

CROSS-VM NETWORK ATTACKS & THEIR COUNTERMEASURES INSIDE CLOUD COMPUTING ENVIRONMENTS

Thesis

Submitted for the award of
Degree of Doctor of Philosophy

In

Computer Science & Engineering
BY

Chandramani Singh
(Enrollment No. **MUIT0119038098**)

Under the Supervision of

Supervisor

Dr. Vaishali Singh

Associate Professor, MSOE&T
MUIT, Lucknow

Co-Supervisor

Dr. Mohd Waris Khan

Assistant Professor, Integral University,
Lucknow



Under the Maharishi School of Engineering & Technology
Session 2019-20 (Phase-I)

MAHARISHI UNIVERSITY OF INFORMATION TECHNOLOGY

Sitapur Road, P.O. Maharishi Vidya Mandir
Lucknow 226013

October, 2024



MAHARISHI UNIVERSITY OF INFORMATION TECHNOLOGY LUCKNOW, 226013, INDIA

DECLARATION BY THE CANDIDATE

I declare that this thesis entitled “Cross-VM Network Attacks & their Countermeasures inside Cloud Computing Environments” in fulfillment of the requirements for the award of Degree of Doctor of Philosophy submitted in the discipline of Computer Science & Engineering, School of Engineering and Technology, Maharishi University of Information Technology, Lucknow is an authentic record of my own research work carried out under the supervision of **Dr. Vaishali Singh**, Assistant Professor, School of School of Engineering and Technology in this University and Dr. Mohd. Waris Khan, Assistant Professor, Integral University, Lucknow. I also declare that the work embodied in the present thesis-

- i) is my original work and has not been copied from any journal/ thesis/ book; and
- ii) has not been submitted by me for any other Degree or Diploma of any University/ Institution.

(Signature of the candidate)

Chandramani Singh

Registration Number: MUIT0119038124



MAHARISHI UNIVERSITY OF INFORMATION TECHNOLOGY LUCKNOW, 226013, INDIA

Supervisor Certificate

This is to certify that this work entitled “**Cross-VM Network Attacks & their Countermeasures inside Cloud Computing Environments**” is an original research work done by Sri **Chandramani Singh** under my supervision for the degree of Doctor of Philosophy in **Computer Science** to be awarded by **Maharishi University of Information Technology, Lucknow, U.P.**, No part of this thesis has been submitted by the candidate for the award of any other degree or diploma in this or any other University around the globe.

Dr. Vaishali Singh
(Supervisor)
Associate Professor,
Dept. of Computer Science
MUIT University, Lucknow, U.P.

Dr. Mohd Waris Khan
(Co-Supervisor)
Assistant Professor,
Dept. of Computer Application
Integral University, Lucknow, U.P.



MAHARISHI UNIVERSITY OF INFORMATION TECHNOLOGY LUCKNOW, 226013, INDIA

BONAFIDE CERTIFICATE

This is to certified that **Mr. Chandramani Singh**, Reg. No. MUIT0119038124 has completed the necessary academic term and the work presented by him is a faithful record of bonafide original work under the guidance and supervision of **Dr. Vaishali Singh**, Department of Computer Science, at Maharishi University of Information Technology, Lucknow, U.P. and Co-Supervision of **Dr. Mohd Waris Khan**, Assistant Professor, Integral University, Lucknow. He has worked on “**Cross-VM Network Attacks & their Countermeasures inside Cloud Computing Environments**”. No part of this thesis has been submitted by the candidate for the award of any other degree or diploma in this or any other university around the globe.

Dr. Vaishali Singh
(Supervisor)
Associate Professor,
Dept. of Computer Science
MUIT University, Lucknow, U.P.

Dr. Mohd Waris Khan
(Co-Supervisor)
Assistant Professor,
Dept. of Computer Application
Integral University, Lucknow, U.P.

Acknowledgement

It is a matter of great pleasure and privilege for me to express my deep sense of gratitude to my esteemed supervisor **Dr. Vaishali Singh**, Associate Professor, Department of Computer Science, Maharishi University of Information Technology, Lucknow, U.P. and Co-Supervisor Dr. Mohd Waris Khan, Department of Computer Application, Integral University, Lucknow, for the pains he took in the completion of this assignment and in making necessary corrections in the manuscript.

Sincere thanks are also due to Dr. B. P. Singh, Vice-Chancellor, Maharishi University of Information Technology, Lucknow, U.P. and Dean R&D teams for solving different problems faced by me in the subject.

I am also thankful to my colleagues and everlasting list of friends and well-wishers Prakher Mohan and Ayush Tripathi for their help in various ways during hours of my study.

In the last but not least is my gratitude to my parents for their blessings and their constant inspiration and my loving for his persistent encouragement love and confidence in me without which this investigation would not have seen the light of the day.

My sincere thanks also to Mr. Puneet for untired cooperation while typing of this manuscript neatly.

Chandramani Singh

Abstract

Cloud computing is a relatively new idea in which clients' usage of shared server capacity is constantly adjusted on the fly. Cloud computing is also known as utility computing. Customers are provided with these materials in an enhanced, more cost-effective, and readily accessible state upon request. Cloud computing, which is one of the primary trends in the information technology sector in the current day, has gained speed and begun to revolutionize the way in which businesses construct and provide IT solutions.

The reduction in operating expenses is the most compelling argument for utilising cloud computing. These considerations may persuade Information and Communication Technology (ICT) companies to host their private data and mission-critical applications in cloud environments. Because of the complicated nature of the underlying cloud infrastructure, cloud environments are confronted with a significant variety of issues, including cyber-attacks, root-kits, malware instances, and misconfigurations, all of which present themselves as a serious threat to cloud environments. Because of these dangers, the cloud's overall trustworthiness as well as its reliability and accessibility have taken a noticeable hit.

When developing a model for cloud services, security is the first priority. On the other hand, a number of serious issues have shown that cloud systems do not provide the same level of security as one might anticipate they would. In addition, there is a lack of understanding regarding the provision of safe services using a cloud model, which is necessary in order to overcome such obstacles. This demonstrates the significance of

the fact that what constitutes the risk in the cloud model. One of the most significant dangers posed by a cloud model is that of cost-effectiveness. In most cases, cloud service providers cut costs by sharing infrastructure with several virtual machines that cannot be trusted. This sharing has also resulted in a number of difficulties, including the occurrence of co-location attacks. Cloud service providers protect their customers from attacks that take advantage of co-location by implementing the idea of isolation. Because of this, it is impossible for a guest VM to disrupt the operation of its host machine or the operation of any other guest VMs that are operating on the same system. Isolation of this kind is one of the primary pillars upon which the main public cloud providers have built their security infrastructure.

On the other hand, such conceptual barriers are not completely impenetrable. Numerous earlier research have shown how co-resident virtual machines (VMs) can be made vulnerable to assaults by leveraging shared file systems, cache side-channels, or by compromising the hypervisor layer with rootkits. These vulnerabilities have been demonstrated. Because an attacker can use one virtual machine (VM) to control or access other VMs running on the same hypervisor, the potential for cross-VM assaults is still present. As a result, numerous strategies for the strategic placement of virtual machines are developed in order to take advantage of co-residency.

While it is easy to see how co-location attacks could use shared memory and disc, it is not yet possible to demonstrate fine-grained cross-VM network-channel assaults. That's despite the fact that it's technically possible to do so. It is challenging to use current network-based attacks, such as ARP spoofing and DNS poisoning, against VMs since they take advantage of vulnerabilities in networking technologies. Virtual

machines can be compromised by exploiting these vulnerabilities. Most discussions of network-related issues revolve around the fact that, in comparison to non-virtualized settings, cloud providers isolate co-located virtual machines (VMs) further. This is because the attacker and the victim are typically placed on different virtual network segments. However, it has been demonstrated that this is not always enough to safeguard a vulnerable VM from having its traffic manipulated.

When a malicious virtual machine (VM) crosses the security perimeter of the cloud model, it can adversely affect the security and privacy of other co-located VMs. This thesis gives a complete method and empirical analysis on the development of co-location attacks. The development of co-location assaults, as opposed to the attacks themselves, is the subject of this thesis. In this case, it is crucial that the cloud service provider be able to securely send the data by limiting who can access it.

Impersonation and privilege escalation are two novel attack models in leading cloud models that successfully breach the security perimeter of cloud models. The work that is presented in this thesis has the primary contribution of introducing these attack models, as well as proposing countermeasures that stop attacks of this kind.

The impersonation and mirroring attack models are disclosed to be used in this thesis as the revealed attack model. This experimental arrangement is able to take advantage of the network channel provided by the cloud architecture and successfully redirects the network traffic generated by other co-located VMs. The primary

contribution that comes from using this attack model is the identification of a vulnerability in the modern design of network clouds that can be exploited by an adversary. Previous research has also utilised ARP poisoning and spoofing to abuse the network channel. All of these attacks have been thwarted, though, because current cloud service providers use stronger security measures than were standard in earlier architectures. In order to trick the cloud model's security perimeter, impersonation uses regularly occurring network devices that are already in use.

The 'privilege escalation' attack is the other contribution that has been presented as part of this thesis. With this exploit, we see how a non-root user can gain control of the management domain and elevate their privileges by using the RoP approach across the network channel. This gives the attacker the ability to compromise other co-located VMs without being explicitly granted access to them. In conclusion, a countermeasure approach that can stop any and all attacks of this kind has been offered, and it involves directly amending the open source code of the cloud.

Table of Content

<u>Content Details</u>	<u>Page No.</u>
Title Page	I
Candidate's Declaration	ii
Supervisor Certificate	iii
Bonafide Certificate	iv
Acknowledgements	v
Abstract	vi-ix
Table of Content	x-xii
List of Tables	xiii
List of Figures	xiv-xv
Chapter 1	Introduction
	1-10
	Significance of the study
	2
	Security Problems in Cloud computing
	2
	Research Methodology
	4
	Statement of Problem
	4
	Research Questions of the Study
	5
	Objectives of the Study
	6
	Research Design
	6
	Research Tools and Techniques
	6
	Scope of the Research Work
	8
	Chapter Plan
	9
Chapter 2	Overview of Cloud Computing and Review of Literature
	11-77
	Cloud computing
	11
	Essential Characteristics
	13
	Virtualization
	14
	Software support
	15

	Hardware Support	18
	Cloud Computing Models	19
	Deployment Models	19
	Countermeasures	66
Chapter 3	Cloud System Model	78-104
	Nodes of Cloud Model	78
	Modes of Cloud Configuration	79
	Multi-Node Setup (2-Node)	80
	Multi-Node Setup (3-Node)	81
	Network Services in Cloud Model	81
	DHCP Agent (neutron-dhcp-agent)	82
	L3 Agent (neutron-l3-agent)	82
	Network Provider Services (Software Defined Networking (SDN) server/services)	83
	The installation of networking services for the cloud on physical servers	83
	Management Network	84
	Network of Guests	84
	Linked to the outside world	84
	API network	84
	Restricted Capabilities of the Networking Services	88
	Architecture of the Cloud Network	89
	Vulnerability Discovered in the Architecture of Cloud Networks	91
	Analysis	100
	Summary	103
Chapter 4	Cross-VMN Channel Attacks Using Spoofing And Tap Impersonation: Defenses And Countermeasures	105-145
	Introduction	107
	Statement of the Problem	110

	Technical Challenges	111
	Attack Setting and Challenges Attack Setting	112
	Challenge 3: Observing Network Traffic	118
	Challenge 5: Obfuscation	120
	Evaluation Criteria	122
Chapter 5	Countermeasures to Privilege Escalation Attacks and Their Implementation	146-174
	Introduction	147
	Technical Challenges	154
	Evaluation Criteria	158
	Analysis Results	162
	Summary	173
Chapter 6	Summary and Conclusions	175
	Summary	175
	Overview of Thesis	176
	Future Work	182
	References	185-206

List to Table

<u>Table No.</u>	<u>Table Name</u>	<u>Page No.</u>
Table 2.2	Overview of Cloud Attacks.	66
Table 4.1	Comparison of Related Work and Proposed Work	111
Table 4.2	Resource Allocation of Each VM	125
Table 4.3	Summary Statistics of Each VM	125
Table 4.4	An Overview of the Vulnerability of Different Cloud Systems to the Proposed Approach	139
Table 5.1	Resource Allocation of Each VM.	164
Table 5.2	One Week Resource Utilization of Each VM	164
Table 5.3	Cloud Providers that are Vulnerable to This Attack	169

List of Figure

<u>Figure No.</u>	<u>Figure Name</u>	<u>Page No.</u>
Figure 2.1	(a) Type 1 Hypervisor (b) Type 2 Hypervisor	17
Figure 2.2	Cloud Services Model	21
Figure 2.3:	CIA Triangle	24
Figure 2.4:	Potential Cloud Attack Vectors	30
Figure 2.5:	Privileges Level of Root-kit	39
Figure 2.6:	Anatomy of Cross-VM Attacks	67
Figure 3.1:	Single Node Setup	80
Figure 3.2	Double Node Setup	80
Figure 3.3	Triple Node Setup	82
Figure 3.4	Cloud Model Networking Components	83
Figure 3.5	Cloud Model Networking Services	85
Figure 3.6	General Cloud Architecture	90
Figure 3.7	Networking Devices That Are Employed in the Transmission Of Traffic	93
Figure 3.8	Function of br-int and br-eth	94
Figure 3.9	Xen Hypervisor Architecture	96
Figure 3.10	Architecture of OpenStack Xen	98
Figure 4.1	Main Attack Steps for a Network Channel	113
Figure 4.2	Difference between Tap0 and eth0	118
Figure 4.3	Attack Scenario in OpenStack	121
Figure 4.4	Traffic Capturing at Attacking VM	129
Figure 4.5	Normal Co-residing VM-Network Traffic	131
Figure 4.6	Cyclic Attack Pattern of Network Traffic	133
Figure 4.7	Cyclic Attack Pattern of Network Traffic	134
Figure 4.8	VLAN Configuration of VMs in Oracle Ravello System	136
Figure 4.9	Traffic Capturing at Attacking VM in Oracle Ravello System	137
Figure 4.10	Nova-Network	138
Figure 4.11	Connectivity of Virtual Interfaces at OVS	140
Figure 4.12	Attachment of Dummy Interface	141
Figure 4.13	Blockage of Invalid Interface	142
Figure 5.1	Schematic depiction of ROP exploits.	150

Figure 5.2	Attack Model on OpenStack Xen Architecture	158
Figure 5.3	Attacking Machine Console	161
Figure 5.4	Co-located VM Deletion Command	162
Figure 5.5	Co-located VM Deletion	163
Figure 5.6	VMs List After Deletion	163
Figure 5.7	Resource Comparison of Attacking VM Before and After Attack	165
Figure 5.8	Sub VM List	167
Figure 5.9	SSH to Sub-VM (attacking)	167
Figure 5.10	Deleting Co-located VMs	167
Figure 5.11	List of All VMs After Attack	168
Figure 5.12	Searching of Xenapi in Xapi	170
Figure 5.13	Root-Root connections	171
Figure 5.14	Error While Accessing the Root	172

Chapter-1

Introduction

Introduction

The way businesses create and distribute IT solutions has been completely transformed by cloud computing. It is becoming the de facto standard for operating IT systems in businesses of all sizes. It has clearly progressed as a technology, with major players like Amazon, Microsoft, Oracle, and Google investing in the development of extensive cloud infrastructure for the purpose of providing services to customers. When and how to make the transition from on-premises to cloud-based IT infrastructure is a key concern for many businesses today. Cloud Industry Forum (CIF) [1] released a survey in 2016 indicating that 63 percent of UK businesses intend to move their whole IT infrastructure to the cloud within the next several years. CIF surveyed 250 senior IT experts and corporate decision makers in the UK and found that 78% of companies there are open to cloud computing. Since the initial study was done in 2016, its growth rate has averaged 53%, and by the end of 2018, analysts expected it to reach 85% [2].

Cost savings, adaptability, and the ability to activate resource allocation on demand, all thanks to virtualization technology, are three major advantages of service provisioning via cloud computing. Another perk of using cloud services is that businesses no longer have to invest heavily in their own IT infrastructure, instead only paying for the resources they actually utilise. [3–6]. Despite these benefits, Cloud computing introduces novel obstacles not seen in traditional

distributed systems, which necessitate considerable and in-depth study to identify and quantify real security and operational issues.

Significance of the study

Security is a critical issue for clients who are considering making the switch from traditional computing to cloud computing [4]. When users outsource their data storage to a cloud provider, they cede control over their data and network to the cloud provider rather than keeping physical possession of the data or the network themselves [17], [18]. As a result of the disparity between server and client processing power and reliability, the cloud is vulnerable to a number of risks [19]. Whether through network channel exploitation or privilege escalation, these dangers can put data availability and confidentiality at risk [20].

Security Problems in Cloud computing

Thanks to the cloud's standardised infrastructure and data storage methods, it may be equipped with cutting-edge security measures. Because of the shared nature of the cloud's underlying infrastructure, its supplier can put extra effort into keeping data safe. Cloud computing's unique properties allow it to overcome a number of known shortcomings of conventional architectures, but its widespread implementation could usher in a host of previously unanticipated risks [7].

The CIA trinity of confidentiality, integrity, and availability defines cloud security, which is an essential feature of cloud service delivery platforms.

However, significant challenges towards guaranteeing security are presented by the inherent qualities of cloud platforms, such as elasticity, dynamicity, and scalability. To be elastic, a system must be able to "autonomously supply and de-provision resources in response to variations in workload" [8] so that "at any given time, the available resources fit the present demand as nearly as possible." It is challenging to deliver safe cloud services due to the dynamic nature of resource distribution necessitated by client needs. Because cloud service providers employ the insecure return- oriented programming (ROP) method to integrate more resources [9,15]. Multiple virtual machines and apps using the same physical infrastructure produces complex interactions. A threat actor can compromise other co-located VMs by using this sharing to perform side or network channel assaults. When this happens, other VMs in the same data centre may be at risk of having their data or resources compromised[9]. By centralising various tasks, cloud computing environments maximise the efficiency of their hardware. This, however, renders them more susceptible to dangers such hardware and software failures, unexpected loadswings, and network attacks [9,10], as well as those stemming from unpredictable demands on resources. Assorted heterogeneous platforms may deliver services in the cloud, and VMs share hardware. When it comes to computing, "the cloud" refers to a network of remotely located servers rather than a single physical location. Virtual machine (VM) infrastructure is vulnerable to exfiltration and assaults such distributed denial of service. The TCP/IP stack's built-in flaw [10] is the primary cause of these kinds of assaults. Several new cross-VM attacks that use polymorphism and metamorphism techniques to remain undetected have also been

released recently. Assist attacks on VMs [11, 13] are possible because of the ease with which information about a victim's virtual machines may be gathered and abused under an IaaS cloud paradigm.

Research Methodology

In order to determine where the weaknesses are in the current network and hypervisor architecture of the cross-VM cloud computing model, this thesis proposes an analysis approach and conducts an in-depth analysis of large-scale cloud computing environments. A variety of cross-virtual machine attacks that take advantage of the network architecture [30] and hypervisor [29] have already been discovered, though. The cloud platform's security perimeter, however, currently prevents such attacks. By offering cutting-edge service models or network layers, cloud service providers are constantly improving the security of their customers' infrastructure. Most issues arise from the fact that cloud providers use more isolation between co-resided VMs than in non-virtualized situations, which is the topic of this article. Because of this, virtual networks are often partitioned into "attacker" and "victim" zones. Threats can enter this secluded area as well. In the cloud architecture, malicious VMs can get through this isolation and start an assault. The privacy of the client or the disclosure of sensitive information could be jeopardised if a security flaw of this type were not patched.

Statement of Problem

According to research into cloud computing's security issues, virtualization allows numerous VMs to share the same physical infrastructure, which opens the door to

attacks including cross-VM network channel attacks and privilege escalation. An attacker with low privileges can gain control of additional co-located VMs through a privilege escalation attack, and sensitive customer data can be compromised through a network channel assault.

Research Questions of the Study

The primary objective of this study is to understand the effects of virtualization on cloud computing network channel vulnerabilities and to propose a mitigation approach for cloud security. These sub-goals constitute a more nuanced and targeted version of the overall aim.

Research the cloud computing model's network architecture and its constituent parts to craft a zero-day attack model that takes advantage of a hole in the cloud computing model's network architecture. How many different parts of the network contribute to making the cloud more secure? is one of the fundamental research topics related to this objective. What does the structure of a network look like, and which parts of it regulate data transmission?

Examine the pros and cons of using return-oriented programming (RoP) and similar techniques to exploit the hypervisor and gain root access to a non-root VM.

With this objective in mind, some major research questions are:

- ☐ How can isolation in the cloud best be supported by which domain properties?
- ☐ How may ROP's unique insights aid in exploiting the hypervisor's domain isolation properties?
- ☐ When assessing the viability of the suggested method, what criteria and tools

should be used?

- In what ways might proposed countermeasures close potential security holes in the architecture?

Propose a mitigation strategy after assessing the overall approach using real-world scenarios gleaned from the literature review.

Objectives of the Study

- The main objective of the research study is to suggest the appropriate methods and parameters to evaluate the proposed approach with the help of introducing two innovative attack models in leading cloud models.
- Second objective is to suggest a solution or a way to counteract various attacks by directly altering the cloud model's open source code.

Research Design

This work uses a systematic approach to experimental systems research, with iterative development based on quantitative evaluation of functioning systems.

Examine the cloud's network and hypervisor architecture to find potential weak spots. We create two zero-day attacks—TAP impersonation and mirroring and privilege escalation—to repeatedly test and disclose vulnerabilities in the architecture's network and hypervisor components. These architectures are studied using a large-scale analysis in order to make them more generally effective.

Research Tools and Techniques

The examination of a variety of conventional attack techniques was used to identify the attack strategies that are being suggested. While planning the proposed

attack tactics that will be employed, it is important to gain insight into the network and hypervisor architecture for the deployment setups, therefore it is useful to investigate existing attack strategies such as ARP spoofing, IP spoofing, poisoning, and Row Hammer attacks. This is because the proposed attack strategies will eventually be used. The investigation of these various assault techniques is motivated by the utilisation of numerous baseline attack models that are not only useful for diagnostic purposes but also for determining the degree to which they are accurate. The process of determining a definitive attack strategy begins with an investigation of the conventional techniques of attack, which is then followed by the presentation of a new model. After that, the new model is tested out on a live network and hypervisor architecture, and then it is put through additional examination through experimental methods for update and re-design. Following this iterative method and conducting more evaluations within the network and hypervisor architecture allows one to create a complete set of attack strategies. In order to determine whether or not the proposed attack techniques are viable options, a comprehensive experimental research is carried out.

In addition, the methodology that we use in our research is based on a quantitative analysis of real-world scenarios conducted in a real-time context. On a large scale experimental setup, a quantitative analysis of two major IaaS providers is carried out. The first is an open source cloud platform known as OpenStack, while the second is a commercial cloud known as Oracle Ravello System. When performing the evaluation, multiple co-located virtual machines from a variety of categories are employed as a representative subset of the issues. The evaluation of the suggested

model makes use of the real-time network traffic generated by the co-

located virtual machines (VMs), which operate using a variety of deployment configurations.

Scope of the Research Work

The following are some of the significant contributions that the work that is described in this thesis makes:

The architectural understanding of a network system that is required to build a cross-VM attack using a mix of TAP Impersonation and Mirroring, as well as to conduct a probe on both a public and a private cloud computing model in order to determine whether or not it is feasible. When it comes to cloud computing, where multiple Virtual Machines (VMs) might share the same underlying physical computer, data privacy and security is a primary concern for both the cloud computing service providers and their end users. It is well known that multi-tenant virtualized cloud environments are vulnerable to side channel attacks. These kinds of attacks have been attempted to be mitigated by the security perimeter of cloud computing by limiting interaction between virtual machines using common hardware. In spite of this, there is now a hole in the model of cloud computing's network architecture that has to be investigated. By taking use of the best possible slice of the existing code (the RoP), we were able to design and implement a novel attack (privilege escalation), which we then put through its paces in an actual attack and evaluate empirically for efficacy. An untrusted third party can gain control of the tool stack (root domain) by executing ROP, connecting to the root domain through a malicious network channel, and then using the root domain's access privileges to

steal the tool stack. The number of network channels an adversary needs to accomplish this is minimal. A real-world cloud testbed analysis of the suggested assaults is necessary to construct effective responses against them. The research that has been done up to this point has proposed a cross-VM network channel detection technique [16], [17]. However, this kind of technique is vulnerable since attackers can undertake network channel attacks that get around these mitigation strategies and so launch attacks. The vulnerability lies within a network component, as was discovered by inspecting the insight network architecture of the cloud. As a result, a countermeasure method to these attacks has been developed, and it involves directly changing the open source network code.

The following are some more contributions made by this thesis: A comprehensive investigation into the most cutting-edge methods of cloud assaults across a variety of industries, with a focus on how they operate in cross-VM environments. A thorough investigation on the role of partitioning and the domain isolation capabilities of hypervisors. experiential knowledge of the changes in network and hypervisor architecture that are caused by cross-VM attacks of varying types.

Chapter Plan:

The present study is presented in the following six chapters.

Chapter – 1 is Introduction which presents an overview of cloud computing. This chapter also describes the Objectives and research methodology of the study which covers the objectives, Hypotheses, Significance of the study, Data sources and methodology, Limitations of the study and Tools for analysis.

Chapter – 2 has been described the review of literature. This chapter contains

significant background information for this research and gives a full literature review on assaults in cloud models. In particular, it analyses cross-VM attacks that can occur in cloud environments.

Chapter –3 presents the architecture of the network and how the traffic flows on the network while using the cloud computing model.

In Chapter 4, we see how a modern impersonation and mirroring attack is conceived within the context of a real-time assault on numerous cloud platforms at varying degrees of severity. The conclusion also includes an evaluation of this attack that was proposed.

Chapter – 5 depicts the design of a novel real-time privilege escalation attacking technique is discussed. A combination of network channels and Return oriented programming (RoP) is at the heart of this method. An empirical analysis of the RoP attack is also provided.

Chapter – 6 depicts the summary of findings, conclusions and suggestions. This Chapter presents a summary of the study that was carried out as well as directions for further work.

Chapter-2

Overview of Cloud Computing and Review of Literature

This chapter explores potential security threats and countermeasure solutions that are disclosed in cloud systems. It also gives background information on the work that is described in this thesis. Discussion and examination of existing literature on cloud-based attacks and vulnerabilities revealed by researchers, as well as novel methodology and planned architectures of prior works, might mitigate such attacks and improve cloud security. Security threats were detailed, and the suggested methodology was compared to the existing state of the art. In addition, the evaluation of secure public, private, and commercial cloud platforms as well as previous research work that is connected to the suggested technique are discussed.

The primary objectives of this chapter are twofold: first, to conduct an analysis of the current threat model; second, to evaluate the most recent attacks that can be launched against cross-VM environments; and third, to discuss possible defences against these threats. In order to put this work into perspective, the chapter also provides a more comprehensive analysis of the security concerns raised by virtualization and the solutions proposed to address those concerns.

Cloud computing

Cloud computing is a paradigm of computing that is based on the Internet and functions as a virtualization platform [18]. Customers have access to the centralised

computing resources and environments that are deployed by cloud providers under this approach. These resources are then made available to customers. Customers are able to easily and conveniently outsource the compute and data storage jobs that they need to cloud systems that have tremendous elasticity, are economically viable, and have a high level of energy efficiency. Cloud computing introduces new features such as on-demand computing resources and applications, resource pooling, and broad network access [19]. These new features are designed to reduce the ongoing operational costs of cloud providers and to ensure that customers receive services of the highest possible quality. On the other hand, these capabilities, in turn, pose new security dangers that could affect the computations and data of consumers. In this dissertation, a new attack model and countermeasures are designed to discover and eliminate security flaws in cloud computing. This chapter provides some background information as well as a description of the attack vectors that can be found in cloud systems.

Definition of Cloud Computing

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

Essential Characteristics

According to the definition offered by the National Institute of Standards and Technology (NIST), cloud computing brings with it some new essential aspects in comparison to previous computing models:

- ***Resources available on demand:*** Cloud service providers make it possible for their clients to hire computing resources whenever they require them, which is referred to as "on demand." They provide the idea of a pricing model known as "pay-as-you-go," which bills consumers in accordance with the length of time they use computing resources and the total quantity of those resources. Customers can avoid the expense of purchasing and maintaining real servers thanks to this functionality, which can result in a significant reduction in the upfront infrastructure expenditures incurred by the company [19].
- ***Access to a wide variety of networks :***Cloud computing is a form of computing that is based on the Internet and can be accessed through various networks. Customers are able to remotely manage their computing activities and sensitive data storage in remote datacenters, where they also perform their computational jobs and store their sensitive data. Customers can access the cloud services regardless of where they are or what devices they are using thanks to this feature, which can provide a great deal of flexibility to customers. They also have the ability to remotely share cloud services with other users [19].
- ***Resource pooling:*** In cloud computing, the computing resources of a cloud provider are shared across numerous tenants in a multi-tenant environment. This allows the cloud provider to service multiple clients while utilising a diverse

set of physical and virtual resources. In response to the fluctuating demands of the various clients, these resources are dynamically assigned and unassigned. There is a notion known as location independence, which states that customers often do not have any control or information regarding the precise location of the resources that are delivered to them, but that customers may be able to identify location at a higher level of abstraction (e.g country, state or data center). Storage, processing power, memory space, and bandwidth on a network are all examples of resources [19].

- ***Rapid elasticity:*** Cloud service providers make their offerings to their customers scalable, which enables rapid elasticity. Customers have the ability to scale up their computing resources at any moment in response to an increase in their demand for computing power, or scale down in response to a decrease in that demand [19].
- ***Measured service:*** When it comes to monitoring and measuring the delivery of services, different cloud service providers use various metrics. These measures make it possible to price the cloud services using a "pay as you go" model. Cloud providers may handle automatic on-demand scaling and failure recovery based on these metrics, which enables them to achieve resource optimization as well as analytical planning, which is another benefit of using cloud computing. Full transparency can be offered to both the provider and the client through the examination, measurement, and informing of resource utilisation [19].

Virtualization

Virtualization is critical to the framework of cloud computing, which heavily relies

on it. Through the use of virtualization, it is possible to simultaneously operate many versions of the same or distinct operating systems on a single physical server [20]. Despite the fact that they share the same physical server, all of the operating systems (OS) have the same experience when it comes to the level of abstraction they use to make use of the machine. Each Operating System is Executed Inside of a Virtual Machine (VM). The implementation of virtualization requires support from both software and hardware.

Software support

A specialised piece of software known as the hypervisor or the Virtual Machine Monitor is responsible for implementing virtualization at the software level (VMM). This piece of software comes with a wide range of functions [20]. First, it virtualizes the physical hardware resources (CPU cores, memory, I/O devices, and so on) so that numerous VMs can run at the same time on one physical server. This allows for greater efficiency. Second, it provides a high level of isolation between the various VMs, such that each VM can operate in its own independent CPU context and memory area. Thirdly, it is responsible for managing the actions of VMs, such as the launch and termination of VMs, the suspension and resume of VMs, migration, and so on. There are two different kinds of virtualization [21]:

- Full Virtualization operates at a privilege level that is higher than that of the operating system. It makes it easier for guest operating systems to run on guest virtual machines (VMs). The guest operating system is able to release privileged instructions and sensitive calls in a manner that is analogous to those operating on the actual

hardware. The hypervisor then converts these instructions into instructions that can be executed by the computer (a process known as binary translation virtualization), or else such instructions are directly dealt with by the

hardware (Hardware Assisted virtualization). The hypervisor includes two primary features that are designed to facilitate and secure virtualized environments. These features are (i) protection of OS-Independent Storage Management to provide resources that are shielded from unauthorised access, and (ii) conversion of virtualized environments to allocate physical computing resources to virtual environments. Both of these features are designed to make virtualized environments more accessible and more secure. Despite all of these features, full virtualization still offers various benefits. These benefits not only benefit the consumers in terms of privacy and security, but they also benefit the cloud provider by reducing the cost of infrastructure. These features include:

- emulating new hardware to achieve improved reliability, security, and productivity;
- maintaining isolation between different users and from the control programme;
- sharing hardware resources among multiple users; and maintaining isolation between different users and from the control programme.

- Para-Virtualization is an additional method of virtualization. This method requires the guest operating systems to be modified in order to make special hypercalls that enable them to connect directly with the hypervisor. This type of virtualization is known as "OS Assisted." The requirement that the guest operating system be specifically crafted to execute on top of the virtual machine monitor (VMM), the host programme that enables a single computer to support multiple configuration settings

that are very similar to one another, is the primary limitation of paravirtualization. Paravirtualization, on the other hand, removes the necessity for the virtual machine to configure privileged instructions. The process of setting up the instruction, which is used to manage unexpected or unallowable conditions, can be time-consuming and has the potential to have a negative influence on performance in systems that use full virtualization. An industry-leading hypervisor called Xen is being developed as part of an open-source software effort called paravirtualization. The hypervisor can be further divided into the following two categories: (1) A Type-1 or native hypervisor is one that operates natively on the hardware of the host computer and is used to control privileged host virtual machines as well as other guest virtual machines. (2) A Type-2 hosted hypervisor is one that operates within an existing operating system (called host operating system). It does it from within the operating system that is serving as the host. The conceptual architectures of these two categories of hypervisors are depicted in Figure 2.1. Figure 2.1b draws attention to the fact that a new layer has been added to the type-2 hypervisor.

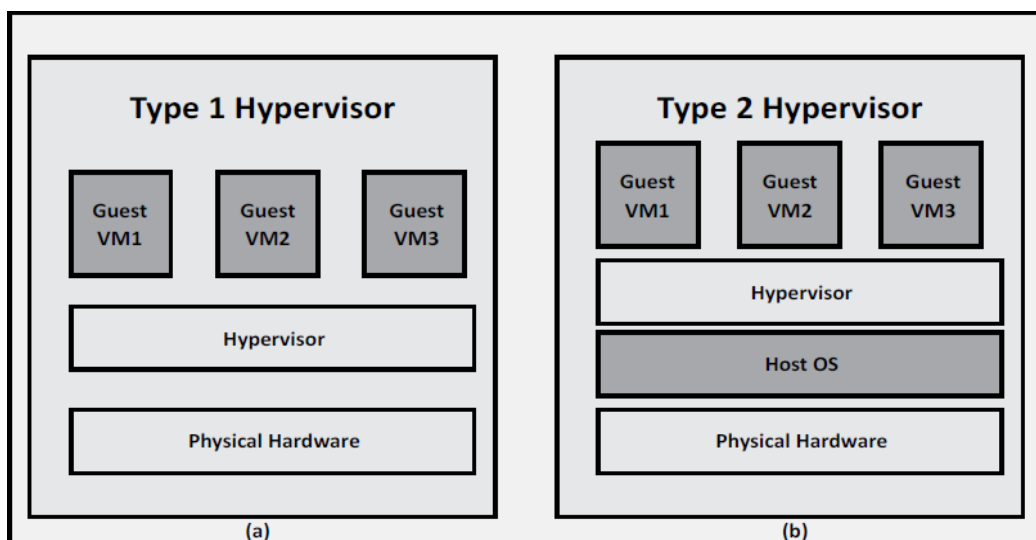


Figure 2.1: (a) Type 1 Hypervisor (b) Type 2 Hypervisor.

Hardware Support

Hardware, as opposed to software, is designed to help with virtualization and to enable the rapid rise of cloud computing. To begin, the proliferation of multi-core processors has led to an increase in the number of processing units that can coexist on a single server. This trend is expected to continue. This improves the cloud server's capacity to host more VMs at the same time, as well as the efficiency with which it uses its resources and its power consumption. Second, key companies that manufacture CPUs, such as Intel VT-x [22] and AMD-V [23], have begun to incorporate hardware virtualization extension features into their products. The management of virtualized functions is made easier as a result of these enhancements' introduction of fresh hardware components, instructions, and execution modes. For example, Intel VT-x has added support for a variety of new functions [22, 23]. I VT-x is capable of enabling two different modes of operation: the VMX root operation, which is a mode with full privileges and is proposed for the hypervisor; and the VMX non-root operation, which does not have full privileges and is proposed for the guest VMs. (ii) VT-x is responsible for the implementation of a Virtual Machine Control Structure, or VMCS, which is used by each VM to manage and store the non-root activities and VMX transitions specific to that VM. (3) Intel VT-x makes use of the hardware known as Extended Page-Table (EPT) in order to allow the virtualization of physical memory, and (4) VT-x provides new instructions that are utilised specifically for the management of VMCS operations and memory. AMD-V also offers support for functions that are very similar. When compared with software solutions, the performance of hardware improvements like these can be greatly

improved upon, which can lead to a large increase in virtualization's overall effectiveness.

Cloud Computing Models

Deployment Models

The provision of cloud services can be accomplished through a variety of cloud computing deployment techniques [19]. Different cloud environments and use cases are exemplified by the deployment models presented here.

- ***Private cloud*** A private cloud model is one in which the cloud's underlying infrastructure is customised to meet the requirements of a single company. The computing requirements of these types of businesses have undergone dynamic shifts, or they require direct control of the computing environment. In most cases, the private cloud system is set up behind firewalls and is controlled by the company. As a result, it restricts access to only those users inside the organisation who have been given permission to use it. In addition, a private cloud can be set up either internally by the company that owns it or externally by a third-party cloud provider.

- ***Community cloud:*** Cloud for the community In most cases, various organisations belonging to the same community and sharing a common interest would collaborate on the creation of a community cloud (security, compliance, authority, etc.). As a result of the fact that these organisations are responsible for an equal portion of the total cost, this model may be more cost effective than private clouds. A community cloud, which is similar to private clouds, can be installed and managed either on the local level by the community itself or on the global level by a third-party service provider.

- **Public cloud** : A public cloud is one that makes its service available to the whole public. In most cases, individual users will share the entirety of the system with other users. This kind of cloud model provides services based on the "pay as you go" model's guiding idea. Renting services can be paid for by any consumer who possesses a credit card that is still in good standing. Customers do not have direct control over the system when using this cloud model as it is configured. A public cloud model is typically managed and run by a commercial cloud provider such as Amazon Web Services (AWS), Microsoft Azure, Google Compute Engine, Rackspace, Oracle, etc. A private cloud model is typically managed and controlled by an internal IT team.

- **Hybrid cloud**: The services of two or more clouds can be accessed through a hybrid cloud, which delivers the advantageous characteristics of different deployment types. Customers are able to more easily integrate, customise, or aggregate its cloud services with those of other cloud models thanks to this feature. Customers of a hybrid cloud, for instance, have the ability to combine the services of a private cloud with a public cloud. The organization's private cloud can be used to store sensitive or confidential data, while the organization's public cloud is used to access computer resources in order to execute an application that is dependent upon the data. This is due to the fact that businesses do not want to store their private information in a public cloud architecture.

Service Models

Cloud computing is a model for facilitating ubiquitous, appropriate, on-demand network access to a shared pool of configurable computing resources (such as

networks, servers, storage, applications, and services) that can be quickly set up and released with negligible administrative work or service provider collaboration [19]. Cloud computing is a model for facilitating cloud computing is a model for facilitating ubiquitous, appropriate, on-demand network access to a shared pool of configurable computing resources (such as networks, servers, storage, applications, This cloud model is constructed using three different service models as its foundation. The acronyms Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service are the names that the NIST gives to these three different types of standard service models (IaaS). At an abstract level, the designs of these three service models are depicted in Figure 2.2. As shown in this diagram, the cloud service provider has control over the components that are highlighted in grey, while the customers have control over the components that are coloured white. These services have been broken out into their own descriptions.

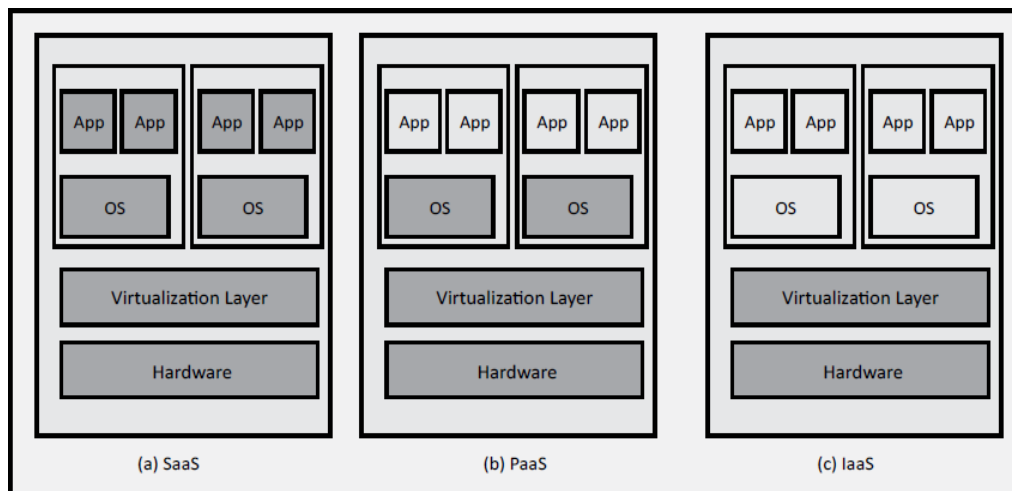


Figure 2.2: Cloud Services Model

- **Software-as-a-Service** Customers receive application software as a form of on-demand service from the cloud provider when using this approach of servicedelivery. The cloud service provider is responsible for managing the underlying

infrastructure and platforms, installing and configuring a variety of application software on the cloud servers, and providing consumers with network access to these software programmes. Customers of cloud services can be granted the right to access these apps directly through the use of programme APIs or web browsers if this strategy is implemented. It is not necessary for customers to configure or install apps on their own personal computers at this time. Email, file storage, social networking, and other similar services are examples of well-known cloud applications. As a result of the incorporation of virtualization into the cloud computing architecture, the cloud provider is able to achieve scalability by simultaneously running numerous copies of the same application in distinct virtual machines. The cloud provider can run numerous distinct applications on the server at the same time [19] in order to maximise the use of the available resources.

- ***Platform-as-a-Service*** Customers can execute their apps on this infrastructure thanks to the programming environments that are provided to them. In this paradigm, the cloud provider is responsible for the implementation and provisioning of computing platforms and environments to clients in the form of a service. These may include operating systems, programming libraries, and databases. Then, clients may directly design, develop, and run their own applications on this platform, without the difficulty of creating and maintaining the underlying infrastructure such as servers and operating systems. This allows customers to save time and money. The provision of context isolation and the allocation of resources based on the requirements of the application are both the responsibility of virtualization in cloud models [19].

- ***Infrastructure-as-a-Service*** Customers do not need to purchase actual servers or network equipment since the cloud provider gives them access to a whole virtual machine that includes operating systems, central processing units, memory, storage, and networking. Customers are able to select the necessary computer resources when it comes time to launch virtual machines. These options include things like the number of CPU cores, the amount of memory, the disc space, and various operating systems. The cloud service provider will then allot these resources from the physical host servers and boot the virtual machines (VMs) on the host servers with the necessary parameters. Customers are able to remotely access their virtual machines (VMs) once they have successfully booted up and may then customise and run whatever software they choose within their VMs. Customers have complete control over their virtual machines (VMs), including the ability to suspend, restart, or terminate their VMs at any moment by making a request to the cloud provider during the life cycle of their VMs. In addition to this, a cloud provider might migrate the virtual machines (VMs) of its clients to different servers for the purpose of improving energy efficiency or fault tolerance. The live migration strategy is often the one that is used by cloud providers at the moment. Using this strategy, it is possible to transfer VMs without causing any disruption to their running processes. When using the live migration strategy, the downtime of migrated virtual machines is practically nonexistent, and there is almost no noticeable impact on performance. In addition, clients are in complete ignorance regarding the migration of their virtual machines [19].

Cloud Computing Security

The term "cloud computing security" refers to a comprehensive collection of techniques, tools, and procedures that have been compiled in order to safeguard cloud computing-related data, applications, and associated infrastructure [24]. When a corporation moves its mission-critical data to geographically spread cloud platforms, which are not directly under the control of that organisation, this is the primary worry of the company. In addition, typical information technology security procedures, security configurations, and firewall rules can all contribute to the reduction of the attack surface that is presented by the cloud. Confidentiality, integrity, availability, authentication, authorization, auditing, and accountability are the primary security principles that secure information assurance. Other security concepts include availability and availability. These ideas are broken down into their essential components in the following sections.

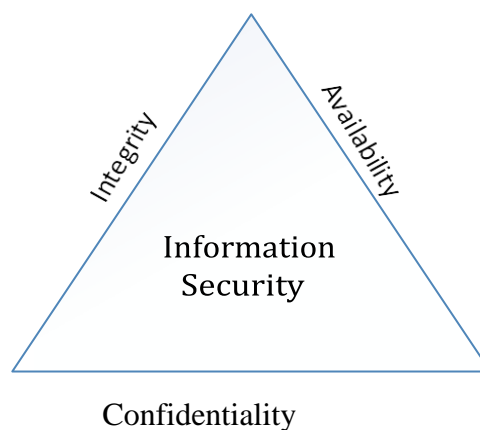


Figure 2.3: CIA Triangle

Figure 2.3 illustrates the concept of the CIA triangle, which refers to the three pillars of information system security: confidentiality, integrity, and availability. They are sturdy pillars that support the guarantee of cloud security [25].

Confidentiality

The prevention of unauthorised disclosure of information or data, whether on purpose or by accident, is what we mean when we talk about confidentiality. The areas of hidden channels, traffic analysis, encryption, and inference are all connected to the concept of confidentiality in cloud computing systems.

- ***Covert channels:*** A covert channel is a means of communication that is both unlawful and unplanned, yet which still enables the transmission of information. It is possible to create covert channels by carefully timing messages or making improper use of storage mechanisms [26].

- ***Traffic analysis:*** Traffic analysis is another domain of confidentiality breach that can be accomplished by analysing the traffic patterns such as the volume, rate, source, and destination of message traffic. This can be done regardless of whether the message traffic is encrypted or in plain text. Traffic analysis is a type of analysis that can be accomplished by analysing the traffic patterns. There may have been a significant event if there was a sudden spike in the volume of traffic and the number of messages being sent. The objective of the mitigation technique for traffic analysis is to keep the volume of traffic at a rate that is essentially constant while also concealing the origin and destination locations of the traffic [27, 28].

- ***Encryption:*** Encryption is the process of concealing original messages within a substitute message so that even if the messages are intercepted by an unauthorised user, the original messages cannot be read by that user. The amount of work required to decrypt the message is wholly determined by the function of the

resilience and quality of the encryption method, as well as the strength of the encryption key [27].

- ***Inference:*** The ability of an entity to use information that is protected at one level of security and correlate it with other information in order to reveal information that is protected at a higher degree of security. The concept of inference is typically associated with database security [27].

Integrity

Data must retain its dependability, correctness, and reliability over its entire life cycle for the integrity of the data to be maintained. The data must not be altered while it is in transit, and precautionary measures have to be taken to ensure that the data cannot be altered by unauthorised individuals. Controls over user access and permissions for individual files are included in these safeguards. In addition to these precautions, there needs to be a system in place that can identify any changes in the data that may have been brought about by events that were not brought about by humans, such as a server crashing. Checksums, or even cryptographic checksums, could be included in some data in order to verify their integrity. In order to bring the corrupted data back to its original, right state, you need to have backups or redundancy available. [28].

The model of information integrity that is provided by cloud computing necessitates the necessity of the following principles:

Internal and external data consistency is essential, as is the prevention of data manipulation by unauthorised parties or software.

Availability

The term "availability" refers to the upkeep of the cloud data centre or the hardware resources, the prompt completion of hardware repairs whenever they are required, and the upkeep of an operating system environment that functions appropriately. ensuring that there is sufficient bandwidth for communication and preventing the formation of bottlenecks. When there are major hardware difficulties, redundancy, fail over, RAID, and even high-availability clusters can help reduce the significance of the problem. In order to be prepared for the worst-case circumstances, you need a disaster recovery plan that is both quick and flexible. Random occurrences, like as natural disasters and fires, must be accounted for in any defences taken against the loss of data or interruptions in connections. A backup copy might be saved in a geographically remote location in the event that such an incident occurs; this would help prevent the loss of data. Security software such as firewalls and proxy servers can protect against harmful acts such as denial- of-service (DoS) assaults and network intrusions, which can cause downtime and make data inaccessible. In addition, this theory guarantees that the safety policies or models used by the cloud system must be in operational condition [28].

Cloud Security Services

Authentication, authorisation, auditing, and accountability are some additional aspects that have a direct bearing on cloud system security.

These elements will be discussed in further detail in the following sections.

© *Authentication*

The validation of a user's identity can be accomplished through authentication by doing analysis or reconciliation. It does both, establishing the user's identification and verifying that they are who they say they are. For instance, a user must first provide their identity by entering their user ID into the computer's login screen before proceeding to input their password. The user's identity is verified by the computer system, which does this by confirming that the person providing the ID is the owner of the associated password [19].

© *Authorization*

The term "authorization" refers to the rights and privileges that are granted to a person or process and which make it possible for them to access computer resources and information. After a user's identification and authentication have been verified, the authorization levels that a user possesses will determine the scope of the system privileges the user is permitted to exercise [19].

© *Auditing*

Auditing and monitoring of the system are the two primary practises that companies implement in order to keep their operations running smoothly. Depending on the design and deployment of the cloud, either the cloud customer or the cloud provider (or both) can use these approaches to manage their cloud resources. An audit of a system can be a one-time or periodic event to evaluate the level of security. Nevertheless, monitoring refers to a continuing activity that observes either the system or the users, as the identification of intrusions [19].

© *Accountability*

Accountability refers to the process of defining the actions and accomplishments of a single person within a cloud computing environment and giving credit to that person for those activities and accomplishments. To perform review for the purpose of analysing previous proceedings and the personnel or processes linked with those procedures, audit tracks and logs provision accountability can be used. There is a connection between the idea of accountability and the principle of non-repudiation, which states that a person is unable to successfully refute the conduct of an activity [19].

Threats in the Cloud Computing Environment

As was previously mentioned, cloud computing offers new features that make it easier for businesses and individuals to implement information technology infrastructure. On the other hand, these new traits, in turn, increase weaknesses that currently exist and introduce new vulnerabilities. The Cloud Security Alliance (CSA) has identified a number of well-known attacks that, when carried out in cloud environments, pose a threat to the computations and data belonging to customers [29]. On the basis of the attack vectors, these assaults have been separated into the several categories that have been developed. The abstract architecture of a cloud system is depicted in Figure 2.4, along with the many attack vectors that could be used. The letter "x" denotes the potential entry points for an attack, which are explained further below:

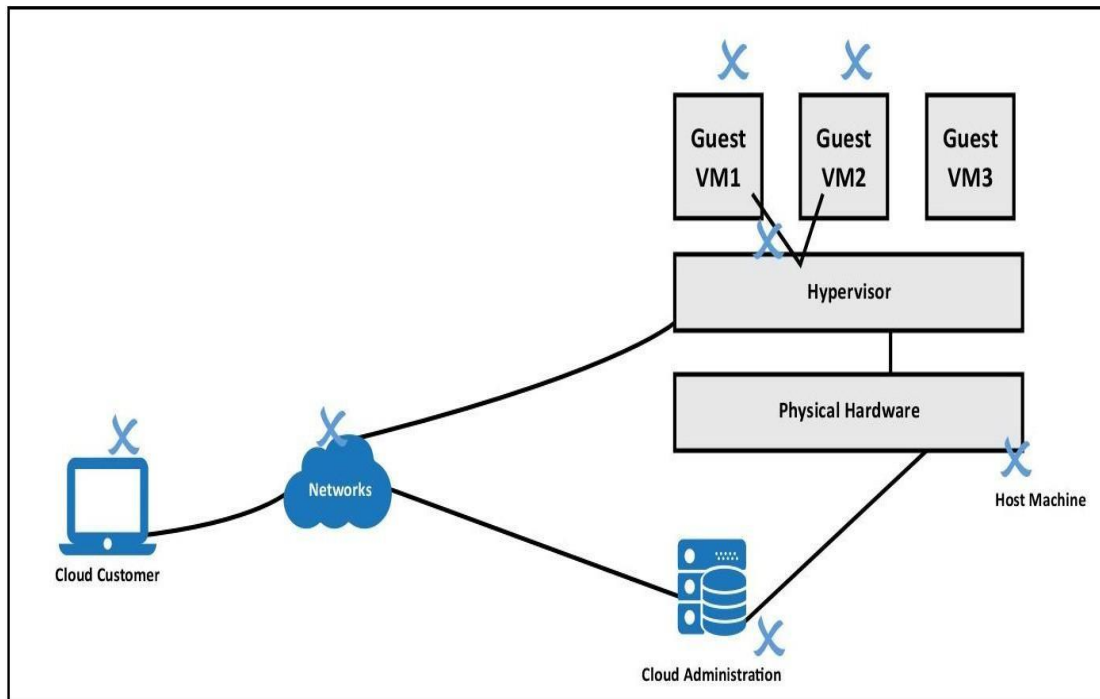


Figure 2.4: Potential Cloud Attack Vectors

Potential Attack Vectors

- ***Service interface:*** A customer must first create an account on the official website of the cloud provider in order to gain access to the cloud computing services. This is a prerequisite for using cloud computing. In order for the consumers to use the cloud services, they will need to log into their accounts. An adversary can now penetrate themselves within a cloud system if they have reached this phase. Because certain cloud systems lack a robust user identification and authentication mechanism, there is a significant risk that an adversary may take over a customer's account and gain access to or breach sensitive or private portions of cloud services. Second, users engage with cloud services through the utilisation of User Interfaces (UI) or Application Program Interfaces (APIs). An adversary might simply use the

vulnerabilities to hack credentials or compromise the cloud services if these interfaces are not configured correctly or securely [30].

- **Networks:** Customers and end-users can access cloud computing's services over networks. This is made possible by cloud computing. When networks are not adequately secured, it is possible for adversaries to steal sensitive data while it is being transmitted. In addition, attackers can perform a variety of assaults on networks, such as denial-of-service (DoS) attacks, which make full use of the resources provided by cloud computing systems. Because of this, clients or end-users are unable to access the data or apps as a result of these kinds of attacks [30].

- **Cloud Administrators:** In order to facilitate management of the cloud infrastructure and services, an administrative position, which is a privileged job, has been delegated to the cloud administrators. The following are some of the ways in which unreliable cloud administrators or administrators who have been compromised might be the primary source of severe security threats for clients. To begin, an unpleasant mistake or accident might take place within a cloud system, which can either result in the permanent loss of data or the disclosure of sensitive data belonging to consumers. Accidental deletion of data by cloud administrators or a natural or man-made calamity, such as an earthquake or a fire in a cloud data centre, are examples of the types of mishaps and accidents that might endanger cloud storage. Second, an authorised user will always have complete access to the information that is kept on the cloud. It is simple for him to mishandle or exploit the sensitive information of clients that is kept in the clouds, or to breach the cloud's key management [30].

Virtualized System

Attackers can take use of system vulnerabilities within cloud servers or virtual machines to tamper with the servers or VMs in order to leak data, seize control of the entire server system, or disrupt service operations. These vulnerabilities can be exploited in a number of different ways. The inclusion of the hypervisor brings about an increase in the likelihood of an attack. The hypervisor is a piece of software that has an extensive amount of code, and as a result, it invariably has a number of flaws related to safety [31]. An attacker can use these defects to compromise the server if they are successful. In addition, the operating system, which is referred to as a guest OS, is loaded into virtual machines. Each guest operating system comes with its own set of vulnerabilities, which puts the safety of the information and services provided to customers at jeopardy.

Shared Infrastructure

A single physical server can host several tenants or virtual machines (VMs), all of which can share the same underlying physical resources. This capability is offered by cloud service providers. It is possible that new cross-VM attacks will emerge as a result of the sharing of hardware resources and components. Within this shared architecture, a malicious virtual machine (VM) might initiate an attack to ex-filtrate or leak sensitive data from other co-located VMs by leaking the data. It is also able to take advantage of shared resources by conducting a denial of service attack on the main server or host machines in order to prevent any other co-located VMs from gaining access to the physical resources.

Cloud computing services

Cloud computing makes it possible to have flexible and inexpensive cloud services. However, users with malevolent intentions can use these services to execute attacks on cloud servers if they exploit them. These consumers have the ability to employ a huge quantity of cloud resources through either fraudulent use of payment cards or free trials of cloud computing services. Customers have the ability to launch attacks against other customers, businesses, or even the cloud provider by using the resources provided by the cloud. They are responsible for a variety of well-known assaults, such as Distributed Denial of Service attacks, large-scale email spam and phishing attacks, brute-force password guessing attacks, port scanning attacks, and many others.

Even though cloud providers offer fine-grained services and strict configuration of security models, there is still the possibility for attackers to attack cloud computing models by exploiting shared memory, cache, network channel, covert channel, and other possible channels. This is the case despite the fact that cloud providers offer these services. In the following section, we will cover the well-known assaults that have been made against the cloud model, as well as the defences that have been developed for them. Cloud computing presents a number of security risks.

Vulnerabilities in Cloud Computing

Traditional security risks, such as the vulnerabilities in networks and the accompanying operating system attacks that are faced in local networks and systems, are equally applicable to the area of cloud computing. Cloud providers are constantly developing new and more sophisticated capabilities for cloud

computing, which might in turn result in the emergence of new security risks for cloud providers as well as customers. When employing computing resources and storage devices, first and foremost, customers should have faith in the cloud providers. Consumers are not responsible for the handling or management of security perimeters, resource allocations, or the management of cloud services. Instead, cloud providers exercise stringent control and monitoring over these aspects of cloud computing. Consumers are extremely reliant on cloud service providers for the safety of their private data and the calculations they perform. Second, cloud service providers typically offer multi-tenancy architecture, which helps to both minimise the cost of the infrastructure and make the most efficient use of its resources. Because of this capability, it is possible for one physical computer to host numerous virtual machines (VMs), each of which may belong to a different type of untrusted user. This facility has the potential to significantly boost the resource utilisation across the entire system and can also minimise the costs of operations. However, due to the fact that the cloud system's hardware resources and storage devices are shared by multiple users, this feature might also bring about new security risks for the system. Thirdly, in order to more effectively manage resources, cloud service providers employ a technique known as virtualization. This additional software layer has the potential to make systems more difficult to understand and may introduce new attack vectors. The purpose of this chapter is to conduct an analysis of the existing cloud vulnerabilities, as well as a discussion of the countermeasures that have been recommended in previous work with the purpose of reducing cloud-based vulnerabilities. The classification of these

vulnerabilities and the solutions to address them are, in general, determined by the attack vectors.

Vulnerabilities in Virtualization

The idea of virtualization, in which a single physical machine is assigned to several users at the same time, is the basis of cloud computing's underlying architecture. In circumstances like these, there is always the possibility of data being stolen [32]. If the virtual machine monitor is breached, the addition of a new layer to the virtualization process could result in the creation of a single point of entry for attackers.

There are potentially three different types of vulnerabilities associated with virtualization, which are as follows:

- ***OS level virtualization*** A single host operating system can support multiple guest operating systems, each of which can be controlled and monitored by the host OS. An attacker can take control of all of the guest operating systems that are running on the host operating system if they are able to compromise the host operating system in this type of configuration [33].
- ***Virtualisation depending on application workloads*** The host operating system sits atop the virtualization layer. Each virtual machine (VM) in this configuration has its own guest operating system, which allows it to execute a unique set of programmes. [33] Vulnerabilities in application-based virtualization are the same kind as OS-based vulnerabilities, hence both types of virtualization suffer from the same problem.

- The same thing as a *Hypervisor, the Virtual Machine Monitor (VMM)* The hypervisor is a piece of code that is built into the host operating system. This kind of code could have some native errors. This code is executed when the host operating system boots up in order to control numerous operating systems running on guests. If the attacker is successful in compromising the hypervisor, then any and all guest operating systems that are controlled by the hypervisor can be compromised [33]. The following is a list of well-known attacks on the hypervisor.

VM escape strategies were defined in [33, 34] as being when an attacker produces a programme that executes a VM, the objective of which is to access the hypervisors' root privileges by breaking the isolation layer. This was described as the "attacker creating a programme that executes a VM." An adversary can gain access to the host operating system by utilising VM escape. This allows the adversary to sidestep the hypervisor layer as well as any other VMs that are operating on the same physical machine. Another difficulty that cloud enterprises face is the proliferation of virtual machines. The term "virtual machine sprawl" refers to the phenomenon in which the number of virtual machines that are operating in an environment that is virtualized grows as a result of the development of new virtual machines that are not required for the operation of the business. Because of this, the newly created virtual machine will be abusing the resources provided by the cloud [35].

Cloud computing, which may be accessible by way of the internet, is capable of running virtual machines. This suggests that the thieves could steal them from a distant location. The majority of hypervisors have the capability to save the

contents of the virtual disc for each VM in the form of a file. This makes it possible for VMs to be replicated and operated on other physical machines. This may be a useful feature, but it also poses a risk to the safety of the system. The virtual machine (VM) can be copied over the network or saved to a portable storage medium, allowing attackers to access data on their own system without having to steal a hard drive in person [36]. After an attacker has gained direct access to the virtual disc, they have a limitless amount of time at their disposal to use offline attacks to circumvent any and all security features, including passwords. The second security flaw of the virtual discs that was discussed in [36] is the possibility that an attacker may corrupt a file or edit it externally while the VM was offline. This indicates that the integrity of an offline virtual machine (VM) could be jeopardised if the host is not properly protected [37].

The hypervisor is responsible for managing the distribution of resources between the host machine and any machines that are running as guests. The ultimate objective of the attacker is to gain access to the host operating system with the same privileges as those of the hypervisor [38]. This will be accomplished by first compromising the hypervisor.

Rootkit

A rootkit is a tool that may be implemented at the software or application level and gives an unauthorised user the ability to take control of a computer system without being discovered. Within the realm of virtualization, cross-virtual machines have the ability to breach rootkits protecting other virtual machines, bring down hardware, and even gain access to confidential information. Rootkits have

numerous characteristics, one of which is the ability to hide malware. In addition to this, rootkits can hide malware from the analysis and detection methods that are used by defenders [39].

Rootkit in Hypervisor

When virtualization is used, the new operating system that runs on top of the host OS assumes that it has full control over the hardware and resources at its disposal. On the other hand, there is no concept of the host's existence in the real world. A compromised hypervisor can also be used to provide a covert channel for executing illegal code into the system. This can be accomplished by using the hypervisor to construct a backdoor. An adversary can exert control over any virtual machine (VM) that is running on the host machine by utilising this method, and as a result, they can alter the actions of the system [33].

Taking Control of the Hypervisor

If the rootkit is able to instal itself below the guest operating systems, then it will have complete control over the machine [40]. This is exactly what rootkits accomplish by utilising various modes of modern x86 architecture, as will be detailed in more detail below. The current x86 architecture is depicted in Figure2.5, together with the level of root-kit privileges.

- ***The User-Mode Rootkit*** and a few additional apps that run in user mode, rather than as low-level system processes, are located in Ring 3. These are able to assist in the accomplishment of goals by either taking the place of a computer's binary applications or by writing over a Dynamically-Linked Library (DLL) [39]. Hooking up with DLL and injecting it Utilizing DLL hooks allows user-mode

rootkits to take advantage of an API hooking. The Vanquish rootkit [41] is a user mode rootkit that is well-known for its ability to redirect requests to the Windows API in order to hide files, folders, and registry entries [42]. This can be achieved by injecting a malicious DLL into a target process. This DLL acts as a middleman for API calls, allowing it to filter requests for files, folders, or registry entries [39, 43].

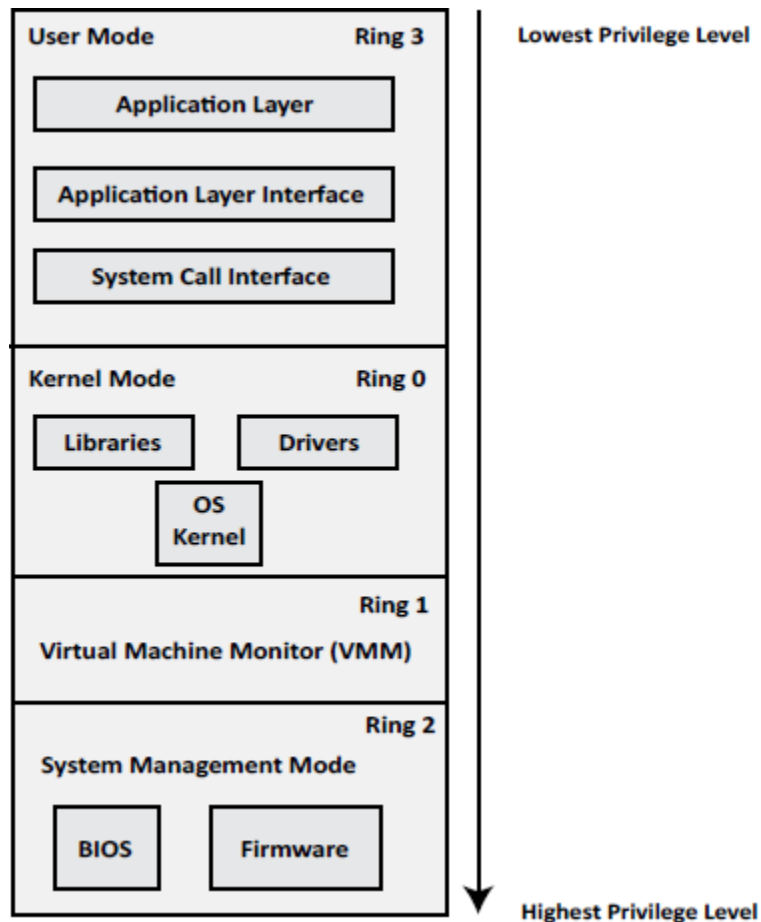


Figure 2.5: Privileges Level of Root-kit

- **Kernel-Mode Rootkit** These rootkits are able to exist in Ring 0, which has the highest privileges level in the operating system, by either adding some code or changing the basic operating system components, such as the kernel and associated device drivers. Ring 0 has the highest privileges level.

Rootkits that run in kernel mode exploit a wide array of vulnerabilities in both the operating system kernel and the security solutions designed to protect against it.

Hooking Tables When it comes to modern operating systems, there are a number of different tables that can be used to perform lookups for a variety of purposes. These lookups can be performed in response to receiving an interrupt, or they can be used to determine the location of a specific system call (also known as `asyscall`). Changing the memory mappings within a target operating system can be accomplished in an efficient manner by modifying the memory descriptor tables [39].

Routine Detours A code segment can be overwritten with jump sequences by using routine detouring [43]. This can happen either at the beginning (pro-log detouring) or at the end (epi-log detouring) of the routine that is being targeted. This helps maintain the original size of the function that was adjusted, but it does not affect the checksum. The overwritten sequence has been copied at the right point in the malicious code segment in order to ensure that the routine functions as intended [39]. This was done in order to prevent any unexpected behaviour from occurring during the execution of the routine.

Binary patching The binary patching technique allows for the direct replacement of the binary representation of the target routine, which frequently involves the replacement of system drivers or files in their entirety. Prior to the activation of the operating system (OS), this vulnerability is frequently exploited via influencing the boot process, namely the Master Boot Record (MBR) and the Basic Input/output System (BIOS). One such illustration is the Vbootkit, which takes advantage of the

method to generate a personalised boot sector code in order to circumvent the safety safeguards that are built into Windows [39], [44].

The Subversion of Virtual Memory This is a further extra sort of rootkit, which demonstrates subversion of virtual memory. ShadowWalker is the name of the rootkit, which has the ability to hook and circumvent the protections placed on virtual memory. When run, ShadowWalker changes the way read and write operations are carried out in order to conceal executable code [45]. The returned frame is changed to an uncorrupted one whenever a read or write operation is attempted on the hidden code region. This is done so that the normal code execution of the hidden code can continue uninterrupted. This is achieved by exploiting the split Translation Look, which is a side Buffer (TLB) of the x86 architecture, and de-synchronizing the two components in order to effectively filter executable code regions from the read/write accesses [39, 46]. The split TranslationLook is a side Buffer of the x86 architecture.

Rootkits that are Based on Virtual Machines ***Virtual Machine Based Rootkits (VMBR)*** function in a layer that is referred to as the Ring -1 layer. The major goal of VMBR is to enable the guest operating system to operate with Ring 0 capabilities, without having an impact on other guest operating systems operating at the same privilege level.

Subversion of the VMM Rutkowska [47] was the first person to exploit this technique to circumvent the operating system in a more advanced manner. This is where it inserted itself beneath the target operating system by utilising these processor extensions and virtualizing the target operating system in order to control

any and all access to hardware peripherals, context switching, memory management operations, and other components as well [48]. Virtual Machine Monitor (VMM) or Hardware Virtualization Machine (HVM)-based rootkits are two common names for these [49].

- ***System Management Mode Rootkits*** Typically, these rootkits, known as SMMR, are found in Ring2. Their development is being done with the intention of providing support for low-level strategy, which will be produced using the following methods:

Rootkits in the BIOS and the firmware This rootkit was able to infect the lowest layer of computer systems, known as the Basic Input/output System (BIOS), as well as the firmware for PCI devices [39].

Modifications to the BIOS and Bootstrap This attack takes place in real-address mode and serves as a first step in corrupting the operating system before it has even had the chance to be initialised [39].

Manipulation of the Firmware This attack makes it possible to remotely insert and execute malicious code within the memory region, which opens the door for Direct Memory Access (DMA) into the operating system that is being targeted [50]. Using this way to perform DMA enables the examination or modification of the operating system that is being targeted [39].

The Interface for Hardware These devices can be utilised to communicate with a system by means of hardware vectors that were not specifically intended for that purpose. [51] A project carried out at RMC is a good illustration of this type of thing. The project's primary focus was on leveraging unintentional USB channels in

order to establish two-way communications with a target system. This work made use of two different unintended channels for exfiltrating data. One of these channels was a keyboard LED channel, which used a combination of the Scroll Lock, Caps Lock, and Num Lock keys. The other channel was an audio channel, which uses waveform files to communicate data to the target operating system [52].

- ***Adaptive Changes to the Hypervisor*** When an initial hypervisor is remotely controlled or modified by an external attacker, the security provided by the original hypervisor is rendered meaningless. The latest generation of rootkits gives an adversary the ability to install an additional layer of hypervisor between the computer's hardware and software. The original operating system is dynamically transformed into a virtual guest after the hypervisor assumes control of the machine and makes the transformation. This sort of hijacking does not require any restart, and therefore renders intrusion detection difficult [33], [37], [53]. This is in contrast to the software-based virtualization, which does require a restart.

Virtual Machine (VM) Monitoring on the Host

The host machine in a virtual environment can be thought of as the central nervous system of the system. There are various ramifications that make it possible for host machines to view and communicate with the execution of VM programmes. Because of this, the security of the host machine is an absolute requirement. There are a variety of virtualization methods that support the implications of the host computer interfering with the operation of VMs within the system. The following is a list of possible ways for the host to influence the VMs:

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 Hooking, through using the DLL hook	Vanquish Rootkit [39], [43]
Kernel Mode Rootkit	Hooking Table [39]	Alters the memory descriptor table	BIOS/Memory Boot Record <i>MBR</i> . e.g, the Vbootkit Shadow walker [45]
	Routine Detours [43].	Overwrites a code segment with a jump sequence	
	Binary Patching [39], [44]	Exploits the boot process	
	Redirects read/write to hide code [39], [46]. Virtual memory sub-version [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.	Caps, NUM locks and Audio channel to communicate data with the target OS, for instance with the RMC [52]
	BIOS and Bootstrap modification [39].	Operates in a real address mode	
	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	

- The host is able to carry out the fundamental tasks associated with virtual machines (VMs), such as starting them up, shutting them down, pausing them, and restarting them.

- The host has the ability to monitor and alter the resources that have been allotted to the virtual machines.
- The host possesses adequate privileges to monitor the programmes that are running within the VMs.
- The data associated with the VM is eventually written to the virtual disc that has been assigned to it, and the host machine is able to read and write to this drive. Because of this, the host is able to do fundamental operations on the content of this data, such as copying, pasting, modifying, or simply viewing it.

More crucially, all communications to and from the VMs are routed through the host, which, once again, enables the host to monitor all conversations that take place within its VMs. In situations like these, the security of every virtual machine (VM) that is operating on the host is put at a significant risk if the host is compromised in any way.

Related Efforts

Attacking from Different Channels While in the Cloud:

All of these different kinds of attacks provide a general risk to a cloud model. The following are some well-known cross-channel assaults in a cloud model that are relevant to our research.

Side Channel Attacks

Side-channel attacks are a type of physical attack that belong to the family of physical attacks called side-channel attacks. In side-channel attacks, an adversary tries to steal important information from other virtual computers. The usage of virtualization can result in the introduction of new security flaws, such as cross

VM-Side channel attacks, which are designed to collect information from the computer that is the target [54]. The following is a discussion of some of the most well-known side channel attacks that have been discussed.

An access-driven attack that takes advantage of shared micro-architecture components like caches is the type of cross-VM assault that is proven to be the most effective. During this phase of the attack, the adversary will run a piece of code on the target system that will carry out actions related to cryptography. This application, which is performed by the attacker, monitors how the cache is used in order to gain knowledge about the key [55].

It is feasible to launch a time-driven side channel attack when the total execution times of the cryptographic operations with a fixed key are affected by the value of the key. An adversary who is able to calculate such time can use this influence to their advantage and exploit it in order to statistically obtain information about the key [56] [57].

For the purpose of capturing a profile of the cache's activities, a trace-driven attack is deployed. According to this, an attacker can get access to a running profile in which cache activity is watched, then process the profile in order to extract the actual activity from the rest of the profile material [56] [57]. This is stated in both of the aforementioned references.

Covert Channel

Covert channels are a type of virtual communication channel that are used between entities in order to circumvent the rules that govern their contact with one another.

Within the context of virtualization, these provide the attackers with new

communication options that are undetected by the VMM (Virtual Machine Monitor) security module [58]. Therefore, the development of security threats occurs within the framework of virtualization, and these need to be avoided in order to achieve the full advantages that may be obtained from utilising this technology.

Researchers demonstrated a covert channel between virtual machines running on the Xen hypervisor in [59], and they did it again in [60]. This conclusion was reached due to the fact that any guest virtual machine is able to access the table that maps the machine address frames to the pseudo physical frames of the virtual machine.

Rutkowska [61] developed a technique for TCP/IP steganography that she named NUSHU. This technique allowed sensitive data to be leaked from a system that had been compromised through network packets created from it. Through the use of TCP/IP header steganography, Murdoch and Lewis [62] have discussed the many different covert channel options that are available. It has been brought to light that this might be relevant within the context of the virtualized environment.

In their paper from 1962, Murdoch and Lewis detailed the header fields that allow for steganography and established the innovative 'Lathra' approach for a covert channel that makes use of TCP ISN (an initial sequence number). It was also stated that an external warden, which is a programme or entity that can watch and analyse data transfer between two different computer systems or programmes, is unable to differentiate between an ISN generated by a machine and a manipulated TCP header. An external warden is defined as follows: Only by decrypting the message

with the key that was used to generate the ISN is it feasible to recover the original message. This covert channel can be established between VMs, and the VMM will be unable to determine its presence [59].

It was hypothesised in [59] that a timing covert channel, which can be used to communicate covertly between VMs and an attacker, can also be used to leak information from possible VMs. This was a possibility because timing covert channels can be used to communicate covertly between VMs and an attacker. It is necessary to send TCP packets from a virtual machine at varying intervals of time in order to send information secretly. When a TCP packet is sent during a particular time interval, the recipient will recover the message as bit 1; otherwise, the receiver will recover bit 0 [63].

It has been discovered that there is a new network-based covert channel that makes use of two sockets in order to send data in a covert manner. An adversary could utilise this to their advantage in order to steal information from a VM [59].

Indexes might hold a significant quantity of information that is pertinent to the data itself. They are generated when the service provider receives data from the users, and decides to build indexes in order to increase search performance. This causes the development of the indexes. Users are completely unaware that their data is being used in this manner, which almost certainly contributes to the disclosure of a great deal more information. The searcher will first query the index in order to receive a list of documents that are relevant to their query, and the index will then map each term to a collection of documents that include the phrase. After that, the index host is responsible for applying these policies for each searcher in order to

filter search results in an appropriate manner. Since only the indexing host needs to be contacted in order to completely execute a search, searches can be highly efficient. Hackers could cause a total and catastrophic loss of privacy if they gain access to the index host [64].

Assaults on Different Images

An adversary can launch an attack against OS images in order to collect sensitive data from other guest operating systems and to cause the OS to fail. Attackers can still gain access to dormant VMs even if they are not actively being used. This is due to the fact that the backend data centre is always operational, and an offline VM is not deemed to be the same thing as a powered-off home computer. Second, in order to bypass legacy-vulnerable programmes or trapdoors, pre-built images need to be properly checked. This is necessary in order to ensure that no vulnerabilities are missed. For instance, Amazon Web Services (AWS) pre-built images save the SSH keys of their builders locally, which means that publishers have access to any and all servers that use these types of images [34].

The creation of a template image of an operating system (OS) by a cloud provider, followed by the cloning of that image to many machines, is a standard operation procedure in cloud computing. If there are any VM template images that are vulnerable, then it could spread to a large number of systems. An adversary can gain access to one of these VMs through rental, and once inside, they will be able to investigate any significant configurations, including administrator rights. Another significant concern that they have brought to light is the fact that a picture

can even be obtained from a source that cannot be relied upon, which can give an adversary back-door access [65].

According to the information presented in [66], the Guest VM carries the potential to bring down the Host OS as well as any other guest VMs that are hosted by it. Sharing virtual machine images on the cloud, as Modi et al. [33] detailed, presents a number of potential security vulnerabilities. The owner of an image's primary concern is its level of confidentiality, which includes factors such as the likelihood of unauthorised users accessing the image. The user of a picture is worried about its security, such as the possibility of a malicious image corrupting or stealing the user's own personal data. For instance, instances that are operating on Amazon's EC2 platform are vulnerable to a wide variety of attacks, including signature- wrapping attacks, cross site scripting (XSS) attacks, and denial of service (DoS) assaults. These vulnerabilities make it simple to compromise these instances. Attackers are able to build, modify, and destroy VM images when they use these types of attacks. Additionally, they are able to change administrative passwords and settings that are put into instances using EC2 for S3 access. Before being uploaded, images of pre-configured virtual applications and computers run the risk of being tampered with or given incorrect settings [67].

Attacks to One's Memory

Attacking the actual hardware of a system, such as memory or storage units, allows an adversary to exfiltrate data. In this section, a variety of instances have been provided to illustrate how successfully attackers might activate different memory areas.

Once an attacker has managed to co-reside on the same physical computer as a target, the next step is to exploit the hardware resources and extract sensitive data through cross-VM assaults. This can be done once the attacker has successfully co-resided on the same machine. It does this by utilising a method for encoding information and then accessing the latencies of a shared L2 cache [56], [68]. This constitutes an abuse of the hardware.

The malicious VM will first write continuous blocks of memory, and then it will release those blocks after it has finished writing. After some time, the attacker will release those blocks, and the host VM will then use its own instruction set to overwrite those blocks. One of the operational characteristics performed by the target is the manner in which those memory blocks are written to by the machine that is the target after the attacker has released them. The attacking VM will next attempt to reproduce the possible instruction set, check for any missing blocks of memory, and read back the identical instruction set [69].

An event channel functions in the same way as a signal channel in that it notifies parties involved in communication about the arrival of a new event. The data that was just written are described for the reader by the writer. Then, once a reader has finished reading it, it deletes the data and tells the writer that a new space is ready for the next input through the event channel. Lastly, the data can be retrieved again by the writer. Therefore, adequate security must be provided for the event channel; failing to do so will result in a breakdown of the entire communication system. It is possible that delivering incorrect information will result in the shared memory channel becoming out-of-sync [34].

The grant table includes support for two distinct kinds of grants between several VMs. The first one is called page-flipping, while the second one is called page-sharing. The high frequency of hypercalls results in an unacceptable amount of performance overhead for the per-packet page-flipping method. As a result, the new communications do not include the ability to turn pages, but they do include a grant for page sharing. The technique that Xen uses to share memory operates at the page granularity level. The number that is used to identify shared pages is called a grant reference, and it is an integer. The hypervisor is responsible for storing the grants information, communicating it to the communication VMs by way of the event channel, and passing on the grant reference. When it comes to authenticating communication parties, the hypervisor will serve as the authority. For reasons related to performance, the system may, under certain conditions, give communication parties the responsibility of managing the grant table all by themselves. [34, 70].

According to Ranjuth [59], if a virtual machine (VM) is moved to a new host, the memory that was being used by that VM would be recovered by the Virtual Machine Monitor (VMM). In the event that a new VM is created on that VMM, there is a potential that the RAM that was being utilized by the previous VM will be allotted to the new VM. It's possible that this new VM is an adversary. As a result, the adversary has the ability to search through all of the pages in the memory for certain information such as passwords, session keys, and other elements pertaining to the initial VM.

When a VM is deleted or turned off, the information contained within the virtual machine runs the risk of becoming public. Following the destruction of a virtual machine or its subsequent shutdown, the memory of that machine might be allotted to a different virtual machine that operates on the same VMM [59].

Row Hammer Attacks

The most recent DRAM chips have an enormous capacity and an extremely high density of memory cells. As a result, a memory cell may experience disturbance errors as a result of electrical interference from cells that are located nearby. Some data bits in the memory area, to which the opponent does not have access rights, can be flipped as a result of electrical interactions, and this is especially likely to occur when the adversary makes frequent and rapid visits to the DRAM with very specific patterns. Row hammer attacks are one type of vulnerability that can exist in DRAM hardware.

Xiao and colleagues [71] took use of a memory flaw in order to launch an attack on a para-virtualized platform using a guest virtual machine. An enemy virtual machine (VM) maintained accessing certain data in the DRAM during the course of this attack in order to flip important bits of this VM's page table entry. When this is done, the entry in the page table leads to a phony page table, which prevents the hypervisor from observing or checking the real page table. The VM's virtual page is interpreted by the fake page table as a physical page that does not have any link with this VM. As a consequence of this, the attacker VM has the capability of stealing or otherwise manipulating the sensitive data stored on the co-located VMs.

Code Reuse

Exploits that involve code reuse rely on code pieces called gadgets, which can be found in memory at specific addresses [23, 36, 39]. Code diversification and randomization techniques, also known as fine-grained address space layout randomization (ASLR) [105], can prevent code-reuse attacks by tampering with executable code at the function [13, 64], basic block [38, 118], or instruction [57, 87] level. This causes the precise location of gadgets to become unpredictable [72]. "just-in-time" ROP, also known as JIT-ROP, is a method that was developed by Snow et al. [105] as a way to get around fine-grained ASLR for programs that have integrated scripting capabilities. JIT-ROP is a staged attack: first, the attacker takes advantage of a memory disclosure 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 of the attack.

Denial of Service

An assault known as a denial of service (DoS) poses a significant risk. Due to the lack of cluster authentication provided by the various communication protocols that are currently in use [33, 34, 72], this kind of attack poses a risk to both the privileged host OS and the regular guest OS. Attacks that use denial of service render other hosts unable to carry out their tasks in a timely manner. Sharing of hardware is another vulnerability that can be used to launch host-based denial of service attacks. Both the performance of the victim VM and the performance of the

adversary VM can be negatively impacted by the adversary VM's ability to cause contention over various sorts of shared resources.

The central processing unit is the first resource that has been negatively impacted and is the primary focus of this discussion. It was indicated in Grunwald and Ghiasi's paper that it is possible to flush the shared processor pipeline; however, doing so will result in a performance hit for the victim. They were able to accomplish this by introducing de-normalized floating point numbers, which led to the creation of an underflow in the pipeline. Because of this, the pipeline needed to be flushed in order to deal with the exceptional condition. Zhou et al. [74] demonstrated a CPU resource attack, in which an attacker virtual machine (VM) can take advantage of the boost mechanism contained inside the Xen credit scheduler in order to improve its scheduling priority and acquire more CPU resources than are paid for.

The memory system is the subject of the second applicable example. Varadarajan et al. [75] anticipated the resource-freeing attack, which is when an attacker deliberately increases the victim VM's use of one form of resource, such as network I/O, in order to force it to release other types of resources, such as CPU caches. In this type of attack, the victim VM is forced to release resources that it has been holding onto. Because of this, an attacker can exert more influence over a co-located virtual machine (VM) and utilise more of its resources. Grunwald and Ghiasi [73] looked at the effect of trace cache expulsions on the victim's execution while Hyper-Threading was enabled on an Intel Pentium 4 Xeon processor. They found that a malicious thread can slow down a victim's performance by a factor of

10 to 20, which can have a significant impact on the application's overall throughput. The researchers Woo and Lee [76] found that regularly emptying shared L2 caches on multi-core platforms could cause a victim's processing speed to be slowed down. They investigated both the saturation and the locking of the buses that link the L1/L2 caches to the main memory [76]. Contention attacks on the schedulers of memory controllers were researched by Moscibroda and Mutlu [77].

I/O resources and networking make up the third category of denial of service attacks. Contention for I/O resources is another factor that might be to blame for a victim's deteriorating performance. Bedi et al. [78] presented a network-initiated DoS attack for network resources. This attack would involve the attacker VM causing contention inside the Network Interface controller. The goal of this attack would be to make the victim's operation less effective. A method of reverse-engineering the I/O scheduling within the virtualization platform was proposed by Yang et al. [79] for disk resources. This method helps the attacker construct particular Denial-of-Service assaults on the disk I/O resources. A more effective adaptive attack was developed by Chiang et al. [80], which classified the victim's I/O consumption pattern and synchronized the assault phase with the victim. Cascading performance attacks were proposed by Huang and Lee [81], and they include an attacker VM consuming the I/O processing capabilities of the Xen Dom0. This causes the performance of the victim VM to suffer as a result. In a similar manner, Alarifi and Wolthusen [82] made use of VM migration in order to lessen Dom0's capacity for I/O processing.

The effect that it has on power is the final aspect of DoS that will be examined. An adversary may lessen the amount of power that is utilized by the host server in order to disrupt the victim's cloud services or perhaps the victim's entire server. Xu and colleagues [83] devised an attack that makes use of power over-subscription techniques; this causes harm to the datacenter. This method generates instances on the host server and runs power-hungry programs, which brings the server's total amount of available power up to its maximum level. As a result, it has an impact on the overall power consumption because it exceeds its levels. As a direct consequence of this, the power unit gives out, and the server turns off.

Cross-VM attack

Co-location Attack

In a system that uses IaaS, a VM co-location attack poses a threat to the position of the victim VM, for example its host server, which the attacker is able to identify. When this occurs, the attacker launches their virtual machine (VM) on the same host server as the victim's VM. As a result, both VMs share the same hardware resources, including I/O and network devices. Attacks that take advantage of co- location are a requirement for attacks that target shared infrastructure [72, 84].

Researchers have already offered a variety of strategies that can be utilized to successfully get co-residency. The term "floater" refers to one strategy for establishing co-residency [85]. In order to start the hunt for the machine that is his intended target, the attacker will first generate a large number of instances in the cloud. These instances are known as flooder. Each flooder broadcasts its existence to a master host; the client, which is an adversary operating outside of the cloud, is

the target of the attack. After that, a string of signals is transmitted to every flooder. The flooder responds to these signals by injecting a network activity into the outbound interface of the physical host computer it is running on. After doing so, the client can test whether or not the two locations are colocated [85].

Ristenpart et al. [56] presented the first demonstration of a network-based methodology for initiating colocation attacks in a public cloud, such as Amazon EC2. An adversary can launch a large number of unneeded VM instances inside the same availability region as the victim VM, and then use a variety of methods to determine whether or not their VMs have been successful in co-locating with the victim VM. Executing the TCP SYN traceroute for the purpose of determining the network traffic's first hop (which is Dom0 in the host Xen server) of this VM and the victim VM is one of the ways for investigation that the attacker uses. Other methods include: It will be possible to determine whether or not they are co-located if his VM and the victim VM share the same Dom0 IP address. In addition, the attacker is able to determine the amount of time it takes for a network packet to complete a full circuit between his VM and the VM that is being attacked. A lower value implies that the host machine is shared by two virtual machines (VMs). The attacker can then verify the internal IP addresses of both his virtual machine and the virtual machine that he is attacking. Internal IP addresses that are numerically near to one another suggest that two virtual machines (VMs) most likely reside on the same physical server [86]. Watermarking is a simple signature measure that is used for identification reasons. It might be beneficial for post-hoc leakage point identification since it denotes the

origin of ex-filtrated content in a way that is not identical to any other source [87].

The attacker infiltrates the network activities of each of his malicious VMs while simultaneously measuring how well the victim VM performs on the network. The co-location of the malicious VM and the victim VM is confirmed when there is a delay in the network caused by the victim VM. This delay indicates that the performance of the victim VM has been negatively impacted by the malicious VM.

A method for co-location that consists of two steps was proposed by Herzberg et al. [88]. The first thing the attacker does is determine the internal IP address of the virtual machine that they are attacking. The attacker will next configure a client computer to connect to the victim's web service using the victim's public IP address. For example, the attacker may download a file or a web page from the victim's server.

An additional prober VM is used by the attacker in order to send a huge number of network packets to each and every conceivable internal address inside the range of the address block. If the client machine of the attacker observes that the performance of the victim has been negatively impacted by the prober VM, then the internal address of the victim VM is the one that is being flooded by the prober VM. The attacker then makes use of a Time to Live (TTL) scanner, using the victim's internal addresses to determine the number of hops that exist between his virtual machine and the one belonging to the victim. A zero TTL suggests possible co-residency.

The DNS lookup method was utilized by Xu et al. [89] in order to determine the victim VM's internal IP address, which led to the subsequent confirmation of the co-location through the use of two stages. In the initial step of the process,

implausible pairs of co-located VMs were pre-filtered out by inspecting the /24 prefix that was present in each of the internal IP addresses. If the internal IP addresses of two virtual machines do not share the same /24 prefix, then it is unlikely that the virtual machines are co-located. The second step is to construct a bus locking covert channel between each pair of virtual machines (VMs) so that the bus locking covert channel may be used to justify the co-location of the machines. If two virtual machines (VMs) are able to interact with each other over this hidden channel, then it indicates that the VMs are running on the same physical machine. Varadarajan et al. [90] used the bus locking covert channels as a means of analyzing the financial implications of co-location inside various public clouds. This was done by exploiting the fact that the buses lock themselves.

Inter VM-Communication, or Communication Between VMs and Hosts

Isolation is the primary benefit that may be gained from using virtualization. In the event that the settings for this feature are not adjusted appropriately, it has the potential to pose a risk to the cloud infrastructure. The virtualization feature known as isolation ensures that applications running on one virtual machine (VM) do not disrupt or otherwise interfere with those running on another VM. When it comes to virtual machines, isolation means that breaking into one virtual machine should not allow access to its co-located virtual machines within the same environment or even to its underlying host machine. This is an important point that should be carefully observed, so pay attention to it.

A appropriate software that enables data to be exchanged between virtual machines (VMs) and the host is a shared clipboard that is contained within a virtual machine.

This application may also be regarded as a gateway for the transmission of data between several malicious apps that are running in VMs. It is possible for it to be used to "exfiltrate sensitive data to/from the host operating system" in the worst possible scenario.

Through the use of virtual terminals, the VM layer of certain VM technologies is able to keep a log of screen changes and keystrokes. This log is used to successively issue the appropriate authorization to the kernel of the host operating system. Because these collected logs are maintained in the host, hosts have the ability to view these logs, including those that pertain to encrypted terminal connections that are taking place within the VMs themselves.

Some virtualization circumvents isolation. The fundamental concept that underpins this line of reasoning is that it should be possible to run apps that were designed for one operating system on another operating system without making any adjustments. The proposed technique makes improper use of the security carriers that are present in both operating systems. They give the virtual machines unfettered access privileges to the underlying host's resources, including file systems and networks, in such a way that there is no isolation between the host and the VMs. This occurs in systems in which there is no physical separation between the host and the VMs.

This type of assault is known as a guest-to-guest attack, and the communication that occurs between virtual machines (VMs) is referred to as guest-to-guest communication. In this scenario, the attackers use one virtual machine (VM) to access or control another virtual machine (VM) through the use of the same hypervisor. It is possible to carry out these assaults without compromising the

hypervisor layer. Shared memory, network connections, and other shared resources are all points of entry for a malicious virtual machine to gain access to other virtual machines [91]. For instance, if a malicious virtual machine (VM) discovers where the memories of other co-located VMs are situated, it may then be able to read from or write to that area, which may then cause it to interfere with the activities of other VMs. An attacking virtual machine (VM1) that is carrying out a guest-to-guest assault is able to get access to subsequent guests (VM2 and VM3). The attacker could or might not have permission to access other virtual machines (VMs), but in this example, it accesses VMs without permission [37].

Survey of the state of the art

In this section, we will survey the state-of-the-art relevant work in the field of cross-VM attacks, specifically cross-VM network channel attacks for the purpose of data leakage or to elevate the privilege level of a non-root user.

Vulnerabilities in Network channel

Access to cloud services is gained through the utilization of ordinary networking protocols, such as Internet Protocol (IP), which are regarded as being unreliable [65]. The Internet Protocol, sometimes known as IP, is the protocol used to transmit data from one computer to another. For the purpose of communication, an IP address is given to each system. The Internet Protocols itself contain a number of flaws, some of which are going to be covered in the next section. The usage of the same Internet Protocol (IP) address by several users can sometimes result in those users gaining access to the resources of other users [33], [32]. The Address Resolution Protocol,

sometimes known as ARP, is a protocol that is used to map an IP address to a physical machine address that corresponds to it [92].

Within the context of cross-VM Address Resolution Protocol (ARP) assaults [93], the attacking virtual machine (VM) initiates an ARP spoofing attack by fabricating an identical IP address within the target VM and then sending an ARP packet to the virtual router. When the faked ARP packet is received, the virtual router will immediately update the routing table. As a consequence of this, any traffic that was supposed to be routed to the target VM is instead forwarded to the attacking VM, which can then choose to either modify the traffic or perform sniffing on it. The bridge functions as a virtual hub when the network configuration mode is set to bridge [93]. The virtual hub that allows communication with the network is shared by all VMs. By utilizing a sniffer tool like Wireshark, an attacker VM is able to monitor the activity on the virtual network [94]. When operating in router network mode [93], a router assumes the function of a virtual switch and makes connections to each VM through the use of a specific virtual interface.

Poisoning of the Address Resolution Protocol (ARP) [95] is another example of a vulnerability that is regarded to be well-known for Internet protocols [96]. Due to the fact that ARP does not need Proof-of-Origin [33], a malicious VM is able to reroute all of the inbound and outbound traffic of a co-located VM to the malicious VM by utilizing this vulnerability.

Return Oriented Programming

Real-time assaults, often known as ROP attacks, come in a variety of forms and can exploit ROP systems. On the application layer, Adobe announced that a significant

vulnerability had been present in the Adobe Flash Player 10.0.45.2, as well as earlier versions of the software [97]. The use of ROP is required to exploit these vulnerabilities, which can also be found in Adobe Reader and Acrobat 9.x. This vulnerability, as a consequence, bypasses data execution prevention (DEP) [98], which is a security system that is implemented by Windows. As a consequence of this, the entire system is compromised, and it is possible for an attacker to take control of the system that has been compromised [99, 100]. Rootkits based on ROP have been implemented at the kernel level of the Windows operating system. When these rootkits are executed, they are able to successfully conceal harmful processes, files, and network connections through Windows. These rootkits are constructed using ROP techniques, which allows them to avoid being detected by kernel integrity protection technologies like SecVisor [101]. The ROP approach can be used to exploit the Apple iPhone, in which an unauthorized user installs applications or exposes a customer's SMS database [102]. This can be accomplished by exploiting the fact that the ROP technique is available. Cloud assaults that are already in existence are outlined in Table 2.2.

Analysis

In this section, we took a look at cross-VM assaults to determine which areas of research are the least covered by this thesis. [32, 93, 103]. Different methodologies, such as ARP spoofing, sniffing the virtual network, and ROP, have been utilized by the researchers that worked on these studies. These attacks have two goals: the first is to reroute the network traffic of a compromised VM, and the second is to explain how an unprivileged VM can use the ROP approach to successfully modify the code

of the hypervisor, so increasing the privilege level that it possesses. The security perimeter of cloud computing prohibits such attack scenarios by establishing an additional layer of separation between virtual machines (VMs) or by preventing the execution of arbitrary code. Despite this, each and every one of these attack tactics has a few drawbacks.

The most important things that were discovered about the general techniques are summarized as follows: there are a number of prospective pathways of research channels that have not been explored yet. Researchers have not shown any possibility for privilege escalation by applying ROP in conjunction with exploiting the network channel, nor have they shown any indication for the redirection of network traffic by exploiting the network channel through an impersonation and mirroring approach. Researchers have also not suggested any indication for the redirection of network traffic. Exploitation of network channel is the primary focus of this body of study for a multitude of reasons, including the following: (i) It is possible that it has the maximum potential to reroute the real-time network traffic of a target virtual machine (VM), as well as the potential to raise the privilege level of an unprivileged VM. (ii) Previous work merely redirected the network traffic and increased the privilege level by utilizing classic methods such as ARP spoofing and the 'ret' command, both of which are straightforwardly defeatable in virtualization environments. The methodologies that have been suggested offer an in-depth investigation of the procedures that are involved as well as the consequences, in terms of both quality and quantity, that they have in a variety of contexts. (iii) If this strategy is successful, the outcome will be so powerful that it will be able to control the toolstack and redirect network traffic to a concealed point. Additionally, it will be able to handle other virtual machines.

Table 2.2: Overview of Cloud Attacks.

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

Countermeasures

Reducing Side/Covert Channel Leakage

Cloud providers have a few different options available to them when it comes to putting up a defense against side-channel cache attacks. The first technique for preventing cache sharing partitions the cache into distinct areas that are dedicated to individual virtual machines (VMs) or applications. This can be accomplished

through the utilization of either software or hardware-based methodologies. In terms of software, a number of academics have erroneously applied the page coloring strategy to cache partitioning [104], [105].

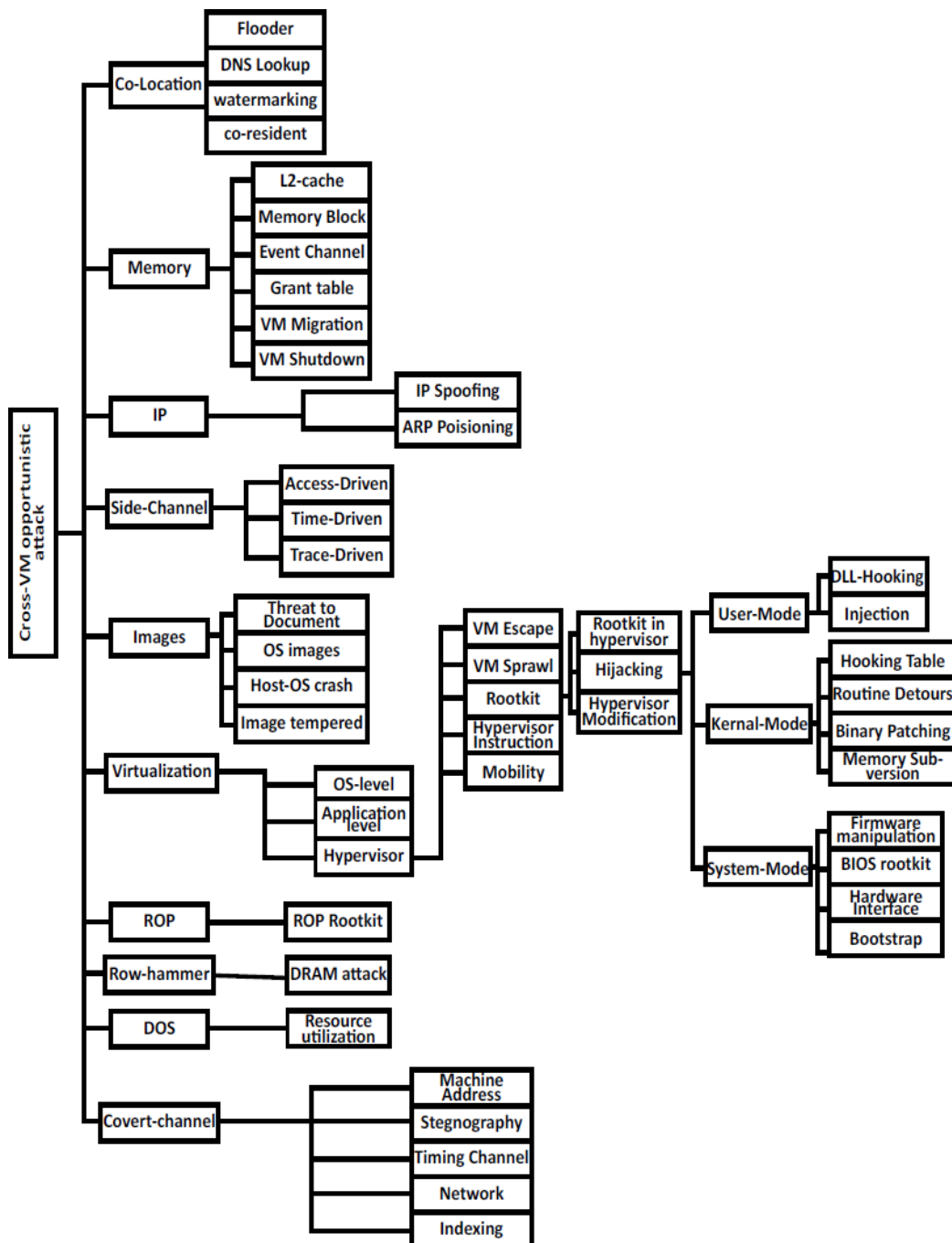


Figure 2.6: Anatomy of Cross-VM Attacks

The application known as STEALTHMEM was developed by Kim and colleagues [106], and they utilized it to partition the Last Level Cache (LLC). It provides each virtual machine with a certain amount of covert pages, which can't be modified in the cache. The researchers Liu et al. [107] took use of Intel's Cache Allocation Technology in order to prevent the attacker VM from exchanging sensitive data with the victim VM while the LLC was in session. In order to defend against side-channel attacks that are caused by cache exclusions, new hardware caches have been developed that partition the caches by ways or sets [108].

The second strategy incorporates randomization into the design of the system so that malicious actors do not gain any knowledge that could be beneficial to them. In the case of software, the measurements of the system clock are muddled up in order to disrupt the monitoring that the attackers are performing [109, 110]. Zhang et al. [111] launched Duppel, which adds noise to an attacker's observations while also periodically executing cache cleaning throughout VM's operations. New caches have been developed for the hardware that randomize memory-to-cache mappings as well as cache fetching [112, 108, 113, 114]. [112]; [108]; [113]; [114]

Optimizing Resources

The cloud provider is able to optimize resource consumption between multiple domains and decrease performance interference, both of which can help mitigate DoS assaults generated by contention for resources. In order to alleviate memory conflict, the method partitions memory resources among a number of different domains, such as the Intel Cache Allocation Technology [115]. In the event that there is contention for input or output (I/O), the cloud server is able to monitor and

control the bandwidth of I/O traffic in order to prevent resource depletion. Direct Device Assignment is another option that can be used by it to avoid I/O encroachment between each virtual machine [116]. These technologies have garnered widespread support from suppliers of public cloud storage. PAD was proposed by Li et al. [117] as a secure data center for power attacks in an over-subscription situation. This was in reference to the power resource. PAD will generate a virtual battery pool, which will then enable load sharing and control the amount of power that is utilized by each rack. In order to prevent a failure caused by insufficient power consumption, it has the ability to detect and cut power surges.

Protection of VM images

It is essential to protect VM images stored in the cloud application store and to eradicate any potential security flaws that may exist in the publishers or retrievers of those images. Mirage is an image management system that was established by Wei et al. [118] to deal with the many approaches of addressing the concerns regarding the security of VM images. (1) Access control: In order to keep a tight eye on who is retrieving the photos, Mirage implements access permission in the form of two distinct types: check-out and check-in. (2) Image filters: In order to protect critical information in the original photographs, Mirage employs specialized filters that can be applied to them.

Network Defenses

"CAPTCHAs" were implemented into the protocols by Georgiev and Shmatikov [119] in order to differentiate between human users and malicious machine scanners.

Eliminating the Hypervisor's Vulnerabilities

In previous research, a variety of approaches to strengthening the security of hypervisors were investigated. The first strategy involves the creation of brand new secure hypervisors. TrustVisor is a miniature version of the hypervisor that was described by McCune and colleagues [120]. Its purpose is to secure the dependability of sensitive data and codes stored within applications. In this, a new secure guest mode has been developed for running applications that are processed on the x86 hardware architecture. This mode supports virtualization in order to impose tight memory separation between the host operating system, the hypervisor, and any VM applications that are running. In order to carry out the integrity attestation process, TrustVisor further started a software micro-TPM instance for each application. An open-source eXtensible and Modular Hypervisor Framework was designed, configured, and validated by Vasudevan and colleagues [121]. [Citation needed] (XMHF). The XMHF is made up of a variety of XMHF main libraries in addition to a few XMHF associate libraries. In order to implement the hypervisor application's preferred security measures, the XMHF core and the fundamental functionalities offered by this framework can be extended by the hypervisor application.

The second strategy that might be taken in this regard is to protect the hypervisors' data integrity. At runtime, Wang et al. [122] presented HyperSafe, a lightweight approach for verifying the integrity of Type-I hypervisors. HyperSafe was developed by Wang and his colleagues. Write Protect (WP) is a bit that is implemented by HyperSafe in order to protect hypervisor pages from being corrupted by malicious

applications. As a means of implementing the control flow of the hypervisor program at runtime, HyperSafe enables the configuration of a target table that includes all acceptable destinations for indirect control flow instructions. This table may be used to implement the control flow. HyperSentry was developed as a tool for calculating the integrity of the hypervisor while it is running by Azab and colleagues[123]. The Intelligent Platform Management Interface (IPMI) can be used by a remote client that needs to validate the hypervisor in order to initiate the server into the System Management Mode (SMM) and measure the hypervisor's code, data, and CPU state. This is done by initiating the server into SMM. Reducing the rights and capabilities of the hypervisors is another potential path to take.

During the runtime of the virtual machine (VM), NoHype [124, 125] is intended to remove the hypervisor, which will result in a decreased attack surface caused by the hypervisor. This is accomplished by NoHype through the pre-allocation of processors, cores, and memory for each VM during the VM formation process, as well as the allocation of virtualized I/O devices directly to VMs, which eliminates the requirement for a hypervisor to perform I/O emulation. Additionally, it alters the guest operating system such that it stores the settings of the host system in a cache for subsequent use. Self-service cloud computing was presented by Butt et al. [126] as a method for restricting the rights of the host virtual machine. It does this by dividing the administrative privileges between a virtual machine (VM) that is system-wide and administrative VMs that are client-specific. The system-wide administrative VM is unable to investigate the client VMs in any way, be it their

code, data, or computations. On the other hand, the per-client administrative VMs

are able to carry out certain privileged system functions within their respective VMs. In this scenario, security and privacy are maintained even in the event that the host virtual machine (VM) is breached.

Avoiding Co-location

One of the most important aspects of virtualization is the utilization of common underlying infrastructure. One method for removing the risks associated with using shared infrastructure is to minimize or eliminate the possibility of co-residency between several virtual machines (VMs). Launching a static virtual machine (VM) or a dynamic virtual machine (VM) migration is one way to accomplish this goal. During the process of launching virtual machines (VMs), some cloud providers offer clients the option to select dedicated VMs. In this configuration, a customer's VM has exclusive usage of a dedicated server and does not have to split resources with any other VMs. In addition, new policies for the deployment of virtual machines have been revised, which has led to a reduction in the likelihood that an attacker and a victim will share the same virtual machine [127, 128]. Some academics have suggested that virtual machines (VMs) be migrated often in order to make it more difficult for attacking VMs to co-locate with the VMs they are assaulting [129–131]. ***Defeating***

Row Hammer Attacks

Cross-VM Defending oneself against row hammer attacks can be accomplished by the implementation of either hardware or software solutions. Error-Correcting Code, sometimes known as ECC memory, is a type of memory that can be used in hardware to assure the correctness of one single-bit error and to detect two-bit faults.

This makes it significantly more difficult to execute row hammer attacks [132]. Two

different approaches were proposed by Brasser et al. [133] with regard to software. The first possible remedy is to modify the system bootloader so that it can identify pages of vulnerable memory. Offline execution of row hammer exploitation tools is required in order to ascertain which memory pages may be vulnerable to row hammer attacks [134]. The boot loader will then signal that the potentially exploited memory pages are inaccessible during the boot process. This will ensure that the pages in question are not carried out during runtime. The second alternative is to extend the OS kernel in order to impose stringent isolation onto the physical memory of various system entities, such as user and kernel areas. This will allow for more memory to be available for usage by the system. This ensures that the memories of the various entities are physically separated by at least one row, which means that the memories of one creature cannot interfere with the memories of another entity. Irazoqui et al. [135] developed MASCAT, a static code analysis tool that can analyze an application's binaries and identify potential micro-architectural attacks. Some examples of these types of attacks are row hammer assaults. This tool uses an algorithm for signature-based detection to search binary files for implicit characteristics that micro-architectural attacks typically display in their design. These files are searched using a signature-based search. In order to carry out row hammer assaults, the adversary needs to consistently avoid the cache and access a predetermined spot inside the DRAM. Row hammer assaults use this as their signature move to identify themselves.

Table 2.3 provides an overview of all of these attacks as well as the relevant defenses that have been tabulated for your convenience.

Table 2.3: Attack and their Related Countermeasures

Attack	Countermeasures	Ref.
Co-location	Dedicated VMs Runtime VM migration VM launch placement	[127] [130, 131] [128]
Virtualization	Designing secure hypervisors NoHype Protecting hypervisor integrity	[120] [124] [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 Assigning user rights CAPTCHA	[137] [137] [119]
Row Hammer Attacks	Error Correction Codes Memory Isolation Signature-based detection algorithm	[132] [133] [135]
Return-Oriented Programming (ROP)	Defeating ROP Attacks	[138]

Analysis

In this section, we took a look at the various defenses available against cross-VM assaults. There is a strong connection between this idea and a few of the countermeasures [126, 137, and 138]. In these investigations, the researchers have offered various solutions, such as giving user rights, combating ROP assaults, and using specialized virtual machines (VMs). Cross-VM assaults, network channel attacks, and ROP attacks are what these solutions are designed to defend against. All of these countermeasures, however, come with a few drawbacks, such as the fact that specialized virtual machines on cloud computing are not an option for cloud

providers. This is because cloud providers reduce their operational costs by sharing the same hardware across several VMs. Similarly, attacking VMs using privilege escalation is a way to get around the requirement of providing user rights to VMs and the need to defend against ROP attacks. The researchers have provided numerous countermeasures, but there is very limited study in offering the countermeasures of varied attack techniques. This is the lesson that was learnt from the overall approaches.

Discussion and Findings

In the previous chapters, we covered cross-VM attacks and the various countermeasures that may be taken against them in a cloud computing setting. This chapter will continue that discussion. These works are broken down into the many kinds of assaults that an attacker used and their respective summaries may be found in Figure 2.6. It has been shown in Figure 2.6 that researchers put in a significant amount of effort not only to exploit shared hardware like images, memory, and virtualization but also to exploit hypervisors by means of ROP and changes in memory codes in order to elevate their privilege rights. According to the literature review and the current state of the art, it is clear that there is a clear gap in the 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. This gap is observable due to the fact that there is a clear lack of research in this area. On the one hand, the studies are totally focused on how the attacking VMs are making optimum use of or exploiting the shared resources to degrade or exploit the hardware performance. On the other hand, the

analyses completely disregard the impact that these security flaws have. On the other hand, theoretical approaches emphasize how important it is to cut down on these failures without providing any insight into the nature or scope of the problem that is being tackled. As a direct consequence of this, there is now a necessity for conducting in-depth security failure investigations in actual cloud infrastructures.

Summary

A complete literature overview of the state-of-the-art concepts of cloud computing, including their key properties, cloud computing models, security aspects, attack vectors, and countermeasures, has been provided in this chapter. It has also been demonstrated how these methods can be utilized to better research and quantify the security of cloud-based models of computer systems.

The abstraction of the model of the cloud system has been presented, and the concepts of a cloud system being composed of multiple nodes that interact with each other through various interfaces within the system environment have been introduced. These concepts were presented after the presentation of the model of the cloud system. In addition, the idea of virtualization has been dissected in great depth, as has its system support in terms of both hardware and software. Cloud computing and the various service models it offers are also extensively covered here.

The idea of cloud computing, as well as its definition, have been discussed, along with its primary vocabulary and distinguishing traits. In addition, the many aspects of cloud computing security, such as CIA, the various services, virtualization, and servers, have been dissected in great detail.

This article has introduced the idea of potential attack vectors in cloud models and has emphasized the important necessity for empirical study and modeling of cloud system security. 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 here. This review includes attacks on network channels, side channels, and co-location. In conclusion, current holes in the state-of-the-art for these assaults as well as areas where the security of cloud system models might be improved have been brought to light during this study. The solutions to countermeasures for attacks that already exist have been offered. These solutions include optimizing resources, strengthening network defenses, cutting down on side-channel attacks, and developing a defense mechanism for row hammer assaults.

Based on the findings of the analysis, we have come to the further conclusion that there is a compelling argument for future investigation into cross-VM assaults and the channels that are linked with them. In contrast to these ideas, there is a need for research that looks into cross-vm assaults from a ROP perspective. Therefore, additional study is needed to employ ROP and mimicking attacks, which will allow for the acquisition of new scientific insights through the examination of results in unconventional methods.

The following chapter takes a deeper look at the general network architecture and hypervisor architecture of cloud models, specifically focusing on what the primary components of each are. To develop further the research topics and surrounding viewpoints for this thesis, it is necessary to understand how network traffic is routed through VMs so that it can access external networks and how hypervisors create domain isolation between VMs that are co-located with one another.

Chapter 3

Architecture of Networks and Hypervisors in Cloud Computing

This chapter presents a thorough analysis of the designs of networks and hypervisors, and it discusses the insights that may be gained by observing how these systems operate. To be more specific, the primary goal of this chapter is to conduct in-depth research on the distinctive qualities and primary components of network and hypervisor architecture in the context of network traffic flow and domain isolation attributes. This will be accomplished by analysing the results of these investigations.

This chapter's overarching goal is to provide a more in-depth understanding of the cloud network system and the hypervisor in terms of those systems' primary components, how those components connect with each other, the flow of network traffic, hypervisor domains, the privilege levels of root and non-root virtual machines, and the domain isolation properties of those VMs.

Cloud System Model

This section provides an overview of the fundamental aspects of the cloud model, including its key components, the hypervisors that are utilised, the network and its isolation properties, the firewall rules, and the IP addressing methods, as well as their implementation.

Nodes of Cloud Model

The principal nodes of the cloud model are listed below.

Node Types: A hardware machine that functions with the assistance of an operating system and well-defined software that has been configured on it can be referred to as a node. The following are some node specifications, together with their respective explanations [139], at a high level of abstraction.

Controller nodes : Controller nodes are the types of nodes that are in charge of running the management software services that are required for the cloud platform to function [139].

Compute nodes : Compute nodes are responsible for putting virtual machine instances through their paces. On this node, KVM serves in the role of the hypervisor. In addition to that, the provision of firewall services falls under the purview of this node. In a given arrangement, it is possible to run more than one computing node [139].

Network: The roles of such nodes include ensuring the establishment of virtual networks, which are required for the client to create either public or private networks, and connecting the customer's virtual computers with external networks, such as the Internet [139].

Modes of Cloud Configuration

The cloud model is capable of being customised in three distinct modes, each of which is detailed further down.

The Single Node Setup cloud model supports a range of various back ends and network configuration choices, and it provides a variety of distinct configuration setups to choose from. In a setup with a single node, only one server is responsible for

running all of the services and also controls all of the virtual instances [140]. The configuration of a single node is shown in Figure 3.1.

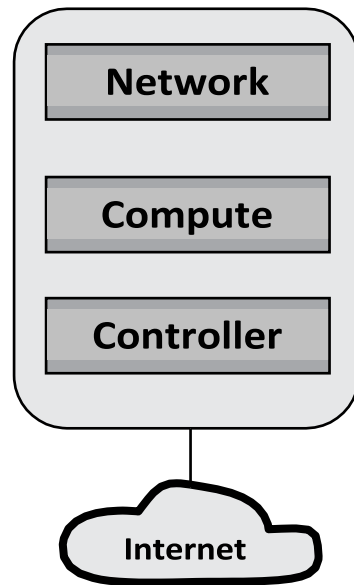


Figure 3.1: Single Node Setup

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.

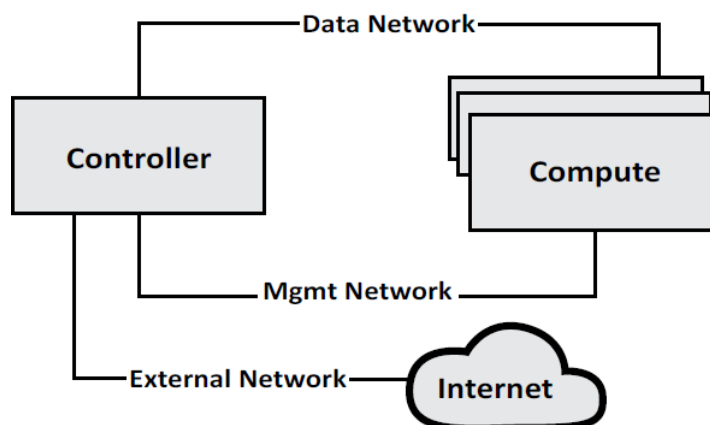


Figure 3.2: Double Node Setup

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.

Network Services in Cloud Model

The networking services that are provided by the cloud model are an independent service that configures a number of different processes across a variety of nodes. These services and processes communicate with one another as well as with other services. A neutron-server, which is a module built in Python that imports the Networking API and is responsible for delivering the tenant requests to various plug-ins for further processing, is the process that is in charge of directing the networking services.

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. Additionally, it is accountable for the implementation of the network model as well as the distribution of IP addresses to each port. Access to a persistent database can be gained indirectly through the use of the neutron-server. This is possible through plugins, which connect with the database using AMQP (Advanced Message Queuing Protocol)[142].

Plugin Agent (neutron-*-agent)

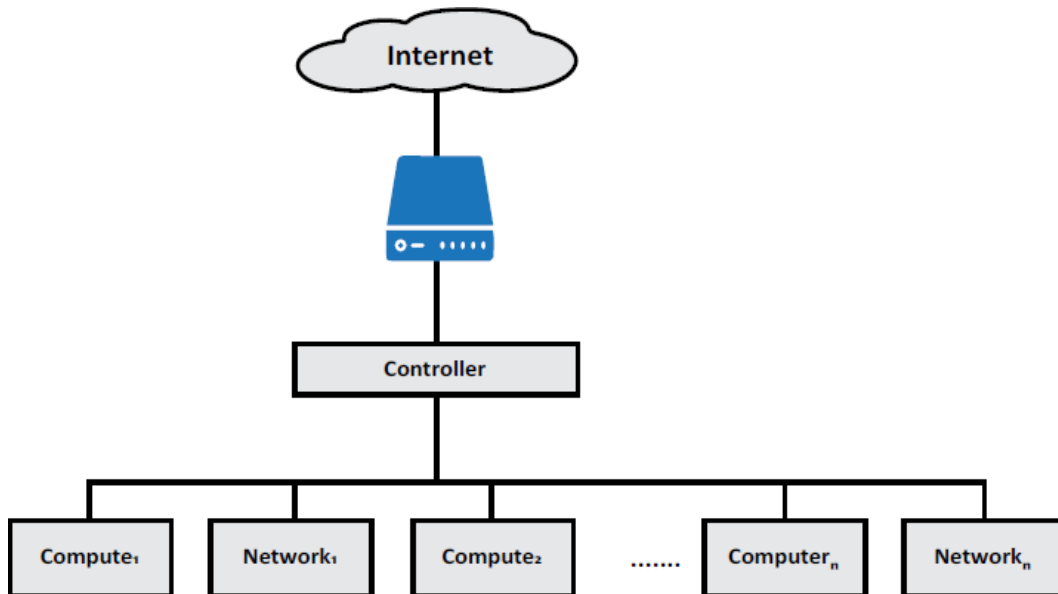


Figure 3.3: Triple Node Setup

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:

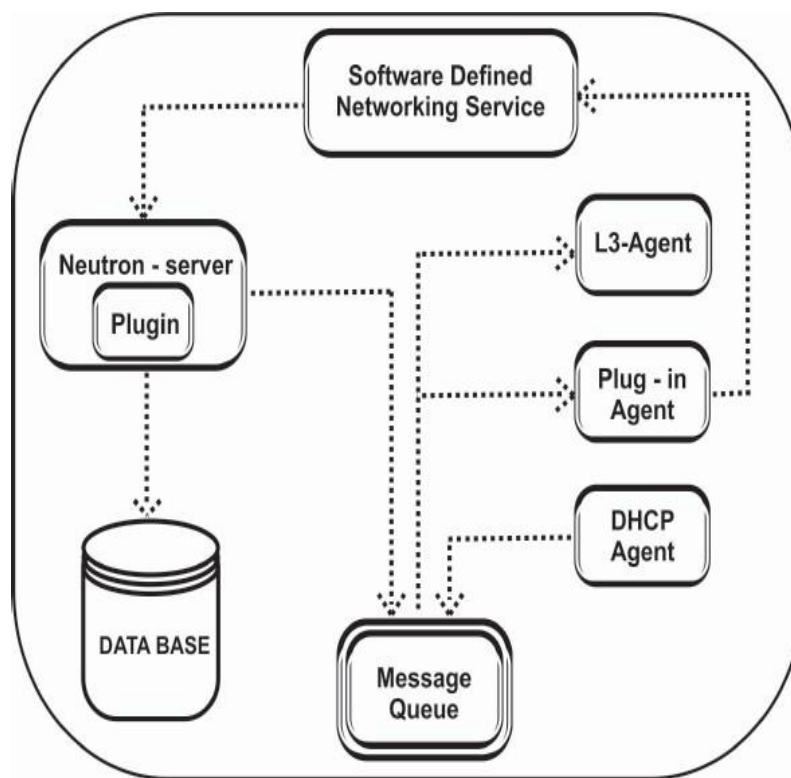


Figure 3.4: Cloud Model Networking Components

The installation of networking services for the cloud on physical servers

A controller (the system that acts as the host), a network machine, and a group of compute machines that may create and run many virtual machines (VMs) are the essential components of a conventional cloud architecture. In order to put up a

conventional networking architecture, you will need to make use of separate physical data centre networks. To name a few of these:

Management Network

By utilising this network, the components of the cloud are able to communicate with one another. The IP address that has been issued to this network is able to communicate with other nodes within the nearby data centre. Their IP addresses are not viewable from a network that is external to their system, such as the Internet [148].

Network of Guests

The communication of virtual machine data within the cloud platform is the responsibility of this network. The IP requirements of such a network are determined by the plug-ins that are utilised as well as the configuration of the network [148].

Linked to the outside world

This network makes it possible for virtual machines to connect to the Internet and other external networks. It is required that the IP addresses that are issued to this network can be accessed from any location on the Internet [148].

API network

An application programming interface (API) network is a specialised network that contains all APIs, including network APIs. The Internet should be able to access the IP addresses that have been assigned to devices in this network. The function of this is that of an external network. Constructing a subnet for an external network can also be done within this network (see reference number 148).

Figure 3.5 provides a visual representation of the Cloud Networking Services as well as their connectivity with the Management, Guest, External, and API networks.

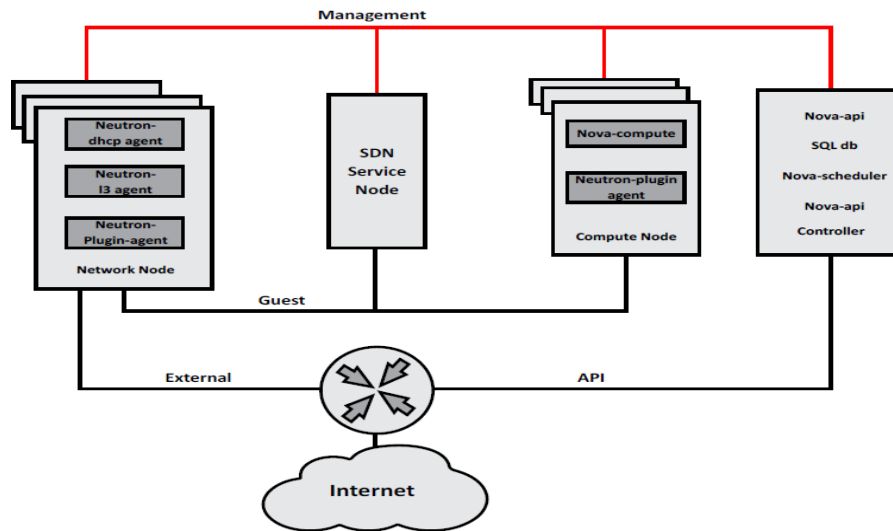


Figure 3.5: Cloud Model Networking Services

Networking Services: During the preliminary architectural phases of the Network infrastructure, it is necessary to make certain that the appropriate level of technical skill is offered in order to contribute to the configuration of the physical networking infrastructure, classify appropriate security controls, and assess appropriate strategies. This is a requirement. Tenants are given the ability to construct their own virtual networks thanks to the addition of a layer in the networking infrastructure that consists of virtualized network services. In comparison to these virtualized services, the traditional networking counterpart services have reached a higher level of development and security up to this point.

Isolation through the use of many techniques: VLANs, which adhere to the standard of IEEE 802.1Q tagging, and L2 tunnels that make use of GRE

encapsulation are the two mechanisms that can be provided by Cloud Networking in order to separate network traffic in distinct ways.

Isolation utilising VLANs and L2-tunneling: The strategy that is used to separate or isolate network traffic is determined by the options presented by the configuration of the network, as well as the choices available to take use of those options.

Virtual LANs, or VLANs for short, [149] maintain the isolation on a specific physical network by using IEEE 802.1Q headers with a predetermined value for the VLAN ID field (VID). The traffic on VLAN networks that share the same physical networks is segmented and isolated from the traffic on the other VLAN networks. Each every physical network that provides access to VLAN networks is regarded as a separate VLAN trunk that has its own set of distinct VID values. VID values are valid from 1 to 4094. The OpenStack network needs will determine the degree of difficulty involved in configuring the VLAN network. OpenStack Networking requires a proper VLAN range to be allocated as well as each compute node physical switch port being allocated to a VLAN trunk port before it can make efficient use of virtual local area networks, or VLANs.

L2 tunnelling: Network tunnelling encapsulates network combination with a unique "tunnel- id" that is used to recognise the network traffic linked to that network combination [150]. L2 network connectivity is not dependent upon the physical zone or underlying network configuration. [Citation needed] L2 network tunnelling: Traffic on a network can be made to flow through Layer-3 borders and avoid preset VLANs and VLAN trunking if it is encapsulated inside an IP packet and sent along the network. When data traffic on a network is tunnelled, an additional layer of

obfuscation is added. This decreases the reflectiveness of individual tenant activity from the perspective of someone monitoring the network. OpenStack networking is capable of supporting both advanced networking capabilities, known as GRE and VXLAN encapsulation respectively. The scope and scale of networks are two important considerations that should guide the approach choice for L2 isolation configuration. Tunneling is recommended in situations in which the network infrastructure will have a large number of L2 networks or will only have access to a limited number of VLAN IDs. In any of these cases.

Additional Services for the Network: Isolation of the network Provide a description of how the network security and the security boundary are implemented. The following network services are available as an additional layer of protection for the design of the network, but they are not required.

Access Control Lists: When setup with standard nova-network services, the compute node supports VM network traffic access controls [151] natively. Using iptable, traditional security groups in nova-network are configured to apply to all of the interface ports (virtual or otherwise) on VMs. The security perimeter gives administrators and virtual machines (VMs) the ability to identify the sort of network traffic and the directions (traffic travelling towards inward or outward) that are permitted to move through a virtual interface port. It is strongly suggested that the security groups be enabled in this service model before attempting to apply the networking services.

L3 Routing and NAT: Cloud networking makes use of routers that are equipped with the capacity to connect many networks. These routers can also provide a

gateway that connects multiple private networks to a shared external network, such as the Internet. Network Address Translation (NAT) [152] capabilities are supported on ports (gateway) that connect the router to an external network by L3 routers [149] in a networking environment. This router supports floating IPs and SNATs (Static NAT) for all traffic by default. SNATs are a form of static network address translation that link a static one-to-one mapping between a public IP address on an external network and a private IP address.

Firewalls (FWaaS): The ironic set of security characteristics requires careful management and application in order to be effective. FWaaS is responsible for providing these characteristics, which are typically more extensive and comprehensive than the security features given by security organisations. [153].

Restricted Capabilities of the Networking Services

The following is a list of the constraints that are commonly associated with cloud networking services.

IP addresses Overlapping: In the event that a physical system (the host) does not support multiple namespaces, it is necessary to operate networking services such as DHCP and L3 agents on separate hosts. The fact that there is no segregation between IP addresses that are established by L3 agents and those that are created by DHCP agents is the primary motivation for this idea. In the event that support for network namespace is unavailable, the L3 agent has another constraint in that it can only handle a single router.

Neutron agents are limited in the following ways: Forwarding was implemented in a lot of plug-ins by using the neutron-l3-agent library, but the neutron agents don't

support IPv6 forwarding. [154]. OpenStack Networking does not provide any services that would facilitate any flavour of NAT when used with IPv6, which is the primary limitation of neutron agents in the case of IPv6 forwarding.

During the process of constructing the network, each of these networking services and plug-ins are utilised. For the purpose of conducting a study, the network architecture of cloud computing has been constructed so that various services and plug-ins are utilised extensively.

Architecture of the Cloud Network

Cloud providers offer extremely comprehensive security groups and rules, which may be applied to a broad variety of factors, including the incoming and outgoing traffic of users, virtual machines (VMs), containers, and apps. To provide the highest level of protection possible, individual virtual machines (VMs) can have their own security groups configured and dynamically deployed. Enterprises have the ability to react appropriately and respond to risks in a timely manner thanks to the control and flexibility of security groups. Each virtual machine (VM) has an IP address that is associated with a Vif. This IP address is only visible to the virtual switch, where all virtual machines (VMs) are linked to br-int, i.e. part of open virtual switch (OVS), using a TAP device that is formed immediately upon connection request. TAP stands for "tape-attached port," and it is an access path that allows network traffic to go from VM to VSwitch. Figure 3.6 demonstrates that the OpenVSwitches denoted by br-int and br-ex are the devices that are accountable for managing ingress and egress requests, respectively. Open-VSwitch is a type of virtual switch that is integrated into the hypervisor. It is capable of carrying out the duties of a Layer-2 switch and provides a variety of functions, including Access Control List, VLAN, and a great

deal more besides. VMs can take advantage of the subnet or private network that it provides [155].

A virtual switch functions in a manner analogous to that of a virtual network interface, which is created by joining one or more virtual Ethernet interfaces. A additional connection is made between the virtual switch and a virtual router using a Veth pair. The creation of a link between VSwitch and vrouter is under the purview of the Veth pair. Maintaining the routing table for incoming and outgoing traffic flow is one of the vrouter's primary functions. Routers are designed with a stringent setup of the firewall (FWaaS) that is responsible for implementing the securityperimeter. The packet is then sent onward by the router to its subsequent hop, which is on the external bridge known as br-ex. A physical network interface known as eth0 is housed within the external bridge. This interface is responsible for finally passing network packets on to an external network such as the Internet.

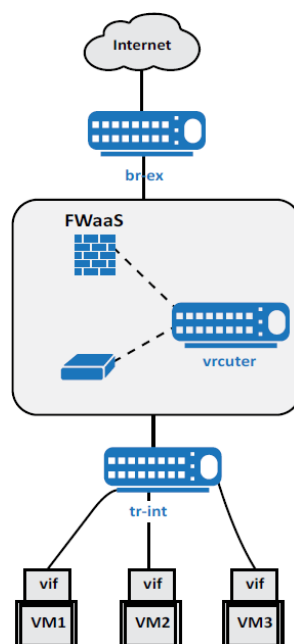


Figure 3.6: General Cloud Architecture

Vulnerability Discovered in the Architecture of Cloud Networks

Individual network components and the functionality of those components, in order to determine the vulnerability using a top-down approach [156], have been explored. This topic was covered in Section 1.4. A virtual machine's (VM) virtual interface (Vif), which is connected to a tap device, is where an outgoing packet begins its journey. The TAP device is connected to a bridge device and serves as an integral component of the network. The bridge, also known as br-int, is the central component of the system to which all VMs are connected through TAP in turn. The br-int is responsible for performing VLAN tagging and untagging on the traffic that is arriving from and going to VM. To ensure that the network traffic from the several virtual machines that are hosted on the same physical hardware is kept separate from one another, a unique VLAN ID is given to each network. Virtual Machines cannot be given direct access to the network bridge if the security setup is in place. It makes it possible for VM to connect by way of a TAP device, as demonstrated in Figure 3.6. Therefore, the only method to attack or access the network is via compromising the TAP. This is also the only way to do either. Therefore, we study the TAP, their properties, and their connection with VM and br-int simultaneously, and the vulnerability that we find in the TAP is as follows: The cloud networking system makes it possible to link a virtual machine's TAP interface with a vswitch even if the backend of the switch does not have a private Ethernet interface. This makes it possible for VMs to communicate with one another and access the internet.

Following the discovery of this vulnerability, an investigation was conducted into

the network designs of various cloud models and the cross-cutting vulnerabilities that were discovered in the OpenStack and Oracle Ravello network systems, both of which share the same network architecture.

Oracle Ravello Networking in Addition to OpenStack

OpenStack and Oracle Ravello offer security policies and groups that may be applied to parameters on incoming and outgoing user traffic, VMs, and containers. These features are available through OpenStack and Oracle Ravello. It is possible to construct and apply security groups dynamically, which will result in improved protection. Some examples of security groups include rules for a firewall, SSH control, and port bounding. Because of their adaptability, security groups enable businesses to react rapidly to a variety of threats. Figure 3.7 illustrates OpenStack's network architecture, which is carried out in practice by use of the following four virtual networking devices:

An external monitoring device known as the Test Access Point (TAP) sits in-between the physical Ethernet card and each virtual machine (VM) that is hosted on the physical machine.

A Veth Pair is a virtual network cable that connects a Linux bridge to a virtual bridge running on a physical machine. This cable is known as the Veth Pair.

The Open Virtual Switch, often known as OVS, is the virtual bridge that is in charge of managing the flow of traffic coming into and leaving the network. It performs the duties of a Layer-2 network switch, delivering a variety of Access Control List and Virtual LAN (VLAN) features to VMs, in addition to the capabilities of delivering subnet or private network functionality. Each virtual

A virtual machine (VM) generates and saves data on an associated virtual network interface card (VNIC), such as eth0. After that, the data is sent to the TAP that is located on the compute host. The typical function of a TAP is to provide an access point to data that is moving through a network. In addition, the TAPs are connected to the Linux bridges, which are responsible for transmitting the data to Veth Pair, which represents one side of the cable. Information that is transmitted to one side of Veth Pair can be received at the opposite end of the network. The other half of the pair can be found on the integration bridge, which is abbreviated as br-int. This bridge is in charge of the attachment of all of the VM's TAPs as well as the attachment of any other bridges that are present on the system. As can be seen in Figure 3.8, the integration bridge also makes a connection with the br-eth network.

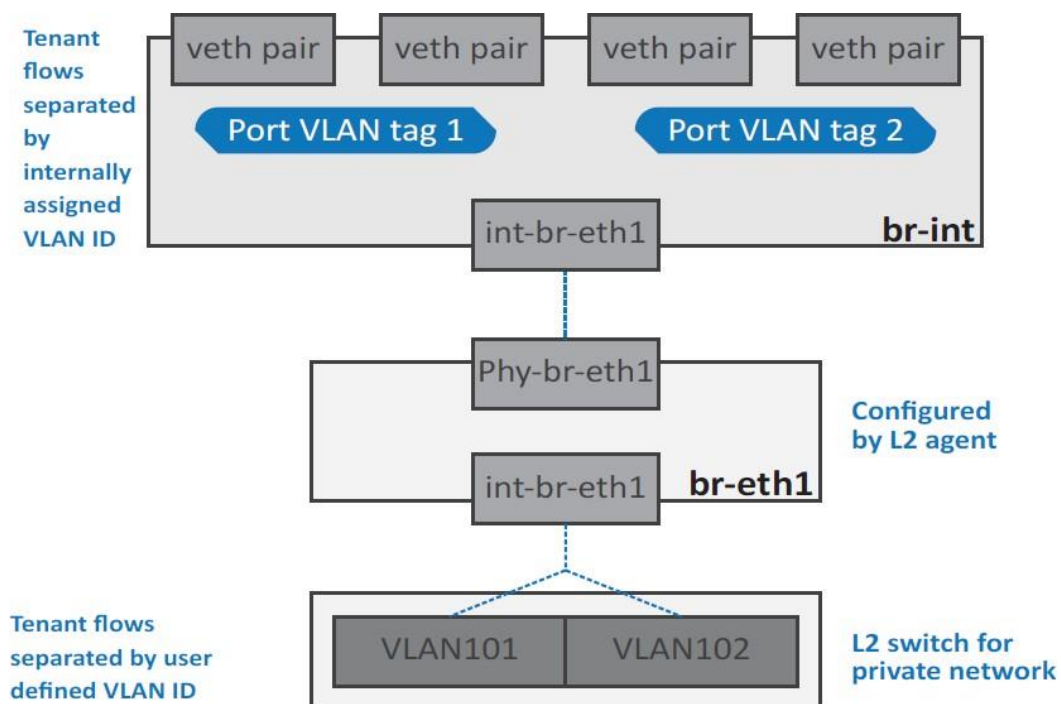


Figure 3.8: Function of *br-int* and *br-eth*

Validation of the Hypothesis As a proof of concept, we exploited the same vulnerability in Google and Microsoft Azure; nevertheless, we were unable to successfully complete the task. Because the successful exploitation of the vulnerability is contingent on the presence of an illegal TAP connection with the bridge, which is not granted in these cloud models due to the unique ways in which their internal network devices are configured, the vulnerability cannot be successfully exploited. Before allowing a connection of a VM's TAP interface with vswitch, these cloud models make certain that a connection of a VM's privateEthernet Interface with TAP has been established. They will only allow the connection to be established if the TAP is linked to the Ethernet.

Architecture of the Hypervisor

In the section that follows, the domain architecture of the Xen hypervisor will be explained.

XEN

Xen is a concrete hypervisor that operates using a microkernel technique. It provides functionalities that make it possible for numerous virtual machines (VMs) to run in parallel on the same physical computer hardware. It offers a high level of isolation between the several virtual computers that can coexist on the same physical hardware. It is licenced under the GNU General Public License version 2 (GPLv2) and is managed by Xen.org, which is a group that works across industries. Xen is composed of a variety of products and ongoing initiatives. 3.9 presents an illustration of the Xen architecture.

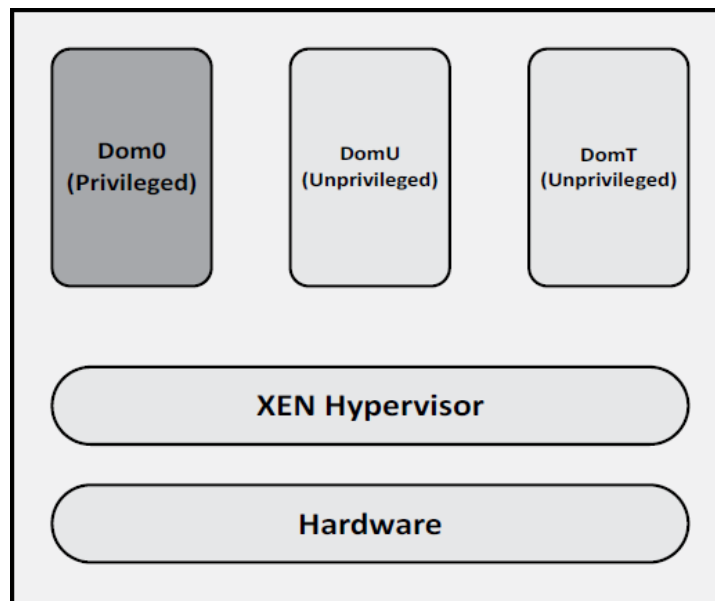


Figure 3.9: Xen Hypervisor Architecture.

The Xen Project runs in a more privileged state of the CPU than any other application currently installed on the computer. Memory management, CPU scheduling for virtual machines, and the launch of the most privileged domain are the primary responsibilities of this component (dom0). Dom0 is the privileged domain, and only the virtual machine has access to the hardware directly. It possesses the authority necessary to control any and all domains that are not privileged (domUs).

Domains with privileges and domains without privileges

On Xen, it is possible to run a variety of virtual machines, also known as VMs, as well as domains. Following the loading of Xen by the bootloader at the beginning of the boot process, Xen then loads its primary domain, known as dom0. Dom0 is the only virtual machine that can contact the hardware and respond to device requests because Xen does not maintain any device drivers. Additionally, it is the only

administrative domain that is accountable for the creation, pausing, unpausing, saving, and eradication of other virtual machines (VMs) that are hosted on the same physical hardware. In contrast to this, other domains that dom0 launches are known as domU, which stands for unprivileged domain, and they do not have admin privileges of any kind. A domain structure of the hypervisor's code contains a Boolean value store, which is used to handle the data for the domain privilege. In the case of dom0, it is set to 1, whereas for domUs, it is set to 0. If you need to escalate the privileges of domU, you will need to change the assigned value for this domU from zero to one [103]. This will allow you to do what you need to do.

Xen Vulnerabilities

It is possible for a non-root user to become a root user of their own domain and get the ability to further subvert their resources if they have the wish to increase their resources in the event that the cloud expands. The use of return-oriented programming (RoP) makes the growth of the cloud model conceivable; nevertheless, RoP leaves the system open to assault. As was covered in section 1.4, we attempted to uncover a vulnerability in the RoP of the cloud system by implementing several ways in an iterative manner. However, due to the stringent cloud security settings, we were unable to do so.

Because of the stringent security setups of modern cloud models, we have come to the conclusion that exploitation of ROP by itself does not present a vulnerability. Since a result, we combine the exploitation of ROP with network channel and determine that it is vulnerable, as the attacker can use this vulnerability to initiate an unlawful network connection with the root user. Therefore, following the successful

(illegal) connection, the attacker is able to elevate the privilege level of its VM by breaking the domain separation of hypervisors and controlling ToolStack, from which it is able to manage other VMs. This allows the attacker to administer other virtual machines. As illustrated in Figure 3.10, we analyse whether or not this vulnerability is exploitable within the OpenStack architecture. Xen serves as the underlying virtualization technology.

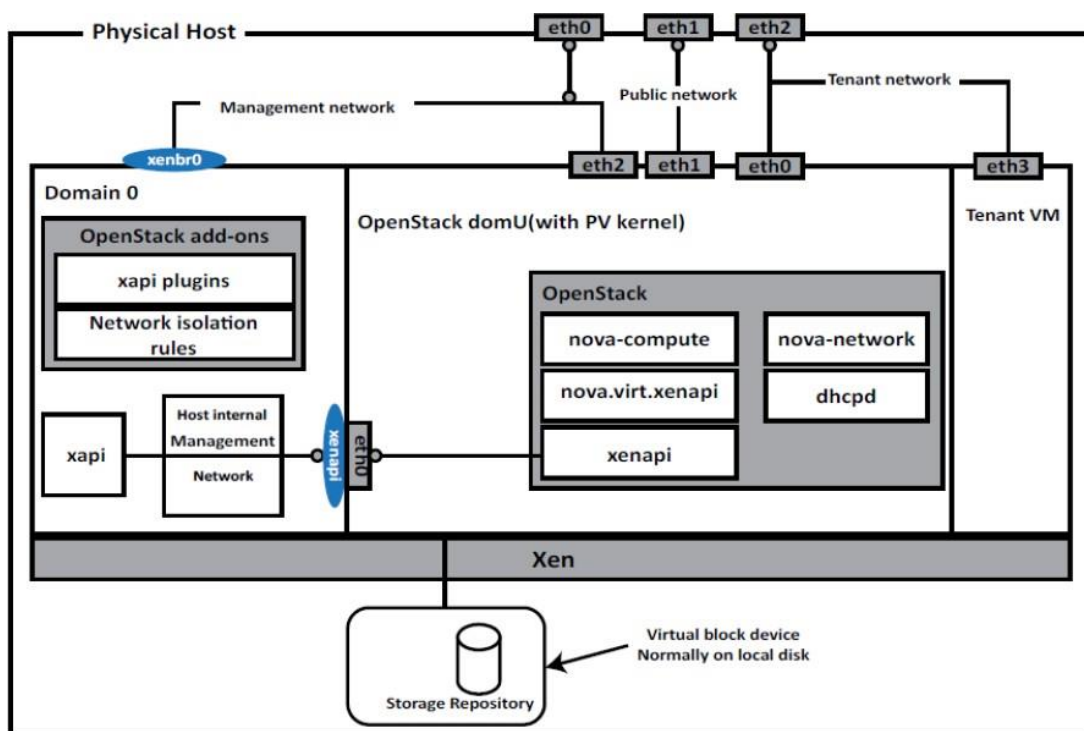


Figure 3.10 shows the architecture of OpenStack Xen.

The following is a brief summary of OpenStack with Xen architecture and its domains, which are referred to as Domain 0 (dom0) and Domain U (domU): **Domain U**

Each guest VM in a virtualized environment runs its own copy of the operating system as well as its own applications. The hypervisor is capable of supporting two distinct types of virtualization modes, which are referred to as Hardware- assisted or

Full Virtualization (HVM) and Para virtualization (PV). Both of these modes are discussed in section 2.1.2. Each sort of guest can be employed concurrently on a single hypervisor without any problems. It is also feasible to apply techniques used for Para virtualization in an HVM guest, effectively forming a continuum between PV and HVM. This can be done through the use of hypervisor migration. This methodology is referred to as PV on HVM. Virtual machines that are running as guests are completely separated from the underlying hardware; in other words, they do not have permission to directly access either the hardware or the I/O functions. Therefore, another name for them is "unprivileged domain" (or DomU).

The Area Under Your Control (or Domain 0)

Domain 0 is a specialised Virtual Machine that is capable of accessing the hardware directly, managing all access to the I/O functions of the system, and interacting with the other Virtual Machines. These are some of the particular administrative privileges that this Virtual Machine possesses. In addition to this, it is in charge of managing a control interface that connects to the outside world and is used to control the system.

Domain 0 is the first virtual machine that is started when the system boots up.

Console Domain 0 is comprised of a specialised management tool known as control stack, which is also known as Toolstack. This tool enables a user to handle the creation, destruction, and configuration of virtual machines. Console Domain 0 is also known as Domain 0.

OpenStack Nova

It executes the XenAPI library code of the domU process in order to communicate with the xapi process of the dom0 process. Python is the language used to write both sets of codes. Without ever leaving the host, it travels all the way from domU to dom0 by way of the communication route provided by the Host Internal Management Network. Communication from dom0 to dom0 is accomplished through the usage of library code that is XAPI to XAPI. This is typically utilised in the event that there is a cloud expansion.

XAPI

A Xen-based hypervisor can be managed by Toolstack using XAPI, which is the essential component of Toolstack. In the world of KVM, the function of libvirt is comparable to that of XAPI's. API for XAPI was supplied by XenAPI. Because the compute driver in OpenStack communicates with xapi, it is possible to use any and all xapi servers with OpenStack [157].

API that was offered by XenAPI XenAPI. API. Python's standard library, which functions as a xapi client, also includes Xenapi [157].

XenServer

XenServer is a virtualization platform that is open source. It enables all of the capabilities that are necessary for any server or datacenter to be implemented, including the Xen hypervisor and the XAPI for management [157].

Analysis

Several conclusions have been determined as a result of the investigation of the

architectures that have been covered in this chapter.

First, after doing research into the architecture of the network and the flow of traffic in cloud computing, it was discovered that there is a significant risk to cloud computing if we did not prevent the penetration of any external device into the network. This was a discovery that was made. The unauthorised connection of a TAP device with an internal network vswitch was identified as the vulnerability that was present in leading cloud computing. The attacker can gain access to the cloud computing network system over this link and enter the system. After conducting research on a variety of cloud architectures in order to find this vulnerability, our team discovered that both OpenStack and Oracle Ravello are susceptible to the assault. As a proof of concept, we evaluated the identical vulnerability in Google and Microsoft Azure, but we were unable to find any evidence that it was executable in either of those environments.

Secondly, various vulnerabilities have been found in the Xen hypervisor architecture in the event that the cloud expands. The cloud model can be expanded with the help of ROP. However, the attacker can acquire control of the root domain by utilising ROP in conjunction with the network channel to do so. Once in control of the root domain, the attacker can then manage other co-located VMs.

In order to investigate the network architecture, the work that is presented in this thesis concentrates mostly on the OpenStack and Oracle Ravello cloud systems. OpenStack and Ravello provide compelling solutions today for the challenges of delivering flexible infrastructure for high-performance computing (HPC) and high-throughput computing (HTC). In addition, the development community is rapidly

expanding services to meet exponential future demands for these types of computing.

Summary

OpenStack's underlying system model is discussed in this section, one of the top cloud platforms, as well as the Oracle Ravello System were covered. This chapter will also cover the presentation of the key components that make up various cloud models. There have been multiple configurations of OpenStack demonstrated in detail, each of which demonstrates the connectedness of nodes despite having a distinct set up. The network channel, in addition to the services that run on networks, was also studied in considerable length. Virtual machines are required to maintain isolation while using shared hardware by virtue of these networking services. The implementation of security features in the cloud was mostly the responsibility of networking services such as VLANs, Access Control Lists, firewall rules, and network traffic routing.

This chapter can be broken down into two distinct but equally important parts: the analysis and the learning. In order to choose a cloud architecture, these processes are carried out in a manner that is both incremental and iterative. The examination of various cloud deployment configurations, which are covered in greater detail later on in this chapter, makes up the bulk of the analytical phase. Following this, the learning phase will make use of the selected cloud deployment methods in order to construct a model for the prediction of system performance.

This chapter also includes an explanation of the default cloud network architecture, which is typically adopted by cloud providers at higher levels. Nevertheless, the internal system level configurations of each provider are distinct from one another.

The network architecture of OpenStack and Oracle Ravello cloud systems has also been extensively discussed, elucidating the central networking nodes and the paths taken by the network traffic of several co-located virtual machines.

It is up to the hypervisor to keep numerous virtual machines (VMs) completely isolated from one another. As a result, the study of hypervisors is also essential for readers who wish to understand what the primary functions of hypervisors are and how they maintain isolation between root and non-root virtual machines (VMs). What are the constraints placed on them? It has also been demonstrated how the Xenhypervisor can be configured to work with OpenStack, which illustrates the connection that can be made between various APIs and root and non-root VMs.

This chapter delves deeper into the security perimeter configurations of cloud network architectures, the steps required to locate the vulnerability in cloud network architecture, the cloud models that are susceptible to this vulnerability, and the configurations of XEN architecture and the location of their vulnerability.

Chapter 4

Cross-VMN Channel Attacks Using Spoofing And TapImpersonation: Defenses And Countermeasures

As was covered in chapter 3, network planning is an essential part of cloud computing, and virtual machines that are co-located must have their network traffic routed through various devices that are configured according to the architecture of the network. When companies are considering moving their operations to the cloud, one of their key concerns is the protection of their customers' data and privacy. This study's objective is to analyse the consequences that virtualization has on network security vulnerabilities in cloud computing and to report its findings. There is a risk of side channel attacks when a virtual machine shares its hardware environment with other virtual machines that are physically located in the same physical location. The existence of such problems has been brought to light by a number of researchers, including how side channel attacks on shared hardware enable attackers to exfiltrate sensitive data between co-resident virtual machines (VMs). These side channels perform the function of a security gateway between the users and the attacker. If a malicious user is able to successfully exploit these side channels, they will have the ability to spy on the data of other virtual machines. As a consequence of this, cloud providers make an effort to handle this kind of problem by ensuring the logical segregation of resources using an internal virtual network that connects numerous virtual machines (VMs). The use of virtual networks is critical to guaranteeing that

one virtual machine will not disrupt the operation of other virtual machines (VMs) running on the same host.

At this time, there is a significant dearth of empirical research that investigate network assaults and the various countermeasures and solutions available to defend against them. The current work is entirely focused on IP, ARP sniffing, and spoofing, all of which have the potential to compromise the co-located virtual machine (VM) network traffic by redirecting it to some destination point; however, such attack techniques have been controlled by countermeasures. The currently available works do not utilise network design by means of an internal regular device that is responsible for the transmission of data and the connection of nodes. In addition, the methodologies that are currently in use do not thoroughly research the utilisation of network isolation in cloud computing, nor do they offer a comprehensive way of network traffic analysis that is able to quantify and extract empirical findings. Researchers and providers of cloud services may find this to be of practical use. This chapter offers the study and analysis of attack design by leveraging the network architecture of cloud computing. The following are the significant contributions that this study and analysis has made: This internal virtual network, in turn, exposes further vulnerabilities in virtualization by mounting cross- VM assaults on a network channel. This finding is the most important contribution of this research, and it indicates that this is the case. This chapter presents the introduction of a novel network channel attack, in which a malicious virtual machine can redirect the network traffic of a victim virtual machine running on the same physical hardware. The attack is carried out by launching a novel network channel.

In order to launch such an attack, it is necessary to first overcome the challenges posed by traffic redirection, then pre-empt the traffic of the victim VM, then enter the existing network infrastructure, then launch an impersonation attack on the interface, and finally remove any footprints left by the attack. This research addresses these challenges and demonstrates the proposed attack on two IaaS cloud platforms: OpenStack, which is an open source IaaS management system, and Oracle Ravello Systems, which is a public IaaS provider. Both of these cloud platforms have multiple setups that are configured with security requirements. In addition to our findings, a countermeasure option has been presented as a possible response to the attack that was reported.

Introduction

The term "cloud" refers to a virtualization platform that functions as a computing model [158] that is built on the Internet. Clouds can take on many different configurations, including grid, distributed, virtualization, utility, and parallel. A user can easily use the cloud's utility services and request access to its resources such as storage, networks, computation, and apps on demand. This is made possible by the cloud's easy design. It presents the idea of "pay as you go" (also known as "pay as you use") [159]. There is no requirement to purchase one's own resources. Instead, one can simply pay to access the resources and then augment or reduce those resources according to their needs. It possesses a number of useful characteristics, including resource pooling, on-demand self-service, broad network access, measurable service, and quick flexibility, among others. The cloud makes it possible

for multiple service models to exist, such as Software as a Service (SaaS),

Infrastructure as a Service (IaaS), and Platform as a Service (PaaS) [19]. These service models were previously covered in chapter 2.

The hypervisor, which is located at the very top of the operating system, is responsible for controlling the virtualization environment and the network layers [19]. The infrastructure of cloud computing is dependent on the hypervisor. In most cases, a hypervisor will have a somewhat limited interface. There are two distinct varieties of hypervisors. The types I and II are as follows: A Type-I hypervisor, also known as a native or bare metal hypervisor, only requires a few lines of code on the host operating system. A Type-I hypervisor is illustrated by the Xen virtual machine. The Type-II hypervisor, often known as a hosted hypervisor, operates within a traditional operating system environment. Examples of a Type-II hypervisor are VMware and Virtual Box [160]. [Citation needed]

Virtualization technologies of the modern era, such as HyperV [161], Xen [60], and VMWare [162], are swiftly becoming the backbone of the security architecture for cloud computing systems. Because the cloud is virtualized, many virtual machines (VMs) can coexist on the same piece of physical hardware. The high isolation between co-resided virtual machines is one of the primary reasons for their desirability [10]. This means that a guest virtual machine that is operating on the same system as other guest VMs cannot disrupt the operation of the latter. Every virtual machine (VM) has its own specific limitations and bounds. Strong isolation, which is one of the fundamental features of public cloud computing platforms like Microsoft Windows Azure [163], EC2 [164], and Rackspace [165], helps to increase the systems' overall level of security.

The creation of an internal virtual network by a virtual machine manager (VMM) [166] is what ensures logical isolation between virtual machines (VMs) in virtualization systems. Despite this, the possibility of an assault still exists. Numerous recent research (some of which are mentioned in Section 2.6) have shown both how co-residency can be achieved and how an adversary can benefit from shared hardware in cross-VM environments. These findings are described in more detail in that section. Researchers have discussed in [93] and [26] how an attacking VM can potentially access other VMs by means of network connections, shared memory, and other shared hardware resources. They have also discussed how an attacking VM can exploit network channel to redirect the network traffic of other co-located VMs and what their limitations are in the current cloud model. [93] and [26] also discuss how an attacking VM can exploit network channel to redirect the network traffic of other co-located VMs.

In spite of the fact that cross-VM attacks have the obvious potential to abuse shared memory and disc space, as mentioned in Sections 2.6 and 2.7, there have been no actual demonstrations of a fine-grained cross-VM network-channel attack. This is despite the fact that such an attack could be performed. The majority of the challenges that are discussed centre on the fact that cloud providers place additional layers of isolation between co-resided virtual machines (VMs) in comparison to non-virtualized settings. This is due to the fact that the attacker and victim are frequently assigned to separate segmentation of virtual networks [46]. The suggested attack, on the other hand, will stealthily reroute the network traffic of the victim VM, making it difficult to identify because it will leave behind few or no traces on the network.

The cloud network model was given in Chapter 3, and its purpose was to uncover vulnerabilities in the current state-of-the-art design of the cloud model's network. According to the findings that have been provided, there are various technological obstacles that need to be overcome in order to take use of these vulnerabilities. Their implementation has been presented in the well-known open-source cloud model OpenStack as well as in the commercial cloud model Oracle Ravello system. This was done in conjunction with a description of how to address the challenges involved in exploiting these vulnerabilities, which we discovered in Section 3.2.1.

Statement of the Problem

Multiple virtual machines are using the same network resources, which means that these resources are open to attack. This was covered earlier. The exploitation of these vulnerabilities in real-time settings becomes more difficult as a result of the stringent security configuration in the cloud network paradigm. This chapter addresses these security concerns and builds a zero-day assault model (as indicated in Section 1.3.1, research goal 1), focusing on the major components of the underlying network architecture, the technique, and how network components are connected, processed, and used for the model.

Ultimately, the goal is to achieve the stated research goal. The creation of a real-time system for OpenStack and Oracle Ravello serves to validate the primary ideas that comprise the new attack model. In order to evaluate how effective the proposed assault model actually is, a number of different tests were carried out. The purpose of this was to demonstrate that it would be feasible to implement the system in a working environment. In the conclusion, a defence approach (similar to the one

described in Section 1.3.1, research target 3) for preventing attacks of this nature is also offered.

Technical Challenges

The suggested attack establishes a network channel assault, as was shown in a laboratory testbed; this attack allows a malicious VM to covertly monitor the network traffic of victim VMs by taking advantage of the isolation that exists between VMs. Demonstration of the proposed attack utilising OpenStack and Ravello cloud systems as a case study to explain how to deploy the mirror in the internal medium of the network channel to optimise location. After positioning a mirror, divert the target virtual machine's network traffic to the destination. Network channels—specifically cross-VM information leakage owing to network resource sharing—are the main entry sites for such an attack (e.g network bridge).

A previous assault [167] on network resources redirected the network traffic of a target virtual machine by using address resolution protocol (ARP) spoofing.

Table 4.1: Comparison of Related Work and Proposed Work

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

Table 4.1 provides a brief comparison of the different assaults that already exist, along with our own proposed method. The details of these assaults are included in the second column of the table.

Attack Setting and Challenges

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.

OpenStack [172], a cloud computing platform, is the primary focus, especially when deployed on modern computer architectures. The experimental setup is influenced by both public clouds like Amazon EC2 and Rackspace, and other OpenStack use cases. Co-resided virtual machines (VMs) are hosted in centralised data centres using a cloud hypervisor, as is the case with many virtual network solutions. Another typical scenario involves isolating and partitioning operating systems into multiple, independently functioning parts with varying degrees of access and security [173, 174]. Qubes [175] is one such system; it is an open-source OS designed to be deployed in a cluster of virtual computers using a hypervisor. Regarding the hardware architecture, we aim for today's multi-core CPUs. This choice was mostly motivated by the processors now in use in public cloud services like Amazon Web Services and Microsoft Azure. Both the attacker and the victim are thought to be on different network domains, and to have access to different sets of resources such as virtual central processing units (VCPUs), virtual local area networks (VLANs), and

virtual storage in this experiment. The access levels of every virtual machine are identical [176].

OpenStack's ability to keep untrusted co-resident VMs separate is a key tenet of the attack model, as is the assumption that the attacker lacks access to software vulnerabilities that would give it complete control of the physical node. Therefore, the suggested attack employs a cross-VM network-channel to reroute the victim's data packets. One such case is the actual flow of data in a network consisting of victim machines in real time. Significant difficulties arise when attempting to build such a network channel in a cross-VM environment. Each of these major hurdles has been dissected, and a strategy for traversing each has been offered. The tests mirror the various stages of the attack pipeline shown in Figure 4.1.

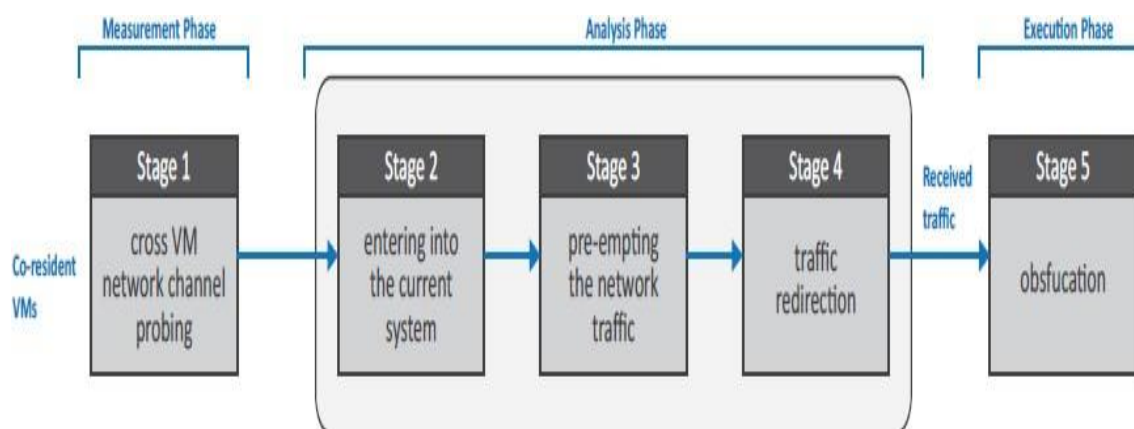


Figure 4.1: Main Attack Steps for a Network Channel

Problem #1: Keeping tabs on the Existing Network Layout

The victim VM's network communication is difficult to eavesdrop on due to the way OpenStack clouds operate by default secure setting, especially if the attacker and victim are on different network domains. For instance, there's great potential in

network channel for eavesdropping on the network traffic of victim VMs, but doing so requires some illusions to manage the network channel. Therefore, in order to eavesdrop on the victim's network traffic, an attacker must first attempt to take control of the network channel by dismantling the firewall between the two networks. This technique has been effective in non-virtualized environments where attackers are trying to eavesdrop on a victim's network traffic by leveraging the victim's virtual network. However, our proposed scheme does not advocate for such virtual network exploitation because OpenStack's default security perimeter, i.e. Firewall as a Service (FWaaS), constantly monitors any anomalous activity in the current OpenStack cloud setting and prevents the attachment of any external devices. Therefore, the greatest difficulty lies in either joining the system or gaining direct access to it. As a result, it is challenging to eavesdrop on a victim VM's network activity while FWaaS is present. The first difficulty is finding a susceptible target and researching the most effective methods of exploiting it. To initiate the suggested attack, there must be a single entryway.

In order to find a solution, it will be necessary to analyse the current network infrastructure thoroughly. Entering or becoming a part of the present system requires a set of well-executed steps. As shown by (1) in Figure 4.3, the first stage in this process is to set up a dummy interface card as a vector for network channels, which is a device that is oblivious of the network and does not have a running NIC adapter. A network card is a typical piece of hardware in a network, and its primary function is to send and receive data. It is necessary to develop a device with similar functionality, which is accountable for carrying out the fundamental network

operations that a typical device would handle. Data transmission and reception via this device have been performed as part of network activities in order to identify it within a system.

As opposed to real network interface cards, dummy cards function in a different manner. It helps computers that rely solely on an IP connection through a dial-up modem for all of their network communication. The idea behind solitary hosts is that they only need to activate a single network device, the local loopback device, which receives the IP address 127.0.0.1 by default. The local host's official IP address may be required for communication under certain conditions. Take, for example, the laptop with the hostname "test," which has been deliberately cut off from the internet so that its owner can conduct experiments on it.

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.

A fake interface is introduced now. It solves the problem by acting as a replacement for the loopback connection. In a test scenario, it would be assigned the IP address 172.16.17.21 and a host route pointing to it would be added to the routing table's entry. Then, all packets destined for 172.16.17.21 would arrive at their destination on the local network. A proper order for this stage is:

ifconfig dummy test

route add test

It is further illustrated how this dummy network interface card is coupled with other devices in order to abuse the security perimeter of the OpenStack cloud and to be a component of an existing OpenStack system. This is the initial stage that enables an attacker to gain an edge and become a member of the system of an existing system, with the assistance of further operations, in order to spy on the network traffic of victims. Disclosure of other virtual machines' private information can occur when a weakness in the architecture of an OpenStack cloud is exploited using an approach that exploits a security perimeter.

Challenge 2: Hiding the Dummy Network Interface

OpenStack's security perimeter can be fooled by changing the ordinary interface's identification to that of a TAP (Figure 4.3, (2)), which is a legitimate device within the network system, but only if OpenStack is aware that the real network card is active. The only way for our gadget to connect to the network is if we can trick TAP into thinking it is the standard interface. The TAP is passive because it is compatible with the network setup and does not affect any current settings. The TAP acts as a middleman between the VM and the network, taking the place of the vNIC and the switch. In other words, a TAP provides unrestricted, full access to all inbound and outbound data streams. Information travels back and forth between the VM's network interface card and the switch (ingress and egress). A TAP can record

transmissions from multiple sources. This ensures data is replicated and oversubscription is prevented.

The malicious virtual machine (VM) now possesses two interfaces: (i) a regular Ethernet card that pretends to be a TAP despite the fact that it does not possess a valid identity, and (ii) the dummy interface. After that, a connectivity request is transmitted to the Linux bridge, which takes it for granted that it is a legitimate TAP and adds it utilising a manner that is analogous to the standard one. Adding a Linux bridge here is necessary because the iptables security rules that are applied to this bridge are what are used to configure the security group rules that apply to the VM. When the attacker has finished this phase of the process effectively, they are now able to access the network. This impersonation can be carried out inside of OpenStack by first manually removing all types of interface identity from the network configuration file located at `/etc/network/interfaces`, and then restarting the networking services by executing the `/etc/init.d/networking restart` command, which ensures that this change is made permanent inside of the system file. Because of this, a networking device that does not have a legitimate identification will function in the same way as a TAP. Figure 4.2 illustrates the fundamental distinction between a TAP and a standard network device. The standard network device is `enp0s3`, and the tap device, `test0`, is the one that does not have a proper identity.

```

test0      Link encap:Ethernet  HWaddr ca:50:2c:ba:4f:58
           BROADCAST MULTICAST  MTU:1500  Metric:1
           RX packets:0 errors:0 dropped:0 overruns:0 frame:0
           TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:500
           RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

-VirtualBox:~$ ifconfig enp0s3
enp0s3     Link encap:Ethernet  HWaddr 08:00:27:ce:8e:88
           inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
           inet6 addr: fe80::a00:27ff:face:8e88/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:344498 errors:0 dropped:0 overruns:0 frame:0
           TX packets:160039 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:288569674 (288.5 MB)  TX bytes:10398218 (10.3 MB)

```

Figure 4.2: Difference between Tap0 and eth0

Challenge 3: Observing Network Traffic

Even though an OpenStack cloud has the robust feature of a security perimeter, there are still some weaknesses in the existing design. These flaws can be exploited by an adversary in order to spy on the network traffic of a victim virtual machine (VM). Redirecting network traffic at the attacker's destination port is the most effective method, not to mention an innovative one, for eavesdropping on the network traffic of a victim VM in a stealthy manner. This adds two obstacles in the way. First and foremost, who is accountable for the rerouting of the real-time network traffic? The second factor is the positioning, which should be such that it conceals you from others and provides the greatest possible advantage. Already existing systems [93] did not succeed in providing any aid in rerouting the traffic on the network. In order to tie network observations to the specific operations carried out by the victim, an innovative technology known as a mirroring approach has been utilised. By carrying out the activities in the following list, this obstacle has been conquered. The utilisation of a mirror provides a solution to the initial problem that the method under

consideration presents. In particular for Linux, a mirror was utilised so that this could be put into action. A powerful Linux feature known as a mirror has the capacity to change the direction of network traffic by switching it from one port to another.

The positioning of the mirror is the next obstacle to overcome. The mirror needs to be hidden from view of any other virtual machines (VMs), and its placement should be chosen so as to increase the likelihood of achieving a favourable positioning. This strategy calls for the creation of an illusion because, if the mirror is installed in any open position on the network, it can be easily discovered. As a result, the target virtual machine will be cautious when sending traffic through the interface, and it will notify the cloud administrator if it suspects that traffic is being spied upon. Setting up the mirror at the internal interface of the network bridge, which is unobtrusive to all other VMs, was necessary to overcome this obstacle. This position is the most challenging position because the traffic from all other VMs passes through this interface. However, this obstacle has been overcome. In order to configure mirror, you will need to carry out the actions outlined below at the bridge. These steps will configure the dummy interface to act as a destination port.

```
create Mirror name=test_mirror select dst-port=dummy0
```

```
set mirror @ br-int
```

This method makes use of the fact that it is possible to penetrate the network at the point from where the network traffic of other virtual machines passes, as shown (3) in figure 4.3.

Challenge 4: The fourth obstacle is the redirection of traffic at the destination point.

The activity of spying on the network traffic of the victim VM is a difficult undertaking. If an attacker is able to see the network traffic of a victim VM from a place with open network access, there is a chance that the behaviour of the attacker is being monitored by the security perimeter of an Open-Stack cloud, which will result in the cloud blocking the attacking VM. The network traffic of the victim has been redirected at the set concealed destination point, which allowed us to successfully overcome this challenge. When the network traffic of the victim VM goes via the network bridge where the mirror is setup, it will redirect the network traffic from the internal bridge port towards the set destination port. This happens when the traffic of the victim VM goes through the network bridge. It functions similarly to the installation of a mirror in that it transfers a copy of all network packets observed on one port to another port, which is then used to perform an analysis on the packets. Because of this, it is important that victim VMs and the security perimeter both remain oblivious to the fact that network traffic has been redirected. The following commands, which are shown in 4.3's (4), are the ones that are used to do this redirection within OpenStack:

```
select-src-port=@br-int select-dst-port=@dummy0
```

Challenge 5: Obfuscation

Following the completion of all of the operations, the next difficulty is to get rid of all of the traces left behind by the attack by covering up all of the devices and routes that were utilised in the initialization of this attack. Obfuscation's primary purpose is

to create confusion and divert attention away from the examination and monitoring activities being carried out [177]. route [178] is a very powerful tool that can be used with Linux as well as other distributions such as Ubuntu. It displays all of the available interfaces and static routes that exist within a network. It has been made impossible to determine the existence of used devices, as well as diverted routes from route tools and other network monitoring tools. Obfuscation can be accomplished through a variety of different methods. After the impersonated interface has been attached to br-int, a Zero will be added out to this interface. This zero eliminates the identity of the interface and is hidden from view in route tool. Because of this activity, the action that was completed cannot be analysed.

Putting it all together

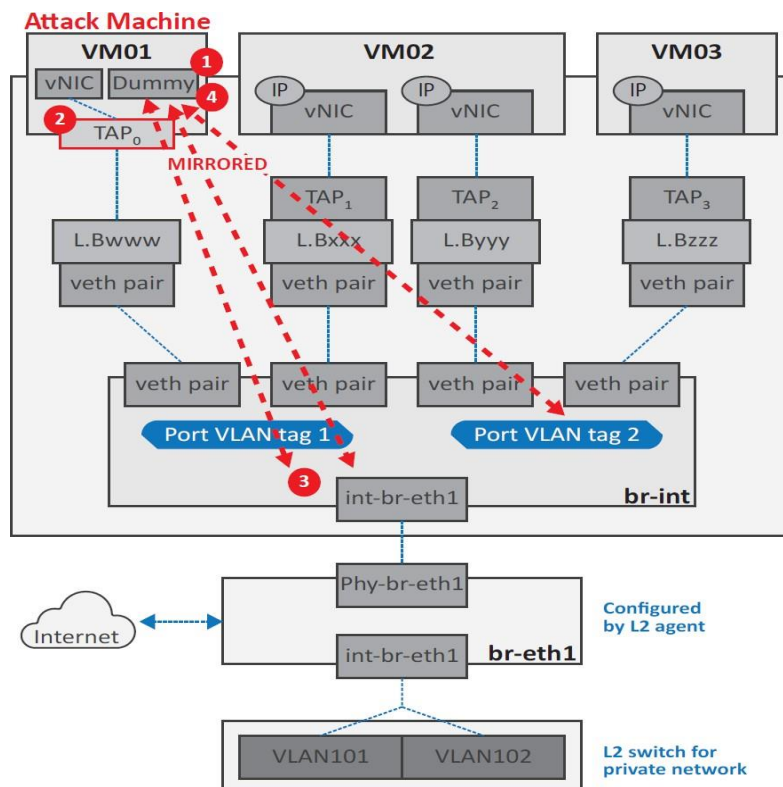


Figure 4.3: Attack Scenario in OpenStack

When we put these steps together, we may conduct a network attack that spans virtual machines, as shown in Figure 4.3. This attack takes advantage of the fact that the target's cloud provider allows them to utilise a TAP interface to connect to the Internet even when there is no private Ethernet present at the interface's back end.

Evaluation Criteria

It is clear from looking at Figure 4.3 that the research approach covered in Section 1.4 serves as the pivot point around which the assessment framework is constructed. The primary objective of the attack is to exploit the network channel in order to reroute the network traffic of co-located virtual machines (VMs). In order to accomplish this goal, the network design of OpenStack was modified to include the five different assault tactics. The evaluation results are mostly dependent on the network traffic, but they also place some emphasis on the resources that are available on the network. This is because network traffic is a significant part of the attack.

The following essential reflective qualitative features will serve as the foundation for the evaluation criteria. (i) Functional demonstrates that the aforementioned five tasks, which are crucial for the execution of attack, have been completed successfully. (ii) Expressive reflects how well the assault tactic does what it sets out to do. (iii) The overall advantages as well as the drawbacks of the attack have been demonstrated at this point.

Evaluation

OpenStack, the most popular open-source IaaS cloud platform, and Oracle's commercial Ravello System, a cloud computing service, were used in the experiment. The default security perimeter of OpenStack and Oracle Ravello cloud

model has been established. After elaborating on OpenStack's experimental setup in the previous section, this one moves on to discuss how to deploy Ravello.

Experiment Setup

OpenStack's cloud architecture utilises hardware and software components (CPUs, KVM hypervisor, and Linux kernels) that are almost identical to those in host OpenStack environments. Firstly, the configuration guarantees that no single VM is in communication with any others, and secondly, it prevents VMs from being separated from each other and from engaging in any kind of passive communication. The attacker can either wait for the target virtual machine (VM) to begin communicating before spying on the messages, or they can actively spy on the communication by rerouting the traffic. An additional scenario where the victim VM is already in communication with other VMs has been considered advantageous to the attacker. Because the configurations are so similar to real-time ones, it's also crucial to realise that the intended architecture is, in and of itself, a realistic secure setup for virtualized settings. Isolating many virtual machines in the cloud can increase their security because no single virtual machine can affect the performance of the others. The significance of these attacks is thus understood to extend beyond their potential impact on OpenStack.

Attack Scenario:

The attack is demonstrated by using a scenario in which an attacker VM (VM1) is able to listen in on and steal information from communication taking place between two victim VMs (VM2 and VM3). After that, one can use the positioning of the mirror to accomplish traffic rerouting in the desired direction. The assault was

carried out inside of OpenStack by utilising multi-node configurations that included single node, double node, and triple node environments respectively. Using the KVM hypervisor, three guest virtual machines have been set up inside of a physical computer that has an Intel Core Z Q9650 running at 3.0 GHz. VM1 was set up to be the attacker VM, and it has now been successfully hijacked; VM2 and VM3 are the ones who are being attacked. Each virtual machine (VM) has been set up with two virtual CPUs and a variety of operating systems, including Ubuntu 15.10 (VM1), cirros (VM2), and Windows 10 (VM3). These operating systems have been set up with both floating IPs and private IPs so that they can access the internet and communicate with one another, respectively. Target virtual machines had their settings adjusted so that they could send between 0 and 15 Kbps to one another. Each of the experiments was carried out an additional 20 times.

The overall resource allotment for all co-located virtual machines is presented in Table 4.2. On the other hand, the statistics of VM resource use for one week have been compiled and presented in Table 4.3. The value of resource usage that is discussed in section 4.3 is not a constant; rather, it varies from time to time depending on the VM and the programmes that it is currently running. Since network traffic does not put a burden on disc or memory, there is no direct relationship between network traffic and these physical resources. Through the use of a network monitoring tool such as mrtg[x] or prtg[x], it is possible to monitor the load on the network.

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

The number of virtual CPUs that are being used by VMs that are being operated on the physical machine is displayed in the CPU column of Table 4.2.

The MEMORY MB column displays the total amount of memory (in megabytes) that has been allotted to the VMs that are now operating on the physical machine.

The DISK GB column provides information regarding the root and ephemeral disc sizes (in gigabytes) that have been allotted to VMs that are currently operating 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

Network Setup:

By allocating IP addresses and VLAN tags within distinct ranges known to provide physical resource separation, VLAN Manager was set to assure VM isolation amongst co-resident VMs. This was accomplished by using a tag system.

VLAN Manager Setup

Each virtual machine (VM) has its own VLAN and allocated network when the VLAN mode is activated. In order for this to work, any physical switches that are placed in between must implement the 802.1q VLAN tagging standard. The

configuration described below must be used in order for the VLAN to operate properly.

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

Association of Public IPs to VMs

When a VM is first created, it is provided with its own unique private IP address by default. This particular range of IP addresses is only available inside the context of the local network. In order for the virtual machine to communicate with the outside network, it has to have a public IP address. When manually attaching a public address, there are two steps involved: (1) selecting an address from the pool of accessible IP addresses, and (2) associating the address with a virtual machine. This experimental set up needs to have a working range of floating IP addresses supplied to it so that it can be allocated. The Nova client was utilised here:

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 makes it possible to communicate with virtual machines (VMs) that have a public IP address.

Security Configuration

Iptables are utilised in networking to accomplish the functionality of security groups; however, iptables also offer linear storage and filtering. It is recommended to use ipset rather than other methods in order to enhance the performance of the security

group. Therefore, the ipset option is enabled in networking in order to increase the efficiency of security group hashes by specifying a table. An additional ipset option is added to the iptables chain whenever a new port is made available for use. The member of the security group that the port belongs to is added to the ipset chain if the security group contains any rules that are also shared by other groups. If you use ipset to change the membership of a group, the corresponding iptables rules will be modified rather than refreshed. As a result, a new virtual machine (VM) security group has been established after conducting a manual search for newly created security group names.

Managing Security Groups

On the nova-compute host that is responsible for the execution of virtual machines (VMs), security groups are established. This enables the host machine to be protected by limiting access to it and preventing intrusion from other VMs that are running on the same host. Port 22, which is the default port for security groups, has now been used to launch a security group. Two stages are necessary to complete the design of a security group. The first stage involves defining a group with the nova secgroup-create command. ii Establishing guidelines for the group by use of the nova secgroup-add-rule command:

Only traffic that is compatible with the rules of the security group is allowed to pass via the entrance point. In the event that no rule is satisfied, all other traffic will be blocked.

Only traffic that is compatible with the rules of the security group is allowed to exit the building. In the event that there is no rule stated, all outbound traffic is blocked.

When a new security group is created, rules that allow or deny any and all ingress and egress traffic are automatically added to the group.

Assumption

The most important presumption made in our developed attack model is that an adversary has successfully taken command of a virtual machine (VM) that is located on the same physical machine as the target VM [56]. Researchers have already proved the effectiveness of this control by utilising a network-based strategy to launch a co-location attack within a public cloud such as Amazon EC2 [56]. This attack was successful. They demonstrated that an adversary is possible to launch several instances of VMs within the same geographical region as the target VM, and they applied a variety of different approaches to determine whether or not a VM is effectively co-located. The following is a rundown of some of the systems that can be used to help determine the location:

By using a trace-route application such as TCP SYN to find the first hop of network traffic between the attacker and the target VM (for example, Dom0 in the host Xen server), you can discover the first hop. The discovery of this software suggests that an attacker has achieved success in locating the co-location of the target virtual machine (VM) if there is an identical Dom0 IP address.

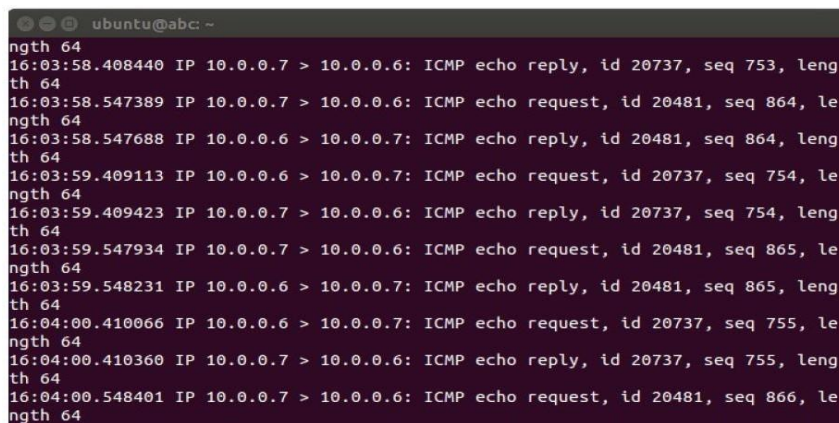
Examine the round-trip time (RTT) [179] of the network packet that travels from the attacker to the target virtual machine. A shorter RTT number suggests that the two virtual machines (VMs) are hosted on the same physical machine.

Verify the internal IP addresses of both the attacker and the target virtual machine. When the two virtual machines' internal IP addresses are numerically close to one another, it indicates that they are most likely hosted on the same physical server.

Analysis of Result

In this section, an in-depth evaluation of the experimental outcomes of the entire strategy through a real-world situation is described. The evaluation focuses on the recording of network traffic and the consumption of network resources by virtual machines (VMs). A definitive conclusion regarding the viability of the underlying attack plan cannot be reached based solely on the one outcome that was obtained. Because of this, the efficiency of the proposed method is evaluated by recording network traffic and determining how effectively network resources are utilised.

Taking Samples of Network Traffic An examination of the collected network traffic that was taken from the testbed is carried out in order to validate the attack technique. Each iteration of the experiment produces redirected network traffic that is annotated with the reality of the situation with relation to the existence of an attack.



```
ubuntu@abc: ~
ngth 64
16:03:58.408440 IP 10.0.0.7 > 10.0.0.6: ICMP echo reply, id 20737, seq 753, length 64
16:03:58.547389 IP 10.0.0.7 > 10.0.0.6: ICMP echo request, id 20481, seq 864, length 64
16:03:58.547688 IP 10.0.0.6 > 10.0.0.7: ICMP echo reply, id 20481, seq 864, length 64
16:03:59.409113 IP 10.0.0.6 > 10.0.0.7: ICMP echo request, id 20737, seq 754, length 64
16:03:59.409423 IP 10.0.0.7 > 10.0.0.6: ICMP echo reply, id 20737, seq 754, length 64
16:03:59.547934 IP 10.0.0.7 > 10.0.0.6: ICMP echo request, id 20481, seq 865, length 64
16:03:59.548231 IP 10.0.0.6 > 10.0.0.7: ICMP echo reply, id 20481, seq 865, length 64
16:04:00.410066 IP 10.0.0.6 > 10.0.0.7: ICMP echo request, id 20737, seq 755, length 64
16:04:00.410360 IP 10.0.0.7 > 10.0.0.6: ICMP echo reply, id 20737, seq 755, length 64
16:04:00.548401 IP 10.0.0.7 > 10.0.0.6: ICMP echo request, id 20481, seq 866, length 64
```

Figure 4.4: Traffic Capturing at Attacking VM

In this actual-time scenario, the two targeted virtual machines (VMs) (VM2 and VM3) are pinging each other. Figure 4.4 illustrates how the attacker machine VM1 can monitor the communications between the targeted VMs by employing an attack mechanism similar to that described in Section 4.2. Figure 4.4 shows an ICMP echo reply being sent to the attacking virtual machine. The ping command makes use of a protocol called Internet Control Message Protocol (ICMP). Success in exchanging "echo" messages between two virtual machines indicates two-way communication has taken place.

Identifying sender and recipient communication between target VMs is crucial to the success of the attack (e.g., packet header source IP address). The attacker will be confused as to which VM traffic originated from which other VM if they are all communicating at the same time.

Network Resource Utilization

When an assault is carried out, this results in the production of network traffic within each virtual machine. The VM network traffic before, during, and after the assault is depicted in Figure 4.5. The results of the experiment show that the attacking VM1 generates random network traffic not dissimilar to that produced by VM2 and VM3 when the time for the experiment ranges from 0 to 30 minutes (point A). At 25 minutes (point B), the attack begins, during which it is observed that the attacking VM1 uses a significant amount of network traffic in comparison to the VMs that are the targets of the attack, and it continues to do so for the remaining 6 minutes until the attack is over (point C). This significant rise is due to the fact that the attacking

VM is now receiving all network traffic destined for the target VM and redirecting it through itself.

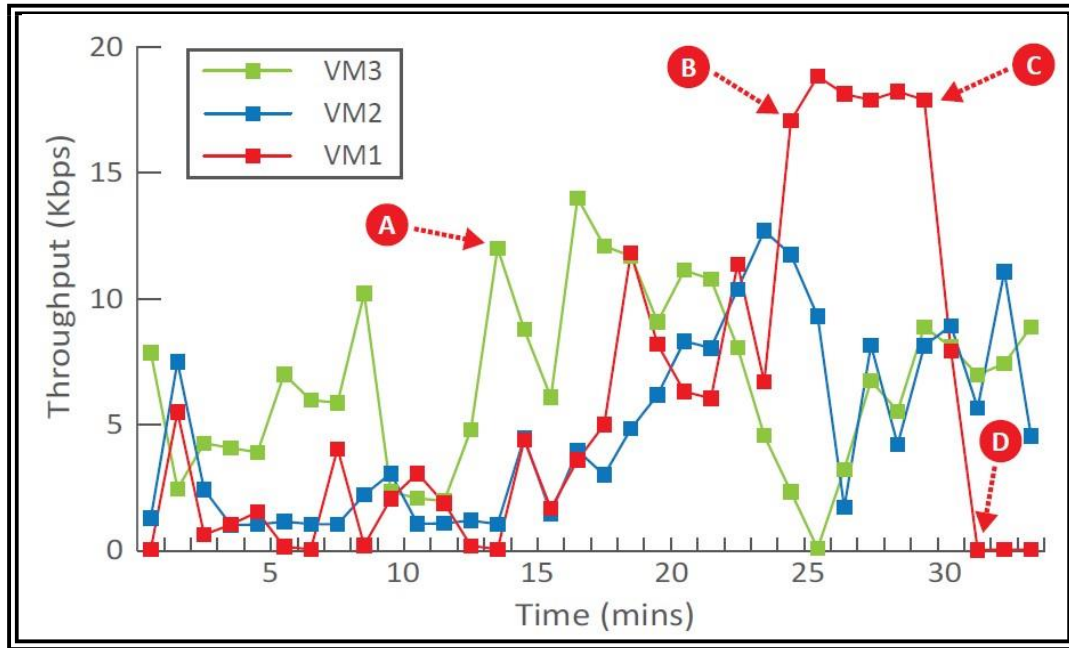


Figure 4.5: Normal Co-residing VM-Network Traffic

When the cloud administrator observes the network traffic of VM1 in relation to that of other VMs for the entirety of the experiment, it is probable that they will identify this as abnormal behaviour due to the abrupt rise in the amount of network consumption. This could result in further study or the implementation of a straightforward countermeasure to restrict virtual machine network traffic that exceeds a certain threshold. On the other hand, it is considered that in the setting of cloud computing, a countermeasure of this kind would be difficult to detect. To begin, the utilisation of virtual machine (VM) resources is regarded as a mystery by the provider in many public cloud environments. This is considered to be standard practise so long as a customer's resource capacity requests are not violated by resource demands made by the customer. Second, even if the system is making an effort to monitor irregular resource patterns by employing bandwidth monitoring

tools such as prtg [180], the countermeasures will most likely include a time delay for determining irregular resource patterns. This is because the system is unable to distinguish between normal and abnormal resource patterns. As a result, it's possible that an attacker only needs a few minutes to accomplish what he set out to do on a virtual machine that he's hacked. For instance, in 2008, the defence solution of a system that was being run by the Georgia Government during an HTTP attack [181] became active for a period of five minutes after the attack had been initiated. Last but not least, if the attacker virtual machine (VM) is capable of create cyclical network patterns prior to an attack, as shown in figure 4.6, then it is much more difficult to notice unusual patterns of traffic on the network.

Network intrusion detection systems (NIDS) [182] deployed at strategic points within the network can monitor traffic to and from all sources on the network as a viable countermeasure technique in such circumstances. It compares traffic over multiple time periods to identify assaults. The alert is delivered to the administrator after unusual activity is detected on the network. Figure 4.6 depicts an attack executed by the attacking machine that follows a resource pattern similar to that seen before during the first two peaks and the third peak (point A to B). This means that the unusual traffic pattern will go undetected. This situation calls for the administrator to ideally analyse all incoming and outgoing data; nevertheless, this could cause a bottleneck and slow down the network.

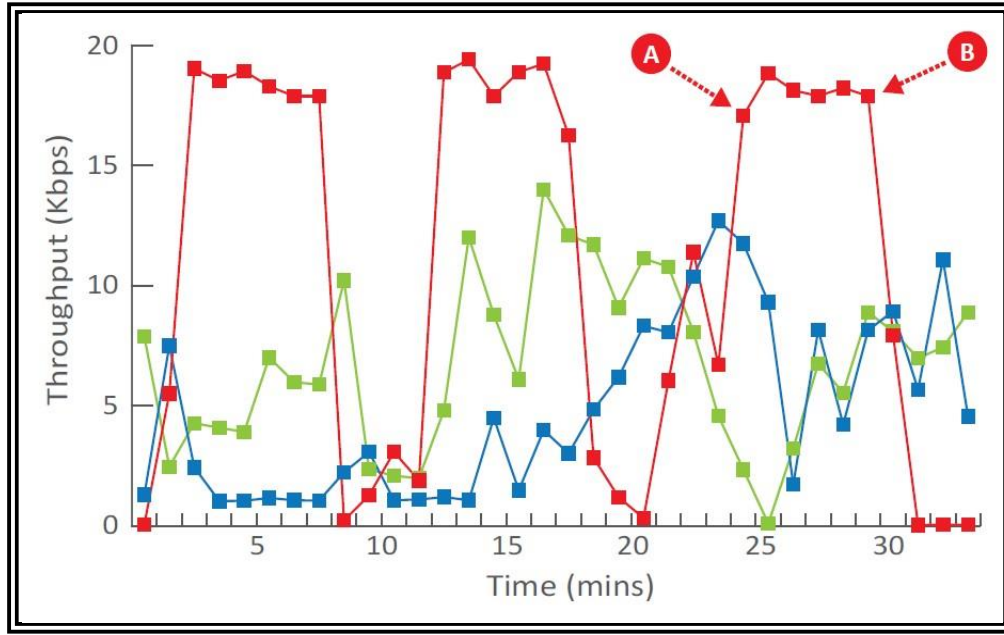


Figure 4.6: Cyclic Attack Pattern of Network Traffic

When evaluating the attack, a more extensive network system is taken into consideration, as depicted in Figure 4.7. Within the scope of this scenario, ten VMs have been taken into consideration. When an assault is carried out, this results in the production of network traffic within each virtual machine. The VM network traffic before, during, and after the assault is depicted in Figure 4.7. The results of experiments conducted over a period of time ranging from 0 to 30 minutes indicate that the attacking VM1 generates random network traffic that is not distinct to that generated by target VMs (point A). At 25 minutes (point B), the attack begins, during which it is observed that the attacking VM1 uses a significant amount of network traffic in comparison to the VMs that are the targets of the attack, and it continues to do so for the remaining 6 minutes until the attack is over (point C). This significant rise is due to the fact that the attacking VM is now receiving all network traffic destined for the target VM and redirecting it through itself. For the purpose of

capturing the network traffic of all VMs, the "post routing traffic graph" (prtg) [180] is utilised.

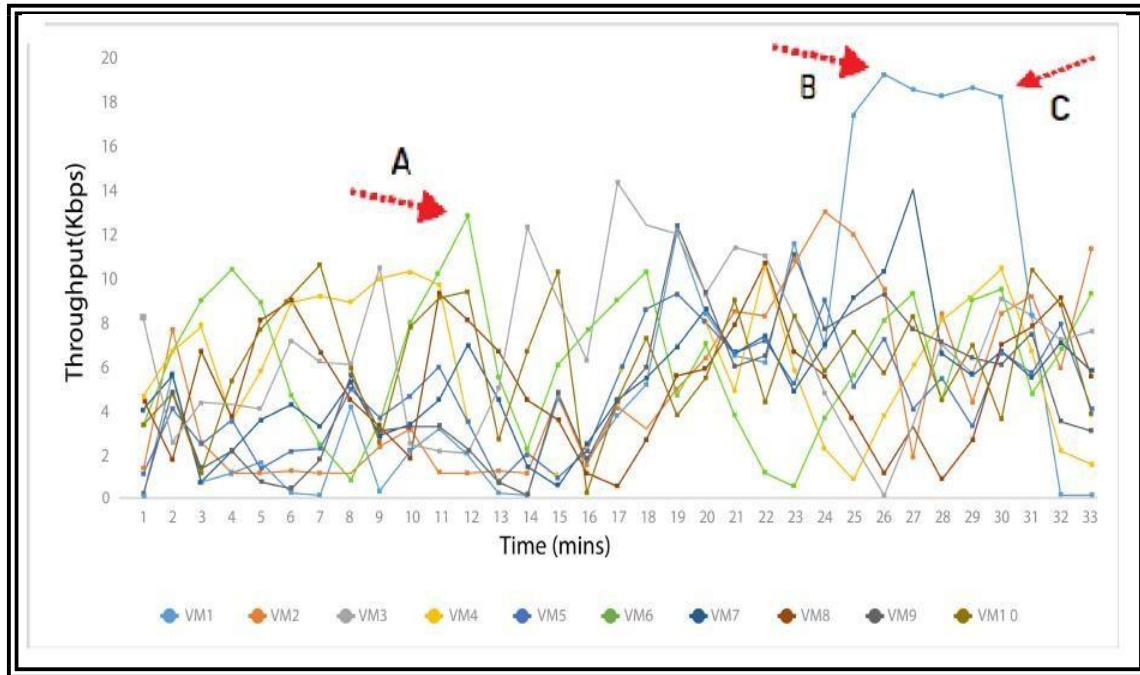


Figure 4.7: Cyclic Attack Pattern of Network Traffic

Attack on Ravello Systems

On Oracle's cross-cloud platform, Ravello Systems, the same assault may be analysed for its effectiveness. Oracle Ravello is a commercial cloud platform that enables users to utilise any of the most popular public clouds on the market today, including GCE, AWS, and others. In our testbed, AWS is set up to function as the underlying infrastructure as a service provider. Ravello provides a one-of-a-kind cloud application hypervisor technology that enables businesses and individuals to encapsulate and abstract an entire multi-VM application and its environment. As a result, the application is able to run on any cloud, regardless of whether it is public or private, without requiring any changes to be made.

Network Configuration

Ravello makes it possible to configure the network settings for each VM, including the assignment of static IP addresses. The static IP, Netmask, Gateway, and DNS server all need to be assigned to this interface before it can be used. Ravello is able to handle the functionality of SDN that is responsible for the automatic creation of a virtual switch based on the IP address that has been assigned to VM as well as the netmask of the interfaces. Each and every interface that belongs to the same subnet will be assigned to the identical virtual switch. If a gateway has been allocated through the user interface, then the SDN of that gateway will follow the properties listed below: Ravello's SDN settings will be in accordance with the VM's Gateway setting if the guest VM has been given a gateway IP and has been configured to act as a router. Alternatively, if the VM has not been given a gateway IP but has been configured to act as a router.

In the event that the guest VM does not have a Gateway IP, Ravello's Software Defined Network (SDN) will connect a virtual Router to the virtual switch and assign it the defined Gateway IP.

Ravello's software-defined networking (SDN) capabilities will be inactive if the Ravello user interface is not used to specify a default Gateway.

Evaluation

All virtual machines are members of distinct VLAN Tags, as can be shown in figure 4.8. Tagging of VLANs is described in IEEE Standard 802.1Q [183]. VLAN tags may be included in components of the network that support the VLAN standard.

When a packet enters a portion of the network that is configured with VLAN, a tag is appended to the packet to indicate the VLAN membership of the packet. VLAN ensures that the network traffic of all VMs is kept completely separate from one another. The network configurations of all three virtual machines are depicted in Figure 4.8 as well.

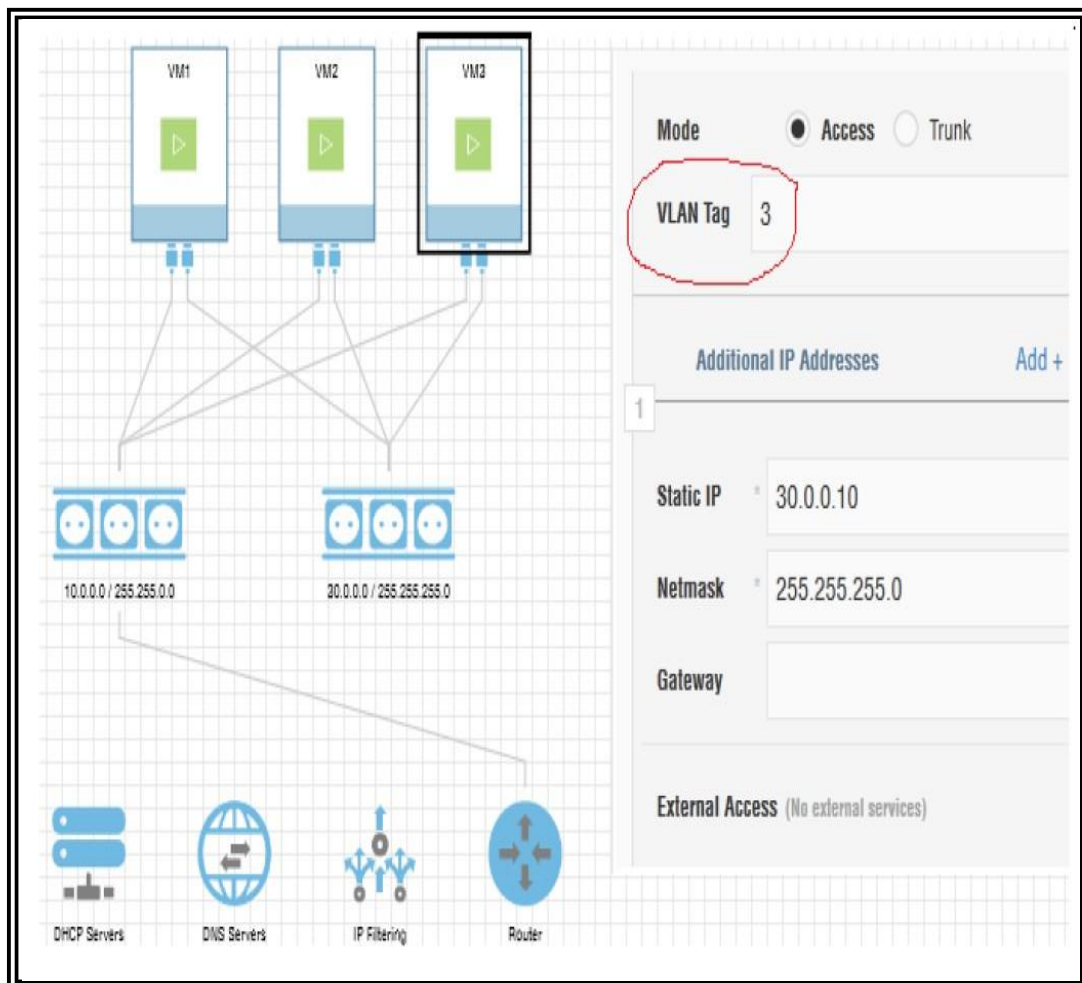


Figure 4.8: VLAN Configuration of VMs in Oracle Ravello System

Evaluation of the Outcome

As can be seen in Figure 4.8, the attacking virtual machine (VM) is in a position to successfully see the network traffic that is occurring between the target VMs when they are conversing with the ping command. Figure 4.8 demonstrates that the

attacking VM is successful in receiving the ICMP echo request and reply, which indicates that the two target VMs are able to communicate with one another. As was previously stated, ICMP is the protocol that is being used to carry out the ping command. The echo request and reply demonstrates that the two VMs are communicating with one another. VM3, which is an attacking VM, does not have the access rights necessary to monitor the conversation taking place on the network connection between the other VMs.

```

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

Limitations

The success of the proposed method depends on the particular network model chosen. Openvswitch (OVS) is commonly used by cloud providers to implement complex network configurations. An advanced implementation of Open Stack's networking capabilities, including the VLAN and ML2 plugin in OVS, is described in the case study we used for our experiment. Only a neutron network enabled by OVS, which supports multiple cutting-edge security and design options, is vulnerable to this attack. It is not compatible with the nova-network, a legacy

network. The latter paradigm is restricted in that it does not allow for the creation of a sophisticated network layout. Before Neutron was added to OpenStack, Nova-networking was the sole option for creating and managing networks. Only the FLAT network and DHCP services are compatible with Nova-network. It's always been a component of OpenStack, but its constraints make it look antiquated. The standard for flat networks and DHCP was to use the same structure. The central idea is that each virtual machine is linked to a single physical computer, or "bridge," in this case a Linux host. The eth0 physical NIC on the host computer serves as an attachment point for the bridge. A number of virtual machines are linked to this bridging device. Figure 4.10 presents this conceptual framework.

A breakdown of the cloud services that could be compromised in this attack is shown in Table 4.4.

Stopping Network-Channel Attacks

In this article, we'll examine the benefits and drawbacks of some of the most promising defences against cross-VM network-channels.

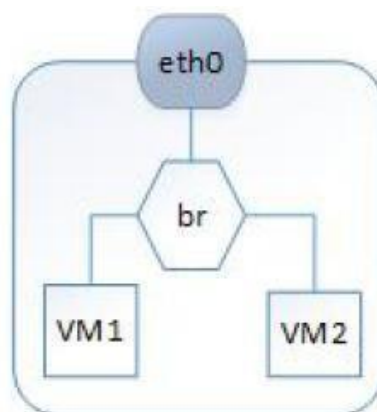


Figure 4.10: Nova-Network

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.

Avoiding Co-residency

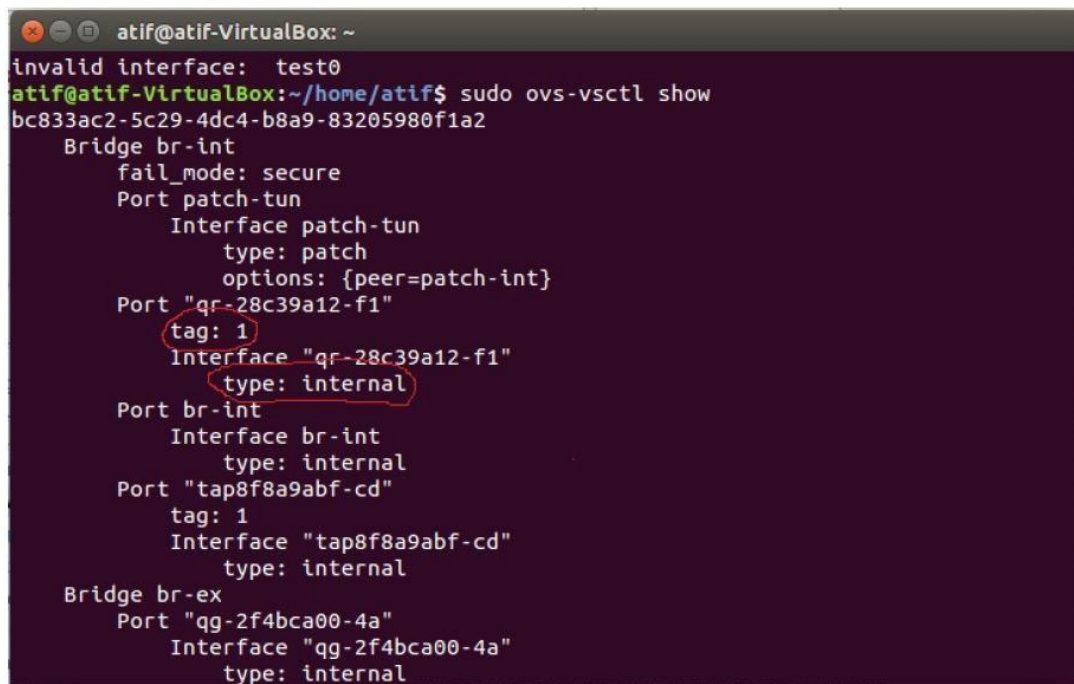
One time-tested method for ensuring the proper separation of duties in a safe setting is to perform different activities on different hardware at different times. When protecting against side-channel attacks (and many others), this method offers the highest level of confidence. This approach, however, would sidestep many of the current and future practises of virtual machines, such as the use of public clouds that multiplex actual servers like Amazon EC2, Windows Azure, and Rackspace, and the use of other VM-powered apps, as detailed in Chapter 1 of this thesis.

Solutions to Prevent Attacks

Several efforts are applicable and could be utilised to protect against cross-virtual-machine threats. Afoulk et al. [184] try to avoid conflict of interest among VMs using priority based scheduling, whereas Zhang et al. [168] propose using side-channels as a detector to identify illegal co-residency based on the timing channel (accessing L2 cache response time). Altering OpenStack's source code directly is an

alternative. It restricts the level of detail at which network-based side channels or devices from the outside can access a live system. The OVS internal network bridge, denoted by br-int, is the connection point for many VM interfaces to the underlying physical device and the outside network. Because the TAP interface is not a private Ethernet interface, an attacker within the internal network can impersonate it.

The output of OVS is displayed in Figure 4.11, which exhibits the interconnection of several virtual Interfaces.



```
atif@atif-VirtualBox: ~
invalid interface: test0
atif@atif-VirtualBox:~/home/atif$ sudo ovs-vsctl show
bc833ac2-5c29-4dc4-b8a9-83205980f1a2
    Bridge br-int
        fail_mode: secure
        Port patch-tun
            Interface patch-tun
                type: patch
                options: {peer=patch-int}
        Port "qr-28c39a12-f1"
            tag: 1
            Interface "qr-28c39a12-f1"
                type: internal
        Port br-int
            Interface br-int
                type: internal
        Port "tap8f8a9abf-cd"
            tag: 1
            Interface "tap8f8a9abf-cd"
                type: internal
    Bridge br-ex
        Port "qg-2f4bca00-4a"
            Interface "qg-2f4bca00-4a"
                type: internal
```

Figure 4.11: Connectivity of Virtual Interfaces at OVS

In-depth examination of the results revealed that all legitimate interfaces have two characteristics: tag and type. The 'tag' of an interface indicates the VLAN that is enabled to provide VM isolation, and the 'type' determines the interface's behaviour. Attachment of the test0 mock interface is depicted in Figure 4.12.

When examining the output of OpenvSwitch, it is clear that neither the "tag" nor the "type" attributes of this fake interface are present.

Close monitoring of the OpenStack code base revealed that the cloud service's networking (neutron) component contains a security flaw. Because of this, the `def run vscctl` method in the class `/opt/stack/neutron/agent/impl vscctl.py`, which is responsible for carrying out virtual switch operations, needs to be updated in neutron code (`self, args`). All the data about the bridge's connected interfaces is gathered in a new method called `get all bridges(self,args)`.

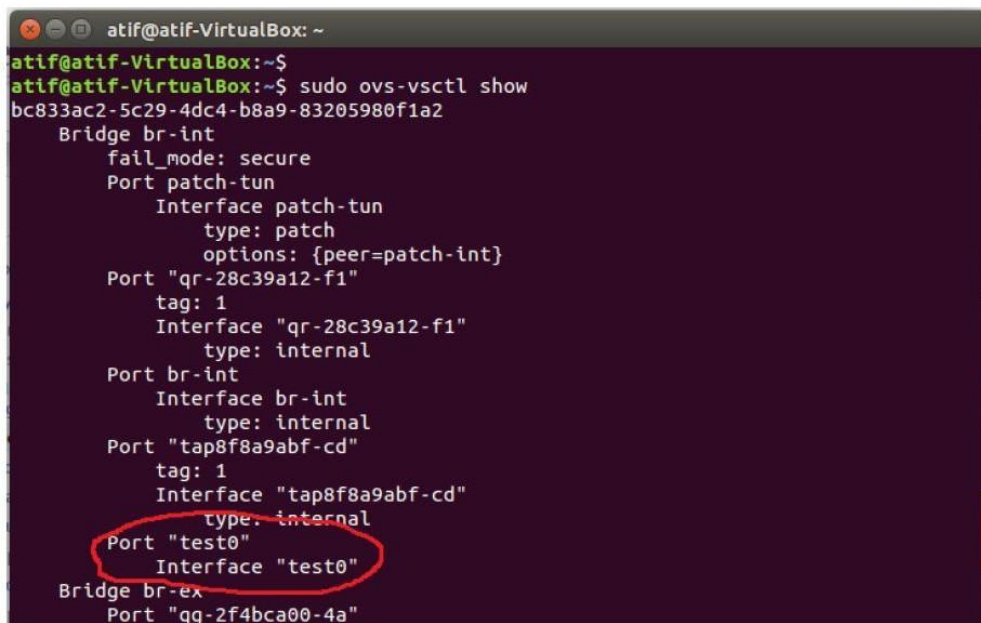
A terminal window titled 'atif@atif-VirtualBox: ~' shows the command 'sudo ovs-vsctl show' being executed. The output lists details for two bridges: 'br-int' and 'br-ex'. Under 'br-int', several ports are listed, including 'qr-28c39a12-f1' (tag: 1, type: internal) and 'tap8f8a9abf-cd' (tag: 1, type: internal). A red circle highlights the entry for 'test0', which is listed as 'Port "test0"' and 'Interface "test0"' without any attributes. The 'br-ex' bridge has a port 'qg-2f4bca00-4a'.

Figure 4.12: Attachment of Dummy Interface

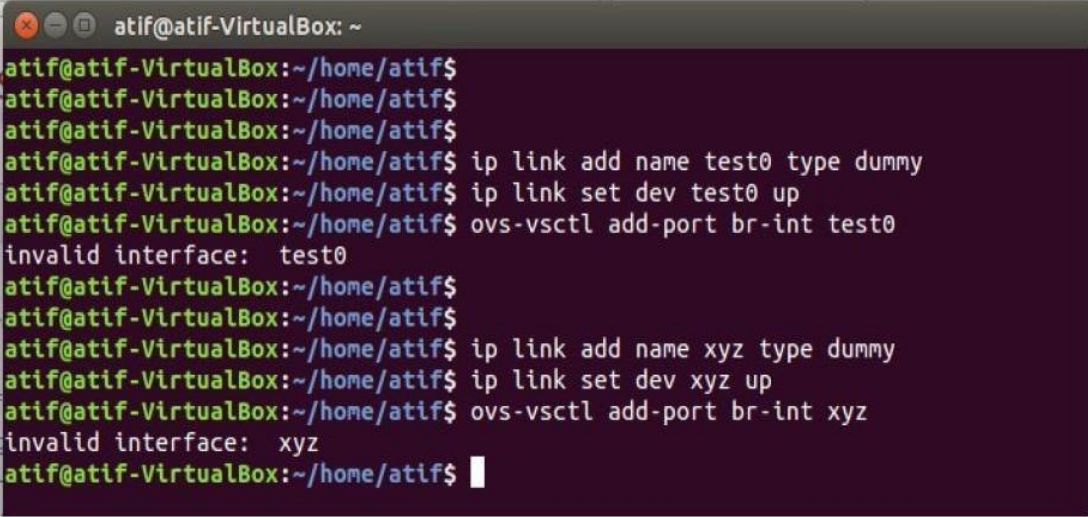
The goal of this feature is to prevent direct connections between TAP interfaces and an OVS bridge utilising the same Ethernet connection to the Internet if the linked interface is capable of communicating using solely the TAP interface. Each valid interface has the following three qualities, according to the interface analysis: tag, interface, and type. The 'tag' attribute specifies that the VLAN is active for VM

isolation, the 'interface' attribute links the VLAN to the private Ethernet in the backend, and the 'type' attribute reveals the interface's behaviour. Before establishing a connection to the bridge, the security checks must validate each of these 'TAP' characteristics.

An invalid interface error is displayed in the OpenStack cloud when a dummy virtual interface is attached to OpenVSwitch (OVS) after a change was made to the neutron code, as seen in Figure 4.13.

Discussion

The chapter concludes with a discussion of the most important results from the empirical examination of the TAP impersonation attack.



```
atif@atif-VirtualBox: ~
atif@atif-VirtualBox:~/home/atif$
atif@atif-VirtualBox:~/home/atif$
atif@atif-VirtualBox:~/home/atif$ ip link add name test0 type dummy
atif@atif-VirtualBox:~/home/atif$ ip link set dev test0 up
atif@atif-VirtualBox:~/home/atif$ ovs-vsctl add-port br-int test0
invalid interface: test0
atif@atif-VirtualBox:~/home/atif$
atif@atif-VirtualBox:~/home/atif$
atif@atif-VirtualBox:~/home/atif$ ip link add name xyz type dummy
atif@atif-VirtualBox:~/home/atif$ ip link set dev xyz up
atif@atif-VirtualBox:~/home/atif$ ovs-vsctl add-port br-int xyz
invalid interface: xyz
atif@atif-VirtualBox:~/home/atif$
```

Figure 4.13: Blockage of Invalid Interface

Through the use of a network's channel, attacks such TAP impersonation and mirroring were made possible. The initial step in breaking into the network was to impersonate a TAP. The network traffic was being redirected to a secret location, thus a mirror was established up at the main bridge. The bridge's attention was

narrowed to the data streams generated by virtual machines. Evidence from the results shows that the disclosed attack successfully exploits cloud network architecture by rerouting the traffic of co-located virtual machines.

The described attack, as seen in the network traffic graph, is capable of diverting the high-rate network traffic of co-located VMs to a secret target point. When comparing different methods for evaluating the network bandwidth of each virtual machine in real time, PRTG was shown to be the most effective overall. The graph suddenly spiked, showing that the attack was successful.

It was intriguing to see that this attack configuration can be protected from being exploited. By spoofing the TAP interface, the attacker can gain access to the network. By comparing impersonated TAPs to the actual thing, we were able to determine that they have distinct characteristics that can be mitigated by modifying the open source cloud's code.

The evaluation requirements (Section 4.3) have been met using the following means: The attack tactic is assessed for its ability to carry out all of the goals for which it was developed. All of these assault tactics are depicted in Figure 4.3, which is used to assess the functional parameter's inference capacity. Even more reassuring was the fact that every event had been correctly inferred. The results of the analysis confirmed that the attack model effectively achieved its design goals.

In the context of this thesis, "expressiveness" is how well the TAP Impersonation attack performs across the board when applied to the network environment. Figure 4.4 shows the captured network traffic of other co-located VMs, providing

significant proof regarding the working of attack, and therefore is used to evaluate the expressive parameter.

The evaluation findings show that the capacity to reroute network traffic from other co-located VMs is the approach's main strength. The approach was outlined, however it had a few flaws that are discussed in Section 4.7.

Summary

In conclusion, this chapter gives the following information in order to evaluate this assault in a real-time setting: details of experiment, their setup, configuration, limits, and mitigation plan; and comparisons to work described in the literature on cross- VM attack. In addition, we show how to initiate a cross-virtual machine network channel attack inside of OpenStack, an open-source cloud platform. The security weakness in the cloud model has been investigated using a divide-and-conquer approach. To illustrate just how difficult it is for an attacker to breach the network system and to explain how to go over the cloud's security, we have also gone into detail about attack settings and hurdles. The configurations and properties of the virtual machine (VM) have been discussed in detail.

The attack described here involves masquerading as a TAP interface and creating a network mirror in the bridge interface through which the connections of all co- located VMs are routed. The goal of the experiment is to use the underlying network channel as an empirical reference point for tracking the data transfers of physically separated virtual machines. The empirical analysis confirms that the attackers are able to successfully exploit the vulnerability and reroute the network traffic of co- located VMs. Since the attacking VM does not exceed the allotted VM resource

capacity, it is difficult for cloud providers to notice and monitor such attacks. A countermeasure approach that closes the security issue in OpenStack's source code has also been presented.

Our next discovery, obtained from an examination of the OpenStack Xen architecture, is described in detail in the following chapter through the use of experiments on various configurations and assessments of their efficacy.

Chapter 5

Countermeasures to Privilege Escalation Attacks and Their Implementation

The researchers have presented various methods for exploiting memory and have investigated vulnerabilities in kernel code, as was covered in chapter 2. As a result of this exploitation, the installation of hardening systems to prevent privilege escalation assaults is required. As memory separation techniques between the kernel and user space become increasingly stringent, such as Intel's SMEP, attackers are forced to rely more and more on approaches that involve code reuse in order to exploit vulnerabilities, particularly kernel-related vulnerabilities. In contrast to comparable attacks in more protective settings, such as web browsers, non-privileged limited adversaries have a large deal of latitude when exploiting memory disclosure vulnerabilities. This enables them to dynamically determine the placement of a certain code segment and design code-reuse content. The linkage of code variation with the development of a "read XOR execute" (RX) memory security technique is an effective defence against the misuse of user-land software, according to recent study [185]. However, this method has not yet been implemented for the kernel's own defence.

This study aims to analyse the implications of hypervisor vulnerabilities in cloud computing so as to better understand those implications. In this study, a novel method of attacking a cross-VM cloud system that was running Xen hypervisor on the bottom was revealed. The hypervisor's primary duty is to guarantee that there is

adequate separation between domains, often known as the root domain and the non-root domain. The suggested method makes use of the ROP model, which stands for return-oriented programming. An adversary with limited privileges who resides on the same physical machine as the target VM is able to launch ROP, establish a connection with root domain by abusing the network channel, and obtain possession of tool stack despite the fact that they are not authorised to access it directly is able to do so. Research has been done on the ROP approach in cross-VM environments. This technique is used when a malicious VM reuses the existing code that is already stored in the Kernel memory and does not modify or inject any external programme in order for it to be executed. Attackers have been successful in exploiting the integrity protections by using this strategy. According to the findings, the proposed assault is doable in modern public clouds such as OpenStack and Azure, which are setup with Xen para-virtualization underneath. In addition to our findings, a solution for a countermeasure that can be used against the assault that was detailed has been proposed. This suggests that the cross-VM attack that is being presented is of some consequence when used to cloud platforms. The findings of this research point to the existence of a significant security risk for the industry's most prominent cloud providers.

Introduction

There have been several studies done in the past that have shown how hypervisors can be utilised to improve the safety of virtual machines (VMs) that are running on them [33, 186, 187, 187–191]. Their primary hypothesis is based on the idea that the hypervisor is a piece of software that, in comparison to operating systems, consists

of a relatively small amount of code. As a result, they believe that it can be thoroughly examined, which in turn reduces the number of bugs that exist within its code. However, modern services of hypervisors like Xen and VMware are intricate. These hypervisors are also massive softwares with a high number of lines of code, which contradicts the theory. For example, Xen 4.1.2 has approximately 350 thousand lines of source code in the hypervisor itself. Only 10,000 lines of source code can be processed by the most advanced formal authentication method available today [103]. It is not in anyone's best interest to formally vet such a massive hypervisor code base. In addition, the National Vulnerability Database [192] provides a rundown of the most recent security flaws that have been discovered in hypervisor software while also stating that it is not easy to produce hypervisor software that is bug-free. Because of these vulnerabilities, hypervisors are frequently the focus of attacks [193–195].

The primary goal of a software attack is to divert the normal flow of the currently operating programme in order to carry out some other instructions. The phrase "software assault" is commonly used in a wide variety of applications, ranging from desktop to eeb to even a basic tool like notepad. The exploitation of memory vulnerabilities is a more specific type of software attack that can come in many different forms [196]. Some examples of memory vulnerabilities are buffer overflow, heap overflow, and string vulnerability. The most dangerous vulnerability is known as Buffer Overflow [197], and it allows an attacker to overflow buffers in the stack process and overwrite the return address of a function with a random memory location. This is the most serious type of threat. When the vulnerable

function completes its execution and returns, the attacker can take advantage of the overwriting by executing whatever code they choose. The researchers have carried out this attack in a variety of guises, and they have also provided their solution to the problem in a number of different guises. Their mitigation strategies prevent the execution of code that is located in the data regions of the process, such as on the stack or in the heap, for example. Data Execution Prevention (DEP) is a feature found in modern operating systems, and it is implemented with the help of the security model [198]. In these models, the memory location can either be executable or writable; it cannot do both jobs at the same time. Rather, it must choose one or the other. Because of this, the attacker will be unable to create code and then execute it at the same address on the stack and the heap.

In spite of the deployment of such countermeasure tactics, attackers are still able to identify the gaps to execute proposed code by making reuse of already existing code in the process address space rather than injecting new code. This allows them to circumvent the countermeasures. The return-into-libc attack, which was designed by Solar [199], is one of the well-known subcategories of this attack. During this type of attack, the adversary modifies the return address of a susceptible function so that it points to the address of any function contained within the libc library. For instance, the attacker may overwrite the value with the address of a "system" function, which would allow for the opening of a shell. A shell is the most common type of backdoor, because it allows for the simultaneous execution of numerous jobs. Return-Oriented Programming, or ROP, is a more specific form of the return-into-libc attack. ROP is an attack in which the attacker attempts to implement parts of

code that are scattered throughout the process address space by connecting them with unintended transfer instructions, most commonly the 'ret' instruction.

According to the theory put forward by Shacham [15, 197], the attacker in ROP typically carries out the attack in two phases:

The initial step in an attack is the attacker determining the appropriate sequence of instructions to carry out the intended tasks. A gadget is a brief sequence of instructions (often between 2 and 5 steps) that performs a specific task. However, the length of the devices is not restricted in any way, and it has not been proven that devices with a greater range are impossible to create. Additionally, it is possible for the attacker to discover multiple tools.

The second phase involves the attacker setting up a chain reaction of the previously identified devices.

Figure 5.1 is a schematic depiction of ROP exploits.

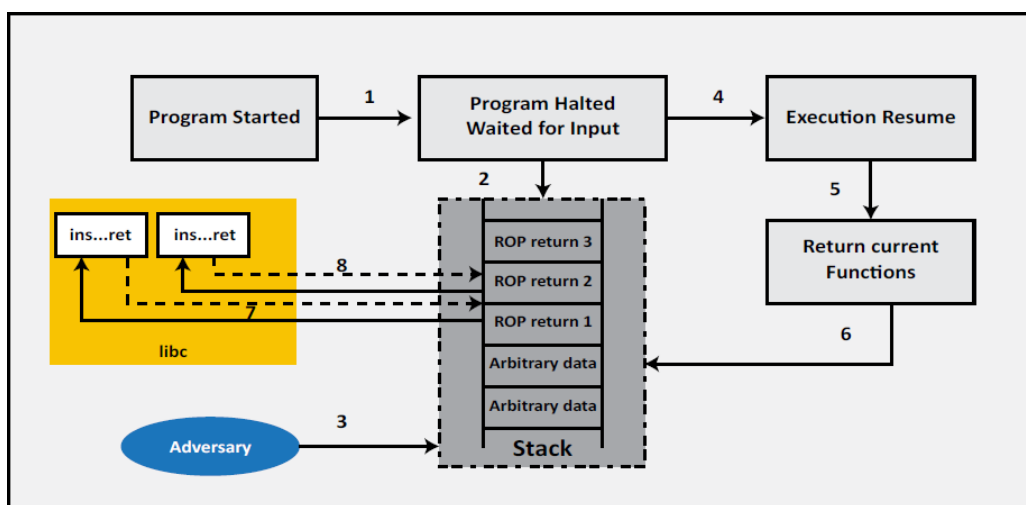


Figure 5.1: Overview of RoP

An attacker begins by locating a vulnerable function in the process address space. This function could be part of the application itself or a component of one of the

process's internal or external libraries. This potentially exploitable function would then be called while the programme is running. The suggested return addresses are overwritten on the stack by the attacker (i.e the start addresses of the gadgets). The 'ret' instruction is commonly used to achieve the goal of the gadgets ending in an accidental control transfer. The 'ret' instruction at the end of a gadget pops the next value (the address of the next gadget) from the stack, which is then used as the new execution instruction the next time the actual action (gadget) is executed. Data values inserted by the attacker onto the stack can be used as arguments for the proposed instructions [197, 200-203].

The first-described method simply uses the 'return' instruction to switch the program's control flow to the new one that is being proposed. However, there are other ways to manipulate the controller than the "return" instruction. Other control transfer instructions include the address-storing indirect 'imp' and indirect 'call' directives. So now the attacker can choose to use such instructions to run their own code by changing the flow. According to what Checkoway et al. [204] proposed In ROP, similar commands to return are used instead of the return instruction itself. A 'pop' instruction sequence, for instance, is functionally equivalent to a 'ret' instruction in that it moves the contents of the stack from the top of the register to the instruction location.

The hypervisor in cloud computing is in charge of establishing security boundaries between root and non-root virtual machines. The hypervisor, however, is a piece of software made up of many individual lines of code, all of which can be exploited in some way. Recent studies (covered in section 2.9.2) have shown several techniques

for attacking the hypervisor in a way that allows an attacker to employ ROP to elevate the privilege level of non-root users and to inject an external programme into system memory for arbitrary execution. In [210], the authors demonstrate in detail how an attacker might escalate their privileges by changing the memory-resident value of a non-root virtual machine.

Although it is evident that abusing RoP and shared memory might lead to privilege escalation assaults, as mentioned in Section 2.9, a fine-grained privilege escalation attack that uses ROP in conjunction with the cross-VM network-channel attack has not yet been carried out. By adding new layers of isolation between virtual machines (VMs) in the form of domains, hypervisors have significantly improved data security. Different types of virtual machines (VMs) are housed in different domains. Since each virtual machine in the cloud has its own unique set of permissions and lives in its own separate domain, this type of assault appears to be less of a concern in the cloud.

In Section 3.3 of Chapter 3, we discussed the hypervisor architecture and domain attributes that are the focus of the study into the state-of-the-art hypervisor's security flaw. According to the results presented, there are obstacles that must be overcome before these weaknesses can be exploited. The next part details how they've been implemented in OpenStack and Microsoft Azure, two of the most popular open-source cloud platforms, and how to attack the vulnerabilities discussed in Section 3.2.

This chapter has introduced a cutting-edge, novel attack mechanism known as privilege escalation. It exploits ROP in tandem with network-channel to launch an

attack in cross-VM scenarios, providing insight into the state of security in cloud environments supporting many tenants. While studying the security of cloud servers is not the primary focus of this research, the isolation qualities of cloud systems have been examined to see whether or not the suggested attack may be used to bypass virtual machines and hypervisors, and if so, how.

Problem Statement and Contribution

As was previously mentioned, Cloud providers run several virtual machines (VMs) on the same physical server. All virtual machines (VMs) have been given a certain amount of access, and the hypervisor then places them in the appropriate security domain. However, by abusing domain isolation, the attacker can raise the privileges of the VM it controls. However, such attacks are impossible due to the cloud's built-in security measures. It's getting harder to pull off these kinds of exploits in real time.

Throughout this chapter, we will examine the RoP and the associated approaches that have been developed to circumvent these security flaws. Additionally, this chapter presents the design of a zero-day attack model (as mentioned in Section 1.3.1, research goal 2), with particular emphasis on the primary components of the underlying hypervisor architecture, the technique, and the manner in which domains are connected, processed, and applied for the overall model. The construction of real-time systems for OpenStack and Oracle Ravello verifies the basic principle of the suggested attack model. In order to measure the efficacy of the suggested assault model, a set of experiments was carried out. The objective was to demonstrate the system's viability in a real-world setting. At last, we give our plan of action for

defending against these assaults (objective 3 of the research, as described in Section 1.3.1).

Technical Challenges

In this research, we present the design and implementation of a cross-virtual-machine privilege escalation attack. To each virtual machine in the cloud, a unique set of permissions has been granted. An attack that raises the privileges of normally unprivileged virtual machines is called a "privilege escalation assault" [195]. One privileged virtual machine (VM) typically resides in dom0, while numerous unprivileged VMs (domU) coexist in a cloud infrastructure. All virtual machines are kept separate from one another by being assigned to various "domains" by the hypervisor. This assault, like many others, is a privilege escalation attack in which the attacking virtual machine (VM) employs the ROP technique to gain access to higher levels of privileges and then uses the network channel to communicate with the root domain. Many of the steps taken to execute the proposed attack efficiently and precisely in a virtualized setting are fresh to this study. In particular, a description of how to deal with the major obstacles of privilege escalation and root domain connection has been presented. If an attacker is able to elevate the privileges of his virtual machine (VM), he can compromise any other VMs hosted on the same physical computer. By spawning numerous malicious VMs that hog the host machine's resources, this attack can also lead to a denial-of-service.

Case of an Attack

In this part, we show how a rogue VM can use the ROP approach to abuse the network

channel and gain elevated privileges. Attackers have been found to be able to administer other co-resided VMs once they have gained access to the necessary privileges and then take control of Toolstack. Having control of Toolstack is a severe threat to other virtual machines (VMs) and cloud providers, and it's also fairly important to attackers. Many unique applications have been introduced to execute this attack, including breaking into the root domain, stealing Toolstack, and using ROP in conjunction with a network channel. This procedure has been thoroughly tested on an actual, live instance of the OpenStack cloud.

The Attack

Through the use of the attacker machine, there is the possibility of gaining control of Toolstack, which is located in the hypervisor of dom0. Toolstack is in charge of managing the operation of additional virtual machines (VMs) that sit on the same physical machine. The primary objective of this research is to find a way beyond OpenStack's security perimeter by first introducing the idea of code RoP and then utilising the network channel as a means of attack. By compromising the network channel, the attacker was able to successfully break into the domain, also known as dom0, of the running system. This is the domain in which the privilege VMs reside. It is necessary to do in-depth research on the network connections that enable OpenStack to allow for its expansion and via which other cloud computing systems can be joined.

The root or administrative virtual machines will reside at dom0 in an OpenStack cloud that uses Xen as its underlying architecture. These VMs will have direct access to the hardware that the cloud is running on. These root or administrative VMs have

specific privileges, which allow them to exert influence over the other VMs in the system. Dom0 is the initial machine that is started up when the system boots up. There is a specialised stack in this industry that is referred to as the Toolstack, and it is what enables a virtual machine to control the creation, destruction, and allocation of resources. Because they are located in the unprivileged domU domain, guest virtual machines (VMs) cannot directly access the hardware. These guest VMs are not permitted to interfere with the management or operations of other guests. 1) An OpenStack client, a specialised component of OpenStack, has been set up in an attacking guest virtual machine (VM), which is located in domU. The configuration of this unique feature of OpenStack is based on the idea that a malicious VM may use this function to copy the code of OpenStack into its own domain, known as domU. This is the basic concept that underlies the setting of this feature. The unique method of RoP, known as code-reuse, was implemented in the device's internal memory as a result of this action. This code that runs in memory has previously been validated and examined by the security perimeter of OpenStack, and it is currently in a state where it may be run. Therefore, security perimeters are unable to prevent it from happening. The most important advantage of using previously used code in a guest virtual machine is that it allows a malicious guest VM to become the host of its very own local computer (sub-host from main root). After acting as the host of its own local computer, it was awarded the status of "dom0," which enables it to manage its own sub-guest virtual machines. This architecture resembles a tree in which there is one main root that lives in dom0 and that have additional guest VMs that belong to domU. The tree has branches that belong to both dom0 and domU.

One of these guest VMs is functioning as a root of its local system, which also contains its own guest VMs; yet, on a global scale, it functions as a sub-root.

These malicious virtual machines use the access rights and (dom0 and domU). Dom0 to control his own locally hosted virtual machines (VMs), which he can then sublease, and domU, which is automatically added to his account. When two domains need to communicate with one another, they can use a specialised python API called xapi in dom0 to establish a connection over the br0 network bridge. One uses a specialised python API, called XenAPI, on domU to establish a connection with the xapi service on dom0. The path from domU to dom0 uses the internal management network and begins at the Ethernet device and ends at the bridge.

2) In the proposed attack scenario, the attacker connected the veth bridge pair that links the python library's xapi with the xapi of the xapi. One of veth's bridges is present in the main root, while the other is a piece of the malicious guest VM he acquires by code reuse. Thirdly, if a xapi establishes a connection with another xapi via a bridge, it does so under the assumption that the roots of both clouds are trying to establish a link for the purposes of cloud expansion or hybrid cloud deployment. This allows the attacking machine to get access to dom0, obtain root capabilities, and take command of the tool stack, allowing it to administer other guest virtual machines. Once an attacker takes control of Toolstack, they not only have access to all of the guest VMs operating on the same physical machine, but they can also create dummy VMs and flood them with resources to launch a denial-of-service assault. As can be seen in Figure 5.2, the OpenStack Xen architecture is vulnerable to a variety of attacks because of the way it is designed.

Evaluation Criteria

As can be seen in Figure 5.2, the evaluation is centred on the attack mechanism covered in Section 5.2.1. The primary objective of the attack is to elevate the privilege level of non-root virtual machines so that they can act as root virtual machines by taking control of the tool stack and then managing additional co-located virtual machines. This is done by putting into practise the attack techniques covered in 5.2.1 and then gauging the outcome.

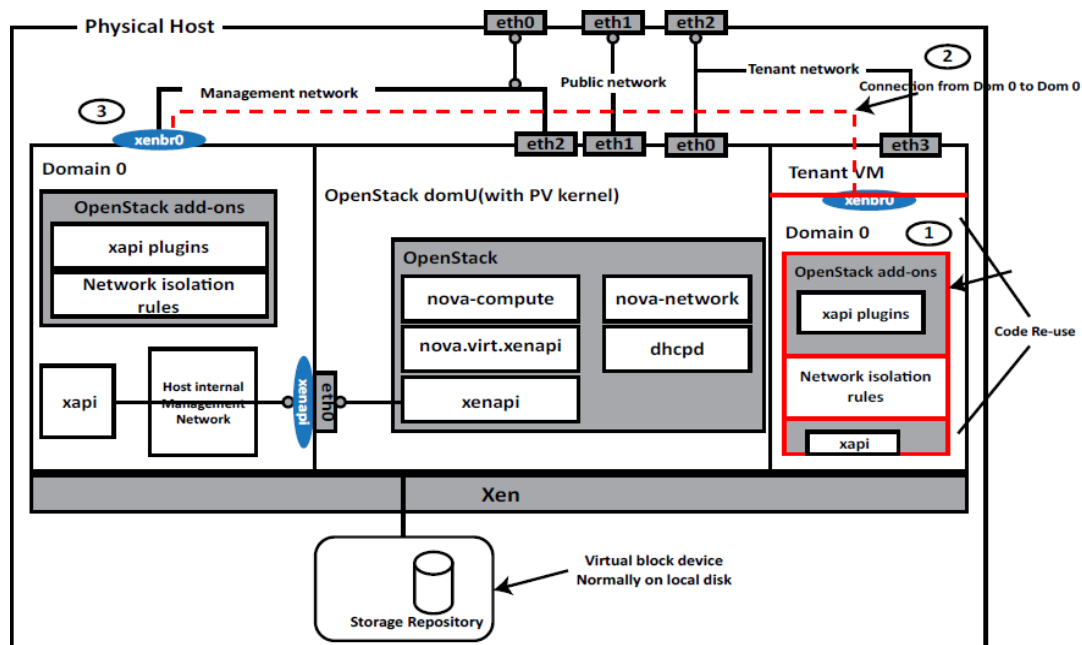


Figure 5.2: Attack Model on OpenStack Xen Architecture

Results on Performance (ROP) is the primary metric used in judging the outcomes. Criteria for assessment are based on the following central facets of introspective quality. I Functional demonstrates that the aforementioned five steps, critical to the execution of attack, have been completed. (ii) The level of expressivity denotes the efficiency with which the attack tactic is implemented.

As a result, the attack's advantages and disadvantages have been demonstrated.

Evaluation

Both OpenStack, a popular open-source IAAS cloud platform, and Microsoft Azure, a commercial cloud system, are used to test the assault outlined in chapter 5. The system is now set up with the subsequent configurations.

Experiment setup

OpenStack Networking is fully compatible with the Virtual Networking Infrastructure (VNI) and the Physical Networking Infrastructure (PNI) access layer stages (PNI). When turned on, virtual machines (VMs) can activate complex virtual network topologies, including firewalls, load balancers, and VPNs (VPN). Connectivity paves the way for further functions including networks, subnets, and routers. Each component functions similarly to its physical analogue: networks are made up of subnets, and routers direct data packets between them. There must be at least one external network and any number of internal networks in every defined networking setup. Virtual machines can communicate with these SDNs by hooking up to them straight away. To connect to VMs located in other networks, networking routers are required. Both an exterior and an internal network interface are present on each router. Subnets function in much the same way as a real router does in terms of granting access to computers on different networks.

Security Configuration

Cloud computing is highly secure. With OpenStack, administrators may create security groups to manage firewall rules collectively. It's a set of rules from various security groups that together define how users can interact with the network. One or more security groups can be assigned to a virtual machine. The primary function of

security groups is to control the flow of network traffic to and from virtual machines by either blocking or opening specific ports, defining ranges of ports, or defining the routes taken by certain types of network traffic. Security groups can also be supported by a network. To improve OpenStack networking security, a security group plug-in is applied to every defined Network setup.

Attack Setup

Developments in reusing existing code (Remote Procedure Calls) have been detailed in this section. Because of ROP, the attacker can elevate the privileges of themalicious VM, which allows them to launch the proposed attack over the network channel. The adversary's objective in this attack is to manage the co-resided VMs and take over the Toolstack in dom0. Once connected to root (dom0), an attacker can use Toolstack to gain access to the root domain and ultimately take command of the victims. As described in [103], an attacker can utilise the ROP approach to compromise the hypervisor and then alter its code to gain root access. The proposed attack has already been implemented in an OpenStack cloud environment via ROP and a network channel. Toolstack in Xen architecture allows full management of operating virtual machines (VMs) on a cloud platform in real time. Toolstack could fall into the hands of a hostile VM if it is able to bypass the security perimeter and raise its privilege level. If the attacker has access to the tool stack, they can not only take over the host system but also any virtual machines (VMs) operating on the same host.

Attack Scenario

The effectiveness of the suggested assault against the planned architecture of OpenStack is evaluated, based on the attack presented in chapter 5. Two virtual machines have been started up to analyse the breach. More virtual machines can be created if necessary. In order to begin using newly created virtual machines (VMs), an OS must be installed and set up on each one. The customer can set up their virtual machines with whichever OS they like. Each virtual machine (VM) has been set up with a unique operating system for the purposes of testing.

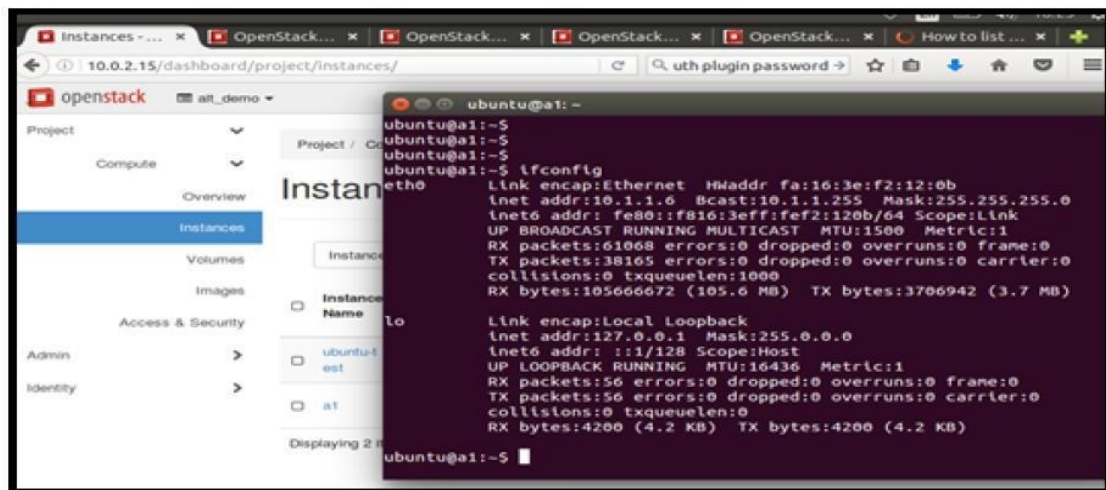


Figure 5.3: Attacking Machine Console

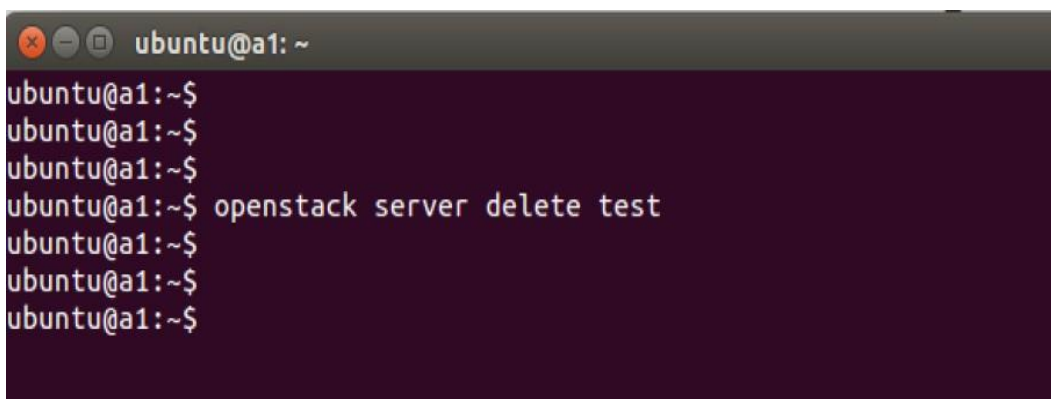
Ubuntu 15.10 is installed on the first virtual machine, and Windows is installed on the second. Both virtual machines can use either a private IP address or a floating IP address. The Internet and other external resources can be accessed with the use of floating IP addresses. While public IP is used for external communication between VMs, private IP is utilised for inside VM communication. Assigning each virtual machine to a unique slice of the network provides a layer of logical separation.

Analysis Results

In this section, we detail the experimental results that were used to assess the whole method in a real-world setting, looking at how well domains could be exploited by stealing the ToolStack and how efficiently physical resources could be used by virtual machines. The following indicators are used to assess the efficacy of the suggested method: Two things stand out: 1) a non-root VM performing a root action, and 2) the attacker VM's use of hardware resources both before and after the attack.

Virtual Machine with No Root Permissions Performing Root Tasks The attack approach is verified by inspecting a set of root commands produced by the (attacker) non-root VM in the testing environment. These commands, annotated with the actual truth about an attack's presence, are displayed during each experiment run.

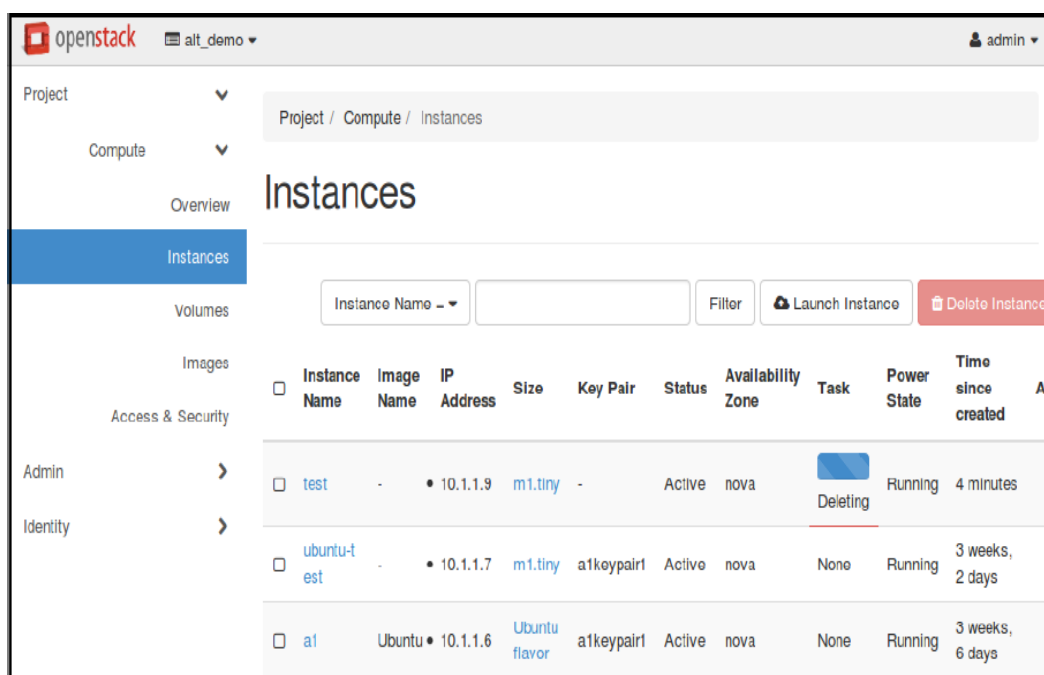
In Figure 5.3, we see the login prompt for the attacking computer (a1) after using the ssh command, which necessitates a key pair set during instance formation. After executing the attack strategy, the attacking VM not only gains access to higher-level resources but also communicates with dom0, the system's root domain. Once connected, the attacking VM takes control of a Toolstack and could potentially modify or delete the victim VMs as depicted in Figure 5.4.

A terminal window with a dark purple background and a title bar that reads 'ubuntu@a1: ~'. The terminal shows a series of shell prompts 'ubuntu@a1:~\$' and the command 'openstack server delete test' being entered. The output of the command is not visible.

```
ubuntu@a1:~$  
ubuntu@a1:~$  
ubuntu@a1:~$  
ubuntu@a1:~$ openstack server delete test  
ubuntu@a1:~$  
ubuntu@a1:~$  
ubuntu@a1:~$
```

Figure 5.4: Co-located VM Deletion Command

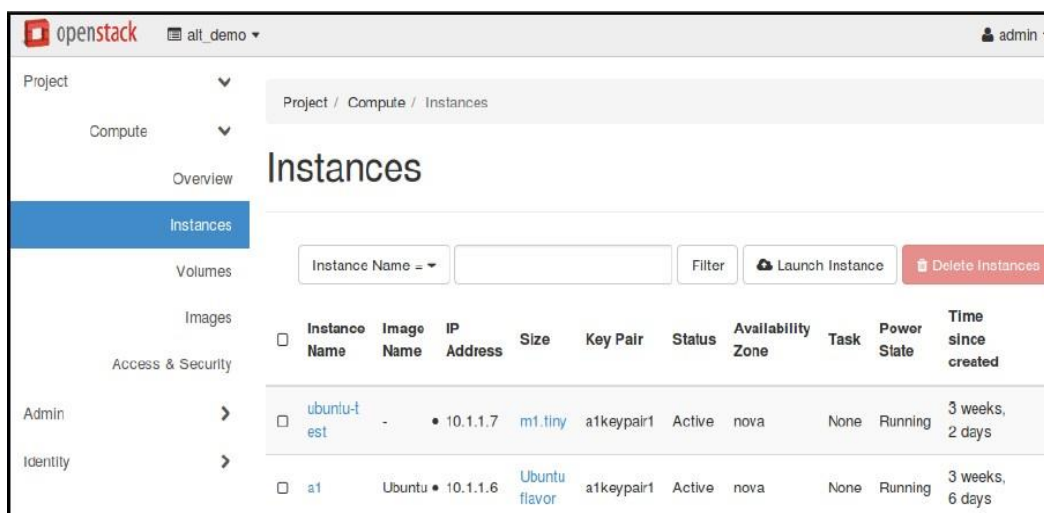
After taking control of the Toolstack, an attacker machine can conduct a denial-of-service assault against a physical machine by establishing many useless virtual machines and reserving a lot of resources for them. Attack results for the OpenStack server's test instance are displayed in Figure 5.5. Figure 5.6 displays the instance list following the delete action.



The screenshot shows the OpenStack dashboard with the 'Instances' page selected. The left sidebar contains navigation links for Project, Compute, Overview, Instances (selected), Volumes, Images, Access & Security, Admin, and Identity. The main content area displays a table of instances. The 'test' instance is highlighted with a red 'Deleting' button. The 'a1' instance is also visible.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created
test	-	10.1.1.9	m1.tiny	-	Active	nova	Deleting	Running	4 minutes
ubuntu-test	-	10.1.1.7	m1.tiny	a1keypair1	Active	nova	None	Running	3 weeks, 2 days
a1	Ubuntu	10.1.1.6	Ubuntu flavor	a1keypair1	Active	nova	None	Running	3 weeks, 6 days

Figure 5.5: Co-located VM Deletion



The screenshot shows the OpenStack dashboard with the 'Instances' page selected. The 'test' instance has been removed from the list. The 'a1' instance remains.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created
ubuntu-test	-	10.1.1.7	m1.tiny	a1keypair1	Active	nova	None	Running	3 weeks, 2 days
a1	Ubuntu	10.1.1.6	Ubuntu flavor	a1keypair1	Active	nova	None	Running	3 weeks, 6 days

Figure 5.6: VMs List After Deletion

Resource Allocation

All co-located VMs' total resource allocations are displayed in Table 5.1 below. But table 5.2 summarises how much resources each VM used throughout the course of a week. Table 5.2's resource usage values are not static; they change over time based on how many virtual machines (VMs) are being used and what applications are being run on them. A runtime measurement was taken to observe the resource use of attacker VMs after a privilege escalation attack was executed.

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

In Figure 5.7, we can see how the attacking virtual machine's resource consumption changed before and after the attack was launched. The pre-attack consumption of resources is depicted by the dashed lines. Since the attacking VM utilises ROP to increase its privilege level, it increases the processing load even if it is running the same set of code in its local memory, which is the main cause of the disparity in resource utilisation.

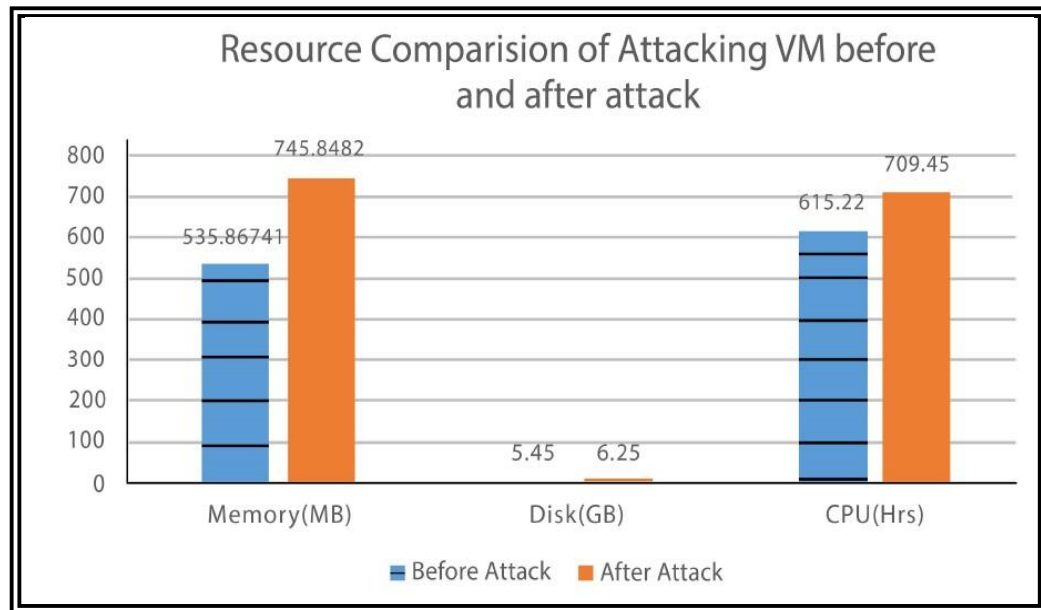


Figure 5.7: Resource Comparison of Attacking VM Before and After Attack

Threat to Microsoft's Cloud

Microsoft's cross-cloud Azure System may be used to conduct the same kind of attack analysis. As a public cloud, Azure competes with the best. In the lab, Azure is set up as the primary IaaS. Microsoft Azure's cloud application hypervisor technology allows businesses and individuals to entirely encapsulate and abstract a multi-virtual machine application and its environment, allowing it to run unmodified on any cloud, public or private.

Setup of a Network

Every virtual machine (VM) in Azure can have its own unique network setup, complete with a static IP address. One must set the Static IP, Netmask, Gateway, and DNS server for this interface. Azure's support for SDN allows for the creation of a virtual switch to be generated based on the IP address of the VM and the netmask of the interfaces. The same virtual switch is used for all interfaces in the same subnet.

The following characteristics apply to a gateway's SDN if it has been assigned via the user interface:

If a Gateway IP has been allocated to a guest VM (that is, if the VM has been configured to act as a Router), then Azure's SDN will behave in line with the VM's Gateway configuration.

Azure's SDN will add a virtual Router to the virtual switch with the specified Gateway IP if the guest virtual machine does not already have one.

Simply doing nothing if the Azure user interface does not have a default Gateway specified for the SDN to utilise.

Two virtual machines, VM1 and VM2, have been set up to simulate a basic network topology for the sake of this experiment. Each virtual machine has two network interface controllers (NICs), which are isolated from one another. While one virtual machine (VM1) is running as root in dom0, the other (VM2) is running as a non-privileged user in domU. An attacking VM, VM2, exploits a network channel to connect to dom0, the cloud's root domain, as explained in Section 5.2.1. If VM3 takes command of the Toolstack, he can potentially instal or remove the designated virtual machines.

A Microsoft Azure account has been set up. Microsoft Azure VM2 has been built for testing purposes. The proposed attack has been tested in VM2, the primary working domain. After that, ssh into VM2. As illustrated in Figure 5.8, we now build two more virtual machines under the root account (VM2), named MyLinuxVM and MyWinVM. Now we'll use the ssh command to log into one of the child virtual machines (VM), called "myLinuxVM," which will be used in the attack.

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

Figure 5.8: Sub VM List

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

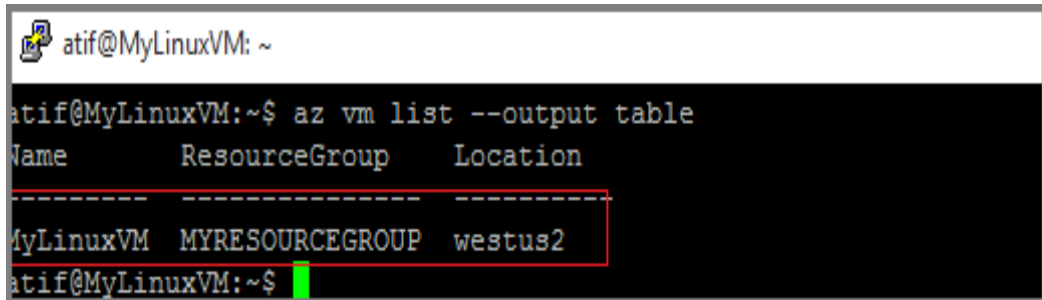
Figure 5.9: SSH to Sub-VM (attacking)

Once the attack mechanism is in place, the attacking machine will destroy any other co-located VMs, as seen in Figure 5.10.

```
atif@MyLinuxVM:~$
atif@MyLinuxVM:~$ az vm delete -n MyWinVM -g MyResourceGroup
Are you sure you want to perform this operation? (y/n): y
{
  "endTime": "2017-04-19T12:24:34.874070+00:00",
  "error": null,
  "name": "4756029f-dbd4-492e-b3e5-a55d20e5c13e",
  "startTime": "2017-04-19T12:22:28.840460+00:00",
  "status": "Succeeded"
}
atif@MyLinuxVM:~$
```

Figure 5.10: Deleting Co-located VMs

Figure 5.10 shows a non-root user using the delete command, which normally requires the root user's permission. Finally, all of the co-located virtual machines are displayed in Figure 5.11.



```

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

```

Figure 5.11: List of All VMs After Attack

Table 5.3 summarises the variations in susceptibility to this attack across popular cloud service providers.

Stopping Network-Channel Attacks

The advantages and disadvantages of several defences against the suggested attack are addressed.

Algorithm that is Unaffected by Network Channels

The primary line of defence could be to alter the OpenStack cloud's open source code to make network-based side-channels less granular. The primary step in initiating the described attack is reusing the code in order to bypass the security perimeter and enter the root domain of the system. Careful analysis of the code and limitations on access to the live system can effectively thwart this assault. There is now a firewall-like security perimeter in place to protect the system. The xenapi connection verification code is depicted in Figure 5.12.

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.

Such a security perimeter's purpose is to prevent intruders from reaching the primary domain. When a new security perimeter is implemented, its underlying logic looks at the code's internal state to spot malicious connectivity from unauthorised users. Because these codes are already in the system's memory and the security perimeter has previously inspected such codes, it treats all of them the same and has vetted them, existing security rules cannot prevent such attacks. Extending the reach of the cloud infrastructure calls for a direct link between the cloud's top-level domains (root and root). As was previously mentioned, Xen's xapi serves as the mechanism

through which such connections are made. The proposed security api will do an internal xapi check and verify the VM's dual registration status upon receiving a connection request. If that's not the case, the connection will be accepted. The suggested security perimeter includes a specialised API that verifies, at runtime, if the local domain of this xapi is connected to any xenapis. If the function returns true, the connection request will be denied. The only catch with this type of security barrier is the time it takes for the network to react. The typical root-to-root link for a growing cloud is seen in Figure 5.13a. The activation of the security check in case of a root-root connection via xapi is depicted in Figure 5.13b. For the purpose of determining whether to allow or deny the connection, this security check will look at the xapi's source code. Figure 5.14 depicts the problem encountered while attempting to access the root directory after the security check API has been executed.

```

20 curr = os.popen( pwd ).read().split( / )[-1].strip() # GET THE CURRENT DIRECTORY AND
21 if curr: # NOT EMPTY DIRECTORY
22     curr = "/" + 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

Discussion

In this section, we'll go over the most important results from the chapter's empirical analysis of Privilege Escalation using the RoP attack.

The goal of this research is to find potential entry points into the main open-source OpenStack cloud platform and the commercial cloud platform, Microsoft Azure, by examining the configuration of the underlying hypervisor.

When cloud growth is authorised, it was found that careful network configuration is required. Given that an attacker can successfully breach the network by making an unauthorised connection to the root domain via network expansion using ROP.

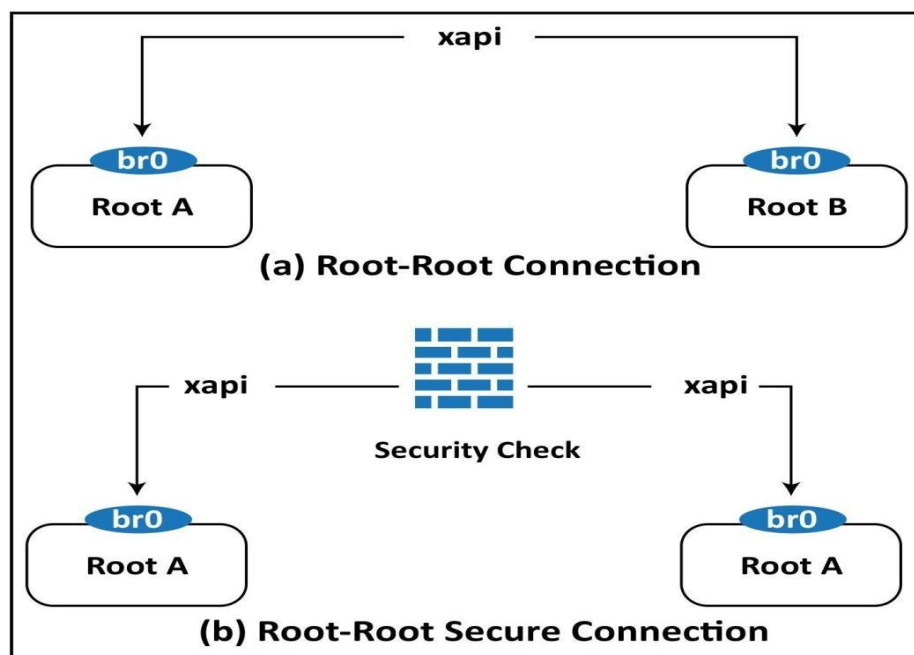
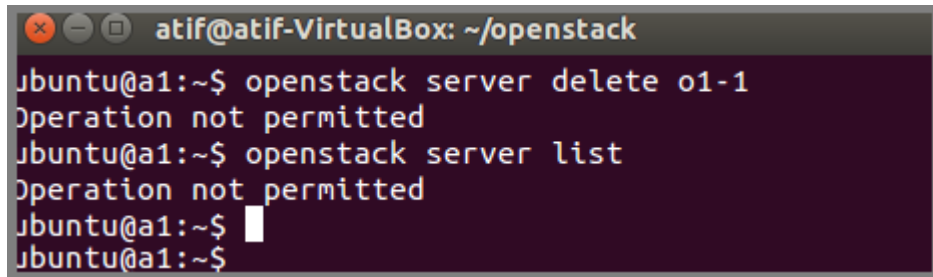


Figure 5.13: Root-Root connections

It was intriguing to see that the tool stack of the root domain, from which other co-located VMs may be managed or controlled, could be successfully controlled in this experimental situation.

A terminal window titled 'atif@atif-VirtualBox: ~/openstack' with a dark background and light-colored text. The window shows a series of commands and their outputs. The first command is 'openstack server delete o1-1', which results in 'Operation not permitted'. The second command is 'openstack server list', which also results in 'Operation not permitted'. The prompt 'ubuntu@a1:~\$' is shown three times, indicating the user is at the root of the VM but cannot execute OpenStack commands.

```
atif@atif-VirtualBox: ~/openstack
ubuntu@a1:~$ openstack server delete o1-1
Operation not permitted
ubuntu@a1:~$ openstack server list
Operation not permitted
ubuntu@a1:~$ 
ubuntu@a1:~$
```

Figure 5.14: Error While Accessing the Root

It was demonstrated that the exploitation of this attack setting can be thwarted through thorough inspection of the request for a network connection from non-root VMs.

The criteria for evaluation (Section 5.3) are as follows:

The attack tactic is assessed for its ability to carry out all of the goals for which it was developed. All of these attack tactics have been successfully developed, as shown in Figure 5.2, and its functional parameter is evaluated by looking at their inference ability. All of the events could be inferred, as demonstrated by the results. The study also revealed that the assault model was successful in its intended role.

In the context of this research, "expressiveness" refers to the degree to which the Privilege escalation assault is capable of performing a wide variety of operations when applied to the network environment. Figures 5.5 and 5.6 demonstrate that a non-root VM may successfully impersonate a root VM, providing compelling evidence for the attack's viability and allowing for an evaluation of the expressive parameter.

As can be seen from the examination, the ability to illicitly connect to the root domain and control tool stack is the approach's greatest strength. The biggest

drawback of this method is that it only supports the XEN architecture and not the other hypervisors like KVM and VMWare.

Summary

This chapter has covered the topic of how to use the ROP (return-oriented programming) methodology. As a result of this vulnerability, the Xen hypervisor is subject to a privilege escalation attack, as the domain separation between root and non-root VMs is compromised. Previous ROP-based attacks have targeted specific software components like operating systems and applications, as well as libraries like libc; however, they have never attempted to exploit network channels by recycling previously used code. Because virtualization is so pervasive, learning more about its domain isolation properties is essential.

This study focuses primarily on the isolation qualities (or lack thereof) of a popular open-source virtual machine manager known as OpenStack as well as the commercial cloud service known as MS-Azure. The obstacles that must be conquered in order to carry out this assault include privilege escalation, ROP, and the difficulty of penetrating the root domain of a system that is currently functioning. A new attack model that was sufficiently influential in managing the victim VM has been created by the integration of these many innovative techniques, which have led to the discovery of new novel ways. The empirical evaluation of privilege escalation reveals that it is successful in penetrating the security perimeter of cloud providers. [Case in point:] [Case in point:] It is of utmost significance to take note of the fact that the utilization of return oriented programming (ROP), which generates very high risk, is present in practically every conceivable configuration that is covered in this

chapter. A countermeasure option has also been presented, and it involves directly editing the open source code of OpenStack, which is the main cloud platform. In a nutshell, the overall strategy that was detailed in the experimental assessment is capable of resolving the issues that were discovered throughout the course of the research, and the researchers were also successful in accomplishing their primary goal. The following chapter provides a synopsis of the research effort that was given in this thesis and focuses on the most important contributions. In addition to this, a number of unsolved research difficulties in the field are discussed, and a variety of potential future study areas are outlined.

Chapter 6

Summary and Conclusions

Summary

The purpose of the study that is given in this thesis is to investigate whether or not a newly discovered flaw in the current network architecture of cloud computing can be made use of in some malicious way. In addition to this, the presentation included the introduction of innovative cross-VM attacks in addition to their empirical study and characterization in leading cloud computing architecture. These attackers' countermeasure solutions have also been presented as potential solutions. Within the realm of cloud computing, an in-depth discussion is held regarding the analytic requirements of leading cloud systems as well as the benefits these requirements provide to the advancement of research and technical processes. In particular, this study gives an examination as well as a method to illustrate the architectural components that are contained inside the environment of cloud computing. These studies are then exploited for practical utilisation, such as the exploitation of network channels, the impersonation of conventional network cards, the use of RoP in conjunction with network channels, the privilege escalation of non-root users, and they have also been used to increase the security perimeter of cloud computing systems.

Overview of Thesis

In this section, the most important findings of this thesis are summarized for your convenience.

In Chapter 2, we discussed the idea of a system model and the various components that make up the model, as well as the evolution of the current distributed system to cloud computing. Cloud computing's deployment patterns, service models, and attack vectors are among the many aspects of its concept and taxonomy that are dissected in great detail here. The idea of security, as well as the ways in which it might be utilized to improve research on cloud computing, is investigated and addressed in considerable detail. It has come to light that although cloud isolation in a shared environment is currently a topic of ongoing research, there is still a difficulty in determining the level of security and privacy it offers. Following this, a demonstration of the concept of system architecture models and how they might be utilized to improve the security of cloud computing is given. We present and analyze in depth the current state of the art in investigation for cloud components, and we also identify current gaps in the relevant body of literature. In conclusion, a presentation on the significance and usefulness of cloud analytics is given, which includes a discussion of the ways in which it may be utilized to improve the practical and commercial operation of cloud computing. Using impersonation and Linux's regular tool, this chapter revealed the gap in exploitation of network channel, which is the primary contribution of the thesis. This contribution was accomplished by combining the two.

The network architecture of the cloud concept is the primary topic of discussion in Chapter 3. This chapter contains a description of the system model, its components, its configurations, as well as various service models and advanced network capabilities, together with their respective constraints, which are employed inside open source and commercial cloud computing systems. The analysis infrastructure also reveals the inner network devices that are employed during communication and consists of the means by which network traffic enters and leaves the system. Everything that allows for the entry and exit of network traffic is part of this architecture. This chapter concludes with a summary of the primary cloudmodels' network architecture, service models used to accomplish isolation, and the network devices through which all network traffic from co-located virtual machines flows. Implementing isolation is also covered in this chapter.

The cross-VM network channel assault is demonstrated in Chapter 4 using two of the most prominent open source and commercial cloud platforms, namely OpenStack and Oracle Ravello System. An original tactic was utilized in the assault that was carried out, which consisted of impersonating a TAP interface and establishing a network mirror within the bridge interface, which is the point through which the network traffic of all virtual machines that were co-located passed. This effective experimental configuration enables attackers to compromise the privacy of co-located VMs by redirecting the network traffic of such VMs to a defined destination point that is under the attacker's control. As the attacking VM is not exceeding the resource limit that has been allotted by the cloud provider, and as all footprints are removed, this study will most likely leave very little to no trace on the

cloud architecture. An empirical analysis of cloud setup, including their setups, constraints, and mitigation method, is also presented in this chapter. The purpose of this analysis is to evaluate the attack in a real-time situation. The purpose of the experiment was to make use of the underlying network channel in order to monitor the network traffic produced by virtual machines that were co-located. The graph that was generated as a result displays the amount of resources that were used by the attacker machine. This attack presents a challenge because the virtual machine that is doing the attacking can use resources up to a limit that has been set, and it does not break any of the resource utilization factors. In the end, we have also researched ways to overcome the obstacles of separating regular resource patterns from target attacks coming from VMs. This was done as part of our work toward completing the project. A countermeasure method has also been given in the form of a patch to the open source code of a cloud platform. This update restricted the connection of any external device without the appropriate parameter defined by the cloud model.

In Chapter 5, a new creative study is presented that raises the privilege level of a non-root user, as well as an empirical evaluation of an assault that successfully crosses the security perimeter of cloud providers. In the cloud model, domains have been defined so that the privilege levels of users, such as root and non-root users, can be distinguished. Users without root privileges often occupy domU, while root users are located in dom0. Users with the root privilege in dom0 have the ability to supervise non-root users in domU. It has been demonstrated in this chapter how a non-root user can break out of domU and build a connection with dom0.

Additionally, it has been demonstrated how a non-root user can carry out the

responsibilities of a root user. In order to accomplish this, the attacking machine made use of RoP, which is a process that runs code already present in the memory of cloud platforms, in connection with network channel. The findings of the analysis demonstrate that such an attack poses a significant risk in each of the three cloud scenarios that were covered in this chapter. At the very end, a countermeasure approach that modifies the open source code of the leading cloud platform, namely OpenStack, has been offered.

Major Findings of Research Study

One of the most important things that has come out of this research has been an investigation into how cross-VM settings might be exploited in the network architecture of the cloud model. A literature study and the development of experimental procedures that deal with real-time settings are both parts of the mixed-methods approach that has been taken with this thesis. This methodology is known as a mixed-methods approach. Because of this, the following contributions have been made.

TAP Impersonation and Mirroring

This thesis offers some crucial new insights into the network architecture of the cloud computing model as well as some specific new perspectives on the issues that are unique to this branch of the scientific discipline. In specifically, the following have been uncovered by this thesis:

- The significance of the idea of network architecture in cloud computing, particularly as it relates to the flow of network traffic for virtual machines (VMs). In cloud computing networking, there is a significant number of small network devices

that are often engaged in the flow of network traffic for virtual machines (VMs). These network devices might be heterogeneous and possibly complicated in nature.

- Conceiving of a zero-day attack in the form of TAP Impersonation and Mirroring in a contemporary cloud computing environment, with the goal of successfully rerouting the network traffic of other co-located virtual machines (VMs). But before we can launch this attack with any degree of success, we need to solve a few fundamental difficulties.

- There are five key problems that have been identified with this attack; these difficulties set this region apart from others in a significant way and call for a unique security response. These difficulties are now publicly known thanks to the attack. Some examples of these problems are as follows: i.) investigating the existing cloud computing network architecture in detail.

ii) Penetration of the current network architecture iii) prioritising its own network traffic above that of any other virtual machines sharing the same physical host (VMs).

iv) the redirection of data packets originating from the co-location VM v) eradicating all physical evidence of the attack by hiding any weapons or other implements used;

- In addition, a countermeasure method is offered, which, at first glance, appears to prevent these attack strategies from being carried out in the contemporary cloud environment.

Privilege Escalation using RoP in conjunction with Network Channel

Additionally, the thesis provides contributions in the form of insights into existing practices of hypervisor architecture in cross-VM scenarios. These contributions

include a significant exploitation of privilege escalation utilizing RoP. The following are a few of the realizations that have resulted from this work:

- The practices of conventional privilege escalation attack strategies are insufficient, suffer from methodological limitations (old technologies), and are easy to counteract due to the advanced security feature of the cloud computing model. This is the case because of the fact that cloud computing is a model.
- The assault that is being presented addresses important issues related with the escalation of privilege level, which comprises the following: i) making efficient use of RoP ii) access to the root domain from a domain that is not the root domain by exploiting a point of presence (RoP) in conjunction with a network channel iii) a control tool stack for managing the other virtual machines (VMs).
- The repercussions of this attack are so alarming that it puts other co-located virtual machines (VMs) in jeopardy in terms of the data and resources they possess.
- A countermeasure approach to this attack is also offered, which prohibits unlawful connections from virtual machines that do not have root privileges.

Revisiting the Research Goals

In this section, the primary contributions made by the thesis are discussed by returning to the study objectives outlined in Section It ought to be obvious at this point that there is a solid mapping between the contributions and the initial research topics, which are as follows:

1. Conducting research on the network architecture and the components that are associated with it in order to develop a zero-day attack model for the purpose of exploiting a vulnerability in the network architecture of the cloud computing model.

The first objective has been accomplished by the invention of a TAP impersonation and mirroring attack model, which is detailed in Chapter 4.

2. The research of return-oriented programming (RoP) and associated approaches for the exploitation of the hypervisor, as well as the identification of the benefits and drawbacks of this method.

The Privilege Escalation assault, which is implemented by the exploitation of RoP in conjunction with network channel as outlined in Chapter 5, has been successful in achieving the second target. This attack was provided via the Privilege Escalation attack.

3. In order to suggest a remedy for mitigating the problem, evaluate the entire strategy by applying it to real-world scenarios that were created from the study of the relevant literature.

The Evaluation and Mitigation techniques, which are discussed in Sections 4.8 and 5.7, have been successful in achieving the third objective of the project.

Future Work

On the basis of the findings presented in this thesis, the following part will examine some of the potential future avenues that research could proceed in.

Implementation of Cloud Model on Mobile Platforms

Numerous cloud service providers have moved their operations onto mobile platforms, which are part of the cloud platform. Implementing the previously carried out attack on mobile infrastructure is something that could be done in the future. It is impossible to deny that the introduction of mobile applications has also ushered in a

new era that is fraught with opportunities for malicious actors to launch assaults. When a user is using an app or exchanging information on mobile platforms, contextual information plays a more significant role. Examples of this type of information include location. It has already been demonstrated that co-location in and of itself poses a significant risk. The attacking machine will first attempt to co-locate with the target as the first phase in the attack. Extending the design of mobile applications such that user data can be kept private is one potential area for future research and development.

RoP on KVM or other VMM

It has been demonstrated that RoP poses a significant risk to cloud models. Experiments with RoP were carried out in OpenStack, which is a cloud platform that runs xen in the background and is free to use. It's possible that in the future, we'll be concentrating on KVM or some other hypervisor in cloud model. Because each hypervisor utilizes its own unique setups and settings, exploiting a hypervisor calls for the utilization of a variety of distinct illusions.

Analysis Extension

The study that is described in this thesis has the potential to be expanded so that it can investigate a variety of other system contexts. As was discussed in chapter 5, the extension of workload models involves selecting virtual machines (VMs) based on the resource use of each candidate. It is feasible to do an analysis of the VMs based on the tasks that they perform individually. It is possible that future work will involve applying the workload analysis method that has been outlined in order to determine the characteristics and behavior of the attacking machine in comparison

with individual activities. In addition, the workload model does not include the development constraints of tasks onto servers. These limitations can be incorporated to investigate the behavior of attacking machines in comparison to non-constrained jobs. However, the model does not include this information.

Conclusions

This thesis details research on the challenges of privacy breaches for co-located VMs in the cloud paradigm, specifically with regard to traffic redirection of target machines and privilege escalation. These issues were uncovered as a direct result of the research presented in this thesis. The main contributions of this thesis are the impersonation attack, which may redirect the live network traffic of other co-located VMs, and the privilege escalation, which can use RoP to exploit network-channel.

Impersonation attack redirects real-time network traffic of other co-located VMs. These assaults are carried out in real time on open source and commercial cloud platforms, each of which has been designed to meet a unique set of security needs. The evaluation of these assaults that is described in this thesis reveals a flaw in the network channel of cloud architecture that can be exploited by an attacker machine. When carrying out this implementation strategy, it is crucial to take extra precautions to guarantee the security measures taken when setting up the network channel in the cloud model. We cannot stress the significance of this enough.

References

1. 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.
2. 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.
3. B. Blunden, *The Rootkit arsenal: Escape and evasion in the dark corners of the system*. Jones & Bartlett Publishers, 2012.
4. 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.
5. 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.
6. B. McCorkendale and P. Ferrie, —Using a hypervisor to provide computer security, Aug. 9 2011. US Patent 7,996,836.
7. B. Tuan-Anh, M. Canini, V. H. Tran, and R. Sadre, —Cloud network performance analysis: An openstack case study, 2016.
8. B.-A. Yassour, M. Ben-Yehuda, and O. Wasserman, —Direct device assignment for untrusted fully-virtualized virtual machines, 2008.

9. 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.
10. 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.
11. C. Intel, —Improving real-time performance by utilizing cache allocation technology,|| Intel Corporation, April, 2015.
12. 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.
13. 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.
14. 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.
15. 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.
16. 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.
 17. 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.
 18. D. Hyde, —A survey on the security of virtual machines,|| Dept. of Comp. Science, Washington Univ. in St. Louis, Tech. Rep, 2009.
 19. D. Marshall, —Understanding full virtualization, paravirtualization, and hardware assist,|| VMWare White Paper, p. 17, 2007.
 20. 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.
 21. D. Page, —Defending against cache-based side-channel attacks,|| Information Security Technical Report, vol. 8, no. 1, pp. 30–44, 2003.
 22. D. Zissis and D. Lekkas, —Addressing cloud computing security issues,|| Future Generation computer systems, vol. 28, no. 3, pp. 583–592, 2012.
 23. 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.

24. e the cloud, in 24th {USENIX} Security Symposium ({USENIX} Security 15), pp. 929–944, 2015.
25. 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.
26. 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.
27. E. Pulier, F. Martinez, and D. C. Hill, –System and method for a cloud computing abstraction layer, Jan. 6 2015.
28. 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.
29. 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.
30. 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.
31. 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.

32. 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.
33. 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.
34. 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.
35. 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.
36. 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.
37. G. Irazoqui, T. Eisenbarth, and B. Sunar, —Mascot: Stopping microarchitectural attacks before execution.,|| IACR Cryptology ePrint Archive, vol. 2016, p.1196, 2016.
38. 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.

39. Grobauer, T. Walloschek, and E. Stocker, —Understanding cloud computing vulnerabilities,|| IEEE Security & Privacy, vol. 9, no. 2, pp. 50–57, 2011.
40. H. Debar, M. Dacier, and A. Wespi, —Towards a taxonomy of intrusion-detection systems,|| Computer Networks, vol. 31, no. 8, pp. 805–822, 1999.
41. 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.
42. 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.
43. 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.
44. 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.
45. 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.

46. 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.
47. 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.
48. 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.
49. 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.
50. 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.
51. 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.
52. 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.

53. 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.
54. J. Rutkowska, -Subverting vistatm kernel for fun and profit,|| Black Hat Briefings, 2006.
55. 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.
56. 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.
57. 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.
58. 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.
59. 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.
60. 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.
 61. 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.
 62. 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.
 63. 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.
 64. 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.
 65. Journal in Computer Virology, vol. 8, no. 3, pp. 85–97, 2012.
 66. K. Chiang and L. Lloyd, —A case study of the rustock rootkit and spam bot.,^{||} HotBots, vol. 7, no. 10-10, p. 7, 2007.
 67. 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
68. Korkin and I. Nesterov, —Applying memory forensics to rootkit detection,|| arXiv preprint arXiv:1506.04129, 2015.
 69. Kozłowski, —Comparative analysis of cyberattacks on estonia, georgia and kyrgyzstan,|| European Scientific Journal, ESJ, vol. 10, no. 7, 2014.
 70. 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.
 71. M. Ali, S. U. Khan, and A. V. Vasilakos, —Security in cloud computing: Opportunities and challenges,|| Information sciences, vol. 305, pp. 357–383, 2015.
 72. 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.
 73. 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.
 74. 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.

75. 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.
76. M. Georgiev and V. Shmatikov, —Gone in six characters: Short urls considered harmful for cloud services, arXiv preprint arXiv:1604.02734, 2016.
77. 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.
78. 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.
79. M. Pomonis, T. Petsios, A. D. Keromytis, M. Polychronakis, and V. P. Kemerlis, —kr^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.
80. 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.
81. 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.

82. N. Kumar and V. Kumar, —Vbootkit: Compromising windows vista security,|| Black Hat Europe, vol. 2007, 2007.
83. 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.
84. O. Aciğmez, 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.
85. 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.
86. P. Guide, —Intel® 64 and ia-32 architectures software developer’s manual,|| Volume 3B: System programming Guide, Part, vol. 2, 2011.
87. 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.
88. P. Mell, T. Grance, et al., —The nist definition of cloud computing,|| 2011.
89. P. Ranjith, C. Priya, and K. Shalini, —On covert channels between virtual machines,||
90. 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.

91. 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.
92. 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.
93. 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.
94. 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.
95. R. L. Krutz and R. D. Vines, Cloud security: A comprehensive guide to secure cloud comput- ing, 2010.
96. R. Ledyayev and H. Richter, —High performance computing in a cloud using openstack,|| Cloud Computing, pp. 108–113, 2014.
97. R. M. Yoo and H.-H. S. Lee, –Adaptive transaction scheduling for transactional memory sys- tems,|| in Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures, pp. 169–178, ACM, 2008.
98. R. Masoudi and A. Ghaffari, —Software defined networks: A survey,|| Journal of Network and Computer Applications, vol. 67, pp. 1–25, 2016.

99. R. Mohandas, V. Thomas, and P. Ramagopal, —Malicious media files: Coming to a computer near you,||
100. 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.
101. 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.
102. 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.
103. 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.
104. 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.
105. 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
106. 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.

107. S. Iqbal, M. L. M. Kiah, B. Dhaghghi, 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.
108. S. J. Murdoch and S. Lewis, —Embedding covert channels into tcp/ip,|| in International Workshop on Information Hiding, pp. 247–261, Springer, 2005.
109. 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 oppor- tunities,|| in Resilience Assessment and Evaluation of Computing Systems, pp. 65–81, Springer, 2012.
110. 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.
111. 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.
112. 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.
113. 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.

114. 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.
115. S. Ristov, M. Gusev, and A. Donevski, —Openstack cloud security vulnerabilities from inside and outside, Cloud Computing, pp. 101–107, 2013.
116. S. Sparks and J. Butler, —Shadow walker: Raising the bar for rootkit detection, Black Hat Japan, vol. 11, no. 63, pp. 504–533, 2005.
117. 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.
118. 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.
119. S. Zhang, —Deep-diving into an easily-overlooked threat: Inter-vm attacks, tech. rep., Technical Report). Manhattan, Kansas: Kansas State University, 2012.
120. 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.
121. 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.

122. 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.
123. T. Garfinkel and M. Rosenblum, –When virtual is harder than real: Security challenges in virtual machine based computing environments.,‖ in HotOS, 2005.
124. T. Garfinkel, M. Rosenblum, et al., —A virtual machine introspection based architecture for intrusion detection.,‖ in Ndss, vol. 3, pp. 191–206, 2003.
125. 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.
126. T. M. O. Mutlu, –Memory performance attacks: Denial of memory service in multi-core systems,‖ in USENIX security, 2007.
127. T. Olzak, —Secure hypervisor-based virtual server environments,‖ Tech Republic, 2007.
128. T. Ormandy, —An empirical study into the security exposure to hosts of hostile virtualized environments,‖ 2007.
129. T. Ormandy, —An empirical study into the security exposure to hosts of hostile virtualized environments,‖ 2007.
130. 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.
131. T. T. W. Group et al., —The treacherous 12: cloud computing top threats in 2016,|| Cloud Security Alliance, 2016.
 132. Tereshkin and R. Wojtczuk, —Introducing ring-3 rootkits,|| Black Hat USA, 2009.
 133. U. Tupakula, V. Varadharajan, and N. Akku, —Intrusion detection techniques for infrastruc- ture as a service cloud,|| in 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, pp. 744–751, IEEE, 2011.
 134. 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.
 135. 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.
 136. 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.
 137. Vasudevan, S. Chaki, L. Jia, J. McCune, J. Newsome, and A. Datta, —Design, implemen- tation and verification of an extensible and modular hypervisor

- framework,|| in 2013 IEEE Symposium on Security and Privacy, pp. 430–444, IEEE, 2013.
138. 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.
 139. 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.
 140. 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.
 141. Y. Azar, S. Kamara, I. Menache, M. Raykova, and F. B. Shepard, —Co-location-resistant clouds.,|| CCSW, vol. 14, pp. 9–20, 2014.
 142. 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.
 143. 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.

144. 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.
145. 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.
146. 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.
147. 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.
148. 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.
149. 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.
150. 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.
151. Z. Afoulki, A. Bousquet, and J. Rouzaud-Cornabas, —A security-aware scheduler for virtual machines on iaas clouds,|| Report 2011, 2011.
 152. Z. Tari, —Security and privacy in cloud computing.,|| IEEE Cloud Computing, vol. 1, no. 1, pp. 54–57, 2014.
 153. 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.
 154. 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.
 155. 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.
 156. 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.
 157. 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.

158. Z. Xu, H. Wang, Z. Xu, and X. Wang, —Power attack: An increasing threat to data centers.,‡ in NDSS, 2014.
159. Z. Xu, H. Wang, Z. Xu, and X. Wang, —Power attack: An increasing threat to data centers.,‡ in NDSS, 2014.
160. 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.