Cross-VM Network Attacks & their Countermeasures within Cloud Computing Environments

By

ATIF SAEED

School of Computing and Communication
LANCASTER UNIVERSITY

Submitted in accordance with the requirements for the degree of DOCTOR OF PHILOSOPHY

MARCH 24, 2020

DEDICATION

This work is dedicated to my loving parents

ACKNOWLEDGEMENTS

This thesis would not have been possible without guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this research. Before contributing to those, first of all I would like to mention my thanks to COMSATS University Islamabad, Lahore campus Pakistan for sponsoring my PhD at Lancaster University, UK.

I would also like to express my gratitude to *Prof. Awais Rashed and Dr. Peter Garraghan*, for their technical guidance and moral support from the very early stage of this research. Thank you for believing in me and providing all the needed support. Secondly, I would like to thank *Prof. Syed Asad Hussain*, my supervisor at COMSATS University Islamabad, Lahore campus Pakistan, without his knowledge and assistance, this research would not have been successful. I would like to express special thanks to *Dr. Syed Asad Ali* who was always there when I needed him. His assistance and personal guidance have been of great value on both academic and personal level, for which I am extremely grateful. Thank you for giving me the confidence and help to work out difficult situations.

Thanks to all for the time spent together and for having shared your knowledge, this was a valuable and great experience for me. For those who have contributed but not mentioned, please accept my thanks and apologize for not mentioning their name.

AUTHOR'S DECLARATION

I Atif Saeed declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: .....

PUBLICATION

The work presented in this thesis has led to papers in a prestigious refereed journal and conference.

1. Hussain, Syed Asad and Fatima, Mehwish and **Saeed, Atif** and Raza, Imran and Shahzad, Raja Khurram " Multilevel classification of security concerns in cloud computing" Applied Computing and Informatics, Elsevier.
2. **Atif Saeed**, Peter Garraghan, Barney Craggs, Dirk van der Linden, Awais Rashid, Syed Asad Hussain "A Cross-Virtual Machine Network Channel Attack via Mirroring and TAP Impersonation In: IEEE International Conference on Cloud Computing July 2-7, San Francisco, CA, USA, 2018.

Abstract

Cloud computing is a contemporary model in which the computing resources are dynamically scaled-up and scaled-down to customers, hosted within large-scale multi-tenant systems. These resources are delivered as improved, cost-effective and available upon request to customers. As one of the main trends of IT industry in modern ages, cloud computing has extended momentum and started to transform the mode enterprises build and offer IT solutions.

The primary motivation in using cloud computing model is cost-effectiveness. These motivations can compel Information and Communication Technologies (ICT) organizations to shift their sensitive data and critical infrastructure on cloud environments. Because of the complex nature of underlying cloud infrastructure, the cloud environments are facing a large number of challenges of misconfigurations, cyber-attacks, root-kits, malware instances etc which manifest themselves as a serious threat to cloud environments. These threats noticeably decline the general trustworthiness, reliability and accessibility of the cloud.

Security is the primary concern of a cloud service model. However, a number of significant challenges revealed that cloud environments are not as much secure as one would expect. There is also a limited understanding regarding the offering of secure services in a cloud model that can counter such challenges. This indicates the significance of the fact that what establishes the threat in cloud model. One of the main threats in a cloud model is of cost-effectiveness, normally cloud providers reduce cost by sharing infrastructure between multiple un-trusted VMs. This sharing has also led to several problems including co-location attacks. Cloud providers mitigate co-location attacks by introducing the concept of isolation. Due to this, a guest VM cannot interfere with its host machine, and with other guest VMs running on the same system. Such isolation is one of the prime foundations of cloud security for major public providers.

However, such logical boundaries are not impenetrable. A myriad of previous studies have demonstrated how co-resident VMs could be vulnerable to attacks through shared file systems, cache side-channels, or through compromising of hypervisor layer using rootkits. Thus, the threat of cross-VM attacks is still possible because an attacker uses one VM to control or access other VMs on the same hypervisor. Hence, multiple methods are devised for strategic VM placement in order to exploit co-residency.

Despite the clear potential for co-location attacks for abusing shared memory and disk, fine-grained cross-VM network-channel attacks have not yet been demonstrated. Current network-based attacks exploit existing vulnerabilities in networking technologies, such as ARP spoofing and DNS poisoning, which are difficult to use for VM-targeted attacks. The most commonly

discussed network-based challenges focus on the fact that cloud providers place more layers of isolation between co-resided VMs than in non-virtualized settings because the attacker and victim are often assigned to separate segmentation of virtual networks. However, it has been demonstrated that this is not necessarily sufficient to prevent manipulation of a victim VM's traffic.

This thesis presents a comprehensive method and empirical analysis on the advancement of co-location attacks in which a malicious VM can negatively affect the security and privacy of other co-located VMs as it breaches the security perimeter of the cloud model. In such a scenario, it is imperative for a cloud provider to be able to appropriately secure access to the data such that it reaches to the appropriate destination.

The primary contribution of the work presented in this thesis is to introduce two innovative attack models in leading cloud models, impersonation and privilege escalation, that successfully breach the security perimeter of cloud models and also propose countermeasures that block such types of attacks.

The attack model revealed in this thesis, is a combination of impersonation and mirroring. This experimental setting can exploit the network channel of cloud model and successfully redirects the network traffic of other co-located VMs. The main contribution of this attack model is to find a gap in the contemporary network cloud architecture that an attacker can exploit. Prior research has also exploited the network channel using ARP poisoning, spoofing but all such attack schemes have been countered as modern cloud providers place more layers of security features than in preceding settings. Impersonation relies on the already existing regular network devices in order to mislead the security perimeter of the cloud model.

The other contribution presented of this thesis is 'privilege escalation' attack in which a non-root user can escalate a privilege level by using RoP technique on the network channel and control the management domain through which attacker can manage to control the other co-located VMs which they are not authorized to do so. Finally, a countermeasure solution has been proposed by directly modifying the open source code of cloud model that can inhibit all such attacks.

LIST OF ACRONYMS

API	Application Program Interface
ARP	Address Resolution Protocol
AWS	Amazon Web Services
br-int	Integration bridge
br-ex	External bridge
CMS	Cloud Management System
CPU	Central Processing Unit
DNS	Domain Name System
dom	Domain
IaaS	Infrastructure-as-a-Service
KVM	Kernel-based Virtual Machine
NIST	National Institute of Standards and Technology
OS	OpenStack
OVS	Open Virtual Switch
PaaS	Platform-as-a-Service
QoS	Quality of Service
RAM	Random Access Memory
RoP	Return-Oriented Programming
SaaS	Software-as-a-Service
SDN	Software Defined Networking
TAP	Test Access Point
VCPU	Virtual Central Processing Unit
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMM	Virtual Machine Manager
VPN	Virtual Private Network
xapi	Xen Application Program Interface

TABLE OF CONTENTS

List of Tables

Tables **Page**

List of Figures

Figures **Page**

CHAPTER 1

INTRODUCTION

Cloud computing has revolutionized, the means of how enterprises developing and providing IT solutions. It has become the leading standard for running IT infrastructures from the smallest to the largest organizations. It noticeably has evolved as a technology, to the extent that large organizations such as Amazon, Microsoft, Oracle and Google have invested to build large scale cloud infrastructure to provision services. The question raised by many organizations here is, when and how to shift from their existing IT infrastructure to the cloud computing model. Recent report generated by Cloud Industry Forum (CIF) [1] in 2016 indicate that 63% of UK industries are planning to shift their entire IT infrastructure to the cloud in nearby future. CIF conducted an interview of 250 senior IT professionals and business decision makers and disclosed that 78% of UK organizations are willing to use the cloud. Its growth rate is 53% since when the first research was conducted in 2016 and it was predicted that by the end of 2018, this ratio would grow to 85% [2].

A strong benefit of using cloud computing to provision service is due to cost effectiveness, flexibility and on demand resource allocation activation via leveraging virtualization technology. The other advantage of adopting cloud computing services is that organizations can circumvent the upfront cost and complexity of running and maintaining their own IT infrastructure, and instead pay for what and when they use. [3–6]. Regardless of these advantages, Cloud computing faces new challenges not found within traditional distributed systems and require extensive and in-depth research to characterize and quantify real security and operational issues.

1.1 Research Motivation

In cloud computing, security is a major concern for customers to adopt this technology and move from conventional computing to cloud computing [4]. By outsourcing, users lose their physical control over data or network when it is stored in a remote server and they delegate their control to cloud providers [17], [18]. Despite powerful and reliable servers compared to client processing power and reliability, many threats are facing the cloud which can utilize cloud vulnerabilities to harm [19]. These threats may jeopardize data availability, data confidentiality by exploiting the network channel or by escalating the privilege level [20].

1.2 Security Threats in Cloud

Cloud implementations often contain advanced security technologies, mostly available due to the centralization of data and universal architecture. The resource pool leveraged in cloud computing enables the cloud provider to focus resources on securing the cloud architecture. Cloud computing has the capability to address a number of identified deficiencies of traditional architectures due to its unique characteristics, but the adoption of this innovative architecture may introduce a number of uncategorized threats [7].

Cloud security is characterized by three criteria of Confidentiality, Integrity, and Availability (CIA triad), and is a fundamental property of cloud service provisioning platforms. However, the innate properties of cloud environments such as elasticity, dynamicity, and scalability pose substantial problems towards ensuring security.

Elasticity is defined as “the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible” [8]. Resource allocation is constantly changing because of customer requirements due to which it is difficult to provide secure cloud services. Because cloud providers implement return-oriented programming (ROP) technique to add their resources and this ROP technique,in turn, is vulnerable to attack [9,15]. Virtualization introduces complex interactions between multiple virtual machines and applications sharing the same physical infrastructure. Due to this sharing an adversary can launch side or network channel attacks on other co-located VMs. The consequence of these attacks can jeopardize for other co-located VMs in terms of their data and resources[9]. Cloud environments achieve higher utilization of physical resources through the consolidation of workloads. However, this makes them more vulnerable to threats resulting from unpredictable resource demands as well as operational failures, such as hardware and software failures, unforeseen load fluctuations, and network attacks [9,10]. In Cloud environments services may be provided by an assortment of

heterogeneous platforms, and resources are shared between multiple VMs. Cloud systems comprise of a number of on-site computer organizations that may have a large number of hardware and software systems. Attackers can exploit VM infrastructure to ex-filtrate data or conduct attacks such as DDoS. The main reason for such attacks is the inherent loophole in the TCP/IP stack [10]. Additionally, a number of different cross-VM novel attacks have been launched in recent times that implement polymorphism and metamorphism schemes to avoid detection. In an IaaS cloud model, for example, data about a victim's virtual machines can be acquired and exploited; thus, assist attacks on VMs [11–13].

This thesis presents an analysis method and an in-depth analysis of large-scale cloud computing environments to identify the vulnerabilities in the existing network and hypervisor architecture of the cross-VM cloud computing model. Although, researchers have already identified a number of cross-VM attacks that exploit the network architecture [30] and hypervisor [29]. But now these attacks are controlled by the security perimeter of the cloud platform. Cloud providers are continuously enhancing their security features to secure architectures by introducing new and advanced service models or layers in the networks. As the most common challenges focus on the fact that cloud providers place more layers of isolation between co-resided VMs than in non-virtualised settings. Due to this, the attacker and the victim are often assigned to separate the segmentation of virtual networks. This isolation is also vulnerable to attack. Malicious VMs can circumvent this isolation to launch an attack in the cloud model. Such vulnerability need to be secured as it can compromise the customer's privacy or can lead to the disclosure of customer's sensitive data.

1.3 Research Problem

The investigation into the security challenges of cloud computing indicates that virtualization facilitates multiple virtual machines to share the same physical infrastructure, this in turn creates new forms of cross-VM network channel or privilege escalation attacks. The network channel attack can compromise the customer sensitive data and in privilege escalation an attacker having limited rights can perform the root operations to control other co-located VMs.

1.3.1 Research Goals and Questions

The main goal of this research is to investigate the implications of virtualisation on vulnerabilities in the network channel of cloud computing and to suggest a mitigation solution for cloud security. This goal can be further divided into the following more specific objectives.

Investigate the network architecture and their associated components for designing a zero-day attack model to exploit a vulnerability in the network architecture of the

cloud computing model. The key research questions associated with this goal are:What are the potential roles of network components in enhancing the security of network architecture of cloud computing? How does a network architecture look like and which of its components is responsible for managing network traffic?

Investigate return-oriented programming (RoP) and associated techniques for the exploitation of hypervisor to escalate the privilege level of non-root VM, and identify strengths and limitations of this approach.

The key research questions associated with this goal are:

a) What domain properties are most applicable to support isolation in the cloud?

b) How can ROP provide the insight necessary to help the exploitation of domain isolation properties of hypervisor?

Evaluate the overall approach through real-world scenarios derived from the analysis of literature and to propose a mitigation solution.

The key research questions associated with this goal are:

a) What are the appropriate methods and parameters to evaluate the proposed approach?

b) What are the potential loophole in the architecture that can be fixed by proposing countermeasures ?

1.4 Research Methodology

This work adopts an experimental systems research methodology, with an iterative approach that is based on quantitative analysis of real systems.

Experimental: Investigate the network and hypervisor architecture in cloud computing to identify the vulnerabilities. To exploit the vulnerabilities, two zero-day attacks named as *TAP Impersonation and Mirroring* and *Privilege Escalation* are created to iteratively test the exploitation of individual network and hypervisor components in the architecture. A large-scale analysis is used to study these architectures to make it more effective in a generalised manner.

Iterative: The proposed attack strategies are selected by the investigation of different traditional attack strategies. The investigation of these existing attack strategies such as ARP spoofing, IP spoofing, poisoning, Row Hammer attack is helpful in designing the proposed attack strategies to be eventually used, keeping in view the need for getting insight about the network and hypervisor architecture for the deployment setups. The intuition behind investigating these different attack strategies is the use of diverse baseline attack models that can be used for

diagnostic and accuracy assessment as well. The process of identifying a final attack strategy starts with the exploration of traditional attacks methods followed by proposing a new model. The new model is then evaluated on deployed network and hypervisor architecture and further examined using experimental methods for update and re-design. By following this iterative process and further evaluation within the network and hypervisor architecture a final attacks strategies are derived. A detailed experimental study is performed to check the feasibility of the proposed attack strategies.

Quantitative: Furthermore, our research methodology follows a quantitative assessment of real world scenarios in a real-time environment. A quantitative analysis is performed on a large scale experimental setup of two major IaaS providers; the first one is open source cloud platform i.e OpenStack and second is commercial cloud i.e Oracle Ravello System. For evaluation, multiple co-located VMs of different categories are used as a representative subset of problems. The real-time network traffic of these co-located VMs, running on different deployment setups, are used for assessment of the proposed model.

1.5 Contributions

The major contributions of the work presented in this thesis are:

The architectural insight of a network system to design a cross-VM attack, combination of TAP Impersonation and Mirroring, probe on public and private cloud computing model to investigate its feasibility. Data privacy and security is a leading concern for providers and customers of cloud computing, where Virtual Machines (VMs) can co-reside within the same underlying physical machine. Side channel attacks within multi-tenant virtualised cloud environments are an established problem. Security perimeter of cloud computing have attempted to mitigate such attacks by preventing VM-to-VM interference on shared hardware. However there is still a gap in network architecture that needs to be explored in the cloud computing model.*Design and implementation of new attack (privilege escalation) by exploiting the most ideal subset of the existing code (RoP)identified and empirical evaluation of performance.* An adversary having limited privilege rights may execute ROP, establish a connection with root domain by abusing the network channel and get the possession of tool stack (root domain) which they are not authorized to access directly. *An empirical study to evaluate the proposed attacks in real-time cloud testbed and design the most suitable countermeasures to block these type of attacks.* Existing research proposes cross-VM network channel detection algorithm [16], [17]. However, such approaches are insecure, with attackers capable of performing network channel attacks which bypass these mitigation strategies and thus launch attacks. The insight network architecture of cloud revealed that there is a vulnerability within network component. Hence, the countermeasure solution of

these attack is proposed by directly modifying the open source network code.
Additional contributions of this thesis are: A detailed study of state-of-art cloud attacks techniques in different domains in general and more specifically in cross-VM settings. A comprehensive study about RoP and domain isolation properties of hypervisors. Experiential insight about network and hypervisor architecture variations across different cross-VM attacks.

1.6 Thesis Outline

The rest of this thesis is structured as follows:

Chapter 2 covers relevant background information for this research and provides a detailed literature study on attacks in cloud model and specifically, discusses cross-VM attacks in cloud environments. Chapter 3 discusses about the network architecture and how network traffic flows in the cloud computing model. Chapter 4 describes the design of state-of-the- art impersonation and mirroring attack in the context of real-time attack under various cloud platforms and intensities. Evaluation of this attack also been proposed at the end. Chapter 5 describes the design of a novel real-time privilege escalation attacking by exploiting Return oriented programming (RoP) technique in conjunction with network-channel along with empirical evaluation of the RoP attack. Finally, Chapter 6 summarizes the research conducted and provides future work directions.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter provides the background of the work presented in this thesis and investigates potential security threats and countermeasure solution revealed in cloud systems. This is possible through discussion and evaluation of existing literature regarding cloud-based attacks and the vulnerabilities discovered by researchers, as well as novel methodologies and the designed architectures of prior works, which can help mitigate such attacks, and thereby help improve cloud security. Security threats have been described in detail, comparing the proposed methodology with state-of-the-art. The evaluation secure public, private, commercial cloud platforms, and prior research work, related to the proposed methodology are also presented.

The main goal of this chapter is twofold: to review the existing threat model and to assess the state-of-the-art attacks in cross-VM settings along with their countermeasures. To place this work in context, the chapter also offers a broader perspective on security issues in virtualization and their countermeasures.

Background Cloud computing, a virtualization platform, is an Internet-based computing model [18]. In this model, cloud providers deploy centralized computing resources and environments at large-scale, and provision these resources to customers. Customers can conveniently outsource their computation and data storage tasks to the cloud systems with great elasticity, economically and high energy efficiency. Cloud computing introduces new features including computing resources on demand, applications on demand, resource pooling andsssss broad network access[19] to minimize the running operational costs of cloud providers, and deliver high-quality services to customers. However, these features, in turn introduce new security threats to customers' computations and data. This dissertation designs new attack model and countermeasures

that detect and mitigate security vulnerabilities in cloud computing. In this chapter, background information and the attack vectors in the cloud systems are described.

2.1 Cloud Computing Definition

The National Institute of Standards and Technology (NIST) [19] defines the concept of cloud computing as follows: - Cloud computing is a model for enabling 'convenient', 'flexible', 'on-demand network access' to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

2.1.1 Essential Characteristics

According to NIST definition, cloud computing introduces several new key features as compared to traditional computing models:

- *On-demand resources:* Cloud providers facilitate customers to hire computing resources on demand, i.e when they need it. They introduce the concept of "pay-as-you-go" pricing model, which charges customers depending on the duration and amount of computing resources they use. This feature can significantly save customers upfront infrastructure costs, as they save a cost of purchasing and maintaining physical servers [19].

- *Broad network access:* Cloud computing is an Internet-based computing model that can be accessed over the network. Customers execute their computational tasks and store their sensitive data to remote datacenters, and are able to manage them remotely. This feature can provide great flexibility to customers as they can access the cloud services regardless of their locations and the devices. They can also share cloud services with others remotely [19].

- *Resource pooling:* Cloud providers' computing resources are shared to serve multiple customers using a multi-tenant environment, with different physical and virtual resources. These resources are dynamically assigned and unassigned to customers according to their demand. There is a concept of location independence in which the customers generally have no control or information about the exact location of the provided resources but may be able to identify location at a higher level of abstraction (e.g country, state or data center). Examples of resources include storage, processing, memory and network bandwidth [19].

- *Rapid elasticity:* Cloud providers offer scalable services to customers. At any time, customers can scale up their computing resources when their computing needs increase, or scale down by decreasing their computing needs [19].

- *Measured service:* Cloud providers adopt different measurements to monitor and measure the delivery of services. These measurements can price the cloud services in the "pay-as-you-go" manner. They can also achieve resource optimization and analytical planning: cloud providers can control automatic on-demand scaling and failure recovery based on these measurements. Resource usage can be examined, measured and informed, offering full transparency to the provider and customer [19].

2.1.2 Virtualization

Cloud computing structure are highly dependent upon virtualization technique. Virtualization facilitates multiple operating systems (different or same) to run on the same physical server at the same time [20]. Each Operating System (OS) experience the same level of abstraction for utilizing the entire machine, whereas they share the same server physically. Each OS executes within a Virtual Machine (VM). Virtualization is implemented by both software and hardware support.

2.1.2.1 Software support

In the software layer, virtualization is attained by a special software called the hypervisor or the Virtual Machine Monitor (VMM). This software contains variety of features [20]. First, it virtualizes the physical hardware resources (CPU cores, memory, I/O devices, etc.) so that multiple VMs can simultaneously execute on one physical server. Second, it offers strong isolation between different VMs so each VM has its own CPU context and memory space. Third, it is responsible to manage VMs' activities, e.g VM launch/termination, suspension/resumption, migration, etc. Virtualization are of two types [21]:

- **Full Virtualization** works at a higher privilege level than the OS. It facilitates guest operating systems running on guest VMs. The guest OS can release the similar privileged instructions and sensitive calls as those running on real hardware. These instructions are then interpreted to executable instructions by the hypervisor (Binary Translation virtualization), or such instructions are directly handled by the hardware (Hardware Assisted virtualization). The hypervisor entails two main features to facilitate and secure virtualized environments (i) protections of OS-Independent Storage Management to provide resources from unauthorized access and, ii) conversion of virtualized environments to allocate physical computing resources to virtual environments. Despite all these features, full virtualization provides numerous advantages that not only give benefits to the customers in terms of privacy and security, but cloud provider also get benefits by reducing the cost of infrastructure. These features include:
  - Sharing of hardware resources between multiple users;
  - Maintaining isolation between different users and from the control program;

  - Emulating new hardware to achieve improved reliability, security and productivity.

- **Para-Virtualization** is another type of virtualization technique in which the guest operating systems need to be amended to release special hypercalls to communicate directly with the hypervisor. This is called OS Assisted virtualization. The main restriction of paravirtualization is the fact that guest OS must be custom-made precisely to execute on top of the virtual machine monitor (VMM), the host program that permits a single computer to support multiple, similar configuration settings. However, paravirtualization eradicates the requirement for the virtual machine to setup privileged instructions. Setting up the instruction is a mean of managing unexpected or unallowable conditions, can be time-consuming and can badly effect the performance in systems that deploy full virtualization. Xen, a leading hypervisor, is an open-source software project that incorporates paravirtualization.

Hypervisor can further be classified into two categories: (1) a Type-1 or native hypervisor runs directly on the host's hardware to control a privileged host VM and other guest VMs. (2) A Type-2 hosted hypervisor runs inside an operating system (called host operating system). It manages the guest VMs from the host operating system. Figure 2.1 shows the abstract structure of these two types of hypervisors. The addition of new layer in type-2 hypervisor is highlighted in figure 2.1b.

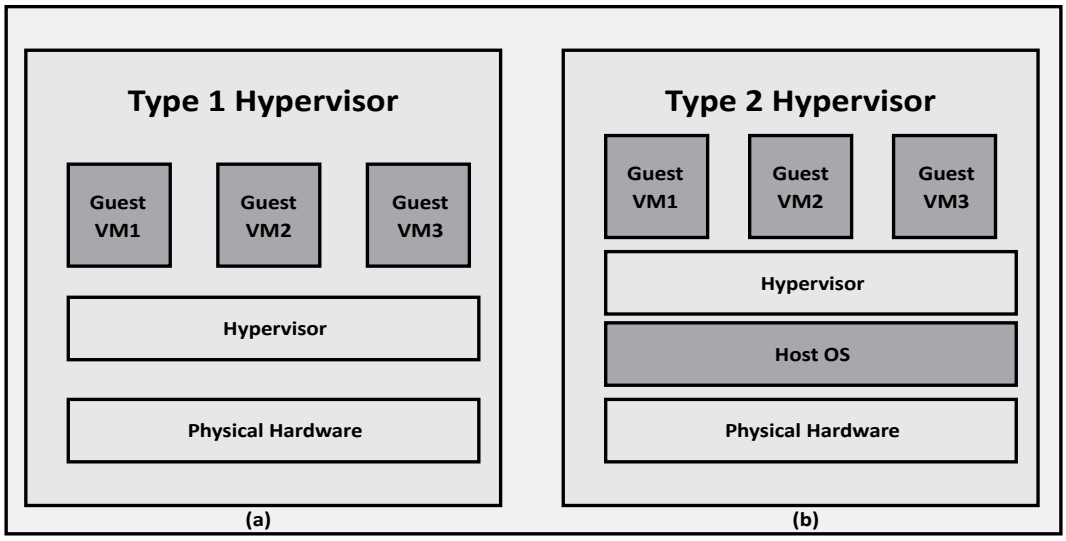

Figure 2.1: (a) Type 1 Hypervisor (b) Type 2 Hypervisor.

2.1.2.2 Hardware Support

Contrary to software, hardware is also aimed to facilitate virtualization and support the rapid growth of cloud computing. First, with the expansion of multi-core processors, the number of

processing units that can be integrated on the same server is increasing. This enhances the cloud server's ability to host more VMs simultaneously and improve resource utilization and power efficiency. Second, leading processor vendors such as Intel VT-x [22], AMD-V [23], integrate hardware virtualization extensions into their processors. These additions bring in new hardware components, instructions and execution modes for the handling of virtualization functions. For instance, Intel VT-x has several new functions [22, 23]. (i) VT-x enables two operation modes: VMX root operation, a fully privileged mode and proposed for the hypervisor; VMX non-root operation, which is not fully privileged and proposed for the guest VMs (ii) VT-x introduces a Virtual Machine Control Structure (VMCS) that is used for each VM to manage and store its non-root operations and VMX transitions (3) Intel VT-x uses the Extended Page-Table (EPT) hardware to support the virtualization of physical memory and (4) VT-x introduces new instructions that is particularly used to handle VMCS operations and memory management. AMD-V has also support similar functions. These additions in hardware can significantly enhance the virtualization efficiency and shows betterment in performance as compared to software solutions.

2.2 Cloud Computing Models

2.2.1 Deployment Models

Cloud computing offers different deployment models to provide cloud services [19]. These deployment models characterize different cloud environments and scenarios.

- ***Private cloud*** In a private cloud model, the cloud infrastructure is particularly designed for a single organization. Computing needs of such organizations have changed dynamically, or they demand direct control of the computation environment. Generally, the private cloud system is configured behind firewalls under its organizational control, so it only allows access to authorized users from the organization. Additionally, a private cloud can be configured locally to the organization itself, or externally by a third-party cloud provider.

- ***Community cloud*** A community cloud is normally shared between different organizations from a specific community having common interest (security, compliance, authority, etc.). The overall cost is equally distributed by these organizations, so this model can be more cost effective than private clouds. Analogous to private clouds, a community cloud can also be implemented and managed internally by the community, or externally by a third-party provider.

- ***Public cloud*** A public cloud offers its service for public use. The complete system is normally shared between different customers. Such cloud model offers services under the principle of "pay-as-you-go" model. Any customer with a valid credit card can pay to rent services. This cloud model doesn't give direct control to customers over the system. A public

cloud model is usually managed and operated by a commercial cloud provider such as Amazon Web Services (AWS), Microsoft Azure, Google Compute Engine, Rackspace, Oracle etc.

- ***Hybrid cloud*** A hybrid cloud offers the services of two or more clouds, delivering the fruitful features of multiple deployment models. It facilitates the customers to integrate, customize or aggregate its cloud services with other cloud model. For example, in a hybrid cloud customer can merge the services of a private and a public cloud. The organizational secret or confidential data can be stored in its private cloud and uses computing resources from the public cloud to run application that depends upon the data. This is because organizations do not want to store their sensitive data in public cloud model.

2.2.2 Service Models

Cloud computing is a model for facilitating ubiquitous, appropriate, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be swiftly set up and released with negligible administrative work or service provider collaboration [19] . This cloud model is based upon three service models. NIST defines these three standard service models as Software-as-a-Service (SaaS), Platform as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). Figure 2.2 illustrates the designs of these three service models at an abstract level. Components highlighted in grey are controlled by the cloud provider while white components are controlled by the customers as depicted in this figure. These services have been described separately.

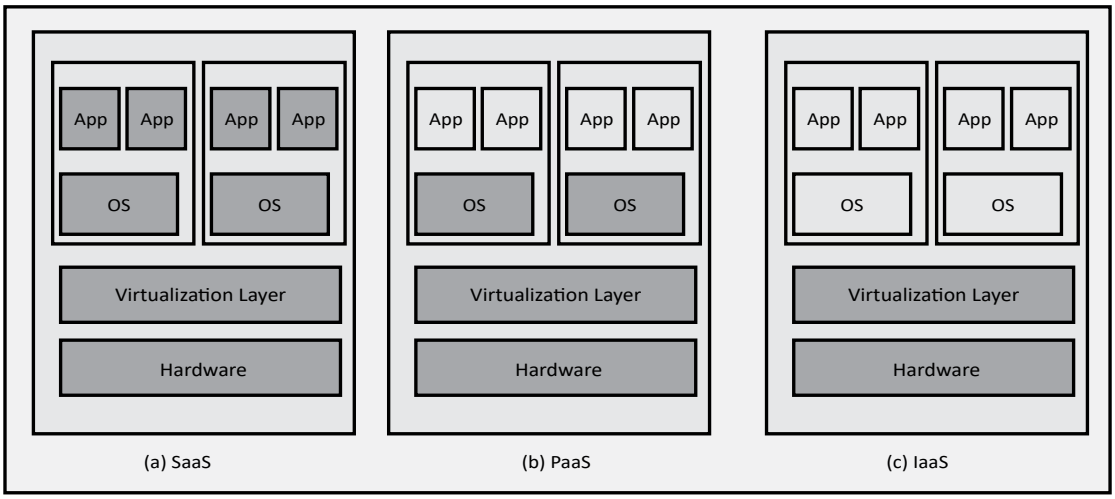

Figure 2.2: Cloud Services Model

- **Software-as-a-Service** In this service model, the cloud provider delivers applications software to customers as on-demand services. The cloud provider handles the infrastructure and platforms, configures different applications software on the cloud servers and provides customers network accesses to these software applications. Through this approach, cloud customers can have a right to access these applications directly through web browsers or program APIs. Customers are not required to configure or install applications on their own local computers. Well-known cloud applications include email, storage, social networks, etc. By the implementation of virtualization in cloud model, the cloud provider is able to run multiple copies of one application in different virtual machines to achieve scalability. In order to fully utilize the resources, cloud provider can execute multiple different applications on server [19].

- **Platform-as-a-Service** This infrastructure offers programming environments to customers for running their applications. The cloud provider implements and provides computing platforms and environments such as OSes, programming libraries and databases, to customers, as a service in this model. Then customers can directly design, develop and execute their own applications with in this platform, without the complexity of building and maintaining the underlying infrastructure such as servers and OSes. Virtualization in cloud model is responsible for providing context isolation and allocate resources based on the application's demands [19].

- **Infrastructure-as-a-Service** The cloud provider provisions an entire virtual machine (OSes, CPUs, memory, storage and networking) to customers, so they do not need to purchase physical servers or network devices. At the time of VMs launching, customers can select the required computing resources. These selections comprise of number of CPU cores, memory size, disk size, operating systems, etc. The cloud provider then allocates these resources from physical host servers and boots the VMs with these required configurations on the host servers. After the VMs are booted up, customers can remotely access them to configure and execute arbitrary software inside their VMs. Customer can fully control their VMs such as to suspend, resume or terminate their VMs at any time by sending the request to the cloud provider during the VMs' life cycle. Other than this, cloud provider can also migrate customers VMs to other servers for energy optimization or fault tolerance. Currently, the cloud provider usually adopts the live migration approach. This approach can migrate the VMs without disturbing their execution. Migrated VM downtime in live migration approach is near to zero and experience negligible performance overhead. Additionally, customers are totally unaware about the migration of their VMs [19].

2.3 Cloud Computing Security

Cloud computing security denotes to a broad set of strategies, technologies, and controls organized to protect data, applications, and the associated infrastructure of cloud computing [24].It is the main concern of enterprises when shifting its critical information to geographically distributed cloud platforms and these platforms are directly not under the control of that organization. Additionally, traditional IT information system security procedures, security configurations, firewall rules can help in reducing the cloud attack surface. The main security principles that protect information assurance are confidentiality, integrity, availability, authentication, authorization, auditing, and accountability. These concepts are summarized in the following sections. Confidentiality, integrity, and availability are referred as the CIA triangle of information system security as shown in Figure 2.3. They are strong pillars of cloud security assurance [25].

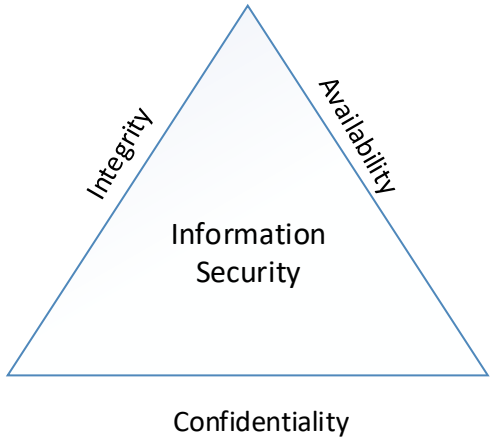

Figure 2.3: CIA Triangle

2.3.1 Confidentiality

Confidentiality refers to the inhibition of intentional or unintentional unauthorized exposure of information or data. Confidentiality in cloud systems is linked to the domains of covert channels, traffic analysis, encryption, and inference:

- **Covert channels** A covert channel is an unauthorized and unintended communication path that allows the transmission of information. Covert channels can be accomplished through timing of messages or inappropriate use of storage mechanisms [26].
- **Traffic analysis** Traffic analysis is another domain of confidentiality breach that can be accomplished by analyzing the traffic patterns like volume, rate, source, and destination of message traffic, irrespective of the fact whether it is encrypted or is in plain text. A huge

traffic burst and an increased message activity can indicate the occurrence of a major event. Mitigation strategy of traffic analysis contains maintaining approximately constant rate of traffic and hiding the source and destination location of the traffic [27], [28].

- **Encryption** It involves in hiding original messages into some dummy message so that they can- not be disclosed by an unauthorized user, even if they are captured. The effort need to decrypt the message is totally dependent upon the function of the strength of the encryption key and the robustness and quality of the encryption algorithm [27].
- **Inference** Inference is the ability of an entity to use and correlate information secured at one level of security to disclose information that is secure at a higher security level.Inference is usually connected with database security [27]..

2.3.2 Integrity

Integrity contains maintaining the reliability, accuracy, and dependability of data over its entire life cycle . Data must not be transformed in transit, and pre-cautionary measures must be adopted to ensure that data cannot be transformed by unauthorized people. These measures include file permissions and user access controls. In addition to these measures, some means must be in place to detect any changes in data that might occur as a result of non-human-caused events such as server crash. Some data might include checksums, even cryptographic checksums, for verification of integrity. Backups or redundancies must be available to restore the affected data to its correct state. [28].

The model of cloud information integrity demands the necessity of the following principles:

- Data cannot be modified by unauthorized users or processes.
- The data must be consistent internally and externally.

2.3.3 Availability

Availability is maintaining the cloud data center or hardware resources, performing hardware repairs immediately when needed and maintaining a correctly functioning operating system environment. Providing suitable communication bandwidth and preventing the occurrence of bottlenecks. Redundancy, fail over, RAID even high-availability clusters can lessen serious significances when hardware concerns occur. Fast and adaptive disaster recovery is essential for the worst case scenarios. Defenses against data loss or interruptions in connections must contain random proceedings such as natural disasters and fire. To avoid data loss from such incidences, a backup copy may be saved in a geographically-isolated location. Security software such as firewalls and proxy servers can save against the downtime and unreachable data due to malicious actions such as denial-of-service (DoS) attacks and network intrusions.Additionally,

this theory assure that the security rules or models of the cloud system must be in working order [28].

2.3.4 Cloud Security Services

Additional features that directly relate to cloud system assurance include authentication, authorization, auditing, and accountability, as described in the following sections.

2.3.4.1 Authentication

Authentication is analysis or reconciliation for the confirmation of a user's identity. It creates the user's identity and confirms that users are who they claim to be. For example, a user gives an identity (user ID) to a computer login screen and then has to entered a password. The computer system validates the user by authenticating that the password relates to the individual offering the ID [19].

2.3.4.2 Authorization

Authorization refers to rights and privileges approved for an individual or process that enable access to computer resources and information. Once a user's identity and authentication are recognized, authorization levels determine the degree of system rights a user can avail [19].

2.3.4.3 Auditing

Organizations adopts two basic methods in order to maintain operational assurance: system audits and monitoring. These methods can be applied by the cloud customer, the cloud provider, or both, depending on cloud architecture and deployment. A system audit is a one-time or intermittent event to appraise security. However, monitoring relates to an ongoing activity that observes either the system or the users, such as intrusion detection [19].

2.3.4.4 Accountability

Accountability is the procedure to define the activities and performances of a single individual within a cloud system and to recognize that particular individual. Audit tracks and logs provision accountability can be used to conduct review in order to analyze historical proceedings and the individuals or processes associated with those proceedings. Accountability is related to the concept of non repudiation, wherein an individual cannot successfully deny the performance of an action [19].

2.4 Attack Vectors in Cloud Computing

As discussed cloud computing provides new characteristics to facilitate enterprises and individuals to deploy IT infrastructure. However, these new characteristics in turn strengthen already existing vulnerabilities and introduce new vulnerabilities. The Cloud Security Alliance (CSA) has recognized several well-known attacks that can compromise customers' computations and data in clouds [29]. These attacks have been categorized into different classification based on the attack vectors. Figure 2.4 illustrates the abstract architecture of a cloud system with the possible attack vectors. Here "x" indicates the possible attack vectors which are discussed as follows:

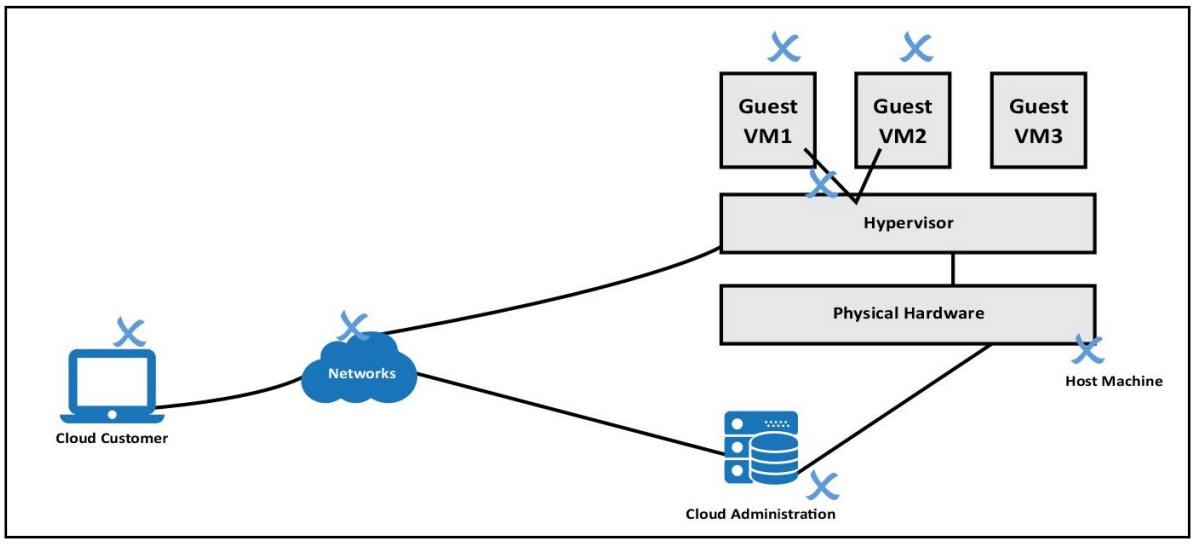

Figure 2.4: Potential Cloud Attack Vectors

2.4.1 Potential Attack Vectors

- **Service interface:** To access the services in cloud computing, a customer first requires to register an account on the official website of the cloud provider. The customers need to login into their account to use the cloud services. At this point, an attacker can penetrate itself in a cloud system. As some cloud systems have not strong user identification and authentication process, this gives a strong possibility to an attacker to hijack customers' account and access or compromise the sensitive or private areas of cloud services. Secondly, customers interact with the cloud services by using some User Interfaces (UI) or Application Program Interfaces (APIs). If these interfaces are not configured properly or securely, an attacker can easily exploit the vulnerabilities to hack credentials or compromise the cloud services [30].

- **Networks:** Cloud computing allows customers and end-users to access its services through networks. If networks are not properly secured, attackers can steal sensitive data during

transmission. Additionally, attackers can launch different attacks on networks e.g, Denial-of-Service (DoS) that fully utilize the cloud system resources. As a result, such attacks prevent customers or end-users from accessing the data or applications [30].

- **Cloud Administrators:** The cloud administrators have been granted an admin role, a privileged role, that supports management of the cloud infrastructure and services. Non reliable or compromised cloud administrators can be the main source of high security risks to the customers in the following ways. First, an unfortunate mishap or accident can occur in a cloud system that leads to permanent data loss or leak of customers' sensitive data. Such mishap or accidents compromise of accidental data deletion by cloud administrators, or a physical disaster such as a fire in cloud data center or an earthquake. Second, an insider regardless have full access of the data stored in the cloud. He can easily misuse or exploit the sensitive information of customers stored in the clouds, or compromise the key management of cloud [30].

2.4.2 Virtualized System

Attackers can exploit system vulnerabilities within the cloud servers or virtual machines to interfere in the servers or VMs to leak data or to take control of the entire server system or to disrupt service operations. The attack possibility increases with the addition of hypervisor. The hypervisor is a software that has a large piece of code and unavoidably consists of several security bugs [31]. Attacker can exploit these bugs to compromise the server. Additionally, VMs load the operating system which is called as a guest OS. Each guest operating system has its own vulnerabilities that put the security of customers' data and services at high risk.

2.4.3 Shared Infrastructure

The cloud provider can run multiple tenants or Virtual machines (VMs) on the same physical server, that can share the same underlying physical resources. This sharing of hardware resources and components may lead to new cross-VM attacks. In this shared infrastructure, a malicious VM can launch some attack to ex-filtrate or leak sensitive data of other co-located VMs. It can also exploit shared resources by launching DoS attack on host machines or main server so that no other co-located VMs can access the physical resources.

2.4.4 Cloud Services

Cloud computing offers adaptable and low-cost cloud services. However, malicious customers can exploit these services to launch attacks on cloud servers. These customers can first hire a large amount of cloud resources through free cloud service trials or credit card fraud. Customers can exploit the cloud resources to attack other customers, organizations or the cloud provider. Some well-known attacks they launch include Distributed DoS attacks, large-scale email spam and

phishing attacks, brute-force password guessing attacks, port scanning attacks, etc.

Despite the fine-grained services and strict configuration of security models offered by the cloud providers, there is still potential for attackers to attack cloud computing model by exploiting shared memory, cache, network channel, covert channel and other possible channels. In the following section, the well-known attacks on cloud model and their countermeasures solutions have been discussed.

2.5 Vulnerabilities in Cloud Computing

Traditional security threats such as the vulnerabilities in networks and the related operating system attacks encountered in local networks and systems, are also applicable to the domain of cloud computing. Cloud providers introduce new, advanced features in cloud computing, which can in turn lead to new security threats to both cloud providers and consumers. Firstly, consumers should trust cloud providers when using computational resources and storage devices. Security perimeters, resource allocations and the management of cloud services are not handled or managed by consumers, in fact these are strictly controlled and monitored by cloud providers. Consumers are highly dependent upon cloud providers for securing their sensitive data and computations. Secondly, cloud providers usually facilitate multi-tenancy infrastructure, which helps reduce its cost and maximize resources usage. Due to this feature, multiple VMs belonging to different untrusted consumers can be located on the same physical machine. This facility can effectively enhance whole system's resource utilization and can likewise reduce operational costs. However, this feature can also bring new vulnerabilities to a cloud system, due to sharing of the same hardware resources and storage devices. Thirdly, the cloud providers use virtualization technique to manage resources more efficiently. This extra software layer can make systems more complicated, and can add new attack vectors. The aim of this chapter is to analyse the existing cloud vulnerabilities, and also to discuss countermeasures proposed in prior work for mitigating these cloud-based vulnerabilities. In general, the classification of these vulnerabilities and countermeasures are based on attack vectors.

2.5.1 Vulnerabilities in Virtualization

The infrastructure of cloud computing runs through the concept of virtualization, in which a single physical system is assigned to multiple users at the same time. In such situations, there are possibilities of exfiltration of data [32]. The introduction of a new layer in virtualization may create a single point of entry for attackers, if virtual machine monitor is compromised.

Three types of vulnerabilities on virtualization can exist which are as follows:

OS level virtualization Multiple guest OSs run on a host OS, that has the control and visibility of each guest OS. Within this type of configuration, by compromising the host OS, an attacker can obtain control of the entire guest operating systems running on the host OS [33].

Application-based virtualization Virtualization is layered above host OS. In this type of virtualization, each VM has its guest OS that is running different applications. Application-based virtualization also suffers from the same type of vulnerability as OS-based vulnerabilities [33].

Hypervisor or Virtual Machine Monitor (VMM) Hypervisor is a piece of code embedded in host OS. Such a code may contain native errors. This code runs at a boot time of the host OS, to control multiple guest OSs. If the attacker is successful in compromising the hypervisor, then the entire controlled guest OSs can be compromised [33]. Well-known attacks on hypervisor are given below.

In [33, 34] VM escape techniques were defined as being where an attacker creates a program that executes a VM, whose purpose is to access the hypervisors' root privileges by breaking the isolation layer. Using VM escape, an attacker can access the host OS, bypassing the hypervisor layer and other VMs running on the same physical machine. Virtual machine sprawl is another challenge for cloud organizations. With virtual machine sprawl, the number of virtual machines runnng within a virtualized environment increases due to the creation of new virtual machines that are not necessary for business. Due to this, the new virtual machine will misuse the cloud infrastructure [35].

Virtual machine can run on cloud computing which can be accessed through the Internet. This indicates that their theft can take place remotely. Most hypervisors can store contents of the virtual disk for each VM as a file, which allows VMs to be copied and run from other physical machines. While this is a convenient feature, it is also a security threat. Attackers can copy the VM over the network, or to a portable storage media, and then access data on their own machine without physically stealing a hard drive [36]. Once attackers have direct access to the virtual disk, they then have an unlimited time to defeat all security mechanisms, such as passwords, by using offline attacks. The second security breach of the virtual disks discussed in [36], is how attackers could corrupt or externally-modify a file while the VM is offline. This means the integrity of an offline VM may be compromised if the host is not securely protected [37].

The hypervisor manages the resource allocation between the host and guests' machines. The ultimate goal of the attacker is to compromise the hypervisor, in order to access the host OS with the same privileges as that of the hypervisor [38].

2.5.2 Rootkit

Rootkit is a software or application-level tool that enables an unauthorized user to gain control of a computer system without being detected. In Virtualization, cross virtual machines can penetrate rootkits to other virtual machines, can crash hardware, or even can access sensitive information. Rootkits have multiple capabilities, they are able to not only hide malware, but can also conceal malware from analysis and detection processes utilized by defenders [39].

2.5.2.1 Rootkit in Hypervisor

The new guest OS in virtualization assumes it is running at the host OS, with the corresponding control over hardware and resources. However, in reality there is no concept of the host's existence. A compromised hypervisor can also be used to create a covert channel for executing unauthorized code into the system. Through this approach, an attacker is able to control any VM running on the host machine, and can consequently manipulate system activities [33].

2.5.2.2 Hijacking the Hypervisor

If the rootkit can insert itself beneath the guest operating systems, it can control the entire system [40]. This is exactly what rootkits achieve through different modes of x86 modern architecture, as explained below. Figure 2.5 shows the x86 modern architecture, along with root-kit privileges level.

- **The User-Mode Rootkit** resides in Ring 3, along with some other applications as user, rather than low-level system processes. These can help in achieving objectives by replacing a system's binary applications, or by over-writing a Dynamically-Linked Library (DLL) [39]. **DLL Hooking and Injection** User-mode rootkits can exploit an API hooking by using DLL hooks. A well-known user mode rootkit is the Vanquish rootkit [41], which redirects Windows API calls to hide files, folders and registry entries [42]. This is accomplished by injecting a malicious DLL into a target process, acting as an intermediary for API calls to intercept requests for files, folders or registry entries, to filter them [39, 43].

- **Kernel-Mode Rootkit** These rootkits reside in Ring 0 which has the highest operating system privileges level by adding some code or modifying the portions of the core operating system, including both the kernel and associated device drivers.

  Kernel-mode rootkits use a variety of techniques for subverting the OS kernel and AV solutions.

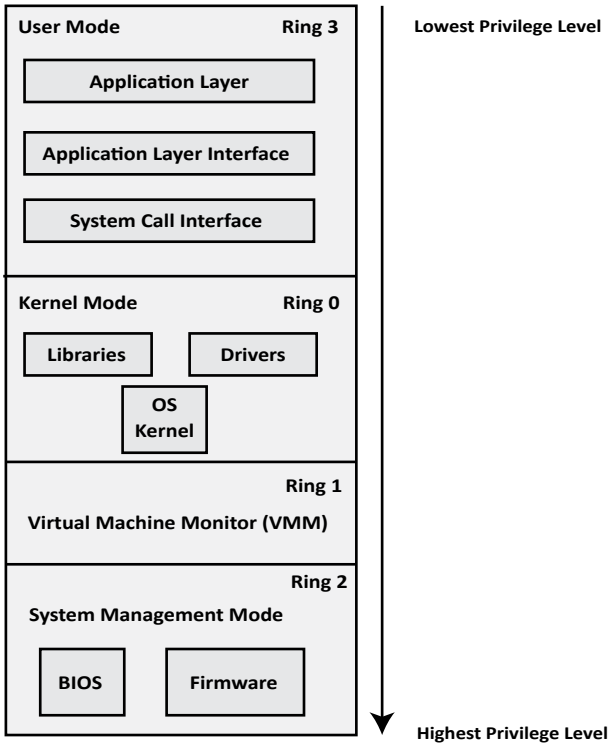

Figure 2.5: Privileges Level of Root-kit

Hooking Tables In modern operating systems there are a number of different tables which can be used to perform lookups for different purposes, whether they be upon receiving an interrupt, or knowing where a given system calls (syscall) is located. Altering the memory descriptor tables offers an effective means of changing the memory mappings within a target OS [39].

Routine Detours Routine detouring can overwrite a code segment with jump sequences, either at the beginning (pro-log detouring) or at the end (epi-log detouring) of the target routine [43]. This helps maintain the original size of the altered routine, but not the checksum. The overwritten sequence has been replicated at the appropriate location in the malicious code segment, in order to ensure that the routine operates as expected [39].

Binary patching Binary patching can directly replace the binary representation of target routine, often replacing system drivers or files altogether. This is often exploited by manipulating the boot process, such as the Master Boot Record (MBR) and the Basic Input/output System (BIOS), prior to OS initialization. One such example is the Vbootkit, which uses the technique to create a custom boot sector code, in order to subvert the Windows security mechanisms [39], [44].

Virtual Memory Subversion This is another additional form of rootkit, which demon-

strates virtual memory subversion. The rootkit is called ShadowWalker, which is capable of hooking and subverting virtual memory protections. ShadowWalker, when executed, redirects read/write in order to hide executable code [45]. When a read/write attempt is made on the hidden code region, the returned frame is diverted to an untainted one, while allowing the normal code execution of the hidden code to continue. This is accomplished by exploiting the split Translation Look, which is a side Buffer (TLB) of x86 architecture, and de-synchronizing the two components in order to effectively filter executable code regions from the read/write accesses [39, 46].

Virtual Machine Based Rootkits Virtual Machine Based Rootkits (VMBR) operate in what is referred to as the Ring -1 layer, as its primary purpose is to allow the guest OS to run with Ring 0 privileges, without affecting other guest OSs at that privilege level.

VMM Subversion Rutkowska [47] was the first to use this technology to subvert the OS one step further. Here it inserted itself beneath the target OS, by utilizing these processor extensions and virtualizing the target OS in order to control all access to hardware peripherals, context switching, memory management operations, and other components as well [48]. These are widely referred to as Virtual Machine Monitor (VMM) or Hardware Virtualization Machine (HVM)-based rootkits [49].

- **System Management Mode Rootkits** These rootkits (SMMR) normally reside in Ring2. The purpose of their development is to support low-level strategy that is developed through the following methods:

  BIOS and Firmware Rootkits This rootkit infected the Basic Input/output System (BIOS) and PCI device firmware, at the lowest layer of computer systems [39].

  BIOS and Bootstrap Modification This attack operates in real-address mode, acting as a step towards subverting the OS before it has even been initialized [39].

  Firmware Manipulation This attack allows for the remote injection and execution of malicious code within the memory region, allowing Direct Memory Access (DMA) into the target OS [50]. Performing DMA using this method allows for the exploration or alteration of the target OS [39].

  Hardware Interface These devices can be used to interact with a system through unintended hardware vectors. One such example is a project at RMC, which had a primary focus on exploiting unintended USB channels, in order to create two-way communications with a target system [51]. This work used two different unintended channels for ex-filtrated data, including a keyboard LED channel which used a combination of the Scroll Lock, Caps Lock and Num Lock, as well as an audio channel which uses waveform files to communicate data to the target OS [52].

- **Hypervisor Modification** The security of a original hypervisor does not make any sense if it is remotely controlled or modified by an external attacker. The new generation rootkits allow an attacker to add an additional layer of hypervisor between the hardware and software. The hypervisor takes control of the system and converts the original operating system into a virtual guest on the fly. Comparaed to the software-based virtualization, this type of hijacking does not require any restart, and it makes intrusion detection impossible [33],[37], [53].

Table 2.1: Overview of Root-Kit Classification

Rootkit Mode	**Technique**	**Process**	**Placement/System**
User-Mode Rootkit [39]	DLL Hooking and Injection [42]	Exploits API Hooksing, through using the DLL hook	Vanquish Rootkit [39], [43]
Kernel Mode Rootkit	Hooking Table [39]	Alters the memory descriptor table	
	Routine Detours [43].	Overwrites a code segment with a jump sequence	BIOS/Memory Boot Record *MBR*. e.g, the Vbootkit Shadow walker [45]
	Binary Patching [39], [44]	Exploits the boot process	
	Redirects read/write to hide code [39], [46].		
	Virtual memory subversion [48]	Inserts itself beneath the target OS to control all the access to hardware peripherals	
System Management Mode Rootkit	BIOS and Firmware [39].	Infects the BIOS and firmware.	
	BIOS and Bootstrap modification [39].	Operates in a real address mode	Caps, NUM locks and Audio channel to communicate data with the target OS, for instance with the RMC [52]
	Firmware Manipulation [39]	Allows for the remote injection and execution of malicious code in the memory region	
	Hardware Interface [51], [39]	Interacts with the system via an unintended hardware vector	

2.5.3 VM Monitoring from the Host

In virtual environment, host machine can be considered as a system's main hub. There are some significances that allow host machines to observe and communicate with the running of VM applications. Therefore, the host machine's security is absolutely necessary. Different virtualization technologies support implications for the host machine to interfere with VMs operating in the system. The following are the potential approaches for the host to affect VMs:

- The host can perform the basic operations of VMs, such as starts, shutdowns, pauses and restarts.
- The host can manage to observe and change resources allocated to the virtual machines.
- The host has sufficient rights to observe programs executing inside the VMs.
- The VM's data is eventually stored on the virtual disk assigned to it, which the host machine can access. Due to this, the host can undertake basic operations, such as copying, pasting, editing, or simply viewing the content on this data.

More importantly, all communication to and from the VMs passes through the host, which again allows the host to observe all communications of its VMs. In such cases, if the host is compromised by some means, then the security of all VMs running on it is at high risk.

2.6 Related Work

2.6.1 Cross-Channel Attack in Cloud:

The aforementioned attacks are general threats to a cloud model. Following are well-known Cross-Channel attacks in a cloud model that are close to our research.

2.6.1.1 Side Channel Attacks

Side-channel attacks are a class of physical attacks in which an adversary tries to exfiltrate the sensitive information of other virtual machines. The use of virtualization can introduce new security vulnerabilities, such as using cross VM-Side channel attacks, to extract information from a target machine [54]. Some of the most well-known side channel attacks have been discussed as follows.

The most effective cross VM-attack is an access-driven attack that exploits shared micro-architectural modules like caches. In it, the attacker executes a program of code on the system, which performs cryptographic operations. This program executed by attacker monitors the use of cache to learn information about the key [55].

A time-driven side channel attack is possible when the total execution times of the cryptographic operations with a fixed key, is influenced by the value of key. The influence is exploited by an attacker who can calculate such timing, in order to statistically gather information about the key [56] [57].

A trace-driven attack is used to capture a profile of the cache activity. This states that the attacker can obtain access to a running profile in which cache activity is monitored, and then process it in order to extract the actual activity from the other profile content [56] [57].

2.6.1.2 Covert Channel

Covert channels are virtual communication channels between entities, which bypass the rules of communication between them. Within the virtualization context, these give the attackers new opportunities for communication, without being noticed by the VMM (Virtual Machine Monitor) security module [58]. Therefore, the evolution of security threats arises within the context of virtualization, and this need to be eliminated in order to obtain the full advantage of using this technology.

In [59], researchers demonstrated a covert channel between the virtual machines on the Xen hypervisor [60]. This was based on the fact that table that maps the machine address frames to the pseudo physical frames of the virtual machine can be read by any guest VM.

Rutkowska [61] implemented a method of TCP/IP steganography called NUSHU, leaking sensitive data from a compromised system through network packets generated from it. Murdoch and Lewis [62] have addressed the various possibilities of covert channels, using TCP/IP header steganography. It has been observed that this may be applicable within the virtualized scenario.

Murdoch and Lewis [62] described the header fields that make room for steganography and developed the novel 'Lathra' method for a covert channel, using TCP ISN (an initial sequence number). It also stated that an external warden, being a program or entity which can watch and analyze data transfer between two systems or programs, cannot distinguish the ISN generated by a machine from a manipulated TCP header. The recovery of encoded message is only possible with the key used for generating ISN. This covert channel can be implemented between VMs, and cannot be identified by the VMM [59].

It was hypothesized in [59] that a timing covert channel, which can be used to communicate covertly between VMs and an attacker, can be used to leak information from possible VMs. To send information covertly from a VM, TCP packets need to be sent at different time intervals. If a TCP packet is sent at a specific time interval, the receiver then recovers message as bit 1 else bit 0 [63].

A new network-based covert channel has been identified, using two sockets to transfer data covertly. This can be used by an attacker, in order to leak information from a VM [59].

Indexes may contain a great amount of information regarding the data itself. They are developed when the service provider receives data from the users, and decides to build indexes in order to improve search performance. The users have no idea that their data is being used in this way, which probably leads to the leaking of much more information. The index maps each term to a set of documents which contains the term and is then queried by the searcher in order to obtain a list of matching documents. The index host must then apply these policies for each searcher, in order to filter search results appropriately. Since only the indexing host needs to be contacted in order to completely execute a search, searches can be highly efficient. The compromising of index host by hackers could lead to a complete and devastating privacy loss [64].

2.6.1.3 Attacks on Images

An attacker can attack OS images, in order to obtain sensitive information from other guest OSs, and to crash the OS. Even if VMs are inactive, attackers can still manage to access them. This is because the backend data center is permanently activated, and an offline VM is not considered to be the powered-off home computer. Secondly, pre-built images need to be carefully scanned, in order to circumvent legacy-vulnerable applications or trapdoors. As an example, AWS pre-built images store builders SSH keys internally, meaning that all hosts using such types of images are accessible by publishers [34].

As a common practice in cloud computing, cloud providers can create a template image of an Operating System (OS), and then clone it to multiple machines. If there is some vulnerable VM template images, then it may spread over many systems. An attacker can rent one of these VMs and can accordingly analyze all important configurations, including administrative rights. Another key issue they have raised is that an image can even be taken from an untrustworthy source, which may provide back-door access to an attacker [65].

In [66], it was stated that the Guest VM can crash the Host OS, along with other guest VMs hosted by it. Modi et. al [33] detailed that sharing VM images in the cloud introduces security risks. The main concern of an image's owner is confidentiality, for instance the possibility of unauthorized accesses to the image. An image's user is concerned about safety, such as the potential of a malicious image corrupting or stealing the users' own private data. For example, instances running on Amazon's EC2 platform can be easily compromised through the performance of various attacks, such as signature-wrapping attacks, cross site scripting (XSS) attacks, and DoS attacks. Through such types of attack, attackers can create, modify and delete VM images,

and can change administrative passwords and settings that are put into instances with EC2 for S3 access. Images of pre-configured virtual applications and machines may be tempered or misconfigured, before being uploaded [67].

2.6.1.4 Memory Attacks

An attacker can ex-filtrate data by attacking physical hardware, such as memory, storage units, and others. Different cases have been described in this section, showing how successfully attackers can engage different memory regions.

Once an attacker manage to co-reside on the same physical machine with a target, the next step is to exploit the hardware resources, and extract sensitive data through cross VM-attacks. It exploits hardware by using a technique for encoding information, thereby accessing the latencies of a shared L2 cache [56], [68].

The hostile VM first writes continuous blocks of memory, and then releases those blocks. Later the attacker will release those blocks, and the host VM will overwrite those blocks with its own instruction set. The way the target machine writes to those memory blocks, after the attacker has released them, is an operational characteristic performed of the target. The attacking VM then tries to read back the same instruction set, checks for missing blocks of memory, and tries to replicate the possible instruction set [69].

An event channel is just like a signal channel used to inform communication parties about the occurrence of a new event. The writer tells reader about the data it just wrote. Then, once a reader finishes reading it, it deletes the data and tell the writer that a new space is ready for the next input through the event channel. Therefore, the event channel must be well protected, otherwise it will mess up the whole communication. Delivering the wrong information may cause the shared memory channel to go out-of-sync [34].

The grant table provides two types of grants between different VMs. One type is page-flipping, while the other is page-sharing. Per-packet page-flipping has too much performance overhead, due to the high frequency of hypercalls. Therefore, the new communications have dropped page-flipping but have preserved a page-sharing grant. The Xen memory sharing mechanism is at a page granularity level. Shared pages are identified by an integer, which is known as a grant reference. The hypervisor keeps the grants information, passes the grant reference to the communication VMs and signals it via the event channel. The hypervisor will be the authority for authenticating communication parties. Under certain circumstances, the system may delegate communication parties to manage the grant table by themselves, for performance reasons [34, 70].

Ranjuth [59] has stated that if a VM is migrated to a new host, then the memory used by that VM will be recovered by the VMM. If a new VM is started on that VMM, there is the possibility that the memory used by the old VM will be allocated to the new one. This new VM may be an attacker. Therefore, the attacker can search all the memory pages for some specific information such as passwords, session keys, and other aspects about the first VM.

When a VM is destroyed or shut down, the information from the virtual machine can then be leaked. After the destruction or shut down of a virtual machine, its memory can be allocated to a new virtual machine which runs on the same VMM [59].

2.6.1.5 Row Hammer Attacks

Recent DRAM chips have huge capacity and a high density of memory cells. Therefore, a memory cell can suffer from disturbance errors due to electrical interference from neighboring cells. Especially when an adversary quickly and frequently accesses the DRAM with precise patterns, some data bits in the memory area, where the adversary has no access rights, can be flipped due to electrical interactions. Such DRAM hardware vulnerability is called a row hammer attack.

Xiao et al. [71] abused this memory loophole in order to attack a para-virtualized platform from a guest VM. In this attack, an adversary VM kept accessing selected data in the DRAM, in order to flip critical bits of this VM's page table entry. By doing so, the page table entry points to a fake page table, without being observed and checked by the hypervisor. The fake page table interprets the VM's virtual page to a physical page that does not have a connection with this VM. Consequently, the attacker VM can steal or tamper with the sensitive data of the co-located VMs.

2.6.1.6 Code Reuse

Code reuse exploits rely on code fragments (gadgets) located at predetermined memory addresses [23, 36, 39]. Code diversification and randomization techniques (colloquially known as fine-grained ASLR [105]) can thwart code-reuse attacks by perturbing executable code at the function [13, 64], basic block [38, 118], or instruction [57, 87] level, so that the exact location of gadgets becomes unpredictable [72].

Snow et al. introduced "just-in-time" ROP (JIT- ROP) [105], a technique for bypassing fine-grained ASLR for applications with embedded scripting support. JIT-ROP is a staged attack: first, the attacker abuses a memory disclo- sure vulnerability to recursively read and disassemble code pages, effectively negating the properties of fine-grained ASLR (i.e., the exact code layout becomes known to the attacker); next, the ROP payload is constructed on-the-fly using gadgets collected during the first step.

2.6.1.7 Denial of Service

The Denial of Service attack (DoS) is a serious threat. Both the privileged host and the normal guest OS are under threat from this type of attack, due to poor authentication with current communication mechanisms [33, 34, 72]. DoS attacks make other hosts unable to perform actions in a timely way. Hardware sharing can also be exploited in order to conduct host-based DoS attacks. The adversary VM can generate contention regarding different types of shared resources, in order to degrade the victim VM's performance, or to increase its own performance.

The first affected resource of focus is the CPU. Grunwald and Ghiasi [73] mentioned that it is possible to flush the shared processor pipeline, but this degrades the victim's performance. They achieved this by implementing de-normalized floating point values, thereby creating an underflow so that the pipeline has to be flushed to handle the exceptional condition. Zhou et al. [74] exhibited a CPU resource attack, where an attacker VM can abuse the boost mechanism within the Xen credit scheduler, in order to increase its scheduling priority and to acquire more CPU resources than are paid for.

The second applied case is the memory system. Varadarajan et al. [75] anticipated the resource-freeing attack, where an attacker deliberately increases the victim VM's use of one form of resource, such as network I/O, to force it to release other types of resources, such as CPU caches. This allows a co-located VM controlled by the attacker, to use more of the latter resources. Grunwald and Ghiasi [73] considered the effect of trace cache expulsions on the victim's execution, with Hyper-Threading enabled through an Intel Pentium 4 Xeon processor, and indicated that a malicious thread can slow down a victim's performance by a factor of 10 to 20. Woo and Lee [76] discovered that frequently flushing shared L2 caches on multi-core platforms could slow down a victim's processing speed. They studied saturation and the locking of buses that connect L1/L2 caches and main memory [76]. Moscibroda and Mutlu [77] studied contention attacks on the schedulers of memory controllers.

The third category of DoS is that of I/O resources and networking. I/O resource contention can also be responsible for degradation of the victim's performance. For network resources, Bedi et al. [78] proposed a network-initiated DoS attack where the attacker VM causes contention within the Network Interface controller, in order to degrade the victim's working. For disk resources, Yang et al. [79] proposed a method of reverse-engineering the I/O scheduling within the virtualization platform, assisting the attacker to design specific Denial-of-Service attacks on the disk I/O resources. Chiang et al. [80] designed a more efficient adaptive attack, classifying the victim's I/O usage pattern, and synchronizing the attack phase with the victim. Huang and Lee [81] proposed cascading performance attacks, in which an attacker VM consumes the I/O processing capabilities of the Xen Dom0, and thereby degrades the victim VM's performance. Similarly, Alarifi

and Wolthusen [82] exploited VM migration in order to reduce Dom0's capability in I/O processing.

The last considered case in DoS is its impact on power. An attacker can reduce the host server's power utilization, in order to break down the victim's cloud services, or even the whole server. Xu et al. [83] designed an attack, using power over-subscription methods, this damages the datacenter. The technique creates instances on the host server and executes power-hungry programs which enhance server's power capacity to its peak. It therefore affects overall power consumption by exceeding its levels. As a result, the power unit fails and the server shuts down.

2.7 Cross-VM attack

2.7.1 Co-location Attack

In an IaaS system, VM co-location attack threatens victim VM's position, for instance its host server, which can be recognized by the attacker. At that point the attacker launches their VM on the same host server as the victim VM, whereby they share the same hardware resources I/O and network devices. Co-location attacks are prerequisite for other shared infrastructure attacks [72, 84].

Researchers have already presented a number of ways for achieving co-residency. 'Flooder' is one approach to achieving co-residency [85]. To begin the search for his target machine, the attacker creates a large number of instances on the cloud, which are referred to as flooder. Each flooder publicizes its existence to a master host, the client is an attacker outside the cloud. A series of signals are then sent to each flooder. According to these signals, the flooder then injects a network activity into the outbound interface of its physical host machine. Through this, the client can then test for colocation [85].

Ristenpart et al. [56] first exhibited a network-based methodology for launching colocation attacks in a public cloud, e.g., through Amazon EC2. An attacker can launch many unnecessary VM instances in the same availability region as the victim VM, and follow different ways to examine if their VMs are successful in co-locating with the victim VM or not. Some of the attacker's methods for examination include, executing the TCP SYN traceroute for detecting the network traffic's first hop (which is Dom0 in the host Xen server) of this VM and the victim VM. An identical Dom0 IP address between his VM and the victim VM will specify the co-location. Additionally, the attacker can calculate the network packet round-trip time between his VM and the victim VM. A smaller value indicates that two VMs share the same machine. Then, the attacker can check the internal IP addresses of his VM and that of his victim VM. Numerically-close internal IP addresses indicate that two VMs likely reside on the same server [86].

Watermarking is a simple signature measure used for identification purposes, which can be useful for post-hoc leakage point identification, as it uniquely marks the origin of ex-filtrated material [87]. The attacker penetrates the network activities of each of his malicious VMs, while measuring the victim VM's network performance. A delay in the network from the victim VM shows that its performance has been affected by the malicious VM, and hence the co-location is confirmed. Herzberg et al. [88] proposed a co-location method through two steps. In the first step, the attacker identifies the victim VM's internal IP address. The attacker then configures a client machine to connect to the victim's web service, for instance by downloading a file or a web page, via its public IP address. The attacker also uses a prober VM to transmit a large number of network packets to every possible internal address in the address block range. If the attacker's client machine notices that the victim's performance has been affected by the prober VM, then the victim VM's internal address is the one that the prober VM is flooding. Secondly, the attacker use a Time to Live (TTL) scanner, applying the internal addresses to count the number of hops between his VM and that of his victim. A zero TTL indicates possible co-residency.

Xu et al. [89] applied the DNS lookup method in finding out the victim VM's internal IP address, and consequently confirmed the co-location through two steps. The first step involved pre-filtering unlikely pairs of co-located VMs, by checking the /24 prefix in the internal IP addresses. If two VMs do not share the /24 prefix of the internal IP addresses, then they are not likely to be co-located. The second step is to use the bus locking covert channel to justify co-location, which involves building a bus locking covert channel between each pair of VMs. If two VMs can communicate with each other via this covert channel, then they are located on the same physical machine. Varadarajan et al. [90] also exploited the bus locking covert channels, as a means of evaluating the financial costs of co-location within different public clouds.

2.7.2 Inter VM-Communication, or Communication Between VMs and Hosts

Isolation is the main feature supported by virtualization. Such a feature, if not configured properly, can result in a potential threat to cloud infrastructure. Virtualization's isolation property ensures that applications executing one VM, do not interfere with the applications of another VM. It should be carefully observed that isolation means that breaking into one virtual machine should not allow for access to its co-located virtual machines within the same environment, and not even to its underlying host machine.

A shared clipboard within a virtual machine is a suitable program that permits data to be transmitted between VMs and the host. This application can also be considered a gateway for transmitting data between malicious programs in VMs. In the worst case, it can be used to "exfiltrate sensitive data to/from the host operating system".

In some VM technologies, the VM layer can keep a log of screen updates and keystrokes, through the virtual terminals that successively grant necessary authorization to the kernel of host operating systems. These captured logs are stored in the host, making it a possibility for hosts to observe these logs, even those of encrypted terminal connections inside the VMs.

Some virtualization circumvents isolation. The basic idea behind this logic is to support applications considered for one operating system, to be executed on another operating system, without any modification. This solution abuses the security bearers within both of the operating systems. In such systems, where there is lack of isolation between the host and the VMs, they provide the virtual machines unlimited access rights for the utilization of underlying host's resources of file systems and networks.

The communication between VMs is referred to as guest-to-guest communication, and such attacks are called guest-to-guest attacks. Here the attackers use one VM to access or control other VMs, through the same hypervisor. These attacks can happen without compromising the hypervisor layer. A malicious VM can potentially access other VMs through shared memory, network connections, and other shared resources [91]. For example, if a malicious VM determines where the memory of other co-located VMs lie, then it could be written or read in that location, and then it could interfere with other VM operations.In guest-to-guest attack, an attacking VM (VM1) can access VM2 and VM3 and so on. The attacker may or may not be authorized to access other VMs, but in this example, it accesses VMs without authority [37].

2.8 Survey of the state of the art

This section survey the state-of-the-art related work in the area of cross-VM attacks, particularly cross-VM network channel attacks for data leakage or to escalate the privilege level of non-root user.

2.8.1 Vulnerabilities in Network channel

Cloud services are accessed through the network using standard protocols e.g., IP which is considered to be untrustworthy [65]. Internet Protocol (IP) is the method of transferring data from one machine to another. Each machine is assigned an IP address for communication. There are several vulnerabilities within Internet Protocols, which are discussed as follows. The use of same IP addresses by different users may at times lead to accessing different resources of other users [33], [32]. Address Resolution Protocol (ARP) is a protocol used to map an IP address to a corresponding physical machine address [92].

Within cross-VM Address Resolution Protocol (ARP) attacks [93], the attacking VM launches an ARP spoofing attack by forging an identical IP address within the target VM, and sends an ARP packet to the virtual router. The virtual router updates the routing table when the spoofed ARP packet is received. As a result, any traffic directed to a target VM is instead sent to the attacking VM, which can then decide to either perform sniffing or modification. In bridge network configuration mode [93], the bridge acts as a virtual hub. All VMs share the virtual hub to communicate with the network. An attacking VM can sniff the virtual network by using a sniffing tool, such as Wireshark [94]. In the router network mode [93], a router plays a role of a virtual switch using a dedicated virtual interface to connect to each VM.

Address Resolution Protocol (ARP) Poisoning [95] is also considered to be a well-known vulnerability for Internet protocols [96]. By using this vulnerability, malicious VM can redirect all the inbound and outbound traffic of a co-located VM to the malicious VM, as ARP does not require Proof-of-Origin [33].

2.8.2 Return Oriented Programming

There are a number of different real-time attacks that exploit ROP systems. On the application layer, Adobe declared that a critical vulnerability had existed in the Adobe Flash Player 10.0.45.2, and its earlier versions [97]. These vulnerabilities also occur in Adobe Reader and Acrobat 9.x and are executed by applying ROP. As a result, this vulnerability bypass data execution prevention (DEP) [98], which is a security system implemented by Windows, thereby compromising the full system and making it possible for the attacker to take control of the victimized system [99], [100]. For the Windows operating system, the ROP-based rootkits have been used at the kernel layer. Upon execution, these rootkits can manage to hide malicious processes, files and network connections through windows. These rootkits are developed through ROP techniques, consequently circumventing kernel integrity protection systems such as SecVisor [101]. The ROP technique can be used to exploit the Apple iPhone, in which an unauthorized user installs applications, or leaks a customer's SMS database [102]. Table 2.2 presents already-existing cloud attacks.

2.8.3 Analysis

This section has surveyed cross-VM attacks to ascertain the most unknown research to this thesis. [32, 93, 103]. In these papers, researchers have used different approaches such as ARP spoofing, sniffing the virtual network and ROP. The purpose of these attacks are to redirect the network traffic of compromised VM and to describe how an unprivileged VM using ROP technique can manage to modify the code of hypervisor through which it can escalate its privilege level. However, all these attack strategies have some limitations as the security perimeter of cloud computing blocks such attack settings by placing more layer of isolation between VMs or to stop

executing arbitrary code.

Key lessons learned for the overall approaches are described as- there are number of potential avenues of research channels which are remain unexplored. Researchers have neither suggested any indication for the redirection of network traffic by exploiting the network channel through impersonation and mirroring approach nor show any possibility for the privilege escalation by applying the conjunction of ROP and exploitation of the network channel. This research work focuses on the exploitation of network channel for a variety of following reasons: (i) It arguably has the highest potential to redirect the real time network traffic of a target VM and to escalate the privilege level of unprivileged VM. (ii) Prior work only redirects the network traffic and escalate the privilege level using traditional approaches such as ARP spoofing and 'ret' statement that can be easily countered in virtualisation. The proposed approaches provides an in-depth analysis of the techniques involved and their effects, both in qualitative and quantitative terms, in various settings. (iii) The result of this proposed approach is so effective upon success, it can redirect the network traffic at hidden point as well as can control toolstack to manage other VMs.

Table 2.2: Overview of Cloud Attacks.

Attack	**Description**	**Ref.**
Co-location	Flooder	[85]
	DNS lookup	[89]
	Watermarking	[87]
Virtualization	OS level virtualization	[33]
	Application-based virtualization	[33]
	Hypervisor/VMM Rootkits	[33]
Side	Time-driven	[57]
	Access-driven	[55]
	Trace-driven	[56]
Covert Channel	TCP/IP Steganography	[61]
	TCP/IP Header Steganography	[62]
	Timing Channel	[59]
Images	Image Tempered	[67]
Memory	L2-cache	[56]
	Event Channel	[34]
	Grant Table	[70], [34]
DoS	The illegal use of resources	[72], [33], [34]
Network Channel	ARP Poisoning	[33], [95]
	Sniffing	[93]
	Spoofing	[93]
Row Hammer Attacks	DRAM Hardware Attack	[71]
Return-Oriented Programming (ROP)	ROP based rootkit	[99], [100] [101]

The anatomy of Cross-VM attacks is presented in Figure 2.6

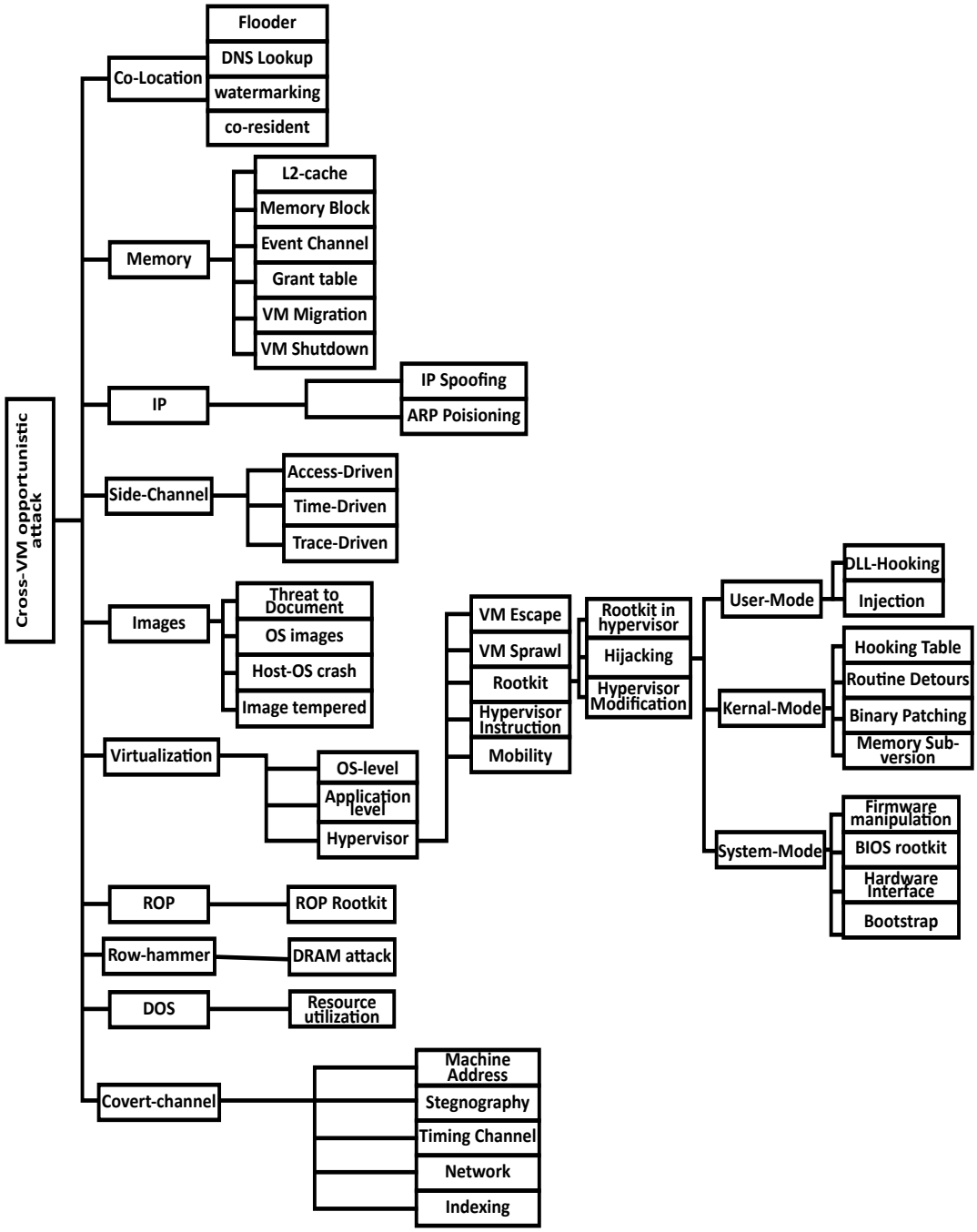

Figure 2.6: Anatomy of Cross-VM Attacks

2.9 Countermeasures

2.9.1 Reducing Side/Covert Channel Leakage

There are several methods cloud providers can use to defend against side-channel cache attacks. The first method to inhibit cache sharing divides it into different regions for different VMs or applications. This is achievable through the use of software or hardware approaches. As for software, some researchers have misused the page coloring approach for cache partitioning [104], [105]. Kim et al. [106] designed the application 'STEALTHMEM', used it to partition the Last Level Cache (LLC). It offers each VM a number of covert pages, which cannot be changed in the cache. Liu et al. [107] abused the Intel Cache Allocation Technology to prevent the victim VM's sensitive data from being exchanged, by the attacking VM in the LLC. New hardware caches have been introduced to partition the caches by ways or sets in order to protect against side-channel attacks due to cache exclusions [108].

The second approach practices randomization in system design, so that attackers do not obtain any useful information. For software, system clock measurements are blurred in order to interrupt the attackers' monitoring [109], [110]. Zhang et al. [111] launched Duppel, which periodically executes cache purging during VM's executions, and adds noise to attacker's observations. For hardware, new caches have been designed to randomize memory-to-cache mappings, and cache fetching [112], [108], [113], [114].

2.9.2 Optimizing Resources

To alleviate DoS attacks caused by resource contention, the cloud provider can optimize resource usage between different domains and reduce performance interference. For memory contention, the approach divides memory resources between different domains, such as Intel Cache Allocation Technology [115]. For I/O contention, the cloud server can observe and manage the bandwidth of I/O traffic, in order to avoid resource exhaustion. It can also use Direct Device Assignment to physically eliminate I/O intrusion between each VM [116]. These solutions have been widely adopted by public cloud providers. For the power resource, Li et al. [117] proposed PAD as a safe data center for power attacks under the over-subscription setting. PAD creates a virtual battery pool, which activates load sharing and adjusts power utilization for each rack. It can sense and shear power spikes, in order to avoid power consumption failure.

2.9.3 Protection of VM images

It is important to safe VM images in the cloud application store and eliminate potential vulnerabilities of their publishers and the retrievers. Wei et al. [118] introduced an image management system, Mirage to deal with the security issues of VM images by different methods: (1) Access control: Mirage follows access permission of two types, check-out and check-in, to closely observe

who is retrieving the images. (2) Image filters: Mirage uses special filters to delete sensitive information from the original images.

2.9.4 Network Defenses

Georgiev and Shmatikov [119] introduced 'CAPTCHAs' in the protocols to separate human users from malicious automated scanners.

2.9.5 Eliminating the Hypervisor's Vulnerabilities

Prior work has considered different methodologies for improving the security of hypervisors. The first approach develops new secure hypervisors. McCune et al. [120] presented TrustVisor, a tiny version of the hypervisor used to ensure the reliability of applications' sensitive data and codes. In this, a new secure guest mode has been introduced for running applications executed on x86 hardware architecture, supporting virtualization in order to impose strong memory isolation between the hypervisor, the host OS, and VM applications. TrustVisor also launched a software micro-TPM instance for each application, in order to perform integrity attestation. Vasudevan et al. [121] developed, configured and certified an open-source eXtensible and Modular Hypervisor Framework (XMHF). XMHF consists of a number of different XMHF main and small associate libraries. A hypervisor application can extend the XMHF core and the basic functionalities provided by this framework in order to execute preferred security features.

The second approach in this direction is to secure the integrity of hypervisors. Wang et al. [122] introduced HyperSafe, a lightweight method for ensuring the integrity of Type-I hypervisors at runtime. HyperSafe applies the Write Protect (WP) bit to prevent hypervisor pages from being compromised by malicious applications. HyperSafe can also configure a target table consisting of all legitimate destinations for indirect control flow instructions, as a means of implementing the hypervisor program's control flow at runtime. Azab et al. [123] designed HyperSentry as a tool for calculating the integrity of hypervisor at runtime. A remote client who needs to validate the hypervisor can use the Intelligent Platform Management Interface (IPMI) to initiate the server into the System Management Mode (SMM), and to measure the hypervisor's code, data and CPU state. Another direction is to reduce the hypervisors' privileges and functionalities.

NoHype [124], [125] is designed to eradicate the hypervisor during the VM's runtime, thereby reducing the attack surface from the hypervisor. NoHype attains this by pre-allocating processor, cores and memory for each VM during VM creation, allocating virtualized I/O devices directly to VMs, in order to circumvent the need for a hypervisor to do I/O emulation. It also modifies the guest OS to cache host system configuration for later use. Butt et al. [126] proposed self-service cloud computing as a means of confining the host VM's privileges. It splits the administrative privileges between a system-wide VM, and per-client administrative VMs. The per-client ad-

ministrative VMs are able to implement some privileged system tasks at their own VMs, while the system-wide VM cannot examine the code, data or computation of client VMs. In this case, security and privacy are conserved even if the host VM is compromised.

2.9.6 Avoiding Co-location

The main feature of virtualization is shared infrastructure which can be exploited. One way to eliminate the vulnerability of shared infrastructure, is to reduce or avoid the potential of co-residency between multiple VMs. This can be achieved through the launching of static VM or dynamic VM migration. During the launching of VMs, some cloud providers facilitate a choice of dedicated VMs, where the customers' VM can completely use a dedicated server without sharing with other VMs. Moreover, new VM placement policies have been redesigned, thus reducing the possibility of the co-residency of attacker and victim VMs [127, 128]. Some researchers have proposed frequently migrating VMs, so as to add greater difficulty for attacking VMs to co-locate with the target VMs [129–131].

2.9.7 Defeating Row Hammer Attacks

Cross-VM row hammer attacks can be defended against through the use of hardware or software solutions. For hardware, Error-Correcting Code (ECC) memory can be used to ensure the correctness of one single-bit error, and to detect 2-bit errors. This makes row hammer attacks much harder [132]. For software, Brasser et al. [133] has suggested two solutions. The first solution is to extend the system bootloader to recognize exploited memory pages. Row hammer exploitation tools are executed offline, determining which memory pages could be tempered by row hammer attacks [134**?**]. Then the boot loader indicates that these exploitable memory pages are inaccessible at boot-time, so that these pages will not be executed at runtime. The second solution is to extend the OS kernel, in order to impose strong isolation onto the physical memory of different system entities, such as user and kernel spaces. This ensures that memory between different entities is physically separated by at least one row, meaning that one entity cannot interfere with the memory of another. Irazoqui et al. [135] designed MASCAT, a static code analysis tool which can scan the application of binaries, and detect possible micro-architectural attacks, such as row hammer attacks. This tool applies the signature-based detection algorithm for searching binary files with implicit characteristics that micro-architectural attacks usually demonstrate in their design. In row hammer attacks, the attacker needs to continuously circumvent the cache, and access a fixed DRAM location. This is used as the signature of row hammer attacks.

The overview of all these attacks and their related countermeasures have been tabulated in Table 2.3.

Table 2.3: Attack and their Related Countermeasures

Attack	**Countermeasures**	**Ref.**
Co-location	Dedicated VMs	[127]
	Runtime VM migration	[130, 131]
	VM launch placement	[128]
Virtualization	Designing secure hypervisors	[120]
	NoHype	[124]
	Protecting hypervisor integrity	[123]
Side/Covert Channel	Sharing memory by dividing it into different regions	[104–106]
Images	Managing VM images	[118]
Memory	Scheduling adjustment to limit interruptions in memory	[136]
DoS	Optimizing Resources	[115, 116]
Network Channel	Client ID verification	[137]
	Assigning user rights	[137]
	CAPTCHA	[119]
Row Hammer Attacks	Error Correction Codes	[132]
	Memory Isolation	[133]
	Signature-based detection algorithm	[135]
Return-Oriented Programming (ROP)	Defeating ROP Attacks	[138]

2.9.8 Analysis

This section has surveyed the countermeasures of cross-VM attacks. Some of the countermeasures [126,137,138] are closely related to this thesis. In these studies, researchers have proposed different countermeasures such as dedicated VMs, assigning user rights and defeating ROP attacks. The purpose of these countermeasures are to block cross-VM, network channel and ROP attacks. However, all these countermeasures have some limitations as dedicated VMs on cloud computing are not appropriate for cloud providers because they save their operational cost by sharing the same hardware among multiple VMs. Similarly, assigning user rights to VMs and defeating ROP attacks can be circumvent by attacking VMs through privilege escalation. The lesson learned from the overall approaches is: researchers have proposed several countermeasures, but there is very limited research in proposing the countermeasures of heterogeneous attack strategies.

2.10 Discussion and Findings

This chapter has discussed the previous works which focus on cross-VM attacks and their countermeasure solutions within a cloud computing environment. Figure 2.6 summarizes these works according to the different types of attacks , which an attacker leveraged. It has been demonstrated in Figure 2.6 that researchers conducted substantial work to not only in exploiting

the shared hardware such as images, memory, virtualization but also in exploitation of hypervisor through ROP and change in memory codes to escalate the privilege rights. From the literature review and the current state-of-the-art, it is observable that there is a clear gap in exploitation of network channel and ROP through combination of different approaches to redirect the network traffic of co-located VMs and to escalate the privilege level of non-root VMs. On the one hand, the analyses are completely focused on how the attacking VMs are maximum utilizing or exploiting the shared resources to degrade or exploit the hardware performance and completely neglect the impact of those security failures. On the other hand, theoretical approaches emphasize the importance of reducing these failures without providing any insight about the characteristics and dimensions of the addressed problem. As a result, this creates the requirement for comprehensive security failure analyses in real cloud environments.

2.11 Summary

This chapter has provided a comprehensive literature review of state-of-the-art concepts of cloud computing, their essential characteristics, cloud computing models, security features, attack vectors and countermeasures. It has also been presented how they can be used to better study and quantify systems security in cloud model.

The abstraction of the cloud system model has been presented, and the concepts of a cloud system being composed of multiple nodes which interact with each other through different interfaces within the system environment have been introduced. Furthermore, the concept about virtualization has been discussed in detail and also its system support in terms of hardware and software. Cloud computing and their service models are also discussed in detail.

The concept and definition of Cloud computing, as well as the key terminology and characteristics have been presented. Furthermore, the components of Cloud computing security including CIA, different services, as well as virtualization, and servers have been discussed in detail.

The concept of potential attack vectors in cloud model has been presented, and the critical need for empirical analysis and modeling of Cloud systems security discussed. A literature review of the current state-of-the-art of analyzing and characterizing Cloud attacks, including co-location attacks, side-channel attacks and network-channel attacks, has been presented and discussed in detail. Finally, current gaps in the state-of-the-art for these attacks as well as opportunities where security of cloud system model can be enhanced has been highlighted. The countermeasures solution have been presented of already existed attacks which includes optimizing resources, network defenses , reducing side-channel attacks and defense mechanism of row hammer attacks.

From the analysis, we further conclude that all of the above work suggests a strong need for further exploration of cross-VM attacks and associated channels. In contrast to these points, there is a need for research to consider a ROP perspective on cross-vm attacks. Hence further research is required to apply ROP and impersonating attacks allowing new scientific insights to be gained through examining results in novel ways.

The next chapter further examines the general network and hypervisor architecture of cloud model, what are their main components. How network traffic passes through VM to access the external network and how hypervisor creates a domain isolation between co-located VMs to develop further the research questions and surrounding perspectives for this thesis.

CHAPTER 3

NETWORK AND HYPERVISOR ARCHITECTURE IN CLOUD COMPUTING

This chapter provides a qualitative study of network and hypervisor architecture and shares insights gained from their operations. More specifically, the main goal of the chapter is to examine the unique characteristics and main components of network and hypervisor architecture in the context of network traffic flow and domain isolation properties through in-depth observations.

The overarching purpose of this chapter is to provide deeper understanding of the cloud network system and hypervisor in terms of their main components, how they connect with each other, network traffic flow, hypervisor domains, privilege levels of root, non-root VMs and their domain isolation properties.

3.1 Cloud System Model

This section describes the core components, hypervisors used, network and their isolation properties, firewall rules, IP addressing schemes and their implementation in the network architecture of cloud model.

3.1.1 Nodes of Cloud Model

Following are the main nodes of cloud model.
Node Types: A node is a hardware machine that operates with the support of an operating system and well defined software are configured on it. At a high level of abstraction, following are node specifications and their descriptions [139].

Controller: These types of the nodes are responsible for executing the services of management software that are needed for the cloud platform to run [139].
Compute: Compute nodes execute the virtual machine instances. KVM is used as a hypervisor in this node. This node is also responsible for providing the firewall services. One can run more than one compute node in a setup [139].
Network: The responsibilities of such nodes are to ensure creation of virtual networks needed for the customer to create public or private networks and connects their virtual machines with the external networks, i.e. the Internet [139].

3.1.2 Modes of Cloud Configuration

The cloud model can be configured in three different modes as described below.
Single Node Setup
Cloud model offers a variety of different configuration set-ups that support different back ends and network configuration options. In a single node setup, only one server runs all services and also drives all the virtual instances [140]. Figure 3.1 illustrates the single-node setup.

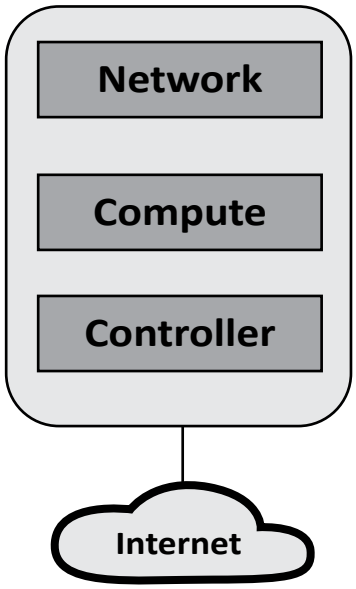

Figure 3.1: Single Node Setup

Multi-Node Setup (2-Node)
Two node architecture separates the control and compute nodes. Each node has its own functionality and features. Multiple services are run on these nodes separately [140]. Double-Node setup has been shown in Figure 3.2.

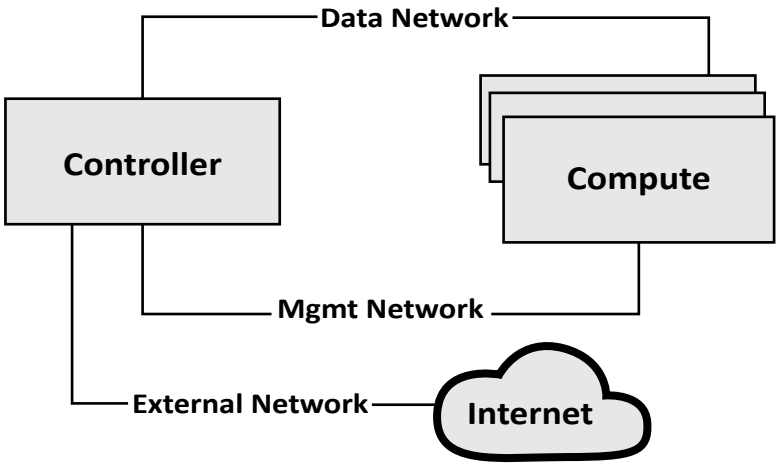

Figure 3.2: Double Node Setup

Multi-Node Setup (3-Node)

Cloud model meets different needs by enabling several options such as compute, networking, and storage, thus it proves highly flexible. In three-node architecture, a controller and a network node can also be added as additional nodes in a more complex multiple node configuration [140]. Figure 3.3 shows the three-node setup.

3.1.3 Network Services in Cloud Model

The networking services offered by cloud model is a standalone service that configures multiple processes across a number of nodes. These processes communicate with each other and other services. The leading process of the networking services is a neutron-server, a module written in Python that imports the Networking API and is responsible for forwarding the tenant requests to different plug-ins for further processing. The main Networking components [141] are:

Neutron Server (neutron-server and neutron-*-plugin)

This service configures and executes on the network node to service the Networking API and its extensions. It is also responsible for implementing the network model and assigning the IP addressing to each port. The neutron-server provides indirect access to a persistent database. This is possible through plugins, which connect with the database using AMQP (Advanced Message Queuing Protocol)[142].

Plugin Agent (neutron-*-agent)

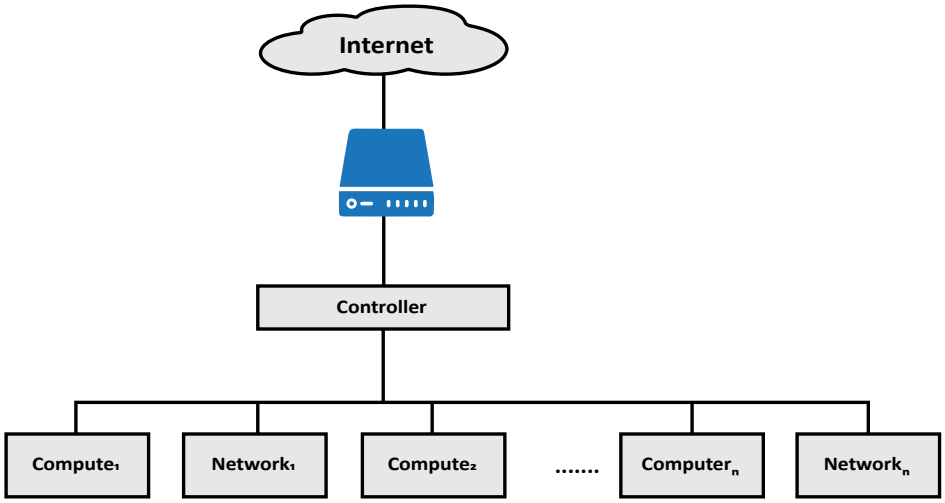

Figure 3.3: Triple Node Setup

It runs on each compute node and is responsible for managing the local virtual switch (vswitch) [143] configuration. This service involves message queue access and depends on the plugin used. Some special plugins like OpenDaylight(ODL) [144] and Open Virtual Network (OVN) [145] do not require any python agents on compute nodes.

DHCP Agent (neutron-dhcp-agent)
Offers DHCP services to tenant networks. This agent is responsible for managing DHCP configuration [146]. The neutron-dhcp-agent involves message queue access.

L3 Agent (neutron-l3-agent)
Includes L3/NAT forwarding that is used for external network access of VMs. it requires message queue access for message forwarding [146].

Network Provider Services (Software Defined Networking (SDN) server/services)
It offers additional networking services to user networks. These SDN [147] services may communicate with neutron-server, neutron-plugin, and plugin-agents through network channels such as REST APIs. Figure 3.4 depicts an architectural and networking flow diagram of the OpenStack Networking components:

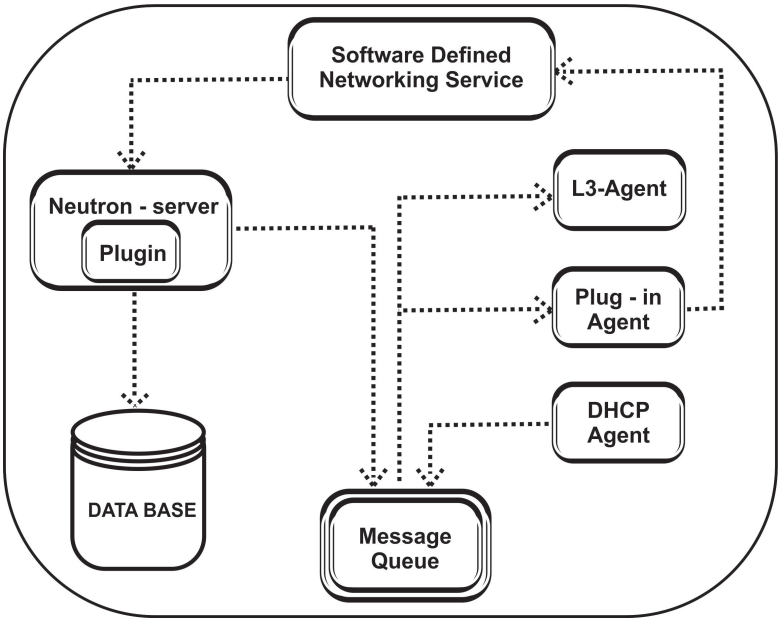

Figure 3.4: Cloud Model Networking Components

Placement of Cloud Networking Services on Physical Servers

A typical cloud architecture consists of a controller (host machine), a network machine and a set of compute machines for launching and executing multiple VMs. A distinct physical data center networks is required to setup a regular networking setup. These includes:

Management Network

Cloud components can be internally communicating by using this network. The IP address assigned on this network communicate within the local data center. Their IP addresses are not accessible from external network i.e the Internet [148].

Guest Network

This network is responsible for VM data communication within cloud platform. The IP requirements of such a network is based upon the plug-ins used and the network configured [148].

External network

This network facilitates VMs to access external world i.e. Internet. The IP addresses assigned in this network must be accessible everywhere on the Internet [148].

API network

API network is a special network that includes all APIs, including network APIs. The IP addresses assigned in this network should be accessible on the Internet. This acts as an external

network. In this network, it is also possible to construct a subnet for the external network [148].

The Cloud Networking Services and their connectivity with Management, Guest, External and API networks can been shown in Figure 3.5.

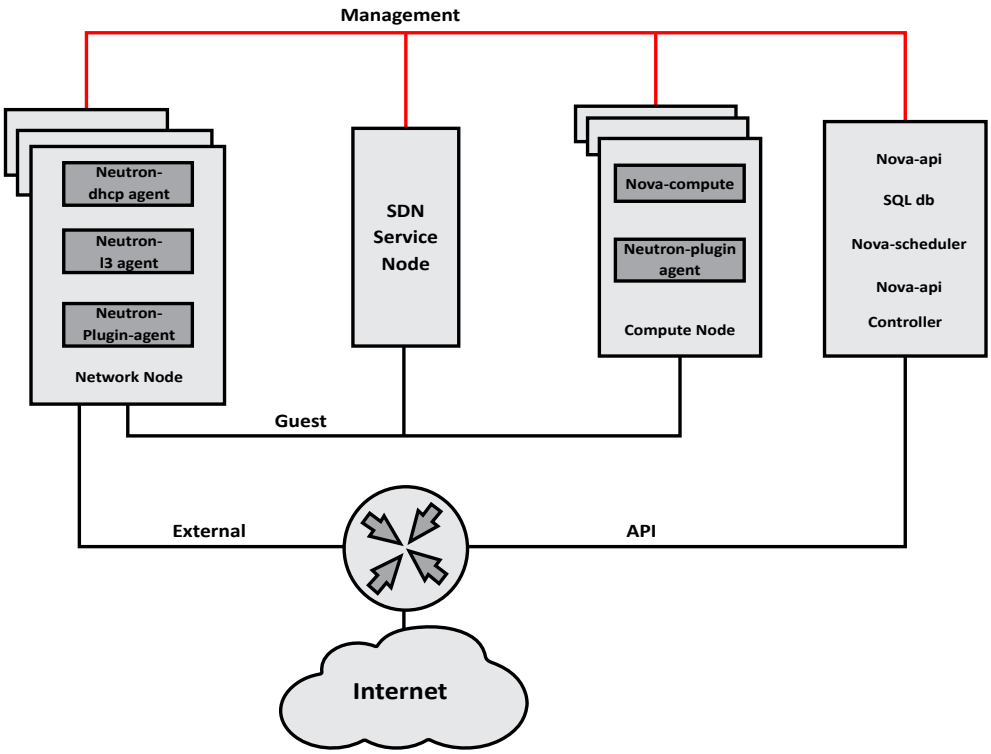

Figure 3.5: Cloud Model Networking Services

Networking Services: In the preliminary architectural phases of Network infrastructure, it is necessary to ensure that proper technical skill is offered to contribute to the configuration of the physical networking infrastructure, classify suitable security controls and assessing strategies. Virtualized network services are added as a layer in the networking which facilitates tenants to design their own virtual networks. Currently, the traditional networking counterpart services are more mature and secure as compared to these virtualized services.

Isolation using different methods: Cloud Networking can offer two different mechanisms for network traffic separation, VLANs (which follows the standard of IEEE 802.1Q tagging) or L2 tunnels using GRE encapsulation.

Isolation using VLANs and L2-tunneling: The choice and opportunity of network configuration determines which technique is applied for network traffic separation or isolation.

VLANs: Virtual LANs [149] keeps the isolation on a particular physical network comprising IEEE 802.1Q headers having fixed field value of VLAN ID (VID). Traffic in VLAN networks sharing the same physical networks is separated or isolated from each other. Each separate physical network offering VLAN networks is considered as a distinct VLAN trunk, with a unique VID values. VID values are valid from 1 to 4094. The complexity in configuring the VLAN network depends upon OpenStack network requirements. In order to grant OpenStack Networking to expeditiously use VLANs, valid VLAN range must be allocated and each compute node physical switch port is allocated to a VLAN trunk port.

L2 tunneling: Network tunneling encapsulates network combination with a unique "tunnel-id" that is used to recognize the network traffic linked to that network combination [150]. L2 network connectivity is not dependent upon the physical zone or underlying network configuration. By encapsulating network traffic inside IP packets, that traffic can pass Layer-3 boundaries, bypassing the preconfigured VLANs and VLAN trunking. Tunneling includes a layer of obfuscation to network data traffic, decreasing the reflectiveness of individual tenant traffic from a monitoring point of view. Both advance networking features i.e. GRE and VXLAN encapsulation are supported by OpenStack networking. The selection of methodology to configure L2 isolation is dependent upon the range and size of networks. If the network infrastructure has the availability of limited VLAN ID or will have a large number of L2 networks, tunneling is recommended in that case.

Additional Network Services: Network isolation describe how the network security and security perimeter is implemented. The following network services are optionally available to improve the security feature of network architecture.

Access Control Lists: Compute node supports VM network traffic access controls [151] directly when configured with traditional nova-network services, Traditional security groups in nova-network are configured to all interface (virtual) ports on VM using iptable. Security perimeter permits administrators and VMs to identify the network traffic type, and directions (traffic moving towards inward/outward) endorsed to move through a virtual interface port. When applying the networking services, it is recommended to enable security groups in this service model.

L3 Routing and NAT: Cloud networking implements routers that have the capability to connect multiple networks, and can also offer a gateway that further connects multiple private

networks to a shared external network, i.e., the Internet. The L3 routers [149] in an networking support Network Address Translation (NAT) [152] capabilities on ports (gateway) that connects the router to external network. This router supports floating IPs and SNATs (Static NAT) for all traffic by default, that links a static one-to-one mapping from a public IP on the external network to a private IP.

Firewalls (FWaaS): There is the need to manage and apply the ironic set of security features. Such features are provided by FWaaS which are normally broader and dense than security features offered by security groups [153].

3.1.4 Networking Services Limitations

Following are the well-known limitations in cloud networking services.

IP addresses Overlapping: If multiple namespaces are not supported by physical(host) machine, then networking service like DHCP and L3 agents must be run on different hosts. The main reason behind this concept is that there is no isolation between IP addresses created by L3 agent and the DHCP agent. If support of network namespace is not available then the other limitation of the L3 agent is that it only supports a single router.

Limitation of Neutron agents: Many plug-ins used neutron-l3-agent to implement forwarding, but neutron agents do not support IPv6 forwarding. [154]. The main limitation of neutron agents in case of IPv6 forwarding is OpenStack Networking does not support any services to facilitate any flavor of NAT with IPv6.

All these networking services and plug-ins are used in designing the network. As a case study, the network architecture of cloud computing has been designed where these services and plug-ins are widely used.

3.2 Cloud Network Architecture

Cloud provider provides highly rich security groups and policies that can be applied on a wide range of parameters on ingress and egress traffic of users, VMs, containers and applications. Security groups can be defined and applied dynamically per VM to ensure maximum protection. The flexibility and control of security groups allow enterprises to act accordingly and respond to threats quickly. Each VM has an IP address assigned to associated *Vif*. This IP address is only visible to the virtual switch, where all VMs are connected to *br-int*, i.e. part of open virtual switch (OVS) through *TAP* device which created automatically upon connection request. *TAP* is an access path where network traffic passes from VM to VSwitch. *br-int* and *br-ex* are OpenVSwitches

which are responsible for managing ingress and egress requests respectively as shown in Figure 3.6. Open-VSwitch is a virtualized switch and is part of hypervisor. It performs the functionality of Layer-2 switch and offers different features such as Access Control List, VLAN and many more. It offers subnet or private network to VMs [155].

Virtual switch is similar to virtual network interface which is organized by combining one or more virtual Ethernet interfaces. Virtual switch is further connected to a virtual router through *Veth pair*. *Veth pair* is responsible for making a connection between VSwitch and vrouter. The vrouter is used for maintaining routing table for ingress and egress traffic flow. Routers are implemented with the strict configuration of firewall (FWaaS) that implements security perimeter. The router further transmits the packet to its next hop, i.e. on the external bridge br-ex. The external bridge contains a physical network interface, eth0 which finally pass the network packet on the external network such as the Internet.

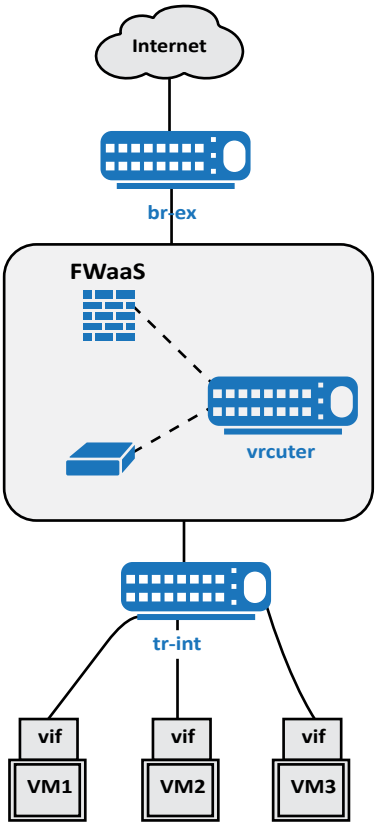

Figure 3.6: General Cloud Architecture

3.2.1 Vulnerability Found in Cloud Network Architecture

As discussed in Section 1.4, individual network components and their functionality to determine the vulnerability in a top down approach [156] have been investigated. An outbound packet starts on virtual interface (*Vif*) of the VM, which is connected to a *TAP* device. The *TAP* device is a part of the network and is attached to a bridge device. The bridge (*br-int*) is the central part where all VM's are connected through *TAP* respectively. The *br-int* performs *VLAN* tagging and un-tagging of traffic coming from and to VM. Each network is assigned a different *VLAN ID* to ensure the network traffic segregation between different VMs co-resided on the same hardware. Security configuration cannot allow VM to directly access the network bridge. It allows VM to connect through *TAP* device as shown in Figure 3.6. Hence, the only possible way to attack or penetrate in the network is by compromising the *TAP*. So we investigate the *TAP*, their attributes, connection with VM and *br-int* simultaneously and identify the vulnerability in *TAP* is as below.

The cloud network system allows connection of a VM's *TAP* interface with vswitch that does not have a private Ethernet Interface at the backend for internal communication between VMs or to access the Internet.

After discovering this vulnerability, network architectures of different cloud models and cross-cutting vulnerabilities found in OpenStack and Oracle Ravello network systems that share the same network architecture have been explored.

3.2.2 OpenStack and Oracle Ravello Networking

OpenStack and Oracle Ravello provides security groups and policies that can be applied to parameters on incoming and outgoing user traffic, VMs, and containers. Security groups, such as firewall rules, SSH control, and port bounding can be defined and applied dynamically to improve protection. The flexibility of security groups allows enterprises to quickly respond to threats. The network architecture of OpenStack (as shown in Figure 3.7) is implemented through four virtual networking devices:

- The Test Access Point (TAP) is an external monitoring device between the physical Ethernet card and each VM within the physical machine.
- The Veth Pair is a virtual network cable connecting the Linux bridge to a virtual bridge residing in a physical machine.
- The Open Virtual Switch (OVS) is the virtual bridge responsible for managing incoming and outgoing traffic. It functions as a Layer-2 network switch providing different features of Access Control List and Virtual LAN (VLAN), as well as providing subnet or private network functionality to VMs. Each VM has an IP address visible to the virtual switch and

is connected to a OVS bridge interface. The OVS contains two sub-components: (1) br-int is responsible for VLAN tagging (assigning the ID network traffic of VMs) and is used by cloud providers to implement isolation between VMs, and (2) br-ext, used to bridge the virtual bridge to the physical network device.

- The Linux bridge (L.B) is responsible for communication between the br-int and each VM's TAP, recording it through a MAC caching table that saves the address and port number of packets between the VM and the Ethernet card. This is used to prevent packets of unknown IPs flooding to all VMs.

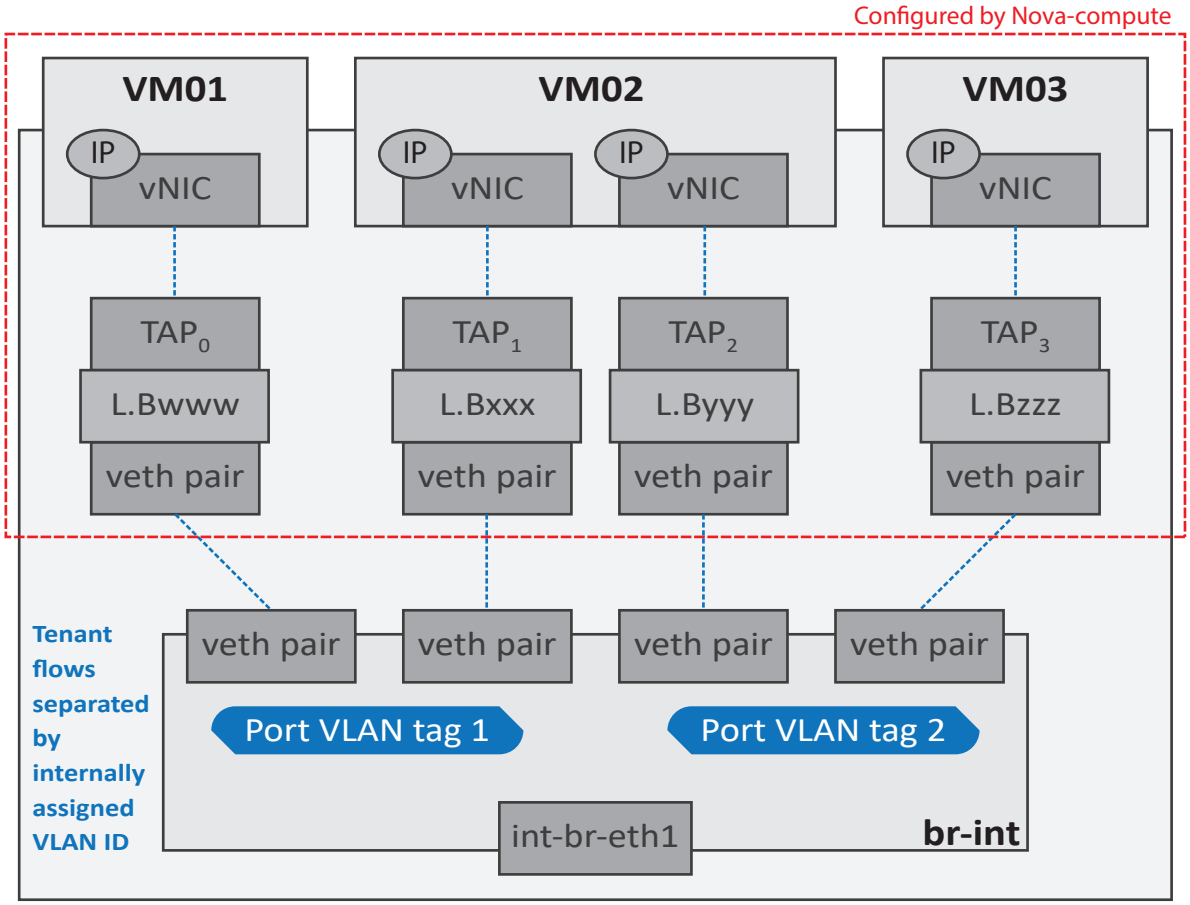

Figure 3.7: Networking Devices Used for Traffic Transmission

A VM creates and stores data on the associated VNIC, such as eth0. The data is then transmitted to the TAP on the compute host. Normally, a TAP offers an access path to data passing through a network. The TAPs are further linked to the Linux bridges that pass the data to Veth Pair which acts as one side of the cable. Data sent to one side of Veth Pair can be received

at the other end. The other end of the pair is on the integration bridge: br-int. This bridge is responsible for the attachment of all the VM's TAPs and any other bridge on the system. The integration bridge further connects with br-eth as shown in Figure 3.8.

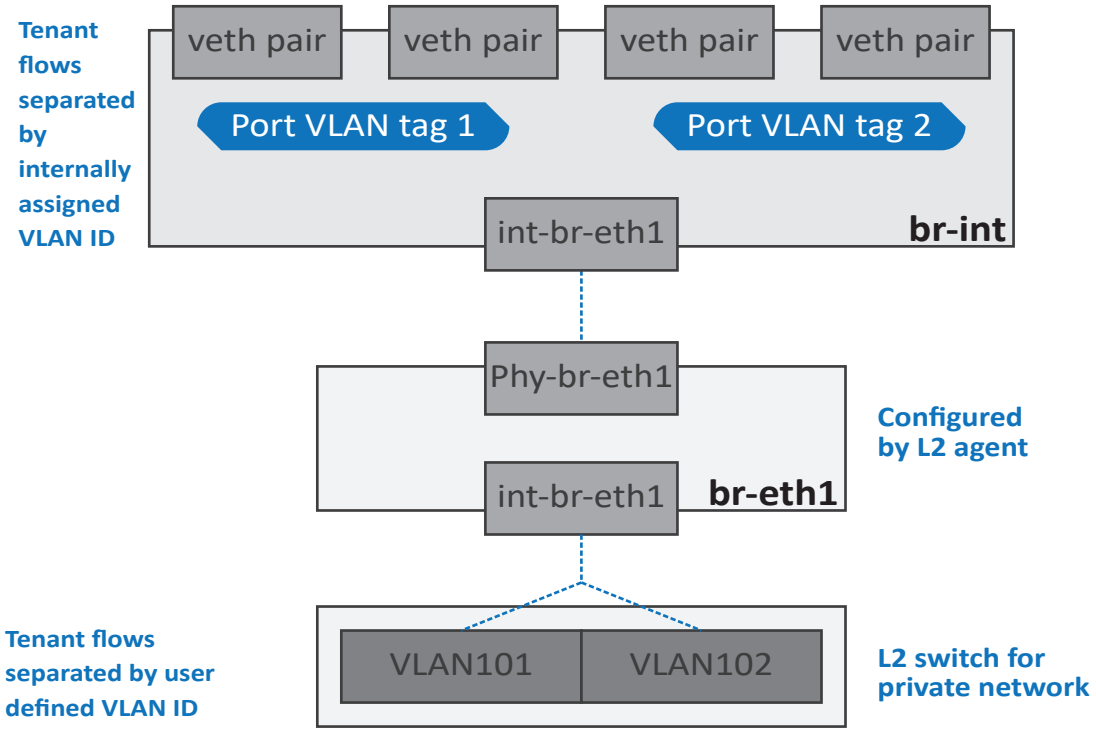

Figure 3.8: Function of *br-int* and *br-eth*

Proof of Concept As a proof of concept, we exploit the same vulnerability for Google and Microsoft Azure but didn't find it feasible. Since the success for exploitation of the vulnerability depends upon the factor of illegal *TAP* connection with the *bridge*, which is not granted in these cloud models, because of their different internal configurations of network devices. These cloud models before granting the connection of VM's *TAP* interface with *vswitch* ensure the connection of VM's private Ethernet Interface with *TAP*. They will grant the connection only if *TAP* is connected with the Ethernet.

3.3 Hypervisor Architecture

The domain architecture of *xen* hypervisor is explained in below section.

3.3.1 XEN

Xen is a concrete hypervisor that uses a microkernel strategy, furnish functions that let multiple computers (VMs) to parallel run on the same computer hardware . It provides the strong isolation between virtual machines co-resided on same physical machine. It is open source (GPLv2) and is operated by *Xen.org*, a cross-industry organization. Xen comprises of different products and projects. The Xen architecture is shown in 3.9

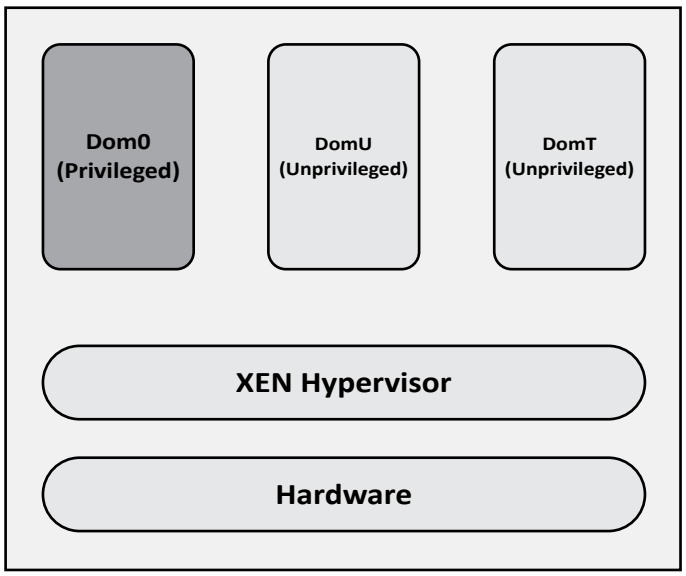

Figure 3.9: Xen Hypervisor Architecture.

Xen Project executes in a more privileged CPU state than any other software on the machine. It main responsibility entails memory management, CPU scheduling of VMs and launching of most privileged domain (dom0). Dom0 is the privileged domain and only virtual machine has direct access to hardware. It has full rights to control all the unprivileged domains (domUs).

3.3.2 Privileged and Unprivileged domains

A number of virtual machines VMs or domains can run on Xen. When the system boots, bootloader firstly loads the Xen, then Xen loads its main domain dom0. As Xen does not manage any device drivers, dom0 is used to access the hardware and can handle device requests. It is also the only administrative domain that is responsible for creating, pausing, unpausing, saving, and destroying other VMs residing on the same physical hardware. Contrary to this, other domains that dom0 launch have no such admin privileges and are called domU, unprivileged domain. Data that manages the domain privilege is a Boolean value store in a domain structure of hypervisor code. It is set to be 1 for dom0 and 0 in case of domUs. If the escalation of the privilege of domU is required, it is needed to modify the allocated value for this domU from zero to one [103].

3.3.3 Xen Vulnerabilities

If a non-root user desires to expand its resources in case of cloud expansion, it becomes a root user of its own domain and can further sublet its resources. The expansion of cloud model can be possible through return-oriented programming (RoP) but this RoP in turn is vulnerable to attack. As discussed in section 1.4, we implemented different approaches iteratively to find vulnerability in RoP of cloud system but failed to do so, because of strict cloud security configuration.

Hence, we conclude that exploitation of ROP alone is not vulnerable because of strict security configurations of modern cloud models. Therefore, we combine exploitation of ROP in conjunction with network channel and find it vulnerable through which the attacker can make an illegal network connection with the root user. Consequently, upon the successful (illegal) connection, the attacker can escalate the privilege level of its VM by breaking the domain isolation of hypervisors and control *ToolStack* from where it can manage other VMs. We investigate the feasibility of this vulnerability in the OpenStack architecture with Xen running underneath as shown in Figure 3.10.

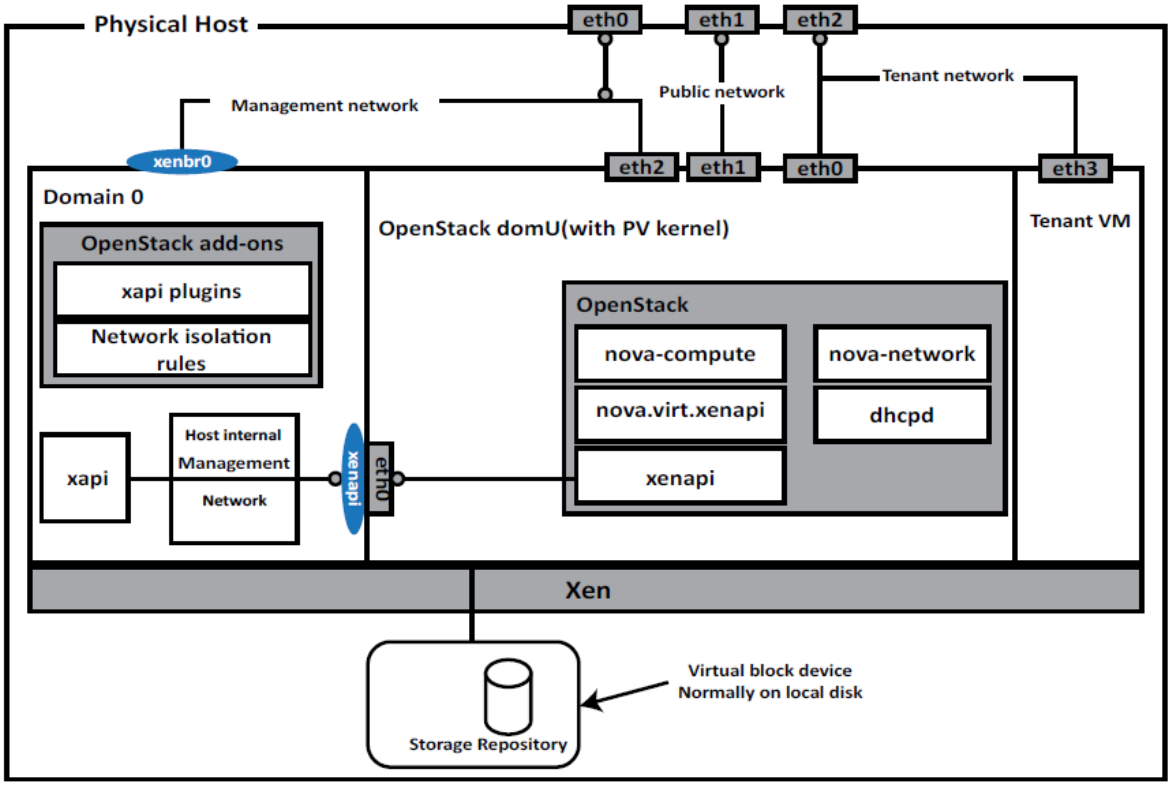

Figure 3.10: OpenStack Xen Architecture

A brief description of OpenStack with Xen architecture and its domains i.e Domain 0 (dom0)

and Domain U (domU) are discussed as follows:

3.3.3.1 Domain U

In virtualized environments, each guest VM run their own operating system and applications. The hypervisor supports two different virtualization modes: Para virtualization (PV) and Hardware-assisted or Full Virtualization (HVM) discussed in section 2.1.2. Both guest types can be used at the same time on a single hypervisor. It is also possible to use techniques used for Para virtualization in an HVM guest: essentially creating a continuum between PV and HVM. This approach is called PV on HVM. Guest VMs are totally isolated from the hardware: in other words, they have no privilege to access hardware or I/O functionality directly. Thus, they are also called unprivileged domain (or DomU).

3.3.3.2 The Control Domain (or Domain 0)

Domain 0 is a specialized Virtual Machine that has special admin privileges like the capability to access the hardware directly, handles all access to the system's I/O functions and interacts with the other Virtual Machines. It also manages a control interface to the outside world, through which the system is controlled. Domain 0, the first VM started by the system.

3.3.3.3 Toolstack and Console

Domain 0 consists of a special management tool called control stack (also called Toolstack) that lets a user to manage virtual machine creation, destruction, and configuration.

3.3.3.4 OpenStack Nova

It runs the XenAPI library code of domU to communicate with xapi of dom0. Both codes are written in python. It follows the communication path of Host Internal Management Network to reach from the domU to dom0 without leaving the host. Xapi to xapi library code is used for communication between dom0 to dom0. This normally is used in case of cloud expansion.

3.3.3.5 XAPI

XAPI is the fundamental component of Toolstack that manage a Xen based hypervisor. The job of XAPI's is analogous to libvirt's in the KVM world. XenAPI provided API of XAPI. Compute driver in OpenStack communicate to xapi, hence all xapi servers could be used with OpenStack [157].

3.3.3.6 XenAPI

XAPI provided API of XenAPI. Xenapi is also included in python library which is xapi client [157].

3.3.3.7 XenServer

XenServer is an open source virtualization platform that facilitates all features needed for any server and datacenter implementation entails Xen hypervisor and XAPI for management [157].

3.4 Analysis

From the analysis of architectures discussed in this chapter, several findings have been identified.

First, it has been observed after investigating the network architecture and traffic flow in cloud computing that there is a serious threat to cloud computing if we did not block the penetration of any external device into the network. The vulnerability found in leading cloud computing is the illegal connection of *TAP* device with internal network *vswitch*. Through this connection, the attacker can penetrate itself in the network system of cloud computing. We have surveyed different cloud models to discover this vulnerability and found OpenStack and Oracle Ravello are vulnerable to this attack. As a proof of concept, we have tested the same vulnerability in Google and Microsoft Azure but didn't find it executable.

Secondly, some vulnerabilities have also been discovered in Xen hypervisor architecture in case of cloud expansion. ROP can be used to expand the cloud model. But the attacker can exploit ROP in conjunction with the network channel to take control of the root domain from where the attacker can manage other co-located VMs.

The work in this thesis focuses in particular on the OpenStack and Oracle Ravello cloud system to explore the network architecture. OpenStack and Ravello provide compelling solutions today for the challenges of delivering flexible infrastructure for high-performance computing (HPC) and high-throughput computing (HTC), furthermore, the development community is rapidly expanding services to meet exponential future demands.

3.5 Summary

This chapter has discussed the system model of leading cloud platform OpenStack and Oracle Ravello System. The fundamental components of these cloud models are also presented in this chapter. Multiple configurations of Openstack have been presented in detail which shows the connectivity of nodes with different setups. Network channel along with the services running on networks were also discussed in detail. These networking services enforces VMs to observe isolation in shared hardware. Networking services like VLAN, Access Control List, firewall rules and network traffic routing were responsible to implement security feature in cloud.

This chapter comprises of two major phases: Analysis, and Learning. These phases run in an incremental and iterative manner to select cloud architecture. The analysis phase is comprised of discussion about different cloud deployment setups discussed in detail in this chapter. The selection of cloud deployment models is then used by the learning phase to build a model for the prediction of system performance.

Default cloud network architecture has also been explained in this chapter which usually followed by cloud providers at higher level. However, internal system level configurations are different of each providers. Network architecture of OpenStack and Oracle Ravello cloud system has also been discussed in detail which shows the main devices of networks and how network traffic of multiple co-located VMs passed through these devices.

Hypervisor is responsible to maintain isolation between multiple VMs. Hence, the study of hypervisor is also important for readers what are the main domains of hypervisors and how hypervisor keep isolation between root and non-root VMs. What are their limitations. The configuration of Xen hypervisor with OpenStack has also been depicted which shows the relationship between different APIs that makes a connection with root and non-root VMs.

Chapter 4 elaborates how cloud network architectures are configured with security perimeter, what are the steps to find the vulnerability in cloud network architecture, cloud models that are affected with this vulnerability, configurations of XEN architecture and finding of their vulnerability.

CHAPTER 4

A Cross-Virtual Machine Network Channel Attack via Mirroring and TAP Impersonation and their Countermeasure

As discussed in chapter 3, network designing is a fundamental component of cloud computing co-located VMs network traffic passes through different configured devices of network architecture. Data security and privacy is a primary concern for enterprises when they consider adoption of cloud computing. The purpose of this study is to investigate the implications of virtualisation on network security vulnerabilities in cloud computing. Physical co-residency with other co-located virtual machines in a shared hardware environment leads to the possibility of side channel attacks. Such issues have been highlighted by various researchers, i.e how such side channel attacks in a shared hardware support attackers to ex-filtrate sensitive data across co-resident VMs. These side channels act as a security gateway between users and an attacker. If an attacker manages to exploit these side channels, they can monitor the data of other VMs. As a result, cloud providers attempt to resolve such issues by ensuring the logical isolation of resources through an internal virtual network between multiple VMs. Virtual networks play a vital role in ensuring that one VM cannot interfere with other VMs running on the same host.

Currently, there is a substantial lack of empirical studies comprehensively determining network attacks and their countermeasures solutions. Current work focuses entirely on IP, ARP sniffing and spoofing that can compromise the co-located VM network traffic by redirecting it to some destination point but such attack schemes have been controlled through countermeasures. Current works neglect the exploitation of network architecture through some internal regular device responsible for data transmission and connection. Additionally, existing approaches do not comprehensively study the exploitation of network isolation in cloud computing and do not

provide a detailed method of network traffic analysis capable of quantifying and extracting empirical findings. This can be of practical use to researchers and cloud providers. This chapter presents the study and analysis of attack design by exploiting the network architecture of cloud computing and following are the major contributions: The key contribution of this research indicates that this internal virtual network, in turn, exposes further vulnerabilities in virtualization by mounting cross-VM attacks on a network channel. The launching of a novel network channel attack in which a malicious VM can redirect the network traffic of a victim VM running on a same physical hardware is presented in this chapter. Launching of such an attack entails overcoming the challenges of traffic redirection, pre-empting the victim VM traffic, entering into the current network infrastructure, launching an impersonation attack on interface and removing the footprints. This research addresses these challenges and demonstrates the proposed attack on two IaaS cloud platforms: OpenStack, an open source IaaS management system, and Oracle Ravello Systems, a public IaaS provider, with multiple setups that are configured with security requirements. In conjunction with our findings, countermeasure solution to the described attack has also been proposed.

4.1 Introduction

A cloud, a virtualization platform, is an Internet-based computing model [158] that comes in a variety of forms like grid, distributed, virtualization, utility and parallel. A user can conveniently use cloud utility services and request access to its resources such as storage, network, computing and applications on demand. It introduces the concept of 'pay as you go' [159]. One does not need to buy its own resources. Instead, one can just pay, use the resources and can enhance or reduce them on demand. It has some important features such as resource pooling, on-demand self-service, broad network access, measured service, and rapid elasticity. There are various service models, which are enabled by the cloud: Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS) [19] as discussed in chapter 2

The infrastructure of cloud computing depends upon the hypervisor that controls the virtualization environment and network layers which are at the top of the operating system [19]. Generally, a hypervisor has a narrow interface. Hypervisors are of two types Type-I and Type-II. Type-I hypervisor (native or bare metal) has a few lines of code on the host OS. Xen is an example of Type-I hypervisor. Type-II hypervisor (hosted) runs within a conventional OS environment. VMware and Virtual Box are the examples of Type-II hypervisor [160].

Contemporary virtualization technologies such as HyperV [161], Xen [60], and VMWare [162] are rapidly becoming the foundation for the security of cloud computing systems. Multiple VMs can co-reside on the same physical hardware due to virtualization of the cloud. The key

features of their attractiveness is a strong isolation between co-resided VMs, i.e. the guest VM running on the same system cannot interfere with other guest VMs [10]. Each VM has its limitations and boundaries. The key feature of strong isolation strengthens the security of public cloud computing systems such as Microsoft Windows Azure [163], EC2 [164] and Rackspace [165].

Virtual machine manager (VMM) [166] of virtualization systems ensure logical isolation between VMs by creating an internal virtual network. However, the threat of an attack remains. A myriad of recent studies (as discussed in Section 2.6) have demonstrated how co-residency can be achieved, and how an attacker can take advantage of shared hardware in cross-VM settings. In [93][26], researchers have discussed how an attacking VMs can potentially access other VMs by means of network connections, shared memory, other shared hardware resources and how an attacking VM can exploit network channel to redirect the network traffic of other co-located VMs and what are their limitations in the modern cloud model.

Despite the clear potential of cross-VM attacks for abusing the shared memory and disk as aforementioned in Section 2.6 and Section 2.7 , no actual demonstrations of the fine-grained cross-VM network-channel attack have been performed. The most commonly discussed challenges focus on the facts that cloud provider places more layers of isolation between co-resided VMs than in non-virtualized settings because the attacker and victim are often assigned to separate segmentation of virtual networks[46]. However, the proposed attack covertly redirects the network traffic of victim VM and are hard to detect as they leave little or no traces on the network.

Chapter 3 presented the cloud network model to identify the vulnerability in the state-of-the-art network architecture of the cloud model. The presented finding indicates that there are some technical challenges that needs to circumvent to exploit these vulnerabilities. Their implementation has been presented in the prominent open- source cloud model OpenStack and commercial cloud model Oracle Ravello system along-with the description of how to address the challenges to exploit these vulnerabilities which we found in Section 3.2.1.

Problem Statement and Contribution As discussed earlier, multiple VMs are sharing the same network resources which in turn are vulnerable to attack. But due to strict security configuration in the cloud network model the exploitation of these vulnerabilities in real-time scenarios becomes more challenging. This chapter overcomes these security challenges and designs a zero-day attack model (as mentioned in Section 1.3.1, research goal 1), focusing on the key components of the underlying network architecture, the technique and the way network components are connected, processed and applied for the overall model. The main concepts of the new attack model are validated through the implementation of a real-time system for OpenStack and Oracle Ravello. A series of experiments were conducted to assess the effectiveness of the proposed attack model. This aimed to prove the viability of implementing the system in an

operational context. At the end, defense strategy (as mentioned in Section 1.3.1, research goal 3)for controlling such attacks is also proposed.

4.1.1 Technical Challenges

As demonstrated in a lab testbed, the proposed attack establishes a network channel attack in which the malicious VM can passively monitor the network traffic of victim VMs by exploiting the isolation between VMs. Using OpenStack and Ravello cloud system as a case study, demonstration of the execution of proposed attack as how to place the mirror in internal medium of network channel in a way that maximizes the likelihood of an advantageous positioning. Having managed to position a mirror at target location, the next step is to redirect the network traffic of the target VM at destination point. While there are many avenues for such an attack, the main focus is on network-channels: cross-VM information leakage due to the sharing of network resources (e.g network bridge). An earlier attack on network resources [167] used ARP spoofing to redirect the network traffic of a target VM.

Table 4.1 presents a comparison summary of already existing attacks with our proposed approach. The second column shows the description of these attacks.

Table 4.1: Comparison of Related Work and Proposed Work

Attack	**Description**	**Ref.**
Side Channel	Time-driven	[168]
	Access-driven	[168]
	Trace-driven	[168]
Covert Channel	TCP/IP Steganography	[47]
	TCP/IP header Steganography	[169]
	Timing Channel	[170], [171].
DoS	Illegal use of resources	[32, 72, 93]
Network Channel	ARP Poisoning	[93]
	Sniffing	[93]
	Spoofing	[93]
Proposed Approach	TAP Impersonation and mirroring	

4.2 Attack Setting and Challenges

4.2.1 Attack Setting

The experimental investigations assume that an attacker has through some means achieved control of a VM co-resident on the same physical computer as the victim VM [56] by compromising an existing VM that is co-resident with the victim. The main focus is on the OpenStack cloud platform [172] running on contemporary hardware architectures. The attack setting in this

experiment is inspired not only by public clouds such as Amazon EC2 and Rackspace, but also by other OpenStack use cases. For example, many virtual network solutions are configured in the same way, where co-resided VMs are accommodated in centralized data centres on top of cloud hypervisor. Another representative use case separates operating systems into several components with different privilege levels and that are isolated by virtualization [173, 174]. An example of such systems are Qubes [175], which is an open source operating system running as multiple virtual machines on a hypervisor. As for the computer architecture, we target modern multi-core processors. This selection is mainly inspired by current processors used in public clouds such as Amazon AWS and Microsoft Azure. In this experiment, it is assumed that the attacker and victim are on separate network domains, each are assigned some number of disjoint resources such as virtual CPUs (VCPUs), virtual networks (VLANs), and virtual storage. All VMs are assigned the same privilege levels [176].

The attack model assumes that OpenStack ensures logical isolation between mutually untrusted co-resident VMs, and that the attacker is unable to exploit software vulnerabilities that permit it to take control of the entire physical node. The proposed attack, therefore, uses cross-VM network-channel to redirect network traffic of the victim machine. An example of a real-time network traffic of victim machines have been considered. Constructing such a network channel encounters significant challenges in this cross-VM setting. These key challenges have been discussed and a solution is presented to overcome each of them. The challenges correspond to the individual steps of full attack pipeline, which are depicted in Figure 4.1.

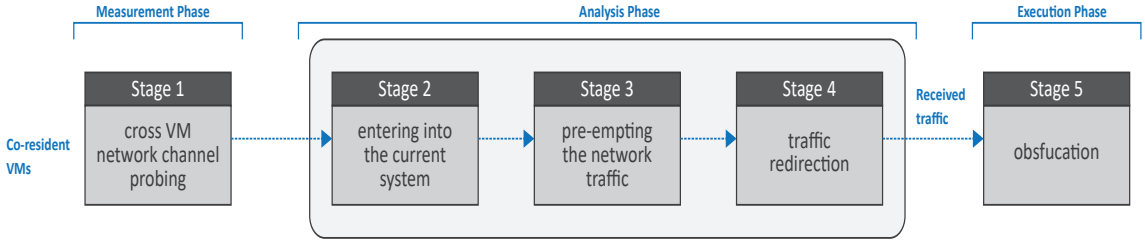

Figure 4.1: Main Steps in Network Channel Attack

4.2.2 Challenge 1: Observing the Current Network Architecture

The way OpenStack cloud works as per default secure setting makes spying of network traffic of the victim VM challenging, particularly when attacker and victims are on separate network domains. For example, network channel has high potential for spying the network traffic of victim VMs, but these require some illusions to control the network channel. An attacker must, therefore, attempt to control the network channel by breaking up the separation of the network domain to spy the network traffic of the victim. This approach has proven to be successful in

non-virtualised settings in which attackers that want to spy the network traffic of a victim exploiting the virtual network. However, no such exploitation of virtual network is suggested in our proposed scheme as the default security perimeter, i.e Firewall as a Service (FWaaS) of OpenStack continuously monitors any unusual activity in the current setting of OpenStack cloud and blocks the attachment of any external devices. Hence, to become a part or to directly access the current system is the main challenge. Therefore it is difficult to spy the network traffic of a victim VM in the presence of FWaaS because FWaaS block such spying. Before launching this attack, the first challenge is to identify a vulnerable target and explore the best ways to exploit it. In order to successfully launch the proposed attack, a single point of entrance is required to get started.

To overcome this challenge, a careful analysis of the current network system is required. To enter or to be the part of current system is a sequence of successful operations. The initial step in this sequence is to create an unaware device that does not have a running NIC adapter and that device should not be the part of an active network, i.e a dummy interface card as a vector for network channels (as shows by (1) in Figure 4.3). A regular device in a network that is responsible for data transmission and receiving is a network card. An equivalent device which is responsible for performing the basic network operations that a normal regular device executes is required to be created. Some network operations have been performed to recognize it in a system, i.e data transmission and reception through this device.

Normally, a dummy network interface card works in different ways as compared to regular network interface card. It gives benefit to those machines whose only communication source i.e IP connection is a dial-up connection. The concept with standalone hosts is that a single network device is activated in it, i.e the local loopback device, which is by default assigned an IP address 127.0.0.1. In some circumstances, one needs to communicate with the official IP address of the local host. As a case study, consider the laptop having hostname ‘test’, to experiment it has been disconnected from any network. An application running on test need now to send some data to another application on the same host. By observing test host entries on the path /etc/hosts shows an IP-address of 172.16.17.21, so the application attempts to send at this address. But in our experiment, currently, the local loopback interface is the only active interface in this system. The operating system kernel has absolutely no idea that this IP-address actually refers to itself locally. Resultantly, the kernel discards the packet by sending an error to the application.

At this point a dummy interface device comes in. It overcomes this challenge by simply serving as the alternate of the loopback interface. In case of test, it would simply be given the address of 172.16.17.21 and in the entry table of routing, a host route pointing to it is added. Every packet for 172.16.17.21 would then be successfully delivered locally. The most suitable command for this step is:

ifconfig dummy test
route add test

It is further demonstrated that how this dummy network interface card is connected with other devices to exploit the security perimeter of OpenStack cloud and to be the part of a current OpenStack system. This is the first step that gives the advantage to an attacker to become the part of the system, with the help of other operations, of an existing system and to spy the network traffic of victims. The strategy used to exploit a security perimeter is a vulnerability in an OpenStack cloud architecture, resulting in disclosure of privacy of other VMs.

4.2.3 Challenge 2: Hiding the Dummy Network Interface

In the event of OpenStack detecting that the real network card is activated, the next challenge is to mislead OpenStack's security perimeter by converting the identity of regular interface into a TAP (as shows by (2) in Figure 4.3), which acts as a valid device within the network system.

Impersonating the regular interface to TAP is necessary to ensure that we can connect our device to the network. The TAP acts as an inactive device as it does not interfere in any running network settings, and is compatible with the network configuration. The functioning of TAP is as: instead of VM's vNIC and switch, a TAP is configured in between them and is used to connect these two devices. A TAP delivers unfiltered complete access to bi-directional traffic streams. The data is passed between two network devices (i.e. VM's NIC and switch) in both direction (ingress and egress). A TAP makes copy of the transmit signals from each device. This certifies that data is copied and any chance of over-subscription is avoided.

The attacker VM now has two interfaces: (i) a normal Ethernet card, impersonating a TAP with no valid identity, and (ii) the dummy interface. Next, a connectivity request is sent to the Linux bridge, which presumes it to be a valid TAP, and adds it using a method similar to the conventional one. The reason for adding Linux bridge here is that iptables security rules on this bridge are used to configure the security group rules for the VM. After successfully completing this phase, the attacker can now penetrate into the network. This impersonation can be performed within OpenStack by: Manually removing all types of interface identity in the network configuration file at */etc/network/interfaces* and then restarting networking services by executing */etc/init.d/networking* restart that makes this change persistent within the system file. This results in a network device without a valid identity functioning like a TAP. The difference between TAP and regular network device can be seen in Figure 4.2. enp0s3 is the regular network device and test0 is the tap device which has no valid identity.

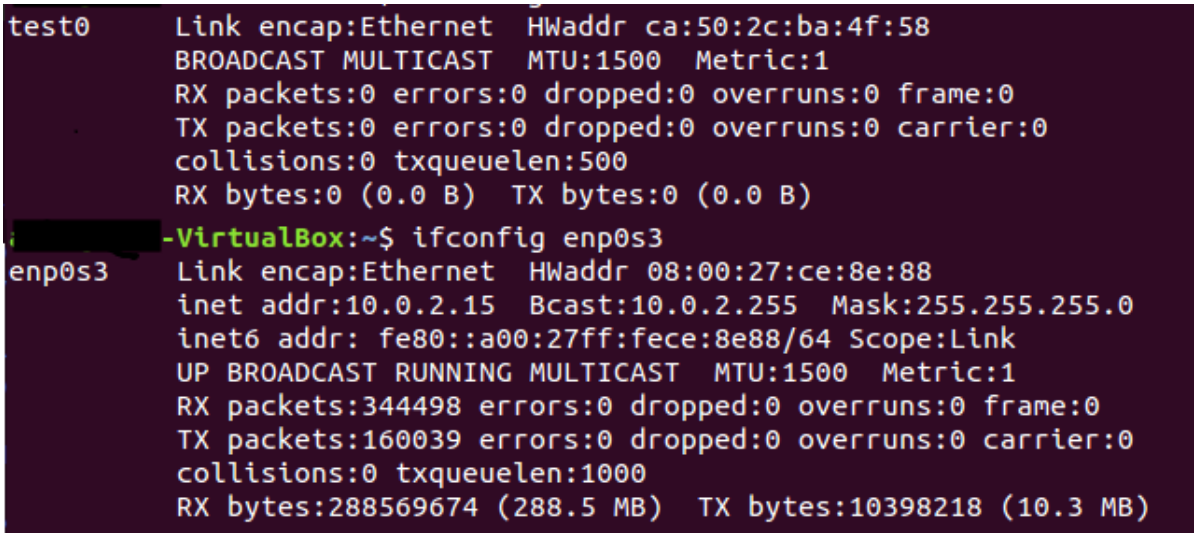

Figure 4.2: Difference between Tap0 and eth0

4.2.4 Challenge 3: Observing Network Traffic

Even with the strong feature of security perimeter in an OpenStack cloud, there are still some vulnerabilities in the current architecture that one can exploit to spy the network traffic of a victim VM. The best and novel way to spy the network traffic of victim VM in an unobtrusive way is to redirect the network traffic at attacker's destination port. This creates two hurdles. First, who is responsible for redirecting the real time network traffic. Second is the placement so to hide from others and that gives the maximum advantage. Already existing schemes [93] failed to provide any assistance in redirecting the network traffic. A novel method: a mirroring approach to relate network observations for the particular operations of the victim has been used. This challenge has been overcome by performing the following tasks. The first challenge of the proposed technique is overcome by implementation of a mirror. To implement this, a mirror has been used, specifically in Linux. A mirror is a powerful Linux tool that has the ability to redirect the network traffic from one port to other.

The next challenge is placement of the mirror. The mirror should be placed in a position that maximizes the likelihood of an advantageous positioning and that must be hidden from other VMs. This approach requires an illusion because if the mirror has been placed in any open position of the network, it can be easily detected, and the target VM will be careful in sending the traffic through that interface and report about traffic spying to the cloud administrator. This challenge has been overcome by setting up the mirror at the internal interface of the network bridge that is unobtrusive to all other VMs and this is the most challenging position as the traffic of all other VMs passes through this interface. In order to set mirror, the following steps need to be executed at bridge that will set the dummy interface as a destination port.

create Mirror name=test_mirror
select dst-port=dummy0
set mirror @ br-int

This technique takes advantage of penetrating into the network from where the network traffic of other VMs passes as shows (3) in fogure 4.3.

4.2.5 Challenge 4: Traffic Redirection at Destination Point

Spying the network traffic of victim VM is a challenging task. If an attacker observe the network traffic of victim VM at some open network location, there is a probability that attackers' activity is being monitored by security perimeter of an Open-Stack cloud and it blocks the attacking VM. This challenge has been overcome by redirecting the network traffic of victim at the set hidden destination point. When the network traffic of victim VM passes through the network bridge where the mirror is configured, it will redirect the network traffic from internal bridge port towards the set destination port. As with the installation of mirror, it sends a copy of all network packets seen on one port to another port, where the packets can be analyzed. Due to this, not only victim VMs but also security perimeter should be unaware about the redirection of network traffic. This re-direction is performed within OpenStack by the following commands as shows by (4) in 4.3:
select-src-port=@br-int
select-dst-port=@dummy0

4.2.6 Challenge 5: Obfuscation

After the implementation of all the operations, the challenge is to eliminate all the footprints of the attack by hiding all the devices and routes which are used in launching this attack. The key objective of obfuscation is to confuse and diverting examination and monitoring processes so that the attack is not be traceable [177]. route [178] is a very powerful Linux and other distribution tool such as Ubuntu that show all the possible static route and interfaces within a network. The existence of used devices as well as redirected route from *route tool* and other network monitoring tools have been concealed. There are many approaches that are used for obfuscation. After attachment of the impersonated interface to br-int, a Zero will be added out to this interface, that removes identity of the interface and is not visible in route tool. Due to this activity, performed action cannot be examined.

4.2.7 Putting it all together

Combining all these stages together allows us to launch a cross-VM network attack as depicted in Figure 4.3. The vulnerability that is exploited in this attack is cloud provider's permission

to bridge a TAP interface with no private Ethernet at the back end which is used to access the Internet.

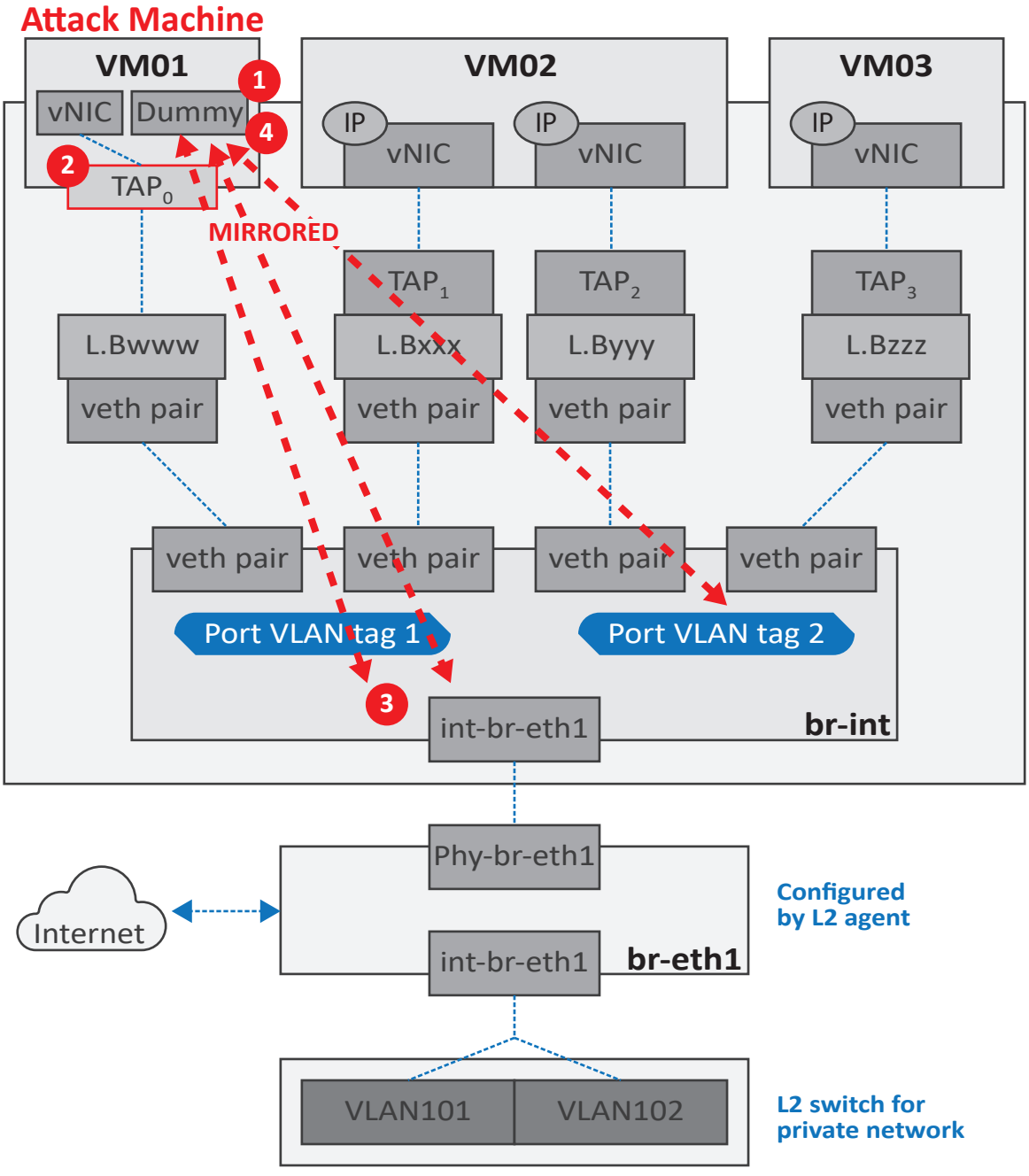

Figure 4.3: Attack Scenario in OpenStack

4.3 Evaluation Criteria

From Figure 4.3, it is observable that the evaluation framework revolves around the research methodology discussed in Section 1.4. The main goal of the attack is to redirect the network traffic of co-located VMs by exploiting the network channel. To achieve this objective, the five attack strategies are implemented in the network architecture of OpenStack. As network traffic is a major component of this attack, the evaluation results are mainly dependent on the network traffic but also partly focus on network resources.

The evaluation criteria are based on the following key reflective qualitative aspects. *(i) Functional* shows the accomplishment of the above said five tasks that are important for the execution of attack.*(ii) Expressive* indicates how well does the attack methodology do the job. (iii) Finally, the overall strengths and limitations of the attack has been shown.

4.4 Evaluation

The experiment was conducted in leading open source IaaS cloud platform i.e OpenStack and a commercial cloud system Oracle Ravello System. The OpenStack and Oracle Ravello cloud model has been configured with default security perimeter. Firstly, the experimental setup of OpenStack has been described in detail then in next section Ravello setup has been explained.

4.4.1 Experiment Setup

The configuration running on OpenStack cloud model consists of CPUs, KVM hypervisor and Linux kernels that are very similar to that used for host OpenStack instances. The configured architecture avoids two technical complications in conducting the attacks: it ensures that one VM is completely isolated with others, and secondly, VMs are actively communicating with each other. In general, the attacker can either passively wait for the victim VM to start communicating and then to spy message, or can actively spying the communication by redirecting its traffic. Moreover, a situation has been considered to the attacker's advantage, in which victim VM is already communicating with other VMs. It is also important to understand that the designed architecture is by itself a realistic secure setup for virtualised environments as the configured setups are similar to real time setups. In fact, in a cloud data center isolation among multiple VMs may improve the security as one VM cannot interfere the operations of other VMs. It is thus realised that these attacks are of interest beyond their progress towards an attack within OpenStack.

4.4.1.1 Attack Scenario:

The scenario that is used to demonstrate the attack is for an attacker VM (VM1) to intercept and spy on communication between two target VMs (VM2 and VM3). The placement of the mirror can then be used to perform redirection of target traffic. The attack has been performed within OpenStack using multi-node configurations encompassing single node, double node, and triple node. Three guest VMs have been deployed within a physical machine (Intel Core Z Q9650 @ 3.0Ghz) using the KVM hypervisor. VM1 was configured to be the attacker VM that has been successfully hijacked, while VM2 and VM3 are targets. Each VM is configured with two vCPUs different operating systems including Ubuntu 15.10(VM1), cirros (VM2), Windows 10 (VM3), are configured with both floating and private IPs for external network access and internal machine communication, respectively. Target VMs were configured to send between 0 - 15 Kbps to each other. All the experiments were repeated 20 times each.

Table 4.2 shows the total resource allocation of all co-located VMs. However, VM resource utilization statistics of one-week has been tabulated in Table 4.3. The value of resource utilization mentioned in 4.3 are not fixed; it varies from time to time depending upon the VM and their running programs. Network traffic has directly no connection with these hardware resources as network does not put traffic load on disk and memory. Network load can be observed through network monitoring tool i.e. mrtg[x], prtg[x].

Table 4.2: Resource Allocation of Each VM

Co-located VMs	**Memory (MB)**	**Disk (GB)**	**CPU**
VM1 (small)	512	10	1
VM2 (Medium)	2048	20	2
VM3 (large)	4096	40	3

In Table 4.2, the *CPU column* shows the number of the virtual CPUs for VMs running on the physical machine.

The *MEMORY MB* column shows the memory (in MB) allocated to the VMs running on the physical machine.

The *DISK GB* column shows the the root and ephemeral disk sizes (in GB) assigned to VMs running on the physical machine.

Table 4.3: Summary Statistics of Each VM

Co-located VMs	**Memory (GB)**	**Disk (GB)**	**CPU (Hours)**
VM1 (small)	365.06444	5.15	525.12
VM2 (Medium)	644.09474	9.64	564.83
VM3 (large)	4523.06489	12.45	746.0

4.4.1.2 Network Setup:

VLAN Manager was configured to ensure VM isolation between co-resident VMs by assigning IP addresses and VLAN tag within different ranges known to produce physical resource separation.

4.4.1.3 VLAN Manager Setup

When VLAN mode is enabled, each VM has its own VLAN and assigned network. Any physical switches placed in between must support 802.1q VLAN tagging for this to function. For correct functioning of the VLAN, the following configuration is specified in /etc/nova/nova.conf:
network_manager=nova. network.manager.VlanManager
vlan_start=100
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge

4.4.1.4 Association of Public IPs to VMs

A private IP address is automatically assigned when an VM is instantiated. This range of IPs are only accessible within the local network environment . Public IP addresses are required by the VM to be accessible to the external network. Manual attachment of a public address consists of (1) assigning an address from the available IP range, and (2) linking the address with a VM. This experimental setup must have a valid range of floating IPs assigned to it for allocation. Nova client has been used:

nova floating-ip-create and associating this address to a VM (such as 172.10.1.1) *nova add-floating-ip <VM-id> 172.10.1.1*

This allows for communication with VMs that contain public IP address.

4.4.1.5 Security Configuration

Networking uses **iptables** to achieve security group functionality, but **iptables** support linear storage and filtering. In order to improve security group's performance, **ipset** is preferable. So, **ipset** option is enabled in networking to improve security group performance by denoting a hash table. When a new port is created, an additional ipset option is added to the iptables chain. If the security group that the port belongs to contains rules shared by other groups, the group member is added to the ipset chain. If a group member is changed by using ipset, iptables rules are updated instead of being reloaded. Therefore, a new VM security group has been initiated by first individually checking for new security group names.

4.4.1.6 Managing Security Groups

Security groups are configured on the nova-compute host responsible for VM execution, enabling safeguarding of the host machine by limiting access and preventing intrusion of other VMs

running on the same host. A security group has been launched on port 22, which is the default port for security group. The design of a security group requires two phases,

i Defining a group by using the command nova secgroup-create.

ii Setting rules in the group using nova secgroup-add-rule:

- For ingress traffic, only traffic matched with security group rules is permitted. All other traffic is dropped, if there is no rule matched.
- For egress traffic, only traffic matched with security group rules is permitted. All egress traffic is dropped, if there is no rule defined.
- When a new security group is created, rules are automatically added to allow/disallow all ingress/egress traffic.

4.4.2 Assumption

The main assumption in our designed attack model presumes that an attacker has managed to take control of a VM residing on same physical machine as the target VM [56]. Researchers have already demonstrated success of this control by using a network-based approach to launch co-location attack within a pubic cloud such as Amazon EC2 [56]. They showed that an attacker is able to launch many VM instances within the same geographical region as the target VM, and applying different schemes to locate whether a VM is co-located successfully. Following is the summary of such schemes that helps in identifying the location:

- By running a trace-route program such as TCP SYN to detect the first hop of network traffic (e.g. Dom0 in the host Xen server) between attacker and target VM. The finding of this program indicates that if there is an identical Dom0 IP address means attacker is successful in finding co-location of target VM.
- Analyze round-trip time (RTT) [179] of network packet between attacker and target VM. A smaller value of RTT indicates that the two VMs share the same physical machine.
- Check internal IP addresses of attacker and target VM. Numerically close internal IP addresses reflects that the two VMs are more likely to be on the same server.

4.5 Analysis of Result

This section describes the in-depth evaluation of the experimental results of the overall approach through the real-world scenario in terms of capturing the network traffic and network resources utilization by VMs. The single result alone is not sufficient to make a firm conclusion about the feasibility of the underlying attack strategy. Therefore, the effectiveness of the proposed

technique is evaluated using network traffic capturing and network resource utilization.

Network Traffic Capture To validate the attack strategy, a collection of network traffic captured from the testbed is analysed. Each experimental run yields redirected network traffic that is labeled with the ground truth regarding the presence of an attack.

The real-time attack scenario is- Target VMs (VM2 and VM3) are communicating to each other by sending ping command. By applying an attack methodology as discussed in section 4.2, the attacking machine VM1 is able to observe the traffic between target VMs as shown in Figure 4.4. In Figure 4.4, the attacking VM is getting an ICMP echo reply. ICMP is an protcol running behind ping command. echo *request* and echo *reply* shows the successful communication between two the VMs.

The attack is effective when it is possible to determine sender and receiver communication between target VMs (e.g., packet header source IP address). If VMs communicate at the same time, the attacker is unable to distinguish the origin of VM traffic.

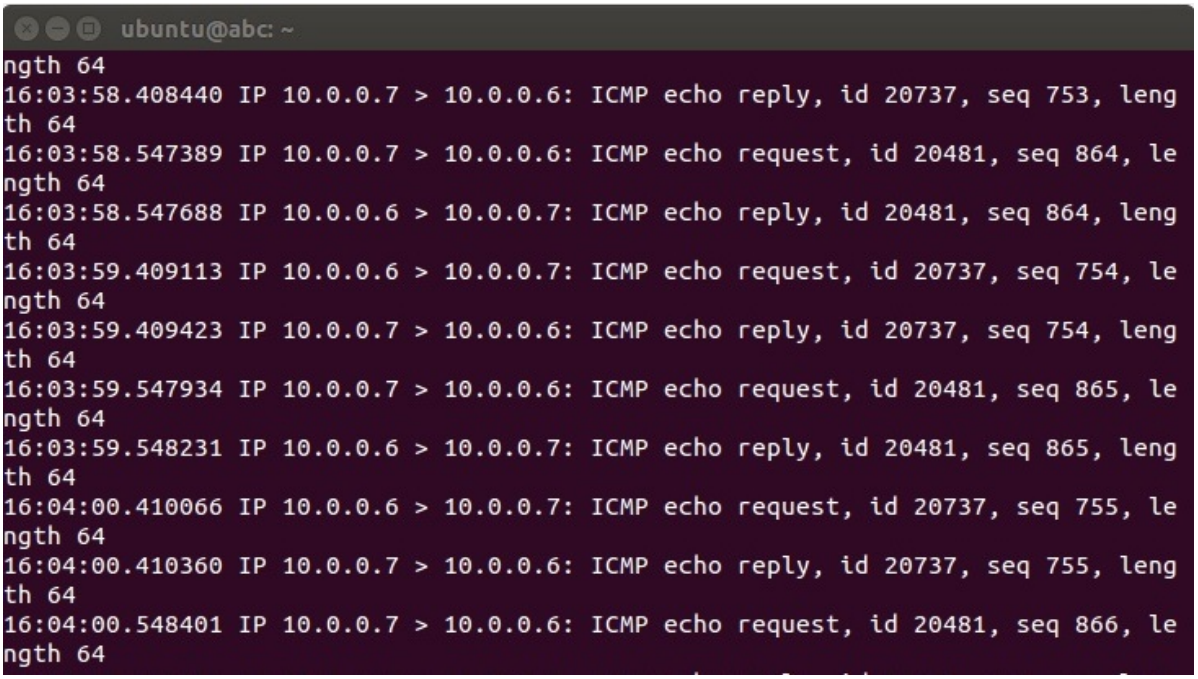

Figure 4.4: Traffic Capturing at Attacking VM

Network Resource Utilization

Subsequently network traffic is generated within each VM when an attack is performed. Figure 4.5 shows VM network traffic prior, during, and post attack. Experiment time between 0 to 30 minutes depicts that the attacking VM1 exhibits random network traffic not dissimilar to that of

VM2 and VM3 (point A). The attack commences at 25 minutes (point B), where it is observable that the attacking VM1 consumes substantial network traffic compared to target VMs, and continues to do so for 6 minutes until attack completion (point C). The reason for this sudden increase is because all target VM network traffic is redirected through the attacking VM.

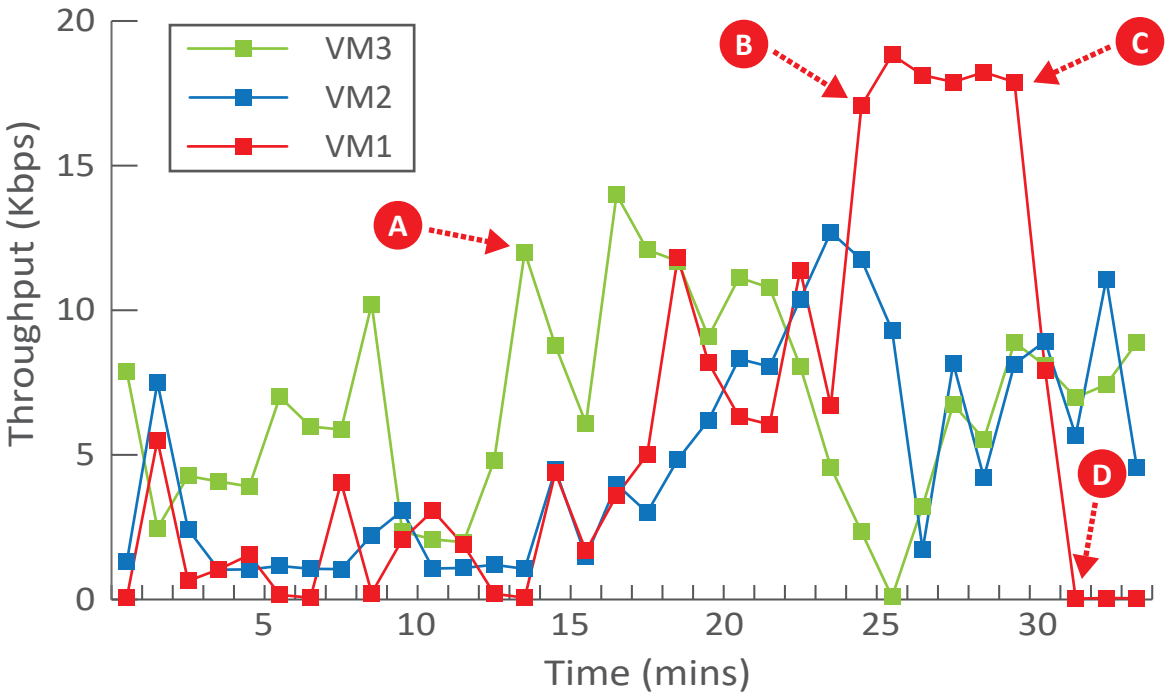

Figure 4.5: Normal Co-residing VM-Network Traffic

When observing network traffic of VM1 in comparison to other VMs during the entire experiment, it is possible that the cloud administrator could detect this as anomalous behavior due to the sudden spike in network usage. This may lead to further investigation, or deploying a simple countermeasure to restrict VM network traffic that reaches a defined threshold. However, it is believed that in the context of cloud computing such a counter measure would be challenging to detect. Firstly in many public cloud settings, VM resource usage are seen as black boxes by the provider. As long as resource demands do not violate resource capacity requested by a customer, this is seen as typical behavior. Secondly, even if the system is attempting to monitor atypical resource patterns using bandwidth monitoring tools such as prtg [180], countermeasures will likely include a time delay for determining irregular resource patterns. Therefore, even a few minutes of a VM being compromised may be sufficient for an attacker to achieve his objectives. For example, in 2008 the defense solution of a system operated by the Georgia Government during an HTTP attack [181] activated for 5 minutes after the attack had been launched. Finally, detection of atypical network traffic patterns becomes incredibly difficult if the attacking VM is capable of creating cyclical network patterns prior to an attack, as shown in figure 4.6.

The potential countermeasure strategy for such cases is to place a Network intrusion detection systems (NIDS) [182] at a points within the network to monitor traffic to and from all sources on the network.It performs a traffic analysis at different time intervals and compare the traffic to find out the attacks. Once an abnormal behavior in network traffic is observed, the alert can be sent to the administrator. But in our case, the attacking machine is generating the same amount of traffic periodically and in the third peak as shown in Figure 4.6 executes an attack following a similar resource pattern seen previously (point A to B). Hence, the abnormality in traffic pattern cannot be observed. In this particular case, ideally the administrator would need to scan all inbound and outbound traffic, but the limitation in doing so might create a bottleneck that would effect the overall speed of the network.

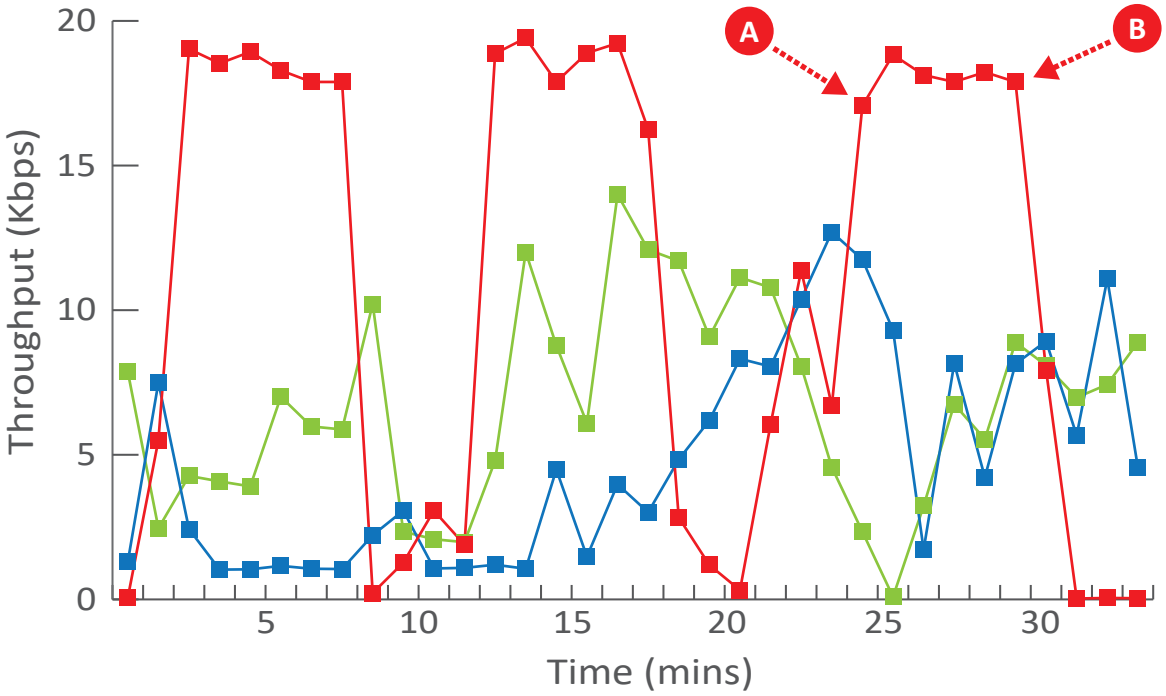

Figure 4.6: Cyclic Attack Pattern of Network Traffic

A larger network system is also considered to evaluate the attack as shown in Figure 4.7. In this scenario, 10 VMs have been considered. Subsequently network traffic is generated within each VM when an attack is performed. Figure 4.7 shows VM network traffic prior, during, and post-attack. Experiment time between 0 to 30 minutes depicts that the attacking VM1 exhibits random network traffic not dissimilar to that of target VMs (point A). The attack commences at 25 minutes (point B), where it is observable that the attacking VM1 consumes substantial network traffic compared to target VMs, and continues to do so for 6 minutes until attack completion (point C). The reason for this sudden increase is because all target VM network traffic is redirected through the attacking VM. Post routing traffic graph *'prtg'* [180] is used to capture network traffic

of all VMs.

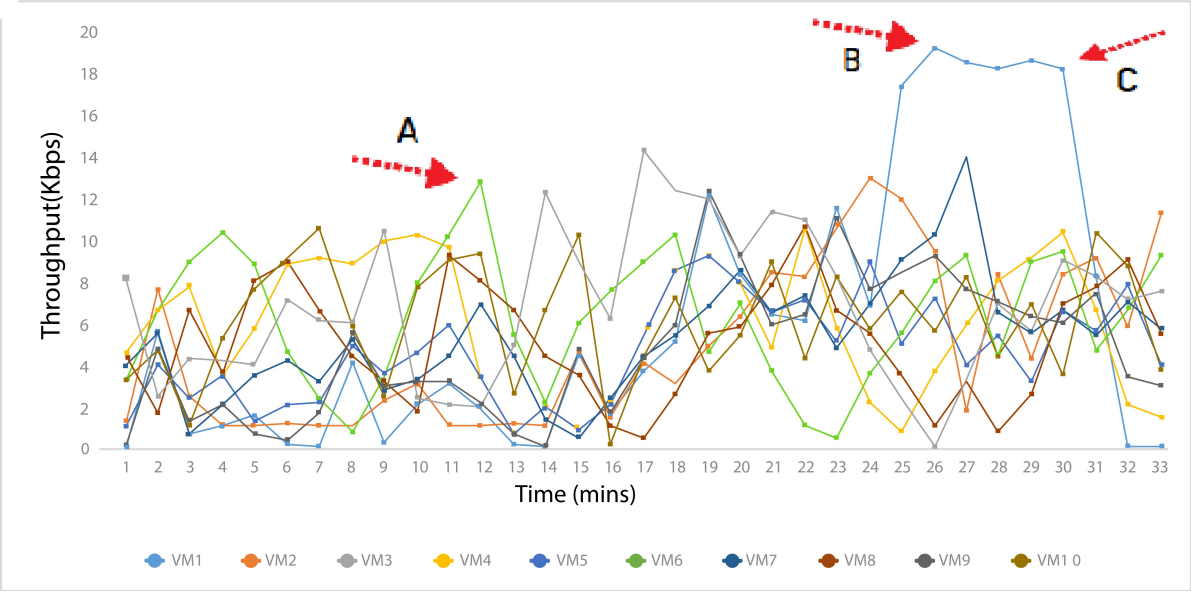

Figure 4.7: Cyclic Attack Pattern of Network Traffic

4.6 Attack on Ravello Systems

The same attack could be evaluated on Ravello Systems, Oracle's cross-cloud platform. Oracle Ravello is a commercial cloud platform that allows the use of any of the leading public clouds such as GCE, AWS, etc. In our testbed, AWS is configured to be the underlying IaaS provider. Ravello offers unique cloud application hypervisor technology facilitating enterprises and individuals to encapsulate completely and abstract an entire multi-VM application and its environment so that it can run on any cloud (public or private) without any modifications.

4.6.1 Network Configuration

Ravello allows to set up the network configuration for each of the VMs including static IPs. One needs to assign the static IP, Netmask, Gateway and DNS server for this interface. Ravello support the feature of SDN that automatically creates a virtual switch based on the IP address assign to VM and netmask of the interfaces. All interfaces that have same subnet are assign to the same virtual switch. If a gateway has been assigned at user interface, then its SDN will follow the following properties:

1. The guest VM has been assigned a gateway IP (i.e the VM has been configured to serve as a router), then Ravello's SDN settings will be in accordance with the VM's Gateway setting.

2. If the guest VM does not have a Gateway IP, Ravello's SDN will add a virtual Router with the defined Gateway IP, and add it to the virtual switch.

3. If a default Gateway is not defined on Ravello's user interface, Ravello's SDN will simply do nothing.

4.6.2 Evaluation

As shown in figure 4.8, all VMs belong to different VLAN Tags. VLAN tagging is defined in *IEEE 802.1Q* [183]. Network parts in which VLAN is enabled can include VLAN tags. when a packet enters into the network part in which VLAN is configured, a tag is added to show the VLAN membership. VLAN ensures the isolation between network traffic of all VMs.Figure 4.8 also shows the network configuration of all three VMs.

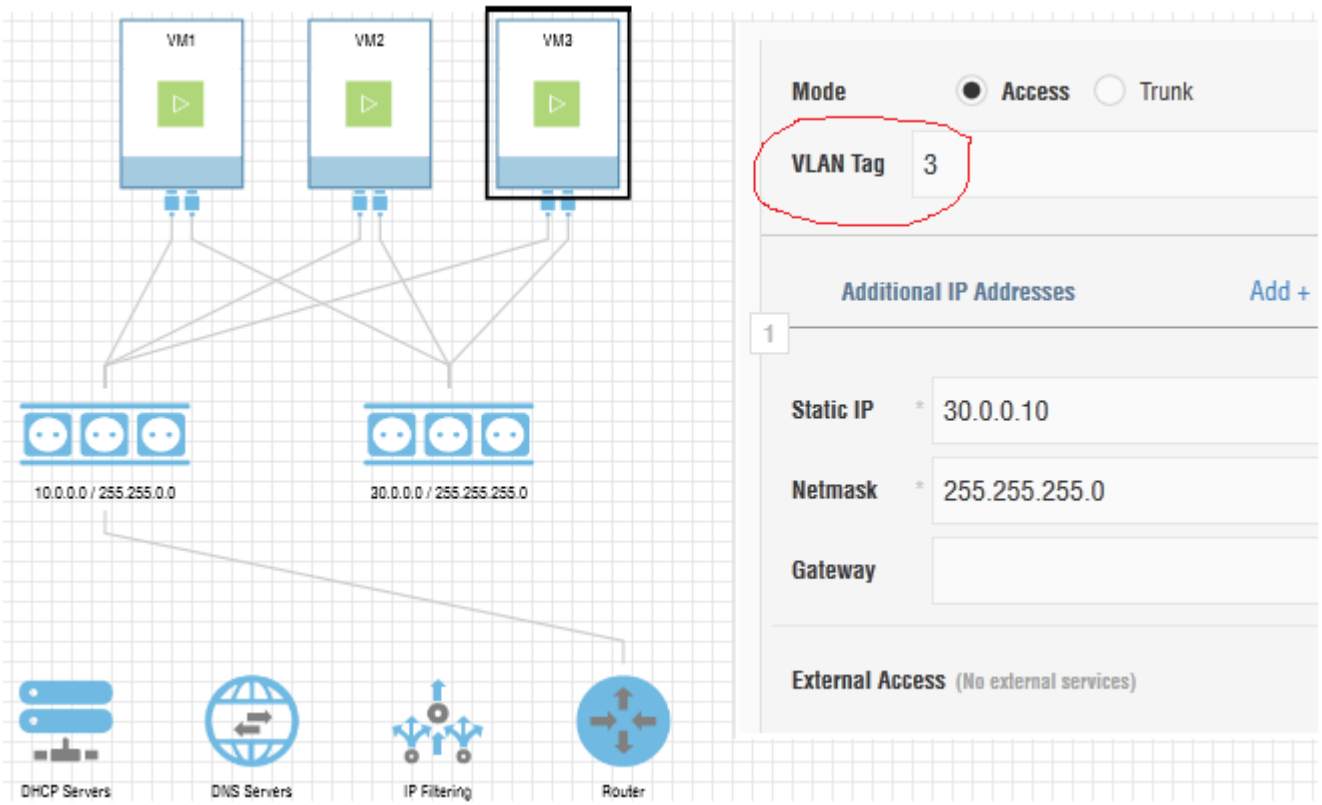

Figure 4.8: VLAN Configuration of VMs in Oracle Ravello System

4.6.3 Aalysis of Result

As shown in Figure 4.8, the attacking VM is able to successfully observe the network traffic between target VMs that are communicating through ping command. In this Figure 4.8,the

attacking VM is getting the ICMP echo request and reply which shows that both target VMs are communicating with each other. As mentioned, *ICMP* is the protocol running behind ping command and echo *request* and *reply* shows that both VMs are talking to each other. VM3 which is an attacking VM having no access rights to monitor the communication between other VMs on network channel.

```
IP ip-192-168-0-3.ec2.internal > ip-192-168-0-5.ec2.internal: ICMP echo request, id 19973, seq 85, length 64
IP ip-192-168-0-5.ec2.internal > ip-192-168-0-3.ec2.internal: ICMP echo reply, id 19973, seq 85, length 64
IP ip-192-168-0-5.ec2.internal > ip-192-168-0-3.ec2.internal: ICMP echo request, id 20741, seq 111, length 64
IP ip-192-168-0-3.ec2.internal > ip-192-168-0-5.ec2.internal: ICMP echo reply, id 20741, seq 111, length 64
IP ip-192-168-0-3.ec2.internal > ip-192-168-0-5.ec2.internal: ICMP echo request, id 19973, seq 86, length 64
IP ip-192-168-0-5.ec2.internal > ip-192-168-0-3.ec2.internal: ICMP echo reply, id 19973, seq 86, length 64
IP ip-192-168-0-5.ec2.internal > ip-192-168-0-3.ec2.internal: ICMP echo request, id 20741, seq 112, length 64
IP ip-192-168-0-3.ec2.internal > ip-192-168-0-5.ec2.internal: ICMP echo reply, id 20741, seq 112, length 64
IP ip-192-168-0-3.ec2.internal > ip-192-168-0-5.ec2.internal: ICMP echo request, id 19973, seq 87, length 64
IP ip-192-168-0-5.ec2.internal > ip-192-168-0-3.ec2.internal: ICMP echo reply, id 19973, seq 87, length 64
IP ip-192-168-0-5.ec2.internal > ip-192-168-0-3.ec2.internal: ICMP echo request, id 20741, seq 113, length 64
```

Figure 4.9: Traffic Capturing at Attacking VM in Oracle Ravello System

4.7 Limitations

The proposed approach works well but is limited to the selection of network model. Cloud provider normally uses Openvswitch (OVS) for the implementation of advance network setting. The case study used in our experiment describes a state-of-the-art configuration of the Open Stack networking feature by applying the VLAN and ML2 plugin in OVS. This attack only works on neutron network facilitated by OVS which can support the different advance features of network design and security. It does not work on legacy network i.e. nova-network. The latter model has some limitations, i.e., it can't facilitate the designing of an advance network architecture. Nova-networking was the only network solution prior to the implementation of Neutron in OpenStack. Nova-network only support FLAT network and DHCP services. It is the part of OpenStack till now, but it seems to be obsolete because of its limitations. Normally Flat network or DHCP followed the same model. The main concept is that all the VMs are attached to the bridge i.e Linux bridge. The bridge is attached to physical host where there is a support of physical NIC i.e. eth0. Multiple VMs are connected to this bridge. This concept model is shown in Figure 4.10.

Table 4.4 depicts the different cloud providers vulnerable to this attack.

4.8 Inhibiting the Network-Channel Attack

There are several possibilities for promising defences against cross-VM network-channels, whose pros and cons are discussed here.

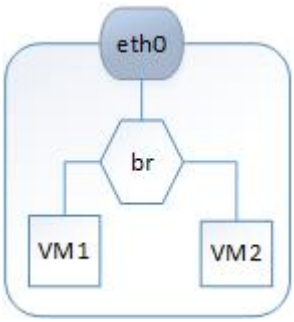

Figure 4.10: Nova-Network

Table 4.4: An Overview of the Vulnerability of Different Cloud Systems to the Proposed Approach

Cloud Provider	**Vulnerable to attack**	**Reason**
OpenStack	Yes	Allow bridging of a TAP interface that does not have a private Ethernet interface at backend.
Ravello System	Yes	Allow bridging of a TAP interface that does not have a private Ethernet interface at backend.
Microsoft Azure	No	Prohibited to connect a TAP interface with bridge having no Ethernet at backend.
Google CE	No	Prohibited to connect a TAP interface with bridge having no Ethernet at backend.

4.8.1 Avoiding Co-residency

Within secure environments, a longstanding strategy is simply not to use the same hardware to execute the tasks that should be separated from each other, i.e., to keep a time gap between the tasks. This strategy provides the best high-assurance defence against side-channel (and many other) attacks. However, this strategy would avoid many of the existing and upcoming practices of VMs, including public clouds that multiplex physical servers such as Amazon EC2, Windows Azure, and Rackspace, and the other VM-powered applications discussed in the chapter 1 of this thesis.

4.8.2 Potential Attack Mitigation Strategies

There exist several relevant works that could be used to mitigate cross-VM attacks. Work from Zhang et al. [168] proposed to utilize side-channels as a detector to identify illegal co-residency based on the timing channel (accessing L2 cache response time) while Afoulk et al. [184] attempt to avoid conflict of interest among VMs via priority based scheduling. Another approach would be to directly modify the open source code of OpenStack. It limits the granularity of network based side-channels and penetration of any external device into the current running system. Multiple VM interfaces connect to the OVS internal network bridge, i.e br-int that further connects with

the physical device and external network. The attacker itself in the internal network through impersonating of the TAP interface which does not have a private Ethernet interface.

Figure 4.11 shows the output of OVS that displays the connectivity of different virtual Interfaces.

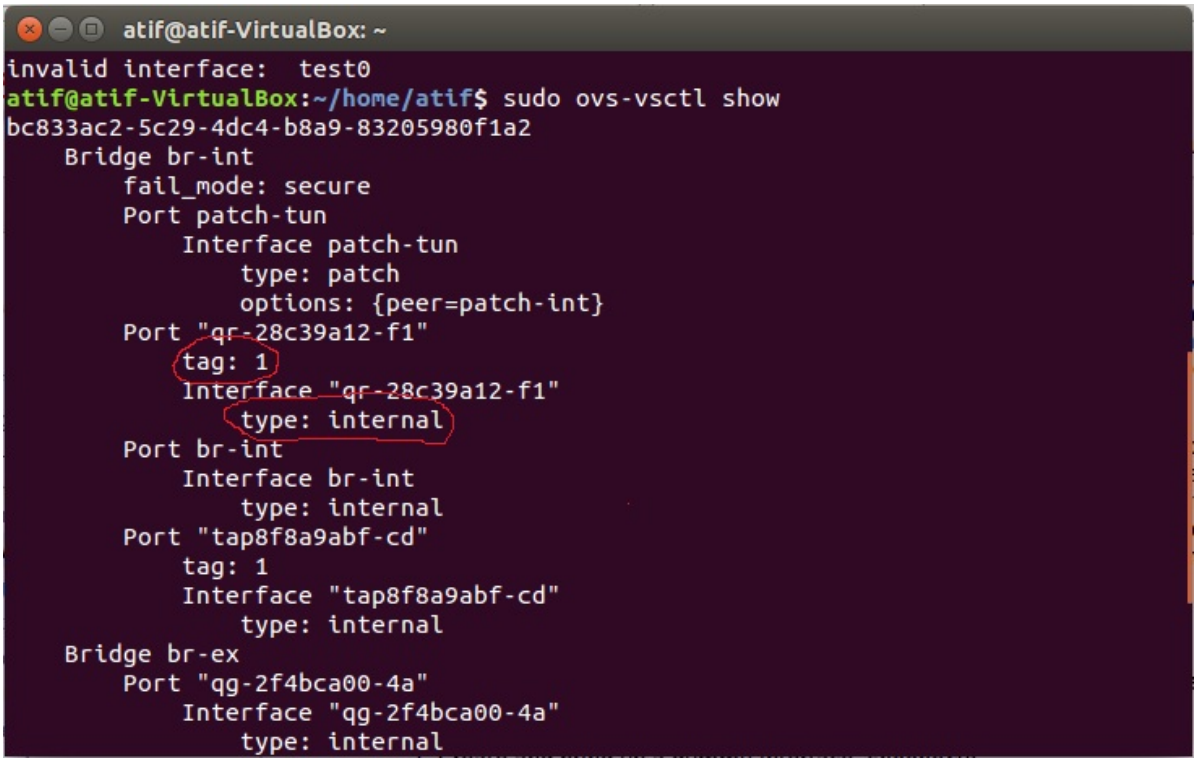

Figure 4.11: Connectivity of Virtual Interfaces at OVS

After deep analysis of the output, it has been observed that each valid interfaces have two attributes: tag and type. 'Tag' associated with Interfaces describes that VLAN is enabled in order to ensure the isolation between VMs and 'type' shows the behaviour of the interfaces. Figure 4.12 shows the attachment of dummy interface named as 'test0'.

One can see that there is no attribute of this dummy interface i.e 'tag' and 'type' both attributes are missing in the output of OpenvSwitch.

As OpenStack code is an open source, it is identified through close monitoring of OpenStack code that the security weakness resides within the networking (neutron) component of OpenStack cloud. As a result, there is a need to amend in neutron code within the class */opt/stack/neutron/agent/impl_vsctl.py* that includes a method which executes virtual switch operations *def run_vsctl(self,args)*. A new method has been included *get_all_bridges(self,args)* that collects all the information of connected interfaces at the bridge. The purpose of this function is to determine whether the connected interface can communicate using only the TAP interface,

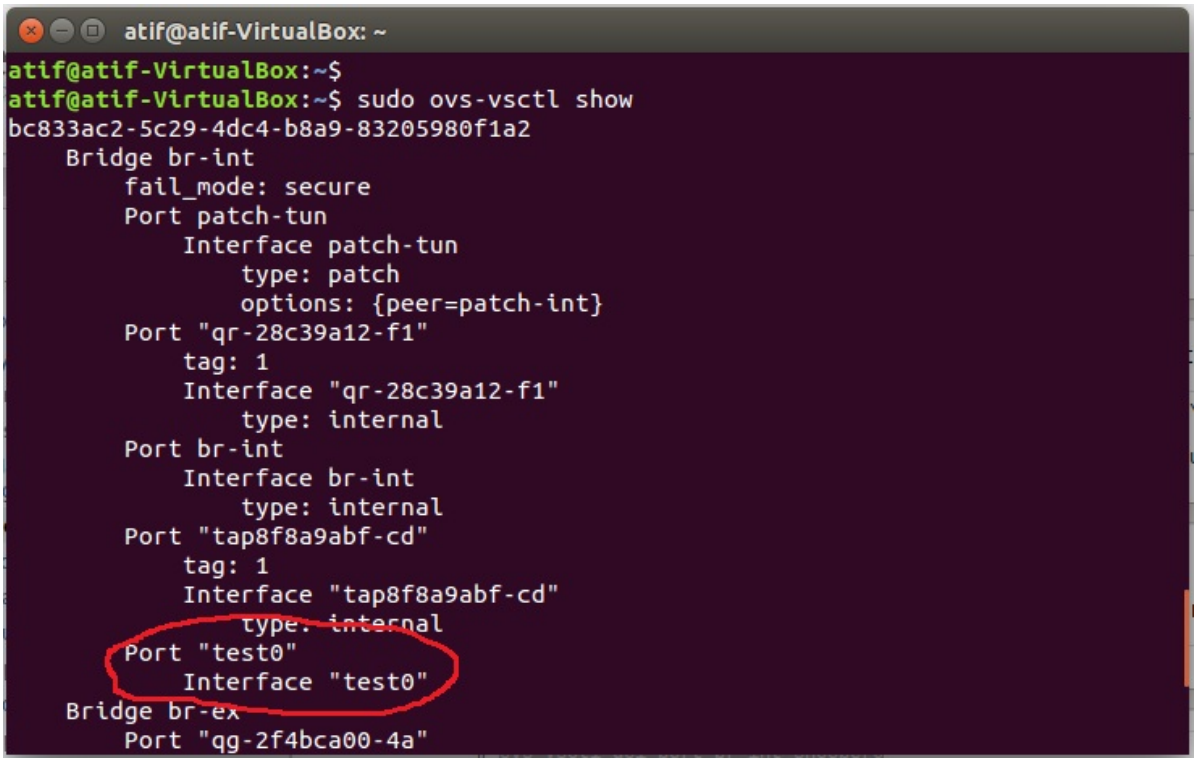

Figure 4.12: Attachment of Dummy Interface

and if capable, block direct connection of all TAP interfaces with an OVS bridge accessing the Internet with the same Ethernet. Analysis of the interface reveals that each valid interface has three attributes: tag, interface and type. 'Tag' describes that the VLAN is enabled to ensure VM isolation, 'interface' shows its association with back-end private Ethernet and 'type' reveals interface behavior. The security checks need to ensure all these 'TAP' attributes before connecting to the bridge.

After the amendment in a neutron code, the attempts to attach a dummy virtual interface with OpenVSwitch (OVS), OpenStack cloud creates an error by displaying that it is an invalid interface as shown in Figure 4.13.

4.9 Discussion

This section discusses the major findings from the empirical evaluation of TAP impersonation attack presented in this chapter.

- TAP Impersonation and the mirroring attack were implemented by the exploitation of the network channel. TAP Impersonation was the first step to penetrate into the network

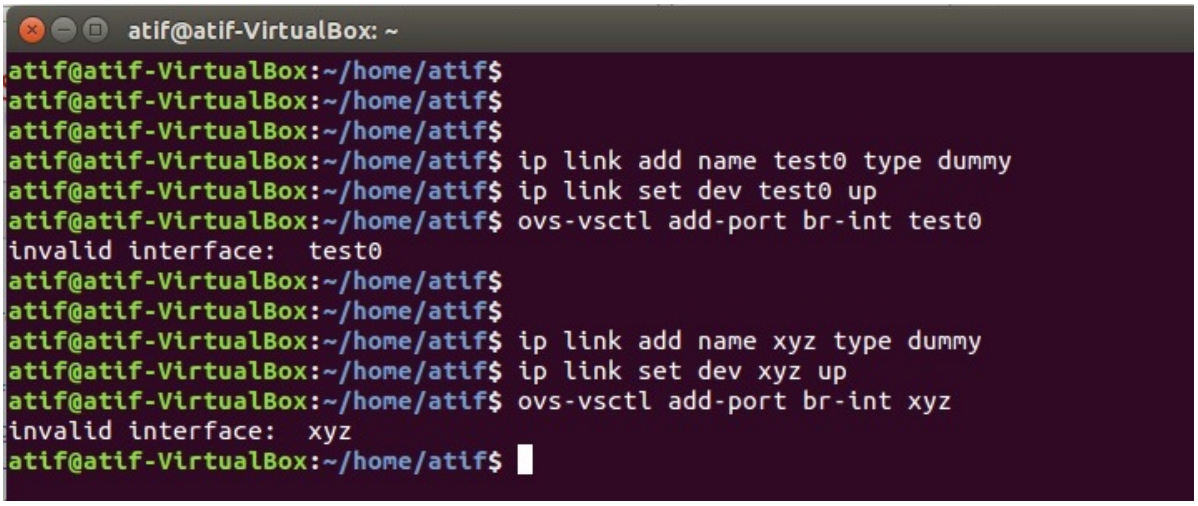

Figure 4.13: Blockage of Invalid Interface

system. Mirror was configured at the central bridge to redirect the network traffic to set hidden destination point. The bridge was focused as VMs network traffic are passed through it. The results showed that the presented attack successfully redirects the network traffic of co-located VMs which indicates the exploitation of cloud network architecture.

- It was observed that the presented attack redirects high rate network traffic of co-located VMs at hidden destination point which can be seen from the network traffic graph. PRTG was found to be the most appropriate network traffic graph monitoring technique for measuring the network bandwidth of each VM in real-time. Sudden spikes were observed in the graph which indicates the success of the attack.

- It was interesting to examine that this attack setting can be blocked for exploitation. The attacker can penetrate into the network system by impersonating the TAP interface. By carefully analysis the attribute of real-TAP, it has been observed that impersonated TAP has different attributes which can be blocked through modification in source code of open source cloud.

- Evaluation criteria (Section 4.3) has been satisfied via:

  Functional qualitative parameter, the attack methodology is evaluated to check whether it accomplishes all the tasks for which it is designed. The functional parameter is evaluated by looking at its inference ability as shown in Figure 4.3 in which all these attack strategies have been implemented. Moreover, the results showed that all the events were inferred successfully. The evaluation showed that the attack model fulfilled the functional purpose.

  Expressiveness, in the context of this thesis, is defined as how well the TAP Impersonation attack works the full range of tasks when used within the network domain. The

expressive parameter is evaluated by looking at the results in Figure 4.4 which provides the network traffic capturing of other co-located VMs that are the strong evidence about the working of attack.

From the evaluation of the results, it is demonstrated that the major strength of the presented approach is its ability to redirect the network traffic of other co-located VMs. However, the presented approach suffered from some limitations that are described in Section 4.7.

4.10 Summary

In summary and compared to work reported in the literature on cross-VM attack, the chapter presents:

Details of experiment, their setup, configuration, limitations and mitigation strategy in order to evaluate this attack in a real-time scenario. Launching of cross-VM network channel attack within an open source cloud platform i.e OpenStack is also presented. Divide and conquer methodology has been used to explore the security flaw in the cloud model. Attack settings and challenges have also been elaborated to show how challenging is for attacker to penetrate into the network system and how to circumvent the security perimeter of the cloud system. VM's configuration has been discussed to show the VM's properties and settings.

The presented attack uses a combination of impersonating a TAP interface and constructing a network mirror in the bridge interface, where network traffic of all co-located VMs passes. The experiment is designed to obtain ground truth to monitor the network traffic of co-located VMs by exploiting the underlying network channel. The empirical evaluation shows that the attackers successfully redirect the network traffic of co-located VMs by using the attack strategy. It is very challenging for cloud providers to detect and observe such attacks because attacking VM does not violate assigned VM resource capacity. Countermeasure solution has also been presented to block this type of attack by fixing the loophole in the code of OpenStack.

The next chapter provides a complete description of our next finding through experiments on different setups and their evaluation, derived from the analysis of OpenStack Xen architecture.

CHAPTER 5

PRIVILEGE ESCALATION ATTACK AND THEIR COUNTERMEASURE

As discussed in chapter 2, the researchers have demonstrated different approaches for memory exploitation and explored vulnerabilities in kernel code. This exploitation in turn demands placement of hardening systems to inhibit privilege escalation attacks. As there is strict memory isolation mechanisms between the kernel and user space, like Intel's SMEP, attackers gradually depend upon code reuse methods to exploit vulnerabilities especially those that are kernel related. Contrary to analogous attacks in more preventive configurations, such as web browsers, non-privileged limited opponents have great flexibility in exploiting memory disclosure vulnerabilities to dynamically determine, or conclude, the location of certain code segment and design code-reuse contents. Recent research has publicized that the connection of code variation with the implementation of a "read XOR execute" (R$\hat{\text{X}}$) memory security strategy is an operative defense against the misuse of user- land software, but so far this method has not been implemented for the defense of kernel itself [185].

The purpose of this study is to investigate the implication of hypervisor vulnerabilities in cloud computing. In this study, a novel attacking scheme has been unveiled on cross-VM cloud environment having Xen hypervisor running underneath. The core responsibility of the hypervisor is to ensure proper isolation between domains, i.e root and non-root domain. The proposed approach exploit return-oriented programming (ROP) model. An adversary having limited privilege reside on the same physical machine where target VM is, may launch ROP, establish a connection with root domain by abusing the network channel and get the possession of tool stack which they are not authorize to access it directly. ROP technique has been investigated in cross-VM settings, in which a malicious VM reuse the existing code that is already stored in the Kernel memory and don't modify or inject any external program for its execution. Through

this technique, the attackers are successful in exploiting the integrity protections. The findings indicate that the proposed attack is feasible in contemporary public clouds i.e OpenStack, Azure where Xen para-virtualization is configured underneath. In conjunction with our findings, a countermeasure solution has also been proposed to the described attack. This indicates that the offered cross-VM attack is of applied significance in cloud platform. The results of this research illustrates that there is a serious security threat to leading cloud providers.

5.1 Introduction

A myriad of previous research have demonstrated how hypervisor are used to enhance the security of VMs running on it [33, 186, 187, 187–191]. Their main theory is that the hypervisor is a software that consists of a small piece of code as compared to operating systems therefore it can comprehensively be examined, as a result its reduces bugs on its code. But modern services of hypervisors such as Xen, VMware are complicated and they are huge softwares having large lines of codes, which falsify the hypothesis i.e Xen 4.1.2 has around 350K line of source code in hypervisor itself. The best formal authentication technology today can only deal with 10K lines of source code [103]. It is not favorable to formally vet such a large code of hypervisor. Additionally, National Vulnerability Database [192] list down the latest security breaches in hypervisors software by expressing that it is not simple to present bug-free hypervisor software. These exposures make hypervisors the target of attackers [193–195].

The main objective of software attack is to distract the running program flow and execute some other instructions. The term *software attack* is generally used in different types of applications ranging from desktop to eeb to simple program such as notepad. A more detailed form of *software attack* is the exploitation of memory vulnerability that appears in many kinds such as buffer overflow, heap overflow and string vulnerability [196]. Buffer Overflow vulnerability [197] is the most serious threat in which an attacker can overflow buffers in the stack process and overwrite the return address of a function with random memory address. The attacker can take advantages of such overwriting by executing any code when the vulnerable function returns. Researchers has executed this attack in many different forms and their mitigation solution has also been proposed in numerous forms. Their mitigation solutions avoid code execution that is present in the data regions of the process, for example in stack or/and heap. Contemporary operating systems includes Data Execution Prevention (DEP) using the security model [198]. In these models, the memory location cannot perform both tasks simultaneously, it can either be executable or writable. By doing so, the attacker cannot write and execute code at the same address of stack /heap.

Despite the deployment of such countermeasure strategies, attackers still are able to find

the gaps to execute proposed code by re-using of already existing code in the process address space instead of injecting new code. A well-known category of this attack is the return-into-libc attack, designed by Solar [199]. In this attack, the attacker overwrites the return address of a vulnerable function with the address of any function in the libc library. For example, the attacker can overwrite with the address of a 'system' function through which a shell can be opened. A shell is a dominant backdoor which supports in executing multiple tasks. Return-Oriented Programming(ROP) is a precise form of the return-into-libc attack, in which the attacker attempts to implement parts of code dispersed through the process address space by connecting them with unintended transfer instructions, typically the 'ret' instruction.

As proposed by Shacham [15, 197], in ROP the attacker usually implements the attack in two phases:

1. In the first phase, the attacker detects the sequence of instructions suitable in performing the planned jobs. This short set of instructions is called a *gadget* and normally is from 2 to 5 set of instructions in length. However, there is no limitation on the gadgets' length and it has not been verified that it is in-feasible to have longer gadgets. Also the potential for the attacker to find more than one gadget is also possible.

2. In the second stage, the attacker connects these known gadgets (from the first step) together in such a way that they are executed successively.

   A overview of ROP exploit is shown in Figure 5.1

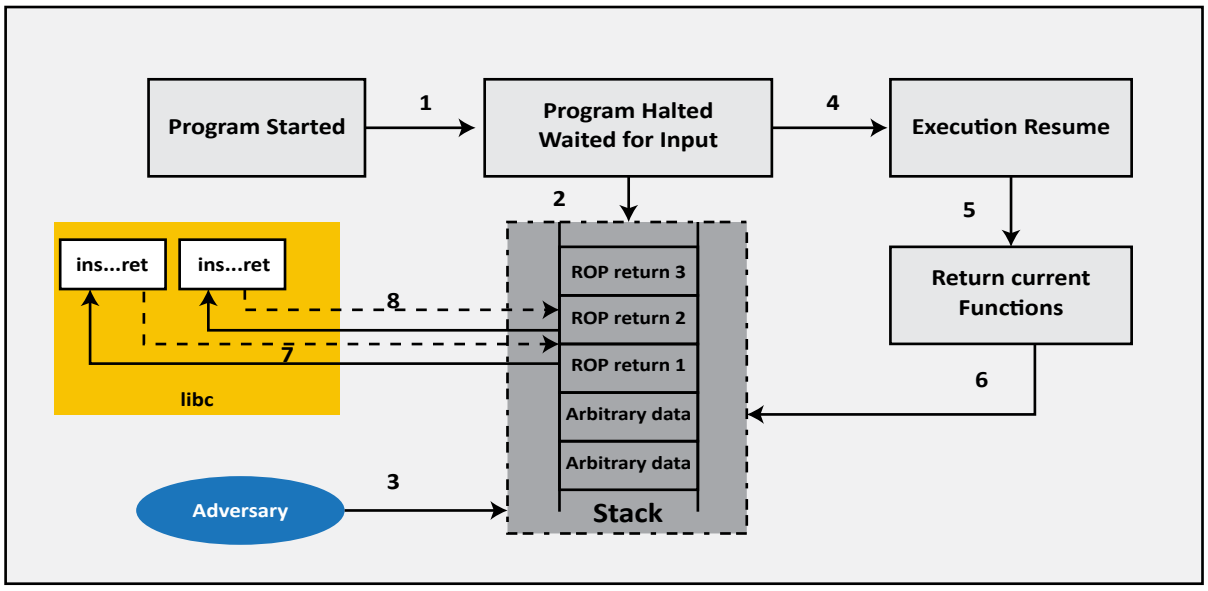

Figure 5.1: Overview of RoP

The attacker first discovers a vulnerable function in the process address space, either related to the application or a part of external or internal libraries. This vulnerable function would then be executed during the actual program execution. The attacker overwrite the stack data with the proposed return addresses (i.e the start addresses of the gadgets). The gadgets should end in unintended control transfer instruction, typically the 'ret' instruction assists this purpose. Every time the actual operation (gadget) is executed, the 'ret' instruction at the end of the gadget pops next value (that is the next gadget's address) in the stack, which becomes the new execution instruction. The attacker may also insert data values into the stack which can be used as arguments for the proposed instructions to be executed [197, 200–203].

The mentioned technique which is first recognized and described uses only the 'return' instruction for redirecting the control flow of the program and moves it to the proposed program's flow. But "return" instruction is not the only way of moving the control. There are some other control transfer instructions such as the indirect 'imp' and indirect 'call' instruction, which store the address of the target instructions. Therefore, the attacker now has the choice to use such instructions as well to redirect the flow and execute code that they propose. Checkoway.et.al [204] proposed Return-Oriented Programming use return like instructions rather than directly using the return instruction. For example, the instruction sequence 'pop' is similar to a *'ret'* instruction which pops the top of stack to register and executes the instructions at address.

In cloud computing the hypervisor is responsible for creating domain isolation between the root and non-root VMs. However, the hypervisor is a software that consists of multiple lines of code that are vulnerable to attack. Researchers have already presented numerous approaches in recent studies (as discussed in section 2.9.2) to attack the hypervisor in a way how an attacker using ROP to escalate the privilege level of non-root users, and how an attacker can inject an external program in system memory for arbitrary execution. In [210] researchers have explicitly shows how an attacker can alter the privilege value of non-root VM stored in memory for privilege escalation.

Despite the clear potential of privilege escalation attacks by abusing RoP and shared memory as discussed in Section 2.9, fine-grained privilege escalation attack that practices ROP in conjunction with the cross-VM network-channel attack has not yet been performed. The fact is that hypervisor has enhanced the security level by placing more layers of isolation through domains between VMs. VMs reside in their respective domains according to their privilege level. Due to which, cloud architecture does not seem to have a threat of such attacks because VMs having different privileges level and are resides at different domains.

Chapter 3 presented hypervisor architecture and their domain properties (in Section 3.3)

where the investigation is made to find the vulnerability in the state-of-the-art hypervisor. The presented finding indicates that there are some challenges that needs to circumvent to exploit these vulnerabilities. Their implementation has been presented in the next section in the leading open source cloud models OpenStack and Microsoft Azure along with the description of how to address these challenges to exploit the vulnerabilities found in Section 3.2.

In this chapter, an advance innovative attack methodology called privilege escalation has been presented. It abuses ROP in conjunction with network-channel to launch an attack in cross-VM settings and shed some light on the security, or lack thereof, in the multi-tenant infrastructure cloud environment. The main aim of this investigation is not to comprehensively study how vulnerable the cloud servers are, rather the isolation properties of cloud systems have been explored, as virtual machines and hypervisors can be circumvented by the proposed attack (and if so, how?), if the underlying hardware become vulnerable.

Problem Statement and Contribution As discussed earlier, Cloud providers place multiple VMs on the same hardware. Each VMs have been assigned privilege levels and hypervisor place them in their respective domains according to their privilege levels. But the attacker can escalate the privilege level of its VM by exploiting the domain isolation properties. However in cloud computing, strict security features are configured that protect from such exploitation. Such exploitations in real-time scenarios are becoming more challenging.

This chapter overcome these security challenges by investigating the RoP and their associated techniques. Moreover, designing a zero-day attack model (as mentioned in Section 1.3.1, research goal 2), mainly focusing on the key components of the underlying hypervisor architecture, the technique and the way domains are connected, processed and applied for the overall model is also presented in this chapter. The main concept of the proposed attack model is validated through the implementation of real-time systems for OpenStack and Oracle Ravello. Experimental series was conducted to evaluate the effectiveness of the proposed attack model. This aimed to prove the practicality of implementing the system in an operational context. In the end, defense strategy (as mentioned in Section 1.3.1, research goal 3) for controlling such attacks is also presented.

5.1.1 Technical Challenges

Development and application of a cross-VM privilege escalation attack are presented in this study. All VMs have been assigned different privilege level in cloud environment. The 'privilege escalation attack' is the attack that escalate the privilege level of unprivileged VMs [195]. In cloud environment, one virtual machine normally declared as a privileged VM reside in dom0 and there are multiple unprivileged VMs in domU. Hypervisor ensures isolation between all such VMs by placing them in different domains. Like many attacks, the proposed attack is a

privilege escalation attack in which the attacker VM escalate its privilege level by exploiting the ROP technique and abuses the network channel to establish a connection with root domain. However, many of the phases performed to achieve the proposed attack effectively and with high precision in a virtualized environment are novel in this research. In particular, an account has been provided of how to overcome the significant challenges of privilege escalation and connection with root domain. This attack is so influential that if an attacker is successful in escalating the privilege level of his VM, he can destroy other VMs residing on the same physical machine. This attack can also cause a DOS attack on the physical machine by creating multiple malicious VMs that unnecessary engaged physical resources.

5.2 Attack Scenario

In this section, the capability of a malicious VM is demonstrted that exploit ROP technique to escalate privilege level by abusing the network channel. It has been shown explicitly that after gaining the privilege rights, it enables an attacker to take over the control of Toolstack through which it can manage the other co-resided VMs. Controlling Toolstack is not only pretty valuable to attackers but also serious threat to other VMs and cloud providers. The introduction of many novel applications to execute this attack has been presented: penetration into the root domain, illegal possession of Toolstack and the usage of ROP in conjunction with network channel. All these steps have been practically performed on the running OpenStack cloud testbed.

5.2.1 The Attack

There is a potential to take over the control of Toolstack in hypervisor at dom0 through the attacking machine. Toolstack is responsible for operating other VMs co-resided on the same physical machine. The main focus of this research is to circumvent the security perimeter of OpenStack by introducing the concept of code RoP and then to exploit the network channel. By compromising the network channel, the attacker manage to penetrate into the domain i.e dom0 of running system where the privileges VMs resides. The network connections through which other cloud systems can be connected or OpenStack allow its expansion are needed to be throughly investigated.

In OpenStack cloud having Xen architecture running underneath, the root or admin VMs resides at dom0 and are able to access the underlying hardware directly. These root/admin VMs have special privileges i.e. they are able to control the other VMs. Dom0 is the first machine started by the system. There is a special stack (called as Toolstack) in this domain that allows a VM to manage virtual machine creation, destruction and resource allocation. Guest VMs reside in domU, an unprivileged domain, can't directly access the hardware. These guest VMs can't interfere in other guests' operations and management. 1) In an attacking guest VM which is in

domU, a special feature of OpenStack named as OpenStack client has been configured. The main concept behind the configuration of this special feature of OpenStack is that through this feature, a malicious VM copied the code of OpenStack in its own domain i.e domU. By doing so, the special technique of RoP i.e code-reuse has been applied in its internal memory. This in-memory code is already testified and scrutinized by security perimeter of OpenStack and is already in running form. Hence security perimeters are unable to block it. The main benefit of code-reusing scheme in guest VM is that this malicious guest VM becomes a host of its own local machine (sub-host from main root). After being a host of its own local machine, it has been granted a "dom0" for managing its own sub-guest VMs. This model looks like a tree in which there is one main root that resides in dom0 and that have further guest VMs that belongs to domU. Among these guest VMs, one guest VM is acting as a root of its local system which has its own guests VMs, but globally it acts as a sub-root.

This malicious guest VMs avail both the privileges (dom0 and domU). Dom0 to manage its own local guests VMs which can be further subleted and domU which is added in his account by its inherent characteristic. There is a special python API in dom0 i.e xapi that make a connection between other domains through network bridge i.e. br0. In order to make a connection between domU and dom0, a special python API i.e XenAPI at domU is used to connect with xapi of dom0. This connection follows a network path from Ethernet device to bridge and it uses internal management network to reach from domU to dom0.

2) In proposed attack scenario, an adversary established a connection with dom0 by connecting veth bridge pair that connects xapi with xapi of python library. In veth bridge pair, one bridge already exists in main root and other bridge is the part of the malicious guest VM that he acquires by applying code reusing technique. 3) Whenever, xapi connects with other xapi through bridge, it presumes that root of one cloud is attempting to connect with root of other cloud for cloud expansion or for hybrid cloud and it automatically allows the connection. Due to this, the attacking machine is penetrates into dom0, gain the root privileges and takes over the control of tool stack through which thye can manage other guest VMs. After gaining control of Toolstack the attacker is not only able to manage the other guest VMs running on same physical machine but can also cause DoS attack by creating unnecessary VMs and assigning them huge amount of resources. Figure 5.2 depicts the concept of attack model on OpenStack Xen architecture.

5.3 Evaluation Criteria

From Figure 5.2, it can be seen that evaluation revolves around the attack methodology discussed in Section 5.2.1. The main goal of the attack is to escalate the privilege level of non-root VMs so it can perform the operation of root VM by controlling tool stack from where it can manage other

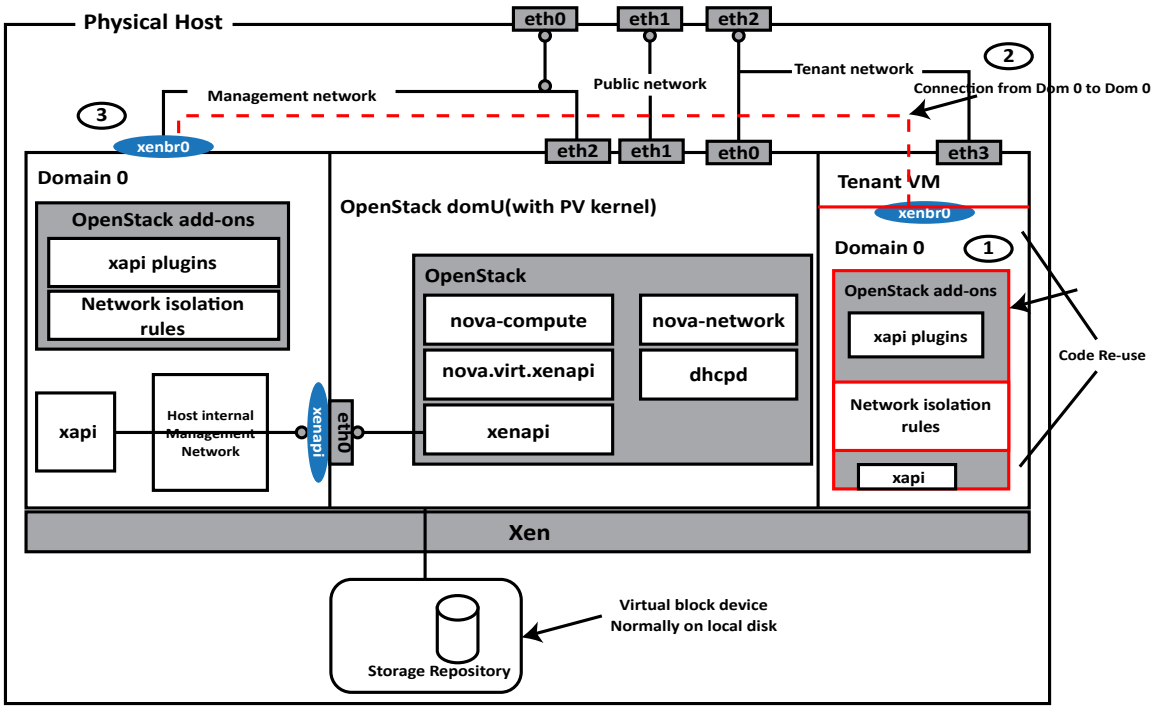

Figure 5.2: Attack Model on OpenStack Xen Architecture

co-located VMs. To achieve this objective, the attack strategies discussed in section 5.2.1, are implemented to evaluate the result. The evaluation of the results is mainly dependent on ROP.

The evaluation criteria are based on the following key reflective qualitative aspects. *(i) Functional* shows the accomplishment of the above said five tasks that are important for the execution of attack.*(ii) Expressive* indicates how well does the attack methodology does the job. (iii) Finally, the overall strengths and limitations of the attack have been shown.

5.4 Evaluation

The experimentation on attack described in chapter 5 is conducted in leading open source IAAS cloud platform i.e OpenStack and commercial cloud system Microsoft Azure System. Following configurations have been implemented on the system.

5.4.1 Experiment setup

OpenStack Networking supports all advance networking features for the Virtual Networking Infrastructure (VNI) and the access layer phases of the Physical Networking Infrastructure (PNI). It activates VMs to configure advance topologies of virtual network which may entails features such as a firewall, load balancer, and virtual private network (VPN). Networking facilitates some

other features like networks, subnets, and routers. Each features have different functionality that act like its physical counterpart: networks consist of subnets, and routers route traffic between different subnets and networks. Any configured networking set up has at least one external network and one or more internal networks. VMs are directly connected to these software-defined networks. Networking routers are needed to access VMs that are outside the network. Each router has two interfaces; one is connected to an external network and other for an internal network. Similar to physical router, subnets can be used to access machines on other subnets.

5.4.2 Security Configuration

Security is a strong feature of cloud computing. OpenStack networking supports security groups that enable administrators to set firewall rules in groups. It is a combination of different security group rules which specify the network access rules. VMs can belong to one or more security groups. The main role of security groups are to block or unblock ports, define port ranges or define network traffic types, their routes for VMs. Networking also supports security groups. All configured Network setups apply a security group plug-ins to enhance the security feature of OpenStack networking.

5.4.3 Attack Setup

In this section, the progress on the use of ROP i.e code reuse has been presented. Due to ROP, the attacker escalate the privilege rights of malicious VM then exploit the network channel as a means for launching the proposed attack. In this attack, the adversary's goal is to control the Toolstack in dom0 and manage other co-resided VMs. The connection with root (dom0) can then be used to penetrate into root domain (dom0) and control Toolstack through which the attacker can control victims. In [103], the attacker was supposed to use ROP technique to compromise hypervisor by editing its code to escalate the privilege level. In OpenStack cloud setup, the proposed attack has been launched by using ROP in conjunction with network channel. When VMs are running at real-time on cloud platform their complete management can be controlled through Toolstack in Xen architecture. The malicious VM which can manage to escalate the privilege level by circumventing the security perimeter, may be able to get the possession of Toolstack. If the attacker is successful in controlling the tool stack, they can not only control all other VMs running on the same platform but also launch DoS attack on physical machine.

5.4.4 Attack Scenario

Based on the attack described in chapter 5, the evaluation for effectiveness of the proposed attack on the designed architecture of OpenStack setup is explained here. Two VMs have been launched to evaluate the attack. One can launch more VMs upon requirement. After the launching of VMs, next step is to configure an Operating Systems (OS) on all these VMs. It's a choice of client to

configure any type of Operating System in its VMs. For testing purpose, different Operating systems have been configured on all VMs. First VM is configured with Ubuntu 15.10, Window is loaded on second VM. All two VMs are configured with two types of IPs, i.e floating and private. Floating IP is used to access the external world, i.e the Internet. However, private IP is used for the internal communication between VMs. All these VMs are assigned to different network segmentation to ensure logical isolation between them.

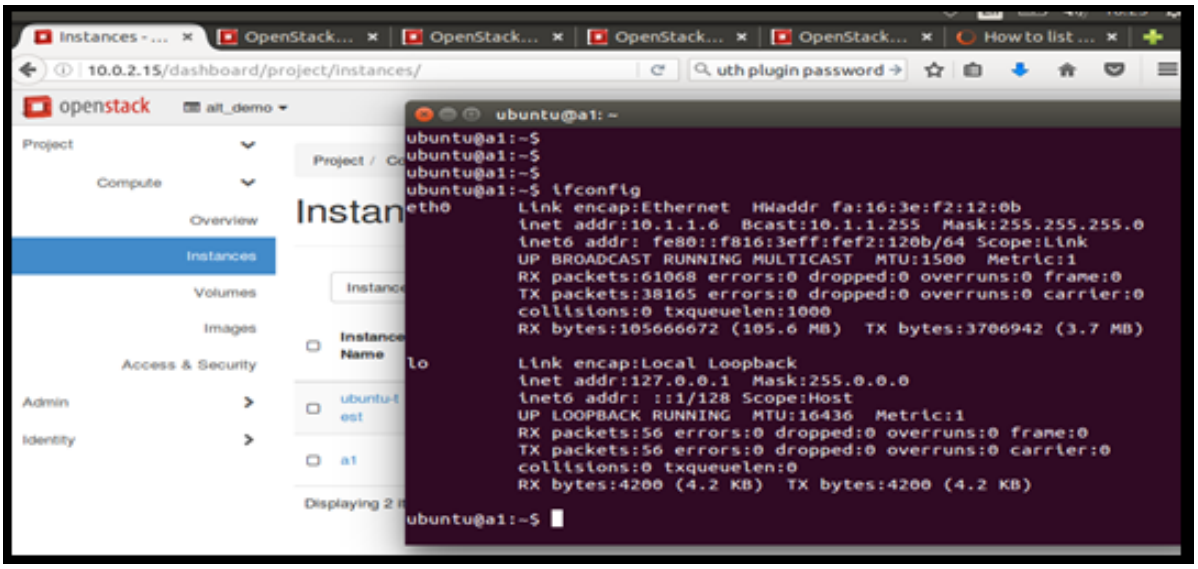

Figure 5.3: Attacking Machine Console

5.5 Analysis Results

This section describes the in-depth evaluation of the experimental results of the overall approach through the real-world scenario in terms of the exploitation of domains by hijacking the ToolStack and physical resources utilization by VMs. The effectiveness of the proposed technique is evaluated using the following metrics 1) root operation performed by non-root VM and 2) hardware resource utilization of the attacking VM before and after the attack.

NoN-Root VM Performing Root Operations To validate the attack strategy, a collection of root commands generated by the (attacker) non-root VM from the testbed is analysed. Each experimental run shows these commands that are labeled with the ground truth regarding the presence of attack.

Figure 5.3 shows the login screen of an attacking machine (a1) by using ssh command that demands key pair which was configured during instance creation. Attacking VM after applying

the attack scheme, not only manage to escalate its privilege level but also establish a connection with root domain i.e dom0. After successful connection, the attacking VM hijack a Toolstack through which they may be able to add or delete the target VMs as shown in Figure 5.4.

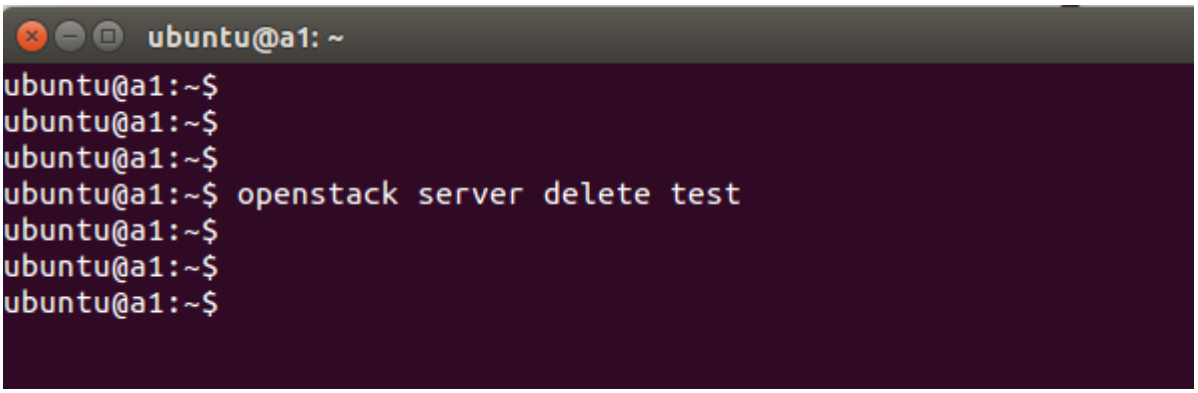

Figure 5.4: Co-located VM Deletion Command

An attacking machine after controlling the Toolstack can also launch DoS attack on physical machine by creating an unnecessary VMs and reserve a large amount of resources. After running the attack, status of test instance on OpenStack server can be seen in Figure 5.5. The instance list after the deletion operation is shown in Figure 5.6.

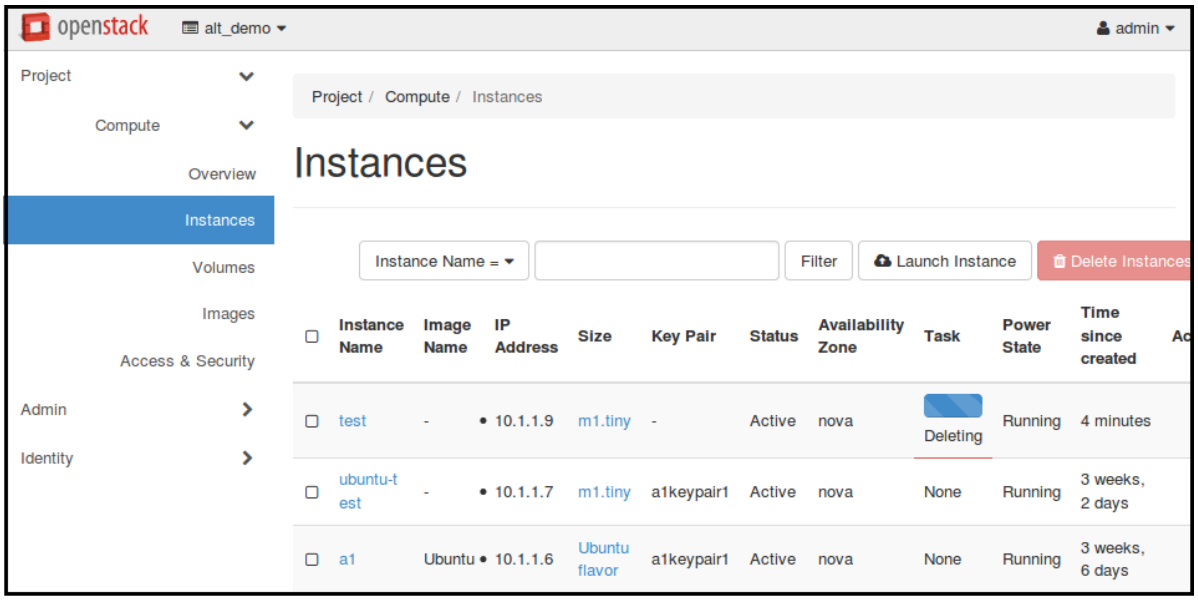

Figure 5.5: Co-located VM Deletion

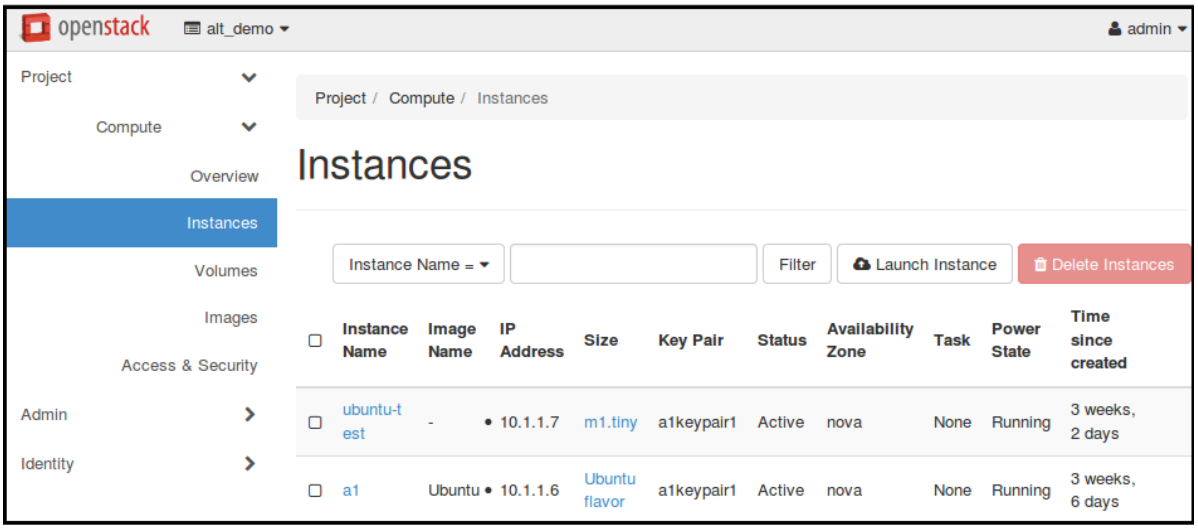

Figure 5.6: VMs List After Deletion

Resource Allocation Table 5.1 shows the total resource allocation of all co-located VMs. However, one-week resource utilization statistics of each VM has been tabulated in table 5.2. The value of resource utilization specified in table 5.2 are not fixed; it varies from time to time depending upon the VM utilization and the programs running on them. After the execution of privilege escalation attack, a runtime measurement has been performed to observe the resource utilization of attacking VMs.

Table 5.1: Resource Allocation of Each VM.

Co-located VMs	Memory (MB)	Disk (GB)	CPU (Hrs)
VM1	745.8482	6.25	709.45
VM2	937.6591	9.13	1054.764
VM3	3109.675	17.45	989.243

Table 5.2: One Week Resource Utilization of Each VM.

Co-located VMs	Memory (MB)	Disk (GB)	CPU (Hrs)
VM1	1024	15	1
VM2	2048	25	2
VM3	3072	40	3

Figure 5.7 compares resource utilization of attacking VM before and after the implementation of attack. Dash lines shows the resource utilization before attack. The main reason behind the variation in resource utilization is that attacking VM uses ROP in order to escalate its privilege level, hence it puts more processing load by executing the same set of code in its local memory.

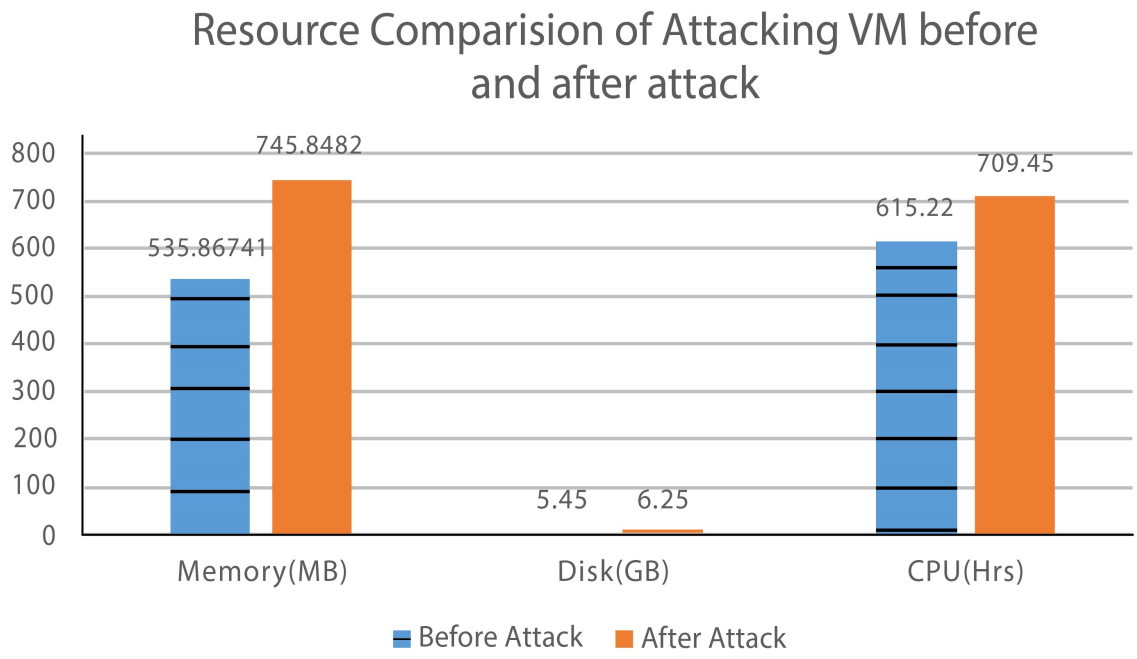

Figure 5.7: Resource Comparison of Attacking VM Before and After Attack

5.6 Attack on Microsoft Azure

The same attack could be evaluated on Azure System, Microsoft's cross-cloud platform. Azure is one of the leading public clouds. In experimental testbed, Azure is configured to be the underlying IaaS provider. Azure offers unique cloud application hypervisor technology facilitating an enterprise and individuals to encapsulate completely and abstract an entire multi-VM application and its environment so that it can run on any cloud (public or private) without any modifications.

5.6.1 Network Configuration

Azure allows to set up the network configuration for each of the VMs including static IPs. One needs to assign the Static IP, Netmask, Gateway and DNS server for this interface. Azure support the feature of SDN that automatically creates a virtual switch based on the IP address assigned to VM and netmask of the interfaces. All interfaces that have same subnet are assigned to the same virtual switch. If a gateway has been assigned at user interface, then its SDN will follow the following properties:

1. The guest VM has assigned a Gateway IP (i.e one has configured the VM to serve as a

Router), then Azure's SDN will be working in accordance with the VM's Gateway setting.

2. If in a case, the guest VM does not have a Gateway IP, Azure's SDN will add a virtual Router with the defined Gateway IP, and add it to the virtual switch.

3. If a default Gateway is not defined on Azure's user interface, Azure's SDN will simply do nothing.

For this experiment, a simple network topology environment has been setup, by creating two VMs say VM1, VM2 . Two NICs have been configured in each VMs and are separated from each other. One VM (VM1) is root that resides in dom0, VM2 is an unprivileged VM and is residing in domU. VM2 is an attacking VM, which is able to circumvent the security perimeter of cloud and establish a connection with root domain i.e dom0 by exploiting a network channel as described in section 5.2.1. VM3 can manage to control the Toolstack through which he may be able to add or delete the target VMs.

An account has been created in Microsoft Azure. For testing purpose VM2 has been created in MS-Azure. VM2 is the main working domain, where the experimentation of proposed attack has been performed. Next step is to ssh into VM2. Now, two further sub-virtual machines are created in the root account (VM2) say 'MyLinuxVM' and 'MyWinVM' as shown in Figure 5.8. Next step is to login into one sub-VM "myLinuxVM" which is an attacking VM through ssh command.

```
atif@VM2:~$ az vm list --output table
Name       ResourceGroup    Location
---------  ---------------  ----------
MyLinuxVM  MYRESOURCEGROUP  westus2
MyWinVM    MYRESOURCEGROUP  westus2
atif@VM2:~$ ~
```

Figure 5.8: Sub VM List

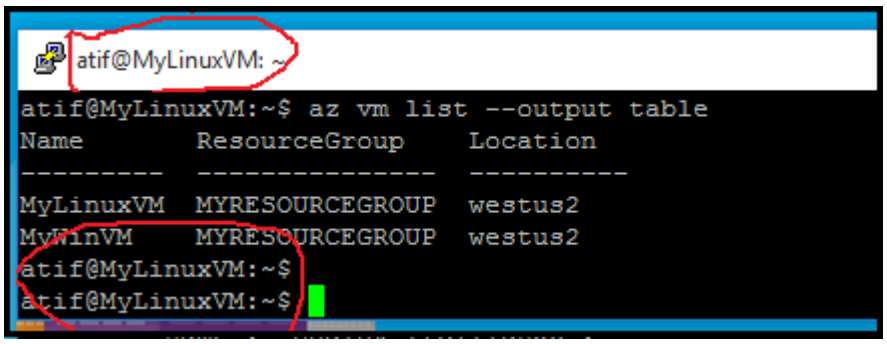

Figure 5.9: SSH to Sub-VM(attacking)

The attacking machine, after implementing the attack methodology, is now running the delete command on other co-located VMs as shown in Figure 5.10

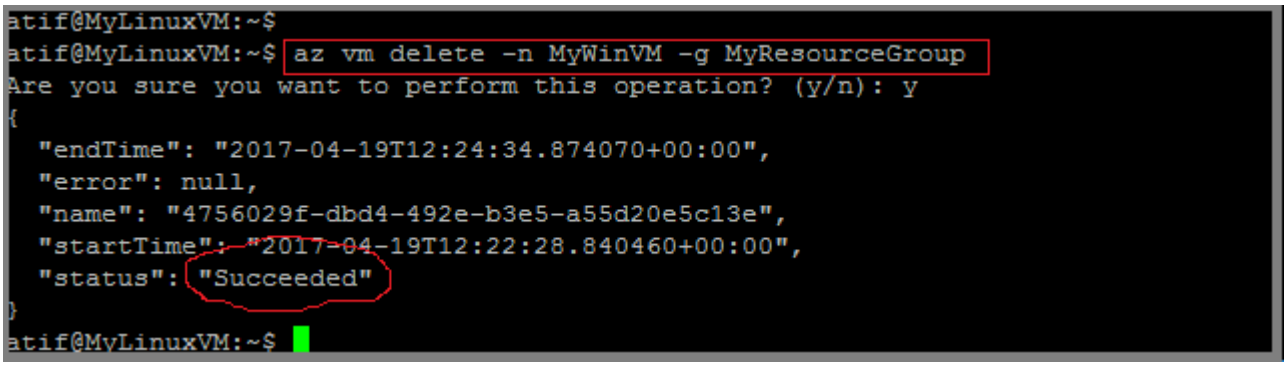

Figure 5.10: Deleting Co-located VMs

Figure 5.10 depicts that non-root user is applying 'delete command' that is under the privilege of root user. In conclusion, the complete list of all co-located VMs are shown in Figure 5.11

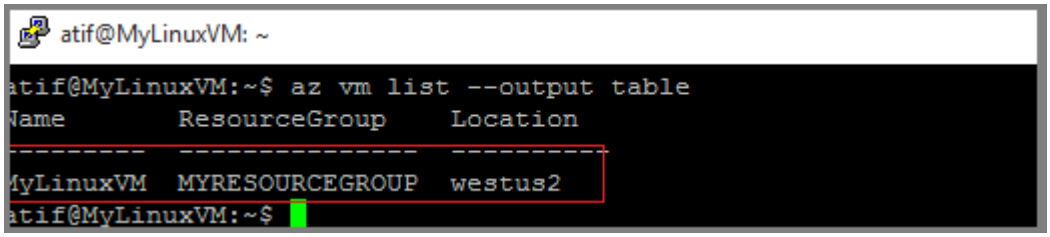

Figure 5.11: List of All VMs After Attack

A summary has been tabulated in table 5.3 that exhibits the difference between different cloud providers that are vulnerable to this attack.

5.7 Inhibiting the Network-Channel Attack

There is the prospect for favorable defences against the proposed attack, the pros and cons of which are discussed here.

5.7.1 Network Channel Resistant Algorithm

The main defence strategy might pursue to modify the open source code of OpenStack cloud to at least limit the granularity of network-based side-channels. The main phase that helps in launching the experimented attack is to reuse the code so as to circumvent the security perimeter and penetrate into the root domain of the system. This attack can be inhibited effectively by thoroughly scrutinizing the code and restricting penetration into the current running system. A

Table 5.3: Cloud Providers that are Vulnerable to This Attack

Sr.	OpenStack	AWS
1	Running instance can be deleted directly	Instance must be stopped before deletion
2	–	Stopped instance remains in the stack. one can restart it. It starts working normally after restarting
3	No hierarchical structure is adopted	There is a hierarchical structure of AWS instances. They are normally created on different layers, i.e EBS Volume, Network, Elastic Load Balancing and Security Layers. Advantage: - Before deleting the instance, it must be removed from layers.
4	–	one cannot delete AWS instances by using the Amazon EC2 console or API because Amazon EC2 actions are not automatically synchronized with AWS. One should delete AWS instances only by using the AWS OpsWorks Stacks console or API.
5	No Auto-healing concept is applied	Every instance has an AWS OpsWorks Stacks agent that communicates regularly with the service. AWS OpsWorks Stacks uses that communication to monitor instance health. If an agent does not communicate with the service for set time, AWS OpsWorks Stacks considers the instance to have failed and try to recover it by using the option of auto-healing.

new security perimeter has been introduced that defends the system in a way akin to a network firewall. Figure 5.12 shows the code implementation that will check the xenapi connection.

The job of such security perimeter is to protect/block the unauthorized users for accessing the root domain. The underneath logic of newly introduced security perimeter is to examine the internal state of the code to identify the malicious connectivity of unauthorized users. Already existing security rules cannot prevent such attacks because these codes are already stored in the internal memory of the system, and security perimeter already examined such codes, so it treats all of such codes equally and vetted. In order to expand the cloud network, the establishment of root to root connection is required. Such connections are established through Xen special api called xapi as explained above. The proposed security api upon connection request will internally check xapi and ensure that whether is there any dual registration of this VM or not. If not, then it will allow the connection otherwise it will be declined. A special api has been implemented in the proposed security perimeter that will check at run time either this xapi has any xenapi connection in its local domain or not? If it returns true, then its connection request would be blocked otherwise connection will be granted. The only limitation of applying such security perimeter is network delay. Figure 5.13a, shows the normal root-root connection in case of cloud expansion. Figure 5.13b shows enabling of security check in case of root-root connection through xapi. This security check will examine the internal code of xapi and then decide to allow or block the connection. After the execution of security check *API*, Figure 5.14 shows the error while

```
20      curr = os.popen( pwd ).read().split( / )[-1].strip() # GET THE CURRENT DIRECTORY AND
21      if curr: # NOT EMPTY DIRECTORY
22          curr = "/%s" % curr # APPEND SLASH
23      else: pass # ELSE IGNORE
24      disp = "%s@%s:~%s$ " % (who, hostname, curr) # GET THE DISPLAY
25      comm = raw_input(disp).strip() # ASKING INPUT
26      nonAdmin = True # SET BOOLEAN TRUE
27
28  def xenapi_local():
29      return Session("https://_var_xenapi_xenapi/", transport=UDSTransport())
30
31  def _parse_result(result):
32      if type(result) != dict or 'Status' not in result:
33          raise xmlrpclib.Fault(500, 'Missing Status in response from server' + result)
34      if result['Status'] == 'Success':
35          if 'Value' in result:
36              return result['Value']
37          else:
38              raise xmlrpclib.Fault(500,
39                                    'Missing Value in response from server')
40      else:
41          if 'ErrorDescription' in result:
42              if result['ErrorDescription'][0] == 'SESSION_INVALID':
43                  return _RECONNECT_AND_RETRY
44              else:
45                  raise Failure(result['ErrorDescription'])
46          else:
47              raise xmlrpclib.Fault(
48                  500, 'Missing ErrorDescription in response from server')
49
```

Figure 5.12: Searching of Xenapi in Xapi

accessing the root.

5.8 Discussion

This section discusses the major finding from the empirical evaluation of *Privilege Escalation using the RoP* attack in this chapter.

- The purpose of this study is to investigate vulnerability in the configuration of a hypervisor running underneath with leading open-source OpenStack cloud platform and commercial cloud platform i.e. Microsoft Azure which an attacker can exploit.
- It was observed that strict network configuration is needed when permission for cloud expansion is granted. Since network expansion using ROP is the main vulnerability which an attacker can exploit and successfully penetrate itself by making an illegal connection with the root domain.
- It was interesting to examine that this experimental setting can successfully control tool stack of root domain from where it can manage or control other co-located VMs.

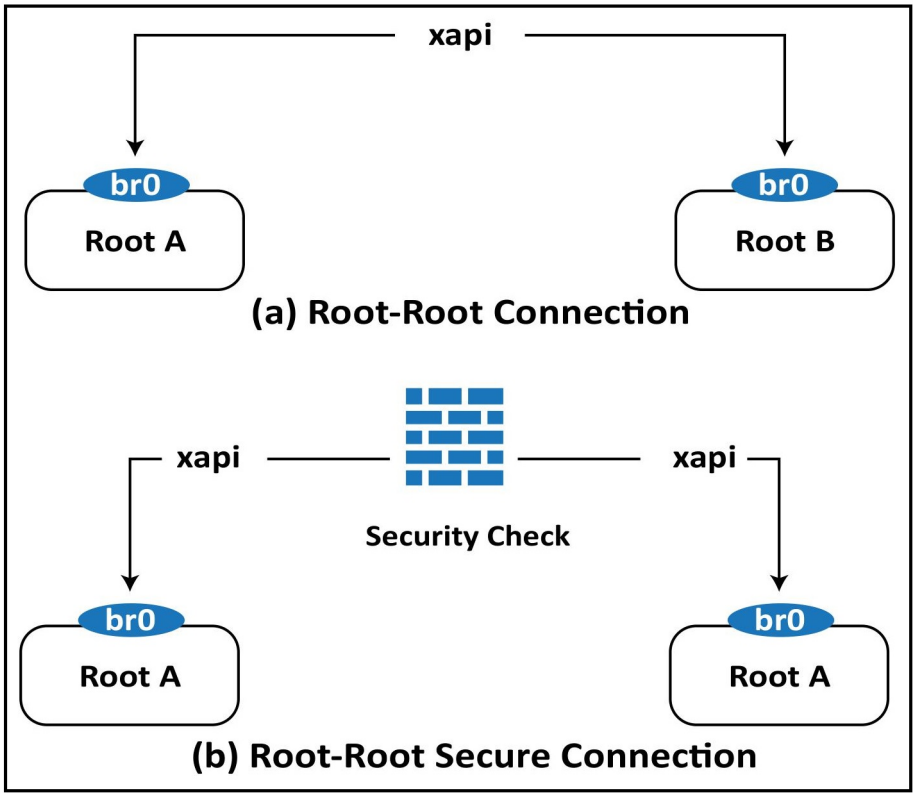

Figure 5.13: Root-Root connections

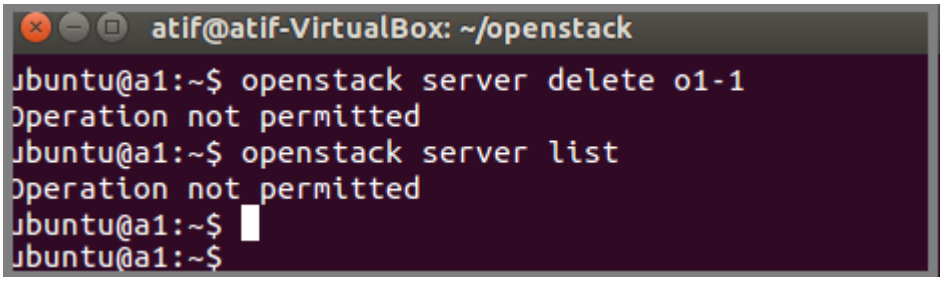

Figure 5.14: Error While Accessing the Root

- It was shown that this attack setting can be countered for exploitation by carefully examine the request for a network connection from non-root VMs.
- Evaluation criteria (Section 5.3) has been addressed as:

  Functional qualitative parameter, the attack methodology is evaluated to check whether it accomplishes all the tasks for which it is designed. Their functional parameter is evaluated by looking at its inference ability as shown in Figure 5.2 in which all these attack strategies have shown to be implemented successfully. The results showed that all the events were inferred. Moreover, the evaluation showed that the attack model fulfilled the functional purpose.

  Expressiveness in the context of this study, is defined as how well *Privilege escalation* attack works the full range of tasks when used within the network domain. The expressive parameter is evaluated by looking at the results in Figure 5.5 and Figure 5.6 which shows a non-root VM is performing the operation of root-VM, which indicates strong evidence about the working of attack.

  From the evaluation of the results, it is demonstrated that the major strength of the presented approach is its ability to make an illegal connections with the root domain and control tool stack. However, the approach works well on *XEN* architecture and does not cover other hypervisors such as *KVM, VMWare* which is the main limitation of this approach.

5.9 Summary

In this chapter, the exploitation of return-oriented programming (ROP) technique has been presented. This exploitation leads to a privilege escalation attack on xen hypervisor by breaking a domain isolation between root and non-root VMs. Existing attacks using ROP have targeted the applications, *libc* library and operating systems, but never experiment the exploitation of network channel through code reusing. Because of the pervasive nature of virtualization, it is thus significant that its domain isolation properties are further needed to be explored and understood. This study mainly deals with the isolation properties (or lack thereof) of a leading open source virtual machine manager (OpenStack) and commercial MS-Azure. Challenges that overcome to execute this attack entails privilege escalation, ROP, difficulty of penetration into the root domain of running system.

Through a combination of these different novel approaches, a new attack model has been assembled that was sufficiently influential in controlling the victim VM. The empirical evaluation of privilege escalation shows that it successfully breaches the security perimeter of cloud providers. It is especially important to note that the exploitation of return oriented programming

(ROP) produces very high risk in almost all the configurations discussed in this chapter. Countermeasure solution has also been proposed by directly modifying the open source code of leading cloud platform i.e OpenStack. In summary, the overall approach described in the experimental evaluation is able to address the research challenges and the the objective of the research have been met to a great extent.

The next chapter summarizes the research work presented in this thesis and highlights the main contributions. It also discusses open research problems in the area and outlines a number of future research directions.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Summary

The objective of work presented in this thesis is to explore a new vulnerability in the existing network architecture of cloud computing can be exploited. Additionally, introduction of novel cross-VM attacks along with their empirical analysis and characterization in leading cloud computing architecture has been presented. The countermeasure solutions of these attacks have also been proposed. The analytical need of leading cloud systems, and their benefits for improving research and technical process within the cloud computing area are discussed in detail. Particularly, this study presents an investigation and technique to illustrate the architectural components within the cloud computing environment. These investigations are leveraged for applied usage, including exploitation of network channel, impersonation of regular network card, RoP in conjunction with network channel, privilege escalation of non-root users, and have further been used to improve security perimeter of Cloud system.

6.2 Overview of Thesis

This section highlights the major findings of this thesis in a nutshell.

Chapter *2* highlighted the concept of the system model and its different components, as well as the development of the modern distributed system to cloud computing. The model and taxonomy of cloud computing is discussed in detail, including deployment models, service models and attack vectors. The concept of security, and how it can be used to enhance cloud computing research is explored and discussed in detail. It is exposed that while cloud isolation in a shared environment is an active research area, still there is a challenge in assessing its security and privacy. Next,

the idea of system architectural model and how it can be used to enhance cloud computing security is demonstrated. The current state-of-the-art in investigation for Cloud components are presented and discussed in detail, and current gaps in the literature are identified. Finally, the significance and applicability of Cloud analytics is presented, including a discussion on how it can be used to improve Cloud practical and commercial operation. This chapter demonstrated the gap in exploitation of network channel, which is the main contribution of thesis, through the combination of impersonation and Linux regular tool.

Chapter *3* focuses on the network architecture of cloud model. This chapter includes the description of the system model, components, its configurations and different service models and advance network features along with their limitations used within the open source and commercial cloud system. Additionally, the analysis infrastructure - consists of how the network traffic comes in and goes out of the system – show the inner network devices used in communication. This chapter concludes by presenting an overview the network architecture, their service model that are used to implement isolation and network devices that are used in leading cloud model through which network traffic of all co-located VMs passes.

Chapter *4* demonstrates the execution of cross-VM network channel attack in leading open source and commercial cloud platforms i.e. OpenStack and Oracle Ravello System. The executed attack applied an innovative strategy by doing a combination of impersonating a TAP interface and constructing a network mirror in the bridge interface, where the network traffic of all co-located VMs passes. This successful experimental setting allows the attackers to compromise the privacy of co-located VMs by redirecting their network traffic at set destination point, which is under the control of attacker. This study will likely leave little to no trace on the cloud infrastructure as attacking VM is not violating the resource limit allocated by cloud provider and removes all the footprints. This chapter also presents an empirical analysis of cloud setup, their configurations, limitations and mitigation strategy to appraise the attack in real-time scenario. The objective of the experiment was to exploit the underlying network channel to observe the network traffic of co-located VMs. The graph result shows the resources consumption by the attacking machine. This attack is challenging in the sense that the attacking VM can consume resources within an assigned limit and does not violate any resource utilization factor. At the end, we have also investigated how to overcome the challenges of distinguishing between normal resource patterns and target attacks from VMs. Countermeasure solution has also been presented in terms of modification in open source code of a cloud platform that prohibited the connection of any external device without proper parameter defined by cloud model.

Chapter *5* presents another innovative study that escalates the privilege level of non-root user and the empirical evaluation of this attack when it successfully breaches the security perimeter

of cloud providers. In cloud model, domains have been defined to identify the privilege level of users, i.e root and non-root user. Normally, root users reside in dom0 and non-root users are in domU. Root users from dom0 can manage all non-root users from domU. In this chapter, it has been shown that how a non-root user can escape from domU and make a connection with dom0 and being a non-root user can perform the task of root user. In doing this, the attacking machine used RoP which is already running code in the memory of cloud platform in conjunction with network channel. Analysis of results shows that such an attack is a serious threat in different cloud settings discussed in this chapter. At the end, countermeasure solution has been presented that modify the open source code of leading cloud platform i.e. OpenStack.

6.3 Major Findings

The main findings of this research have been to investigate the exploitation of cross-VM settings in the network architecture of the cloud model. The thesis has adopted a mixed-methods approach involving a literature review and the development of experimental methods that deal with real-time setups. This has led to the following contributions.

6.3.1 TAP Impersonation and Mirroring

This thesis provides some key insights into the network architecture of the cloud computing model and the particular challenges associated with this area of science. In particular, this thesis has identified:

- The importance of the concept of network architecture in cloud computing as it applies to VM's network traffic flow. In cloud computing networking, there exists a large number of small, heterogeneous and potentially complex network devices that are usually involved in the network traffic flow of VMs.
- Designing a zero-day attack named *TAP Impersonation and Mirroring* in a modern cloud computing model that can successfully redirect the network traffic of other co-located VMs. But to successfully launch this attack some key challenges need to be addressed.
- Five key challenges associated with this attack have been highlighted that make this area quite distinct from the other attacks and which demand a different security response. These challenges are i) Deeply investigating the existing network architecture of cloud computing. ii) penetrating the current network system iii) preempting the network traffic of other co-located VMs. iv) redirection of network traffic of co-location VM at destination point v) elimination of all footprints of attack by hiding all the devices used in launching this attack.

- Additionally, countermeasure strategy is also presented that potentially seems to block these attack strategies for execution in the modern cloud system.

6.3.2 Privilege Escalation using RoP in conjunction with Network Channel

The thesis also contributes insights into current practices of hypervisor architecture in cross-VM settings, including important exploitation of privilege escalation using RoP. Some of the insights that have emerged from this work include:

- The practices of conventional privilege escalation attack strategy are insufficient and suffer from methodological limitation (old technologies) and are easily countered, because of the advance security feature of the cloud computing model.
- The presented attack is dealing key challenges associated with the escalation of privilege level entails i) efficiently usage of RoP ii) penetration into the root domain through non-root domain by using exploitation of RoP in conjunction with network channel iii) control tool stack to manage other VMs.
- The consequence of this attack is so alarming that it can jeopardize other co-located VMs in terms of their data and resources.
- Countermeasure solution of this attack is also presented that disallow illegal connection from non-root VMs.

6.4 Revisiting the Research Goals

The main contributions of the thesis are reviewed by revisiting the research goals set in Section 1.3.1. It should now be apparent that there is a strong mapping between the contributions and the initial research questions which are as:

1. Investigating the network architecture and their associated components for designing a zero-day attack model to exploit a vulnerability in the network architecture of the cloud computing model.

   The first goal has been achieved by designing a *TAP Impersonation and mirroring* attack model as discussed in Chapter 4.

2. The investigation of return-oriented programming (RoP) and associated techniques for the exploitation of hypervisor, and identifying strengths and limitations of this approach.

   The second goal has been delivered by the *Privilege Escalation* attack which is implemented by the exploitation of RoP in conjunction with network channel as detailed in Chapter 5.

3. Evaluate the overall approach through real-world scenarios derived from the analysis of literature and to propose a mitigation solution.

   The third goal has been accomplished by the Evaluation and Mitigation strategies as explained in Section 4.8 and 5.7.

6.5 Future Work

This section discusses some of the future directions in which research can progress on the basis of findings in this thesis.

6.5.1 Implementation of Cloud Model on Mobile Platforms

In cloud platform, there are various cloud providers which are shifted into mobile platforms. A potential future work is to implement the executed attack on mobile infrastructure. It cannot be denied that the advent of mobile applications have also introduced new horizon which are vulnerable to attacks. Contextual information such as location, plays a greater role when the user is using app or sharing information on mobile platforms. It has already been shown that co-location is itself a serious threat. The first step of attacking machine is to co-locate with the target. A possible future work can be to extend the design of mobile applications to save the privacy of users data.

6.5.2 RoP on KVM or other VMM

RoP has been proved to be a severe threat on cloud model. RoP was experimented in OpenStack which is an open source cloud platform having xen running underneath. Future work may focus on KVM or some other hypervisor in cloud model. As all hypervisors have their own configurations and settings, so the exploitation of hypervisor requires some different illusions.

6.5.3 Analysis Extension

The analysis presented within this thesis could be extended to explore a number of additional system settings. Extension of workload models , as stated in chapter 5, analyzing VMs are selected on the basis of resource utilization. It is possible for analyzing the VMs on the basis of individual tasks. Future work may involve applying the described workload analysis method to find out the attacking machine characteristics and behaviour when compared with individual tasks. Furthermore, the workload model does not contain the development limitations of tasks onto servers, which can also be included to study the behaviour of attacking machine as compared to non-constrained tasks.

6.6 Concluding Remarks

The work presented in this thesis identifies the problems of traffic redirection of target machine and privilege escalation which leads to the privacy breaches for co-located VMs in the cloud model. The main contribution of this thesis is impersonation attack, which redirects real-time network traffic of other co-locateed VMs, and privilege escalation which uses RoP to exploit network-channel. These attacks are implemented in real time on open source and commercial cloud platform configured with different security requirements. The evaluation of these attacks presented in this thesis shows the loophole in the network channel of cloud architecture, which an attacking machine can exploit. As part of this implementation process, careful attention shall be paid to ensure the security measurements in the configuration of network channel in cloud model.

BIBLIOGRAPHY

[1] https://www.cloudindustryforum.org/.

[2] http://www.telegraph.co.uk/business/ready-and-enabled/cloud-computing/.

[3] http://www.geglobalresearch.com/innovation/ge-works-bring-air-traffic-management-cloud./.

[4] http://www.lightreading.com/services-apps/cloud-services/orangesita-cloud-prepares-for-takeoff/d/d-id/693612/.

[5] http://www.datacenterdynamics.com/content-tracks/colo-cloud/nasdaq-and-amazon-launch-cloud-platform-forfinancial-services/69672.fullarticle./.

[6] https://www.rickscloud.com/cloud- benefits- in- the- energy- andutility-industry/.

[7] J. Moura and D. Hutchison, "Review and analysis of networking challenges in cloud computing," *Journal of Network and Computer Applications*, vol. 60, pp. 113–129, 2016.

[8] N. Herbst, R. Krebs, G. Oikonomou, G. Kousiouris, A. Evangelinou, A. Iosup, and S. Kounev, "Ready for rain? a view from spec research on the future of cloud metrics," *arXiv preprint arXiv:1604.03470*, 2016.

[9] S. Kounev, P. Reinecke, F. Brosig, J. T. Bradley, K. Joshi, V. Babka, A. Stefanek, and S. Gilmore, "Providing dependability and resilience in the cloud: Challenges and opportunities," in *Resilience Assessment and Evaluation of Computing Systems*, pp. 65–81, Springer, 2012.

[10] S. Iqbal, M. L. M. Kiah, B. Dhaghighi, M. Hussain, S. Khan, M. K. Khan, and K.-K. R. Choo, "On cloud security attacks: A taxonomy and intrusion detection and prevention as a service," *Journal of Network and Computer Applications*, vol. 74, pp. 98–120, 2016.

[11] U. Tupakula, V. Varadharajan, and N. Akku, "Intrusion detection techniques for infrastructure as a service cloud," in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pp. 744–751, IEEE, 2011.

[12] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, "A survey on vehicular cloud computing," *Journal of Network and Computer Applications*, vol. 40, pp. 325–344, 2014.

[13] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Information systems*, vol. 47, pp. 98–115, 2015.

[14] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 8, pp. 805–822, 1999.

[15] R. Roemer, E. Buchanan, H. Shacham, and S. Savage, "Return-oriented programming: Systems, languages, and applications," *ACM Transactions on Information and System Security (TISSEC)*, vol. 15, no. 1, p. 2, 2012.

[16] N. Tripathi and B. Mehtre, "An icmp based secondary cache approach for the detection and prevention of arp poisoning," in *Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on*, pp. 1–6, IEEE, 2013.

[17] S. Kumar and S. Tapaswi, "A centralized detection and prevention technique against arp poisoning," in *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on*, pp. 259–264, IEEE, 2012.

[18] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation computer systems*, vol. 28, no. 3, pp. 583–592, 2012.

[19] P. Mell, T. Grance, *et al.*, "The nist definition of cloud computing," 2011.

[20] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *ACM SIGOPS operating systems review*, vol. 37, pp. 164–177, ACM, 2003.

[21] D. Marshall, "Understanding full virtualization, paravirtualization, and hardware assist," *VMWare White Paper*, p. 17, 2007.

[22] P. Guide, "Intel® 64 and ia-32 architectures software developer's manual," *Volume 3B: System programming Guide, Part*, vol. 2, 2011.

[23] S. Mofrad, F. Zhang, S. Lu, and W. Shi, "A comparison study of intel sgx and amd memory encryption technology," in *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, p. 9, ACM, 2018.

[24] J. Somorovsky, M. Heiderich, M. Jensen, J. Schwenk, N. Gruschka, and L. Lo Iacono, "All your clouds are belong to us: security analysis of cloud management interfaces," in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pp. 3–14, ACM, 2011.

[25] R. L. Krutz and R. D. Vines, *Cloud security: A comprehensive guide to secure cloud computing, 2010*.

[26] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: High-speed covert channel attacks in the cloud.," in *USENIX Security symposium*, pp. 159–173, 2012.

[27] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," *IEEE Security & Privacy*, no. 6, pp. 40–47, 2010.

[28] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.

[29] T. T. W. Group *et al.*, "The treacherous 12: cloud computing top threats in 2016," *Cloud Security Alliance*, 2016.

[30] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities," in *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pp. 5–13, Ieee, 2008.

[31] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[32] V. Nirmala, R. Sivanandhan, and R. S. Lakshmi, "Data confidentiality and integrity verification using user authenticator scheme in cloud," in *Green High Performance Computing (ICGHPC), 2013 IEEE International Conference on*, pp. 1–5, IEEE, 2013.

[33] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, "A survey on security issues and solutions at different layers of cloud computing," *The journal of supercomputing*, vol. 63, no. 2, pp. 561–592, 2013.

[34] S. Zhang, "Deep-diving into an easily-overlooked threat: Inter-vm attacks," tech. rep., Technical Report). Manhattan, Kansas: Kansas State University, 2012.

[35] F. Sabahi, "Secure virtualization for cloud environment using hypervisor-based technology," *International Journal of Machine Learning and Computing*, vol. 2, no. 1, p. 39, 2012.

[36] T. Garfinkel and M. Rosenblum, "When virtual is harder than real: Security challenges in virtual machine based computing environments.," in *HotOS*, 2005.

[37] D. Hyde, "A survey on the security of virtual machines," *Dept. of Comp. Science, Washington Univ. in St. Louis, Tech. Rep*, 2009.

[38] T. Ormandy, “An empirical study into the security exposure to hosts of hostile virtualized environments,” 2007.

[39] J. S. Alexander, T. Dean, and S. Knight, “Spy vs. spy: Counter-intelligence methods for backtracking malicious intrusions,” in *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, pp. 1–14, IBM Corp., 2011.

[40] I. Korkin and I. Nesterov, “Applying memory forensics to rootkit detection,” *arXiv preprint arXiv:1506.04129*, 2015.

[41] Y.-M. Wang, D. Beck, B. Vo, R. Roussev, and C. Verbowski, “Detecting stealth software with strider ghostbuster,” in *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*, pp. 368–377, IEEE, 2005.

[42] D. Molina, M. Zimmerman, G. Roberts, M. Eaddie, and G. Peterson, “Timely rootkit detection during live response,” in *IFIP International Conference on Digital Forensics*, pp. 139–148, Springer, 2008.

[43] B. Blunden, *The Rootkit arsenal: Escape and evasion in the dark corners of the system*. Jones & Bartlett Publishers, 2012.

[44] N. Kumar and V. Kumar, “Vbootkit: Compromising windows vista security,” *Black Hat Europe*, vol. 2007, 2007.

[45] S. Sparks and J. Butler, “Shadow walker: Raising the bar for rootkit detection,” *Black Hat Japan*, vol. 11, no. 63, pp. 504–533, 2005.

[46] T. Ormandy, “An empirical study into the security exposure to hosts of hostile virtualized environments,” 2007.

[47] J. Rutkowska, “Subverting vistatm kernel for fun and profit,” *Black Hat Briefings*, 2006.

[48] J. Rutkowska, “Subverting vistatm kernel for fun and profit,” *Black Hat Briefings*, 2006.

[49] Y. Bulygin and D. Samyde, “Chipset based approach to detect virtualization malware,” *BlackHat Briefings USA*, 2008.

[50] A. Tereshkin and R. Wojtczuk, “Introducing ring-3 rootkits,” *Black Hat USA*, 2009.

[51] K. Chiang and L. Lloyd, “A case study of the rustock rootkit and spam bot.,” *HotBots*, vol. 7, no. 10-10, p. 7, 2007.

[52] J. Clark, S. Leblanc, and S. Knight, “Compromise through usb-based hardware trojan horse device,” *Future Generation Computer Systems*, vol. 27, no. 5, pp. 555–563, 2011.

[53] S. T. King and P. M. Chen, "Subvirt: Implementing malware with virtual machines," in *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pp. 14–pp, IEEE, 2006.

[54] S. Pearson and A. Benameur, "Privacy, security and trust issues arising from cloud computing," in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pp. 693–702, IEEE, 2010.

[55] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-vm side channels and their use to extract private keys," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 305–316, ACM, 2012.

[56] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 199–212, ACM, 2009.

[57] D. Page, "Defending against cache-based side-channel attacks," *Information Security Technical Report*, vol. 8, no. 1, pp. 30–44, 2003.

[58] Z. Wang and R. B. Lee, "New constructive approach to covert channel modeling and channel capacity estimation," in *International Conference on Information Security*, pp. 498–505, Springer, 2005.

[59] P. Ranjith, C. Priya, and K. Shalini, "On covert channels between virtual machines," *Journal in Computer Virology*, vol. 8, no. 3, pp. 85–97, 2012.

[60] `https://www.xenproject.org`.

[61] http://citeseerx.ist.psu.edu/viewdoc/download? [Online; accessed2099].

[62] S. J. Murdoch and S. Lewis, "Embedding covert channels into tcp/ip," in *International Workshop on Information Hiding*, pp. 247–261, Springer, 2005.

[63] H.-C. Li, P.-H. Liang, J.-M. Yang, and S.-J. Chen, "Analysis on cloud-based security vulnerability assessment," in *2010 IEEE 7th International Conference on E-Business Engineering*, pp. 490–494, IEEE, 2010.

[64] A. Squicciarini, S. Sundareswaran, and D. Lin, "Preventing information leakage from indexing in the cloud," in *2010 IEEE 3rd International Conference on Cloud Computing*, pp. 188–195, IEEE, 2010.

[65] B. Grobauer, T. Walloschek, and E. Stocker, "Understanding cloud computing vulnerabilities," *IEEE Security & Privacy*, vol. 9, no. 2, pp. 50–57, 2011.

[66] H. Hlavacs, T. Treutner, J.-P. Gelas, L. Lefevre, and A.-C. Orgerie, "Energy consumption side-channel attack at virtual machines in a cloud," in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pp. 605–612, IEEE, 2011.

[67] https://cloudsecurityalliance. org/wp- content/uploads/.../Domain-13.docx/, 2011. [Online; accessed 19-July-2011].

[68] O. Acıiçmez, B. B. Brumley, and P. Grabher, "New results on instruction cache attacks," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 110–124, Springer, 2010.

[69] Z. Tari, "Security and privacy in cloud computing.," *IEEE Cloud Computing*, vol. 1, no. 1, pp. 54–57, 2014.

[70] J. Wang, K.-L. Wright, and K. Gopalan, "Xenloop: a transparent high performance inter-vm network loopback," in *Proceedings of the 17th international symposium on High performance distributed computing*, pp. 109–118, ACM, 2008.

[71] Y. Xiao, X. Zhang, Y. Zhang, and R. Teodorescu, "One bit flips, one cloud flops: Cross-vm row hammer attacks and privilege escalation," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 19–35, 2016.

[72] S. M. Hashemi and M. R. M. Ardakani, "Taxonomy of the security aspects of cloud computing systems-a survey," *networks*, vol. 2, p. 1Virtualization, 2012.

[73] D. Grunwald and S. Ghiasi, "Microarchitectural denial of service: Insuring microarchitectural fairness," in *35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002.(MICRO-35). Proceedings.*, pp. 409–418, IEEE, 2002.

[74] F. Zhou, M. Goel, P. Desnoyers, and R. Sundaram, "Scheduler vulnerabilities and coordinated attacks in cloud computing," *Journal of Computer Security*, vol. 21, no. 4, pp. 533–559, 2013.

[75] V. Varadarajan, T. Kooburat, B. Farley, T. Ristenpart, and M. M. Swift, "Resource-freeing attacks: improve your cloud performance (at your neighbor's expense)," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 281–292, ACM, 2012.

[76] D. H. Woo and H. Lee, "Analyzing performance vulnerability due to resource denial of service attack on chip multiprocessors," in *Workshop on Chip Multiprocessor Memory Systems and Interconnects*, 2007.

[77] T. M. O. Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," in *USENIX security*, 2007.

[78] H. S. Bedi and S. Shiva, "Securing cloud infrastructure against co-resident dos attacks using game theoretic defense mechanisms," in *Proceedings of the international conference on advances in computing, communications and informatics*, pp. 463–469, ACM, 2012.

[79] Z. Yang, H. Fang, Y. Wu, C. Li, B. Zhao, and H. H. Huang, "Understanding the effects of hypervisor i/o scheduling for virtual machine performance interference," in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, pp. 34–41, IEEE, 2012.

[80] R. C. Chiang, S. Rajasekaran, N. Zhang, and H. H. Huang, "Swiper: Exploiting virtual machine vulnerability in third-party clouds with competition for i/o resources," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1732–1742, 2014.

[81] Q. Huang and P. P. Lee, "An experimental study of cascading performance interference in a virtualized environment," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 43–52, 2013.

[82] S. Alarifi and S. D. Wolthusen, "Robust coordination of cloud-internal denial of service attacks," in *2013 International Conference on Cloud and Green Computing*, pp. 135–142, IEEE, 2013.

[83] Z. Xu, H. Wang, Z. Xu, and X. Wang, "Power attack: An increasing threat to data centers.," in *NDSS*, 2014.

[84] G. van't Noordende, F. M. Brazier, and A. S. Tanenbaum, "Guarding security sensitive content using confined mobile agents," in *Proceedings of the 2007 ACM symposium on Applied computing*, pp. 48–55, ACM, 2007.

[85] A. Bates, B. Mood, J. Pletcher, H. Pruse, M. Valafar, and K. Butler, "On detecting co-resident cloud instances using network flow watermarking techniques," *International Journal of Information Security*, vol. 13, no. 2, pp. 171–189, 2014.

[86] "Common vulnerabilities and exposures. https://cve.mitre.org/.."

[87] R. Schick and C. Ruland, "Introduction of a new non-repudiation service to protect sensitive private data," *Advances in Information and Communication Technologies*, pp. 71–76, 2012.

[88] A. Herzberg, H. Shulman, J. Ullrich, and E. Weippl, "Cloudoscopy: Services discovery and topology mapping," in *Proceedings of the 2013 ACM workshop on Cloud computing security workshop*, pp. 113–122, ACM, 2013.

[89] Z. Xu, H. Wang, and Z. Wu, "A measurement study on co-residence threat inside the cloud," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pp. 929–944, 2015.

[90] V. Varadarajan, Y. Zhang, T. Ristenpart, and M. Swift, "A placement vulnerability study in multi-tenant public clouds," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pp. 913–928, 2015.

[91] T. Olzak, "Secure hypervisor-based virtual server environments," *Tech Republic*, 2007.

[92] http://www.tcpipguide.com/free/t ARPOverviewStandardsandHistory.html/, 1997. [Online; accessed1997].

[93] H. Wu, Y. Ding, C. Winer, and L. Yao, "Network security for virtual machine in cloud computing," in *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*, pp. 18–21, IEEE, 2010.

[94] https://www.wireshark.org/.

[95] H.-C. Li, P.-H. Liang, J.-M. Yang, and S.-J. Chen, "Analysis on cloud-based security vulnerability assessment," in *e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on*, pp. 490–494, IEEE, 2010.

[96] http://www.cisco.com/c/en/us/products/ collateral/switches/catalyst-6500-series-switches/white paper c11 603839.html/, 1999. [Online; accessed1999].

[97] R. Mohandas, V. Thomas, and P. Ramagopal, "Malicious media files: Coming to a computer near you,"

[98] J.-s. Kim and J.-s. Moon, "Detecting code reuse attack using rnn," *Journal of Internet Computing and Services*, vol. 19, no. 3, pp. 15–23, 2018.

[99] Z. Xu, H. Wang, Z. Xu, and X. Wang, "Power attack: An increasing threat to data centers.," in *NDSS*, 2014.

[100] http://www.thetechherald.com/articles/Adobe-confirms-Zero-Day-ROP-used-to-bypass-Windows-defenses/11273/.

[101] A. Seshadri, M. Luk, N. Qu, and A. Perrig, "Secvisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity oses," in *ACM SIGOPS Operating Systems Review*, vol. 41, pp. 335–350, ACM, 2007.

[102] J. Halliday, "Jailbreakme released for apple devices," *Avaliable: http://www. guardian. co. uk/technology/blog/2010/aug/02/jailbreak me-released-apple-devices-legal*, 2010.

[103] B. Ding, Y. Wu, Y. He, S. Tian, B. Guan, and G. Wu, "Return-oriented programming attack on the xen hypervisor," in *Availability, Reliability and Security (ARES), 2012 Seventh International Conference on*, pp. 479–484, IEEE, 2012.

[104] H. Raj, R. Nathuji, A. Singh, and P. England, "Resource management for isolation enhanced cloud services," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, pp. 77–84, ACM, 2009.

[105] J. Shi, X. Song, H. Chen, and B. Zang, "Limiting cache-based side-channel in multi-tenant cloud using dynamic page coloring," in *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 194–199, IEEE, 2011.

[106] T. Kim, M. Peinado, and G. Mainar-Ruiz, "{STEALTHMEM}: System-level protection against cache-based side channel attacks in the cloud," in *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*, pp. 189–204, 2012.

[107] F. Liu, Q. Ge, Y. Yarom, F. Mckeen, C. Rozas, G. Heiser, and R. B. Lee, "Catalyst: Defeating last-level cache side channel attacks in cloud computing," in *2016 IEEE international symposium on high performance computer architecture (HPCA)*, pp. 406–418, IEEE, 2016.

[108] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 494–505, 2007.

[109] B. C. Vattikonda, S. Das, and H. Shacham, "Eliminating fine grained timers in xen," in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pp. 41–46, ACM, 2011.

[110] P. Li, D. Gao, and M. K. Reiter, "Stopwatch: a cloud architecture for timing channel mitigation," *ACM Transactions on Information and System Security (TISSEC)*, vol. 17, no. 2, p. 8, 2014.

[111] Y. Zhang and M. K. Reiter, "Düppel: retrofitting commodity operating systems to mitigate cache side channels in the cloud," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 827–838, ACM, 2013.

[112] Z. Wang and R. B. Lee, "A novel cache architecture with enhanced performance and security," in *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*, pp. 83–93, IEEE Computer Society, 2008.

[113] F. Liu, H. Wu, K. Mai, and R. B. Lee, "Newcache: Secure cache architecture thwarting cache side-channel attacks," *IEEE Micro*, vol. 36, no. 5, pp. 8–16, 2016.

[114] F. Liu and R. B. Lee, "Random fill cache architecture," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 203–215, IEEE Computer Society, 2014.

[115] C. Intel, "Improving real-time performance by utilizing cache allocation technology," *Intel Corporation, April*, 2015.

[116] B.-A. Yassour, M. Ben-Yehuda, and O. Wasserman, "Direct device assignment for untrusted fully-virtualized virtual machines," 2008.

[117] C. Li, Z. Wang, X. Hou, H. Chen, X. Liang, and M. Guo, "Power attack defense: Securing battery-backed data centers," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 493–505, 2016.

[118] J. Wei, X. Zhang, G. Ammons, V. Bala, and P. Ning, "Managing security of virtual machine images in a cloud environment," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, pp. 91–96, ACM, 2009.

[119] M. Georgiev and V. Shmatikov, "Gone in six characters: Short urls considered harmful for cloud services," *arXiv preprint arXiv:1604.02734*, 2016.

[120] J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig, "Trustvisor: Efficient tcb reduction and attestation," in *2010 IEEE Symposium on Security and Privacy*, pp. 143–158, IEEE, 2010.

[121] A. Vasudevan, S. Chaki, L. Jia, J. McCune, J. Newsome, and A. Datta, "Design, implementation and verification of an extensible and modular hypervisor framework," in *2013 IEEE Symposium on Security and Privacy*, pp. 430–444, IEEE, 2013.

[122] Z. Wang and X. Jiang, "Hypersafe: A lightweight approach to provide lifetime hypervisor control-flow integrity," in *2010 IEEE Symposium on Security and Privacy*, pp. 380–395, IEEE, 2010.

[123] A. M. Azab, P. Ning, Z. Wang, X. Jiang, X. Zhang, and N. C. Skalsky, "Hypersentry: enabling stealthy in-context measurement of hypervisor integrity," in *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 38–49, ACM, 2010.

[124] E. Keller, J. Szefer, J. Rexford, and R. B. Lee, "Nohype: virtualized cloud infrastructure without the virtualization," in *ACM SIGARCH Computer Architecture News*, vol. 38, pp. 350–361, ACM, 2010.

[125] J. Szefer, E. Keller, R. B. Lee, and J. Rexford, "Eliminating the hypervisor attack surface for a more secure cloud," in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 401–412, ACM, 2011.

[126] S. Butt, H. A. Lagar-Cavilla, A. Srivastava, and V. Ganapathy, "Self-service cloud computing," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 253–264, ACM, 2012.

[127] Y. Azar, S. Kamara, I. Menache, M. Raykova, and F. B. Shepard, "Co-location-resistant clouds.," *CCSW*, vol. 14, pp. 9–20, 2014.

[128] Y. Han, T. Alpcan, J. Chan, and C. Leckie, "Security games for virtual machine allocation in cloud computing," in *International Conference on Decision and Game Theory for Security*, pp. 99–118, Springer, 2013.

[129] Y. Zhang, M. Li, K. Bai, M. Yu, and W. Zang, "Incentive compatible moving target defense against vm-colocation attacks in clouds," in *IFIP International Information Security Conference*, pp. 388–399, Springer, 2012.

[130] M. Li, Y. Zhang, K. Bai, W. Zang, M. Yu, and X. He, "Improving cloud survivability through dependency based virtual machine placement.," in *SECRYPT*, pp. 321–326, 2012.

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

[132] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," in *ACM SIGARCH Computer Architecture News*, vol. 42, pp. 361–372, IEEE Press, 2014.

[133] F. Brasser, L. Davi, D. Gens, C. Liebchen, and A.-R. Sadeghi, "Can't touch this: Practical and generic software-only defenses against rowhammer attacks," *arXiv preprint arXiv:1611.08396*, 2016.

[134] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer. js: A remote software-induced fault attack in javascript," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 300–321, Springer, 2016.

[135] G. Irazoqui, T. Eisenbarth, and B. Sunar, "Mascat: Stopping microarchitectural attacks before execution.," *IACR Cryptology ePrint Archive*, vol. 2016, p. 1196, 2016.

[136] R. M. Yoo and H.-H. S. Lee, "Adaptive transaction scheduling for transactional memory systems," in *Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*, pp. 169–178, ACM, 2008.

[137] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. R. Weippl, "Dark clouds on the horizon: Using cloud storage as attack vector and online slack space.," in *USENIX security symposium*, pp. 65–76, San Francisco, CA, USA, 2011.

[138] K. Onarlioglu, L. Bilge, A. Lanzi, D. Balzarotti, and E. Kirda, "G-free: defeating return-oriented programming through gadget-less binaries," in *Proceedings of the 26th Annual Computer Security Applications Conference*, pp. 49–58, ACM, 2010.

[139] S. Ristov, M. Gusev, and A. Donevski, "Openstack cloud security vulnerabilities from inside and outside," *Cloud Computing*, pp. 101–107, 2013.

[140] A. Saeed, P. Garraghan, B. Craggs, D. van der Linden, A. Rashid, and S. A. Hussain, "A cross-virtual machine network channel attack via mirroring and tap impersonation," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 606–613, IEEE, 2018.

[141] J. Denton, *Learning OpenStack Networking (Neutron)*. Packt Publishing Ltd, 2014.

[142] K. Pepple, *Deploying openstack*. " O'Reilly Media, Inc.", 2011.

[143] D. Radez, *Openstack essentials*. Packt Publishing Ltd, 2015.

[144] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *2014 IEEE 15th International Symposium on*, pp. 1–6, IEEE, 2014.

[145] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual network computing," *IEEE Internet Computing*, vol. 2, no. 1, pp. 33–38, 1998.

[146] O. Tkachova, M. J. Salim, and A. R. Yahya, "An analysis of sdn-openstack integration," in *Problems of Infocommunications Science and Technology (PIC S&T), 2015 Second International Scientific-Practical Conference*, pp. 60–62, IEEE, 2015.

[147] R. Masoudi and A. Ghaffari, "Software defined networks: A survey," *Journal of Network and Computer Applications*, vol. 67, pp. 1–25, 2016.

[148] L. Coppolino, S. D'Antonio, G. Mazzeo, and L. Romano, "Cloud security: Emerging threats and current solutions," *Computers & Electrical Engineering*, vol. 59, pp. 126–140, 2017.

[149] F. Callegati, W. Cerroni, and C. Contoli, "Virtual networking performance in openstack platform for network function virtualization," *Journal of Electrical and Computer Engineering*, vol. 2016, 2016.

[150] B. Tuan-Anh, M. Canini, V. H. Tran, and R. Sadre, "Cloud network performance analysis: An openstack case study," 2016.

[151] M. Ayache, M. Erradi, and B. Freisleben, "Access control policies enforcement in a cloud environment: Openstack," in *Information Assurance and Security (IAS), 2015 11th International Conference on*, pp. 26–31, IEEE, 2015.

[152] F. Callegati, W. Cerroni, C. Contoli, and G. Santandrea, "Performance of multi-tenant virtual networks in openstack-based cloud infrastructures," in *Globecom Workshops (GC Wkshps), 2014*, pp. 81–85, IEEE, 2014.

[153] S. N. Foley and U. Neville, "A firewall algebra for openstack," in *Communications and Network Security (CNS), 2015 IEEE Conference on*, pp. 541–549, IEEE, 2015.

[154] J. L. Santos and M. Kimmerlin, "Massive-scale deployments in cloud: The case of openstack networking," in *Cloud Engineering (IC2E), 2018 IEEE International Conference on*, pp. 225–232, IEEE, 2018.

[155] A. Vogel, D. Griebler, C. A. Maron, C. Schepke, and L. G. Fernandes, "Private iaas clouds: a comparative analysis of opennebula, cloudstack and openstack," in *Parallel, Distributed, and Network-Based Processing (PDP), 2016 24th Euromicro International Conference on*, pp. 672–679, IEEE, 2016.

[156] X. Zhang, C. Liu, S. Nepal, C. Yang, W. Dou, and J. Chen, "A hybrid approach for scalable sub-tree anonymization over big data using mapreduce on cloud," *Journal of Computer and System Sciences*, vol. 80, no. 5, pp. 1008–1020, 2014.

[157] A. Babu, M. Hareesh, J. P. Martin, S. Cherian, and Y. Sastri, "System performance evaluation of para virtualization, container virtualization, and full virtualization using xen, openvz, and xenserver," in *Advances in Computing and Communications (ICACC), 2014 Fourth International Conference on*, pp. 247–250, IEEE, 2014.

[158] J. Yang and Z. Chen, "Cloud computing research and security issues," in *Computational intelligence and software engineering (CiSE), 2010 international conference on*, pp. 1–3, IEEE, 2010.

[159] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[160] M. Fenn, M. Murphy, J. Martin, and S. Goasguen, "An evaluation of kvm for use in cloud computing," in *Proc. 2nd International Conference on the Virtual Computing Initiative, RTP, NC, USA*, 2008.

[161] `https://www.microsoft.com/en-us/servercloud/solutions/virtualization.aspx`.

[162] `https://www.vmware.com`.

[163] `https://azure.microsoft.com/`.

[164] `https://aws.amazon.com/ec2/`.

[165] https://www.rackspace.com/.

[166] https://technet.microsoft.com/enus/library/gg610610(v=sc.12).aspx/.

[167] J. Sahoo, S. Mohapatra, and R. Lath, "Virtualization: A survey on concepts, taxonomy and associated security issues," in *Computer and Network Technology (ICCNT), 2010 Second International Conference on*, pp. 222–226, IEEE, 2010.

[168] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-vm side channels and their use to extract private keys," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 305–316, ACM, 2012.

[169] S. J. Murdoch and S. Lewis, "Embedding covert channels into tcp/ip," in *International Workshop on Information Hiding*, pp. 247–261, Springer, 2005.

[170] 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*, pp. 313–328, IEEE, 2011.

[171] P. Ranjith, C. Priya, and K. Shalini, "On covert channels between virtual machines," *Journal in Computer Virology*, vol. 8, no. 3, pp. 85–97, 2012.

[172] https://www.openstack.org/.

[173] P. England and J. Manferdelli, "Virtual machines for enterprise desktop security," *Information Security Technical Report*, vol. 11, no. 4, pp. 193–202, 2006.

[174] M. Piotrowski and A. D. Joseph, "Virtics: A system for privilege separation of legacy desktop applications," tech. rep., Technical Report UCB/EECS-2010-70, UC Berkeley, 2010.

[175] https://qubes-os.org/.

[176] A. Marshall, M. Howard, G. Bugher, B. Harden, C. Kaufman, M. Rues, and V. Bertocci, "Security best practices for developing windows azure applications," *Microsoft Corp*, p. 42, 2010.

[177] S. Pearson, Y. Shen, and M. Mowbray, "A privacy manager for cloud computing," in *IEEE International Conference on Cloud Computing*, pp. 90–106, Springer, 2009.

[178] https://www.techrepublic.com/article/understanding-routing-tables/.

[179] https://www.geeksforgeeks.org/program-calculate-round-trip-time-rtt/.

[180] [Online].Available:https://www.paessler.com/bandwidthmonitoring/.

[181] A. Kozlowski, "Comparative analysis of cyberattacks on estonia, georgia and kyrgyzstan," *European Scientific Journal, ESJ*, vol. 10, no. 7, 2014.

[182] B. R. Raghunath and S. N. Mahadeo, "Network intrusion detection system (nids)," in *Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference on*, pp. 1272–1277, IEEE, 2008.

[183] R. Ledyayev and H. Richter, "High performance computing in a cloud using openstack," *Cloud Computing*, pp. 108–113, 2014.

[184] Z. Afoulki, A. Bousquet, and J. Rouzaud-Cornabas, "A security-aware scheduler for virtual machines on iaas clouds," *Report 2011*, 2011.

[185] M. Pomonis, T. Petsios, A. D. Keromytis, M. Polychronakis, and V. P. Kemerlis, "krˆ x: Comprehensive kernel protection against just-in-time code reuse," in *Proceedings of the Twelfth European Conference on Computer Systems*, pp. 420–436, ACM, 2017.

[186] T. Garfinkel, M. Rosenblum, *et al.*, "A virtual machine introspection based architecture for intrusion detection.," in *Ndss*, vol. 3, pp. 191–206, 2003.

[187] E. Pulier, F. Martinez, and D. C. Hill, "System and method for a cloud computing abstraction layer," Jan. 6 2015.
US Patent 8,931,038.

[188] B. McCorkendale and P. Ferrie, "Using a hypervisor to provide computer security," Aug. 9 2011.
US Patent 7,996,836.

[189] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Information sciences*, vol. 305, pp. 357–383, 2015.

[190] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Elasticity in cloud computing: state of the art and research challenges," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 430–447, 2018.

[191] B. Ding, F. Yao, Y. Wu, and Y. He, "Improving flask implementation using hardware assisted in-vm isolation," in *IFIP International Information Security Conference*, pp. 115–125, Springer, 2012.

[192] J. A. Wang and M. Guo, "Ovm: an ontology for vulnerability management," in *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, p. 34, ACM, 2009.

[193] S. T. King and P. M. Chen, "Subvirt: Implementing malware with virtual machines," in *Security and Privacy, 2006 IEEE Symposium on*, pp. 14–pp, IEEE, 2006.

[194] M. Loe *et al.*, *The rise of Viagra: How the little blue pill changed sex in America*. NYU Press, 2004.

[195] R. Wojtczuk, "Subverting the xen hypervisor," *Black Hat USA*, vol. 2008, p. 2, 2008.

[196] S. Chen, J. Xu, E. C. Sezer, P. Gauriar, and R. K. Iyer, "Non-control-data attacks are realistic threats.," in *USENIX Security Symposium*, vol. 5, 2012.

[197] H. Shacham, "The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86)," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 552–561, ACM, 2007.

[198] K. Z. Snow, F. Monrose, L. Davi, A. Dmitrienko, C. Liebchen, and A.-R. Sadeghi, "Just-in-time code reuse: On the effectiveness of fine-grained address space layout randomization," in *Security and Privacy (SP), 2013 IEEE Symposium on*, pp. 574–588, IEEE, 2013.

[199] S. Designer, "return-to-libc" attack," *Bugtraq, Aug*, 1997.

[200] A. Francillon and C. Castelluccia, "Code injection attacks on harvard-architecture devices," in *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 15–26, ACM, 2008.

[201] E. Buchanan, R. Roemer, H. Shacham, and S. Savage, "When good instructions go bad: Generalizing return-oriented programming to risc," in *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 27–38, ACM, 2008.

[202] T. Kornau *et al.*, *Return oriented programming for the ARM architecture*. PhD thesis, Master's thesis, Ruhr-Universität Bochum, 2010.

[203] S. Checkoway, A. J. Feldman, B. Kantor, J. A. Halderman, E. W. Felten, and H. Shacham, "Can dres provide long-lasting security? the case of return-oriented programming and the avc advantage.," *EVT/WOTE*, vol. 2009, 2009.

[204] S. Checkoway, L. Davi, A. Dmitrienko, A.-R. Sadeghi, H. Shacham, and M. Winandy, "Return-oriented programming without returns," in *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 559–572, ACM, 2010.