UtilitasMathematica
ISSN 0315-3681 Volume 120, 2023

Analyzing and Mitigating Cross-VM Network Channel Attacks in Cloud Computing Environments

Chandramani Singh[1], Vaishali Singh[2], Mohd Waris Khan[3*]

[1,2]Department of Computer Science, Maharishi University of Information Technology, Lucknow, 226013, Uttar Pradesh, India

[3]Department of Computer Application, Integral University, Lucknow, 226026, Uttar Pradesh, India

[*]Corresponding Author: Mohd. Waris Khan. Email: waris.khan070@gmail.com

Abstract

Cloud computing has gained popularity due to its service approach, which allows for versatile and on-demand availability of resources for computing. This infrastructure is the foundation for modern business operations. Simultaneously, cloud computing security emerges as a critical concern for enterprises when transferring critical data to geographically dispersed cloud platforms outside of their direct jurisdiction and control. The primary goal of this research is to investigate the impact of virtualization on network safety vulnerabilities in cloud computing and to present the findings. Researchers discovered flaws such as side channel attacks on shared hardware, which allow attackers to extract sensitive data from co-resident virtual machines (VMs). These side channels function as a security entry point, allowing malicious users to spy on data from other VMs. Cloud providers address this by implementing logical resource segregation using internal virtual networks, preventing interference between VM operations on the same host. However, current research lacks empirical exploration of network assaults and countermeasures, primarily focusing on IP, ARP sniffing, and spoofing, thus endangering co-located VM network traffic. While countermeasures do exist, literature overlooks internal device-based network design and comprehensive cloud network isolation. The paper tackles these concerns and showcases the proposed attack by testing it on two prominent IaaS cloud platforms: OpenStack, an open-source IaaS management system, and Oracle Ravello Systems, a publicly accessible IaaS provider, both of which have been set up with specific security requirements. In addition, a counter-measure to the reported attack is proposed.

Keywords: Cloud Computing, Virtual Machine Monitor, Cross-VM Network-Channel attacks.

1. Introduction

The term "cloud" refers to a virtualization platform established on the Internet [1]. Clouds present diverse configurations, incorporating grid, distributed, virtualization, utility, and parallel models. Users can seamlessly access services provided by the cloud, allowing on-demand access to various resources, including storage, networks, computation, and applications. The user-friendly design embodies the concept of "pay as you go" (also known as "pay as you use") [2]. Individual resource procurement is unnecessary; instead, users can pay for resource access and adjust them according to their needs. The cloud offers numerous benefits, including pooling resources, on-demand self-service, broad access to networks, measured assistance, and instant

customization. Additionally, it supports various service models like Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS) [3]. The hypervisor, situated at the highest level of the operating system, manages both the virtualization environment and the network layers [3].

Cloud computing infrastructure depends on the hypervisor, a critical component. Hypervisors usually feature a somewhat restricted interface, falling into two primary categories: Type-I and Type-II. A Type-I, termed native or bare-metal hypervisor, requires only a few lines of code on the host operating system. An example of a Type-I hypervisor is the Xen virtual machine. Conversely, a Type-II hypervisor, known as a hosted hypervisor, operates within a conventional operating system environment. Notable examples of Type-II hypervisors include VMware and Virtual Box [4].

Emerging virtualization technologies such as HyperV[5], Xen [6], and VMWare [7] are foundational elements in the security architecture of cloud computing systems. The virtualized nature of the cloud permits the concurrent existence of numerous virtual machines (VMs) on shared physical hardware. The substantial isolation between co-resident virtual machines significantly enhances their desirability [8]. This implies that the operation of a guest virtual machine on the same system as other guest VMs remains unaffected, highlighting a key attribute of their operational integrity. Each virtual machine (VM) operates under its unique set of limitations and constraints. Robust isolation, a fundamental characteristic in public cloud computing platforms like Microsoft Windows Azure [9], EC2 [10], and Rackspace [11], notably bolsters the overall security level of these systems.

Establishing an internal virtual network by a virtual machine manager (VMM) plays a crucial role in ensuring logical isolation among virtual machines (VMs) in virtualization systems [12]. However, the possibility of an attack still exists. Recent studies have explored methods to achieve co-residency and elucidated how adversaries might exploit shared hardware in cross-VM environments. Researchers in [13] and [14] have investigated various methods by which a malicious virtual machine (VM) can exploit network connections, shared memory, and other shared hardware resources to gain unauthorized access to other VMs. Additionally, they discuss the exploitation of network channels by an attacking VM to redirect network traffic among co-located VMs, highlighting limitations within the current cloud model.

Most challenges discussed revolve around implementing additional layers of isolation by cloud providers between co-resident virtual machines (VMs), particularly when compared to non-virtualized settings. This heightened isolation results from the common practice of assigning attackers and victims to separate segments within virtual networks [15]. In contrast, the proposed attack seeks to covertly redirect the network traffic of the victim VM, posing identification challenges as it leaves minimal or no traces on the network.

Addressing these vulnerabilities entails overcoming several technological obstacles based on the presented findings. The implementation of these findings has been demonstrated in widely recognized open-source cloud models like OpenStack and in commercial cloud models such as the Oracle Ravello system. This demonstration included guidance on navigating and addressing the challenges associated with exploiting these vulnerabilities.

The paper's organization is as follows: Section 2 presents prior research, outlines the identified issue, and lists technical challenges faced. Section 3 details the specific attack environment and delineates five challenges encountered within this setting. Section 4 discusses the parameters and criteria for the evaluation process, covering experiment setup, scenarios, and

security configurations. Section 5 evaluates the outcomes based on the specified criteria, considering limitations and potential interpretations. It also explores and delves into the results, addressing implications, additional considerations. Section 6 of the paper outlines the discussion and proposes potential future directions for research. Subsequently, in Section 7, the author offers a comprehensive conclusion, providing a clear and insightful explanation of the work.

2. Previous Work

Researchers have discerned various cross-VM attacks. In [16], the authors categorized network channel attacks into three distinct types: spoofed ARP, virtual hub, and ARP Poisoning. In an ARP spoofing attack, the attacking VM forges an identical IP address within the target VM and transmits a falsified ARP request to the virtual router. Upon receipt of the spoofed ARP request, the virtual router updates its routing table. Consequently, traffic intended for the target VM is erroneously directed to the attacking VM, which can then choose to engage in either sniffing or modification. In the bridge network configuration mode, the bridge functions akin to a virtual hub. Every VM utilizes the virtual hub as a means of network communication. An attacking VM can utilize a sniffing tool, such as Wireshark [17], to intercept and analyze communication within the virtual network. In router network mode [18], the router takes on the role of a virtual switch, utilizing dedicated virtual interfaces to connect each VM. In this specific configuration, a malicious VM has the capability to execute ARP poisoning [19], redirecting packets toward itself, and subsequently intercepting packets traveling to and from other VMs.

ROP-based rootkits have been developed for the Windows operating system, functioning at the kernel layer. Upon execution, SecVisor [20] can effectively conceal malicious processes, files, and network connections within the Windows environment. These rootkits demonstrate an ability to bypass kernel integrity protection systems. Additionally, the ROP technique is applicable to exploiting Apple iPhone [21], enabling unauthorized users to install applications or compromise the security of the customer's SMS database [22].

In [23], the authors introduced the ZombieLoad attack, revealing a novel Meltdown-type effect within the previously unexplored fill-buffer logic of processors. Their analysis highlights that faulting load instructions, necessitating reissuing due to architectural or micro-architectural reasons, may transiently dereference unauthorized destinations brought into the fill buffer by the current or a sibling logical CPU. Consequently, they documented data leakage of recently loaded stale values across logical cores.

In [24], the authors introduced MemJam, a technique employing aliasing to establish a side-channel attack exploiting the false dependency of memory read-after-write events. This approach yields a high-quality intra-cache line timing channel. As a proof of concept, they demonstrated the first key recovery attacks on constant-time implementations of all symmetric block ciphers supported in the current Intel integrated performance primitives (Intel IPP) cryptographic library, including triple DES, AES, and SM4.

These methodologies rely on measuring code reliability, proving effective in detecting attacks involving the modification of hypervisor code or injection of external malicious code. However, the proposed attack model employing ROP cannot be detected by these defense mechanisms. This is attributed to the fact that the proposed attack scheme doesn't involve the injection of external code or modification of the hypervisor code.

Statement of the Problem

Numerous virtual machines share common network resources, thereby rendering these resources susceptible to potential security breaches, as previously discussed. The robust security configuration within the cloud network paradigm poses a heightened challenge to the real-time exploitation of such vulnerabilities. This paper aims to elucidate these security considerations and formulate a zero-day assault model. The primary focus is on dissecting the key elements of the underlying network architecture, elucidating the technique employed, and detailing the intricate interconnections, processing methods, and utilization of network components within the model.

The ultimate objective is the attainment of the specified research goal. The development of a real-time system tailored for OpenStack and Oracle Ravello serves as a validation mechanism for the foundational concepts underpinning the novel attack model. To gauge the actual efficacy of the proposed assault model, diverse tests were systematically conducted. The rationale behind these tests was to showcase the practicability of implementing the system within a functional operational setting. The concluding section proffers a defensive approach aimed at thwarting attacks of this nature.

Technical Challenges

The recommended attack strategy involves initiating a network channel assault, as substantiated by a laboratory testbed. This particular attack enables a malicious virtual machine (VM) to discreetly monitor the network traffic of designated victim VMs, exploiting the inherent isolation between them. The demonstration of this proposed attack employs OpenStack and Ravello cloud systems as a case study, elucidating the deployment of a mirror within the internal medium of the network channel to optimize its positioning. Following the mirror placement, the network traffic of the targeted virtual machine is redirected to a specified destination. Network channels, particularly vulnerabilities related to cross-VM information leakage due to shared network resources, serve as primary points of entry for such attacks, exemplified by instances like network bridges. A precedent attack on network resources involved redirecting the network traffic of a targeted virtual machine through the utilization of address resolution protocol (ARP) spoofing [25].

Table 1: Comparison of related work and proposed work

Author's Name	Attack	Description
Y. Zhang et al., [26]	Side Channel	Time-driven, Access-driven, Trace-driven
J. Rutkowska [27], S. J. Murdoch, et al., [28] Y. Zhang, et al., [29] P. Ranjith, et al., [30] V. Nirmala, et al., [31]	Covert Channel	TCP/IP Steganography TCP/IP header Steganography timing Channel
S. M. Hashemi, et al., [32] H. Wu, et al., [33]	DoS	Illegal use of resources
H.-C. Li et al., [19]	Network Channel	ARP Poisoning Sniffing Spoofing
	Proposed Approach	TAP Impersonation and mirroring

Table 1 furnishes a concise comparative analysis of existing assault methodologies, juxtaposed with our newly proposed approach. Elaborative information on these assault techniques is encompassed in the third column of the table.

3. Attack Setting and Challenges

This research is primarily centered on leveraging the network channel within cloud computing through the deployment of diverse attack strategies. In this section, we delve into a detailed examination of the attack methodologies, encompassing discussions on their challenges, limitations, and various stages.

Attack Setting

In the experimental investigations, it is assumed that an attacker has gained control of a virtual machine (VM) co-resident on the same physical computer as the victim VM [34]. This control is achieved through compromising an existing VM that shares residency with the victim VM. OpenStack [35], 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 [36, 37].

Qubes [38] 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 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 [39]. 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 1.

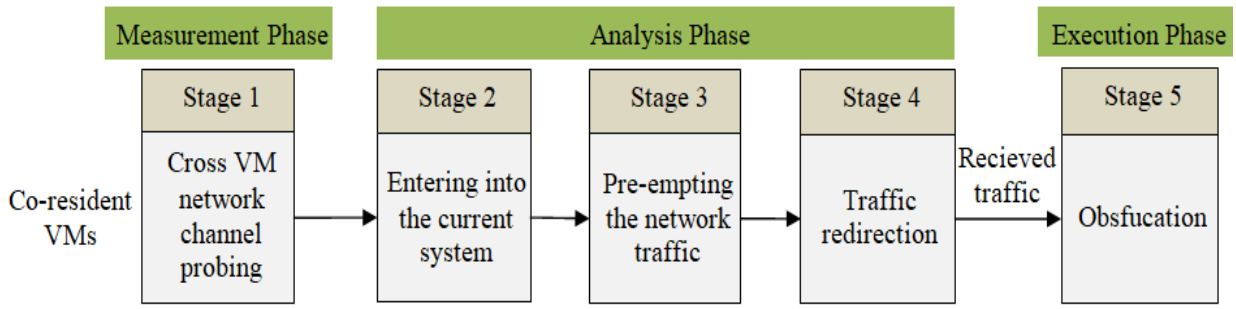


Figure 1: Main Attack Steps for a Network Channel

Challenge 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 innetwork 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 analyze 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 1, 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 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 utilizing 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.

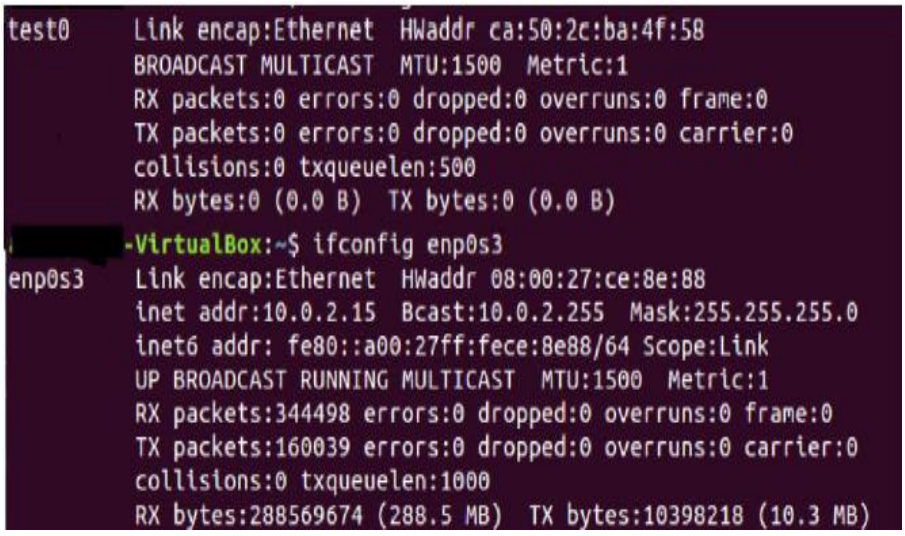


Figure 2: Difference between Tap0 and eth0

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 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.

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 [19] 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 utilized. By carrying out the activities in the following list, this obstacle has been conquered. The utilization of a mirror provides a solution to the initial problem that the method under consideration presents. In particular for Linux, a mirror was utilized 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 favorable 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 3.

Challenge 4: Re-direction

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 figure 3's (4), are the ones that are used to do this re-direction 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 utilized 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 [40]. Route [41] 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 analyzed.

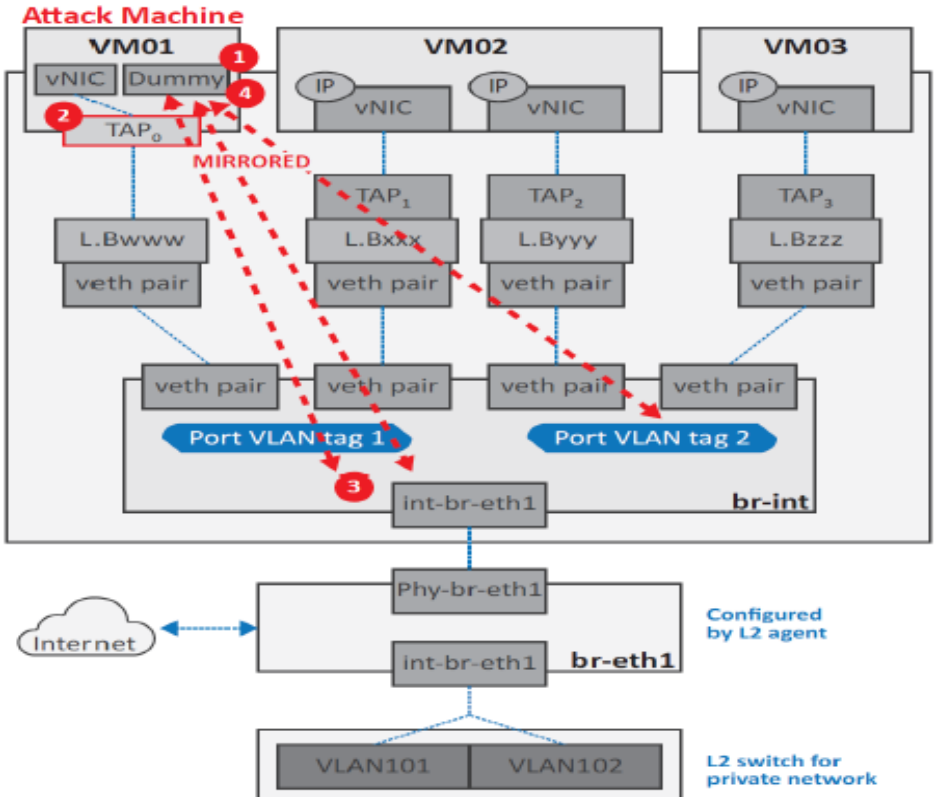


Figure 3: Attack Scenario in OpenStack

Combining all the preceding steps, we can orchestrate a network attack that traverses multiple virtual machines, as depicted in Figure 3. This particular attack exploits the capability provided by the target's cloud provider, which permits the utilization of a TAP interface for internet connectivity, even in the absence of a private Ethernet connection at the interface's backend.

4. Evaluation Criteria

It is clear from looking at Figure 3 that the research approach covered 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.

Experiment Setup

OpenStack's cloud architecture utilizes 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 realize 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 utilizing 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 2. On the other hand, the statistics of VM resource use for one week have been compiled and presented in Table 3. The value of resource usage that is discussed in section 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 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 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 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_flag file=/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 utilized 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. The process of creating a security group consists of two key stages. Firstly, the group is instantiated through the execution of the nova secgroup-create command. Subsequently, the guidelines governing the group's behavior are established using the nova secgroup-add-rule command. This step guarantees that only traffic conforming to the predefined rules of the security group is permitted to ingress or egress. Any traffic failing to meet these criteria is promptly blocked. Additionally, when a new security group is generated, default rules are automatically implemented to either permit or prohibit all inbound and outbound traffic.

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 [34]. 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 [34]. 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) of the network packet that travels from the attacker to the target virtual machine [42]. 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.

5. 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 utilized. 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.

In this actual-time scenario, the two targeted virtual machines (VMs) (VM2 and VM3) are pinging each other. Figure 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 2. Figure 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.

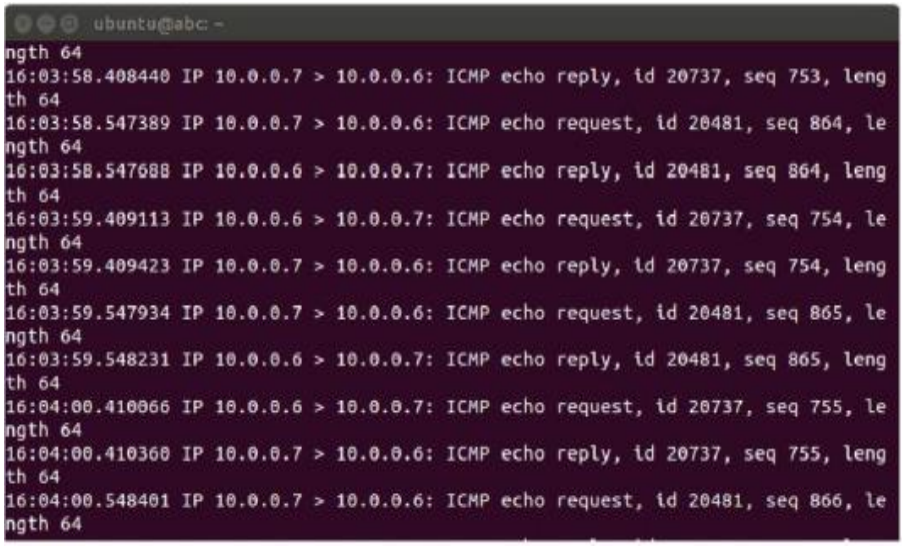


Figure 4: Traffic Capturing at Attacking VM

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

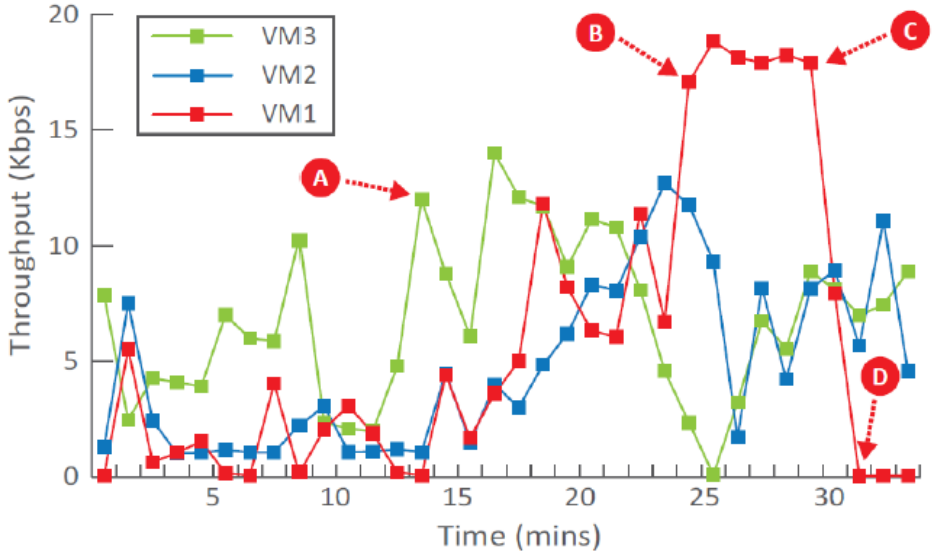


Figure 5: Normal Co-residing VM-Network Traffic

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 utilization of virtual machine (VM) resources is regarded as a mystery by the provider in many public cloud environments. This is considered to be standard practice 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 [43], 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 [44] 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 6, then it is much more difficult to notice unusual patterns of traffic on the network. Network intrusion detection systems (NIDS) [45] 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 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 analyze all incoming and outgoing data; nevertheless, this could cause a bottleneck and slow down the network.

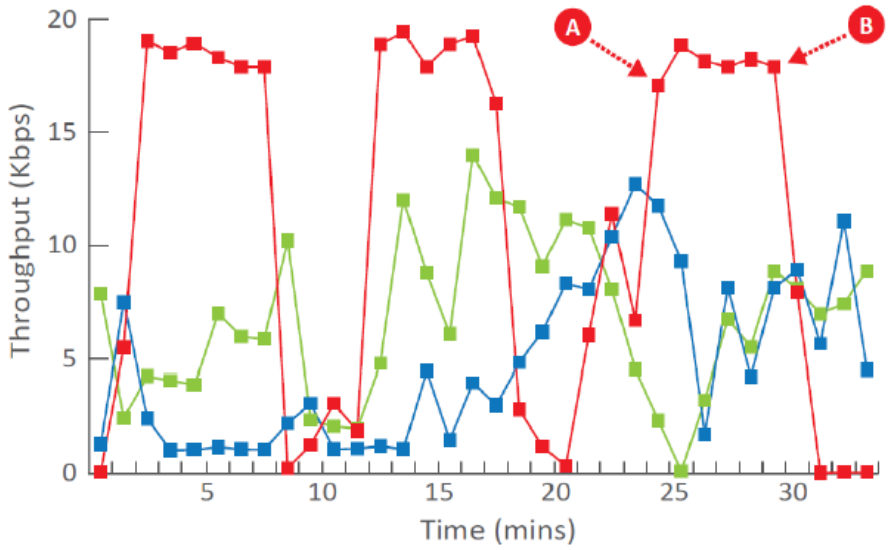


Figure 6: Cyclic Attack Pattern of Network Traffic

When evaluating the attack, a more extensive network system is taken into consideration, as depicted in Figure 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 7.

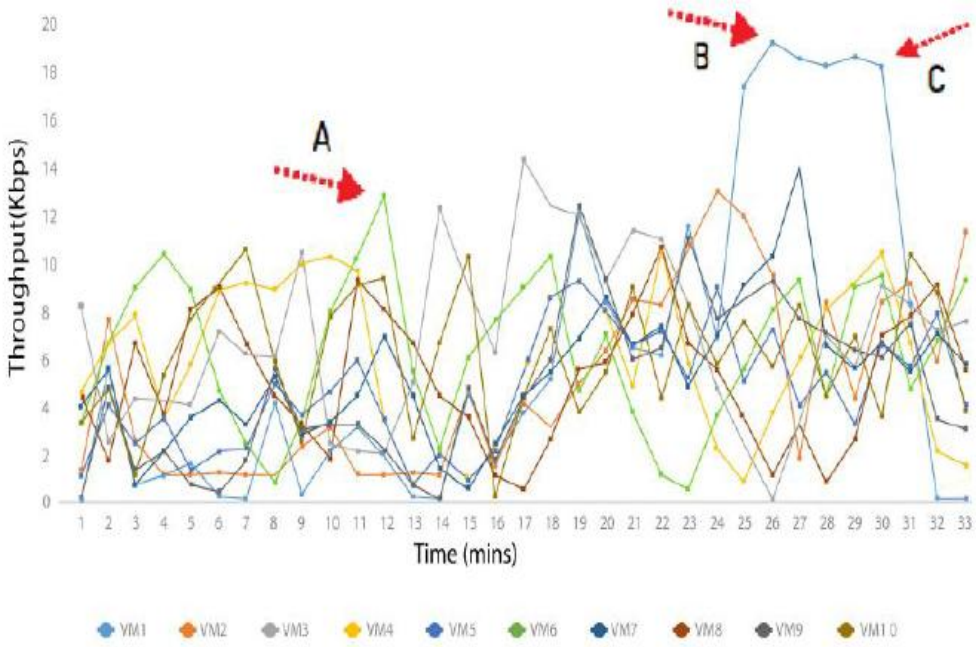


Figure 7: Cyclic Attack Pattern of Network Traffic

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) [43] is utilized.

Attack on Ravello Systems

On Oracle's cross-cloud platform, Ravello Systems, the same assault may be analyzed for its effectiveness. Oracle Ravello is a commercial cloud platform that enables users to utilize 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 8. Tagging of VLANs is described in IEEE Standard 802.1Q [46]. VLAN tags may be included in components of the network that support the VLAN standard.

When a packet enters a VLAN-configured segment of the network, a tag is added to indicate its VLAN membership. VLAN ensures the complete separation of network traffic among all VMs. The network configurations of all three virtual machines are illustrated in Figure 8. As shown in the figure, the attacking VM is positioned to intercept the network traffic between the target VMs during their communication via the ping command.

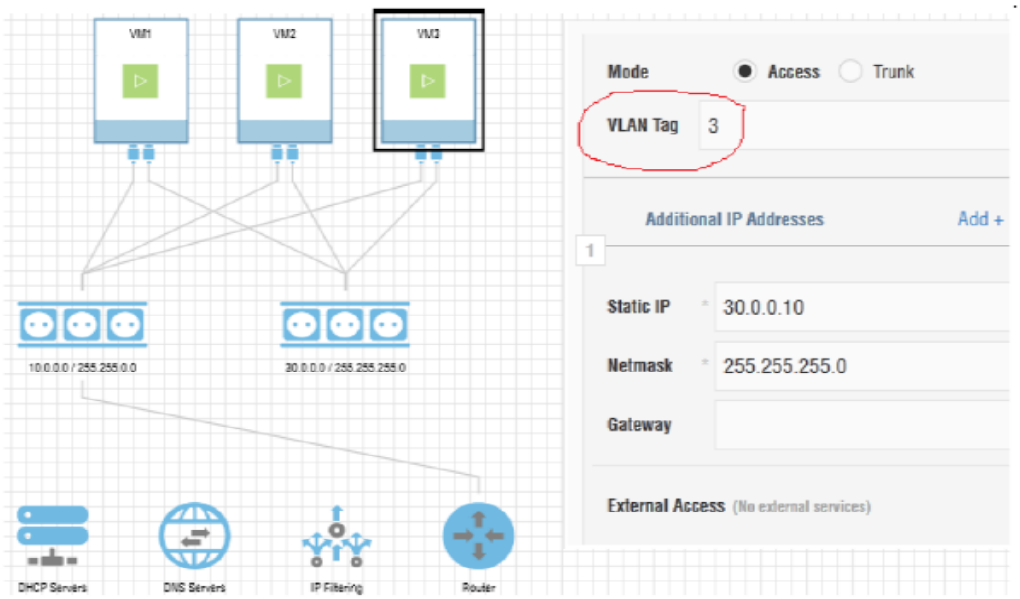
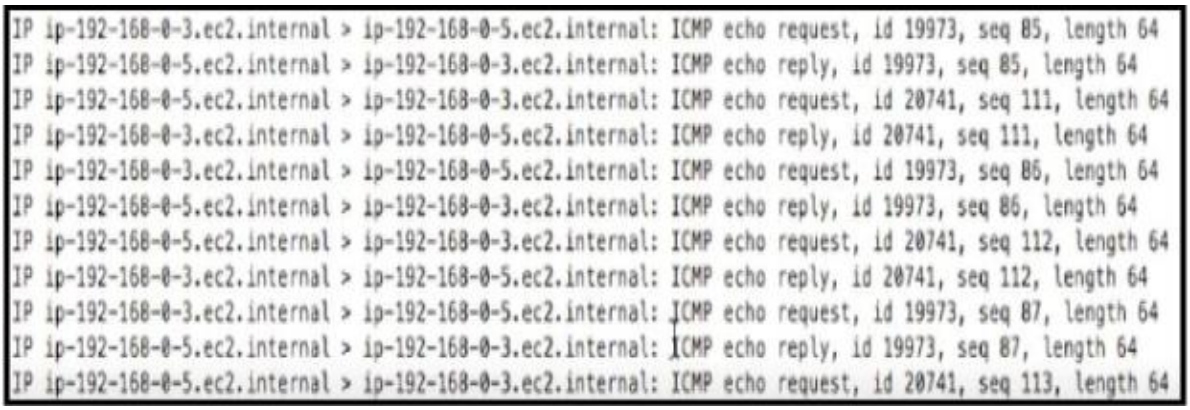


Figure 8: VLAN Configuration of VMs in Oracle Ravello System

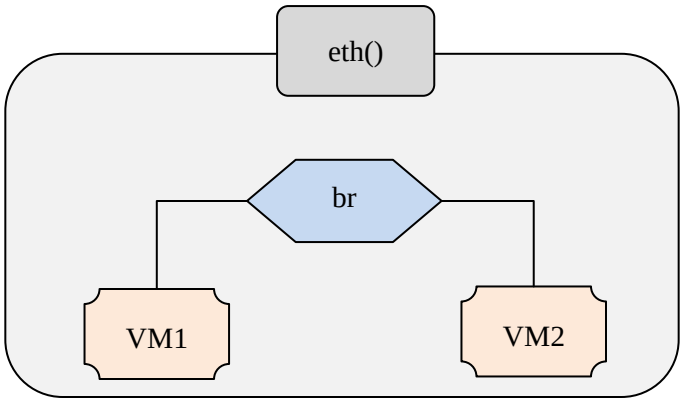
Figure 8 demonstrates that the attacking VM is successful in receiving the ICMP echo request and reply, which indicates that the two target VMs are able to communicate with one another. As was previously stated, ICMP is the protocol that is being used to carry out the ping command. The echo request and 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.



F**igure 8:** 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 9 presents this conceptual framework. A breakdown of the cloud services that could be compromised in this attack is shown in Table 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.



F**igure 9:** Nova-Network

Table 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.

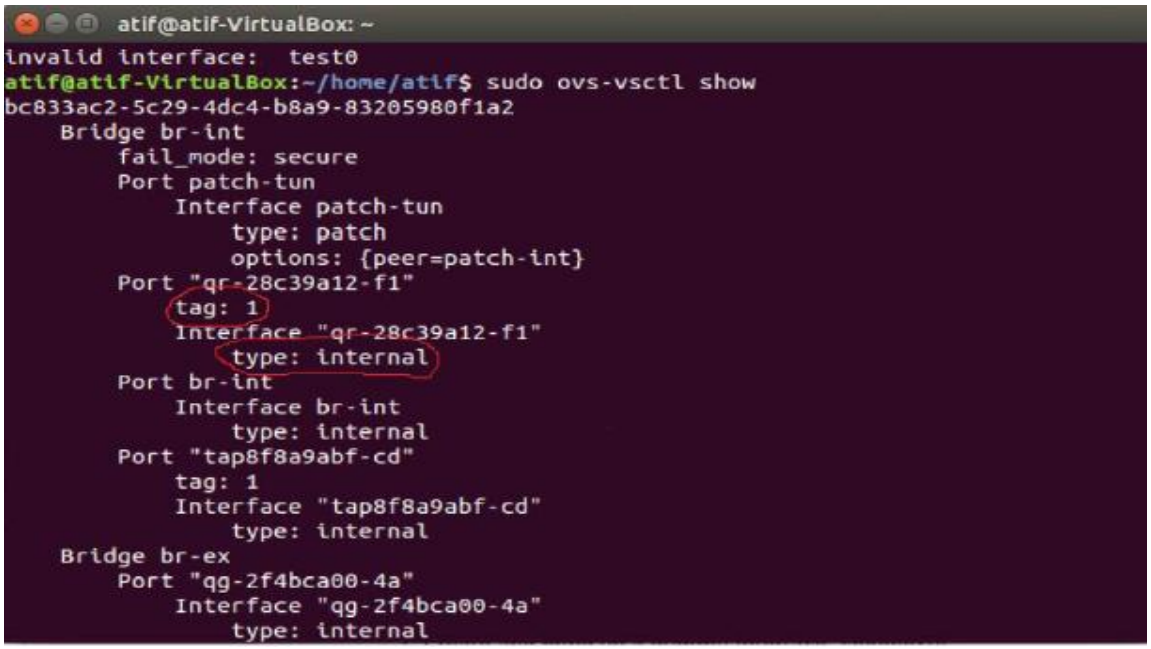
Avoiding Co-residency

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 practices 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, etc.

Solutions to Prevent Attacks

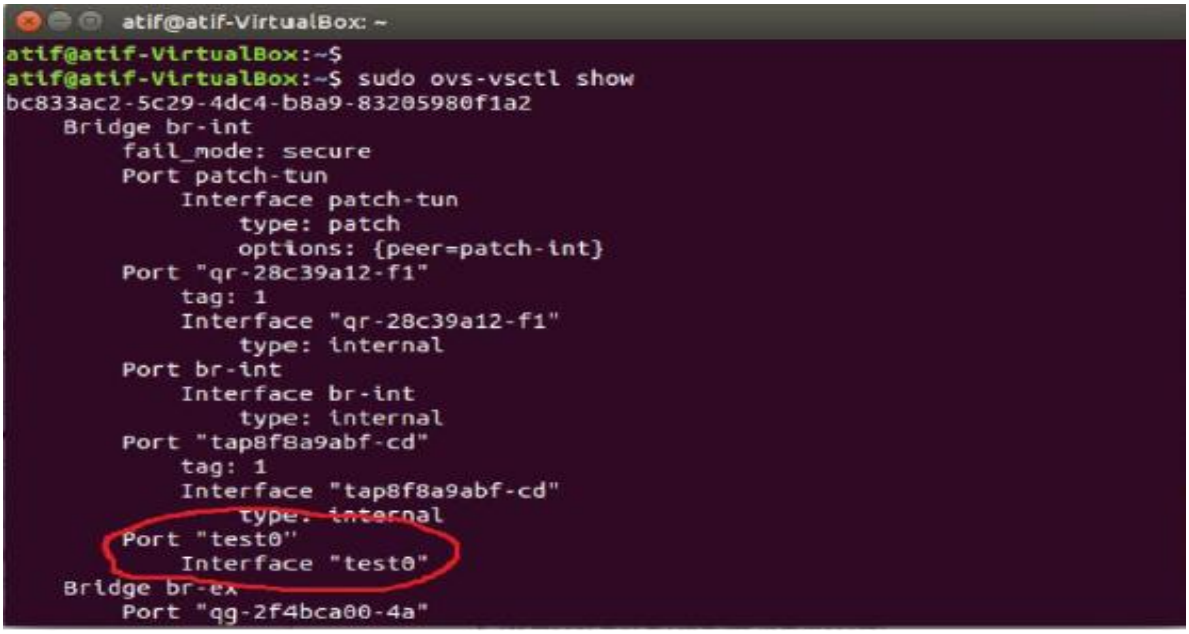
Several efforts are applicable and could be utilized to protect against cross-virtual-machine threats. Afoulk et al. [47] try to avoid conflict of interest among VMs using priority based scheduling, whereas Zhang et al. [48] propose using side-channels as a detector to identify illegal co-residency based 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 10, which exhibits the interconnection of several virtual Interfaces.



F**igure 10:** 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 11.



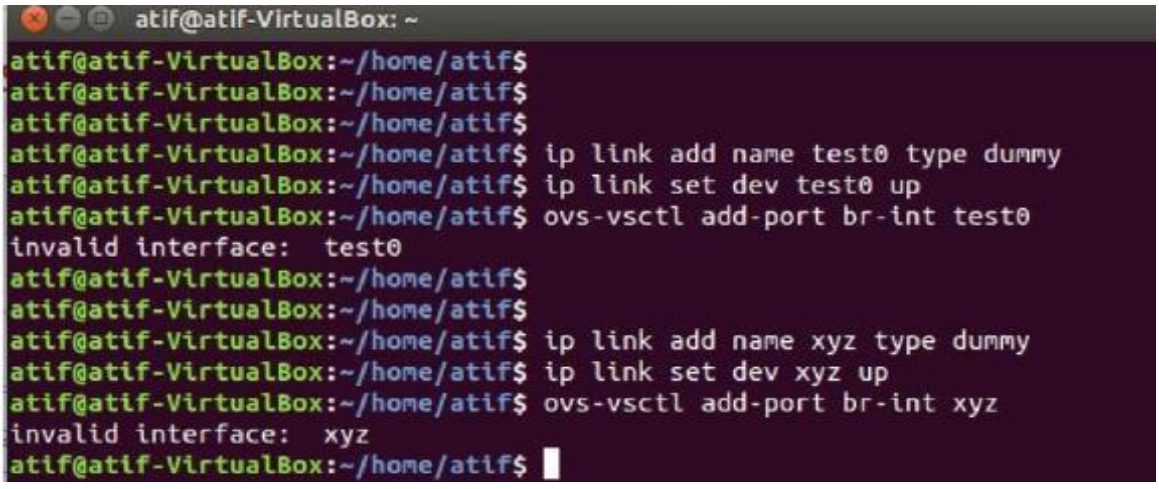
F**igure 11:** Attachment of Dummy Interface

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 vsctl method in the class /opt/stack/neutron/agent/impl vsctl.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).

The goal of this feature is to prevent direct connections between TAP interfaces and an OVS bridge utilizing 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 12.

6. Discussion

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. The results reveal how the disclosed attack manipulates cloud network architecture, rerouting co-located virtual machine traffic effectively. 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.



F**igure 12:** Blockage of Invalid Interface

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) 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 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.

7. Conclusion

In conclusion, this paper provides 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.

References

[1] P. M. Mell and T. Grance, "SP 800-145. The NIST Definition of Cloud Computing," *National Institute of Standards & Technology*, Gaithersburg, MD, USA, 2011.

[2] S. Ibrahim, B. He, and H. Jin, "Towards pay-as-you-consume cloud computing," *in 2011 IEEE International Conference on Services Computing, IEEE*, pp. 370–377, 2011.

[3] J. Schaper, "Cloud Services," *in 4th IEEE International Conference on Digital Ecosystems and Technologies*, pp. 91–91, 2010.

[4] D. T. Vojnak, B. S. Đorđević, V. V. Timčenko, and S. M. Štrbac, "Performance Comparison of the type-2 hypervisor VirtualBox and VMWare Workstation," *in 2019 27th Telecommunications Forum (TELFOR)*, pp. 1–4, 2019.

[5] C. Zhang, J. Bi, Y. Zhou, A. B. Dogar, and J. Wu, "Hyperv: A high performance hypervisor for virtualization of the programmable data plane," *in 2017 26th International Conference on Computer Communication and Networks (ICCCN), IEEE*, pp. 1–9, 2017

[6] S. Xi, J. Wilson, C. Lu, and C. Gill, "RT-Xen: Towards real-time hypervisor scheduling in Xen," *in Proceedings of the ninth ACM international conference on Embedded software*, pp. 39–48, 2011.

[7] A. Gulati, A. Holler, M. Ji, G. Shanmuganathan, C. Waldspurger, and X. Zhu, "Vmware distributed resource management: Design, implementation, and lessons learned," *VMware Technical Journal*, vol. 1, no. 1, pp. 45–64, 2012.

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

[9] A. Sleit, N. Misk, F. Badwan, and T. Khalil, "Cloud computing challenges with emphasis on Amazon EC2 and windows azure," *International Journal of Computer Networks & Communications*, vol. 5, no. 5, p. 35, 2013.

[10] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of EC2 cloud computing services for scientific computing," *in Cloud Computing: First International Conference, CloudComp 2009 Munich, Germany*, Springer, 2010, pp. 115–131, 2010.

[11] D. Talia and others, "Cloud Computing and Software Agents: Towards Cloud Intelligent Services.," in *WOA, Citeseer*, pp. 2–6, 2011.

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

[13] 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, IEEE*, pp. 490–494, 2010.

[14] B. Gulmezoglu, M. S. Inci, G. Irazoqui, T. Eisenbarth, and B. Sunar, "Cross-VM cache attacks on AES," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 211–222, 2016.

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

[16] R. Hund, T. Holz, and F. C. Freiling, "Return-oriented rootkits: Bypassing kernel code integrity protection mechanisms," in USENIX Security Symposium, pp. 383–398, 2009.

[17] Available at: https://www.wireshark.org/.

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

[19] H. -C. Li, P. -H. Liang, J. -M. Yang and S. -J. Chen, "Analysis on Cloud-Based Security Vulnerability Assessment," *IEEE 7th International Conference on E-Business Engineering (ICEBE)*, Shanghai, China, pp. 490-494, 2010.

[20] A. Seshadri, M. Luk, N. Qu and A. Perrig, "Secvisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity OSes," *SOSP '07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, ACM, vol. 41, pp. 335–350, 2007.

[21] V. Iozzo, "Ralf-philipp weinmann & vincenzo iozzo own the iphone at pwn2own," 2010.

[22] B. Ding, Y. Wu, Y. He, S. Tian, B. Guan and G. Wu, "Return-Oriented Programming Attack on the Xen Hypervisor," *2012 Seventh International Conference on Availability, Reliability and Security*, Prague, Czech Republic, pp. 479-484, 2012.

[23] M. Schwarz, M. Lipp, D. Moghimi, J. V. Bulck, J. Stecklina, T. Prescher, D. Gruss "Zombieload: Cross-privilege-boundary data sampling," *CCS '19: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 753–768, 2019.

[24] A. Moghimi, T. Eisenbarth, B. Sunar, "Memjam: A false dependency attack against constant-time crypto implementations," *International Journal of Parallel Programming*, vol. 47, no. 4, pp. 538–570, 2019.

[25] 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.

[26] 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.

[27] J. Rutkowska, "Subverting vistatm kernel for fun and profit," *Black Hat Briefings*, 2006.

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

[29] Y. Zhang, A. Juels, A. Oprea, and M. K. Reiter, "Homealone: Co-residency detection in the cloud via side-channel analysis," *in IEEE symposium on security and privacy*, pp. 313–328, IEEE, 2011.

[30] 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.

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

[32] S. M. Hashemi and M. R. M. Ardakani, "Taxonomy of the Security Aspects of Cloud Computing Systems-A Survey," Taxonomy of the Security Aspects of Cloud Computing Systems - A Survey," *International Journal of Applied Information Systems*, vol.4, no.1, pp. 21-28, 2012.

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

[34] 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.

[35] Available at: https://www.openstack.org/.

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

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

[38] Available at: https://qubes-os.org/.

[39] A. Marshall, M. Howard, G. Bugher, B. Harden, C. Kaufman, M. Rues, and V. Bertocci, "Security best practices for developing windows azure applications," *Microsoft Corp*, pp. 1-26, 2010.

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

[41] Available at: https://www.techrepublic.com/article/understanding-routing-tables/.

[42] A. Saeed et al., "A cross-virtual machine network channel attack via mirroring and tap impersonation," *in IEEE International Conference on Cloud Computing (CLOUD)*, IEEE, pp. 606–613, 2018.

[43] Available at: https://www.paessler.com/bandwidthmonitoring/.

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

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

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

[47] Z. Afoulki, A. Bousquet, and J. Rouzaud-Cornabas, "A security-aware scheduler for virtual machines on IaaS clouds," *Report-2011*, pp.1-12, 2011.

[48] 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.