Design and Analysis of NIDS Placements to Defend Internal Attacks on a Private Cloud

TANAPAT ANUSAS-AMORNKUL[1], WARAKORN SONNUI[1], CHATCHAI KHUNBOA[2,*]
[1]Department of Computer and Information Science,
King Mongkut's University of Technology North Bangkok,
1518 Pracharat 1 Road, Wongsawang, Bangsue, Bangkok 10800,
THAILAND

[2]Department of Computer Engineering, Faculty of Engineering,
Khon Kaen University,
Mittraphap Road, Muang, Khon Kaen 40002,
THAILAND

Abstract: - Cloud computing is widely used in government and business sectors. For the business sector, security is one of the most important topics to be considered to protect user and confidential data, especially in a cloud, which cannot manage the infrastructure physically. A private cloud is deployed to utilize only in an organization for all internal users. In addition, a Network Intrusion Detection System (NIDS) is normally used for detecting intruders from outside organizations. However, if an attacker is inside a private cloud, a typical NIDS placement cannot detect this type of attack. Therefore, the objective of this study is to design and analyze the NIDS locations to defend against internal attacks on a private cloud. A private cloud was deployed using OpenStack cloud with 3 physical servers, which were one admin/network node, and two compute nodes. Two types of attacks, DDoS and SSH brute force attacks, were implemented. Three NIDS placement scenarios inside the cloud were proposed and experimented with. The results indicated that NIDS placements on a network node, and compute nodes with a tap port, gave the best performance. In this scenario, NIDS detected all internal attacks, and Application VMs responded to requests up to 79.16% without performance reduction.

Key-Words: - Cloud computing, Network IDS, Internal attacks, Private cloud, NIDS architecture, Snort.

Received: November 30, 2024. Revised: April 16, 2025. Accepted: May 18, 2025. Published: July 8, 2025.

1 Introduction

Cloud computing is a paradigm for sharable resources by pooling many high-performance servers together with self-service provisioning on demand. By sharing physical resources, cloud computing can share CPU, memory, storage, and network resources for maximum performance and scalability. A set of software and hardware components can create a cloud infrastructure.

Many organizations and data centers widely deploy the OpenStack cloud [1], an open-source software that provides services for cloud infrastructure. OpenStack controls large pools of computing, storage, and networking resources. Examples of important components for OpenStack are 1) centralized management to manage all CPUs, memories, and storages; 2) network management software using Open vSwitch for managing network connections and topology in a cloud, network flow for controlling communication direction, and network zoning for creating subnetworks inside a cloud.

A Network Intrusion Detection System (NIDS) is normally located inside a network to detect intruders or attacks in a network by monitoring suspicious traffic. The NIDS examines all packets passing through it using signature-based or anomaly-based detection, alerting a network administrator if it detects an intruder. A widely used software for NIDS is Snort [2], which is open-source software.

An internal attack is a type of attack that targets inside a network. This attack is not easily detected since most network administrators trust internal computers and are not aware of this attack. The consequences of the attack can be large, such as data loss and service unavailability.

In this study, the NIDS placement scenarios were proposed to defend against internal attacks on a private cloud. The organization of the paper is as follows. Section 2 presents the related works

related to NIDS architecture. Next, a system design is introduced to view the cloud architecture. Section 4 presents the experiment setup for the study, which includes types of attacks, experimental scenarios, performance metrics, and NIDS placement scenarios. The next section shows the results from the experiments with discussions. Section 6 is the conclusion of the study with future work.

2 Related Works

The placements of Network Intrusion Detection System (NIDS) were proposed in a few research studies but with different focuses. [3] proposed Network Intrusion Detection Systems as a Service in the OpenStack cloud. In this work, there were four experiments, i.e., 1) baseline by installing NIDS in a compute node, 2) container by installing NIDS within a Docker container, 3) TaaS by installing NIDS within a VM node, and 4) NIDSaaS by deploying NIDS service on a network node. The results showed that the CPU and memory usage patterns for NIDSaaS were similar to those for the baseline but a lot better than those for TaaS.

[4] proposed a Host-based IDS (HIDS) framework for the private cloud to reduce computational and IDS resources by gathering traffic logs on Elastic Search stored in a centralized host to detect anomaly activities and alert a network administrator. [5] proposed the placements of IDS servers in 3 scenarios, which were 1) inside a cloud server, 2) separate from a cloud server, and 3) both inside a cloud server and separate from a cloud server. The first placement was not able to detect all attacks, while the other two placements were able to detect all attacks. [6] proposed installing a signature-based NIDS for OpenStack Private Cloud to defend against external attacks. Network Address Translation (NAT) was implemented before packets went out to/in from the Internet. An access switch was configured to use port mirroring, which sent all traffic to NIDS for detection. The results were a 100% detection rate, a 99.6% classification rate, a 99.9% precision, and a 0.07% false positive rate. [7] proposed a hierarchical IDS using Support Vector Machine (SVM) for Software-Defined Networking (SDN) networks in Cloud Data Centers. The classification of the IDS involved two levels: a flow-based IDS and a packet-based IDS. The flow-based IDS used SVM for classifying abnormal and normal packets (an anomaly detection engine). The packet-based IDS used Snort (signature-based detection) to detect abnormal packets. The based results from passing through 2 levels were a 5.12% false acceptance rate.

[8] proposed using deep learning with a hybrid NIDS. The deep learning techniques were Sparse Auto-Encoder (SAE) and Multi-layer Perceptron (MLP). SAE technique was used for classifying normal and abnormal packets, and MLP technique is used for classifying types of attacks. The combination of both techniques gave high accuracy at 99.9%. [9] proposed a Deep Recurrent Neural Network for Intrusion Detection in SDN based network. Three components were implemented. The first component was a flow collector, used for detecting packets and analyzing features. The next component was an anomaly detector, which used RNN to train a model to classify abnormal packets. An anomaly mitigator, the final component, controlled the data flow. The highest accuracy was 89% using GRU-RNN, compared to Vanilla RNN, SVM, and DNN. [10] proposed a native SDN Intrusion Detection using Machine Learning by implementing IDS on Native SDN in a control plane. Two important components were the assembly module and the adaptive ML module. The assembly module detected packets on networks and forwarded them to the adaptive ML module, which classified them as normal or abnormal. The best result was using Auto-encoder and Random Forest (AERF) with 90.19% accuracy, 88.78% precision, and 94.70% recall.

[11] proposed using machine learning approaches, which were Naïve Bayes, Sequential minimal optimization (SMO), C4.5 decision tree, and Random Forest, to improve Small and Medium Enterprises (SME) security. With the Information Gain (IG) as a feature selection method, the accuracy for all approaches was above 95%, and the training time was reduced significantly. [12] proposed using deep learning models, which were Generative Adversarial Networks (GAN), Convolutional Neural Networks with three layers (CNN-3L), and CNN with four layers (CNN-4L) for the intrusion detection classifications. The results indicated that the CNN-4L gave the best accuracy at 100%.

3 A system Design

This work configured a small private cloud using OpenStack Cloud with Open Virtual Switch (OVS), a fundamental component in SDN for managing internal networks inside a cloud using VXLAN and connecting to the Internet. VXLAN can manage subnetworks within or between

different servers in a cloud seamlessly and efficiently. In this study, the typical Snort IDS software as a network intrusion detection system was implemented. However, it is possible to implement Snort in various locations in the cloud. The focus of this work is to design a cloud architecture that can detect internal or insider attacks. Internal attacks, such as Denial of Service (DoS) and SSH password guessing attacks, have the potential to significantly reduce the cloud performance. These attacks were easy to be detected if the attacks were from the outside, not inside.

Normally, the placement of NIDS (Snort) in an organization is located behind a firewall, and this location cannot detect any internal attacks. For this work, a private cloud architecture was shown in Appendix in Figure 1 using 3 physical servers, built on OpenStack, an open-source cloud computing infrastructure software. It is important to note that the architecture is small when compared to the enterprise cloud, but it can be used to prove the NIDS placement concept.

The first server functioned as an Admin node and a Network node for controlling the overall cloud system and managing internal networks. The Server 1 specifications are shown in Table 1. The second and third servers functioned as compute nodes, which provided computational resources within the cloud. Both servers, Server 2 and 3, had the same specifications, also shown in Table 1.

Four components inside the Cloud were 1) Networks inside the Cloud, 2) Virtual servers, 3) Virtual clients, and 4) Snort software. Networks inside the cloud were separated into 2 subnets to simulate connections between zones, or different virtual networks. Two virtual servers, or application virtual machines (VMs), provided APIs for two different subnets. Eight virtual clients were attackers attacking the application VMs. Each compute node implemented four virtual clients within two virtual networks. Two virtual clients inside compute node 1 were in the same network as the two virtual clients inside compute node 2. The last component was Snort, widely known for Network Intrusion Detection System (NIDS) software. The Snort placement was under investigation to detect internal attacks. The Snort locations can be at 1) a network node, 2) compute nodes, and/or 3) application VMs.

Table 1. Server specifications

Specifications	Server 1	Servers 2, 3
Functions	Admin and network node	Compute nodes
CPU	Intel® Xeon® E5540 2.53GHz /16 Core	Intel® Xeon® CPU E5-2680 0 @ 2.70GHz /32 Cores
RAM	16 GiB	32 GiB
Storage	420 GB	1000 GB
Operating System	Rocky Linux 9	Rocky Linux 9
Virtualization software	OpenStack Zed	OpenStack Zed
Network management	open vSwitch	-
Internal network	vxlan	-

4 Experimental Setup

The experiments were conducted using the private cloud architecture in Figure 1 (Appendix) with two networks and two application VMs. This section described the types of attacks for this study, performance metrics, experimental scenarios, and NIDS placement scenarios as follows.

4.1 Types of Attack

In the experiments, two types of internal attack, i.e. Distributed Denial of Service (DDoS) and Secure Shell (SSH) password brute force attacks, were deployed together to attack two application VM servers within a cloud. These attacks can occur inside a cloud if a cloud operator does not aware of the internal attacks. The attack details are explained below.

1. A DDoS attack is an attack to deplete a victim's resources such that the victim cannot respond to any requests. In this study, eight VM computers (Attack 01-08) sent requests to Application VMs using Locust, an open-source load testing tool [13], for creating a DDoS attack. Locust was configured to generate traffic of 10,000 simultaneous users for 5 minutes.
2. An SSH Brute force attack is an attack to guess all possible SSH passwords continuously. In this study, eight VM computers (Attack 01-08) sent passwords to Application VMs using Hydra, a password-cracking tool, [14].

Note that any attacks could be implemented since the focus of this study was on the NIDS placement locations such that all attacks can be detected.

4.2 Performance Metrics

This study measures the performance of the experiments using 5 metrics as follows.

1. Attack detection whether the NIDS can detect the attacks or not at each placement.
2. CPU usage percentage, which collected CPU usage from the Network node and the compute nodes in 1-second time interval.
3. Memory usage, which collected memory usage from all physical nodes in 1-second time interval.
4. Percentage of successful requests at Application VMs, which is a ratio of successful requests over total requests.
5. Number of detected packets, which were total packets that can be detected at each NIDS placement.

4.3 Experimental Scenarios

To design the NIDS placements in the cloud architecture, two networks (Network 1 and Network 2) were deployed, and each network contained an Application VM, which was a target of the attacks. In addition, attackers were on 8 VMs. Attackers 1, 2, 5, and 6 VMs were on Compute node 1, and Attacker 3, 4, 7, and 8 VMs were on Compute node 2. However, Network 1 contained Attackers 1, 2, 3, 4, and Application VM1. Network 2 contained Attackers 5, 6, 7, 8, and Application VM2, as shown in Figure 1 (Appendix).

The connectivity inside the cloud utilized the VXLAN system from OpenStack Cloud to be controlled by a Network node through Switch 01-04 and Router 01-04, as shown in Figure 1 (Appendix). There are four attack scenarios in the experiment as follows.

1. The attackers were on the same network and the same host. The example attack flow is shown in Figure 2.
2. The attackers were on the same network but with different hosts. The example attack flow is shown in Figure 3.
3. The attackers were on different networks but with the same host. The example attack flow is shown in Figure 4.
4. The attackers were on different networks and different hosts. The example attack flow is shown in Figure 5.

Note that attackers targeted the Application VM depending on the attack scenarios.

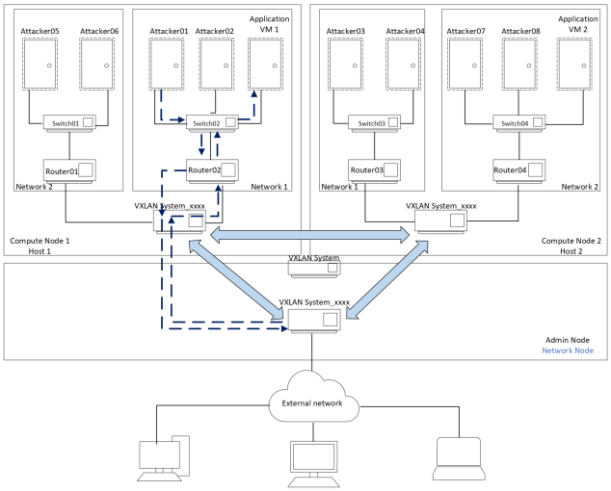

Fig. 2: The attackers were on the same network and the same host

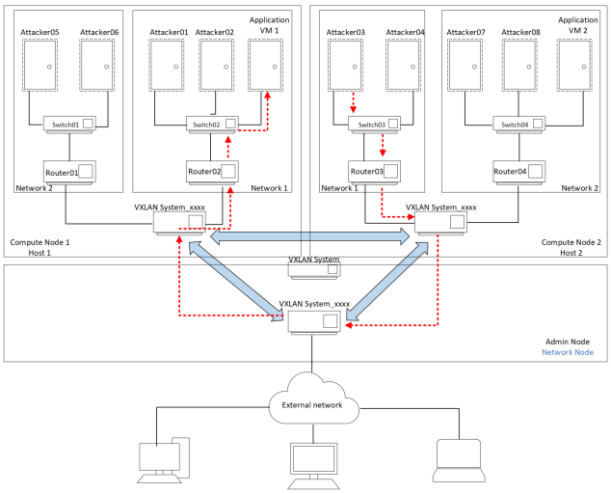

Fig. 3: The attackers were on the same network but different hosts

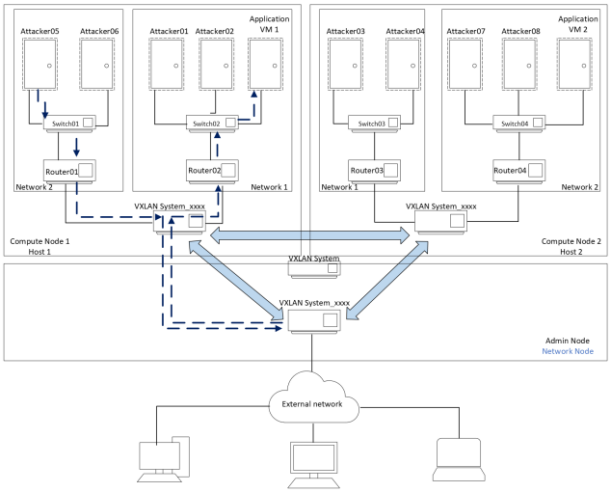

Fig. 4: The attackers were on different networks but with the same host

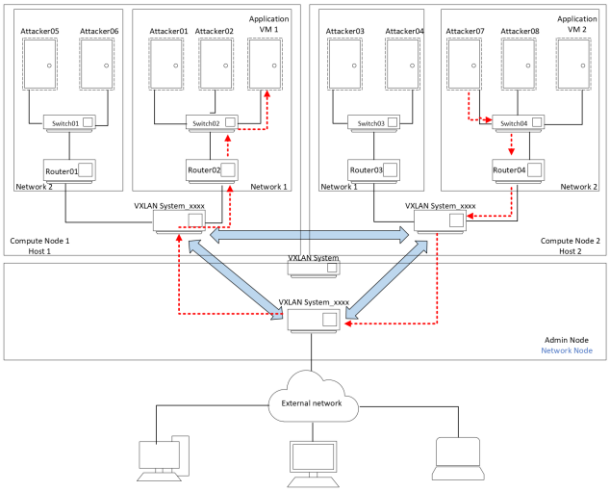

Fig. 5: The attackers were on different networks and different hosts

4.4 NIDS Placement Scenarios

For the NIDS in this study, Snort was implemented using standard community rules to detect anomalous behavior inside networks. There were three placement scenarios as follows.

1. Placement Scenario 1: On a Network node only
The Snort was deployed on a Network node only to minimize resource usage in the cloud, as shown in Figure 6. This scenario was a typical placement for NIDS to detect outsider attacks.

2. Placement Scenario 2: On a Network node, Compute nodes, and Application VMs
The Snort was deployed in all possible locations inside the Cloud to measure the performance as shown in Figure 7. This scenario was to place all possible NIDS locations in the cloud.

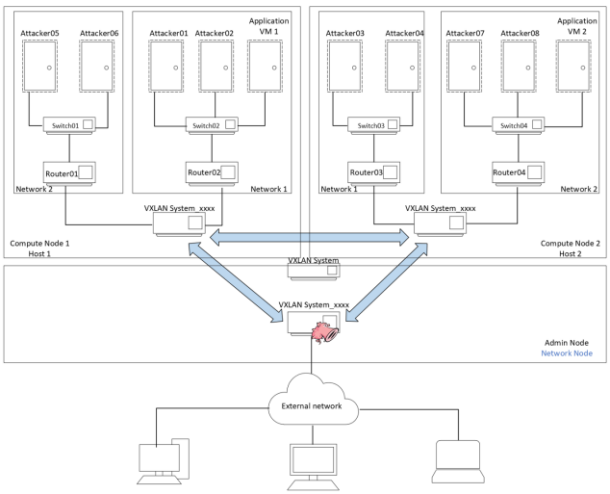

Fig. 6: Placement scenario 1

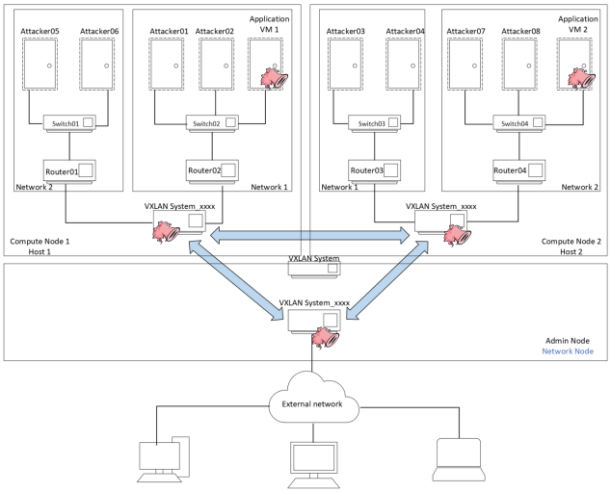

Fig. 7: Placement scenario 2

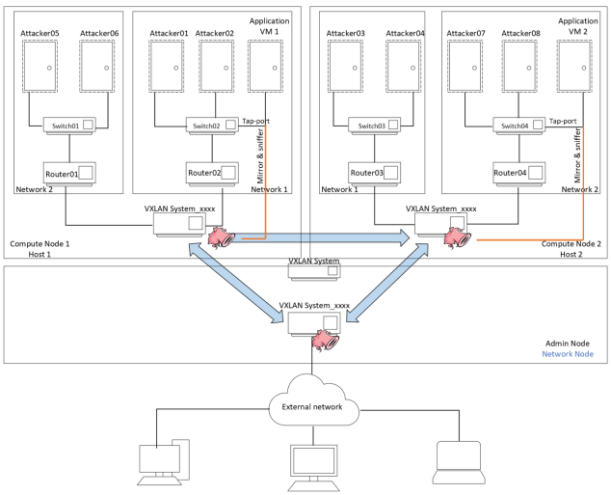

Fig. 8: Placement scenario 3

3. Placement Scenario 3: On a Network node, Compute nodes, with a tap port
The Snort was deployed on a Network node, Compute nodes, and a tap port between a switch and an Application VM. The tab port was used for mirror traffic from an Application VM to a Compute node in the same host. This can reduce the Application VMs' loads as shown in Figure 8.

5 Experimental Results

From the previous section, DDoS and SSH Brute force attacks were running at the same time. To analyze the effects of NIDS on CPU load and memory, the four experiments were conducted as follows.

1. No NIDS / No attack – There was no NIDS installed and no attack.
2. With NIDS / No attack – There was an NIDS installed and no attack.

3. No NIDS / Attacking – There was no NIDS installed, but the attack was ongoing.
4. With NIDS / Attacking – There was a NIDS installed, and the attack was on going.

5.1 Placement Scenario 1

In the placement scenario 1, NIDS was installed on a Network Node only, and Table 2 presented the attack detection at the Network Node, which did not detect any attacks if the attack occurred within the same network. Figure 9 shows the CPU usage percentage when the attacks were ongoing, the CPU usage was higher than normal at the Compute nodes and Application VMs. The CPU usage was increased when the attacks were ongoing, and the NIDS analyzed all packets.

Table 2. Attack detection at a Network Node in Scenario 1

		Attackers			
		Compute Node 1		Compute Node 2	
		Net.1	Net.2	Net.1	Net.2
Victims	App. VM 1 [CN1, Net.1]	Not detected	Detected	Not detected	Detected
	App. VM 2 [CN2, Net.2]	Detected	Not detected	Detected	Not detected

Note: Net. = Network
NN = Network Node
CN = Compute Node
App.VM = Application VM

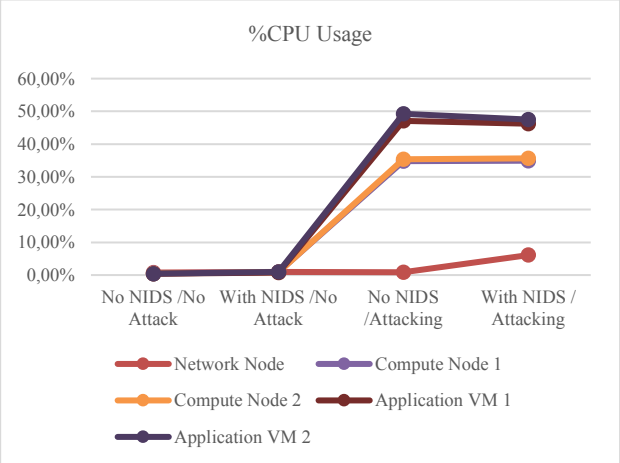

Fig. 9: CPU usage percentage for Scenario 1

Figure 10 shows the memory usage in Gibibytes. With NIDS installed, the memory usage was increased significantly by more than 1 Gibibytes when an attack was ongoing but was changed a bit when running only NIDS. For other nodes, the memory usage did not show any changes. Table 3 shows the percentage of successful requests at Application VMs. Since there was no NIDS installed on Application VMs, the traffic could flow to the VMs and the requests were responded to about 84% of the time while attacking. Table 4 shows the number of detected packets and NIDS was able to detect only 14.22% of all attack packets at the Network node.

Table 3. Percentage of successful requests at Application VMs in Scenario 1

	Request			Response Time (ms)
	Total requests	%Passed	%Failed	
Without NIDS	2,427,535	83.95	16.05	2,216.49

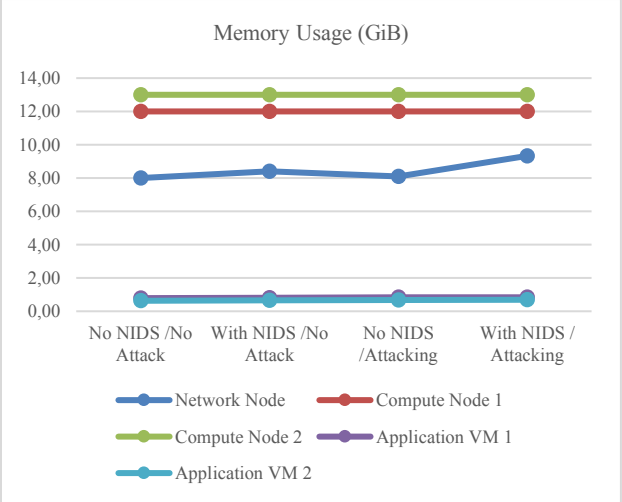

Fig. 10: Memory usage for Scenario 1

Table 4. Detected packets for Scenario 1

	Network Node
Total Packets	6.23M
Attack Detection	0.89M
% Detection	14.22

Note: M = 1,000,000 packets

5.2 Placement Scenario 2

In placement scenario 2, NIDS were installed on a Network node, two Compute nodes, and two Application VMs. Table 5 shows the results that the Application VMs were able to detect the attacks, but the Network and Compute nodes did not detect them in some cases. Figure 11 shows the CPU usage percentage, which increased from about 5% to 10% when the attacks were ongoing, and the NIDS in all nodes analyzed all packets. Figure 12 showed the memory usage for Scenario 2, and the

results were similar to those for Scenario 1 for the Network node, but for the Compute nodes, the memory usage was increased while NIDS was running under attacks.

Table 5. Attack detection in Scenario 2

		Attackers			
		Compute Node 1		Compute Node 2	
		Net.1	Net.2	Net.1	Net.2
Victims	App. VM 1 [CN1, Net.1]	Detected at App VM1	Detected at NN, CN1, App VM1	Detected at CN1, CN2, App VM1	Detected at NN, CN1, App VM1
	App. VM 2 [CN2, Net.2]	Detected at NN, CN2, App VM2	Detected at CN1, CN2, App VM2	Detected at NN, CN2, App VM2	Detected at App VM2

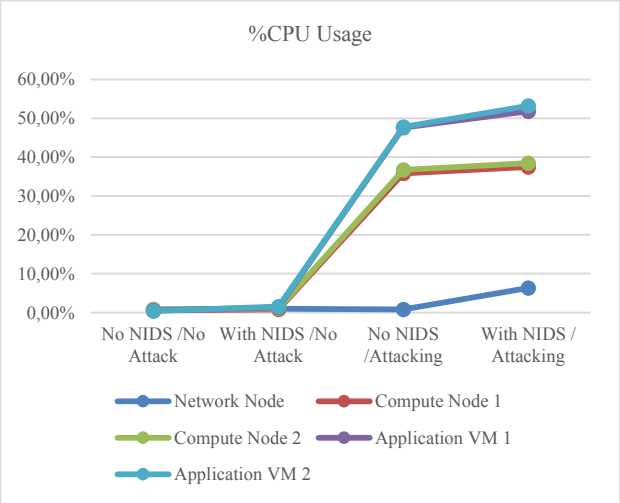

Fig. 11: CPU usage percentage for Scenario 2

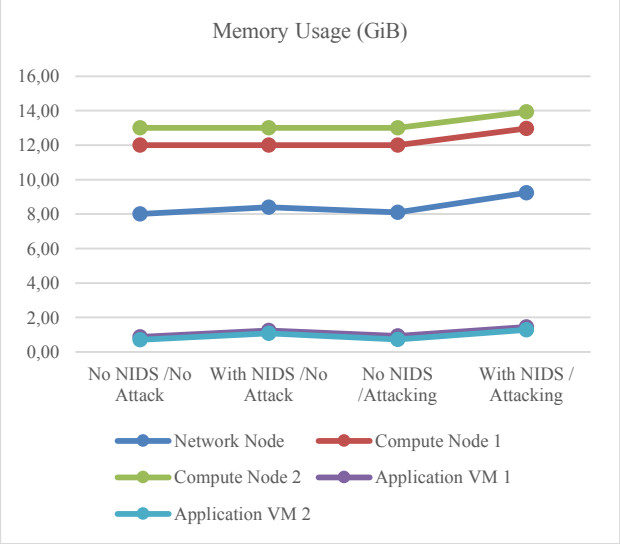

Fig. 12: Memory usage for Scenario 2

Table 6 presents the percentage of successful requests at Application VMs in Scenario 2. The interesting results showed that NIDS in Application VMs reduced the ability to respond to requests from 80.74% to 48.43% when NIDS inspected all packets during the attacks. Table 7 shows the detected packets in all nodes. The detection percentages from the network node and compute nodes ranged from 13.75% to 14.31%, but the attacks were still detected in several cases at the nodes.

Table 6. Percentage of successful requests at Application VMs in Scenario 2

	Request			Response Time (ms)
	Total requests	%Passed	%Failed	
Without NIDS	2,416,489	80.74	19.26	2,354.88
With NIDS	2,419,740	48.43	51.57	2,484.11

Table 7. Detected packets for Scenario 2

	NN	CN 1	CN 2	App.VM1	App.VM 2
Total Packets	6.65M	7.73M	6.44M	7.23M	6.753M
Attack Detection	0.95M	1.06M	0.91M	0.99M	0.95M
% Detection	14.31	13.75	14.08	13.75	14.04

Note: M = 1,000,000 packets

5.3 Placement Scenario 3

In the placement scenario 3, NIDSs were installed on a Network node, two compute nodes with a tap port that mirrored traffic from an Application VM to a compute node in the same host. Table 8 presented the attack detection, and this scenario was able to detect attacks in all cases within Network and Compute nodes. The CPU usage percentage and the memory usage for Scenario 3 are shown in Figure 13 and Figure 14, respectively. The results did not show any significant changes from the previous scenarios. The CPU usage was not increased drastically, while the Application VMs were under attack.

Table 9 shows the percentage of successful requests at Application VMs, and the results showed significant improvement that the Application VMs were able to respond to requests at 79.16%, which was close to that in the placement scenario 1.

Table 10 shows the detected packets and results were not different from that in Scenario 2 ranging from 13.38% to 14.13% but this scenario can detect all cases.

Table 8. Attack detection in Scenario 3

		Attackers			
		Compute Node 1		Compute Node 2	
		Net.1	Net.2	Net.1	Net.2
Victims	App. VM 1 [CN1, Net.1]	Detected at CN1	Detected at NN	Detected at CN1, CN2	Detected at NN
	App. VM 2 [CN2, Net.2]	Detected at NN	Detected at CN1, CN2	Detected at NN	Detected at CN2

Table 9. Percentage of successful requests at Application VMs in Scenario 3

	Request			Response Time (ms)
	Total requests	%Passed	%Failed	
Without NIDS	2,364,450	79.16	20.84	2,292.59

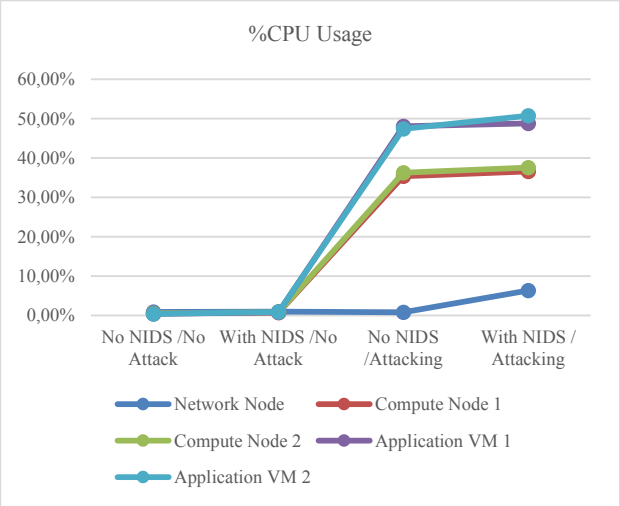

Fig. 13: CPU usage percentage for Scenario 3

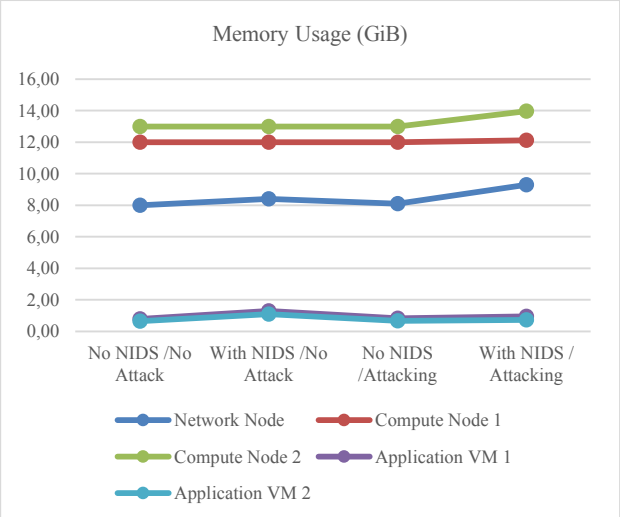

Fig. 14: Memory usage for Scenario 3

Table 10. Detected packets for Scenario 3

	Network Node	Compute Node 1	Compute Node 2
Total Packets	6.75M	2.05M	2.01M
Attack Detection	0.95M	0.28M	0.26M
% Detection	14.13	13.73	13.38

Note: M = 1,000,000 packets

5.4 Discussion

From all placement scenarios, the attacks were not detected if attacked packets did not pass through NIDS. Depending on the performance metrics, the result discussions were explained as follows.

5.4.1 Attack Detection

From all scenarios, when the NIDSs were installed on a Network node, and Compute nodes, the placement scenarios 1, and 2 were unable to detect all attacks, as shown in Table 11. For scenario 1, the attacks were unable to be detected within the same network. However, for scenario 2, the attacks were unable to be detected within the same network and the same host. For scenario 3, the attacks were able to be detected in all cases because of mirroring traffics to Compute nodes from Application VMs.

Table 11. Attack detection at NN, CN1, and CN2

		Attackers			
		Compute Node 1		Compute Node 2	
		Net.1	Net.2	Net.1	Net.2
Victims (Scenario 1)	App. VM 1 [CN1, Net.1]	**Not detected**	Detected at NN	**Not detected**	Detected at NN
	App. VM 2 [CN2, Net.2]	Detected at NN	**Not detected**	Detected at NN	**Not detected**
Victims (Scenario 2)	App. VM 1 [CN1, Net.1]	**Not detected**	Detected at NN, CN1	Detected at CN1, CN2	Detected at NN, CN1
	App. VM 2 [CN2, Net.2]	Detected at NN, CN2	Detected at CN1, CN2	Detected at NN, CN2	**Not detected**
Victims (Scenario 3)	App. VM 1 [CN1, Net.1]	Detected at CN1	Detected at NN	Detected at CN1, CN2	Detected at NN
	App. VM 2 [CN2, Net.2]	Detected at NN	Detected at CN1, CN2	Detected at NN	Detected at CN2

Note that the Application VMs can detect the attacks when Snort (NIDS) software is installed under the attack conditions. Therefore, the

Application VMs were not considered in this comparison.

5.4.2 CPU Usage Percentage

Table 12 shows the CPU usage percentage for the case of node(s) installed NIDS and under attack. This was the case that used the highest CPU resource. From the results, all 3 scenarios did not show any significant differences in CPU usage. The results with bold texts were the CPU usage from the node(s) that installed Snort, which was running during the attacks. However, the CPU usage for Application VMs in Scenario 2 was increased about 3% to 5% because the application VMs were under attack and Snort was inspecting all incoming packets when compared to Scenario 1 and 3.

5.4.3 Memory Usage

The memory usage for all nodes were shown in Table 13. The results with bold texts were the memory usage from the node(s) that installed Snort. The memory was consumed more in the case of using NIDS but there was no significant increase in the memory usage in all scenarios. However, in Scenario 2, the memory usage in Application VMs increased significantly, about 0.5 – 0.6 GiB compared to other scenarios.

Table 12. CPU usage percentage for node(s) installed NIDS, and under attack

Scenario	NN	CN 1	CN 2	App.VM1	App.VM2
Scenario 1	**6.10 %**	34.93 %	35.65 %	46.20 %	47.52 %
Scenario 2	**6.37 %**	**37.47 %**	**38.47 %**	**51.80 %**	**53.23 %**
Scenario 3	**6.37 %**	**36.57 %**	**37.57 %**	48.84 %	50.76 %

Table 13. Memory usage for node(s) installed NIDS, and under attacked

Scenario	NN	CN 1	CN 2	App.VM1	App.VM2
Scenario 1	**9.33 GIB**	12.00 GIB	13.00 GIB	0.84 GIB	0.69 GIB
Scenario 2	**9.24 GIB**	**12.97 GIB**	**13.93 GIB**	**1.44 GIB**	**1.28 GIB**
Scenario 3	**9.25 GiB**	**12.12 GiB**	**13.96 GiB**	0.94 GiB	0.72 GiB

5.4.4 Percentage of Successful Requests at Application VMs

When considering the performance of the Application VMs under the attack conditions, Application VMs were able to successfully respond to requests more than 79% of the time if Snort was not installed in the Application VMs. However, the performance dropped significantly when Snort was installed, as shown in Table 14. This is because the servers had to not only process the requests but also detect the attacks at the same time such that the Application VMs dropped more request packets.

5.4.5 Number of Detected Packets

The number of detected packets at the Network node is shown in Table 15 for all scenarios. Since Snort was installed on the Network node for all scenarios, the results did not indicate any significant differences in all scenarios ranging from 14.13% to 14.31%.

Table 14. Percentage of successful requests at Application VMs (under attack)

	Request			Response Time (ms)
	Total requests	%Passed	%Failed	
Scenario 1	2,427,535	83.95	16.05	2,216.49
Scenario 2	2,419,740	**48.43**	**51.57**	2,484.11
Scenario 3	2,364,450	79.16	20.84	2,292.59

Table 15. Number of detected packets at the Network node

	Scenario 1	Scenario 2	Scenario 3
Total Packets	6.23M	6.65M	6.75M
Attack Detection	0.89M	0.95M	0.95M
% Detection	14.22	14.31	14.13

6 Conclusion

Cloud computing is widely utilized in modern organizations for its flexibility, reliability, and efficiency. However, the cloud can be under attack, especially in a private cloud that does not have good protections. An internal attack is the most unexpected attack that an organization may not be aware of. Therefore, this study proposed to design the NIDS placements for a private cloud to defend against internal attacks, which were DDoS and SSH brute force attacks. The initial cloud architecture was introduced using OpenStack cloud. There were three NIDS placement scenarios, and the results showed that the NIDS placements should be at a Network node, and Compute nodes with a tap port to mirror traffic from Application VMs to Compute nodes, as shown in Figure 15 (Appendix). From the architecture, the Application VMs can respond to requests for 79.16%, and the attacks can be detected in all cases.

The future research is to customize the firewall rules such that attacked packets are not detected more than once at different locations, and this can increase the detection performance such that the CPU and memory usag could be decreased.

Acknowledgement:
This research was funded by King Mongkut's University of Technology North Bangkok, Contract no. KMUTNB-68-BASIC-08.

References:

[1] The Most Widely Deployed Open Source Cloud Software in the World. OpenStack Cloud, [Online]. https://www.openstack.org/ (Accessed Date: December 10, 2024).

[2] Snort, [Online]. https://www.snort.org/ (Accessed Date: December 10, 2024).

[3] Chen Xu, Ruipeng Zhang, Mengjun Xie, Li Yang, "Network Intrusion Detection System as a Service in OpenStack Cloud," *2020 International Conference on Computing, Networking and Communications (ICNC)*, Big Island, HI, USA, 2020, pp. 450-455, doi: 10.1109/ICNC47757.2020.9049480.

[4] Zhijian Wang, Yanqin Zhu, "A centralized HIDS framework for private cloud," *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, Kanazawa, Japan, 2017, pp. 115-120, doi: 10.1109/SNPD.2017.8022709.

[5] A. A. Aryachandra, Y. Fazmah Arif, S. Novian Anggis, "Intrusion Detection System (IDS) server placement analysis in cloud computing," *2016 4th International Conference on Information and Communication Technology (ICoICT)*, Bandung, Indonesia, 2016, pp. 1-5, doi: 10.1109/ICoICT.2016.7571954.

[6] Berkah I. Santoso, M. Rien Suryatama Idrus, Irwan Prasetya Gunawan, "Designing Network Intrusion and Detection System using signature-based method for protecting OpenStack private cloud," *2016 6th International Annual Engineering Seminar (InAES)*, Yogyakarta, Indonesia, 2016, pp. 61-66, doi: 10.1109/INAES.2016.7821908.

[7] Quentin Schueller, Kashinath Basu, Muhammad Younas, Mohit Patel, Frank Ball, "A Hierarchical Intrusion Detection System using Support Vector Machine for SDN Network in Cloud Data Center," *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, Sydney, NSW, Australia, 2018, pp. 1-6, doi: 10.1109/ATNAC.2018.8615255.

[8] Kentaro Takeshita, Michiko Harayama, "Improvement of Hybrid NIDS Using Deep Learning for Practical Use," *2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW)*, Matsue, Japan, 2021, pp. 385-391, doi: 10.1109/CANDARW53999.2021.00071.

[9] Tuan A Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, Mounir Ghogho, "Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks," *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, Montreal, QC, Canada, 2018, pp. 202-206, doi: 10.1109/NETSOFT.2018.8460090.

[10] Mohd Mat Isa, Lotfi Mhamdi, "Native SDN Intrusion Detection using Machine Learning," *2020 IEEE Eighth International Conference on Communications and Networking (ComNet)*, Hammamet, Tunisia, 2020, pp. 1-7, doi: 10.1109/ComNet47917.2020.9306093.

[11] Nevila Baci, Kreshnik Vukatana, Marius Baci, "Machine Learning Approach for Intrusion Detection Systems as a Cyber Security Strategy for Small and Medium Enterprises," *WSEAS Transactions on Business and Economics*, vol. 19, pp. 474-480, 2022, https://doi.org/10.37394/23207.2022.19.43.

[12] Nabeel Refat Al-Milli, Yazan Alaya Al-Khassawneh, "Intrusion Detection System using CNNs and GANs," *WSEAS Transactions on Computer Research*, vol. 12, pp. 281-290, 2024, https://doi.org/10.37394/232018.2024.12.27.

[13] Locust, An open source load testing toolc https://locust.io/ (Accessed Date: December 21, 2024).

[14] Hydra, [Online]. https://www.kali.org/tools/hydra/ (Accessed Date: December 21, 2024).

APPENDIX

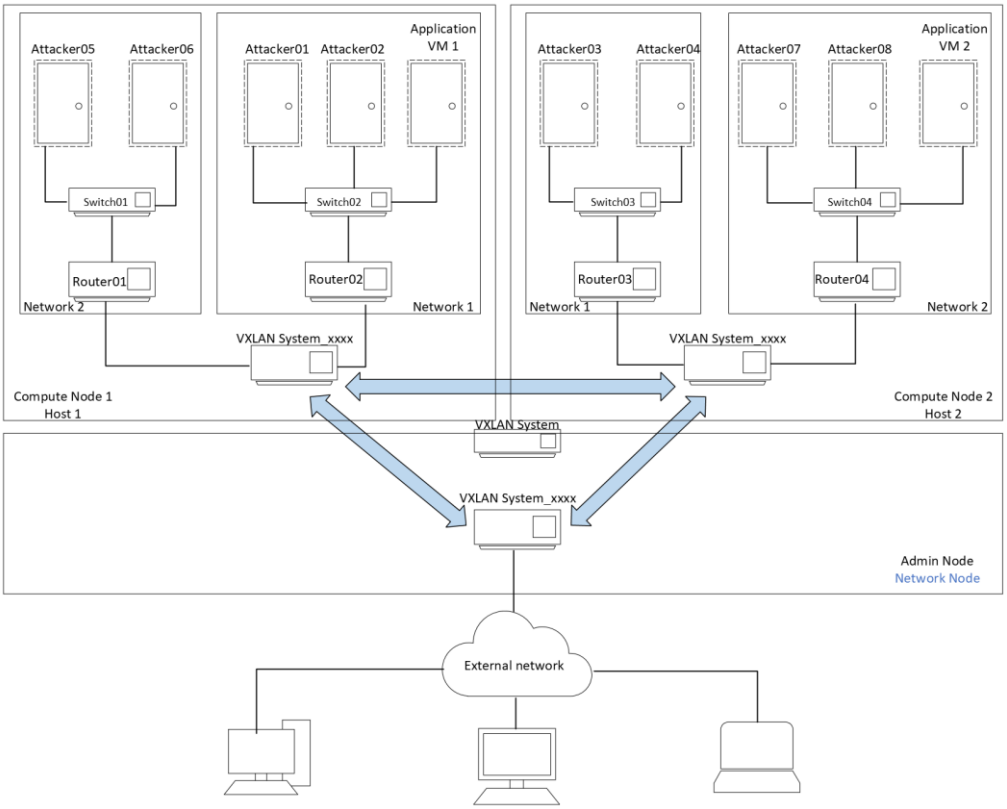

Fig. 1: A private cloud infrastructure

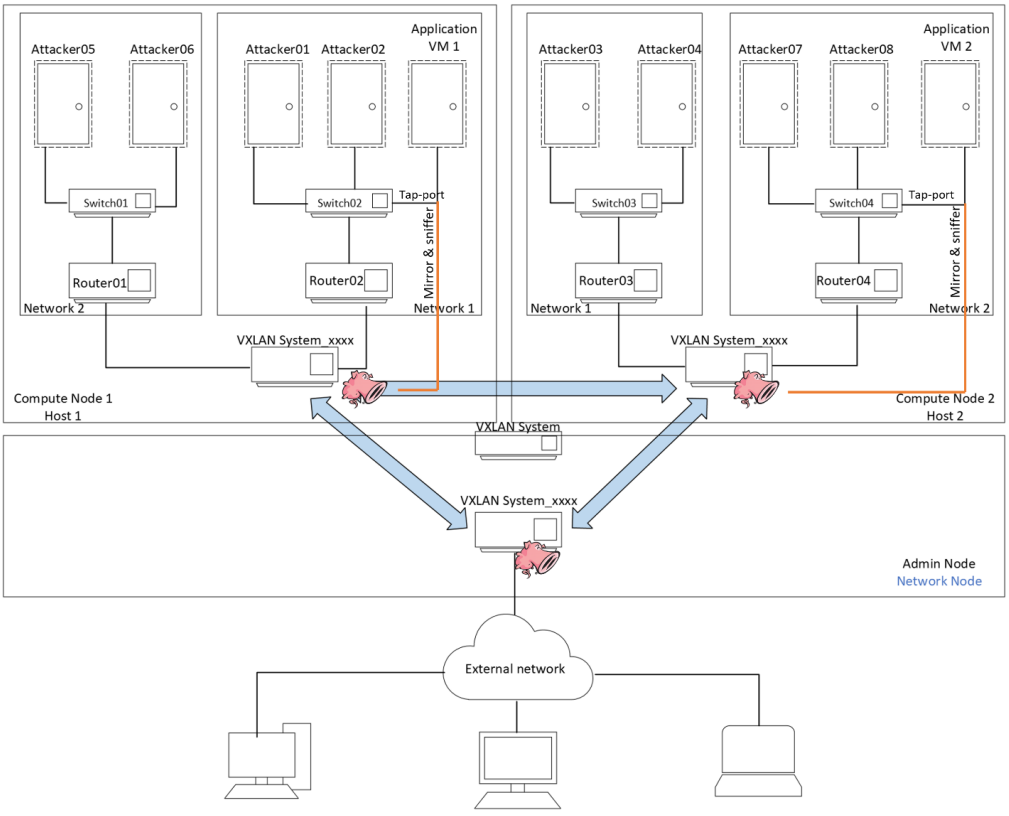

Fig. 15: The best proposed NIDS architecture

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

- Tanapat Anusas-amornkul gave concepts, supervised, analyzed, and wrote this research.
- Warakorn Sonnui implemented and conducted the experiments for this research.
- Chatchai Khunboa supervised this research.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

This research was funded by King Mongkut's University of Technology North Bangkok, Contract no. KMUTNB-68-BASIC-08.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0
https://creativecommons.org/licenses/by/4.0/deed.en_US