Journal of Advanced Engineering Trends
ISSN : 2682 - 2091

Vol. 44, No. 2, July 2025

Detection and Mitigation of Stealth DHCP Attack in SDN network

Nadia H. Mohammed[1,*], Nagwa Salem[1], Kamel Rahouma[1]
[1]*Electrical Engineering Dep., Faculty of Engineering, Minia University, Minia, Egypt*

** Corresponding author(s) E-mail: eng.nadiahassan17@gmail.com*

ARTICLE INFO

Article history:
Received: 23 January 2025
Accepted: 18 Mar 2025
Online: 3 May 2025

Keywords:
SDN
ONOS
Mininet
DHCP
DHCP starvation attack

ABSTRACT

Software-defined networking (SDN) significantly enhances network management through its centralized controller, which operates independently of forwarding devices. However, SDN security remains a major challenge. It inherits vulnerabilities from traditional networks due to shared protocols and introduces new risks from its reliance on software-based systems. Dynamic Host Configuration Protocol (DHCP), a critical protocol in SDNs, also presents security threats. This study analyzes the impact of the stealth DHCP starvation attack in an SDN environment where the Open Network Operating System (ONOS) controller acts as a DHCP relay agent. The analysis reveals that this configuration is susceptible to stealth DHCP starvation attacks, which can disrupt network functionality. A Python script is developed and deployed on the DHCP server Virtual Machine (VM) to address this vulnerability. The script effectively prevents harmful DHCP messages, restores the IP address pool, and mitigates DHCP-related attacks without imposing significant system overhead. Results demonstrate that the proposed solution not only enhances network resilience against attacks, but also improves overall performance. Specifically, it increases the throughput from 66.0 Mbits/sec to 101.5 Mbits/sec, while the average Round Trip Time (RTT) is reduced from 455.0 ms to 0.45 ms. Additionally, the transmission rate improves from 46,800 Packet Per Second (pps) to 72,000 pps, ensuring better resource utilization. The proposed approach provides a practical and efficient method for safeguarding SDN environments against advanced DHCP-related threats, contributing to the secure and efficient operation of modern networks.

1. Introduction

The expansion of online communication necessitates innovative approaches in communication system technology to achieve optimization of resource use and support efficient scaling. The SDN model decouples the control plane from the data plane, shifting network administration tasks from hardware devices to centralized software controllers. The controller manages the network state and determines how data moves through basic network devices. As a result, the SDN controller is fundamental to the SDN architecture, and any breach of its security threatens the entire network's safety. Security in SDN is a complex challenge. It inherits vulnerabilities from traditional networks via shared protocols and introduces new issues related to software-based risks. The DHCP is crucial for SDNs, but its security vulnerabilities also threaten these networks.

The DHCP obtains network configuration settings, such as IP addresses, from a DHCP server. However, it is susceptible to a Denial-of-Service (DoS) attack known as the classical DHCP starvation attacks. In these attacks, a malicious client sends numerous IP requests using spoofed MAC addresses. Each time the server receives one of these requests, it allocates a new IP address, ultimately leading to the depletion of available addresses on the DHCP server. In this paper, A stealth DHCP starvation attack, which differs from the classical attack, is examined. This attack takes advantage of the IP address conflict detection mechanisms present in all DHCP clients. It is particularly stealthy, as many common security features found in modern network switches are unable to detect it. Furthermore, other suggested approaches that fall into one of three categories: encryption, threshold-based, or fair IP address allocation, have several shortcomings, including high misclassification rates, difficult threshold-setting processes in threshold-based mechanisms, and the impossibility of allocating IP addresses to every port in wireless networks.

Specifically, the paper provides the following contributions:

- A stealthy DHCP starvation attack, that is simple to execute and hard to detect using existing security measures, is introduced.
- The effectiveness of the proposed DHCP starvation attack is evaluated in an SDN network where the ONOS controller acts as a DHCP relay agent.
- A Python script is created on the DHCP server VM to detect and block stealth DHCP attacks and remove the malicious hosts from the network.
- The effectiveness of the proposed mitigation strategy is assessed, demonstrating significant improvements in the

throughput (from 66.0 Mbits/sec to 101.5 Mbits/sec) and reducing RTT (from 455.0 ms to 0.45 ms), among other performance gains.

This paper is structured as outlined below: Section 1 is the Introduction. Section 2 offers a brief review of recent research on SDN and an overview of the Dynamic Host Configuration Protocol service and DHCP attacks. Section 3 discusses the Stealth DHCP attack and how to detect and prevent it. Section 4 evaluates the robustness of SDN controllers and DHCP Servers through anti-attack tests. Finally, Section 5 concludes the work by summarizing the findings and proposing future research topics.

Research Significant

A series of tests were conducted to examine the impact of shrinkage inhibitors (SIA) on the mechanical characteristics of both fresh and hardened self-compacting concrete. To achieve this objective, a total of twenty-five distinct concrete mixtures were produced and examined, each including various combinations of concrete additives. This experimental program investigated the impact of using various types of cement, specifically high-range water-reducing and retarding types, with two different contents of supplementary cementitious materials, as well as the inclusion of silica fume, on the properties of both fresh and hardened concrete. Significant findings were derived from conducting several tests on fresh and hardened concrete mixtures.

2. Related Works

2.1. The DHCP

The DHCP protocol allows TCP/IP configurations to be managed centrally, efficiently handling network changes. Manually assigning IP addresses and other network settings (such as the default gateway and DNS server IP addresses) to each device on a network can be time-consuming and error-prone. For example, there's a risk of IP address conflicts if the same address is accidentally assigned to multiple devices. Because of these challenges, most networks today have one or more DHCP servers. The DHCP servers automatically assign IP addresses to devices without conflict. A DHCP server is set up with a range of IP addresses and additional network settings. When a client joins a network, it attempts to configure its interface with an IP address by exchanging four messages with the DHCP server, as illustrated in Figure 1. These messages are DISCOVER, OFFER, REQUEST, and ACKNOWLEDGMENT (ACK), forming the (Discover, Offer, Request, and Acknowledge) (DORA) process for IP address allocation. Additionally, there are other message types exchanged if the IP address configuration is unsuccessful. One example is the DHCPDECLINE message, which the client sends to the DHCP server if it detects that the assigned IP address is already being used by another client. To identify these conflicts, a DHCP-enabled client uses ARP request probes to verify whether the IP address is currently in use.

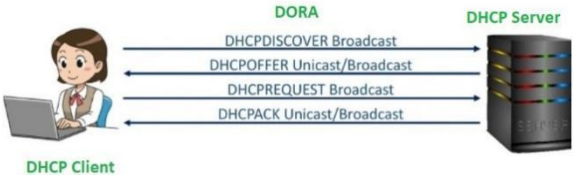

Figure 1: DHCP DORA Process

2.2. The DHCP Attacks

Understanding the types of DHCP attacks is crucial for developing detection and mitigation strategies.

- The DHCP Starvation Attack: The attacker floods the network with DHCP requests, consuming all available IP addresses. Legitimate devices cannot obtain an IP address, leading to a denial of service.

- The DHCP Spoofing Attack: To respond to client's DHCP queries, the malicious client uses an illegal DHCP server, providing them with information about DNS servers, default gateways, and fake IP addresses. By delivering these incorrect settings, the attacker is capable of executing man-in-the-middle (MITM) attacks, redirecting clients to harmful websites, or infecting their systems with malware.

2.3. Detection and Mitigation Techniques for DHCP attacks.

The DHCP protocol is inherently insecure and vulnerable to DoS attacks, such as DHCP starvation. The proposed detection method, which is based on port scanning, proved to be highly effective in identifying DHCP starvation attacks on both local and remote DHCP networks. However, it cannot detect a host that blocks all TCP ports. Furthermore, scanning 1000 TCP ports for each IP address requires a considerable amount of time [1].

A paper [2] introduces another solution to detect and prevent DHCP starvation attacks as well as to address the shortcomings of traditional port security techniques. The proposed solution includes a detailed algorithm for a fair IP address allocation strategy across ports on a network edge switch. Additionally, the solution is applied to both wired and wireless networks. Simulation results indicate that the approach significantly outperforms methods like fixed IP allocation per port and malicious DHCP request rate detection.

Also, authors in [3] introduce a stealth starvation attack that is very efficient in wireless networks, simple to execute, needing less transmitted messages, and challenging to detect using existing methods. Additionally, a structurally comparable attack targeting IPv6 networks is demonstrated, which could disrupt address configuration protocols such as DHCPv6 and State-less Address Auto-configuration (SLAAC). Authors further explain a method for detecting anomalies to detect this attack. An actual network environment was used to design and test the attack, and the results were documented. The proposed detection method leverages the Hellinger distance between two probability distributions derived from training and testing data to identify the starvation attack.

Authors, in [4], evaluate whether the suggested attack is effective in IPv4 and IPv6 networks, demonstrating that it can effectively stop other clients from acquiring IP addresses, which would lead to a DoS scenario. They counteract this threat by proposing an anomaly detection framework based on Machine Learning (ML). Specifically, they utilize well-known one-class classifiers for detection. By capturing IPv4 and IPv6 traffic from an actual network consisting of several devices, they assess the performance of various ML algorithms. According to their tests, these algorithms are highly accurate at detecting attacks in both IPv4 and IPv6 environments.

A study [5] seeks to create an effective solution for detecting and automatically neutralizing rogue DHCP servers using a Python-powered detection system. It begins by illustrating a MITM attack and its consequences through the deployment of a rogue DHCP server. Following this, the research presents a Python-based engine capable of identifying and neutralizing rogue DHCP servers by differentiating them from legitimate ones through a comparison of whitelisted IPs and their corresponding MAC addresses. Lastly, the proposed method's effectiveness is demonstrated and validated using Multivendor Network Emulation Software.

A study [6] examines the primary vulnerabilities of the DHCP protocol, such as its absence of authentication, confidentiality, and integrity, which make it susceptible to various attacks. These include rogue DHCP server attacks, DHCP starvation attacks, and replay attacks. Additionally, the research evaluates the strategies proposed to detect and prevent these threats, providing a detailed analysis of their benefits and limitations.

2.4. The SDN

The SDN has improved the networking industry by solving issues related to scalability, flexibility, and control mechanisms that are common in conventional networks. A major advantage of SDN compared to traditional networks is its centralized management. Traditional networks lack a separate control layer, making network configuration and management difficult. The SDN introduces a distinct control layer, simplifying network management and making it more efficient. Network management is now more flexible and less reliant on costly hardware and configuration changes. This benefits not only network administrators but also potential network intruders. The SDN separates the control and data planes of the network, but the separation of the control plane can create a single point of failure if targeted by DoS or Distributed DoS (DDoS) attacks. Consequently, security is a critical concern for the SDN.

A paper [7] discusses the security challenges across all three layers of the SDN and outlines the solutions that network administrators should implement. It highlights several key benefits of the SDN security which continues to be a prominent topic in networking. The primary objective of this paper is to review the significant advancements in the SDN security, along with their scope and limitations.

A study [8] examines the security of built-in DHCP services on three widely used SDN controllers: POX, ONOS, and Floodlight. The findings reveal that these controllers are susceptible to starvation attacks, and floods of DHCP discovery messages can be exploited to launch denial-of-service (DoS) attacks, causing network slowdowns and overloading the controllers. To address these vulnerabilities, the researchers evaluated advanced DHCP security methods and their application to the DHCP servers embedded in SDN controllers. Based on their analysis, they developed and implemented a DHCP security Guard, on the POX controller, incorporating features such as DHCP snooping, rate limiting, and IP pool recovery.

The controller serves as the "brain" of SDN technology, driving its intelligent functionality. Different SDN controllers have been developed to meet the varied demands and tasks required of them. While many commercial SDN controllers exist, there are also numerous open-source options available. Among these, the ONOS has emerged as a prominent trend in open-source SDN solutions. A paper [9] conducts a comparative analysis of open-source SDN controllers, including Network Controller Platform (NOX), Python-based Network Controller (POX), the component-based SDN framework (Ryu), the Java-based OpenFlow controller (Floodlight), Open Day Light (ODL), and ONOS. The discussion delves into the architecture and evolution of ONOS controllers and examines their use cases across various applications.

A paper [10] aims to compare the performance of two SDN controllers, ODL and ONOS, based on their efficiency. The study utilizes Mininet to simulate switching and end devices, alongside tools like Wireshark Packet Analyzer and iperf, which monitor real-time traffic flow between Mininet and the controller. The performance of an SDN controller is a critical factor in determining how effectively it functions in a network, and various metrics are used to evaluate this performance. The analysis reveals that ONOS outperforms ODL in terms of throughput, TCP window scaling, TCP/UDP bandwidth, burst rate, jitter, goodput, TCP Stevens Graph, RTT, and overall usability. This study provides valuable insights into the performance capabilities of ONOS and ODL, particularly in medium-to-large-scale SDN networks.

A paper [11] explores the control plane architectures and examines various SDN controllers that support them. The authors evaluated SDN controllers based on key performance metrics, including scalability, reliability, consistency, and security. Additionally, they analyzed the mechanisms employed by these controllers to address these metrics, highlighting the advantages and disadvantages of each approach. Furthermore, the manuscript identifies several research challenges and open issues related to different SDN control plane architectures.

3. Methodology

This part introduces the stealth DHCP starvation attacks and how to detect and prevent them in SDN networks.

3.1. *Stealth DHCP Starvation attack*

As shown in Figure 2, the steps of executing a stealth DHCP starvation attack on an IPv4 network are as follows:

- IP address is assigned manually by an illegal host: When an illegal host connects to the network, it disables the DHCP service running in the background and manually assigns itself an IP address along with other configuration details. This prevents the DHCP server from assigning an IP to the malicious client. It keeps its IP-MAC binding out of the server's DHCP database. This tactic allows the malicious client to evade the Dynamic ARP Inspection (DAI) security feature, which relies on the DHCP snooping database to detect discrepancies in IP-MAC bindings.
- A victim client connects to the network and acquires an IP address from the DHCP server through the DORA process. Once the allocation is successful, the DHCP server records the IP-MAC binding for the victim client in its database.
- Victim's ARP request broadcast: The victim client sends a broadcast ARP request to confirm whether the assigned IP is currently being used. At this stage, the victim's source IP is set to "0.0.0.0" since it hasn't configured its network interface with the assigned IP yet.
- Malicious client's ARP request broadcast: Upon receiving the victim's ARP request, the malicious client broadcasts its own ARP request. The malicious request uses a source IP of "0.0.0.0," the target IP as the assigned IP being probed, a target MAC of "00:00:00:00:00:00," and a broadcast destination MAC of "ff:ff:ff:ff:ff:ff." The source MAC in the Ethernet header is the malicious client's MAC.
- Victim's decline message: After detecting the malicious ARP response, the victim client broadcasts a DHCPDECLINE message to the DHCP server, rejecting the assigned IP address. The DHCP server then marks this IP as unavailable for the duration of the lease period. The victim client restarts the process of acquiring an IP address, but this cycle repeats, preventing the victim from obtaining a valid IP. Over time, the DHCP server runs out of available IPs, causing a DoS attack since all declined IPs remain unusable for their lease duration.

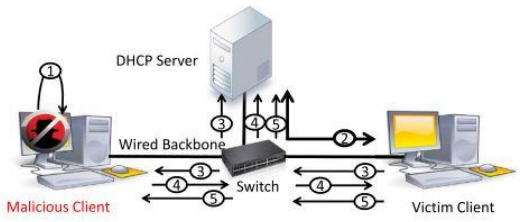

Figure 2: Stealth DHCP attack Process [4]

3.2. *The ONOS*

- The ONOS is an open-source SDN controller written in Java, designed to work seamlessly across various platforms such as Linux, Windows, and macOS. It includes a web-based GUI and it can be deployed with cluster support in Docker containers. Targeting carrier networks, the ONOS adopts a distributed architecture to ensure high availability and scalability. It empowers large networks by facilitating the integration of new SDN services alongside their existing functionalities. Unlike other controllers, the ONOS supports hybrid network environments [12].
- The ONOS includes a built-in DHCP application, and the configuration of the DHCP server is done using a JSON file. To activate the DHCP service, the DHCP server application must be loaded onto the controller through either the GUI or CLI. Afterward, the onos-netcfg binary needs to be executed, specifying the controller's IP address and the JSON file's directory as parameters
- In present architecture, the ONOS SDN controller is not used as a DHCP server. It is used as a DHCP relay agent. The DHCP relay application is activated through the GUI. Then, the onos-netcfg file is edited, specifying the DHCP server's IP.
- Using the ONOS, as a DHCP relay agent, allows it to block rogue DHCP messages before they reach clients. If a DHCP response comes from an unauthorized source, this source is blocked by the ONOS blocks by applying the flow rules. This ensures that only the authorized DHCP server (my DHCP VM) provides IP addresses.
- The forward mode and other applications in the ONOS application are also activated as shown in Figure 3.

3.3. *MININET*

Mininet is a powerful, open-source network emulator for creating, testing, and prototyping SDNs and other network architectures. It allows developers and researchers to create virtual network environments on a single machine, making it easy to experiment with network configurations and protocols without requiring physical hardware.

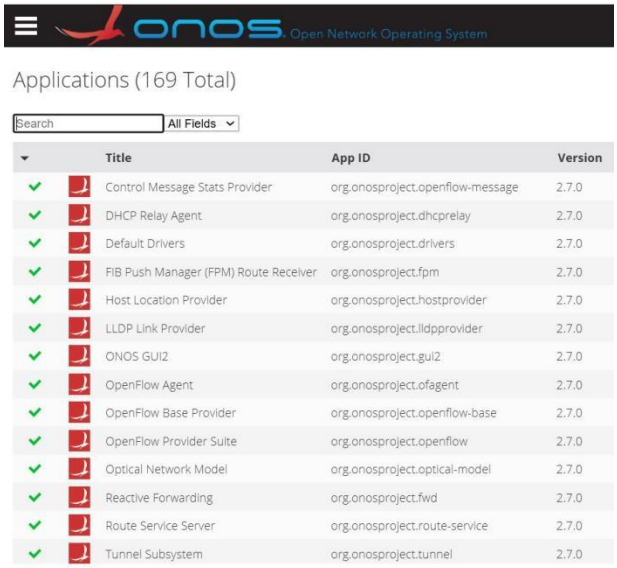

Figure 3: The ONOS active applications

- Mininet can emulate a full network with hosts, switches, links, and controllers running as lightweight processes or virtual machines on a single computer.
- It supports large-scale network topologies, making it suitable for simulating complex networks while keeping resource usage minimal.
- It integrates seamlessly with SDN controllers (e.g., ODL, Ryu, ONOS), making it a popular choice for SDN research and development.

3.4. Principle of the defensive strategy

- The approach should guarantee that the DHCP server handles service requests from authorized hosts accurately and without interruption. It is essential to identify what constitutes an attack and determine the traffic patterns and threshold values that need monitoring.
- The solution must identify the DHCP attack before it disrupts the network communication and causes the DHCP server to become unavailable.
- To stop malicious traffic from flowing over the network, the solution must separate the attacker from the network, block the attack at the point nearest to its source, and repair any harm the assault has caused to the DHCP server and network architecture.

The DHCP security solution aims to accomplish these objectives. This is compatible with other SDN controllers that use the same protection logic, regardless of the network topology or complexity. It imposes no processing load that could affect controller performance. It also offers rapid IP pool restoration to improve DHCP server uptime.

3.5. The architecture design

The proposed design consists of 4 VMs:

- The ONOS SDN controller, which controls traffic and in the present scenario, it works as a DHCP relay agent.
- Mininet VM
- DHCP server VM
- Hacking VM

Table 1 lists every element that has been used in our virtual lab.

Table 1: The present virtual lab components

VM	VM Specification Operating System	RAM	Core
ONOS	Ubuntu 18.04.6	4	1
Mininet	Mininet 22.04	4	1
DHCP Server	Ubuntu 22.04	4	1
Hacker	Ubuntu 20.04	4	1

3.6. Effectiveness of the Stealth DHCP attack on the QOS

In this section, the effects of the stealth DHCP starvation attack on the performance of the network is tested. A tree topology on the Mininet VM is designed and connected to the ONOS SDN controller. The performance of this network is tested during the attack to:

- Determine if the hosts can obtain IPs from the DHCP server.
- Execute a Python script on H7 which functions as the attacker. After that, H5 and H6 are rebooted to verify whether they can obtain an IP address from the DHCP server. Finally, the impact of the attack on network performance is assessed.
- The RTT is measured using ICMP messages between H3 and H16. About 1000 messages are sent and the RTT is assessed.
- The throughput is measured using Wireshark.
- For bandwidth measurements, iperf is utilized.

Next, a detection script on H1 is executed and a check whether H5 and H6 can regain IP addresses is done. Then, the performance measurements is repeated to compare the network's performance during the attack and after detection.

4. Experiments and performance evaluation

We assign a static IP address to each ONOS SDN controller, Mininet VM, hacking VM, and DHCP server VM, as in Table 2.

Table 2: The IP addresses of the virtual lab components

VM	IP Address
ONOS	192.168.1.100
Mininet	192.168.1.30
DHCP Server	192.168.1.150
Hacker	192.168.1.20

4.1. Testing connectivity between Mininet and ONOS

A basic tree topology is set up in Mininet and connected to the ONOS SDN controller, as shown in Figure 4. The Mininet hosts obtained IP addresses from the Mininet pool.

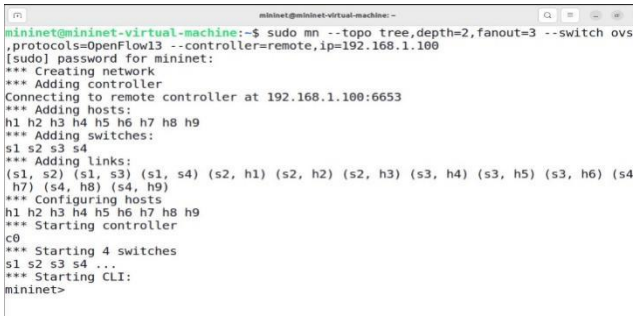

Figure 4: Command to create a simple tree topology.

Figure 5 demonstrates the structure of this tree topology. The ONOS SDN controller is connected to the Mininet hosts. All Mininet hosts take IP from the Mininet pool "10.x.x.x/8". By default, Mininet assigns hard-coded 10.0.0.0/8 range of IP addresses to Mininet hosts.

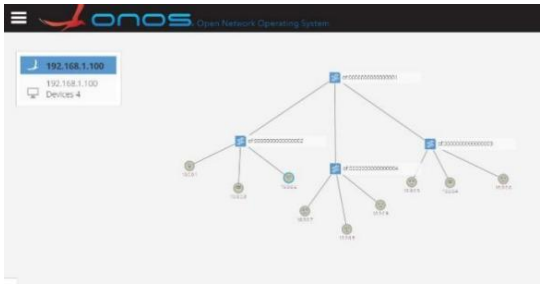

Figure 5: Simple tree topology

4.2. Ensuring that Mininet hosts can take IP addresses from the external DHCP server

The DHCP pool range is set up from "192.168.1.180/24" to "192.168.1.200/24". A Python script is executed on the Mininet VM to create a simple topology, connecting it to the ONOS SDN controller. This setup allows Mininet hosts to obtain IP addresses from our external DHCP server. As shown in Figure 6, the Mininet hosts can connect to the external DHCP server and retrieve IP addresses from its pool.

```
mininet@mininet-virtual-machine:~$ sudo python3 /home/mininet/Desktop/test/dhcp.p
Requesting IP for h1...
Requesting IP for h2...
h1 IP address: h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.181  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::f0f1:6cff:fe7c:8bb3  prefixlen 64  scopeid 0x20<link>
        ether f2:f1:6c:7c:8b:b3  txqueuelen 1000  (Ethernet)
        RX packets 32  bytes 4511 (4.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 15  bytes 2470 (2.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


h2 IP address: h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.182  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::ac94:b6ff:fe7d:9b05  prefixlen 64  scopeid 0x20<link>
        ether ae:94:b6:7d:9b:05  txqueuelen 1000  (Ethernet)
        RX packets 31  bytes 4441 (4.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1514 (1.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 6: Hosts (h1 and h2) can access DHCP server

4.3. Executing the hacking code on the Hacking VM

A Python hacking script is executed on a designated virtual machine, which captures ARP requests with a source IP of 0.0.0.0. The script modifies the source MAC address, replacing it with a new one, and rebroadcasts the altered ARP request. As a result, the victim client sends a DHCPDECLINE message to the DHCP server, rejecting the assigned IP address. The DHCP server marks the declined IP as unavailable for its lease. The victim client then retries the IP acquisition process, but the cycle repeats, preventing it from obtaining a valid IP. Over time, the DHCP server exhausts its pool of available IPs, leading to a DoS attack as all declined IPs remain unusable for the lease period. Figure 7 shows the hacker's flowchart and Figure 8 shows the code running result. The hacking algorithm is as follows:

```
If the packet is an ARP request:
   If the source IP is “0.0.0.0”:
      Replace the source MAC address with a new one
      Rebroadcast the modified ARP request
   Else:
      Drop the packet
   End
Else:
   Drop the packet
End
```

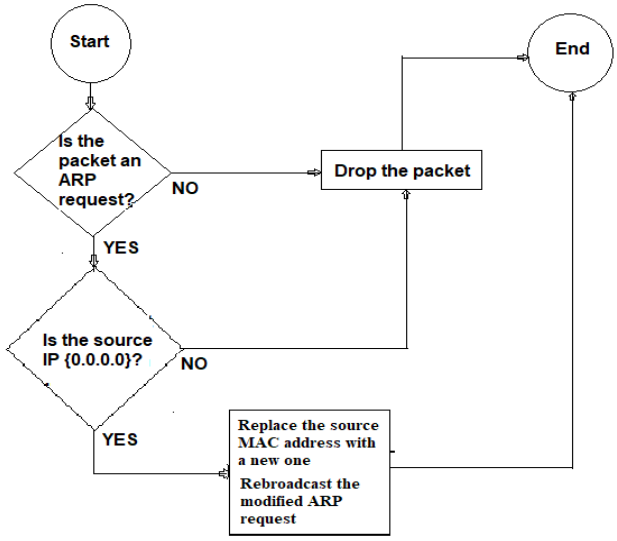

Figure 7: Hacking flow chart

```
mininet@mininet-virtual-machine:~$ sudo python3 /home/mininet/Desktop/test/dhcp.py
[sudo] password for mininet:
Requesting IP for h1...
Requesting IP for h2...
h1 IP address: h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::dc58:27ff:fe6e:8f06  prefixlen 64  scopeid 0x20<link>
        ether de:58:27:6e:8f:06  txqueuelen 1000  (Ethernet)
        RX packets 83  bytes 15247 (15.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 56  bytes 10912 (10.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


h2 IP address: h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::24c3:9aff:febd:20ea  prefixlen 64  scopeid 0x20<link>
        ether 26:c3:9a:bd:20:ea  txqueuelen 1000  (Ethernet)
        RX packets 86  bytes 14521 (14.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 54  bytes 11728 (11.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 8: Hackers (h1 and h2) cannot take IP

4.4. Evaluating the performance of the Python defense code

Before running the detection code on the DHCP server VM, the following prerequisites should be met: install the necessary tools, including **iptables** (for managing network traffic), **tcpdump** (for capturing and analyzing network packets), and **arptables** (for filtering ARP packets). The detection algorithm is:

```
If the packet is a discovery message:
   Process the packet
Else
   If the packet an ARP request with IP {0.0.0.0}:
      If there is discovery message for the MAC address:
         Process the packet
      Else
         Drop the packet
      End
   Else:
      Drop the packet
   End
End
```

Figure 9 shows the detection and prevention flowchart and Figure 10 shows the result of running the code.

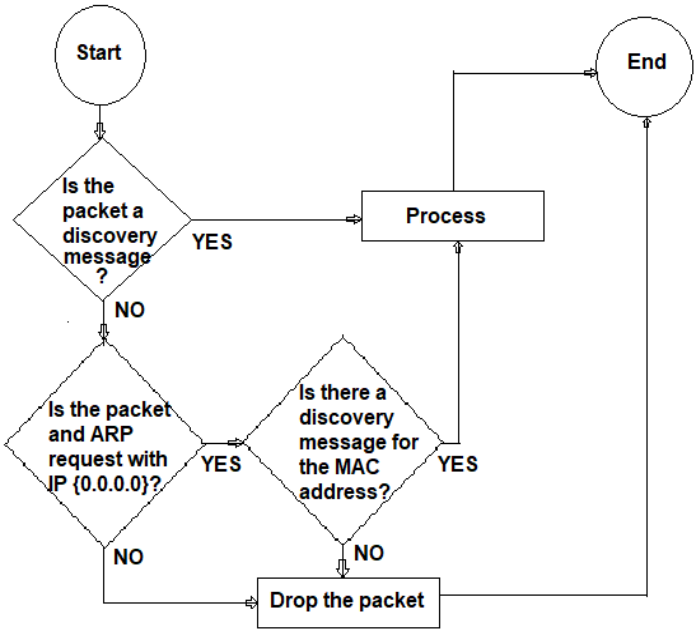

Figure 9: Detection & prevention flow chart

The Python detection script generates a file to store the MAC address table and discovery messages. It then adds MAC addresses and discovery messages to the table while filtering ARP Probe Requests originating from the IP address "0.0.0.0". The script checks whether a discovery message exists for the given MAC address. If a discovery message is found, the message is processed; if not, the packet is dropped using iptables, and the source port is blocked. Additionally, the system blocks network traffic from suspicious MAC addresses or based on specific port activity. The outcome of running this script on the DHCP server is given in Figure 10 showing that H1 and H2 can successfully obtain IP addresses from the DHCP server.

```
mininet@mininet-virtual-machine:~$ sudo python3 /home/mininet/Desktop/test/dhcp.py
Requesting IP for h1...
Requesting IP for h2...
h1 IP address: h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.181  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::f0f1:6cff:fe7c:8bb3  prefixlen 64  scopeid 0x20<link>
        ether f2:f1:6c:7c:8b:b3  txqueuelen 1000  (Ethernet)
        RX packets 32  bytes 4511 (4.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 15  bytes 2470 (2.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


h2 IP address: h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.182  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::ac94:b6ff:fe7d:9b05  prefixlen 64  scopeid 0x20<link>
        ether ae:94:b6:7d:9b:05  txqueuelen 1000  (Ethernet)
        RX packets 31  bytes 4441 (4.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1514 (1.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 10: Hosts (h1 and h2) can obtain IPs after running the detection and prevention code

4.5. Evaluating the network performance under the attack and after mitigation

This section demonstrates the effect of stealth DHCP starvation attack on Quality of Service (QoS). A tree topology is created in Mininet with a fanout of 4 and a depth of 2, connecting it to the ONOS SDN controller. Host H1 is configured as the DHCP server, with a DHCP pool ranging from "192.168.1.75/24" to "192.168.1.95/24", a default gateway of "192.168.1.1", and a DNS server set to "8.8.8.8". Static IPs are assigned to H1 ("192.168.1.50") and H7 ("192.168.1.15"), while the remaining hosts receive their IP addresses dynamically from the H1 DHCP server. The Mininet script is executed to configure this setup, as shown in Figure 11, H1 uses the static IP and H2 receives an IP from the DHCP server.

```
mininet@mininet-virtual-machine:~$ sudo python3 /home/mininet/Desktop/test/qos.py
[sudo] password for mininet:
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.50  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::9899:53ff:fe57:a86c  prefixlen 64  scopeid 0x20<link>
        ether 9a:99:53:57:a8:6c  txqueuelen 1000  (Ethernet)
        RX packets 20  bytes 3006 (3.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 656 (656.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mininet> h2 ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.75  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::b467:4ff:fef9:85ef  prefixlen 64  scopeid 0x20<link>
        ether b6:67:04:f9:85:ef  txqueuelen 1000  (Ethernet)
        RX packets 21  bytes 3209 (3.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 656 (656.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
```

Figure 11: Creating topology

The hacking code is executed on H7. Then, H5 and H6 are rebooted to observe whether they can obtain an IP from the DHCP server. Figure 12 shows that they are unable to acquire an IP. Thus, the impact of the attack on network functionality and performance can be evaluated. The RTT is measured by sending approximately 1000 ICMP messages between H2 and H16. Additionally, the iperf is used to measure the bandwidth utilization, and also both Goodput and Throughput are measured. Next, the detection script is executed on H1. Once H5 and H6 are confirmed to be able to obtain IP addresses, the same network performance tests conducted in the previous section are repeated. This allows to compare the network performance before and after the mitigation of the attack. The results are presented in Table 3.

The results demonstrate that the proposed solution not only enhances network resilience against attacks, but also improves the overall performance. Specifically, it increases the throughput from 66.0 Mbits/sec to 101.5 Mbits/sec, while the average RTT is reduced from 455.0 ms to 0.45 ms. Additionally, the transmission rate improves from 46,800 pps to 72,000 pps, ensuring better resource utilization

```
mininet> h5 ifconfig
h5-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::200:ff:fe00:5  prefixlen 64  scopeid 0x20<link>
        ether 00:00:00:00:00:05  txqueuelen 1000  (Ethernet)
        RX packets 23  bytes 3429 (3.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14  bytes 1212 (1.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mininet> h6 ifconfig
h6-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::200:ff:fe00:6  prefixlen 64  scopeid 0x20<link>
        ether 00:00:00:00:00:06  txqueuelen 1000  (Ethernet)
        RX packets 27  bytes 3975 (3.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 16  bytes 1352 (1.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 12: H5 and H6 after running the hacking code

Table 3: Performance comparison before & after mitigation

Metric	Under Attack	After Mitigation
Max Transmission Rate	48,750 pps	75,000 pps
Min Transmission Rate	44,200 pps	68,000 pps
Average Transmission Rate	46,800 pps	72,000 pps
Max Throughput	67.0 Mbits/sec	103.0 Mbits/sec
Min Throughput	65.6 Mbits/sec	101.0 Mbits/sec
Average Throughput	66.0 Mbits/sec	101.5 Mbits/sec
Max Goodput	63.7 Mbits/sec	98.0 Mbits/sec
Min Goodput	62.1 Mbits/sec	95.5 Mbits/sec
Average Goodput	63.2 Mbits/sec	97.2 Mbits/sec
Min RTT	135.0 ms	0.033 ms
Max RTT	730.0 ms	22.57 ms
Average RTT	455.0 ms	0.45 ms

5. Conclusions

The SDN has improved the network management by separating the control and data planes. However, while SDN infrastructure relies on some fundamental protocols that are likely utilized in conventional networks for client-server communication, many of these protocols have built-in flaws. Efforts to counteract these vulnerabilities have shown varying degrees of success. The SDN offers an ideal framework for developing innovative and practical solutions to address these security challenges. For example, the centralized controller can access information that is not obtainable through packet flow in conventional networks. Additionally, identifying network port, the incoming packet's source device is linked to, simplifies the prevention of ARP and DHCP attacks. It supports the implementation of network access control systems within SDN environments.

The SDN controllers are continuously enhanced with new features and capabilities. However, these enhancements could introduce new vulnerabilities if security guidelines are overlooked within the design stage. The present research aimed to secure the DHCP service within the SDN framework. The robustness of the DHCP server has been assessed using source code and other experiments. The ONOS SDN controller is used as a DHCP relay agent. The findings reveal that the DHCP server is susceptible to DHCP attacks. These attacks could compromise the DHCP server's availability and reduce the functionality of the network.

The present approach combines ONOS as a DHCP relay agents with a custom script running on the DHCP VM. It implements multiple security mechanisms to protect against DHCP-based attacks. The ONOS enforces static rule-based prevention through flow rules, effectively blocking rogue DHCP messages before they reach clients. Additionally, rate-limiting is applied per host to prevent DHCP starvation attacks, ensuring that excessive requests from a single source do not overwhelm the network. The ONOS also provides anomaly detection by monitoring DHCP traffic patterns, helping to identify irregularities that may indicate an ongoing attack. The proposed technique can effectively detect and prevent stealth DHCP attacks, whether preemptively before they occur or in response to an ongoing attack. However, the approach does not currently incorporate ML-based detection, which can enhance adaptability against evolving threats. In terms of scalability, ONOS offers moderate support for large networks, allowing the existence of centralized traffic control and policy enforcement, though performance may need further optimization for high-density environments. The present solution effectively detects and prevents DHCP-related attacks. It does not include automated or dynamic defense capabilities that adjust security measures in real time. In the future, the different threats targeting the SDN network infrastructure will be aimed to be identified & mitigated.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] A. Jony, A. S. M. Miah, and M. N. Islam, "An Effective Method to Detect DHCP Starvation Attack using Port Scanning," *2023 Int. Conf. Next-Generation Comput. IoT Mach. Learn. NCIM 2023*, no. August, 2023, doi: 10.1109/NCIM59001.2023.10212773.

[2] H. Mukhtar, K. Salah, and Y. Iraqi, "Mitigation of DHCP starvation attack," *Comput. Electr. Eng.*, vol. 38, no. 5, pp. 1115–1128, 2012, doi: 10.1016/j.compeleceng.2012.06.005.

[3] N. Hubballi and N. Tripathi, "A closer look into DHCP starvation attack in wireless networks," *Comput. Secur.*, vol. 65, pp. 387–404, 2017, doi: 10.1016/j.cose.2016.10.002.

[4] N. Tripathi and N. Hubballi, "Detecting stealth DHCP starvation attack using machine learning approach," *J. Comput. Virol. Hacking Tech.*, vol. 14, no. 3, pp. 233–244, 2018, doi: 10.1007/s11416-017-0310-x.

[5] A. Jony and M. N. Islam, "An Effective Technique to Automatically Detect and Neutralize Rogue DHCP Server," *2023 Int. Conf. Inf. Commun. Technol. Sustain. Dev. ICICT4SD 2023 - Proc.*, no. September, pp. 244–248, 2023, doi: 10.1109/ICICT4SD59951.2023.10303033.

[6] A. A. Abdulghaffar, S. K. Paul, and A. Matrawy, "An Analysis of DHCP Vulnerabilities, Attacks, and Countermeasures," *2023 Bienn. Symp. Commun. BSC 2023*, pp. 119–124, 2023, doi: 10.1109/BSC57238.2023.10201458.

[7] P. Ohri and S. G. Neogi, "Software-Defined Networking Security Challenges and Solutions : A Software-Defined Networking Security Challenges and Solutions : A Comprehensive Survey," no. June, 2023, doi: 10.12785/ijcds/120131.

[8] M. S. Tok and M. Demirci, "Security analysis of SDN controller-based DHCP services and attack mitigation with," *Comput. Secur.*, vol. 109, p. 102394, 2021, doi: 10.1016/j.cose.2021.102394.

[9] J. A. Rahim, R. Nordin, and O. A. Amodu, "Open-Source Software Defined Networking Controllers: State-of-the-Art, Challenges and Solutions for Future Network Providers," *Comput. Mater. Contin.*, vol. 80, no. 1, pp. 747–800, 2024, doi: 10.32604/cmc.2024.047009.

[10] A. Singh, N. Kaur, and H. Kaur, "Extensive performance analysis of OpenDayLight (ODL) and Open Network Operating System (ONOS) SDN controllers," *Microprocess. Microsyst.*, vol. 95, no. March, p. 104715, 2022, doi: 10.1016/j.micpro.2022.104715.

[11] S. Ahmad and A. H. Mir, *Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers*, vol. 29, no. 1. Springer US, 2021. doi: 10.1007/s10922-020-09575-4.

[12] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "SDN controllers: A comparative study," *Proc. 18th Mediterr. Electrotech. Conf. Intell. Effic. Technol. Serv. Citizen, MELECON 2016*, no. October 2017, 2016, doi: 10.1109/MELCON.2016.7495430.