Bulletin of Electrical Engineering and Informatics
Vol. 14, No. 1, February 2025, pp. 377~387
ISSN: 2302-9285, DOI: 10.11591/eei.v14i1.8103

Detection and prevention of Man-in-The-Middle attack in cloud computing using Openstack

Najat Tissir[1], Noureddine Aboutabit[1], Said El Kafhali[2]
[1]Process Engineering, Computer Science, and Mathematics Laboratory, National School of Applied Sciences Sultan Moulay Slimane University, Khouribga, Morocco
[2]Computer, Networks, Mobility, and Modeling Laboratory: IR2M, Faculty of Sciences and Techniques, Hassan First University of Settat, Settat, Morocco

Article Info

Article history:

Received Dec 27, 2023
Revised Jul 15, 2024
Accepted Aug 7, 2024

Keywords:

Address resolution protocol
Cloud computing
Internet control message protocol
Man-in-the-Middle attack
Openstack
Packet detection ratio
Prevent and detect

ABSTRACT

This paper proposes a new technique designed to prevent and detect address resolution protocol (ARP) spoofing attacks in general, and specifically Man-in-the-Middle (MitM) attacks, within the context of cloud computing. The solution focuses on establishing appropriate flow filtering rules based on parameters such as 'time feature' and internet control message protocol '(ICMP) protocol'. The tests were conducted using the Openstack platform. One of the key benefits of this proposed approach is the improved performance in effectively detecting a significant number of malicious packets. We implemented this solution on the Openstack platform and conducted evaluations to demonstrate its efficacy. The results confirm that our method achieves superior performance in detecting MitM attacks, with a packet detection ratio (PDR) of 60.4%. Moving forward, this work will contribute to protecting cloud environments from a large number of MitM attacks.

Corresponding Author:

Najat Tissir
Process Engineering, Computer Science, and Mathematics Laboratory
National School of Applied Sciences Sultan Moulay Slimane University
Khouribga, Morocco
Email: tissir.najat@gmail.com

1. INTRODUCTION

Cloud computing has become an integral aspect of every business in the modern day. Both the public and private sectors require a well-functioning network in order to operate effectively. With the rising use of cloud computing, numerous challenges have arisen, and security is of paramount importance [1]. According to the report on the state of the cloud in 2022, respondents identified security as the greatest obstacle for both businesses (85%) and SMBs (82%), as well as for advanced cloud users [2]. In cloud systems, prior research has predominantly addressed issues such as denial of service (DoS), distributed denial of service (DDoS), and anomaly detection in service level agreements [3]-[5]. This study shifts its focus to the security vulnerabilities present in cloud computing, specifically examining the risks associated with address resolution protocol (ARP) spoofing and its related indirect attacks, including the Man-in-the-Middle (MitM) attack.

The ARP is an essential protocol within the field of computer networking, operating specifically at the data link layer (Layer 2) of the OSI model. The primary objective of this system is to simplify the process of mapping or resolving an IP address to a media access control (MAC) address within a local network. The ARP mechanism comprises five distinct processes: address resolution request, broadcast ARP request, target device response, updating ARP cache, and communication. The ARP request packet includes the specified IP address to identify the corresponding MAC address. The sender delivers the request to the broadcast MAC

address, which is subsequently distributed to all devices within the network. The target device provides an ARP reply that contains the MAC address. The ARP cache is updated with the new MAC address, optimizing packet routing. Once updated, the device can transmit data packets directly to the intended device, eliminating the need for ARP resolution. In a network, direct communication between the source host and the destination host is only possible if the source host knows about the destination host's MAC address. Nevertheless, ARP is missing a set of security measures to ensure data integrity and authenticity. ARP cache tables will be updated by hosts regardless of whether an ARP packet is legitimate. Even worse, ARP serves as the basis for network connections and could lead to more serious consequences, such as Mac flooding [6], host impersonation, MitM attacks [7], [8], DoS attacks [5], [9], session hijacking [10], and cloning attacks [11]. Consequently, ensuring ARP security in the cloud environment is crucial.

MitM actually involves manipulating the ARP cache in order to intercept and alter data being sent back and forth between two communication parties. In order to mitigate the risks posed by ARP attacks, network administrators frequently employ security measures such as the monitoring of ARP activity and the implementation of algorithms designed to detect and prevent ARP spoofing. Furthermore, the implementation of network segmentation and the utilization of technologies such as virtual local area networks (VLANs) can effectively separate possible points of vulnerability.

Studies have suggested diverse approaches for preventing, detecting, and mitigating attacks associated with the ARP protocol. These methods have been explored across various contexts such as IoT networks, wired networks, Bluetooth low energy (BLE) networks, wireless networks, cloud computing, and software-defined networking (SDN) networks [8], [12]-[21]. Table 1 presents an overview of the main relevant approaches employed in the detection and prevention of ARP spoofing and MitM attacks. The table also outlines the specific characteristics exploited by these approaches, along with the advantages and disadvantages associated with each approach.

Researchers have investigated the detection of attacks within suspicious packets by employing neural networks, as documented in studies [13], [14]. The proposed methodology encompasses the incorporation of payload bytes embedding, as well as the examination of TCP, UDP, and ARP packets, in conjunction with statistical data analysis. The methods employed in this study resulted in a notably high degree of precision. However, it is important to acknowledge that the experiment was carried out with a restricted dataset. One of the most straightforward approaches among the ways that have been analyzed is the utilization of static ARP entries [15]. In the absence of DHCP utilization, the manipulation of IP entries is not feasible. This defense exhibits considerable potential; however, its viability in bigger networks appears limited. In addition, detection tools such as scapy [16] and XArp [17] possess effective detection methods, although they do not offer comprehensive defensive capabilities. The internet control message protocol (ICMP) is employed in various contexts, including the detection of ARP spoofing attacks. Researchers in [18] have utilized ICMP ping packets to determine the characteristics of the host. The advantage of their approach lies in its ability to offer a cost-effective solution through the utilization of open source technologies, while also insuring the preservation of the unchanged state of the ARP protocol. Nevertheless, the authors failed to offer a proficient method for rating.

However, Chkirbene *et al.* [19] provides a framework based on intrusion detection systems. The framework proposed comprises three main stages. Initially, there is a learning phase where the machine learning algorithm is trained and a model is built. Following this, the decision history is stored using the EICD scheme, which records past decisions for each network node in separate databases. Finally, in the combined decision phase, the ultimate classification decision is made based on the stored information. Furthermore, the issue of ARP cache poisoning attack was addressed by Prabadevi *et al.* [20] in their study. Three strategies have been proposed for big data center networks. They provide superior performance in detecting malicious packets, effectively serving 89% of cases. Moreover, Rangisetti *et al.* [12] have developed a solution involving host_certification and floodlight modules to prevent ARP spoofing. The floodlight modules consist of a DHCP server, a link discovery module, and forwarding modules. These modules serve the purpose of obtaining IP information for different hosts during the host certification and authentication process, assisting the host_certification module in authenticating hosts within the network, and facilitating the installation of wildcard forwarding flow rules for legitimate flows. They argue that their proposal is designed to mitigate the risks associated with ARP message spoofing, MITM attacks, and session hijacking. In another connected research, Sun *et al.* [8] studied detecting and mitigating ARP attacks within an SDN-based cloud environment. Their methodology uses DHCP for obtaining reliable IP MAC mappings as well as real-time packet processing. This approach is designed to receive packets transmitted by hosts, process them accordingly and subsequently establish flow entries on switches or discard them. The process involves monitoring statistical data associated with packet transmission on the ports of edge switches. In a related study Kang *et al.* [21] conducted research within the Openstack environment, aiming to address ARP spoofing issues through the enhancement of keystone authentication. The proposal entails the process of maintaining ARP tables, followed by the handling of ARP reply messages using a component referred to as

the 'comparison handler'. However, this work presents some limitations: The proposal focuses on a specific scenario involving the location of the attacker host, OpenStack nodes, and victim host. This limited scope may not fully represent diverse real-world scenarios. Additionally, it assumes that hosts are set up with windows operating systems, which may not be applicable to all network environments and infrastructures.

Table 1. Related works comparison

Studies	Detection/mitigation technique	Method	Pros and cons
Sun *et al.* [8]	SDN	The task involves real-time packet processing, which includes receiving packets from hosts, managing these packets, and subsequently installing flow entries on switches or discarding them. Following this, statistical data on packet activity across edge switch ports is monitored	It makes use of DHCP to acquire trustworthy IPMAC mappings, ensuring the precision of ARP attack detection
Rangisetti *et al.* [12]	A DARP spoofing approach with a Host_Certification and floodlight modules	The authors proposed a D-ARPSpoof module using SDN with a host_certification module and floodlight modules	The D-ARPSpoof plays a significant role in preventing MITM, session hijacking, VLAN-ID spoofing, and ARP message spoofing attacks
Husain *et al.* [13]	Neural networks and ARIMA approach	Several neural networks were trained to take TCP, UDP and ARP packets as input to detect ARP Spoofing attacks	The accuracy rate of this neural network was great (more than 90%)
Lahmadi *et al.* [14]	Neural network	They use a reconstruction technique to identify suspicious network data batches	The detection accuracy was high(~0:99) and false positive rate was low (~0:03) (small dataset)
Rohatgi and Goyal [15]	Static ARP	IP and MAC addresses	It becomes exhausting in large networks
Majumdar *et al.* [16]	Scapy python library	a. Attack generating module b. Detecting module c. Preventing module Using static entries for preventing ARP attacks	A reliable ARP table enhances the dependability of the proposal within a cloud computing environment. However, maintaining both the comparison handler and the ARP table can lead to some partial loss of resources
Rupal *et al.* [17]	Dynamic IP configuration	System has three modules: a. DHCP server b. Radius server and MySQL database for authentication c. Detecting and preventing ARP poisoning	The author observed that while XArp provides a detection method, it does not have a preventative strategy
Arote and Arya [18]	Modified ICMP	The server keeps track of the data and uses ICMP ping packets to determine the host's identity. The central server is chosen using a voting mechanism	Backward compatibility ensures that the remaining systems are unaffected if the primary server fails
Chkirbene *et al.* [19]	Intrusion detection and classification technique	a. Creating the learning model b. Storing the decision history c. Combined decision phase	This method is more effective at detecting attacks and can raise classification accuracy from 66% to 90%
Prabadevi *et al.* [20]	CLCC, TCBA, and extended TCBA	The CLCC technique carries out three operations: updating the fake list table, crosslayer consistency checking, and alert message generation	The method showed 77% of detection ratio individually
Kang *et al.* [21]	Reliable ARP table in Keystone authentication service of OpenStack	a. Creating and maintaining ARP tables b. ARP reply message handling	One of the disadvantages is the partial resource loss required to maintain comparison handler and the table

Upon reviewing the methods and tools discussed, we identified several limitations. These include challenges in large and dynamic network environments, making the proposed methods impractical, more difficult, and time-consuming. There are also limitations related to attack prevention capabilities and the feasibility of implementation in large networks, which pose scalability and effectiveness challenges in complex network infrastructures.

Hence, this paper focuses on building a cloud computing environment using OpenStack. Additionally, we propose an algorithm designed to prevent and detect MitM attacks. Our algorithm utilizes the ICMP protocol and concentrates on establishing suitable flow filtering rules based on various parameters, primarily the calculation of the time interval between successive packets originating from the same Source and destination IP. The calculated time difference is then compared to a predefined threshold value. Our objective is to safeguard the system against a high volume of malicious packets and enhance performance while increasing the packet detection ratio (PDR). The following is a summary of our contributions:

– We present a comprehensive overview on various techniques used to facilitate detecting and preventing ARP spoofing attacks across diverse environments, as documented in existing literature.

- We propose an algorithm for protecting cloud computing networks against ARP spoofing attacks, with a specific focus on MitM attacks. Our strategy involves the utilization of ICMP packets.
- We introduce an implementation on the Openstack platform and conduct an evaluation to demonstrate the efficacy of our technique in enhancing performance by effectively detecting a significant number of malicious packets.

The following parts of this paper are organized as follows: section 2 outlines the approach employed and presents the proposed algorithm. The findings and evaluation of our experiment are presented in the section 3. Section 4 serves as the concluding section of the paper.

2. METHOD

This paper presents an algorithm aimed at detecting and preventing MitM attacks within a cloud computing environment using OpenStack. The experiments cover various scenarios to determine optimal outcomes. The proposed detection technique involves several essential steps. Initially, network sniffing is employed to capture and log IP-MAC addresses, timestamps, and relevant data of instances within the OpenStack cloud. This information is then utilized to construct static IP-MAC tables. The technique comprises two primary components: one for creating and managing ARP tables and the other for implementing attack prevention measures. Specifically, the collected IP and MAC address data is utilized to build a reliable ARP table via the ping command between instances and packet capture using Wireshark. Subsequently, the reply message is compared with the reliable ARP table for verification. In the first stage, any mismatched address marks the package as altered, indicating potential spoofing. The second stage involves comparing time interval calculations between successive ICMP packets, adding an additional layer of protection with a two-stage verification process.

2.1. Environment setup

In order to establish the necessary environment for the experiment, the utilization of the following tools and libraries is required:

- A server Dell EMC 540 for all-in-one Openstack cloud installation.
- Openstack platform: this experiment aims to gather IP and MAC addresses of the instances created within the OpenStack environment. The OpenStack framework is comprised of two main components, namely the controller and compute nodes. These nodes have the capability to be dynamically adjusted in size, either increased or decreased, in alignment with the scale of the cloud infrastructure. The Nova Scheduler, located on the controller node, is responsible for selecting a suitable host for the deployment of virtual instances. The host selection process is contingent upon the availability of the accessible compute nodes. Following the host selection process, the Nova Compute service starts the instantiation of instances according on the memory and volume size criteria supplied by the users. After the instantiation procedure, Nova-Network begins with the construction of both internal and external networks [22], [23]. In the present scenario, the Dell server accommodates both the computing and controller nodes.
- Kali Linux: originating from Debian, Kali Linux is an operating system distribution designed for use in digital forensics and penetration testing [24].
- Nmap: it is a powerful tool for scanning networks, we will use it here to find out the IP addresses allocated to the devices on the local network.
- Wireshark: this tool facilitates real-time recording of network data. The data has the capability for filtration based on IP addresses, protocols, and ports [25].
- Ettercap: it is a crucial tool to use when launching an ARP spoofing attack, manipulating data that has been intercepted, or attacking SSL or SSH connections.

In this paper, we consider the scenario where the attacker is a host within the same network as the OpenStack nodes. The victim, in this context, is attempting to communicate with either an external host or an internal instance, depending on the scenario chosen. For evaluation purposes, we have opted to use Ettercap as the attacking tool, a well-known tool designed for ARP Spoofing. Kali Linux is installed on the attacker's host computer, while the instances created for testing are configured with either CentOS 7 or Ubuntu Desktop operating systems. Table 2 provides the specifications of the hardware used in the experiment.

2.2. Attack generation module

The studies were conducted through real network testing to address MitM attack inside a cloud environment. The attack was executed through a series of four sequential steps. Initially, the 'router' address was defined using the command: *$ip route.*

Table 2. Hardware specification

Nature of machines	Operating systems	Hardware details	Purpose
Server Dell EMC 540	CentOS-7-x86_64	Xeon Quad Core 3,1 Ghz, RAM 16 GB, Disques durs: 3 * 1 TB, Contrôleur Raid H330,	all-in-one OpenStack Cloud
Lenovo T480s	Windows 10 pro 64 bits	Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz, 8,00 GB RAM, 500 GB HDD	Laptop1
HP 290 G2	Windows 10 pro 64 bits	I5-8500 @ 3.00 GHz, 4 GB RAM, 500 GB	Host1
Openstack instance	Kali Linux-2021.4	4 VCPU, 4 GB RAM, 60 GB	Instance1: Attacker
Openstack instance	Centos-7	1 VCPU, 2 GB RAM, 20 GB	Various instances: victims
Openstack instance	Ubuntu-17.04- desktop	1 VCPU, 2 GB RAM, 20 GB	Various instances: victims

After that, we employed the Ettercap graphical tool for the purpose of conducting the attack. It is essential to identify the targets, which refer to the instances of victims of a MitM attack. These instances may consist of physical hosts, routers, or instances that are operational within an OpenStack environment (as seen in Figure 1).

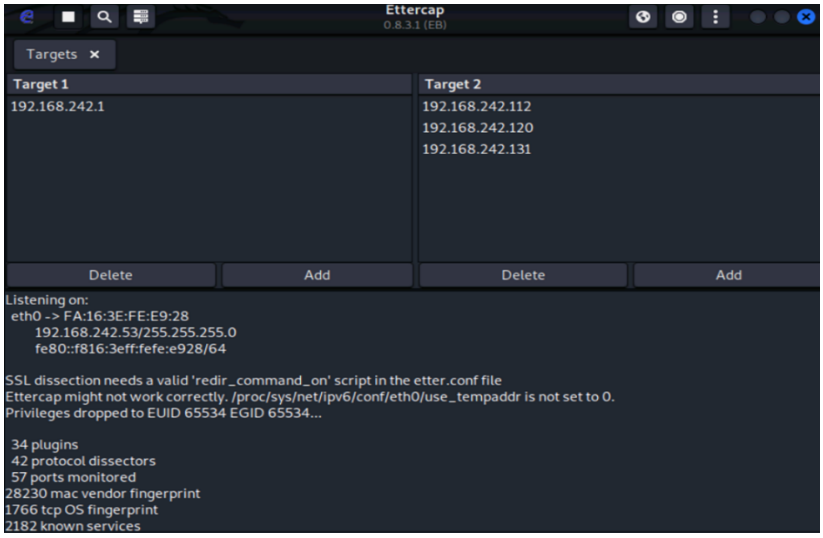


Figure 1. MitM generation at the selected targets

After initiating the attack in the designated instance named "attacker", we proceed to establish a ping between hosts. Subsequently, we utilize the Wireshark protocol analyzer to capture the network traffic for each ICMP connection, both under normal conditions and during the attack phase, as seen in the provided illustration (Figure 2). The capture of around 2408 packets is observed throughout each scanning session.

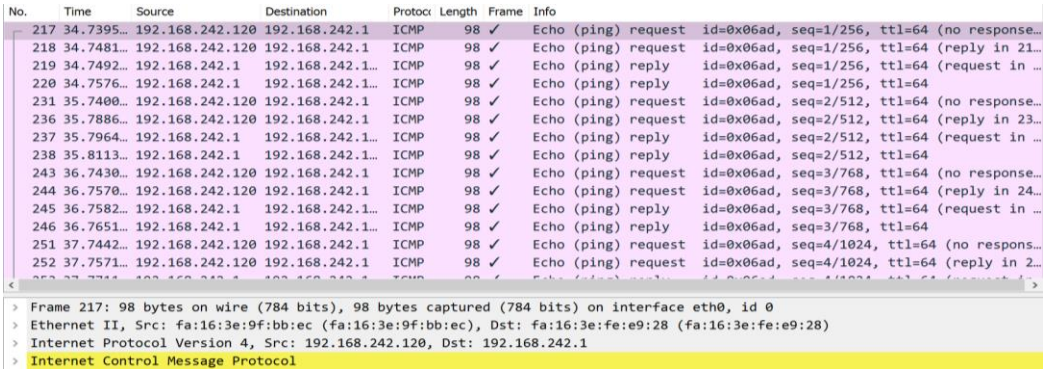


Figure 2. ICMP packet sniffing with wireshark

After initiating an attack, the process unfolds as depicted in Figure 3. Host1, intending to communicate with Victim1 within the OpenStack environment, first checks for Victim1's MAC address. Similarly, Victim1 needs Host1's address for communication, which is stored in its ARP table. When Host1

doesn't find the corresponding address, it broadcasts an ARP request message to its subnet. Upon receiving this request, Victim1 sends a unicast ARP reply message back to Host1, containing its MAC address. Simultaneously, the attacker intercepts these messages and sends modified replies to both Host1 and Victim1, substituting their MAC addresses with its own. Consequently, Host1 now perceives the attacker as Victim1, and Victim1 perceives the attacker as Host1. This deception leads to both hosts communicating with the attacker, believing they are communicating with each other, thereby exposing their information to the attacker.

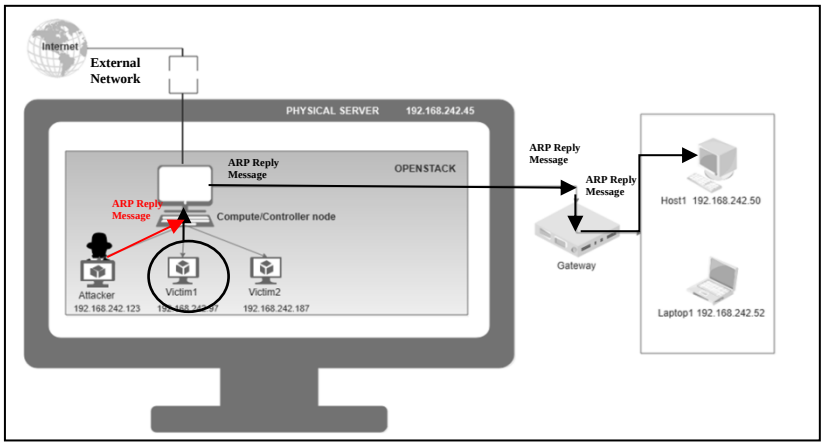


Figure 3. Spoofing attack scenario

Following that, we proceeded to capture ICMP traffic under normal and attack conditions. Figure 4(a) showcases the ICMP traffic in its normal state before the attacker's spoofing activities on the targeted system. The illustration provides a comprehensive view of the packet's information, encompassing details such as the packet number, emission time, source and destination addresses, employed protocol, packet length, and other relevant packet attributes. Upon the attack's initiation, our analysis focuses on identifying anomalous packets, particularly by scrutinizing the IP-MAC addresses and the time intervals between consecutive ICMP packets.

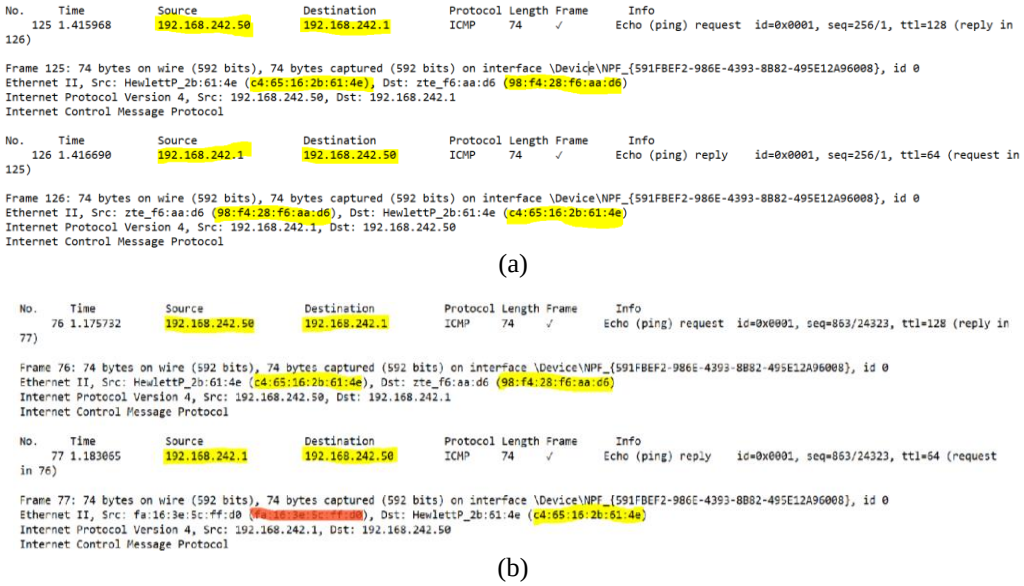
In Figure 4(b), the IP address of the Router is represented by 192.168.242.1, whereas the IP address of Victim1 is represented by 192.168.242.50. Additionally, the picture provides a description of the MAC addresses associated with each IP address. Figure 4(b) depicts the ICMP packet capture subsequent to the occurrence of the attack. The MAC address of Victim1 (c4:65:16:2b:61:4e) has been altered to the MAC address of the Attack Host (fa:16:3e:5c:ff:d0) as a result of Spoofing.

2.3. Attack detection/prevention module

The act of ARP spoofing can lead to many network attacks that pose significant risks. Therefore, it is imperative to prioritize the detection of such activities in order to effectively address any security concerns and prevent hosts from becoming vulnerable to these attacks. Hence, prior to implementing a preventive approach, it is advisable to consider employing a detecting system as the optimal choice.

After capturing network traffic, collecting log files, and establishing IP-MAC entries from the ARP table of each device manually (for non-DHCP environments) or automatically (for DHCP environments), we proceed to read the sniffed files in CSV format. These files consist of seven columns: number of packets, time, source IP, destination IP, protocol, length, ethernet source, and destination MAC. Subsequently, we examine each device's authorization by comparing the destination IP address in the log files with the IP addresses in the built table, and then cross-referencing that address with its associated MAC address to figure out whether the IP-MAC combinations are legitimate. Two cases emerge from this examination: first, if the examined IP-MAC pair is consistent, no attack is detected; second, if the examined IP-MAC pair does not match, it suggests evidence of a MitM attack between the source IP and destination IP.

To filter only ICMP packets belonging to a specific address (for example the router), the following filter is applied in wireshark protocol analyzer: ip.address==192.168.242.1 and we can add also the icmp.type ==0 or 8 to select the request and reply packets. In this paper, we propose the utilization of time as a feature for a MitM attack detection and prevention. In order to enhance preventive measures, an additional step (referred to as step 3-4) is incorporated. This step involves the calculation of the time interval between two consecutive packets originating from the same source_IP and dest_IP. This calculated time difference is subsequently compared to a specified threshold value.



(a)

(b)

Figure 4. Packet capture: (a) in normal condition and (b) in attack condition

Our study has been based on the intercommunication among ten machines situated in various environments, including instances in the Openstack cloud, physical servers, laptops, and personal computers. In the conducted experiment, the ICMP protocol was utilized for the purpose of calculating the threshold. The threshold is determined as the mean of the values of the time interval that have been calculated. Based on our experiment, we determined that the appropriate threshold value is equal to 0.003 seconds. We considered that the value of 'Treply-Trequest' may vary when all the hosts are internal OpenStack instances, as well as all the measurements taken during the experiment. Here, 'Trequest' refers to the time of an echo ping request packet, while 'Treply' is the time of an echo ping reply packet.

2.4. Our proposed detection/prevention algorithm

By utilizing the time interval and comparing it to our predefined threshold of 0.003 seconds, our approach aims to decrease the system's exposure to a MitM attack. This stage often reveals a significant number of anomalous packets. Subsequently, our program enhances the detection process by verifying each device's authorization and comparing the IP-MAC pairs. Upon detection, the program generates a warning message stating "There is a MitM attack between the source and destination IP of the infected host." The steps of our algorithm are shown in Figures 5 and described as Algorithm 1. Table 3 provides a description of the variables employed in our approach.

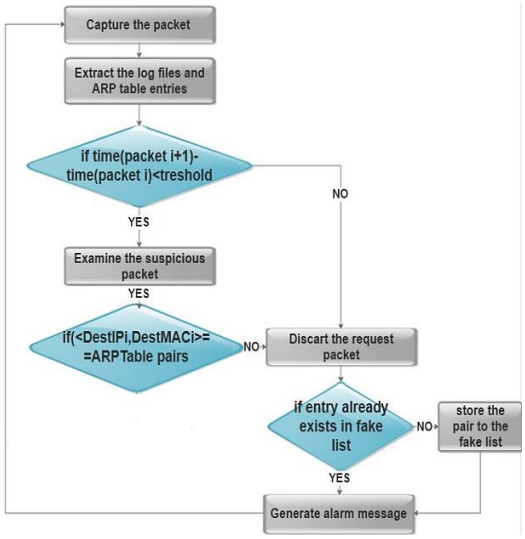


Figure 5. Our proposed algorithm for MitM detection

Algorithm 1: MitM prevention and detection

```
Sniff the network and extract the log files and ARPtab entries
Establish IP-MAC entries from ARP table of every device
for  i from 0 to table.size()
   if (table.Time[i+1] - table.Time[i]) < treshold
      for k from 0 to ARPtab.size()
         if table[i].DestIP == ARPtab[k].ipAddress     then
            if table[i].DestMAC ==ARPtab[k].ethernetAddress then
                  Update the ARP cache every 10 minutes
            else
                  Discart the request packet
                  if the pair <DestIP,DestMAC> already exists in fake list
                        Generate alarm message
                  else
                        Add the pair <DestIP,DestMAC> to the fake list
                  End if
            End if
         End if
      End for
   else
        Discart the request packet
   End if
End for
```

Table 3. Our algorithm's nomenclature

Variables	Description
ARPtab	The file that contains the pairs of the IP-MAC address of the instances in the network
table	The snifed file that contains 7 columns: no. of the packet, time, SourceIP, DestIP, protocol, length, SourceMAC, and DestMAC
DestIP	The destination IP address of the packet in the snifed file
SourceIP	The source IP address of the packet in the snifed file
DestMAC	The destination MAC address of the packet in the snifed file
SourceMAC	The source MAC address of the packet in the snifed file
ipAddress	IP address of a host in ARPtab
ethernetAddress	MAC address of a host in ARPtab
Time	Time of on packet in the snifed file
Treshold	A predetermined threshold is derived by taking the average of the time between two successive packets and dividing it by the length of the packet

3. EXPERIMENTAL RESULTS AND DISCUSSION

The above graphic, labeled as Figure 6, presents data on the duration of a machine's response to an ICMP request both before and after a MitM attack. The calculation process entails measuring the time interval between two successive packets, specifically the "echo ping request" and "echo ping reply" packets. Our experiment encompassed machines with diverse characteristics and operational setups. For instance, Host1 operates as a network station external to the cloud, while instance 1 and the attacker are both internal instances within the OpenStack framework. The variations in results are influenced by factors such as machine placement, technical specifications, and network affiliation. We found that the time interval (Treply-Trequest) before a MitM attack is consistently shorter than the time interval after the attack, as shown in Figure 6. This observation is logical, as the presence of an intermediary connecting the source and the destination would naturally lead to increased packet processing time.

Our findings indicate that the attack scenarios are influenced by various factors such as the position of the attacker and victims, the network infrastructure they are connected to, hardware configurations, and other related aspects. Attack scenarios can involve attackers from physical hosts, OpenStack instances, external networks, or internal networks. We assume that A=an Openstack instance; internal network; B=an internal physical host; internal network; C=an Openstack instance; external network; and D=an internal physical host. We will utilize the fourth scenario presented in Table 4. These scenarios seek to provide outcomes that are ideal and extremely accurate. We can manipulate certain parameters to come up with a suitable threshold value that may be used in any situation where the attack occurs.

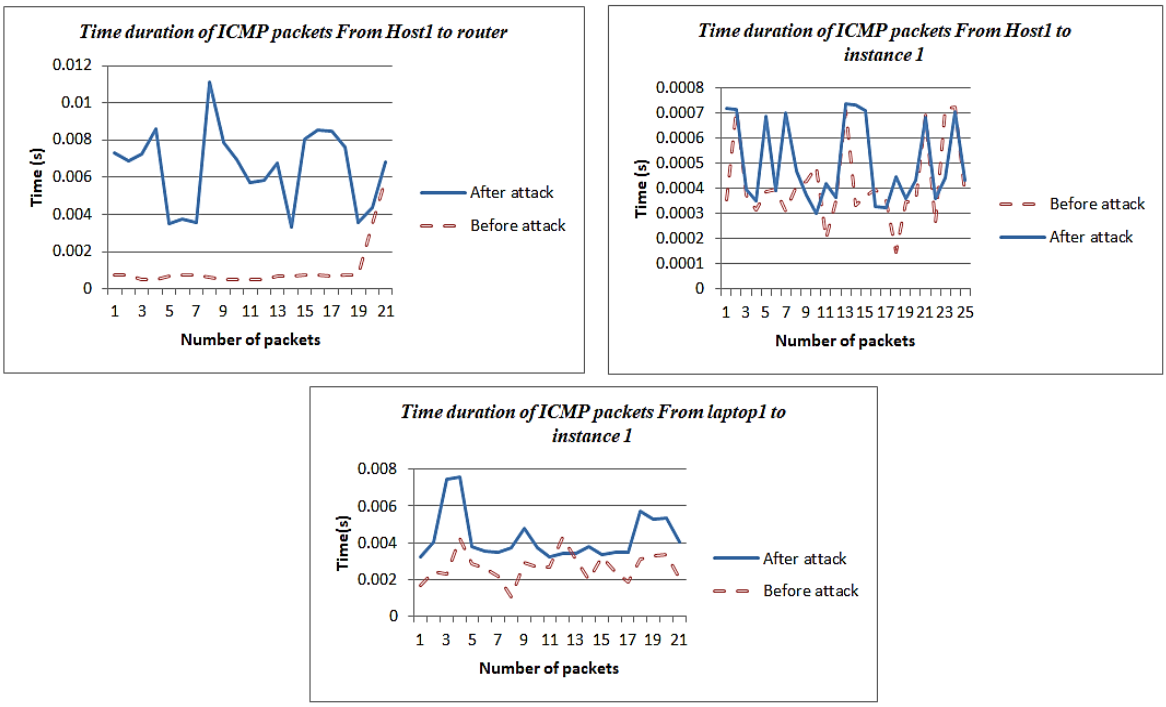


Figure 6. ICMP Packet time comparison before and after the attack

Table 4. The adopted scenarios

	Attacker	Victims
Scenario 1	C	C
Scenario 2	D	C
Scenario 3	D	A
Scenario 4	A	A

The PDR is a metric used to evaluate the performance of intrusion detection and prevention systems (IDPS) in network security. It determines the percentage of malicious packets correctly detected by the IDPS among all malicious packets present in the network. The formula used to calculate the PDR is as follows:

$$PDR = \frac{Number\ of\ abnormal\ packets\ detected}{Total\ number\ of\ malicious\ packets\ sent} \times 100$$

In our testing, 682 ICMP packets were collected, and by integrating our predefined threshold, we successfully identified 41.7% of anomalous packets. The positive outcome of this initial detection stage is evident, as it has enabled us to shield the system from 285 harmful packets. The positive aspect of our proposal is that it improves performance and raises the ratio to 60.4% by detecting 127 additional packets.

Our proposed algorithm is evaluated against several existing approaches, namely CLCC algorithm, reliable ARP table approach ARP, GARP, and centralized approach [5], [19], [26], [27]. The performance of these solutions is dependent on the specific environmental context in which the attack is deployed, as well as the particular detection technique applied. Table 5 makes clear that our algorithm performs better than the majority of existing methods. It is worth noting that, when compared to our method, the CLCC algorithm achieves a greater PDR. However, it is important to realize that our solution is deployed on an Openstack architecture, which introduces more complexity when compared to a typical local area network (LAN).

Table 5. Our algorithm's comparison with other solutions

	Attack's environment	Performance
GARP [26]	LAN	52%
Centralized approach [27]	LAN	60%
Reliable ARP table approach [19]	CLOUD (OPENSTACK)	No PDR calculated and loss of resources limitation is stated
Our algorithm	CLOUD (OPENSTACK)	60.4%
CLCC [5]	LAN	77%

4. CONCLUSION

In this study, we delved into the realm of MitM attacks, evaluating several existing techniques aimed at their detection and prevention. Moreover, we put forth a novel algorithm designed specifically for detecting and preventing MitM attacks, leveraging the ICMP protocol and showcasing its superior performance. Our algorithm's effectiveness was validated by surpassing a notable PDR, substantiating the selection of our threshold. These evaluations were conducted through real network testing using the OpenStack platform.

However, similar to any other academic study, our work possesses several limitations that necessitate acknowledgment. Initially, the tests and implementation were conducted on a single server that simultaneously functions as the controller and compute node. This arrangement may have an impact on the effectiveness of the tests. Additionally, the architecture only allows for the detection of MitM attacks and is not effective in mitigating other forms of attacks. Moreover, in subsequent research work, we plan to evaluate the efficacy of machine learning methods in order to enhance the precision and reliability of our findings.

ACKNOWLEDGEMENT

We truly appreciate the anonymous reviewers' insightful recommendations and informative comments, which have significantly raised the level of this paper.

REFERENCES

[1] N. Tissir, S. El Kafhali, and N. Aboutabit, "Cloud Computing security classifications and taxonomies: a comprehensive study and comparison," *2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech)*, Marrakesh, Morocco, 2020, pp. 1-6, doi: 10.1109/CloudTech49835.2020.9365884.

[2] Flexera, "State of the Cloud Report," Flexera, version 11, 2022. [Online]. Available: https://path.flexera.com/cm/report-state-of-the-. (Accessed: Jan. 20, 2023.)

[3] S. Dong, K. Abbas, and R. Jain, "A Survey on Distributed Denial of Service (DDoS) Attacks in SDN and Cloud Computing Environments," in *IEEE Access*, vol. 7, pp. 80813-80828, 2019, doi: 10.1109/ACCESS.2019.2922196.

[4] P. Singh, S. U. Rehman, and S. Manickam, "Comparative Analysis of EMM with Existing State-of-the-Art EDoS Mitigation Techniques in Cloud Computing Environments," *arXiv*, 2019, doi: 10.48550/arXiv.1905.13447.

[5] B. Prabadevi, N. Jeyanthi, N. I. Udzir, and D. Nagamalai, "Lattice structural analysis on sniffing to denial of service attacks," *arXiv*, 2019, doi: 10.48550/arXiv.1907.12735.

[6] A. F. Daru, K. D. Hartomo, and H. D. Purnomo, "IPv6 flood attack detection based on epsilon greedy optimized Q learning in single board computer," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 5, 2023, pp. 5782-5791, doi: 10.11591/ijece.v13i5.pp5782-5791.

[7] A. Sebbar, K. Zkik, Y. Baddi, M. Boulmalf, and M. D. E. El Kettani, "MitM detection and defense mechanism CBNA-RF based on machine learning for large-scale SDN context," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 12, pp. 5875–5894, 2020, doi: 10.1007/s12652-020-02099-4.

[8] S. Sun, X. Fu, B. Luo, and X. Du, "Detecting and Mitigating ARP Attacks in SDN-Based Cloud Environment," *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, 2020, pp. 659-664, doi: 10.1109/INFOCOMWKSHPS50562.2020.9162965.

[9] M. N. Yasir and M. S. Croock, "Cyber DoS attack based security simulator for VANET," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 6, pp. 5832-5843, Dec. 2020, doi: 10.11591/ijece.v10i6.pp5832-5843

[10] W. S. Hwang, J. G. Shon, and J. S. Park, "Web session hijacking defense technique using user information," Human-centric Computing and Information Sciences, vol. 12, pp. 1-14, 2022, doi: 10.22967/HCIS.2022.12.016.

[11] A. A. Galal, A. Z. Ghalwash, and M. Nasr, "A New Approach for Detecting and Mitigating Address Resolution Protocol (ARP) Poisoning," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. 13, no. 6, 2022, doi: 10.14569/IJACSA.2022.0130647.

[12] A. K. Rangisetti, R. Dwivedi, and P. Singh, "Denial of ARP spoofing in SDN and NFV enabled cloud-fog-edge platforms," *Cluster Computing*, vol. 24, no. 4, pp. 3147-3172, 2021, doi: 10.1007/s10586-021-03328-x.

[13] A. Husain, H. Al-Raweshidy, and W. S. Awad. "ARP spoofing detection for IoT networks using neural networks," *Proceedings of the Industrial Revolution & Business Management: 11th Annual PwR Doctoral Symposium (PWRDS) 2020*, 2020.

[14] A. Lahmadi, A. Duque, N.Heraief, and J. Francq, "MitM attack detection in BLE networks using reconstruction and classification machine learning techniques." *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Cham: Springer International Publishing, pp. 149-164, 2020.

[15] V. Rohatgi and S. Goyal, "A Detailed Survey for Detection and Mitigation Techniques against ARP Spoofing," *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, India, 2020, pp. 352-356, doi: 10.1109/I-SMAC49090.2020.9243604.

[16] A. Majumdar, R. Shruti, and T. Subbulakshm, "ARP poisoning detection and prevention using Scapy," *International Conference on Innovative Technology for Sustainable Development 2021 (ICITSD 2021)*, Chennai, India, 27-29 January 2021, doi: 10.1088/1742-6596/1911/1/012022.

[17] D. R. Rupal, D. Satasiya, H. Kumar, and A. Agrawal, "Detection and prevention of ARP poisoning in dynamic IP configuration," *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, India, 2016, pp. 1240-1244, doi: 10.1109/RTEICT.2016.7808030.

[18] P. Arote and K. V. Arya, "Detection and Prevention against ARP Poisoning Attack Using Modified ICMP and Voting," *2015 International Conference on Computational Intelligence and Networks*, Odisha, India, 2015, pp. 136-141, doi: 10.1109/CINE.2015.34.

[19] Z. Chkirbene, A. Erbad, and R. Hamila, "A Combined Decision for Secure Cloud Computing Based on Machine Learning and Past Information," *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, Marrakesh, Morocco, 2019, pp. 1-6, doi: 10.1109/WCNC.2019.8885566.
[20] B. Prabadevi, N. Jeyanthi, and A. Abraham. "An analysis of security solutions for ARP poisoning attacks and its effects on medical computing," *International Journal of System Assurance Engineering and Management*, vol. 11, no. 1, pp. 1-14, 2020.
[21] H. S. Kang, J. H. Son, and C. S. Hong, "Defense technique against spoofing attacks using reliable ARP table in cloud computing environment," *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Busan, Korea (South), 2015, pp. 592-595, doi: 10.1109/APNOMS.2015.7275401.
[22] N. Tissir, S. El Kafhali, and N. Aboutabit," How Much Your Cloud Management Platform Is Secure? OpenStack Use Case," *The Proceedings of the 5th International Conference on Smart City Applications*, Springer, Cham. 2020, pp. 1117-1129, doi: 10.1007/978-3-030-66840-2_85.
[23] S. Lima, A. Rocha, and L. Roque, "An overview of OpenStack architecture: a message queuing services node," *Cluster Computing*, vol. 22, pp. 7087-7098, 2019.
[24] G. D. Singh, "Learn Kali Linux 2019," 1st ed. Packt Publishing, 2019.
[25] H. Iqbal and S. Naaz, "Wireshark as a tool for detection of various LAN attacks," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 5, pp. 833-837, 2019, doi: 10.26438/ijcse/v7i5.833837.
[26] S. Dangol, S. Selvakumar, and M. Brindha, "Genuine ARP (GARP)," *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 4, pp. 1-10, 2011, doi: 10.1145/1988997.1989013.
[27] S. Kumar and S. Tapaswi, "A centralized detection and prevention technique against ARP poisoning," *Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*, Kuala Lumpur, Malaysia, 2012, pp. 259-264, doi: 10.1109/CyberSec.2012.6246087.

BIOGRAPHIES OF AUTHORS

Najat Tissir graduated from Khouribga National School of Applied Sciences with a state engineering diploma in network and telecommunications engineering, in 2017. She is a Ph.D. student at Process Engineering, Computer Science and Mathematics Laboratory, Sultan Moulay Slimane University, Beni Mellal, Morocco. She is currently working as the head of infrastructure and IT resources, at Hassan First University. Her research areas include cloud computing security, cybersecurity, and ARP spoofing attacks detection. She can be contacted at email: tissir.najat@gmail.com.

Noureddine Aboutabit received his engineering degree from Ecole Normale Supérieure d'Ingénieurs Electriciens de Grenoble (ENSIEG) in 2003 and his M.S. from Grenoble INP (France) in 2004. He earned his Ph.D. in Signal Image Speech Telecom from Grenoble INP in 2007. Since January 2023, he has been a Professor in computer science at ENSA-Khouribga. His research. His research focuses on computer vision, machine learning, multimodal speech processing, big data, and cloud computing security. He can be contacted at email: n.aboutabit@usms.ma.

Said El Kafhali is a computer science professor at the Department of Mathematics and Computer Science, Faculty of Sciences and Techniques, Hassan First University of Settat, Morocco, where he has been a member since January 2018. Prior to joining the Faculty of Sciences and Techniques, he spent four years at the National School of Applied Sciences of Khouribga, Morocco. His current research focuses on queuing theory, performance modeling and analysis, cloud computing, the internet of things, fog/edge computing, networks security, machine learning, and artificial intelligence. He is actively involved with various international journals as a reviewer and has served as an international program committee member for numerous peer-reviewed conferences. He has made significant contributions to the field, with extensive publications in computer systems modeling and performance, queueing theory, cloud computing, internet of things, artificial intelligence, and data sciences. He can be contacted at email: said.elkafhali@uhp.ac.ma.