metric is based on the conditional probability of reaching the given critical assets given that all initial conditions are satisfied. We first construct a Bayesian network based on the resource graph and the conditional probability that each exploit can be executed given its pre-conditions are all satisfied. Such conditional probabilities can be assigned to both known vulnerabilities based on standard vulnerability scores (e.g., the CVSS scores [95]), and zero day exploits based on a nominal value (e.g., 0.08 [100]). Therefore, the model captures both zero day and known vulnerabilities, and it also takes all attack paths into consideration. Finally, the model can also capture additional casual dependencies, e.g., the same vulnerability appearing on multiple hosts may yield a higher probability (e.g., 0.9 in our examples).

By applying the BN-based metric to the resource graph given in Definition 4, we can obtain the probability for each RA to compromise the given critical assets given all the privileges implied by a maintenance task that is assigned to the RA. Since an RA may be assigned to multiple maintenance tasks, the RA can compromise the critical assets as long as at least one of the assigned tasks enables him/her to do so whose probability can be computed as in Equation 1. We redefine the insider threat model based on those discussions in Definition 5. The model also allows assigning the relative likelihood of each RA to be misbehaving, which can be estimated either based on the background (e.g., third party RAs should be assigned a higher probability than RAs of the cloud provider) or behavior-based detection results if available.

Definition 5 (The BN-based Insider Threat Model)**.** *Given the maintenance task assignment (i.e., RA, T, P, RAT, TP, ta, and pa), let* $C_r = \bigcup_{ra \in RA} pa(ra)$ *be the set of privileges implied by the assignment and* E_r *be the set of new exploits enabled by* C_r*. Denote by* $G(E \cup E_r \cup C \cup C_r, R)$ *the resource graph (where E and C denote the original set of exploits and conditions, respectively, and R denote the edges) and let* $BN = (G, \theta)$ *be a Bayesian network where* θ *denotes the BN parameters. Let* $P_{BN}(t)$ *be the conditional probability that an RA assigned with task t can compromise the given critical assets, and* $P_M(ra)$ *the given probability that ra will misbehave. We say* $P(ra)$, $\frac{\sum_{ra \in RA} P(ra)}{|RA|}$, *and* $min(\{P(ra) : ra \in RA\})$ *represent the insider threat of ra, the average case insider threat of the maintenance task assignment, and the worst case insider threat of the maintenance task assignment, respectively, where*

$$P(ra) = 1 - [\prod_{t \in T, (ra,t) \in RAT} (1 - P_{BN}(t) \cdot P_M(ra))] \tag{1}$$

4.3.4 The Service Dependency Model

The insider threat models introduced in previous sections are based on resource graphs, which are mainly designed to model hardware and software resources. The resource graphs, however, do not directly indicate any higher level services or business functions, the relationships between such services or functions, or their dependencies on the underlying hardware and software resources. For this purpose, the concept of service dependency graph [131, 8] has been proposed to model security impact on services [38]. For example, Figure 21 demonstrates an example in which the lower figure shows an attack graph (which is syntactically equivalent to a resource graph but designed for exploits of known vulnerabilities) depicting various exploits and their relationships, and the upper figure shows the service dependency graph depicting various services; the dashed line edges show the dependencies between the services and corresponding resources involved in the vulnerabilities. The service dependency graph and the attack graph can be integrated and flattened as an extended model. This model can be used to identify attack paths exploiting services or leading to critical assets given as services.

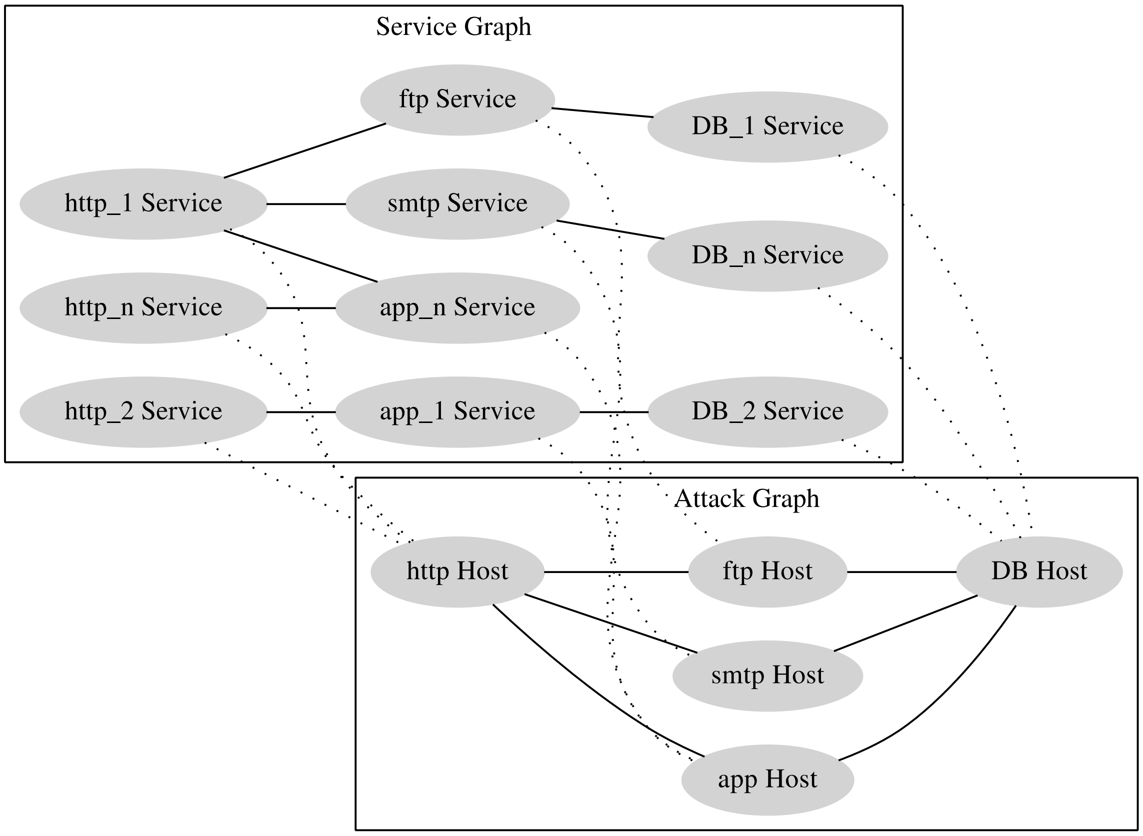


Figure 21: An example of service dependency graph

We apply the service dependency model [131] to extend our insider threat models introduced in previous sections. For example, in Figure 22, the left side of Figure 22 shows examples of service dependency-related exploits which are integrated into the previous resource graph. Each triple inside a shadowed oval indicates a service dependency exploit, and the plaintext nodes in between shadowed ovals indicate the type of impacted service, which can run on either virtual or physical resources; the dash line shows the dependency between services. This model can be used to represent various causal relationships between services and resources, e.g., all services running on top of a server may be compromised if attackers gain full control over the server, a server (and other services) may not be affected when one of the services running on the server is compromised, and a service involving multiple resources may be compromised either when one of the resources is compromised (e.g., a Web service may become unavailable if either the Web, application, or database server is down) or multiple resources are compromised at the same time (e.g., a Web service might be supported by multiple redundant Web servers).

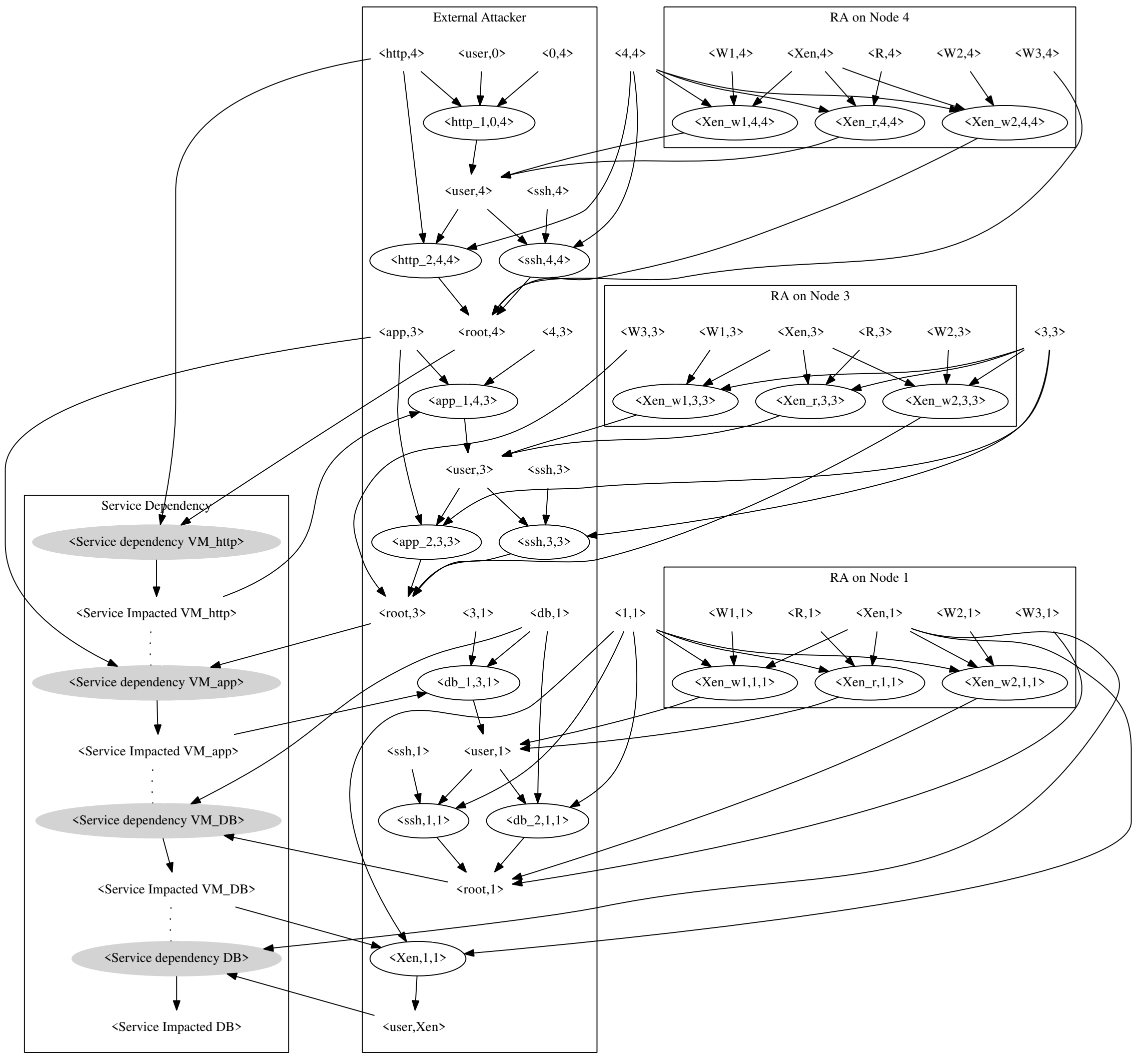


Figure 22: Modeling insider threats using service dependency graph

We formalize the service dependency resource graph concept in Definition 6. The model extends the resource graph by adding nodes for services and their pre- and post-conditions, edges connecting services to those conditions, and edges inter-connecting the services (or their pre- and post-conditions) and the pre- and post-conditions of exploits (or the exploits). Definition 7 then extends the previous insider threat models based on the service dependency resource graph. We will apply this model in the upcoming sections to study the solution for mitigating the insider threats of remote administrators, and to conduct simulations to evaluate the effectiveness of the solution. In practice, the choice between those different models (e.g., k-zero day safety versus BN, or whether to consider service dependencies) will depend on the needs of specific applications and the available information or assumptions.

Definition 6 (Service Dependency Resource Graph)**.** *Given a network with the set of hosts H, the set of resources R, with the resource mapping $res(.) : H \to 2^R$, the set of zero day exploits $E = \{\langle r, h_s, h_d \rangle \mid h_s \in H, h_d \in H, r \in res(h_d)\}$ and their pre- and post-conditions C, the set of services S, their pre- and post-conditions C_s, a service dependency resource graph is a directed graph $G(E \cup C \cup S \cup C_s, R_r \cup R_i$ where $R_r \subseteq (C \cup C_s) \times (E \cup S)$, $R_i \subseteq (E \cup S) \times (C \cup C_s)$ are the dependency relations.*

Definition 7 (The Service Dependency-based Insider Threat Model)**.** *Given the maintenance task assignment (i.e., RA, T, P, RAT, TP, ta, and pa), let $C_r = \bigcup_{ra \in RA} pa(ra)$ be the set of privileges implied by the assignment and E_r be the set of new exploits enabled by C_r. Denote by $G(E \cup S \cup C \cup C_s \cup E_r \cup C_r, R)$ the service dependency resource graph (where E, S, C, and C_s denote the original set of exploits, services, and conditions, respectively, and R denote the edges). We say $k0d(ra)$, $\frac{\sum_{ra \in RA} k0d(ra)}{|RA|}$, and $min(\{k0d(ra) : ra \in RA\})$ (or $P(ra)$, $\frac{\sum_{ra \in RA} P(ra)}{|RA|}$, and $min(\{P(ra) : ra \in RA\})$ in the case of BN-based metrics) the insider threat of ra, the average case insider threat of the maintenance task assignment, and the worst case insider threat of the maintenance task assignment,* *respectively.*

4.4 The Mitigation and Use Cases

In this section, we formulate the optimization-based solution for mitigating the insider threat of remote administrators during maintenance task assignment.

4.4.1 The Optimization-based Mitigation

Based on our definitions of the maintenance task assignment model and the insider threat model, we formulate the problem of optimal task assignment as follows

$$\begin{aligned} \bar{k} = maximizes \frac{\sum_{ra \in RA} k0d(ra)}{\mid RA \mid} \\ subject\ to \qquad ta(ra) \in RAT \end{aligned} \tag{2}$$

$$\begin{aligned} or \qquad minimum k = maximizes (min(\{k0d(ra) : ra \in RA\})) \\ subject\ to \qquad ta(ra) \in RAT \end{aligned} \tag{3}$$

where Equation 2 shows the average of maximizing k and Equation 3 shows the result of maximizing the worst k. The $k0d$ computes the shortest path on a given resource graph G, the set of remote administrators RA, maintenance tasks T, privileges P, the remote administrator task relation RAT, and the task privilege relation TP. The remote administrator task relation RAT basically defines the optimization constraints since it states which tasks may be assigned to which RA. Additional constraints in other forms may also be introduced, e.g., the maximum number of tasks that can be assigned to an RA.

Theorem 1. *The Optimal Task Assignment Problem is NP-hard.*

Proof: First, evaluating the $k0d$ function is already NP-hard w.r.t. the size of the resource graph [138]. On the other hand, we provide a sketch of proof to show the problem is also

NP-hard from the perspective of the maintenance task assignment. Specifically, given any instance of the well known NP-complete problem, *exact cover by 3-sets* [78] (i.e., given a finite set X containing exactly $3n$ elements, and a collection C of subsets of X each of which contains exactly 3 elements, determine whether there exists $D \subseteq C$ such that every $x \in X$ occurs in exactly one $d \in D$), we can construct an instance of our problem as follows. We use X for the set of maintenance tasks, and C for the set of RAs, such that the three elements of each $c \in C$ represent three tasks which can be assigned to c. In addition, no RA can be assigned with less than three tasks, and an RA already assigned with three tasks can choose any available task to be assigned in addition. We can then construct a resource graph in which the critical asset can be reached through any combination of four privileges. It then follows that, the k value for insider threat is maximized if and only if there exists an exact cover D due to the following. If the exact cover exists, then every RA $d \in D$ is assigned with exactly three tasks and therefore the k value of every RA will be equal to infinity since the critical asset cannot be reached with less than four privileges; if the cover does not exist, then to have every task assigned, we will have to assign at least one RA with more than three tasks, and hence the k value will decrease. □

In our study, we use the genetic algorithm (GA) [60] to optimize the maintenance task assignments by maximizing k. Specifically, the resource graph is taken as input to the optimization algorithm, with the (either average case or worst case) insider threat value k as the fitness function. We try to find the best task assignment for maximizing the value k within a reasonable number of generations. The constraints can be given either through defining the remote administrator task relation RAT in the case of specific tasks that can be assigned to each RA, or as a fixed number of tasks for each RA. Other constraints can also be easily added to the optimization problem. In our simulations, we choose the probability of 0.8 for crossover and 0.2 for mutation based on our experiences.

4.4.2 Use Cases

We demonstrate our solution through several use cases with different constraints. The first three use cases are based on the five remote administrators and ten maintenance tasks presented in Table 6 and the fourth use case is based on the motivating example shown in Section 4.2.1. The last use case is based on the service dependency resource graph with three remote administrator and five maintenance tasks.

Table 7: Maintenance tasks assignments for use case A

User	A_1	B_1	C_1	D_1	E_1	A_2	B_2	C_2	D_2	E_2	A_3	B_3	C_3	D_3	E_3
Tasks Number	4	5	6	8	9	6	4	7	8	5	4	5	6	8	9
	1	10	7	3	2	9	3	10	1	2	1	2	7	3	10
k	3	1	2	2	1	1	3	1	2	3	3	3	2	2	1
$\bar{k}$			1.8					2					2.2		
Minimum k			1					1					1		

Use Case A: In this case, each RA should be assigned with exactly two tasks (e.g., to evently distribute the tasks among all the RAs). The three tables shown in Table 7 show three possible assignments and the corresponding k values. Also, Figure 20 shows an example path (dashed lines) for tasks assigned to RA C_1 based on the left table, and also the shortest path yielding the minimum k value. We use the GA to find the optimal task assignment that meets the constraint given in this case, as shown in the right table, the maximal average of k values among all RAs is $\bar{k} = 2.2$. It can also be seen that the minimum k value among all RAs is always $k = 1$ in this special case.

Use Case B: In this case, each RA should be assigned with at least one task (e.g., to ensure all RAs are employed while there is no consideration for their workload). The optimal task assignment under this constraint is (RA1{8,9,10}, RA2{4,5}, RA3{3}, RA4{1,2}, and RA5 {6,7}). This relaxed constraint improves the average k from 2.2 in the previous example to 2.8, which shows relaxing the constraint may increase k (which means less threat).

Use Case C: In this case, each RA can handle a fixed subset of tasks (e.g., due to the level of training or skill). In our example, we assume RA1 can be assigned to any task requiring the read privilege, RA2 to tasks requiring write level 1 privilege, RA3 to tasks requiring write level 1 and 2, RA4 to tasks requiring write level 3, and RA5 can be assigned to any task. After applying our solution, the optimal assignment yields the maximal average of k values to be $k = 2.2$.

Use Case D: This case shows the optimal maintenance task assignment for tasks discussed in our motivating example in Section 4.2.1. We have eight RAs and each RA can handle maximum two tasks. The upper table in Table 8 shows the 15 maintenance tasks to be assigned. In Table 8, the four tables on the bottom show four different scenarios of tasks assigned to RAs and each table shows different average k. The bottom table on the right shows the optimal task assignment in term of the average $k = 3.125$. In Figure 23, the red dashed line represents the path used by RA1 to reach the critical asset when task number 1 is assigned to RA1. Also, the solid red line shows the path when task number 12 is assigned to RA1.

Use Case E: In this use case, we demonstrate how service dependencies may affect the task assignment. We have five maintenance tasks as presented in Table 9. Assume all VMs running on the http compute node have backups but some VMs running on the app compute node do not have a backup. The critical asset is given as the DB service. We have three RAs each of which can be assigned with a maximum of two maintenance tasks. Table 10 shows two possible ways to assign the maintenance tasks to RAs and the corresponding k values. We use the GA to find the optimal task assignment that satisfies the constraints given in this case. As shown in the right table, the maximal average of k values among all RAs is $\bar{k} = 2.3$. It can also be seen that the minimum k value among all RAs is always $k = 2$ in this special case. Figure 24, shows the service dependency graph. The shadowed

Table 8: Maintenance task assignments for use case D (the motivating example)

Task#	Maintenance task	Task#	Maintenance task
1	Read log files for node 1	2	Modify configuration file for node 1
3	Read log files for node 2	4	Install a new system for node 2
5	Read log files for node 3	6	Modify configuration file for node 3
7	Install a new system for node 3	8	Modify configuration file for node 4
9	Install a new system for node 4	10	Read log files for node 5
11	Install a new system for node 5	12	Read log files for node 6
13	Modify configuration file for node 6	14	Read log files for node 7
15	Read log files for node 8		

User	RA1	RA2	RA3	RA4	RA5	RA6	RA7	RA8
Tasks Number	14	1	4	8	2	3	7	6
	5	9	15	12	10	11	13	
k	1	3	2	3	2	3	2	3
$\bar{k}$				2.375				
Minimum k				1				

User	RA1	RA2	RA3	RA4	RA5	RA6	RA7	RA8
Tasks Number	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	
k	3	2	3	3	3	1	2	5
$\bar{k}$				2.75				
Minimum k				1				

User	RA1	RA2	RA3	RA4	RA5	RA6	RA7	RA8
Tasks Number	1	2	3	5	6	15	13	8
	4	7	9	10	11	12	14	
k	3	2	4	4	3	2	1	5
$\bar{k}$				3				
Minimum k				1				

User	RA1	RA2	RA3	RA4	RA5	RA6	RA7	RA8
Tasks Number	1	2	3	5	6	14	4	8
	12	7	9	10	11	15	13	
k	3	2	4	4	3	1	3	5
$\bar{k}$				3.125				
Minimum k				1				

oval represents the path followed by the attacker to compromise the service when the app VM (VMb) does not have a backup.

4.5 Simulations

This section shows simulation results on applying our mitigation solution under various constraints.

Experimental Settings All simulations are performed using a virtual machine equipped with a 3.4 GHz CPU and 4GB RAM in the Python 2.7.10 environment under Ubuntu 12.04 LTS and the MATLAB R2017b's GA toolbox. To generate a large number of resource

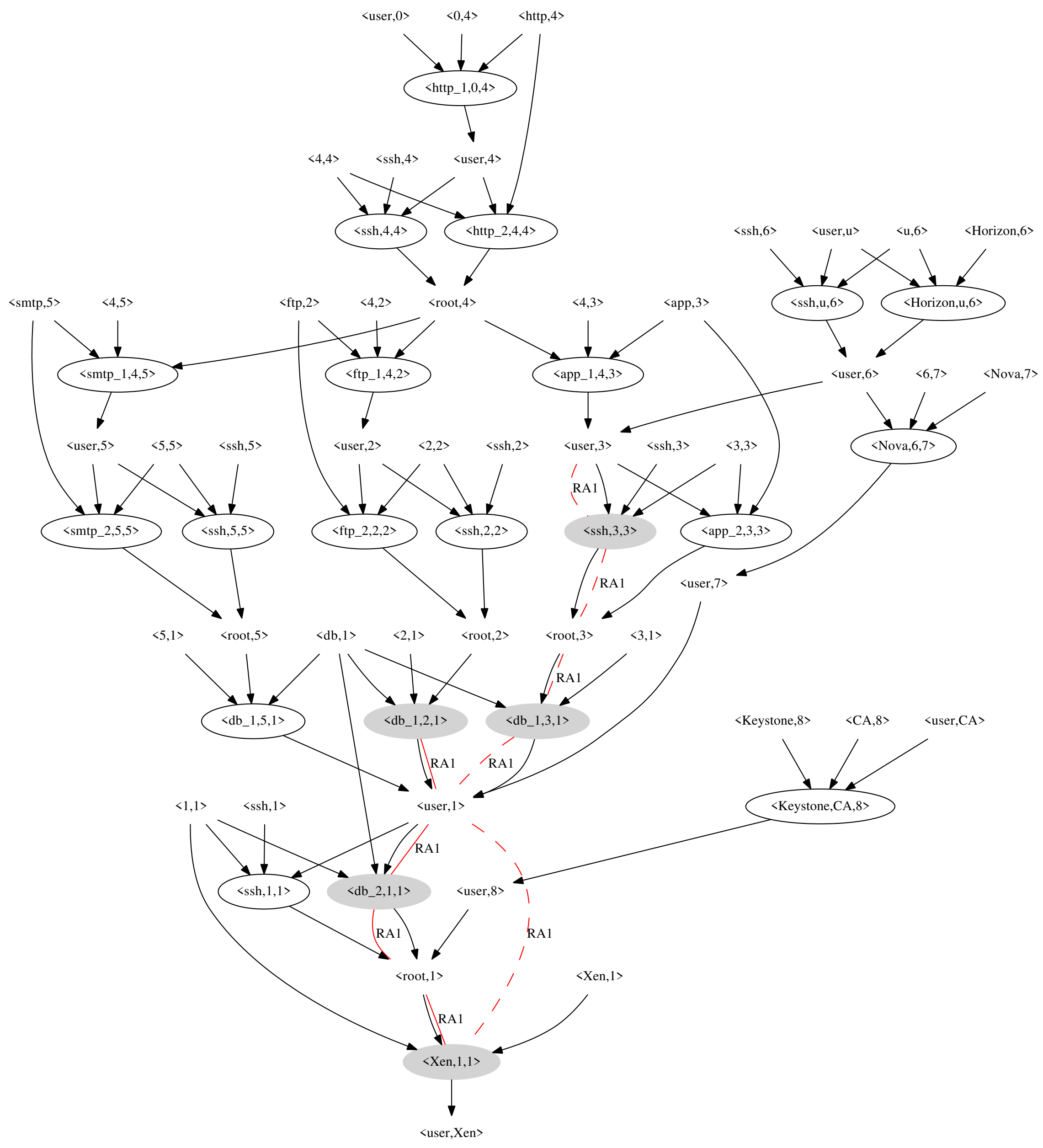


Figure 23: Resource graph for the motivating example

graphs and service dependency graphs for simulations, we start with seed graphs with realistic configurations similar to Figure 19 and then generate random resource graphs and service dependency graphs by injecting new nodes and edges into those seed graphs. Those resource graphs and service dependency graphs were used as the input to the

Table 9: Maintenance tasks and privileges for the service dependency

Task Number	Node Number (in Figure 19)	Task Description	Privilege
1	4 (*http*)	Read log files for monitoring	Read
2	4 (*http*)	Install new systems	Write_L2
3	3 (*app*)	Read log files for monitoring	Read
4	3 (*app*)	Install new systems	Write_L2
5	1 (*DB*)	Read log files for monitoring	Read

Table 10: Maintenance tasks assignments for use case E

User	A_1	B_1	C_1	A_2	B_2	C_2
Tasks Number	1 4	2 5	3	1 2	3 4	5
k	2	2	2	3	2	2
$\bar{k}$		2			2.33	
Minimum k		2			2	

optimization toolbox where the fitness function is to maximize the average or worst case insider threat values (given in Definition 4 and Definition 7); also, we used the optimization toolbox where the fitness function is to minimize the probability of reaching the critical asset by using the BN-based metric (given in Eq. 1) with various constraints, e.g., the number of available RAs and maintenance tasks, how many task may be assigned to each RA, assigning a fixed number of RAs with specific privilege, and assigning some of the maintenance tasks to the local administrators. We repeat each simulation on 300 different resource graphs to obtain the average result.

The Average Case Insider Threats The objective of the first two simulations is to study how the average case insider threat (i.e., the average of k values among all RAs) may be improved through our mitigation solution under constraints on the number of tasks and RAs, respectively. In Figure 25, the number of available RAs is fixed at 500, while the number of maintenance tasks is varied between 500 and 2,000 along the X-axis. The Y-axis shows the average of k values among all RAs. The solid lines represent the results after applying our mitigation solution under constraints about the maximum number of tasks assigned to each RA. The dashed lines represent the results before applying the mitigation

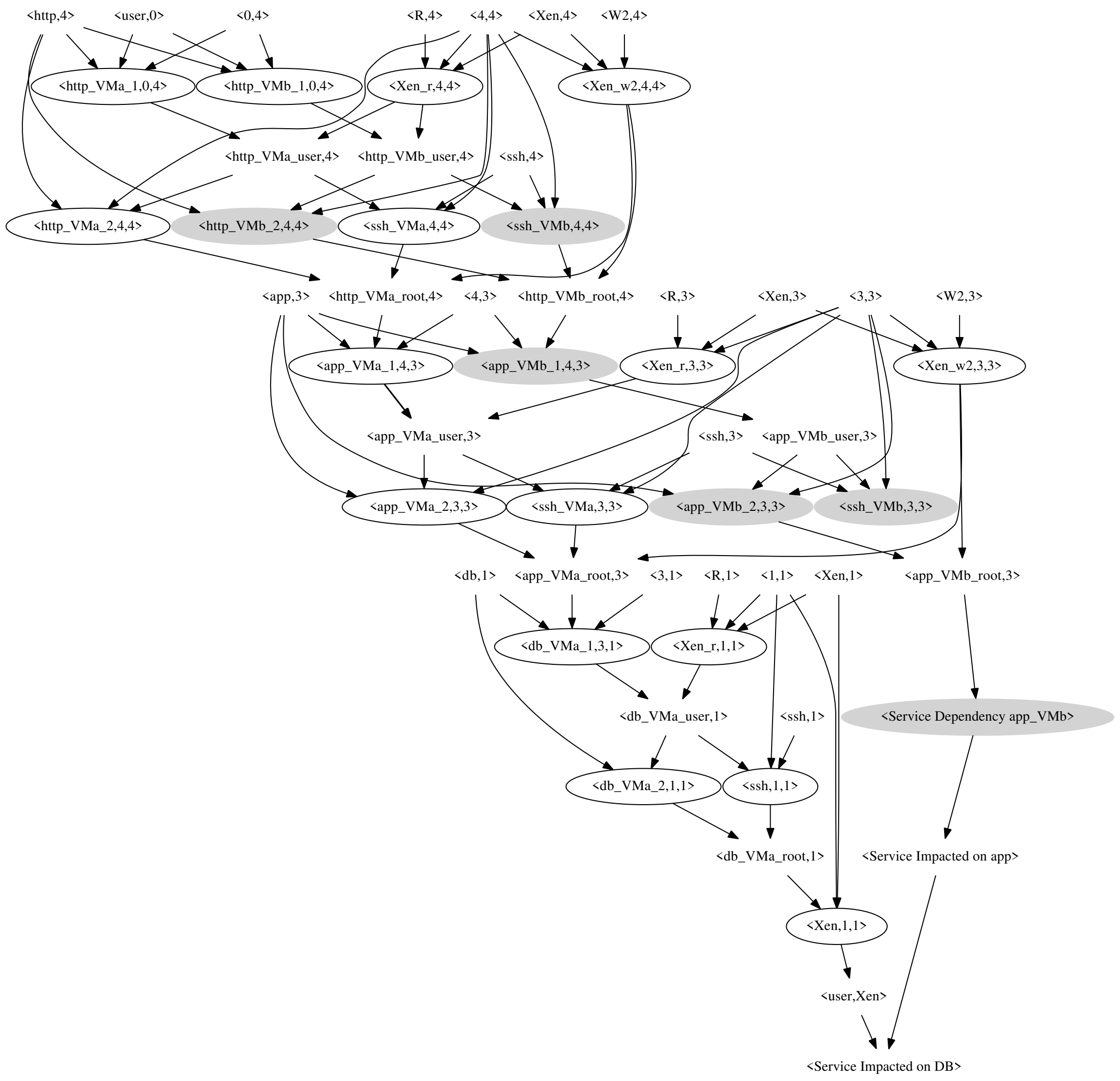


Figure 24: The service dependency resource graph for use case E

solution.

Results and Implications: From the result, we can make the following observations. First, the mitigation solution successfully reduces the insider threat (increasing the average of *k* values) in all cases. Second, the results before and after applying the solution decrease (meaning increased insider threat) following similar linear trends, as the number of maintenance tasks increases until each RA reaches its full capacity. Finally, the result of

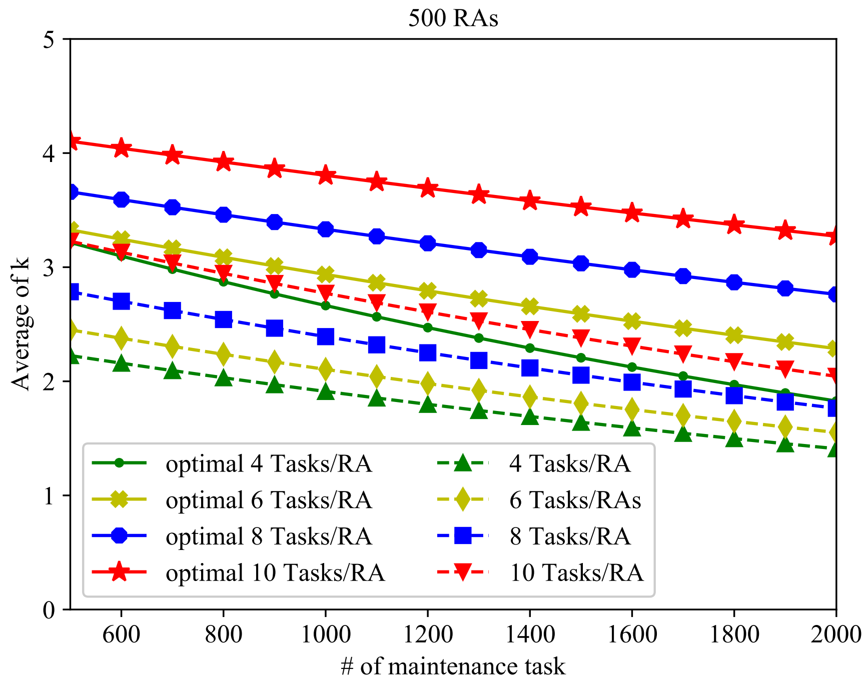


Figure 25: The average k among 500 RAs before and after applying the mitigation solution

maximum four tasks per RA after applying the solution is close to the result of maximum ten tasks per RA before applying the solution, which means the mitigation solution may allow more (more than double) tasks to be assigned to the same number of RAs while yielding the same level of insider threat.

In Figure 26, the number of maintenance tasks is fixed at 2,500 while the number of RAs is varied between 400 and 1,000 along the X-axis. The Y-axis shows the average of k values among all RAs. The solid lines represent the results after applying the mitigation solution and the dashed lines for the results before applying the solution. All the lines start with sufficient numbers of RAs for handling all the tasks since we only consider one round of assignment. We apply the same constraint as in previous simulation.

Results and Implications: Again, we can see the mitigation solution successfully reduces the insider threat (increasing the average of k values) in all cases. More interestingly, we can observe the trend of the lines as follows. The dashed lines all follow a similar near linear trend, which is expected since a larger number of RAs means less insider

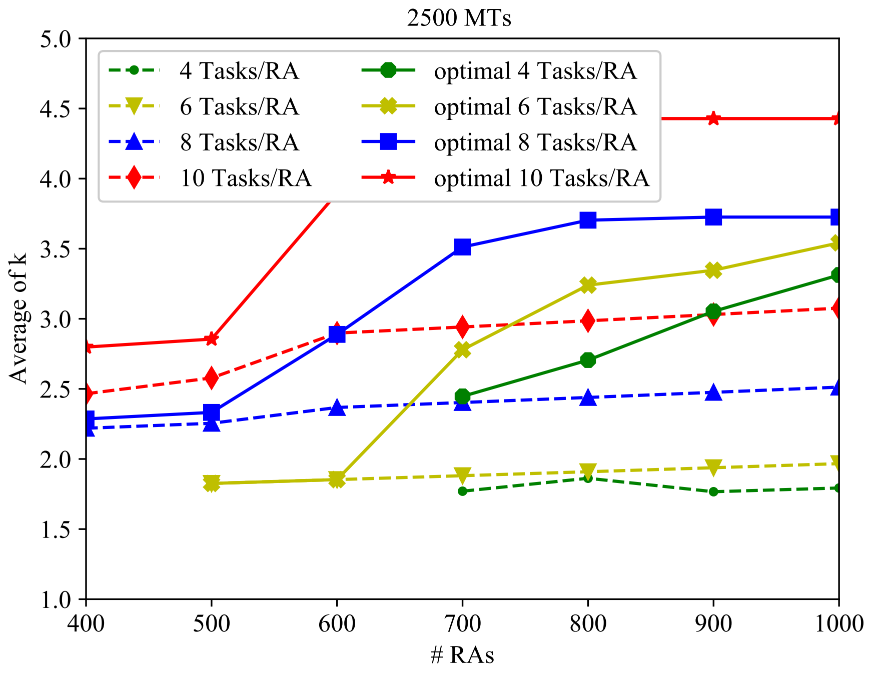


Figure 26: The average k among different number of RAs before and after the solution

threat since each RA will be assigned less tasks and hence given less privileges. On the other hand, most of the solid lines follow a similar trend of starting flat then increasing almost linearly before reaching the plateau. This trend indicates that, the mitigation solution can significantly reduce the insider threat when the number of RAs is within certain ranges past which it becomes less effective (because each RA already receives minimum privileges). The trend of four tasks per RA is slightly different mostly due to the limited number of RAs.

The Worst Case Insider Threats The objective of the next two simulations is to study how the worst case insider threat (i.e., the minimum k values among all RAs) behaves under the mitigation solution. Figure 27 and Figure 28 are based on similar X-axis and constraints as previous two simulations, whereas the Y-axis shows the minimum k among all RAs (averaged over 300 simulations).

Results and Implications: In Figure 27, we can see that the minimum k values also

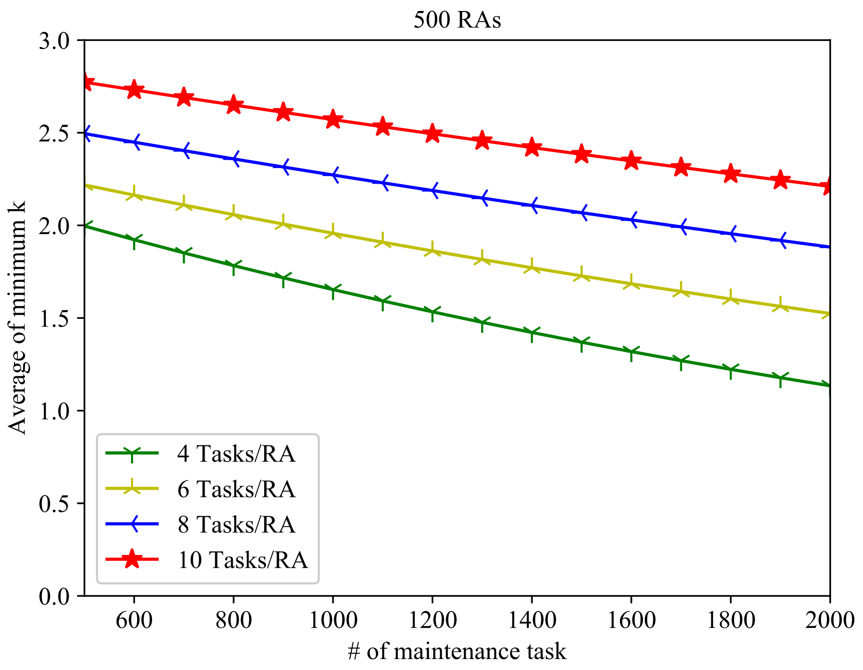


Figure 27: The minimum *k* for 500 RAs

decrease (meaning more insider threat) almost linearly as the number of tasks increases. In contrast to previous simulation, we can see the minimum *k* values are always lower than the average *k* values, which is expected. In Figure 28, we can see the minimum *k* values also increase almost linearly before reaching the plateau as the number of RAs increases. In contrast to previous simulation, we can see the increase here is slower, which means the worst case results (minimum *k* values) are more difficult to improve with a increased number of RAs. Also, we can see that the worst case results reach the plateau later (e.g., 900 RAs for 8 tasks per RA) than the average case results (700 RAs).

The Impact of the Highest Privileges The objective of the next two simulations is to study how the average case insider threat (i.e., the average of *k* values among all RAs) can be when we assign some RAs with the highest privilege (W3) under our mitigation solution. In Figure 29, the number of available RAs is fixed at 500 and each RA can handle 4 tasks as maximum, while the number of maintenance tasks is varied between 500 and

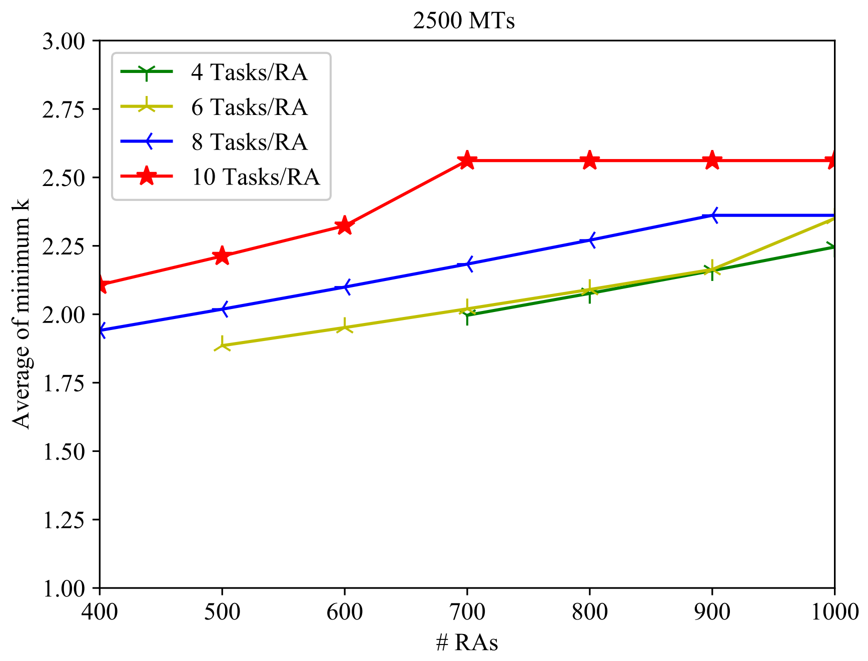


Figure 28: The minimum k for varying # of RAs

2,000 along the X-axis. The Y-axis shows the average k among all RAs. The solid lines show the results of average k after applying our mitigation solution under constraints about the number of RAs grant the W3 privilege before assigning tasks which are 20 RAs, 10 RAs, and no RA are granted the W3 privilege before task assignment, respectively.

Results and Implications: From the results, we can make the following observations. Grant the highest privilege to some of the RAs before assigning maintenance tasks can increase the average k to some degree when compared to the case when RAs are only granted privilege based on tasks needed to be performed. However, this decreases slower than others for the RAs who are granted privileges based on the maintenance tasks. As we can see in the figure, the trend (average k) of 20 RAs granted the highest privilege decreases faster (the insider threat increases) than others, which is expected because the highest privilege does not always correspond to the shortest path (e.g. W3 on the http node corresponds to a longer path than W2 on the app node).

In Figure 30, the number of maintenance tasks is fixed at 2,500 while the number of

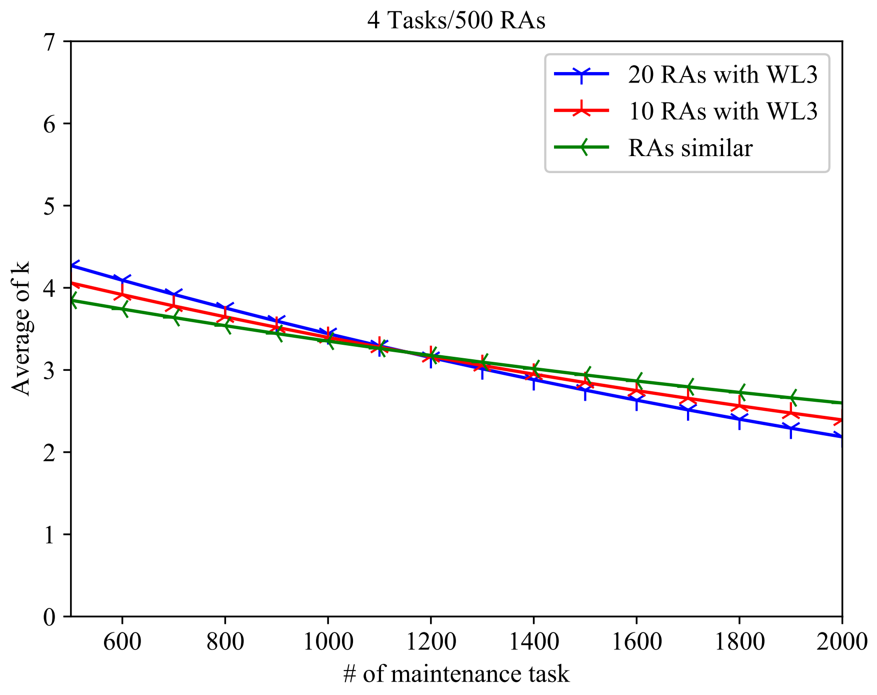


Figure 29: The average k among 500 RAs with some of them granted the highest privilege

RAs is varied between 400 and 1,000 along the X-axis. The Y-axis shows the average k among all RAs. Each RA can perform 10 maintenance tasks at most. The solid lines show the results of average k after applying our mitigation solution under constraints about the number of RAs grant the W3 privilege before assigning tasks which are 40 RAs, 20 RAs and no RA are granted the W3 privilege before task assignment, respectively.

Results and Implications: From the results, we can make the following observations. Granting the highest privilege to some of the RAs before assigning maintenance tasks can increase the average k in all cases, and all cases follow the similar trend of starting flat then increasing almost linearly before reaching the plateau. This trend shows granting the highest privilege to some RAs will increase the average k since the number of RAs are increased and the number of tasks are fixed.

The objective of the next two simulations is to study how the worst case insider threat (i.e., the minimum k values among all RAs) behaves under the mitigation solution when we assign some RAs with the highest privilege (W3). In Figure 31, the number of available

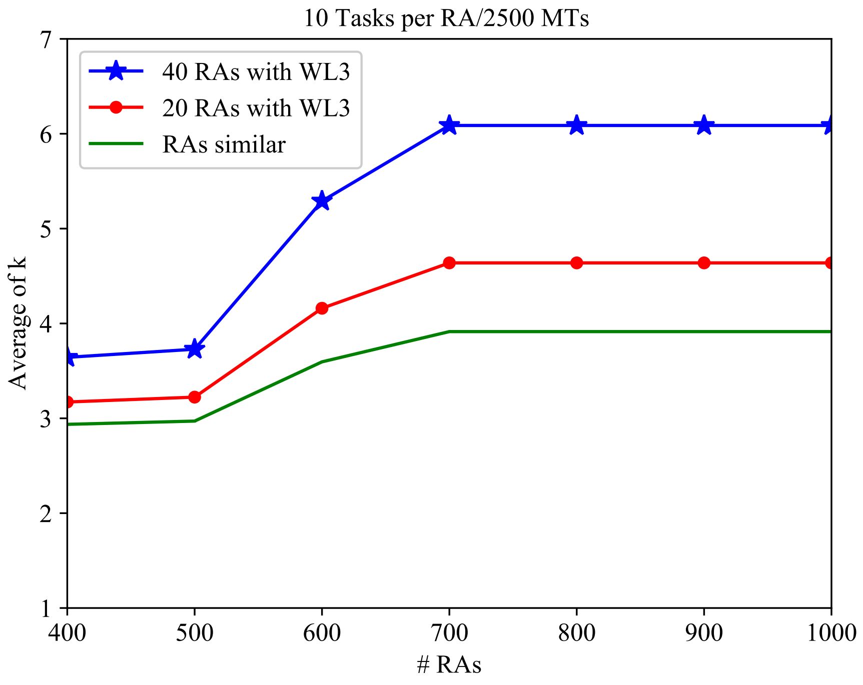


Figure 30: The average k among different numbers of RAs with some of them granted the highest privilege

RAs is fixed at 500 and each RA can handle four tasks at most, while the number of maintenance tasks is varied between 500 and 2,000 along the X-axis. The Y-axis shows the minimum k among all RAs. In Figure 31, the number of maintenance tasks is fixed at 2,500 while the number of RAs is varied between 400 and 1,000 along the X-axis. The Y-axis shows the average k among all RAs. Each RA can perform 10 maintenance tasks at most. The solid lines show the results of minimum k after applying our mitigation solution under constraints about the number of RAs granted the W3 privilege before assigning tasks, which are 10 RAs, 20 RAs, and no RA granted W3 privilege in Figure 31, respectively, and 20 RAs, 40 RAs, and no RA granted W3 privilege in Figure 32, respectively.

Results and Implications: From the result, we can make the following observations. The minimum k in Figure 31 follows the similar trend as in Figure 29 which decreases almost linearly as the number of tasks increases and decrease faster as the number of RAs granted with the highest privilege increases. In Figure 32, the minimum k increases almost

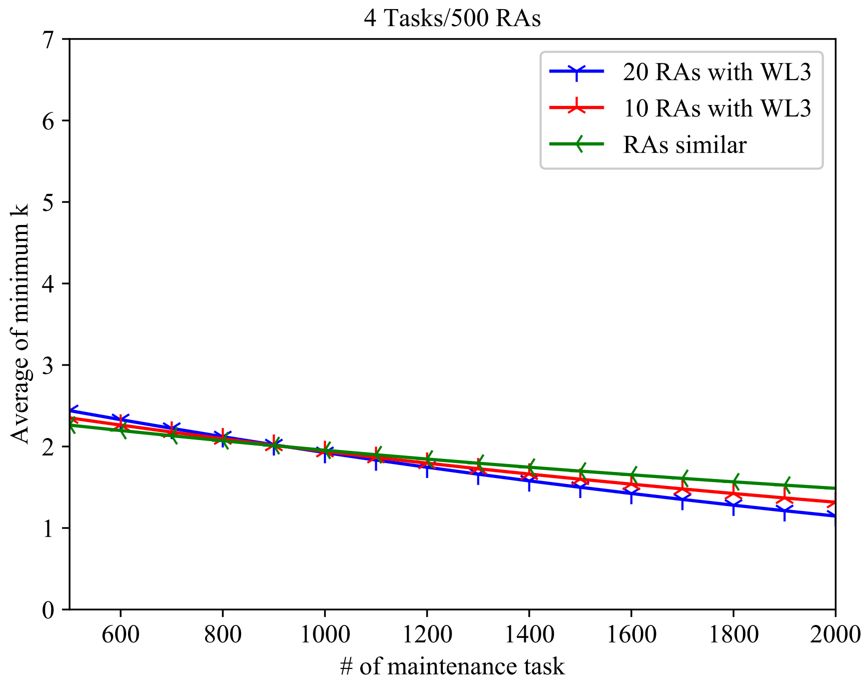


Figure 31: The minimum *k* among 500 RAs with some of them granted the highest privilege

linearly before reaching the plateau.

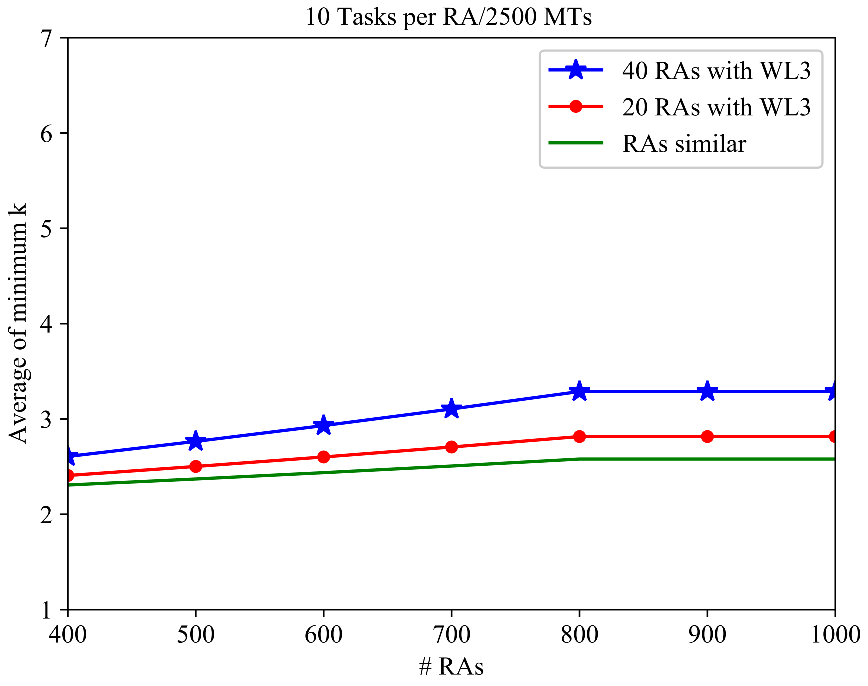


Figure 32: The minimum *k* among different numbers of RAs with some of them granted the highest privilege

The Impact of Local Administrators The objective of the next two simulations is to study how the average case insider threat (i.e., the average of k values among all RAs) behaves when we add a local administrator (LA) to perform MTs with their k value equal to the minimum k. In Figure 33, the number of available RAs is fixed at 500, while the number of maintenance tasks is varied between 500 and 2,000 along the X-axis. In Figure 34, the number of maintenance tasks is fixed at 2,500, while the number of RAs is varied between 400 and 1,000 along the X-axis. The Y-axis shows the average k among all RAs in both figures. The solid lines show the results of average k after applying our mitigation solution under the constraint that an LA can perform MT with its k value equal to the minimum k.

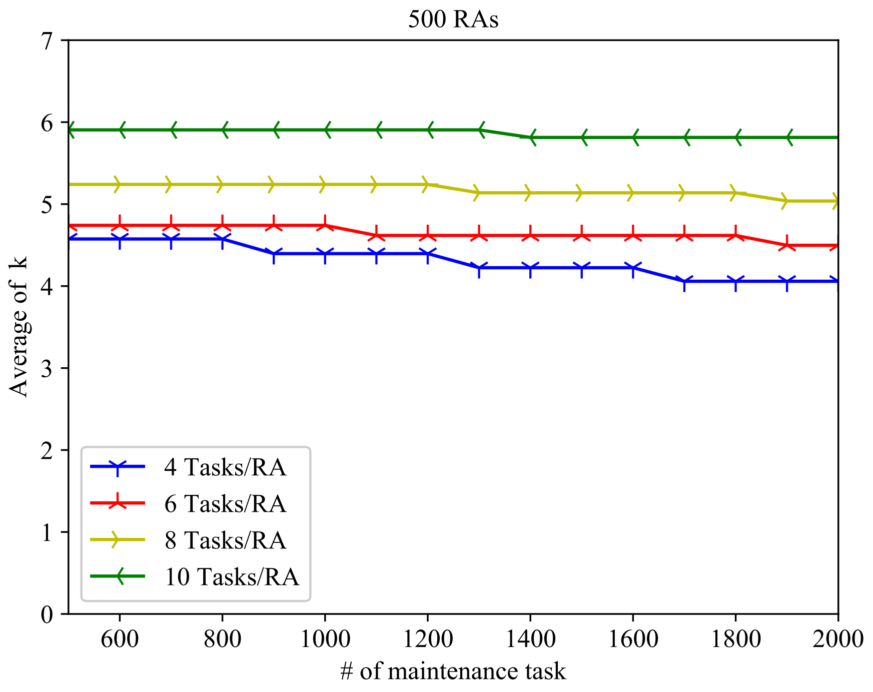


Figure 33: The average k among 500 RAs while assigning some of the tasks to a local administrator

Results and Implications: In Figure 33, we can see that the the average k mostly decreases slowly. The local administrator corresponds to the shortest path (minimum k) MTs needed to be performed. Increasing the number of tasks that can be assigned to each RA can increase the average k and the value of the average k decreases more slowly. In

Figure 34, we can see that increasing the number of RAs and eliminating the highest risk tasks (minimum k) by assigning those tasks to the LAs will increase the average k linearly before reaching the plateau.

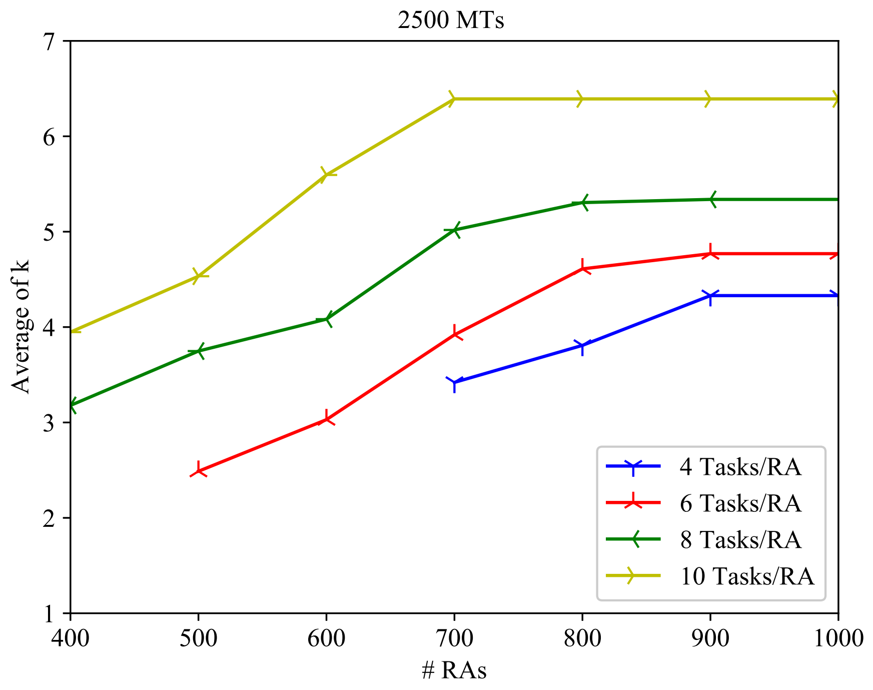


Figure 34: The average k among different numbers of RAs while assigning some of the tasks to a local administrator

The Impact of Service Dependencies The objective of the next simulations is to study how the service dependency can affect the average k. In Figure 35, the number of available RAs is fixed at 500, while the number of maintenance tasks is varied between 500 and 2,000 along the X-axis. The Y-axis shows the average k among all RAs. The solid lines show the results of average k after considering the service dependency in our mitigation solution under constraints about the maximum number of tasks assigned to each RA. The dashed lines represent the results without considering the service dependency in our mitigation solution.

Results and Implications: In Figure 35, we can see that considering the service dependencies will decrease the average k because we would have to consider more critical

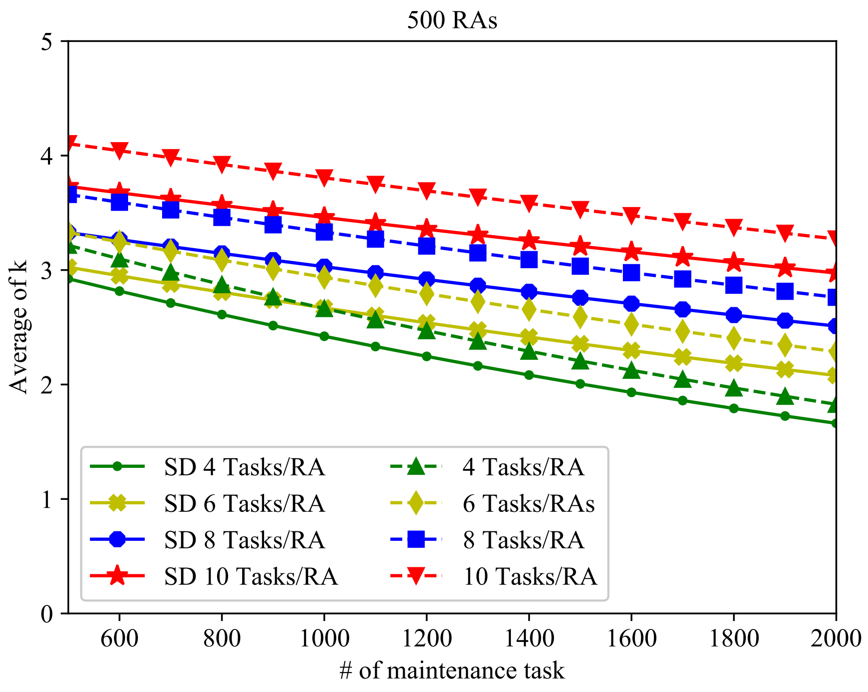


Figure 35: The average *k* among 500 RAs with and without considering service dependency

assets at the same time (e.g. if we want to secure the database VM service from being compromised, we will need to consider any service connected to the database VM as a critical asset). Also, we can see that, the result of *k* when considering the service dependency is relatively close to the result of *k* without considering it, which means the mitigation solution is relatively effective for protecting services as well as resources.

In Figure 36, the number of maintenance tasks is fixed at 2,500 while the number of RAs is varied between 400 and 1,000 along the *X*-axis. The *Y*-axis shows the average *k* among all RAs. The solid lines show the results of average *k* after considering the service dependency in our mitigation solution under constraints about the maximum number of tasks assigned to each RA. The dashed lines represent the results without considering the service dependency in our mitigation solution.

Results and Implications: From the result, we can make the following observations. Considering the service dependency will increase the average *k* almost linearly before reaching the plateau. However, when we compare the average *k* with the result of the

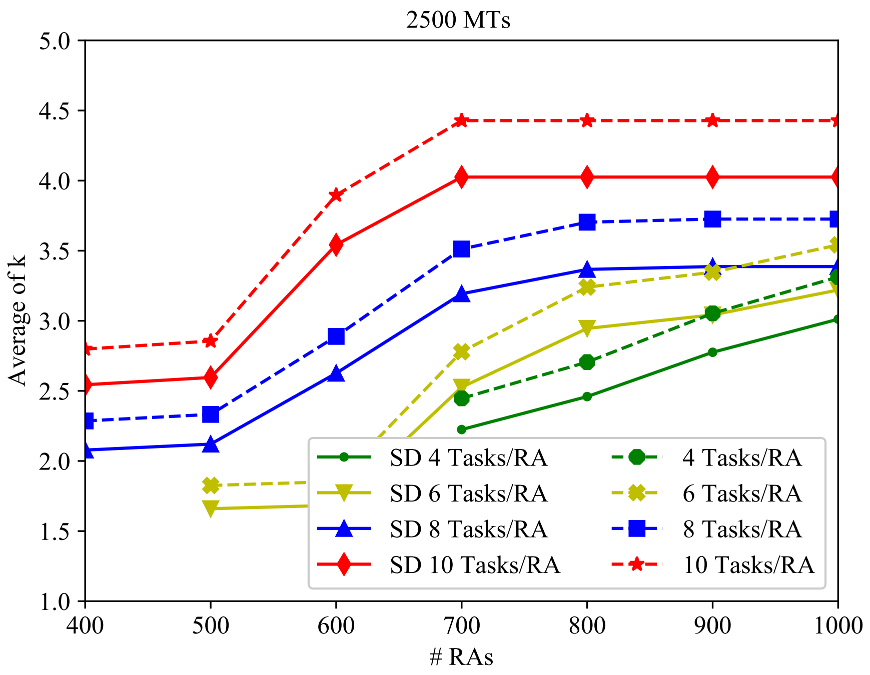


Figure 36: The average k among different numbers of RAs with and withoutconsidering service dependency

resource graph, we find that the average k under service dependency is lower which is expected because we are essentially considering more critical assets under the service dependency.

The objective of this simulation is to show how the minimum k behaves when considering the service dependency. In Figure 37, the number of available RAs is fixed at 500, while the number of maintenance tasks is varied between 500 and 2,000 along the X-axis. The Y-axis shows the average k among all RAs. In Figure 38, the number of maintenance tasks is fixed at 2,500 while the number of RAs is varied between 400 and 1,000 along the X-axis. The Y-axis shows the average k among all RAs.

Results and Implications: In Figure 37, we can see that the minimum k values also decrease almost linearly as the number of tasks increases, which means the insider threat increases. From the results in Figure 38, we can see that the minimum k values also increase almost linearly before reaching the plateau as the number of RAs increases. The increase

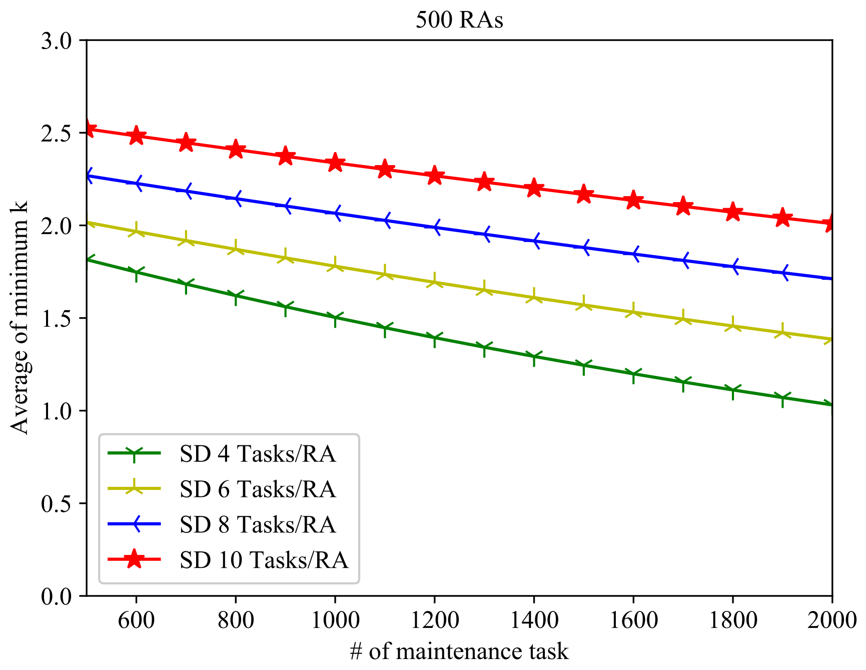


Figure 37: The minimum *k* among 500 RAs under service dependency

here is slower than that in Figure 36, which means the worst case (minimum k values) are more difficult to improve.

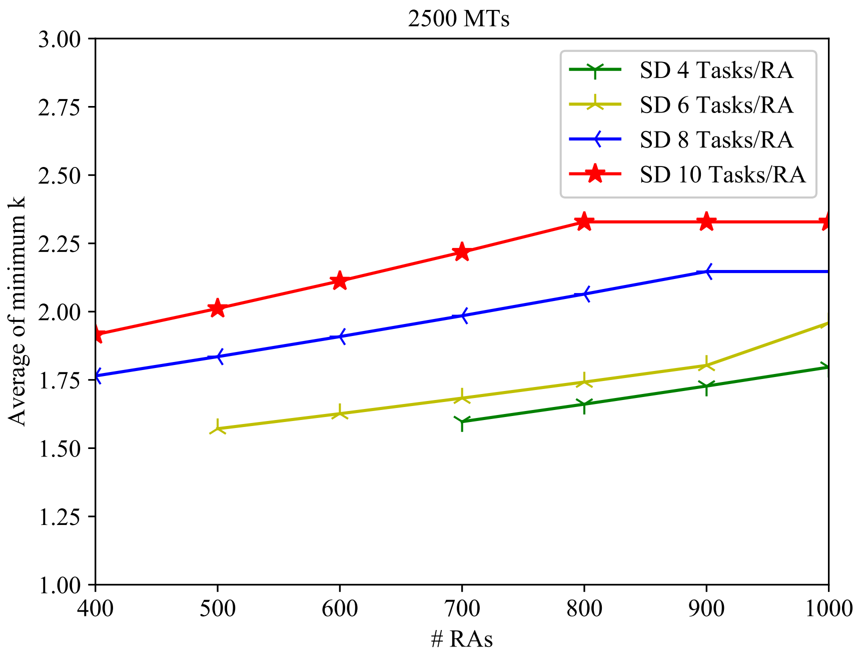


Figure 38: The minimum *k* among different numbers of RAs under service dependency

The BN-based Metric The objective of this simulation is to apply the BN-based metric instead of the k-zero day safety metric on both the resource graph and the service dependency graph. In Figures 39 and 40, the number of available RAs is fixed at 500, while the number of maintenance tasks is varied between 500 and 2,000 along the X-axis. The Y-axis shows the average probability to compromise the critical asset among all RAs.

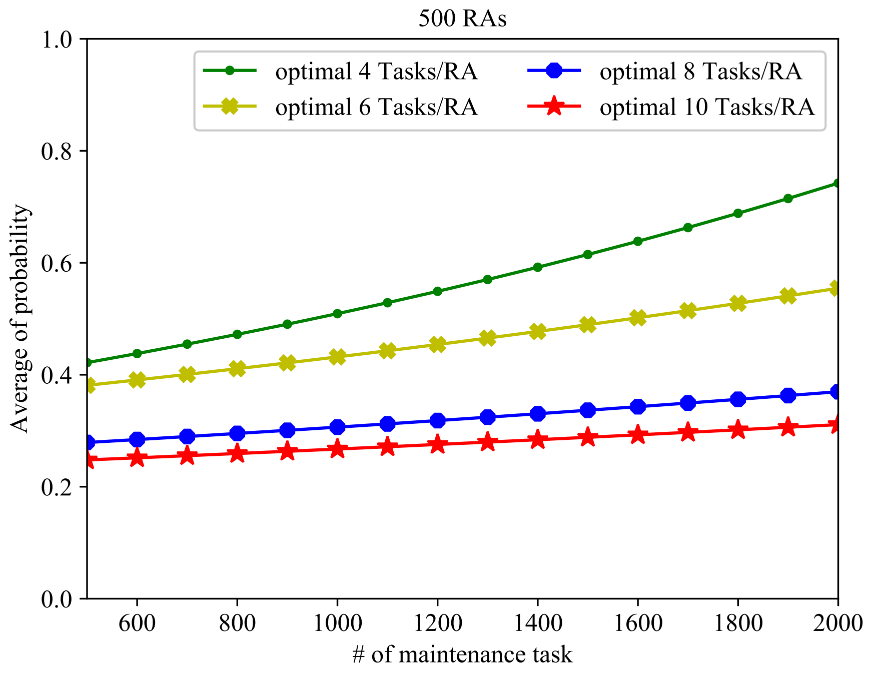


Figure 39: The probability to compromise critical assets based on the resource graph

Results and Implications: From the results, we can make the following observations. In Figure 39, The probability to reach a critical asset almost increases linearly when the number of maintenance tasks increases while the number of RAs are fixed. Also, we find increasing the maximum number of tasks that can be assigned to each RA slows the increasing rate of the probability which means slower increase in the insider threat. In Figure 40, we find the result follows the similar trend as the previous figure. However, the probability to reach the critical assets in the service dependency resource graph is much higher than that in the resource graph (e.g. for a maximum four tasks for each RA, the probability is almost 25% higher in the service dependency resource graph), which is expected because the service dependency means attackers would have more options to compromise the critical assets.

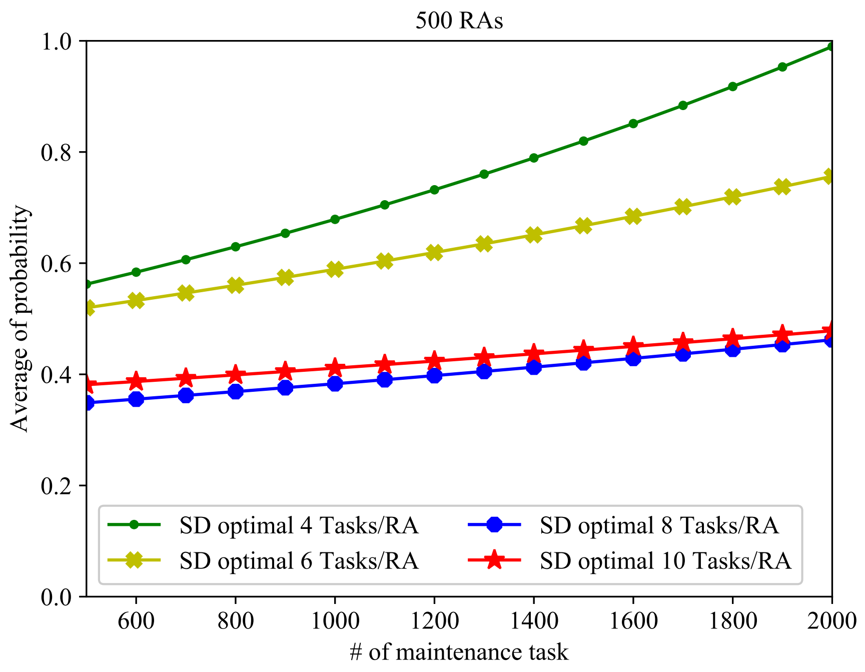


Figure 40: The probability to compromise critical assets based on the service dependency resource graph

4.6 Summary

In this chapter, we have modeled the insider threat during maintenance task assignment for cloud providers to better understand the threats posed by third party remote administrators. We have formulated the optimal assignment as an optimization problem and applied a standard optimization technique to derive solutions under different constraints. We have extended our insider threat models to consider service dependencies. Also, we applied the BN-base metric to take all attack paths and known vulnerabilities into consideration. Based on such models, we have conducted simulations for different use cases whose results show our solution can significantly reduce the insider threat of remote administrators.

Chapter 5

Modeling and Mitigating Security Threats in Network Functions Virtualization (NFV)

5.1 Introduction

As a cornerstone of cloud computing, virtualization has enabled providers to deliver various cloud services to different tenants using shared resources in a cost-efficient way. The trend of virtualization has also led to many innovations in networking in and outside clouds. In particular, traditional networks heavily rely on vendor specific hardware devices with integrated software, such as routers, switches, firewalls, IDSs, etc., which lacks sufficient flexibility demanded by today's businesses. Consequently, the need for decoupling software from hardware in network devices has led to Network Functions Virtualization (NFV) [6], which basically virtualizes proprietary hardware networking devices. As a key enabling technology of 5G, NFV has seen an increased adoption among cloud service providers, especially in the telecommunication industry [73].

However, the reliance on virtualization and the increased complexity together imply that NFV may unavoidably introduce new security concerns. First, as an NFV stack involves several abstraction levels covering the physical and virtual infrastructures as well as the virtual network functions [6], it naturally has a larger attack surface, opening doors to new security threats such as cross-layer attacks. Second, as one of the main advantages of NFV is to provide diverse network services to different tenants using the same hardware infrastructure, NFV would also share similar cross-tenant attacks as those seen in clouds (e.g., [152]). Therefore, security threats introduced by the multi-layer and multi-tenant nature of NFV need to be better understood and mitigated.

Attack modeling and mitigation in NFV has only received limited attention. Existing works have focused on specific vulnerabilities caused by orchestration and management complexities [151] and vulnerabilities resulting from the lack of interoperability [54] or the lack of proper synchronization between different abstraction levels [80]. There also exist works on dynamically managing security functions in NFV [147] and verifying Service Function Chaining (SFC)-related properties [56, 153, 135, 92]. Existing works on co-residency attacks mostly focus on clouds [24, 18] instead of NFV. To the best of our knowledge, there lack a general approach to modeling and mitigating NFV attacks.

In this topic, we take the first step toward modeling and mitigating security attacks in NFV. Our key ideas are threefold. First, we propose a multi-layer resource graph model for NFV in order to capture the co-existence of network services, VMs, and physical resources at different abstraction levels inside an NFV stack, and how those could potentially be exploited by attackers. This model allows us to capture not only attacks that target each layer of the NFV stack, but also attacks that go across different layers by exploiting the inter-dependencies between corresponding resources at those layers. Second, we also model the insider threats posed by malicious or compromised users of co-resident tenants inside the same NFV stack. The model allows us to captures how a co-residency attack may allow insiders to satisfy certain initial security conditions, such as privileges or connectivity, which are normally not accessible to external attackers. Third, we propose a solution to mitigate security attacks in NFV through VM placement and migration, which is a low cost option already available in NFV. The aforementioned model allows us to formulate the attack mitigation in NFV as an optimization problem and solve it using standard optimization techniques. We evaluate our approach through simulations to demonstrate its effectiveness under various situations.

5.2 Preliminaries

This section first provides background information on NFV and co-residency in NFV, and then gives a motivating example.

5.2.1 Background on NFV

As a main technology pillar of network softwarization and 5G, NFV provides network functions through software running on standard hardware. NFV enables network service providers to deploy dynamic, agile and scalable Network Services (NS). Such benefits come from the fact that an NFV deployment stack is usually an integration of various virtualization technologies including cloud and SDN together with various network orchestration and automation tools.

Specifically, the left side of Figure 41 shows the European Telecommunications Standards Institute (ETSI) NFV reference architecture [6]. The architecture builds on three main blocks, i.e., virtual network function (VNF), NFV infrastructure (NFVI), and NFV management and orchestration (MANO). First, the VNF provides network functions, such as router, switch, firewall, and load balancer, running on top of VMs through software. Second, NFVI represents the cloud infrastructure that provides basic computations, network, and storage needs for the execution of VNFs. Lastly, MANO has three management components, virtual infrastructure manager (VIM), virtual Network function manager (VNFM), and network function virtualization orchestrator (NFVO), which together manage and orchestrate the lifecycle of physical and virtual resources.

In addition, the right side of Figure 41 shows a multi-layer NFV deployment model [80] which complements the aforementioned ETSI architecture with deployment details, and divides an NFV stack into four layers, i.e., service orchestration (layer 1), resource management (layer 2), virtual infrastructure (layer 3), and physical infrastructure (layer 4).

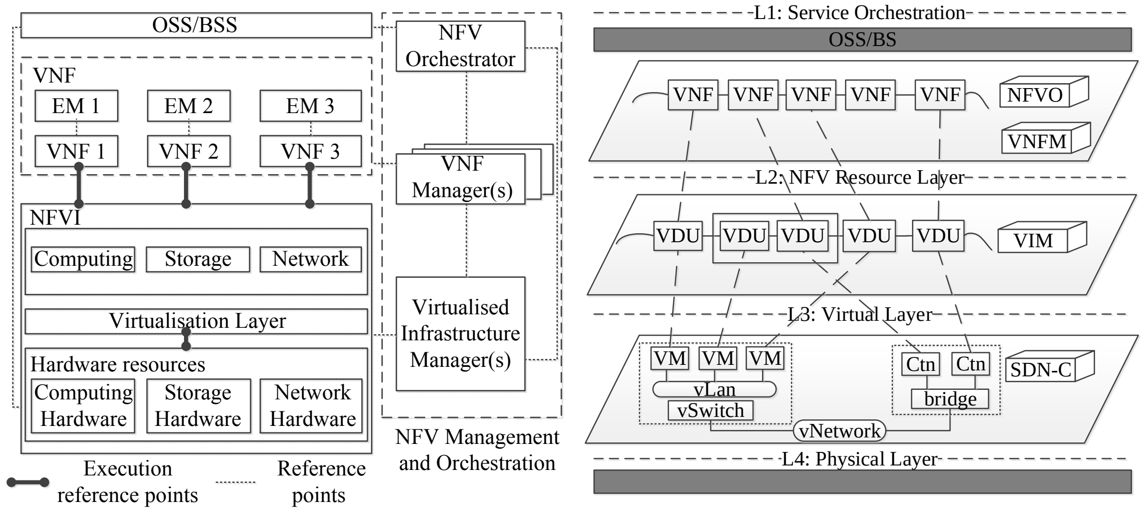


Figure 41: ETSI reference architecture [6] (left) and multi-layer NFV model [80] (right)

5.2.2 Co-Residency in NFV

As an NFV stack is multi-layer and multi-tenant in nature, placing and migrating a VM or VNF can be a challenging task for the provider due to the issue of co-residency. It is well known that co-residency may lead to various security issues, such as side-channel attacks, and additionally the tenant may have specific requirements in terms of (the lack of) co-residency. Co-residency may occur in an NFV environment when a new VM or VNF is first placed, or when an existing VM or VNF is migrated. The tenant requirements may specify that certain VNFs are to be placed on a dedicated host, or a VM needs to have the auto-scaling feature such that its need for more space can be quickly fulfilled. In terms of security, co-resident VMs or VNFs may belong to tenants with conflicting interests, and the co-residency may enable an insider attack with increased privileges and connectivity not available to regular attackers. Figure 42 shows an example of co-residency differencing in cloud and NFV Stack.

A unique aspect of co-residency in NFV is that, in an NFV stack, co-residency can happen between more layers, such as between VNFs and physical hosts, between VNFs and VMs, or between VMs and physical hosts. The co-residency of VNFs or VMs on the same physical host can occur due to placement or migration, which is known to lead to side-channel or resource depletion attacks due to the shared physical resources such as CPU, memory, or cache. The co-residency of VNFs on the same VM can also occur when different tenants employ the same VM to run similar network functions, such as virtual firewall or virtual IDS [75, 70, 112].

5.2.3 Network Slice

The NFV and related technologies enable the 5G infrastructure to support network slicing [102]. A network slice includes a collection of VNFs to compose the network services that are provided to the tenants. Each tenant can have several network slices under its management and a network slice can be used by several tenants in a multi-tenant environment. Network slicing enables concurrent logical networks providing various network services and functionalities to different tenants. For instance, an emergency network slice that allows immediate and on-demand services for patients, a highly secure communication slice for confidential video and audio communications for governmental or military users, and a low-bandwidth network slice for IoT devices used by IoT users and

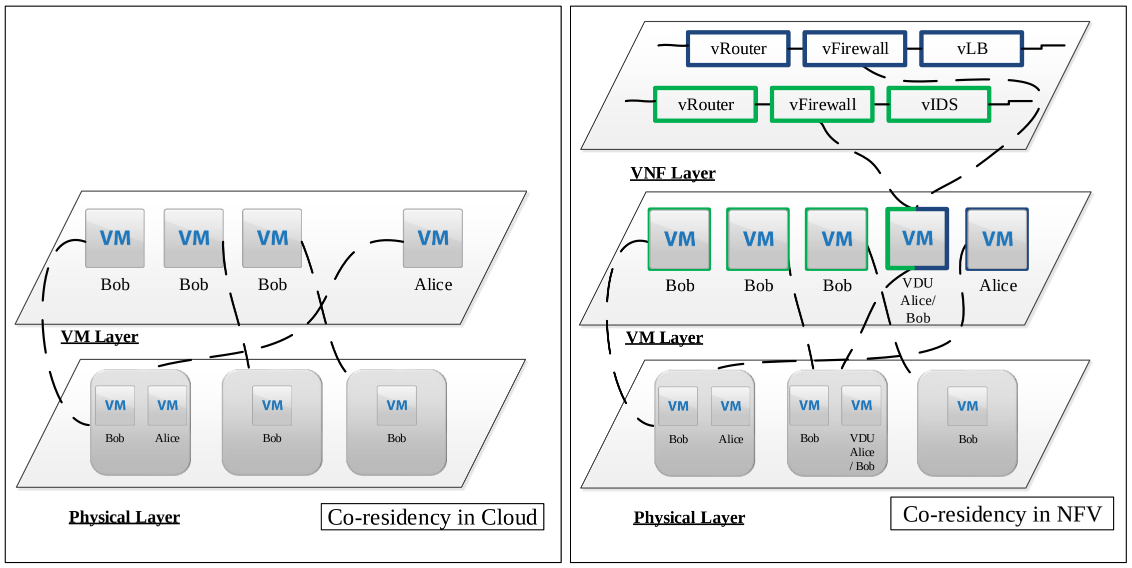


Figure 42: An example of co-residency in the cloud and NFV stack

applications.

The creation and deployment of network slices use the NFV stack resource and infrastructure attributes; NFVO maintains a repository of these network slices with their related attributes. The VNFs in a network slice are deployed in virtual machines (VMs) running over the infrastructure provided by the cloud or network provider. VNFM deploys the VNFs on VMs that are suitable, depending on the VMs' attributes. The VM resources are managed by the VIM through the SDN controller. The SDN controller works in conjunction with the VIM in the creation and establishment of both physical and virtual network communications. Finally, the tenants are assigned to these created and deployed network slices providing them with the requested network services.

5.2.4 Motivating Example

In the following, we present a concrete example of NFV stack to demonstrate the challenges in modeling and mitigating security threats for NFV. First of all, as NFV is a relatively new concept, there lack public access to information regarding the detailed hardware and software configurations used in real NFV environments. As can be seen in Figure 41, both the ETSI architecture [6] and the multi-layer deployment model [80] are quite high level, and lack such details. Most other existing works either focus on high-level frameworks and guidelines for risk and impact assessment [2, 88, 119], or very specific vulnerabilities [81, 105, 106], with a clear gap in between.

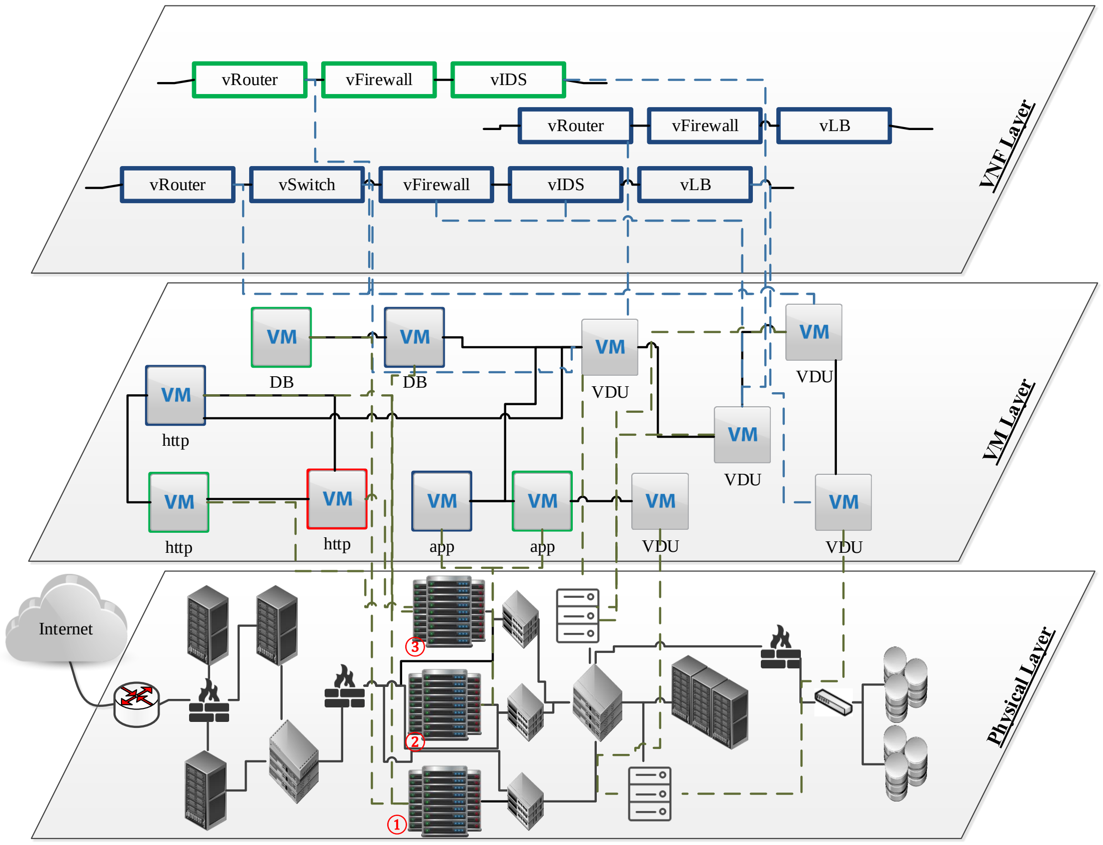


Figure 43: A concrete example of NFV stack (vFirewall: virtual firewall, LB: load balancer, DB: database, VM: virtual machine, VDU: virtual deployment unit)

To address such limitations, we design a concrete example of NFV stack, as shown in Figure 43, based on both the ETSI architecture [6] and the multi-layer deployment model [80], as well as other public available information. As shown in Figure 43, the NFV stack is depicted on three layers where the VNF layer shows three service function chains (SFCs), and the VM and physical layers show the corresponding virtual and physical infrastructures used to implement those SFCs, respectively. The dashed lines between layers demonstrate the correspondence between the services and the virtual or physical resources. We assume there are three tenants, shown in Figure 43 through different colors, i.e., Alice (blue), Bob (green), and Mallory (red)), that are hosted on this NFV stack.

In such a scenario, both the NFV provider and each tenant may want to understand and mitigate potential security threats. While existing threat models such as various attack trees and attack graphs may be applied, there are some unique challenges and opportunities as follows.

- First, as can be seen in Figure 43, the NFV stack is composed of different layers, and the inter-dependencies between resources on those layers may lead to novel cross-layer attacks. The NFV tenant and provider need to consider all layers and the inter-dependencies between layers when analyzing potential security threats because of the possibility of such cross-layer attacks.
- Second, the fact that multiple tenants are sharing both virtual and physical resources in the same NFV stack poses another challenge, i.e., co-residency attacks. In contrast to clouds, NFV may have an increased attack surface in terms of co-residency attacks. As demonstrated in Figure 43, unlike in clouds, co-residency in NFV may occur in terms of both shared physical resources and shared virtual resources such as VMs, which must be considered in modeling the threat of co-residency attacks.
- On the other hand, virtualization in NFV also provides an opportunity for mitigating security threats through a unique hardening option, i.e., through VM placement or

migration. In contrast to other hardening options, such as patching vulnerabilities, disabling services, or stricter firewall and access control rules [31], VM placement or migration provides a lower cost option as it is a built-in feature already employed by providers for other purposes such as maintenance or resource consolidation.

More specifically, we consider concrete problems of threat modeling and attack mitigation based on Figure 43 as follows.

– First, we would like to model potential multi-step attacks that could occur in this NFV stack (Figure 43), by assuming Mallory (whose resources are shown in red color) is a malicious tenant, and the database VM belonging to Alice (whose resources are shown in blue color) is the critical asset in question. The modeling process must consider the multi-layer and multi-tenant nature of NFV and its many security implications, e.g., an attacker's VM placed on the database server (node # 1) or on the $http$ server (node # 3) would certainly incur very different security threats, and an attacker co-residing with the target tenant on the VM level could have a better chance of compromising the target than one co-residing on the physical host level.

– Second, we would like to mitigate the modeled threats posed by Mallory to Alice's database VM, through optimal placement or migration of virtual resources in this NFV stack. The solution must quantify the security threats before and after the hardening process in order to show the amount of improvement in terms of security, and the solution should be able to accommodate other considerations or constraints, e.g., limiting the scope to one layer or multiple layers, supporting different VM placement policies (such as those used in CloudSim [43]), and limiting the cost of placement or migration (such as the maximum number of VM migrations).

To this end, we will present our threat modeling solution in Section 5.3 and our attack mitigation solution in Section 5.4.

5.3 Modeling Security Threats in NFV

This section presents our solutions for modeling potential multi-step attacks on an NFV stack (Section 5.3.1), for modeling insider attacks from co-residing tenants (Section 5.3.2), and for quantifying the threats using a security metric (Section 5.3.3). Also, we use the Bayesian network (Section 5.3.4) and network slicing (Section 5.3.5) to measure security threats.

5.3.1 Multi-layer Resource Graph

Threat Model: Our goal is to help the NFV provider or tenants to better understand and mitigate the security threats from an external attacker, a dishonest or compromised user or tenant, or a tenant administrator or cloud operator with limited privileges. We assume the NFV provider and its administrators are trusted and consequently the inputs to our threat modeling process are intact. Our in-scope threats are security attacks used to escalate privileges or gain remote accesses through exploiting known or zero day vulnerabilities in the physical or virtual entities inside an NFV stack. Out-of-scope threats include attacks which do not involve exploiting vulnerabilities (e.g., phishing or social engineering attacks) or do not propagate through networks (e.g., flash drive-based malware).

To model the security threats in an NFV stack, our key idea is to apply the resource graph concept [125] (which is syntactically similar to attack graphs, but focuses on modeling zero day attacks exploiting unknown vulnerabilities) while considering the multi-layer nature of NFV (as explained in Section 5.2.1). Specifically, based on a model of the NFV stack with three layers, i.e., the VNF layer, VM layer, and physical layer, as previously demonstrated in Figure 43, we propose the concept of *cross-layer resource graph* to represent the causal relationships between different resources both inside each layer, and across different layers, in a given NFV stack.

We first illustrate the concept through an example, before giving the formal definitions.

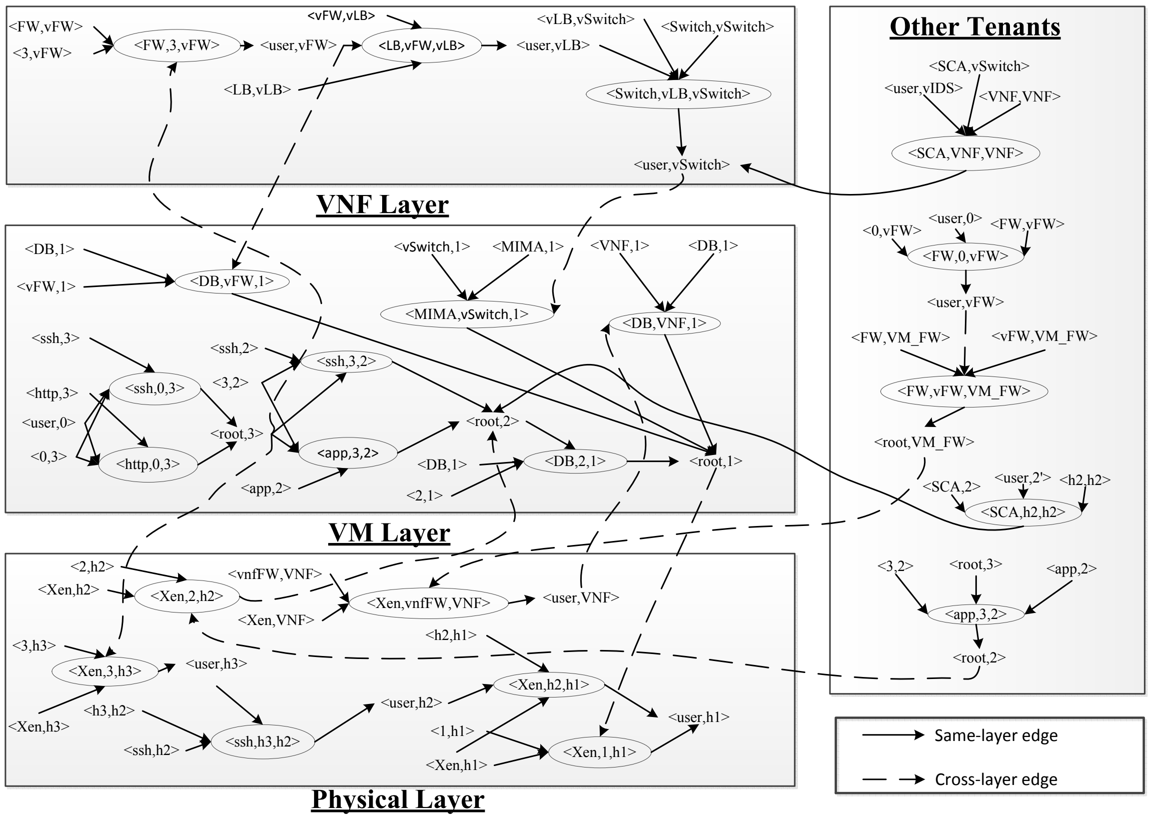


Figure 44: An example of cross-layer resource graph (FW: firewall, LB: load balancer, DB: database, MIMA: man in the middle attack, SCA: side-channel attack)

Figure 44 shows an example of a cross-layer resource graph for our running example (only a portion of the cross-layer resource graph is shown here due to space limitations). The VNF layer maps to layer 1 and layer 2 in the multi-layer NFV deployment model [80], which depicts exploits of various virtual network functions, such as virtual firewalls, load-balancers, switches, etc. The VM layer corresponds to layer 3 in the NFV deployment model, which includes exploits of the VMs that are used to implement the virtual network functions in the cloud layer. Finally, the physical layer includes exploits of the physical hosts. The left-hand side of Figure 44 shows the cross-layer resource graph for one tenant, whereas the right-hand side shows co-residency attacks from other tenants (which will be explained in next section).

Each triple inside an oval indicates a potential zero day exploit (in which case the unknown vulnerability is represented by the exploited service itself) or an exploit of known vulnerabilities, in the form of <service or vulnerability, source host, destination host>. For example, <Xen, 3, h3> indicates an exploit of Xen coming from a VM on physical host 3 to that physical host itself, and the plaintext pairs indicate the pre- or post-conditions of those exploits in the form of <condition, host>, where a condition can be a privilege on the host, e.g., <root,3> means that the root privilege on the VM runs on host 3, and <user,vFW> means the user privilege is on the VNF layer for the virtual firewall. Additionally, conditions may include the existence of a service on the host (e.g., <Xen,3>), or a connectivity (e.g., <0,3> means that attackers can connect to a VM on host 3 from host 0, and <2,h2> means a local exploit is occurring on host 2). The edges point from pre-conditions to an exploit and then to its post-conditions, which indicate that any exploit can be executed if and only if all of its pre-conditions are satisfied, whereas executing an exploit is enough to satisfy all of its post-conditions. The following provides formal definitions of those concepts.

Definition 8 (Same-layer Resource Graph)**.** *Given a collection of hosts (physical hosts or VMs) H and the set of resources HR (services or VNFs running on a physical host or a VM) with the resource mapping* $rm(.) : H \rightarrow 2^{HR}$*, and also given a set of zero day exploits* $E = \{< r, h_s, h_d > | h_s \in H, h_d \in H, h_r \in rm(h_d)\}$ *and the set of their pre- and post-condition C, a same-layer resource graph is a directed graph* $G(E \cup C, HR_r \cup HR_i)$ *where* $pre \subseteq C \times E$ *and* $post \subseteq E \times C$ *are the pre- and post-condition relations, respectively.*

Definition 9 (Cross-layer Resource Graph)**.** *Given the same-layer resource graph for the three layers,* $G_i(E_i \cup C_i, pre_i \cup post_i)(1 \leq i \leq 3)$*, and given the cross-layer edges* $pre_c \subseteq \bigcup_1^3 C_i \times \bigcup_1^3 E_i$ *and* $post_c \subseteq \bigcup_1^3 E_i \times \bigcup_1^3 C_i$*, a cross-layer resource graph is a directed graph* $G(\bigcup_1^3 (E_i \cup C_i), \bigcup_1^3 (pre_i \cup post_i) \cup pre_c \cup post_c)$*.*

5.3.2 Modeling Co-residency Attacks

In modeling co-residency attacks, our main idea is to capture the consequences of such attacks as satisfying certain conditions inside the cross-layer resource graph of the targeted tenant. For example, in Figure 44, the left-hand side shows the cross-layer resource graph of the targeted tenant, which depicts what an external attacker may do to compromise the critical asset <user, h1>. On the other hand, the right-hand side of the figure depicts the insider threat coming from potential co-residency attacks launched by other tenants. The (dashed) lines pointing from the right to the left side of the figure show that, as the consequence of the co-residency attacks (right), some conditions inside the targeted tenant's resource graph (left) may be satisfied, either within the same layer or across different layers. The co-residency could occur w.r.t. the physical layer, which is similar to the co-residency in clouds (when tenants share the same physical host). The co-residency could also occur w.r.t. the VMs when tenants employ the same VM to run their VNFs, which is unique to NFV.

For example, as shown on the right-hand side of Figure 44, a malicious co-resident tenant can potentially gain a user privilege on the vSwitch service of the targeted tenant through a local exploit <SCA,VNF,VNF>, or he/she can gain a root privilege on a VM on host 2 through a similar attack (<SCA,h3,h3>) (where <user,3'> means the privilege of the malicious tenant on a VM on host 3). A malicious tenant who shares VNFs running on the same VM may attack the virtual firewall VNF and subsequently the corresponding VM to eventually be able to control the firewall (<root,VM_FW>) and modify its rules in order to gain access to the critical asset. A malicious tenant can also exploit an application running on VM 2 to gain control of that VM, and subsequently attack the co-residing host h2. These examples show how our model can capture co-residency attacks between different layers of the NFV stack.

5.3.3 Cross-layer Security Metric

Before we could mitigate the modeled security threats, we need to first quantify them such that we could evaluate the level of threats before and after we apply a hardening option. For this purpose, we apply the *k*-zero day safety security metric [138, 150, 139] originally proposed for traditional networks. The metric basically counts how many distinct unknown vulnerabilities must be exploited in order to compromise a given critical asset. A larger *k* value will indicate a relatively more secure network because the possibility of having more unknown vulnerabilities occurring at the same time, inside the same network, and exploitable by the same attacker would be significantly lower. The metric can be evaluated on the resource graph of a network, which basically gives the length of the shortest path (in terms of the number of distinct zero day exploits). The exploits of known vulnerabilities can be either regarded as a shortcut (i.e., they do not count toward *k*) or assigned with a significantly lower weight in the calculation of *k*.

On the basis of the cross-layer resource graph model introduced in previous sections, the *k*-zero day safety metric (*k*0d) can be applied in several ways. First, we could evaluate the metric on the cross-layer resource graph of the targeted tenant, without considering others tenants, whose result provides an estimation for the threat coming from external attackers. Second, we could also evaluate the metric on the cross-layer resource graph including co-resident attacks from others tenants, and we could consider one particular malicious tenant, or multiple such tenants either separately (assuming they do not collude) or collectively (as one, assuming they may collude). Third, we could evaluate the metric before, and after applying a placement or migration-based hardening option, and the difference in the results will indicate the effectiveness of such a hardening option (which we will further investigate in next section).

For example, in Figure 44, by considering a malicious tenant sharing the same physical host with the targeted tenant (indicated by privilege <root,VM_FW>) would yield a *k*0d value of 2 since two zero day exploits <Xen,vnfFW,NFV>, and <DB,VNF,1> are needed to reach the critical asset. Whereas considering a malicious tenant with privilege <user,2'> (here 2' indicates the privilege belongs to a tenant different from the targeted one) would yield a *k* value of 3 since three zero day exploits, <SCA,h2,h2>, <DB,2,1>, and <Xen,1,h1> are needed.

5.3.4 The Bayesian Network-based Security Metric

The previous section has applied the *k*-zero day safety metric to model the cross-layer resource graph. This is a conservative model since the *k* value is defined based on the shortest attack paths, which attacker may or may not be able to follow in practice. Moreover, the model only considers zero day exploits and known vulnerabilities do not contribute to the *k* value or assigned with a significantly lower weight in the calculation of *k*. In this section, we extend this model by applying the Bayesian network (BN)-based metric [57] instead of the *k*-zero day safety.

The BN-based metric is based on the conditional probability of reaching the given critical assets given that all initial conditions are satisfied. We first construct a Bayesian network based on the cross-layer resource graph and the conditional probability that each exploit can be executed given its pre-conditions are all satisfied. Such conditional probabilities can be assigned to both known vulnerabilities based on standard vulnerability scores (e.g., the CVSS scores [95]), and zero day exploits based on a nominal value (e.g., 0.08 [100]). Therefore, the model captures both zero day and known vulnerabilities, and it also takes all attack paths into consideration. Finally, the model can also capture additional casual dependencies, e.g., the same vulnerability appearing on multiple hosts may yield a higher probability (e.g., 0.9 in our examples).

By applying the BN-based metric to the cross-layer resource graph given in Definition 9, we can obtain the probability for each tenant to compromise the given critical assets given all the co-residency implied by a resource that belongs to the tenant. Since a tenant may own multiple resources, the tenant can compromise the critical assets as long as at least one of the owned resources enables him/her to do so whose probability can be computed as in Equation 4. We redefine the security metric based on those discussions in Definition 10. The model also allows assigning the relative likelihood of each tenant to be misbehaving, which can be estimated either based on the background (e.g., tenants who have a conflict of interest with target tenant should be assigned a higher probability than tenants who do not have a relation with target tenant) or behavior-based detection results if available.

Definition 10 (The BN-based Security Metric)**.** *Given a collection of hosts (physical hosts or VMs) H, the set of resources HR (services or VNFs running on a physical host or a VM) with the collection of tenants T with the tenant mapping function* $tm(.) : T \to HR$*. Let* $P_{BN}(c)$ *be the probability that a tenant resource c can lead to compromising the given critical assets, and* $P_M(t)$ *the given probability that a tenant t will misbehave. We say* $P(t)$ *and* $\frac{\sum_{t \in T} P(t)}{|T|}$*, represent the average case threat of tenant t and of all tenants, respectively, where*

$$P(t) = 1 - [\prod_{c \in tm(t)} (1 - P_{BN}(c) \cdot P_M(t))] \tag{4}$$

5.3.5 The Network Slicing Model

The co-residency attacks introduced in previous sections are based on cross-layer resource graphs, which are mainly designed to model hardware, software, VM and VNF resources. The cross-layer resource graphs, however, do not directly indicate any higher level concepts about multiple virtual networks that operate in the same NFV stack. For this purpose, the concept of network slicing [86] has been proposed to model the security threat of network slicing (cross-slice attack).

Therefore, we study how the insider threat can move from one network slice to another slice running in the same NFV stack. Network slicing can help to provide end-to-end delivery of services for several vertical industries over virtualised infrastructure. In Figure 45, we use a smart grid as an example of network slicing with the control slice and corporate slice. The smart grid function may be provided through software running over standard hardware like network function in NFV stack. Attarha et al. [17] applied the VNF concept to provide the virtual grid function (VGF). In Figure 45, we show the control slice and corporate slice for a smart grid and each slice runs as an isolated network. However, since different tenants may share the same infrastructure, the sharing can lead to a cross-slice attack. We formalize the cross-slice resource graph concept in Definition 11. The model extends the cross-layer resource graph by adding nodes for network slices co-residency and their pre- and post-conditions.

Definition 11 (Cross-slice Resource Graph)**.** *Given the cross-layer resource graph for each network slice* $G_k(\bigcup_1^3(E_i \cup C_i), \bigcup_1^3(pre_i \cup post_i) \cup pre_c \cup post_c)(1 \leq k \leq n)$ *where* n *is the number of the network slice, and given cross-slice edges* $pre_s \subseteq \bigcup_1^n C_k \times \bigcup_1^n E_k$ *and* $post_s \subseteq \bigcup_1^n E_k \times \bigcup_1^n C_k$*, a cross-slice resource graph is a directed graph* $G(\bigcup_1^n(\bigcup_1^3(E_{ik} \cup C_{ik})), \bigcup_1^n(\bigcup_1^3(pre_{ik} \cup post_{ik})) \cup (\bigcup_1^n(pre_{ck} \cup post_{ck})) \cup pre_s \cup post_s)$*.*

In Figure 45, the target is the circuit breakers which are responsible for protecting transmission lines from damages due to excess current. If a fault is detected on a transmission line, the circuit breaker responsible for that transmission line cuts the flow of the electricity. However, if circuit breakers are tripped by a malicious attacker, this can cause some transmission lines to carry load higher than their capacity and consequently, it may lead to transmission line failures, which may then cause a blackout. For example, in the Ukrainian power grid [83], attackers have tripped circuit breakers by sending remote commands, and they also have changed firmware contained in IEDs to delay repair attempts. As shown in Figure 45, the left side represents the network slice for the controller, which includes the target; on the right side, we have another network slice which is corporate and the dot line in the figure represents the crossing slice attacks. In network slicing, each network slice is isolated from other slices logically, but they may share the same physical infrastructure. For example, the control slice and corporate slice are sharing host 3. a tenant using the corporate slice has access to the VM running *http* service on host 3 and he/she may trip circuit breakers through a local exploit <Xen,3,h3> to reach the VM belonging to the control slice hosted on the same physical machine which runs vIED.

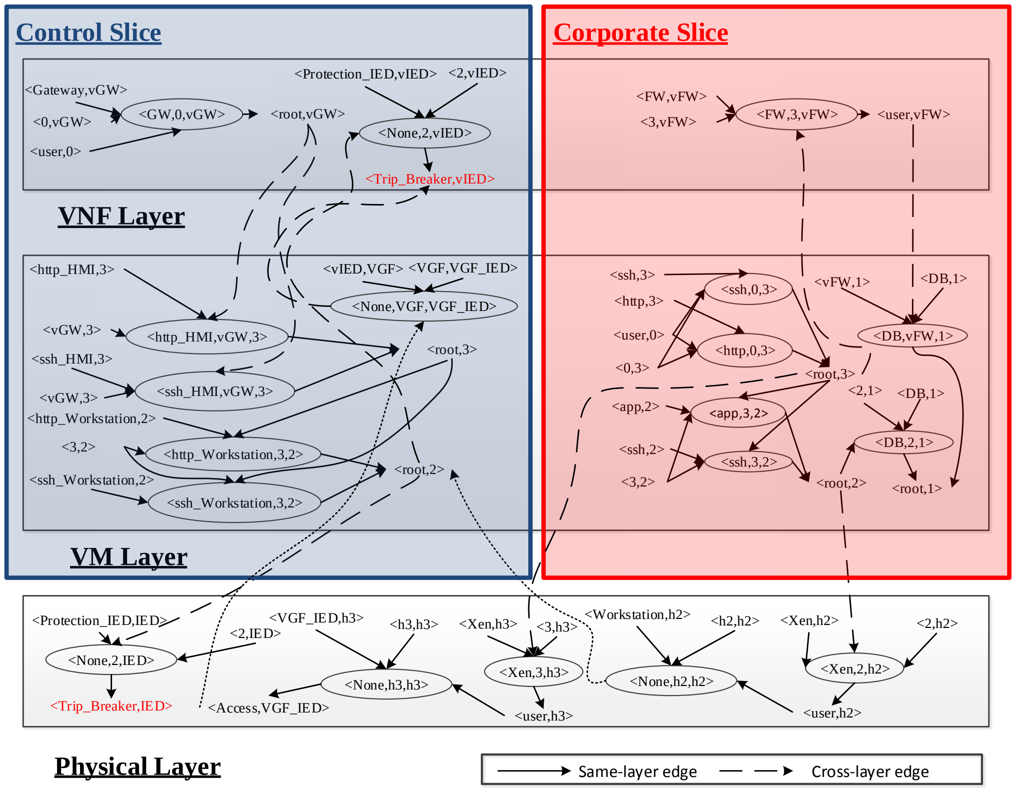


Figure 45: An example of network slicing

5.4 Attack Mitigation

In this section, we present the optimization-based mitigation through placement and migration of VNFs and VMs, and demonstrate its applicability through discussing several use cases.

5.4.1 Optimization-based Mitigation

Based on our previous definition of cross-layer resource graph model and the discussions on modeling co-residency and applying the $k0d$ metric, we can formulate the problem of optimal placement and migration of VMs and VNFs. As shown in Equation 5, hosts and resources are defined in a way such that the placement and migration may apply to both VMs (on physical hosts) and VNFs (on VMs) through the resource mapping function. The objective function is the application of the $k0d$ metric to the cross-layer resource graph (which can be a value assignment of the resource mapping function. Note that the application of the $k0d$ metric could take several forms for different purposes, as discussed in Section 5.3.3), which gives different variations of the optimization problem. Although not specified in the equation, constraints may be given in terms of possible value assignments to the resource mapping function, e.g., which VM (or VNF) may be placed or migrated to which physical host (or VM), or a threshold for the maximum number of migrated VMs.

$$\begin{aligned} k &= maximizes\ k0d(G) \\ subject\ to &\qquad rm(h) \end{aligned} \tag{5}$$

Where G is the cross-layer resource graph, given a collection of hosts (physical hosts or VMs) H, the set of resources HR (services or VNFs running on a physical host or a VM), and the collection of tenants T with the tenant mapping function $tm(.) : HR \to T$. Also, $k0d(G)$ denotes the application of the $k0d$ metric to G, and the $rm(h)$ denotes the resource mapping function $rm(.) : H \to 2^{HR}$.

The optimal NFV co-residency problem we have defined is intractable, since it can be easily reduced to the NP-hard problem of network hardening through diversity in traditional networks [30]. Specifically, the goal in the diversity problem is to maximize the $k0d$ metric by changing the instance of services (e.g., from IIS to Apache for web service), assuming that different instances of the same service along the shortest path would both count toward the k value (conversely, two identical instances would only count as one). Our problem can be reduced to this since, for any given resource graph G under the diversity problem, we can construct a special case of our problem by regarding G as the VM-layer resource graph, and regarding the instance of a service as the physical host on which that service resides (such that identical instances of a service are always co-resident). By further assuming that the attacker can always trivially exploit all co-resident services as long as he/she can exploit one (i.e., co-resident services only count as one toward the k value), the two problems then become equivalent.

In our study, we use the genetic algorithm (GA) [60] to optimize the VM (VNF) placement and migration for maximizing k. Our choice of GA is inspired by [47] and based on the fact that GA provides a simple way to encode candidate solutions and requires little information to search effectively in a large search space [60]. Specifically, the cross-layer resource graph is taken as input to the optimization algorithm, with k (averaged between tenants) as the fitness function. We try to find the best VM placement within a reasonable number of generations. The constraints we have considered include defining the resource mapping function in the case that specific VMs can be assigned to each host (e.g., firewall only), enforcing a given placement policy (e.g., CloudSim [43] placement policy), or satisfying a maximum number of migrating VMs. In our simulations, we choose the probability of 0.8 for crossover and 0.2 for mutation based on our experiences.

5.4.2 Use Cases

We demonstrate the applicability of our solution through several use cases with different types of attackers and while considering different layers. The first use case contrasts an external attacker to an insider launching cross-tenant attacks. The second use case compares a same-layer attack versus a cross-layer attack. The third use case is based on the motivating example shown in Section 5.2.4. The last two use cases apply the BN-based metric and address network slicing, respectively.

- Use Case A: In this case, we have an external attacker using a victim tenant's resources, and an insider malicious tenant co-residing with the victim tenant. Figure 46 shows the cross-layer resource graph for the external attacker (AE) and the insider attacker (AI). The figure shows the shortest path (dashed lines) for calculating the $k0d$ metric, and the critical asset is represented as $< user, h1 >$. After optimization, the value of k for the external attacker (AE) is 4, and for the insider attacker (AI) is 3 (which means there is less room to mitigate the insider threat).

- Use Case B: In this case, we compare a one-layer attack (BOL) to a cross-layer attack (BCL) for an insider malicious tenant. The BOL and BCL dashed lines shown in Figure 46 show the shortest paths for the malicious tenant using his/her co-residency with the victim tenant to reach the target $< root, 1 >$. After optimization, the value $k = 3$ for BOL and $k = 2$ for BCL show that there is less room to mitigate the insider threat when the attack may go across layers.

- Use Case C: This case shows the optimal placement result for our motivating example discussed in Section 5.2.4. We consider three tenants (Alice (A), Bob (B), and Mallory (M)) and three servers each of which can host four VMs. We consider Mallory a malicious tenant and the database VM belonging to Alice a target. Table 11 shows three different placements. The top table shows the placement before applying

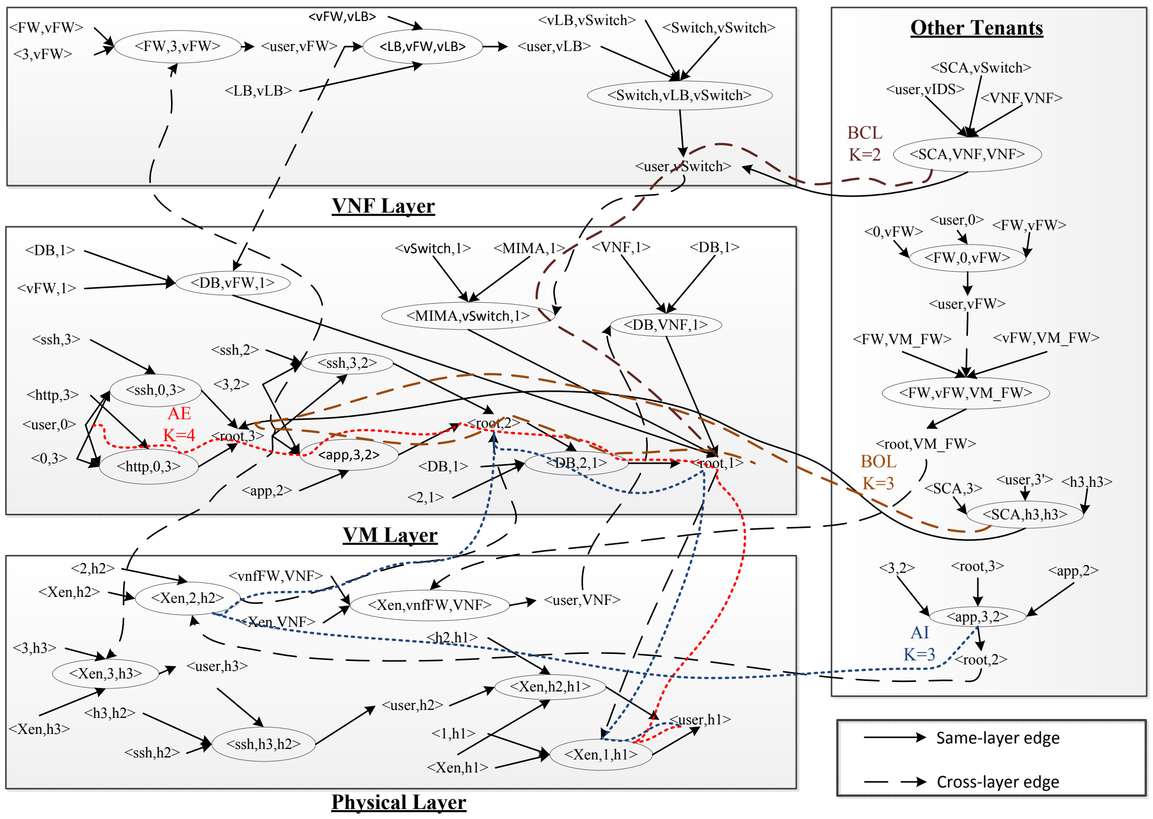


Figure 46: Use Cases A and B (FW: firewall, LB: load balancer, DB: database, MIMA: man in the middle attack, SCA: side-channel attack, AE: external attacker, AI: insider attacker, BCL: cross-layer attack, BOL: one-layer attack)

our optimization solution and the value of $k = 2$. The middle table and the bottom table show the optimal placement after we apply our solution by the victim tenant (where the migration is limited to the tenant's resource) and the provider (where the migration is applied to all resources), respectively. The value of k increases to $k = 4$ and $k = 5$ for mitigation by the tenant and provider, respectively. The result of mitigation by the provider is slightly better than by the tenant because more VMs may be migrated.

- Use Case D: In this case, we apply the BN cross-layer resource graph for three different tenants co-resident with the target tenant. Figure 47 shows a BN cross-layer resource graph; the red color represents the target node, and the blue, green and

Table 11: The optimal solution to the motivating example (Section 5.2.4)

Host	VM / VDU			
1	app A	app B	DB A	DB B
2	LB A / Switch A	Router A	http B	http A
3	FW A / IDS A	FW B / IDS B	http M	Router B
	Before mitigation $k = 2$			

Host	VM / VDU			
1	app A	app B	DB A	DB B
2	LB A / Switch A	FW A / IDS A	http B	http A
3	Router A	FW B / IDS B	http M	Router B
	After mitigation by tenant $k = 4$			

Host	VM / VDU			
1	app A	app B	DB A	FW A / IDS A
2	LB A / Switch A	Router A	http B	http A
3	DB B	FW B / IDS B	http M	Router B
	After mitigation by provider $k = 5$			

orange represent attack paths for each tenant. The under-lined numbers in the figure represent the probability to exploit the vulnerability on that node. Table 12 shows the BN result before optimization which shows that the orange tenant has the highest probability to reach the target. We apply optimization to reduce the probability of reaching the target. After we apply our mitigation technique by migrating a VM belonging to the orange tenant from host 2 to host 3, the probability to reach the target by this tenant droops from 0.037 to 0.027. In this case we have improved the security by only migrating one VM.

- Use Case E: This case shows the network slicing example in Section 5.2.3. In this example, we assume the attacker is a user who uses the corporate slice to trip circuit breakers on the control slice. We use the *k*0d safety security metric to measure the security level. The attacker in Figure 45 needs to exploit <http,0,3> or <ssh,0,3>, then the <Xen,3,h3> to reach the control slice because vIED runs on top of the VM

Table 12: The BN result for Figure 47 (Malicious tenant: M, Not malicious: X)

Tenant	Status						
Blue	M	X	X	M	M	X	M
Green	X	M	X	M	X	M	M
Orange	X	X	M	X	M	M	M
Result	0.024	0.021	0.036	0.023	0.03	0.029	0.027

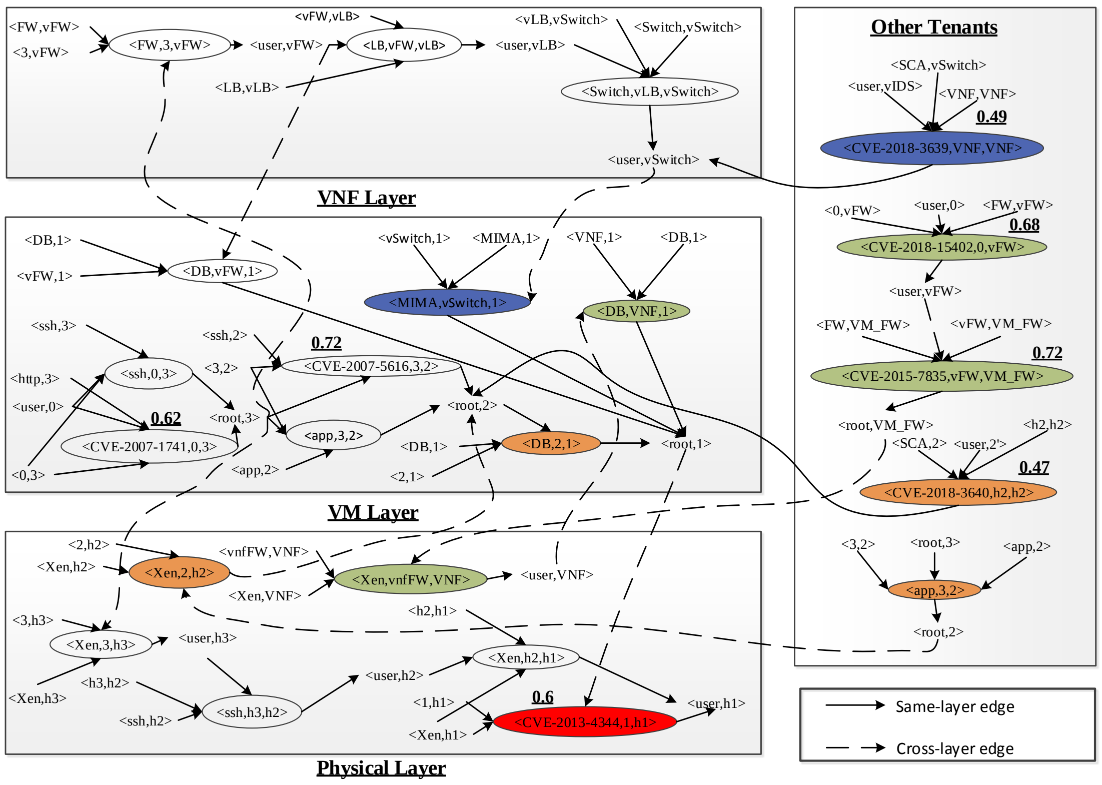


Figure 47: A Bayesian network cross-layer resource graph

hosted on h3; the value of k in this scenario is $k = 2$. We can improve the value of k after we apply our optimization solution; migrating the VM running $http$ on the corporate slice from h3 to h2 increases the value of k from $k = 2$ to $k = 3$.

5.5 Simulations

This section shows the simulation results of applying our mitigation solution under various constraints. All VM placement in the simulations are based on CloudSim [43, 34, 66]. We applied the three placement policies in CloudSim (i.e., the random, least, and most policies) to our NFV environment. We have 300 hosts and 7,000 VMs using the default configurations for the host and VMs from CloudSim.

Moreover, we use a virtual machine equipped with a 3.4 GHz CPU and 8 GB RAM in the Python 2.7.10 environment under Ubuntu 12.04 LTS and the MATLAB R2017b's GA toolbox. To generate a large number of resource graphs for simulations, we start with seed graphs with realistic configurations similar to Figure 43 and the cloud infrastructure configurations presented in [11, 12], and then generate random resource graphs by injecting new nodes and edges into those seed graphs based on the VM placement results of CloudSim. Those resource graphs were used as the input to the optimization toolbox where the fitness function maximizes the average insider threat value k under various constraints. The parameters and constraints used in our simulations include the VMs placement policy, size of the network, type and number of attackers, and maximum number of VMs migrating to malicious users. We repeat each simulation on 400 different resource graphs to obtain the average result.

The objective of the first simulation is to study how cross-layer attacks may affect the security of the NFV stack. We compare the $k0d$ metric on the original resource graph (without any cross-layer attacks), and cross-layer resource graphs. In Figure 48, the number of malicious users (external attackers or insider tenants) is between 5 and 15, while the size of the network varies between 50 and 1,500 along the X-axis. The Y-axis shows the average of k among all malicious users. The red line represents the results of the original resource graph without considering malicious tenants in this particular case. The green line represents the results of the original resource graph while considering malicious tenants.

The blue line shows the result of cross-layer resource graph (with both malicious tenants and cross-layer attacks considered).

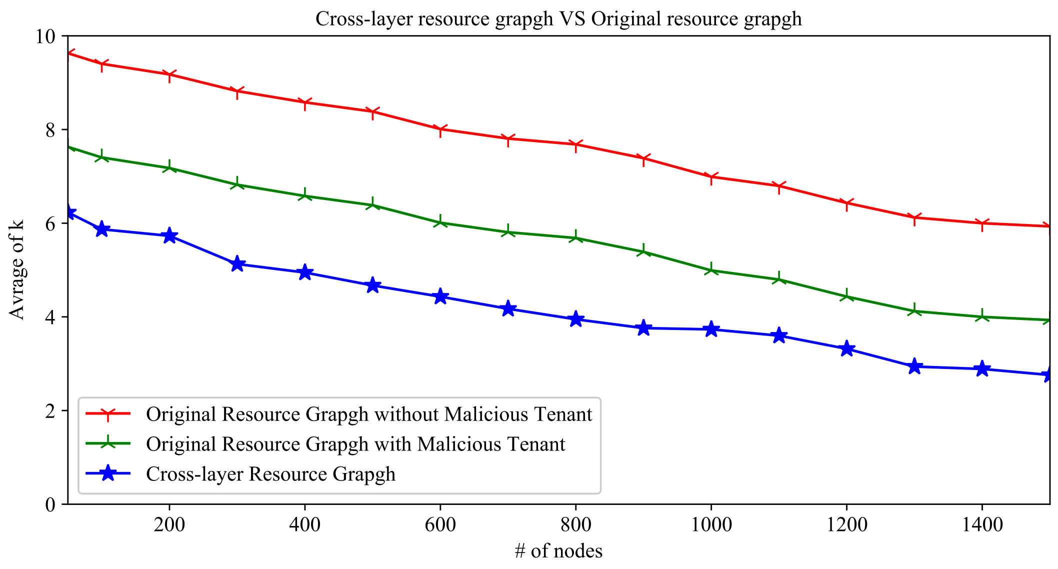


Figure 48: Comparing the original resource graph and cross-layer resource graph

Results and Implications: From the results, we can make the following observations. First, the value of k decreases in all cases almost linearly; this is expected because, as the size of the network increases, there is a higher chance for the length of the shortest path to decrease, which means attackers may require less attack steps. Second, the value of k drops on the original resource graph (without considering cross-layer attacks) after considering the presence of malicious tenants (i.e., co-residency attacks), which is as expected. Third, the value of k drops by approximately 55% between the original resource graph without considering the malicious tenant, and the cross-layer resource graph, which shows the additional threat of cross-layer attacks.

In Figure 49 the objective is to show how different placement policies can affect the value of k. In this simulation, we employ the cross-layer resource graph to measure the value of k for three types of attackers (external, malicious tenant, and lower-layer provider who has access to all the hosts) under three different placement policies used in CloudSim (i.e., the most, least, and random policies). The three figures show how the three placement policies can slightly affect the value of k. Each trend on the figure shows a different type of attackers, while the total number of attackers stays between 5 and 20. The X-axis depicts the size of the network and the Y-axis shows the average value of k among all attackers.

Results and Implications: From the three figures, we can make the following observations. First, similar as in previous results, the value of k in all trends decreases almost linearly as the size of the network increases. Second, the trends of external attackers and malicious tenants decrease faster than the lower-layer provider. This is expected because the lower-layer provider already has access to all the hosts, which enables him/her to either use his/her privileges to attack higher layers, which means much lower k values and hence less room for further decrease as the network size increases. Finally, the most placement policy has the highest value of k both external attackers and malicious tenants and the lowest k for lower layer provider. This is because, under the most policy, the target tenant's VMs tend to stay closer to each other, which renders them less vulnerable to external attackers or malicious tenants, but more so to a lower-layer provider managing the hosts of such VMs.

The objective of the next three simulations is to study how the different types of attackers behave under our attack mitigation solution. Figure 50 shows the simulation of applying the mitigation solution on the least placement policy for external attackers and malicious tenants, and the placement policy for the lower-layer provider. The three simulations are based on similar X and Y axis as in previous simulations. The solid lines represent the results after applying our mitigation solution under the constraints of the maximum number of VMs migration. The dashed lines represent the results before applying the mitigation solution.

Results and Implications: From the simulation results, we can make the following

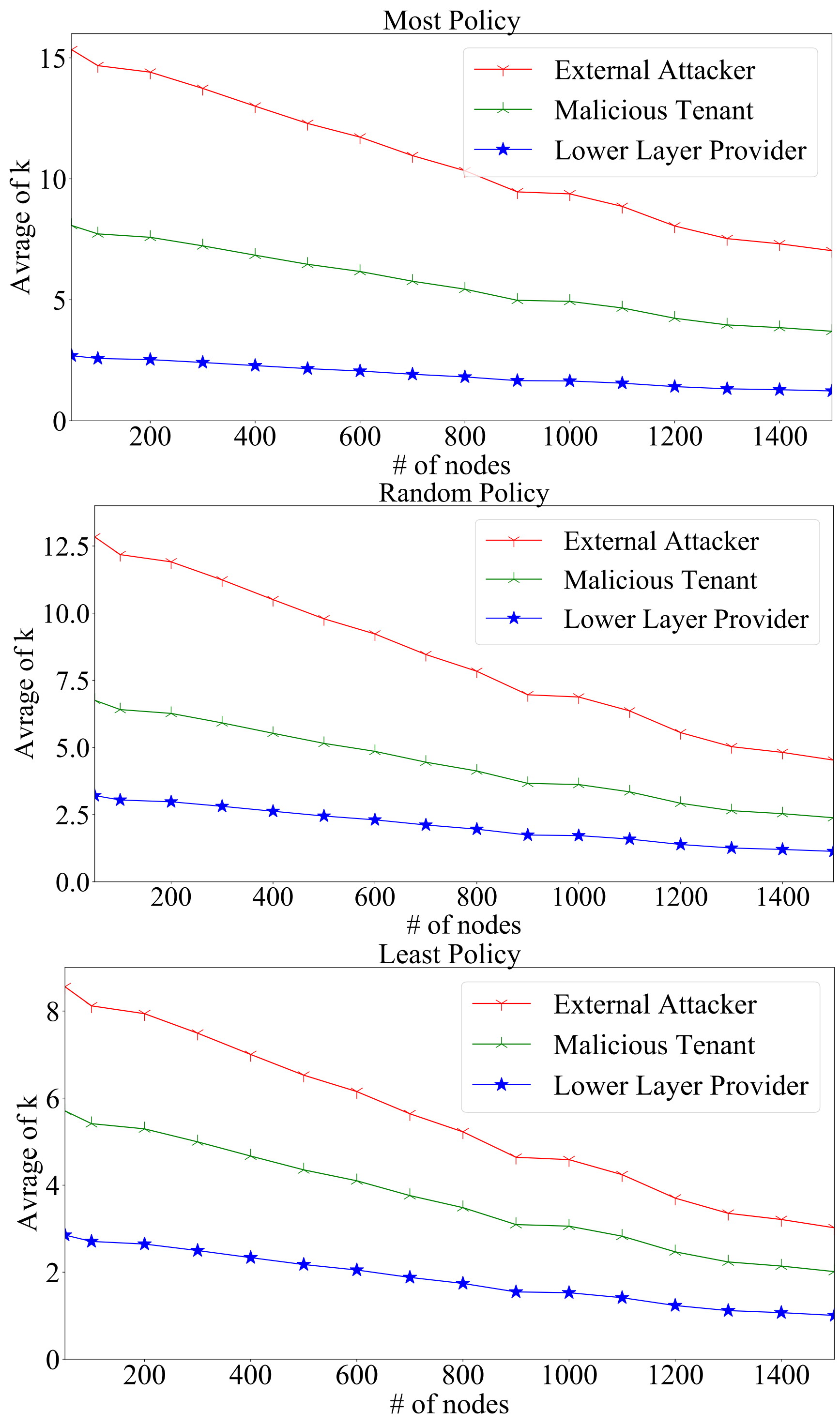


Figure 49: Comparing the three placement policies in CloudSim

observations. First, our solution is improving the value of *k* in all cases. Second, all three simulations follow the same trend and the value of *k* improves when we increase the

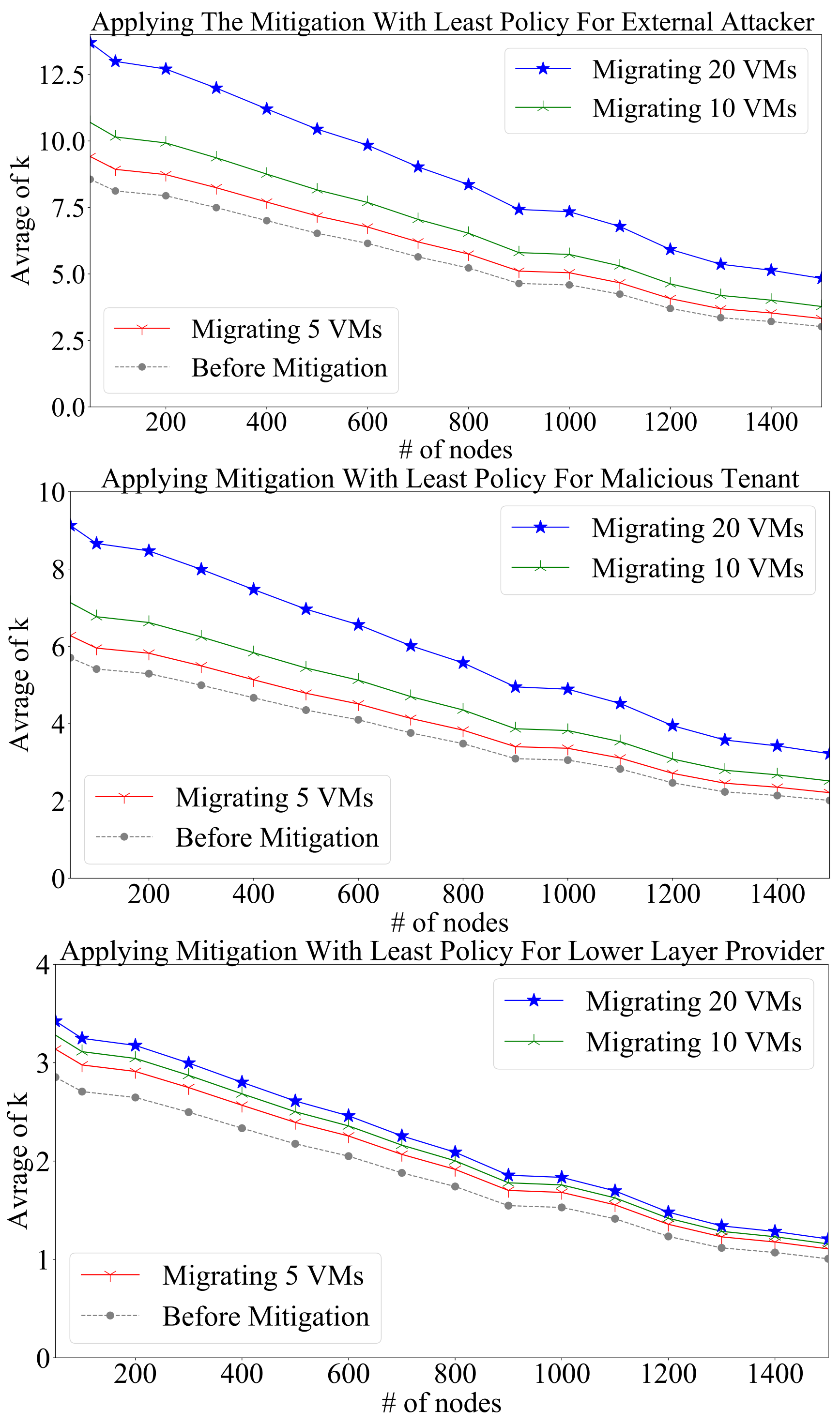


Figure 50: Applying mitigation solution with the maximum number of VMs migrating

maximum number of VMs that are allowed to migrate, i.e., the cost of migration. Finally, improving the result for the lower-layer provider is difficult to attain because the low-layer provider already is assumed to have the power to access more than one host (based on the privilege he/she has) so migration has less effect.

The objective of the last simulation is to study how the number of malicious tenants can increase insider threat under different placement policies, and how the mitigation solution may improve the value of k in each case. In Figure 51, the size of the network is fixed at 700 nodes, while the number of malicious tenants is varied between 0 and 25 along the X-axis. The Y-axis shows the average value of k among all malicious tenants. The solid lines represent the results after applying the mitigation solution, and the dashed lines are for the corresponding results before applying the solution.

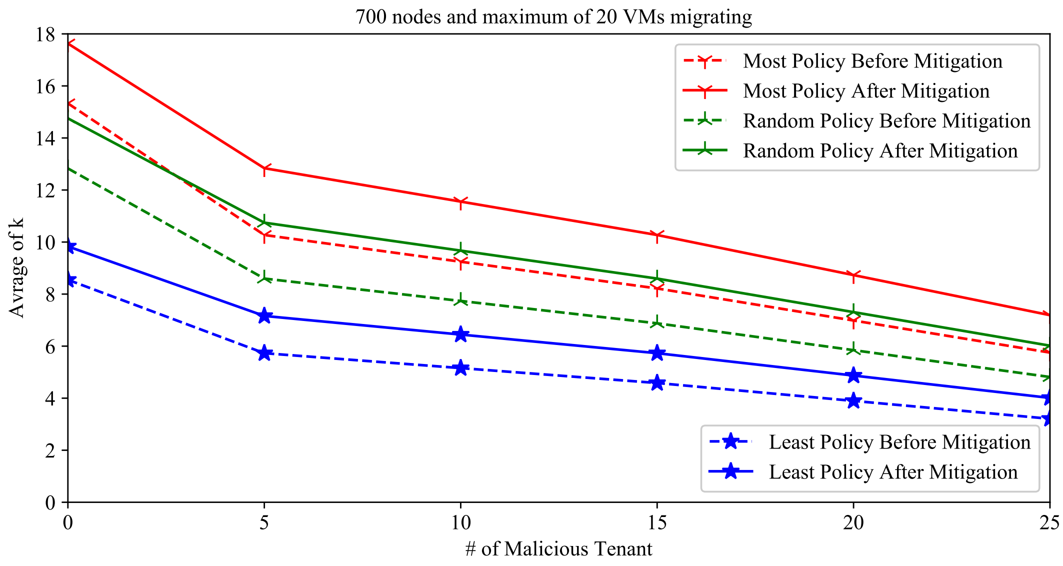


Figure 51: The results of the mitigation solution under different placement policies

Results and Implications: From the results, we can make the following observations. First, the mitigation solution successfully reduces the insider threat (increasing the average of k values) in all cases. Second, the results before and after applying the solution start with a sharp decrease prior to following similar linear trends (meaning increased insider threat) as the number of malicious tenants increase from zero. Finally, the result of the random placement policy after applying the solution is slightly better than the result of the most placement policy before applying the solution, which means that the mitigation solution may improve the placement algorithm w.r.t. the co-residency attack.

5.6 Summary

In this chapter, we have modeled cross-level and co-residency attacks in the NFV stack. We have also formulated the optimal VNF/VM placement problem to mitigate the security threats through standard optimization algorithm. We have extended our NFV stack models to consider network slicing. Also, we applied the BN-base metric to take all attack paths and known vulnerabilities into consideration. Furthermore, we conducted simulations whose results showed that our solution could significantly reduce the level of security threats in NFV.

Chapter 6

Conclusion

This chapter concludes the findings of this thesis and highlights the future directions.

6.1 Concluding Remarks

Cloud infrastructure threat modeling and security metrics are important to improving a cloud system's security level because "you cannot improve what you cannot measure". My Ph.D. research was among the first efforts toward developing a systematic approach to cloud threat modeling and security metrics based on fictitious but realistic cloud data centers by integrating established technologies used by several major players in the cloud market to encompass various cloud infrastructure abstraction levels.

The three components of this thesis are related as follows. In the first topic, we study various cloud infrastructure threat models and security metrics devised by integrating existing technologies used by well-known cloud providers to protect against internal and external attackers. In addition, we apply several hardening options to show how to improve cloud infrastructures' security levels. In the second topic, we use the cloud infrastructure presented on the first topic to model the threat posed by a specific attacker type. In this topic, we model and mitigate threats related to insider attackers (third-party remote administrators) during maintenance task assignments. In addition, we use a standard optimization algorithm to mitigate the security threats. In the last topic, we consider the NFV stack to model a cross-layer attack, which is specific to NFV and the co-residency attack, which can happen inside each layer. In addition, we mitigate the security threats through VM placement and migration using a standard optimization algorithm.

6.2 Future Directions

In the first topic, our future work will be directed to developing a systematic approach to integrating those different threat models and making the generation and analysis of such models more scalable for clouds.

In the second topic, the limitations and corresponding future directions are as follows. First, the current mitigation solution is static in nature, and we will devise incremental solutions to handle streams of new maintenance tasks and dynamics (joining or leaving) of RAs, changing priority or weight of tasks, etc.). Second, our model has kept the cost implicit, and we will consider explicit cost models (e.g., based on the nature of the tasks, the amount or duration of tasks, and privileges needed) and incorporate such cost models into the mitigation solution.

In the last topic, our future work will focus on following directions. First, we will make our solution incremental and more efficient in order to handle more dynamics in terms of VM placement and immigration. Second, we will consider weighing different exploits and asset values to optimally choose among those different options for a given NFV application. Finally, we will study the integration of our solution with existing deployment policies based on an NFV testbed.

Bibliography

[1] National vulnerability database. http://www.nvd.org. [Online; accessed 20/02/2015].

[2] National Institute of Standards and Technology: Cloud Computing Service Metrics Description. http://www.nist.gov/itl/cloud/upload/RATAX-CloudServiceMetricsDescription-DRAFT-20141111.pdf, 2015. [Online; accessed 17/06/2015].

[3] Amazon Web Services. https://aws.amazon.com/, 2018. [Online; accessed 28/02/2018].

[4] Google Cloud Platform. https://cloud.google.com/, 2018. [Online; accessed 28/02/2018].

[5] Microsoft Azure. https://azure.microsoft.com, 2018. [Online; accessed 28/02/2018].

[6] W. L. A. Dutta, K. Sood. Network functions virtualisation (nfv) release 3; security; security management and monitoring specification. Technical report, European Telecommunications Standards Institute, 2017.

[7] B. Adler. Google Compute Engine Performance Test with RightScale and Apica. http://www.rightscale.com/blog/cloud-industry-

`insights/google-compute-engine-performance-test-rightscale-and-apica`, 2013. [Online; accessed 26/03/2016].

[8] M. Albanese and S. Jajodia. A graphical model to assess the impact of multi-step attacks. *The Journal of Defense Modeling & Simulation*, 15:79–93, 01 2018.

[9] M. Albanese, S. Jajodia, and S. Noel. Time-efficient and cost-effective network hardening using attack graphs. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, pages 1–12, June 2012.

[10] N. Alhebaishi, L. Wang, S. Jajodia, and A. Singhal. *Threat Modeling for Cloud Data Center Infrastructures*, pages 302–319. Springer International Publishing, Cham, 2017.

[11] N. Alhebaishi, L. Wang, S. Jajodia, and A. Singhal. Mitigating the insider threat of remote administrators in clouds through maintenance task assignments. *Journal of Computer Security*, 27(4):427–458, 2019.

[12] N. Alhebaishi, L. Wang, and A. Singhal. Threat modeling for cloud infrastructures. *ICST Trans. Security Safety*, 5(17):e5, 2019.

[13] J. Almasizadeh and M. A. Azgomi. Mean privacy: A metric for security of computer systems. *Computer Communications*, 52(Supplement C):47 – 59, 2014.

[14] A. K. Alnaim, A. M. Alwakeel, and E. B. Fernandez. Threats against the virtual machine environment of nfv. In *2019 2nd International Conference on Computer Applications Information Security (ICCAIS)*, pages 1–5, May 2019.

[15] Q. Althebyan and B. Panda. A knowledge-base model for insider threat prediction. In *2007 IEEE SMC Information Assurance and Security Workshop*, pages 239–246, June 2007.

[16] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 217–224, 2002.

[17] S. Attarha, A. Narayan, B. Hage Hassan, C. Krüger, F. Castro, D. Babazadeh, and S. Lehnhoff. Virtualization management concept for flexible and fault-tolerant smart grid service provision. *Energies*, 13(9):2196, 2020.

[18] A. O. F. Atya, Z. Qian, S. V. Krishnamurthy, T. F. L. Porta, P. D. McDaniel, and L. M. Marvel. Catch me if you can: A closer look at malicious co-residency on the cloud. *IEEE/ACM Trans. Netw.*, 27(2):560–576, 2019.

[19] K. Bakshi. Cisco cloud computing-data center strategy, architecture, and solutions. *DOI= http://www. cisco. com/web/strategy/docs/gov/CiscoCloudComputing_WP. pdf*, 2009.

[20] C. Balding. GoGrid Security Breach. `http://cloudsecurity.org/blog/2011/03/30/gogrid-security-breach.html`. [Online; accessed 10/03/2016].

[21] D. Balzarotti, M. Monga, and S. Sicari. Assessing the risk of using vulnerable components. In D. Gollmann, F. Massacci, and A. Yautsiukhin, editors, *Quality of Protection - Security Measurements and Metrics*, volume 23 of *Advances in Information Security*, pages 65–77. Springer, 2006.

[22] J. Barr. Building three-tier architectures with security groups. `https://aws.amazon.com/blogs/aws/building-three-tier-architectures-with-security-groups/`, 2010. [Online; accessed 28/03/2016].

[23] C. Basile, A. Lioy, C. Pitscheider, F. Valenza, and M. Vallini. A novel approach for integrating security policy enforcement with dynamic network virtualization. In *NetSoft'15*, pages 1–5, 2015.

[24] A. Bates, B. Mood, J. Pletcher, H. Pruse, M. Valafar, and K. Butler. Detecting co-residency with active traffic analysis techniques. In *Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop*, CCSW '12, page 1–12, New York, NY, USA, 2012. Association for Computing Machinery.

[25] J. Bayuk. Cloud security metrics. In *2011 6th International Conference on System of Systems Engineering*, pages 341–345, 2011.

[26] J. Bayuk and A. Mostashari. Measuring systems security. *Systems Engineering*, 16(1):1–14, 2013.

[27] M. Bishop, S. Engle, D. A. Frincke, C. Gates, F. L. Greitzer, S. Peisert, and S. Whalen. A risk management approach to the "insider threat". In C. W. Probst, J. Hunker, D. Gollmann, and M. Bishop, editors, *Insider Threats in Cyber Security*, volume 49 of *Advances in Information Security*, pages 115–137. Springer, 2010.

[28] M. Bishop, S. Engle, S. Peisert, S. Whalen, and C. Gates. We have met the enemy and he is us. In *Proceedings of the 2008 New Security Paradigms Workshop*, NSPW '08, pages 1–12, New York, NY, USA, 2008. ACM.

[29] S. Bleikertz, A. Kurmus, Z. A. Nagy, and M. Schunter. Secure cloud maintenance: Protecting workloads against insider attacks. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '12, pages 83–84, New York, NY, USA, 2012. ACM.

[30] D. Borbor, L. Wang, S. Jajodia, and A. Singhal. Diversifying network services under cost constraints for better resilience against unknown attacks. In *Data and*

Applications Security and Privacy XXX - 30th Annual IFIP WG 11.3 Conference, DBSec 2016, Trento, Italy, July 18-20, 2016. Proceedings, pages 295–312, 2016.

[31] D. Borbor, L. Wang, S. Jajodia, and A. Singhal. Securing networks against unpatchable and unknown vulnerabilities using heterogeneous hardening options. In *Data and Applications Security and Privacy XXXI - 31st Annual IFIP WG 11.3 Conference, DBSec 2017, Philadelphia, PA, USA, July 19-21, 2017, Proceedings*, pages 509–528, 2017.

[32] B. Bordel, A. B. Orúe, R. Alcarria, and D. Sánchez-De-Rivera. An intra-slice security solution for emerging 5g networks based on pseudo-random number generators. *IEEE Access*, 6:16149–16164, 2018.

[33] T. Branco, Jr. and H. Santos. What is missing for trust in the cloud computing? In *Proceedings of the 2016 ACM SIGMIS Conference on Computers and People Research*, SIGMIS-CPR '16, pages 27–28, New York, NY, USA, 2016. ACM.

[34] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw., Pract. Exper.*, 41(1):23–50, 2011.

[35] E. Cayirci, A. Garaga, A. Santana de Oliveira, and Y. Roudier. A risk assessment model for selecting cloud service providers. *Journal of Cloud Computing*, 5(1):14, Sep 2016.

[36] A. Chen. GCreep: Google Engineer Stalked Teens, Spied on Chats(Updated). `http://gawker.com/5637234/gcreep-google-engineer-stalked-teens-spied-on-chats`. [Online; accessed 10/03/2016].

[37] F. Chen, L. Wang, and J. Su. An efficient approach to minimum-cost network hardening using attack graphs. In *2008 The Fourth International Conference on Information Assurance and Security*, pages 209–212, 2008.

[38] X. Chen, M. Zhang, Z. M. Mao, and P. Bahl. Automating network application dependency discovery: Experiences, limitations, and new solutions. In *8th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2008, December 8-10, 2008, San Diego, California, USA, Proceedings*, pages 117–130, 2008.

[39] Y. Chen, V. Paxson, and R. H. Katz. What's new about cloud computing security. *University of California, Berkeley Report No. UCB/EECS-2010-5, (2010).*, 2010. [Online; accessed 10/09/2017].

[40] R. Chinchani, A. Iyer, H. Q. Ngo, and S. Upadhyaya. Towards a theory of insider threat assessment. In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 108–117, June 2005.

[41] W. R. Claycomb and A. Nicoll. Insider threats to cloud computing: Directions for new research challenges. In *2012 IEEE 36th Annual Computer Software and Applications Conference*, pages 387–394, July 2012.

[42] Cloud Security Alliance. Security guidance for critical areas of focus in cloud computing v 3.0, 2011.

[43] CloudSim. CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services. `http://www.cloudbus.org/cloudsim/`, 2020. [Online; accessed 27/01/2020].

[44] M. Coughlin, E. Keller, and E. Wustrow. Trusted click: Overcoming security issues of NFV in the cloud. In *SDN-NFV@CODASPY'17*, pages 31–36, 2017.

[45] CSA Security. Trust & assurance registry (STAR). https://cloudsecurityalliance.org/star/.

[46] K. Dahbur, B. Mohammad, and A. B. Tarakji. A survey of risks, threats and vulnerabilities in cloud computing. In *Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications*, ISWSA '11, pages 12:1–12:6, New York, NY, USA, 2011. ACM.

[47] R. Dewri, N. Poolsappasit, I. Ray, and L. D. Whitley. Optimal security hardening using multi-objective optimization on attack tree models of networks. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 204–213. ACM, 2007.

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

[49] K. Djemame, D. Armstrong, J. Guitart, and M. Macias. A risk assessment framework for cloud computing. *IEEE Transactions on Cloud Computing*, 4(3):265–278, July 2016.

[50] M. Doucet and M. Lari. Cyber security and cybercrime in canada, 2017, 2018. Available at: `https://www150.statcan.gc.ca/n1/en/catalogue/71-607-X2018007`.

[51] A. J. Duncan, S. Creese, and M. Goldsmith. Insider attacks in cloud computing. In G. Min, Y. Wu, L. C. Liu, X. Jin, S. A. Jarvis, and A. Y. Al-Dubai, editors, *11th IEEE International Conference on Trust, Security and Privacy in Computing and*

Communications, TrustCom 2012, Liverpool, United Kingdom, June 25-27, 2012, pages 857–862. IEEE Computer Society, 2012.

[52] K. S. Edge, G. C. Dalton, R. A. Raines, and R. F. Mills. Using attack and protection trees to analyze threats and defenses to homeland security. In *MILCOM 2006 - 2006 IEEE Military Communications conference*, pages 1–7, Oct 2006.

[53] D. Farmer and E. H. Spafford. The COPS security checker system. In *Proceedings of the Usenix Summer 1990 Technical Conference, Anaheim, California, USA, June 1990*, pages 165–170. USENIX Association, 1990.

[54] S. K. Fayazbakhsh, M. K. Reiter, and V. Sekar. Verifiable network function outsourcing: Requirements, challenges, and roadmap. In *Workshop on Hot topics in middleboxes and network function virtualization (HotMiddlebox'13)*, pages 25–30, 2013.

[55] M. D. Firoozjaei, J. P. Jeong, H. Ko, and H. Kim. Security challenges with network functions virtualization. *Future Generation Comp. Syst.*, 67:315–324, 2017.

[56] M. Flittner, J. M. Scheuermann, and R. Bauer. Chainguard: Controller-independent verification of service function chaining in cloud computing. In *NFV-SDN'17*, pages 1–7, 2017.

[57] M. Frigault and L. Wang. Measuring network security using bayesian network-based attack graphs. In *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, pages 698–703, July 2008.

[58] L. Gallon and J. Bascou. Cvss attack graphs. In *2011 Seventh International Conference on Signal Image Technology Internet-Based Systems*, pages 24–31, 2011.

[59] Gartner. Gartner forecasts worldwide public cloud revenue to grow 21.4 percent in 2018, 2018. Available at: `https://www.gartner.com/newsroom/id/3871416`.

[60] D. E. Golberg. Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989, 1989.

[61] B. Grobauer, T. Walloschek, and E. Stöcker. Understanding cloud computing vulnerabilities. *Security & privacy, IEEE*, 9(2):50–57, 2011.

[62] N. Gruschka and M. Jensen. Attack surfaces: A taxonomy for attacks on cloud services. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 276–279, July 2010.

[63] G. Gu, P. Fogla, D. Dagon, W. Lee, and B. Skoric. Towards an information-theoretic framework for analyzing intrusion detection systems. In D. Gollmann, J. Meier, and A. Sabelfeld, editors, *Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006, Proceedings*, volume 4189 of *Lecture Notes in Computer Science*, pages 527–546. Springer, 2006.

[64] M. Gupta, J. Rees, A. Chaturvedi, and J. Chi. Matching information security vulnerabilities to organizational security profiles: a genetic algorithm approach. *Decision Support Systems*, 41(3):592 – 603, 2006. Intelligence and security informatics.

[65] T. Halabi and M. Bellaiche. Towards quantification and evaluation of security of cloud service providers. *Journal of Information Security and Applications*, 33(Supplement C):55 – 65, 2017.

[66] Y. Han, J. Chan, T. Alpcan, and C. Leckie. Virtual machine allocation policies against co-resident attacks in cloud computing. In *IEEE International Conference on Communications, ICC 2014, Sydney, Australia, June 10-14, 2014*, pages 786–792, 2014.

[67] M. Hany. VMware VSphere In The Enterprise. `http://www.hypervizor.com/diags/HyperViZor-Diags-VMW-vS4-Enterprise-v1-0.pdf`. [Online; accessed 05/02/2015].

[68] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal. Nfv: state of the art, challenges, and implementation in next generation mobile networks (vepc). *IEEE Network*, 28(6):18–26, 2014.

[69] J. Homer, S. Zhang, X. Ou, D. Schmidt, Y. Du, S. R. Rajagopalan, and A. Singhal. Aggregating vulnerability metrics in enterprise networks using attack graphs. *J. Comput. Secur.*, 21(4):561–597, 2013.

[70] W. Huang, H. Zhu, and Z. Qian. Autovnf: An automatic resource sharing schema for VNF requests. *J. Internet Serv. Inf. Secur.*, 7(3):34–47, 2017.

[71] B. W. P. III, P. Ning, and S. Jajodia. On the accurate identification of network service dependencies in distributed systems. In *Strategies, Tools , and Techniques: Proceedings of the 26th Large Installation System Administration Conference, LISA 2012, San Diego, CA, USA, December 9-14, 2012*, pages 181–194, 2012.

[72] J. A. Ingalsbe, D. Shoemaker, and N. R. Mead. Threat modeling the cloud computing, mobile device toting, consumerized enterprise-an overview of considerations. In *AMCIS*, 2011.

[73] Intel. Realising the benefits of network functions virtualisation in telecoms network. `https://www.intel.com/content/dam/www/public/us/`

`en/documents/white-papers/benefits-network-functions-virtualization-telecoms-paper.pdf./`.

[74] ISO Std IEC. ISO 27017. *Information technology- Security techniques- Code of practice for information security controls based on ISO/IEC 27002 for cloud services (DRAFT)*, `http://www.iso27001security.com/html/27017.html`, 2012.

[75] A. K. B. Ixia. Network Function Virtualization (NFV): 5 Major Risks. `https://www.ixiacom.com/resources/network-function-virtualization-nfv-5-major-risks/`.

[76] G. Jakobson. Mission cyber security situation assessment using impact dependency graphs. In *14th International Conference on Information Fusion*, pages 1–8, July 2011.

[77] W. Jiang, H. Zhang, Z. Tian, and X. Song. A game theoretic method for decision and analysis of the optimal active defense strategy. In *2007 International Conference on Computational Intelligence and Security (CIS 2007)*, pages 819–823, 2007.

[78] V. Kann. A compendium of np optimization problems. In *WWW Spring 1994*, 1994.

[79] L. U. Khan, I. Yaqoob, N. H. Tran, Z. Han, and C. S. Hong. Network slicing: Recent advances, taxonomy, requirements, and open research challenges. *IEEE Access*, 8:36009–36028, 2020.

[80] S. Lakshmanan Thirunavukkarasu, M. Zhang, A. Oqaily, G. S. Chawla, L. Wang, M. Pourzandi, and M. Debbabi. Modeling nfv deployment to identify the cross-level inconsistency vulnerabilities. In *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 167–174, Dec 2019.

[81] S. Lal, T. Taleb, and A. Dutta. Nfv: Security threats and best practices. *IEEE Communications Magazine*, 55(8):211–217, AUGUST 2017.

[82] N. T. Le and D. B. Hoang. Security threat probability computation using markov chain and common vulnerability scoring system. In *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6, 2018.

[83] R. M. Lee, M. J. Assante, and T. Conway. Analysis of the cyber attack on the ukrainian power grid. `https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf`.

[84] W. Lee and D. Xiang. Information-theoretic measures for anomaly detection. In *2001 IEEE Symposium on Security and Privacy, Oakland, California, USA May 14-16, 2001*, pages 130–143. IEEE Computer Society, 2001.

[85] M. Li, W. Zang, K. Bai, M. Yu, and P. Liu. Mycloud: Supporting user-configured privacy protection in cloud computing. In *Proceedings of the 29th Annual Computer Security Applications Conference*, ACSAC '13, pages 59–68, New York, NY, USA, 2013. ACM.

[86] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain. Network slicing for 5g: Challenges and opportunities. *IEEE Internet Computing*, 21(5):20–27, 2017.

[87] Y. Liu, Y. L. Sun, J. Ryoo, S. Rizvi, and A. V. Vasilakos. A survey of security and privacy challenges in cloud computing: Solutions and future directions. *JCSE*, 9(3), 2015.

[88] J. Luna, H. Ghani, D. Germanus, and N. Suri. A security metrics framework for the cloud. In *Security and Cryptography (SECRYPT), 2011 Proceedings of the International Conference on*, pages 245–250, July 2011.

[89] T. Madi, M. Zhang, Y. Jarraya, A. Alimohammadifar, M. Pourzandi, L. Wang, and M. Debbabi. Quantic: Distance metrics for evaluating multi-tenancy threats in public cloud. In *2018 IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2018, Nicosia, Cyprus, December 10-13, 2018*, pages 163–170, 2018.

[90] L. A. Maglaras, G. Drivas, K. Noou, and S. Rallis. NIS directive: The case of greece. *EAI Endorsed Trans. Security Safety*, 4(14):e1, 2018.

[91] P. Manadhata and J. Wing. An attack surface metric. *Software Engineering, IEEE Transactions on*, 37(3):371–386, May 2011.

[92] G. Marchetto, R. Sisto, J. Yusupov, and A. Ksentini. Virtual network embedding with formal reachability assurance. In *CNSM'18*, pages 368–372, 2018.

[93] S. Mathew, S. Upadhyaya, D. Ha, and H. Q. Ngo. Insider abuse comprehension through capability acquisition graphs. In *2008 11th International Conference on Information Fusion*, pages 1–8, June 2008.

[94] J. McHugh. Quality of protection: measuring the unmeasurable? In *Proceedings of the 2nd ACM Workshop on Quality of Protection, QoP 2006, Alexandria, VA, USA, October 30, 2006*, pages 1–2, 2006.

[95] P. Mell, K. Scarfone, and S. Romanosky. Common vulnerability scoring system. *IEEE Security & Privacy*, 4(6):85–89, 2006.

[96] Y. Mundada, A. Ramachandran, and N. Feamster. Silverline: Data and network isolation for cloud services. In I. Stoica and J. Wilkes, editors, *3rd USENIX*

Workshop on Hot Topics in Cloud Computing, HotCloud'11, Portland, OR, USA, June 14-15, 2011. USENIX Association, 2011.

[97] A. Natarajan, P. Ning, Y. Liu, S. Jajodia, and S. E. Hutchinson. Nsdminer: Automated discovery of network service dependencies. In *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*, pages 2507–2515, 2012.

[98] NGMN. `https://www.ngmn.org/`.

[99] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs. Efficient minimum-cost network hardening via exploit dependency graphs. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, pages 86–95, Dec 2003.

[100] W. Nzoukou, L. Wang, S. Jajodia, and A. Singhal. A unified framework for measuring a network's mean time-to-compromise. In *2013 IEEE 32nd International Symposium on Reliable Distributed Systems*, pages 215–224, Sep. 2013.

[101] Openstack. Openstack Operations Guide. `http://docs.openstack.org/openstack-ops/content/openstack-ops_preface.html`. [Online; accessed 27/08/2015].

[102] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira. Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges. *IEEE Communications Magazine*, 55(5):80–87, 2017.

[103] R. Ortalo, Y. Deswarte, and M. Kaâniche. Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Trans. Software Eng.*, 25(5):633–650, 1999.

[104] J. Pamula, S. Jajodia, P. Ammann, and V. Swarup. A weakest-adversary security metric for network configuration security analysis. In G. Karjoth and F. Massacci,

editors, *Proceedings of the 2nd ACM Workshop on Quality of Protection, QoP 2006, Alexandria, VA, USA, October 30, 2006*, pages 31–38. ACM, 2006.

[105] M. Pattaranantakul, R. He, A. Meddahi, and Z. Zhang. Secmano: Towards network functions virtualization (nfv) based security management and orchestration. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 598–605, Aug 2016.

[106] M. Pattaranantakul, R. He, Q. Song, Z. Zhang, and A. Meddahi. Nfv security survey: From use case driven threat analysis to state-of-the-art countermeasures. *IEEE Communications Surveys Tutorials*, 20(4):3330–3368, Fourthquarter 2018.

[107] S. C. Payne. A guide to security metrics. *SANS Institute Information Security Reading Room*, 2006.

[108] M. Pendleton, R. Garcia-Lebron, J.-H. Cho, and S. Xu. A survey on systems security metrics. *ACM Comput. Surv.*, 49(4):62:1–62:35, Dec. 2016.

[109] I. Petri, O. F. Rana, G. C. Silaghi, and Y. Rezgui. Risk assessment in service provider communities. *Future Generation Computer Systems*, 41(Supplement C):32 – 43, 2014.

[110] C. A. Phillips and L. P. Swiler. A graph-based system for network-vulnerability analysis. In B. Blakley, D. M. Kienzle, M. E. Zurko, and S. J. Greenwald, editors, *Proceedings of the 1998 Workshop on New Security Paradigms, Charlottsville, VA, USA, September 22-25, 1998*, pages 71–79. ACM, 1998.

[111] N. Poolsappasit, R. Dewri, and I. Ray. Dynamic security risk management using bayesian attack graphs. *Dependable and Secure Computing, IEEE Transactions on*, 9(1):61–74, 2012.

[112] M. Rates Crippa, P. Arnold, V. Friderikos, B. Gajic, C. Guerrero, O. Holland, I. Labrador Pavon, V. Sciancalepore, D. v. Hugo, S. Wong, F. Z. Yousaf, and

B. Sayadi. Resource sharing for a 5g multi-tenant and multi-service architecture. In *European Wireless 2017; 23th European Wireless Conference*, pages 1–6, May 2017.

[113] I. Ray and N. Poolsapassit. *Computer Security – ESORICS 2005: 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005. Proceedings*, chapter Using Attack Trees to Identify Malicious Attacks from Authorized Insiders, pages 231–246. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[114] M. K. Reiter and S. G. Stubblebine. Toward acceptable metrics of authentication. In *1997 IEEE Symposium on Security and Privacy, May 4-7, 1997, Oakland, CA, USA*, pages 10–20. IEEE Computer Society, 1997.

[115] M. K. Reiter and S. G. Stubblebine. Authentication metric analysis and design. *ACM Trans. Inf. Syst. Secur.*, 2(2):138–158, 1999.

[116] R. W. Ritchey and P. Ammann. Using model checking to analyze network vulnerabilities. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S P 2000*, pages 156–165, 2000.

[117] A. Roy, S. Sural, A. K. Majumdar, J. Vaidya, and V. Atluri. On optimal employee assignment in constrained role-based access control systems. *ACM Trans. Manage. Inf. Syst.*, 7(4):10:1–10:24, Dec. 2016.

[118] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer*, 29(2):38–47, Feb. 1996.

[119] P. Saripalli and B. Walters. Quirc: A quantitative impact and risk assessment framework for cloud security. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 280–288, July 2010.

[120] A. Sarkar, S. Köhler, S. Riddle, B. Ludaescher, and M. Bishop. Insider attack identification and prevention using a declarative approach. In *2014 IEEE Security and Privacy Workshops*, pages 265–276, May 2014.

[121] D. Sattar and A. Matrawy. Optimal slice allocation in 5g core networks. *IEEE Networking Letters*, 1(2):48–51, 2019.

[122] R. E. Sawilla and X. Ou. Identifying critical attack assets in dependency attack graphs. In *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, pages 18–34, 2008.

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

[124] F. B. Shaikh and S. Haider. Security threats in cloud computing. In *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*, pages 214–219, Dec 2011.

[125] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 273–284, 2002.

[126] O. Sheyner and J. M. Wing. Tools for generating and analyzing attack graphs. In F. S. de Boer, M. M. Bonsangue, S. Graf, and W. P. de Roever, editors, *Formal Methods for Components and Objects, Second International Symposium, FMCO 2003, Leiden, The Netherlands, November 4-7, 2003, Revised Lectures*, volume 3188 of *Lecture Notes in Computer Science*, pages 344–372. Springer, 2003.

[127] S. Shigeta, H. Yamashima, T. Doi, T. Kawai, and K. Fukui. *Design and Implementation of a Multi-objective Optimization Mechanism for Virtual Machine*

Placement in Cloud Computing Data Center, pages 21–31. Springer International Publishing, Cham, 2013.

[128] R. Squillace. Azure infrastructure services implementation guidelines. `https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-linux-infrastructure-service-guidelines/`, 2015. [Online; accessed 28/03/2016].

[129] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1):1–11, 2011.

[130] X. Sun, J. Dai, A. Singhal, and P. Liu. Inferring the stealthy bridges between enterprise network islands in cloud using cross-layer bayesian networks. In *International Conference on Security and Privacy in Communication Networks - 10th International ICST Conference, SecureComm 2014, Beijing, China, September 24-26, 2014, Revised Selected Papers, Part I*, pages 3–23, 2014.

[131] X. Sun, A. Singhal, and P. Liu. Towards actionable mission impact assessment in the context of cloud computing. In *Data and Applications Security and Privacy XXXI - 31st Annual IFIP WG 11.3 Conference, DBSec 2017, Philadelphia, PA, USA, July 19-21, 2017, Proceedings*, pages 259–274, 2017.

[132] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-attack graph generation tool. In *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, volume 2, pages 307–321 vol.2, 2001.

[133] W. K. Sze, A. Srivastava, and R. Sekar. Hardening openstack cloud platforms against compute node compromises. In *Proceedings of the 11th ACM on Asia Conference*

on Computer and Communications Security, ASIA CCS '16, pages 341–352, New York, NY, USA, 2016. ACM.

[134] Z. Tian, Y. Sun, S. Su, M. Li, X. Du, and M. Guizani. Automated attack and defense framework for 5g security on physical and logical layers. *CoRR*, abs/1902.04009, 2019.

[135] B. Tschaen, Y. Zhang, T. Benson, S. Banerjee, J. Lee, and J.-M. Kang. SFC-Checker: Checking the correct forwarding behavior of service function chaining. In *NFV-SDN'16*, pages 134–140, 2016.

[136] L. Wang, M. Albanese, and S. Jajodia. *Network Hardening: An Automated Approach to Improving Network Security*. Springer Publishing Company, Incorporated, 2014.

[137] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An attack graph-based probabilistic security metric. In V. Atluri, editor, *Data and Applications Security XXII*, volume 5094 of *Lecture Notes in Computer Science*, pages 283–296. Springer Berlin Heidelberg, 2008.

[138] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel. k-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, 11(1):30–44, Jan 2014.

[139] L. Wang, S. Jajodia, A. Singhal, and S. Noel. *k*-zero day safety: Measuring the security risk of networks against unknown attacks. In *Computer Security - ESORICS 2010, 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings*, pages 573–587, 2010.

[140] L. Wang, S. Jajodia, and A. E. Singhal. *Network Security Metrics*. Springer, 2017.

[141] L. Wang, S. Noel, and S. Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18):3812 – 3824, 2006.

[142] L. Wang, A. Singhal, and S. Jajodia. *Measuring the Overall Security of Network Configurations Using Attack Graphs*, pages 98–112. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[143] L. Wang, A. Singhal, and S. Jajodia. Toward measuring network security using attack graphs. In *Proceedings of the 2007 ACM Workshop on Quality of Protection*, QoP '07, pages 49–54, New York, NY, USA, 2007. ACM.

[144] L. Wang, M. Zhang, S. Jajodia, A. Singhal, and M. Albanese. Modeling network diversity for evaluating the robustness of networks against zero-day attacks. In *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II*, pages 494–511, 2014.

[145] N. Wang, P. Wang, A. Alipour-Fanid, L. Jiao, and K. Zeng. Physical-layer security of 5g wireless networks for iot: Challenges and opportunities. *IEEE Internet of Things Journal*, 6(5):8169–8181, 2019.

[146] S. Wang, Z. Zhang, and Y. Kadobayashi. Exploring attack graph for cost-benefit security hardening: A probabilistic approach. *Computers & security*, 32:158–169, 2013.

[147] Y. Wang, Z. Li, G. Xie, and K. Salamatian. Enabling automatic composition and verification of service function chain. In *IWQoS'17*, pages 1–5, 2017.

[148] W. Yang and C. Fung. A survey on security in network functions virtualization. In *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*, pages 15–19. IEEE, 2016.

[149] D. Zerkle and K. N. Levitt. Netkuang - A multi-host configuration vulnerability checker. In *Proceedings of the 6th USENIX Security Symposium, San Jose, CA, USA, July 22-25, 1996*. USENIX Association, 1996.

[150] M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese. Network diversity: A security metric for evaluating the resilience of networks against zero-day attacks. *IEEE Trans. Information Forensics and Security*, 11(5):1071–1086, 2016.

[151] X. Zhang, Q. Li, J. Wu, and J. Yang. Generic and agile service function chain verification on cloud. In *IWQoS'17*, pages 1–10, 2017.

[152] Y. Zhang, A. Juels, A. Oprea, and M. K. Reiter. Homealone: Co-residency detection in the cloud via side-channel analysis. In *2011 IEEE Symposium on Security and Privacy*, pages 313–328, May 2011.

[153] Y. Zhang, W. Wu, S. Banerjee, J.-M. Kang, and M. A. Sanchez. SLA-verifier: Stateful and quantitative verification for service chaining. In *INFOCOM'17*, pages 1–9, 2017.