Physical Communication 55 (2022) 101902

Contents lists available at ScienceDirect

Physical Communication

journal homepage: www.elsevier.com/locate/phycom

Full length article

HyPASS: Design of hybrid-SDN prevention of attacks of source spoofing with host discovery and address validation

Ramesh Chand Meena [a], Surbhi Bhatia [b], Rutvij H. Jhaveri [c,*], Long Cheng [d], Ankit Kumar [e,*], Arwa Mashat [f]

[a] Technology Parks of India, Ministry Electronics and Information Technology, Govt of India, Jaipur, India
[b] Department of Information systems, college of computer science and information technology, King Faisal University, Al Hasa, Saudi Arabia
[c] Department of Computer Science and Engineering, School of Technology, Pandit Deendayal Energy University, India
[d] School of Control and Computer Engineering, North China Electric Power University in Beijing, China
[e] Department of Computer Engineering and Applications, GLA University, Mathura, India
[f] Faculty of Computing and Information Technology, King Abdulaziz University, Rabigh, Saudi Arabia

ARTICLE INFO

Article history:
Received 14 July 2022
Received in revised form 11 September 2022
Accepted 21 September 2022
Available online 26 September 2022

Keywords:
Software-defined networking
OpenFlow
Secure SDN
Spoofing attack
Source spoofing

ABSTRACT

SDN (Software-Defined Networking) is a new technology that separates data and control planes; the main components of SDN are OFSwitches and Controller. The traffic flow is monitored by the SDN controller. Initially, OFSwitches lack security rules for packet handling. OFSwitch sends the packet to the controller for examination, creating control messages that favor the packet and establishing necessary flow entry. Host packets are sent to their destination, seeing only the destination host address and not the source host address. The attacker takes advantage of this situation and generates packets with forged source addresses in order to conceal his identity and perform various source address spoofed attacks such as Denial of Service (DoS), man in the middle (MiM), Distributed DoS (DDoS), and so on. This paper proposes a design for discovering hosts proactively, preparing HostTable, configuring flow entry during handshaking, and detecting and preventing source-forged attacks in Hybrid SDN. We called it HyPASS: Design of Hybrid-SDN Prevention of Source Spoofing Attacks with Host Discovery and Address Validation. We used Python for Mininet implementation and tested it on RYU and POX controllers. During the experiment, it identifies and drops 99.99% of packets with the forged source address.

1. Introduction

The source host address and filtering of IP spoofed attacks in a network can be checked using Access Control Lists (ACLs), but ACLs' nature is complex. They may conflict with active safety measures and policies in the network. SAVSH [1] and ISASA [2] for SDN to validate source address are proposed with the minor employment of OpenFlow protocol enabled switches. We establish that topology conversion of SAVSH was restricted to inter-links among forwarding appliances. Connected host information is not part of the sink tree, and alterations in host states are not reflected in it. Authors apply IP prefixes in filtering rules, and the IP prefix scanning approach provides a chance to attackers for IP spoofing attacks inside the network through other hosts' IPs of the same network. In PacketChecker [3] and in DosDefender [4], authors have used packet_in messages to assert host details used IP and media access control (MAC) to filter address spoofed packets at switch port level.

In INSPECTOR [5], authors also used packet_in messages to extract details of the source host, and a unique device, INSPECTOR, has been used to identify suspicious packets and filter such packets. Authors of the SDNSAVI [6] approach forwarded address assignment messages (AAM) to the controller by setting up essential flow entries into the OpenFlowSwitch table. The controller changes anchoring table information and creates flow entries into OpenFlowSwitch. The system has applied Switch PortID and a host IP address for binding and filtering packets. An attacker may use AAM packets to misguide binding table updation. Another issue faced during the study was that SDNSAVI used the Switch port and host IP address to bind and verify the source host address. Source host complete identity is MAC and IP address and only authenticity of host IP address may provide lower accurate level. The binding between Switch PortID and source host IP and MAC addresses provides better filtration.

In [7] proposed a technique for verifying packets source address as SIPAVSDN. The system checks the validity of packet's

* Corresponding authors.
E-mail addresses: rutvij.jhaveri@sot.pdpu.ac.in (R.H. Jhaveri), iiita.ankit@gmail.com, kumar.ankit@gla.ac.in (A. Kumar).

https://doi.org/10.1016/j.phycom.2022.101902
1874-4907/

source address and forwards the packets with legal source to their destination. SIPAVSDN uses ARP packets to identify host information and set up flow entry using binding among Switch PortId, Source host IP and MAC. Since technique utilizes ARP(Address resolution Protocol) packets to extract source host details, so an attacker can generate source address spoofed ARP packets to change the behavior of the technique and he can perform variety of IP spoofing attacks. The separation of the control plane and data plane in SDN offers several benefits, simplifying control and management of the network. On the other hand, due to the budget limitations and downtime in the network, the enterprises are unwilling to deploy SDN fully. Partial deployment of SDN [8] by installing a confined number of SDN equipment and traditional (legacy) network equipment creates a hybrid SDN network. The hybrid SDN network gives several benefits of SDN and has many usages. The SDN is gaining immense achievement and simultaneously opening new doors for safety risks. In SDNś working process, a packet_in event is raised with packet details to the controller when the packet is not matching any flow entry of the OFSwitch. Then, the controller processes the packet usually and forwards it to OpenFlowSwitch using a packet_out message. This process will consume the CPU and limited resources of OpenFlowSwitch and Controller, the bandwidth for data and control planes. This situation creates a bottleneck for the network and the possibility of DOS attacks.

1.1. Motivation

The controller in an SDN network tracks data packets and updates the flow table of the OpenFlow forwarding device as needed. Because SDN networks lack defense and control capabilities, they cannot forward packets from source hosts. Data packets are forwarded from source hosts to a controller, which uses the information to construct control packets and add new flows to the flow table. In this case, an attacker launches a source IP spoofing attack in order to carry out a different type of attack. There are numerous safeguards in place to prevent or mitigate the impact of cyberattacks. As this study has shown, current solutions for Source Address Validation (SAV) in the SDN environment have a number of drawbacks and gaps. This is why we believe it is critical to provide a solution for validating origin-host addresses in an SDN environment.

1.2. Major contribution

In this paper, we present work on SDN-based DOS attacks that generate a large number of malicious packets that do not correspond to any OFSwitch flow entry. These matchless packets are typically created by an attacker using a forged origin IP/MAC address in order to steal valuable data, spread malware, circumvent security controls, modify data-in-transit, disable network services/resources, and hijack the session or man-in-the-middle attacks. Such attacks are carried out by rogue hosts, and we must identify such rogue hosts and drop suspicious traffic at the switch port to mitigate such attacks. The following issues need to be resolved for the prevention of such attacks:

- Proactive discovery of hosts connected with the network.
- Automatically identification of the Address-Spoofed packets and
- Prevention of attacks of AddressSpoofing in real-time.
- We have managed the HostLink table at the Controller level to store Host MAC, SDN switch ID and Port ID.
- We have loaded essential flow entries into the OpenFlow switch flow table before actual data packet generation by a host.

Our study observed that no one approach provides an effective solution for the proactive discovery of connected hosts, identifying address-spoofed traffic, and preventing such attacks in real-time. The situation cannot be accomplished without detecting connected hosts and their necessary information like linked switch ports, IP, and MAC addresses. Most approaches do not know these details before the host starts forwarding packets. Here, we present HybridSDN prevention of attacks of Source Spoofing with host discovery and address validation. It discovers the connected hosts proactively, identifies the malicious traffic and verifies the source address, and stops address spoofing attacks at the switch port level. The remaining paper is organized into sections. In Section 2, we highlight related works. Section 3 mentions the new HyPASS Technique. In Section 4, we describe the simulation and testing of HyPASS. Section 5 discusses the measures of issues in HyPASS. Finally, Section 6 concludes and suggests future works.

2. Background and related works

After a fixed period [9], the network device sends Link Layer Discovery Protocol (LLDP) messages to advertise its identity and properties to its immediate neighbors. For sending multicast MAC address at [01:80:C2:00:00:0E] of a bridge, LLDP message is outlined using Ethertype field value 0x88cc. The steps described in [9,10] for LLDP message generation are used to discover the network topology, but it only identifies connecting links between switches and does not find details about connected hosts. All SDN controllers use the primary host detection process. The OFSwitches flow entry table miss forces the SDN forwarding device to forward the packet to the controller. When a host generates and sends IP or ARP packets and an OFSwitch does not have any accessible flow rules for the packets, the packet is forwarded to the controller. By extracting host information from the received packet, the controller completes host identification.

2.1. Topology detection in SDN

Network devices use single-hop LLDP to send their properties and identity to other devices in wired Local Area Networks. Presently various controllers have been using OpenFlow Discovery Protocol (OFDP), and LLDP packet and researchers are working hard to make the link discovery process more secure and efficient. In SwitchAgent [11], the authors offer a system where the controller produces and forwards TDPRequest multicast messages. After receiving such messages, the switches shift to either the Active or Father node from the Standby node. Every node asserts neighbor details, and only Father Node sends the details to the controller asynchronously. In ForCES [12], authors offer computational powers for running LLDP at the switch level to discover links. The controller periodically asks to collect the topology details. In SHTD [13], the authors offer an L2 topology detection and selffault healing protocol. The controller sends a topo Request a multicast message to detect topology. The spreading of such message has four roles in nodes like non-discovered, core, vleaf, or leaf and for states in ports like standby, pruned, child, or parent, respectively, to discover the topology. Automatic supervisor discovers status of port and updates managed elements for selffault healing. In SDNRDP [14], the authors offer a scattered resource detection protocol. In this approach, various switches are managed by two or more controllers. It has two work phases such as the announcement of controllers announce and joining of switches. The packets move at every event in different entities of the network for the formation of topology. In OFDP_HMAC [15], the authors introduce a dynamic HMAC

Table 1
Topology discovery approaches.

Approach	Packets/Messages	Packet type	LLDP Broadcast					
			Integrity	Authentication	Poison	Replay	Flood	
ForCES [12]	Periodic controller queries	LLDP						Y
Switch Agent [11]	Multicast messages	LLDP						Y
SPHINX [17]	Packet_in & features_replies	LLDP			Y			Y
SDN-RDP [14]	Controller announcement & joining switches	LLDP						Y
SHTD [13]	topoRequest messages	LLDP						Y
Literature [19]	Each switch one message	LLDP						Y
TopoGuard [18]	LLDP with HMAC TLV	Modified LLDP	Y		Y			Y
OFDP_HMAC [15]	Dynamic key in HMAC	Modified LLDP	Y	Y	Y	Y		Y
ESLD [16]	Port Classification & LLDP	LLDP	Y		Y	Y		
SLDP [20]	Host information	Modified LLDP		Y	Y	Y	Y	
TILAK [21]	Host information	Modified LLDP		Y	Y	Y	Y	
TrustTopo [22]	Dynamic password & chaotic model	Modified LLDP	Y	Y	Y	Y	Y	

verification for ensuring authentication and integrity. The computed HMAC is included in every packet of LLDP and utilized to protect from the attacks of LLDP Replay/ Poison. In ESLD [16], authors secure the topology detection and offer various stages like classifying a port, forwarding LLDP messages, and interlink discovery. In SPHINX [17], authors utilize abstracted flow graphs using features_replay and packet_in messages, and these graphs are used to verify all changes in the network. The authors make use of obfuscated flow graphs by utilizing features replay and packet in messages, and these graphs are then put to use to validate all of the changes made to the network.

In TopoGuard [18], authors classify failure of integrity verification and origin validation of packets based on threats related to LLDP packet and probable grounds. The controller also stops the forwarding of LLDP packets to any connected host. HMAC TLV is added at the end of the LLDP packet to prove the packetś integrity and source validity. In research [19], researchers propose an exclusive variation for topology detection using LLDP packets by generating only one for every connected switch rather than every port of the switches. In this approach, the controller generates a significantly less number of LLDP packets for the switch. In SLDP [20], the authors propose using host details to discover connected ports and all links with less LLDP packets generation. They used the MD5based authentication key in each LLDP packet to protect from attacks of poison and replay. In TILAK [21], the authors offer a token-based protective method for threats of topology detection. It provides a solution for integrity and authentication checks in every packet of LLDP to prevent several safety threats related to such packets. In TrustTopo [22], the authors offer a checking technique for host location through asynchronous rollback and path tracking methods to cope with host hijacking threads in LLDP packets. It also verifies link details with the dynamic password creation and chaotic model to prevent link fabrication attacks. Table 1 presents a summary of such approaches.

According to research [9], DHCP and ARP packets are sent to the controller via PacketIn to extract host details and perform host discovery actions. According to the paper, LLDP frames are used in the OpenFlow Discovery protocol to teach controllers about connected network nodes. This protocol does not recommend any method of discovering connected hosts before they generate traffic. In their study [9], the authors proposed a method for discovering connected hosts by sending ARP-requests via PacketOut control messages at the controller level. When the switch completes session creation with the controller, the ARP requests are sent to a switch port with a broadcast MAC and random IP destination addresses. If a host with this random IP address responds, the request is ignored. This method sends the ARP response to the controller, which extracts the necessary host information from the response. The controller now has full network topology information, including network equipment and connected hosts via LLDP and the host discovery approach [9]. When OFSwitch is coupled with an operational controller and changes on a switch port, the authors propose a working technique. However, it does not work in two situations: first, when we start the switch before the controller and make changes on an OFSwitch port after disconnecting and reconnecting a switch from the controller. The authors propose implementing it in the PortStatus message. This method also broadcasts ARP requests throughout the network, clogging networking throughput.

Detection of the host through LLDP protocol requires installation and configuration of protocol at the host level. We know that dissimilar individuals control network devices and hosts; therefore, implementing LLDP at the host is a more tedious and repetitive action.

The LLDP packets have few associated threads, as discussed above. Our technique discovers desired information related to connected hosts. It maintains a HostTable by generating and forwarding specialARP (SARP) packets with hashed MAC for every packet's validation and security. Section 3.5 describes the host discovery process.

Detects connected host at the time of handshaking of OpenFlowSwitch with controller by generating and forwarding ARP-Request messages. Extracts host details from ARP-Reply messages forwarded by connected hosts. Extracted host details like host IP, MAC and connected OpenFlowSwitch ID & its port ID from ARP packets are updated into HostTable after checking in HostTable. Now it Creates and manages flow entries into flow table of OpenFlowSwitch for forwarding of host packets to its destination when it receives new host details from HostTable Manager.

2.2. Source address spoofing (SAS) prevention techniques

In [25], researchers used access control lists (ACLs) to check and filter source address spoofing attacks in network setups. The nature of ACLs is complex, and existing safety rules in a network may conflict with them. Technique [26] provides address verification based on a first-come, first-serve (FCFS) for the IPv6 environment to match the ingress filtering method to find and mitigate SAS attacks. It allows holding an address by a host on an FCFS basis until the following change happens. It anchors the host address seen in the first packets of the host for binding with the switch port. FCFS-SAVI-DB stores the host IP, Anchoring, Status, Lifetime, Generation Time Stamp, Interface and Prefix related information. The authors mentioned different methods to discover on-link prefixes, traffic local & transit. The approach bound IP and MAC with a switch port for local traffic. In [27], the authors have examined the effectiveness and processing of resource utilization during SAVI implementation. In DHCP-SAVI [28] and FCFS [26], the authors offer protection at first hop. In IPv6 setup, network equipment & Neighbor Discovery (ND) tracking is used to form

Table 2
Topology discovery approaches.

Parameter	IP support	Messages used	Complexity	Binding/Filter parameter	Filtering accuracy	Dynamic network	Validation at	Development cost
SAVSH [1]	4	SNMP packets	Moderate	Network IP-Prefixes	Less	S	Network	C
SDN-SAVI [6]	4	AAM packets	Simple	IP & Switch Port	Moderate	S	Switch Port	C
SIPAV-SDN [7]	4	ARP packets	Simple	IP, MAC & Switch Port	High	S	Switch Port	C
ISAVA [2]	4	SNMP packets	Moderate	Network IP-Prefixes/ IGuarantee Header and encryption	Less	S	Network	C
PacketChecker [3]	4	Packet_In messages	Moderate	MAC	High	S	Switch Port	C
INSPECTOR [5]	4	Packet_In messages	Moderate and IP, MAC	Moderate	S	Special Device	C and Inspector	
DosDefender [4]	4	Packet_In messages	Moderate	IP, MAC	High	S	Switch Port	C
Literature [23]	4	ARP messages	Moderate	MAC, IP	High	S	Switch Port	C
Literature [24]	4	TTL Value	Moderate	MAC, IP	High	S	Host Level	C
HyPASS	4	Special ARP packets	Simple	MAC, IP	High	S	Controller and Switch Port	C

a binding table and the same is utilized to drop address spoofed traffic.

The SAS techniques can be categorized into two groups. In the first group, traditional networks' techniques do not apply to new technologies like SDN. Such techniques as called non-SDN. Further, we sub-divide them into two categories, Protocol Redesign/Encryption techniques and Source Address Filtering techniques. The techniques SPM [29], SEND SAVI [30], SANE [31] and Passport [32] are in Protocol Redesign/Encryption sub-category. The techniques FCFS SAVI [26], DHCP SAVI [28], ACL [25], Ingress/Egress Filtering [33], DPM [34]/Traceback/iTrace [35] and uRPF [36] are in Source IP Address Filtering sub-category.

The techniques, which support the SDN environment, are in the second group. The highest suppression of spoofing attacks in SAVSH [1] and ISAVA [2] is proposed by adding an inter-domain element. These have various modules such as Checkpoint Selection, Topology Detection, and Filtering Rule Generation. Topology translation into the SinkTree approach is restricted to detecting interconnection between forwarding network appliances. The information about the host and change in its status is not part of SinkTree. It uses prefixes of IP for framing packet filtering rules. ISASA inserts a verification header into a packet which increases the size of the packet. The size may exceed the maximum transmission unit, which is not allowed in packet transmission. The IP prefix filtering mechanism issue is that an attacker can use another hostś IP of the same network to perform IP spoofed attacks. In SDNSAVI [6], authors use the hostś IP address and Switch PortID for anchoring and validating source IP. Anchoring among OFSwitch PortID, source hostIP and hostMAC provides more excellent filtering. In SIPAVSDN [7], authors suggest anchoring HostTable with hostMAC, hostIP and SwitchPortID for defining required packet filtering & security policies based on host details. It uses the hostś first ARP packet to extract hostMAC, hostIP, SwitchPortID, and other required information. In PacketChecker [3] and DosDefender [4], authors have used packet_in messages to assert host details and IP and MAC to filter address spoofed packets at switch port level. In INSPECTOR [5], authors use PacketIn messages to extract the source hostś information and a particular device INSPECTOR to identify and filter the suspicious packets.

In HyPASS, we detect details about the host during the switch's handshaking with the controller using Special ARP (SARP) messages and create a source validation mechanism before transferring the host's actual traffic. It completely supports Hybrid SDN networks. It does not recommend any specific device or change at the host or switch level. As a result, it is simple to implement. Table 2 summarizes some of these SDN techniques.

2.3. Attack vector

The operation of the SDN network is represented in Fig. 1. In the attack model scenario, the H1 host will send a packet to the H2 host so that it may be checked. In the scenario, H1 is linked to the switch labeled SW1, H2 is connected to the switch labeled SW2, and direct communication between the switches SW1 and SW2. To provide controlplane support, Controller C0 has safe connections with Switches 1 and 2, and Table 3 outlines the specifics of the notation that was used. H1 intends to send a packet to H2 with the source and target addresses across the switch ports of SW1 and SW2. Whenever it receives a packet, the SW1 switch compares the values of the packet fields to the entries in its flow table. If any flow entry matches the value of the packetś fields, it acts according to the matched flow entry action; otherwise, it creates a flow miss event.

The value of the packetś fields is used to determine whether or not a flow entry matches the value. The flow miss event notifies the controller that the packet was successfully delivered to it through the PacketIn message. The first packet is always sent to the controller because the SDN switches do not initially contain matching flow entries relating to the host packets. This ensures that the controller is constantly aware of what is happening. This procedure is shown graphically by arrows drawn in red throughout the illustration. The controller is responsible for the extraction of packet field values, comparing those values with the packet forwarding policy, modifying the network topology (if necessary), and generating any necessary control and data packets through PacketOut messages. This procedure is represented graphically in the picture by arrows in green. The switches immediately begin transmitting the subsequent packets following the flow entries. This procedure, depicted by the black arrow lines in Fig. 1, is carried out at every SDN switch entry point for every packet hosts send.

This article has taken three states of the SDN system to describe the address spoofing attacks. The first state is entirely SDN setup, and all forwarding devices are OpenFlow enabled. The second and third states are HybridSDN setups and have legacy forwarding device which does not support OpenFlow protocol.

Address-spoofing State1: Fig. 2 shows addressspoofing state-1. It is a fully SDN setup where H1, H2, H3 and H4 use their IP and MAC addresses and are connected with OFSwitches SW1 and SW2. H2 sends a packet with SpoofedIP to H3, and the network setup will follow the procedure mentioned above and displayed in Fig. 1. In a typical SDN network, the forwarding devices do not

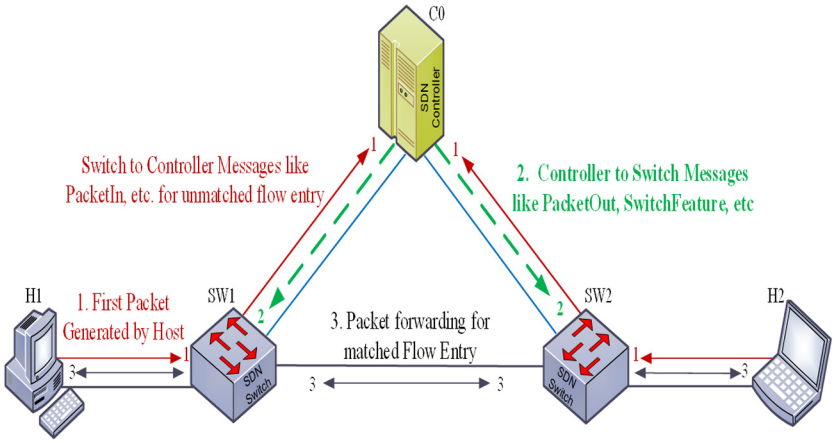

Fig. 1. Packet movement in SDN.

Table 3
Notations for network states.

Notation	Details/Values
C0	SDN Controller
SW1	OpenFlow Switch
SW2	OpenFlow Switch
SW3	Legacy Switch
H1	Host (IP=10.0.0.1, MAC=00:00:00:00:00:01)
H2	Host (IP=10.0.0.2, MAC=00:00:00:00:00:02)
H3	Host (IP=10.0.0.3, MAC=00:00:00:00:00:03)
H4	Host (IP=10.0.0.4, MAC=00:00:00:00:00:04)
!	Attacker Host
SpoofedIP	Spoofed IP address (IP=10.0.1.1)
SpoofedMAC	Spoofed MAC address (MAC=01:01:01: 01:01:01)

check the source host address before forwarding the packet to the destination. A spoof IP packet is sent to the controller for it to generate the necessary control packets. Seeing the target host address, the packet is forwarded to its destination. The destination host attempts to respond to the packet by sending reply messages to the forged address. The attacker has thus completed the address spoofing attacks against the host connected to OFSwitch.

Address-spoofing State2: Fig. 3 shows addressspoofing state-2. It is a HybridSDN setup where H1, H2, H3 and H4 use their IP and MAC addresses. The H1 and H2 hosts are connected with OFSwitch SW1. The H3 and H4 hosts are connected with nonSDN switch SW3. H2 sends a packet with SpoofedIP to H3. The packet is forwarded to its destination seeing the target host address. The destination host tries to respond to the packet. The attacker completes the address spoofing attacks targeting the host connected with a nonSDN switch.

Addressspoofing State3: Fig. 3 shows addressspoofing state-3. It is also a Hybrid-SDN setup where H1, H2, H3 and H4 use their IP and MAC addresses. The H1 and H2 hosts are connected with OFSwitch SW1. The hosts H3 and H4 are connected with nonSDN switch SW3. The host H3 sends a packet with SpoofedIP to H1. The packet is forwarded to its destination seeing the target host address. The destination host tries to respond to the packet. The attacker completes the address spoofing attacks targeting the host connected with a nonSDN switch. The H3 attacker may send the network to the hosts connected with OpenFlow enabled or nonSDN switches. The attacker host may also use SpoofedMAC and SpoofedIP to increase the complexity of the attack.

The above address spoofing states also cover the ARPspoofing attacks used to perform the MiM attacks. In this scenario, the attacker host responds with a forged MAC to host and sniffs private data (see Fig. 4).

2.4. Effect of attacks

The generation of many malicious packets that do not coincide with any flow entry at the OFSwitch entry point is the primary cause of SDN-based DOS attacks. These mismatched packets with bogus origin IP/MAC addresses are sent by an attacker to steal data, spread malware, circumvent security, modify data-in-transit, occupy network resources, hijack the session, and perform man-in-the-middle attacks. The switch and controller receive a large number of packets as a result of the flooding. It wastes resources and clogs switch and controller resources like CPU power, data plane bandwidth, and control plane bandwidth. It also depletes the target host's buffers and resources.

3. HyPASS technique

This section discusses the motivations for the system design, its characteristics, various formats used in it, network characteristics, the design of HyPASS, and the functions of its different modules. Here, we also describe the event sequencer.

3.1. Motivations for HyPASS design

An address forged attack protection system can be regarded well if it detects connected hosts as quickly as possible; it secures the system against identified risks, consuming fewer CPU processing power and network bandwidth. HyPASS intends to expose the connected hosts and secure the network proactively and efficiently. The proposed system will protect the host from source address spoofing attacks by fulfilling the following design characteristics:

Proactive:	HyPASS must detect the hosts connected to ports of OFSwitches and non-OFSwitches. HyPASS must maintain the Host Binding Table and set up necessary flows before generating actual traffic in the network.
Secure:	HyPASS must determine malicious packets at the controller level. HyPASS should verify the Source Host Address of a malicious packet. HyPASS should prevent attacks of address spoofing in realtime.
Efficient:	HyPASS must save bandwidth and Controller CPU power.

We present enough facts that encourage us to intend a new protection system to address spoofing attacks for the hybrid SDN environment.

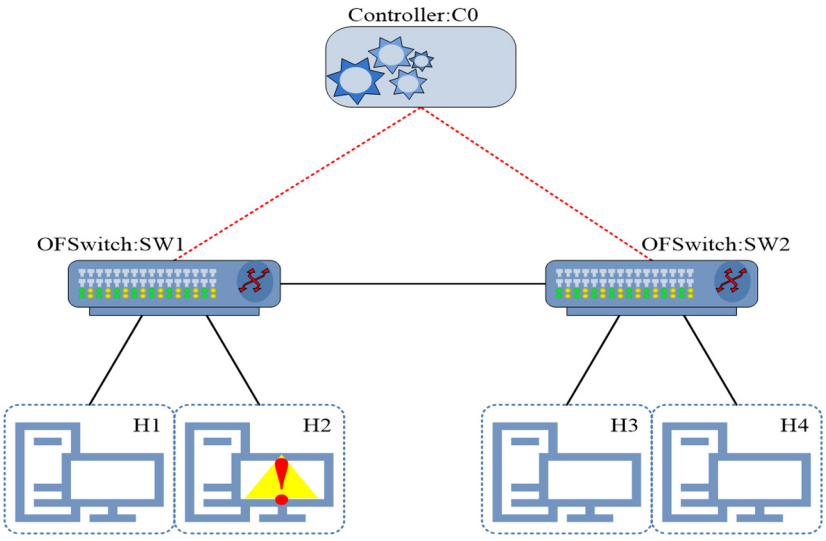

Fig. 2. Attacker with OpenFlow switch in fully SDN scenario.

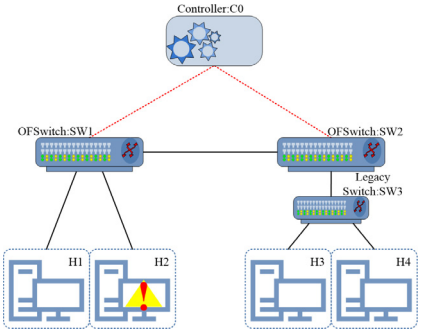

Fig. 3. Attacker with OpenFlow switch in hybrid SDN scenario.

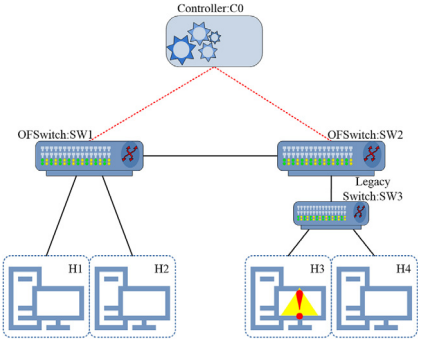

Fig. 4. Attacker with the non-SDN switch in hybrid SDN scenario.

Proactive: A proactive host discovery in SDN confirms the correct finding of host information with minimum resource utilization before generating actual traffic in the network. The existing host discovery solutions overuse the resources. This paper describes link discovery, host detection approaches, risks, and outcomes in the above sections' SDN environment. Our technique generates special-ARP (SARP) messages with hashed MAC to discover host information and maintains a HostTable It discovers the hosts at handshaking between the controller and OFSwitch. It also installs the necessary flow entry before starting actual traffic by hosts as soon as it detects the host. Our system proactively identifies the hosts, collects their details, and sets up the required flow entry.

Secure: The topology discovery approach [9] offers a technique to detect hosts by broadcasting ARP-Requests with random IP destination addresses through PacketOut control messages when handshaking of controller and switch. This approach is broadcasting ARP requests for each port and generating massive traffic in the network. So, it is a bandwidth-intensive approach, wasting the CPU power and creating an operational issue.

Our system uses special-ARP messages with hashed MAC addresses to discover hosts and collect their information. The hashed MAC is used to validate ARP-replay and extract host information. The system completes this process at handshaking and installs the necessary flow rules based on host details. We ensure the initial security for traffic flow control in this manner.

The OFSwitch forwards the malicious traffic to the controller due to a flow miss event. Our system uses HostTable to validate the source address of such packets. If HostTable does not confirm the packet, it is dropped, and a flow entry is created to drop subsequent packets at the port level. To ensure maximum security at the switch port level, such flow entries are configured using the idle timeout field for a limited time. As a result, all subsequent malicious packets are dropped at the port level, saving the system data, controlling plane bandwidth, and controlling CPU processing power.

Efficient: The system prevents addressing spoofing attacks and saves CPU power, data and control plane bandwidth on the controller. The first address spoofed packet is dropped by the controller, and subsequent packets are dropped by the switch port. As a result, the network will remain clean and free of malicious packets. According to the objectives stated in this article, the system operates in various SDN scenarios.

3.2. Formats used in design

Special-ARP (SARP) Packet: The proposed system uses SARP packets to discover hosts connected to the network. The system controller generates SARPRequest packets using Ethernet header and ARP header fields with values, as mentioned in Table 4. If any host's IP address matches with a random IP address of SARP-Request, then the host generates a reply with values shown in Table 5. Otherwise, the SARPRequest is ignored.

HostTable: Each host has its unique MAC address and IP in the network and is linked through a switch port. HostTable records binding details of host IPs, MACs, and switch ports. In [29,35], authors offer identical table to record HostIP and HostMAC.

OpenFlowSwitchID	OpenFlowSwitchPortID	HostMAC	HostIP
1	2	00:00:00:00:00:01	10.0.0.1
2	1	00:00:00:00:00:02	10.0.0.2
3	1	00:00:00:00:00:03	10.0.0.3

Fig. 5. HostTable format.

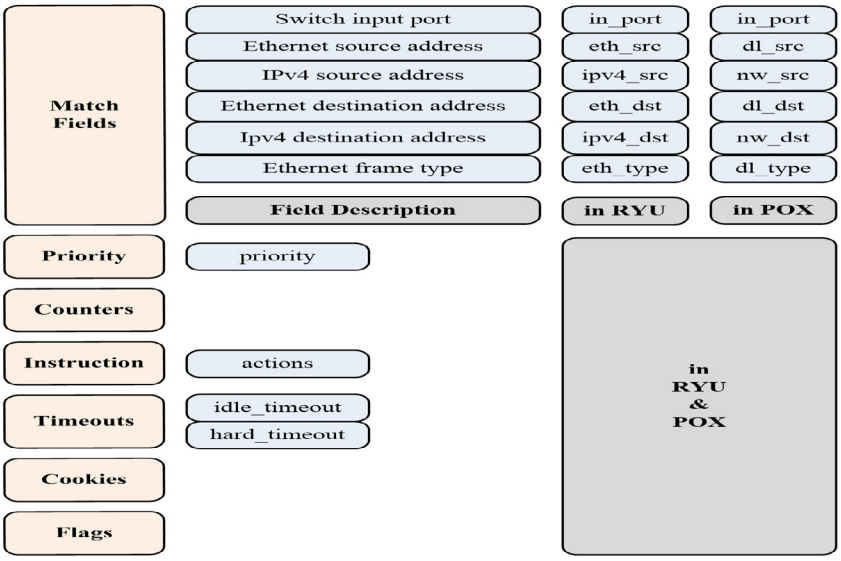

Fig. 6. Flow entry fields for RYU and POX controllers.

Table 4
Special ARP (SARP) packet format.

Ethernet	Field name		Field description	Packet field value	
	RYU	POX		Request	Reply
ARP	dst	Dst	Destination MAC address	ff:ff:ff:ff:ff:ff	Hashed MAC
	src	Src	Source MAC Address	Hashed MAC	Host MAC
	ethertype	Type	Ether Type (0x0806 for ARP)	0x0806	0x0806
	hwtype	Hwtype	Hardware type-(1 for Ethernet) Integer value	1	1
	proto	Prototype	Protocol address type-(0x0800) Integer value	0x0800	0x0800
	hlen	hwlen	MAC byte length-(6) Integer value	6	6
	plen	Protolen	IP byte length-(4) Integer value	4	4
	opcode	Opcode	Operation code(1 for Request & 2 for Reply)	1	2
	src_mac	Hwsrc	Source MAC Address	Hashed MAC	Host MAC
	src_ip	Protosrc	Source IP/Protocol Address	0.0.0.0	Host IP
	dst_mac	Hwdst	Destination MAC Address	ff:ff:ff:ff:ff:ff	Hashed MAC
	dst_ip	Protodst	Destination IP/Protocol Address	Random IP	0.0.0.0

Flow Entry Fields: Fig. 5 shows the flow entry fields used in the POX and RTU controller system. Table 5 shows the fields of various types of flow entry. The field idle_timeout with very low priority drops the suspected packets at the switch port level.

3.3. Network design characteristics

We implement the proposed design with RYU and POX Controllers. In this system, we categorize the ports of forwarding devices into a risky port (RP) and a safe port (SP).

Risky Port (RP): The port of an OpenFlow enabled switch connected directly with any hostuser is the risk-associated port. The attacker uses one or more host ports for generating and injecting malicious packets into a network. In Fig. 7, red linking lines show the RPs.

Safe Port (SP): The port of an OpenFlow enabled switch connected directly with other OpenFlow enabled forwarding devices is not a risk-associated port. The attacker host cannot use this port directly for generating and injecting malicious packets into a network. In Fig. 5, green connecting lines show SPs.

In a hybrid SDN network, legacy forwarding devices are also part of it. A port of OpenFlow enabled forwarding device is risky if any legacy forwarding device is connected with this. Most of the SDN-supported security techniques do not work with legacy devices. It indicates focusing on RPs.

3.4. HyPASS system design

Fig. 8 shows the fundamental modules and their task of HyPASS architecture. The source spoofed attacks prevention activity is a periodic operation. Every cycle needs to perform a few functions such as host detection and host table handling at switch handshaking and state modification of switch port, flow entry, source validation, and packet forwarder-dropper. In the below

Table 5
Flow entry format for ARP, IP and Malicious Packets.

Field description	Fields used for flow entry		
	ARP Packet	IP Packet	Malicious Packet
Switch input port	Yes	Yes	Yes
Ethernet source address	Yes	Yes	$\sqrt{Wildcard}$
IPv4 source address	Yes	Yes	$\sqrt{wildcard}$
Ethernet destination address	Yes	Yes	
IPv4 destination address	Yes	Yes	
Ethernet frame type	Yes	Yes	
Priority	Very High	Very High	Very Low
Actions	Out Port	Out Port	Drop
Idle_timeout			$\sqrt{IdleTime}$ i.e.,20ms

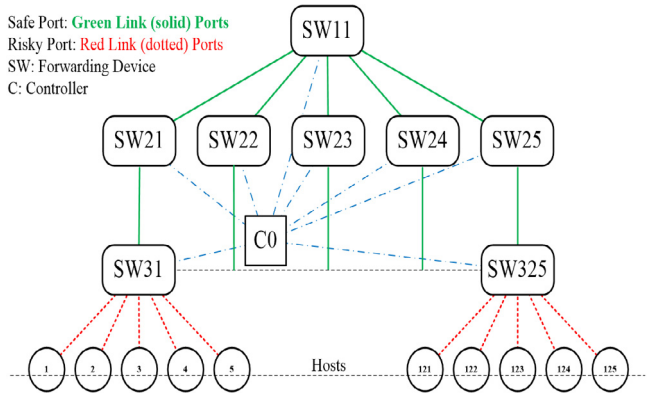

Fig. 7. Switch port state in SDN scenario.

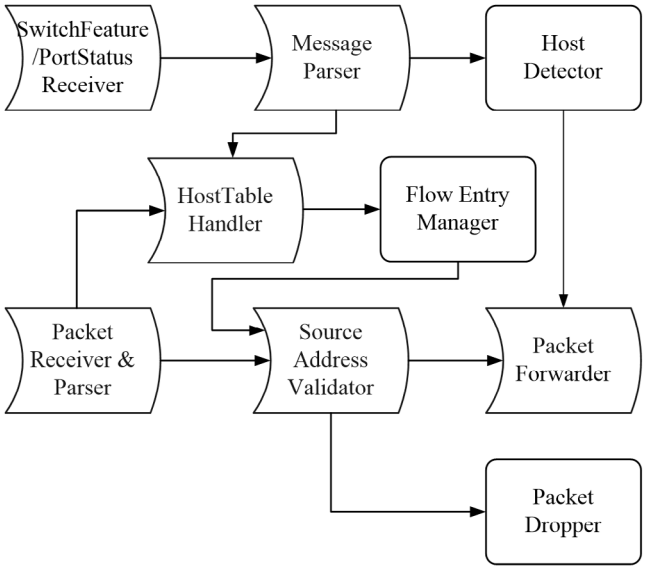

Fig. 8. HyPASS system design.

sub-sections, we describe tasks of different modules coupled in HyPASS design.

3.4.1. Host detector

The IDH-SDN system introduced in [[10] is used in this section and modified to meet the design requirements of HyPASS. SARP messages with hashed MAC addresses can be used to identify network hosts in an SDN system. The SARP format for POX and RYU controllers is listed in Table 5. This is how it works. HOSTDETECTOR of Algorithm 1 uses SwitchFeature and PortStatus events to find hostsínformation. The inputs are Switch ID and Port and the event message. SARPrequest messages are generated in setup due to the algorithm's output. When adding them to the network, we may use them to identify new hosts, switch ports, or switches. It also works when the switch port is in a different state. When a host is no longer connected to the network, HostDetector removes the hostś information from the HostTable.

During the first handshake with the switch, the controller sends out Special ARP Requests, which allows it to find out information about the connected hosts before the actual creation of traffic. Other methods of host detection involve either the creation of a new protocol or the analysis of the traffic that the hosts cause. The new protocol needs to be applied (i.e., host, switch and controller). It is responsible for maintaining the HostTable to retain a record of the critical data associated with the port.

Algorithm 1 Put your caption here

```
1: procedure HOSTDETECTOR
2:     Implementation at: SwitchFeatureEvent and PortStatusEvent
3:     Input: provide the value of SwitchID, SwitchPortID, ipRange
4:     if PortStatusNew or PortStatusModity or SwitchFeature then then
5:         while j=1 to ipRange do do
6:             Compute and create a SARP-Request messages contain as
7:             DstMAC = Broadcast MAC address,
8:             SrcMAC = Hashed MAC, SrcIP= "0.0.0.0"
9:             source address of SARP request packet.
10:        End the scan of switch port
11:    else if PortStatusDelete
12:        delete from HostTable with SwichID, SwitchPortID then
13:        FLOWENTRYMANAGER(SwichID,SwitchPortID,DELETE)
```

3.4.2. HostTable handler

After a generation of SARPRequest by HOSTDETECTOR, the host responds with SARPReply, whose details match SARPRequest. The forwarding device forwards the reply to the controller through the PacketIn event. HOSTTABLEHANDLER procedure extracts the required details of the host from the reply. It checks host details with HostTable for its existence. If details are not available in HostTable, update HostTable and forward to FLOWENTRYMANAGER to set up flow entry into OFSwitch. Fig. 5 shows the format of the HostTable It stores host information in fields like OpenFlowSwitchID, OpenFlowSwitchPortID, HostIP and HostMAC. We offer HOSTTABLEHANDLER procedure implementation at PacketIn event as defined in algorithm 2. It takes the SARPReply message of PacketIn as input.

3.4.3. Flow entry manager (FEM)

FEM's primary function in OFSwitch is to manage and configure flow entries. It generates the flow entry needed to forward the host's packets to the destination, with a focus on the RPs of OFSwitches. It also installs flow entry to drop malicious packets based on the results of the source-address validator module. To implement the system, we use POX and RYU controllers, and their field names in flow entry differ. Fig. 6 summarizes the field names and descriptions. The flow entry fields for various packets are listed in Table 5. When the idle timeout field expires, the system removes the flow entry automatically. When a flow is decommissioned, it frees up space in the flow table.

Algorithm 2 HostTable Handler

procedure HOSTTABLEHANDLER
Implementation at: Packet_In event
Input: ev packet_in/ PortStatusEvent event massage with SARP-Reply procedure HOSTTABLEHANDLER
extract SrcIP, DstIP, DstMAC, SrcMAC, SwitchID, SwitchPortID,PacketType from ev
    if SARPReply Message [] and Source details do not exists in HostTable **then**
        Insert into HostTable with SrcIP, SrcMAC, SwitchID, SwitchPortID
    else FLOWENTRYMANAGER(SrcIP, SrcMAC, SwitchID, SwitchPortID, INSERT)

Algorithm-3 defines the procedure FLOWENTRYMANAGER. The HOSTTABLEHANDLER, HOSTDETECTOR, and VALIDATEADDRESS modules carry out adding or removing flow entries from OFSwitch. The IP, MAC, Switch ID, Switch Port of the source host, and the reason for calling the module are the input parameters. FEM adds the flow entry with very high priority to forward the OutPort when the system identifies a new porthost (i.e., reason=INSERT) in the network. It deletes when the existing port/host is removed (i.e., reason=DELETE) from the net. It also eliminates the current flow entry and adds a new flow entry when porting/host status changes (i.e., reason=MODIFY). The system adds flow entry with very low priority, Idle_TimeOut, to drop the packets when the controller finds a malicious packet in the network. The network administrator decides the Idle_TimeOut as per security requirements like 20 ms, 30 ms, etc.

Algorithm 3 Flow Entry Manager

procedure FLOWENTRYMANAGER
    Implementation at: HostTable Updates and Address Validation
    Input: SrcIP, SrcMAC, SwichID, SwitchPortID, Reason
    if *Reason* == *INSERT* **then**
        add-flow entries into flow table using SwitchID, SwitchPortID, SrcMAC, SrcIP with Very High Priority and OutPort Action
    else if *Reason* == *DROP* **then**
        add-flow entries into flow table using SwitchID, SwitchPortID with Very Low Priority, Idle_TimeOut and Drop Action
    else if *Reason* == *DELETE* **then**
        deleteflow entries from flow table using SwitchID and SwitchPortID
    else if *Reason* == *MODIFY* **then**
        delete-flow entries from flow table using SwitchID, SwitchPortID
    else
        add-flow entries into flow table using SwitchID, SwitchPortID, SrcMAC, SrcIP with Very High Priority and OutPort Action

3.4.4. Source address validator

This module validates the packet's source host information with HostTable and performs source host address verification at the controller. If the source host information is validated, the packet is sent to the target switch port; otherwise, the controller drops it and configures flow-entry to drop subsequent packets at the switch port level. VALIDATEADDRESS is a module defined by Algorithm-4 and implemented at the PacketIn event. It receives the event message as an input. It compares the packet's source host and switches information to HostTable details. If the source host and switch details match, the packet is forwarded to a destination; otherwise, the packet is dropped. A flow entry is installed to drop subsequent such packets by calling the FLOWENTRYMANAGER procedure.

Algorithm 4 Put your caption here

procedure VALIDATEADDRESS
    System Initialization
    Read the value
    Implementation at: Packet_In event
    Input: ev ← packet_in event message
     SrcIP, SrcMAC, SwichID, SwitchPortID extract from ev
    if SrcIP, SrcMAC, SwichID, SwitchPortID do not exist in HostTable **then**
        FLOWENTRYMANAGER(SrcMAC, SwitchID, SwitchPortID, DROP)
        generate alert
        drop packet

3.5. Event sequence

Fig. 9 depicts the sequence of events to learn more about HyPASS. The three perpendicular lines represent the time for entities such as two switches SW1 and SW2 and one controller C0. C0 begins to function after SW1 begins to function. SW1 responds to C0's SwitchFeatureRequest, and C0 responds to SW1's SwitchFeatureRequest. Following that, C0 generates a hashed MAC and sends PacketOut to SW1 with a SARP-Request for a response. If the SARP-Request details match those of any SW1 connected host, the host responds with a SARP-Reply. SW1 sends the response to C0 via PacketIn. If the SARP-Request information does not match any SW1-associated host, it is ignored. C0 validates the source and target host details of a SARP-Reply and updates the source host details in the HostTable. C0 sends FlowMod to SW1 to configure flow entry with host information. SW1 installs flow entry in accordance with the FlowMod instructions.

Now, C0 sends SwitchFeatureRequest to SW2, and SW2 sends SwitchFeatureReply to C0. After that, C0 generates hashed MAC and sends PacketOut with SARPRequest to SW2 for a response. If SARPRequest details match any SW2 connected host, it responds with a SARPReply message. SW2 forwards the response to C0 through PacketIn. If SARPRequest information does not match any SW2 connected host, it is ignored. C0 checks the source and target hostś information of SARPReply and updates the source host details in HostTable. C0 sends FlowMod to SW2 to set up a flow entry with host details. SW2 installs flow entry as per FlowMod instructions.

The regular network traffic starts between SW1 and SW2. Now, SW1 and SW2 connected hosts may generate suspected or a new type of traffic/packet, raising flow miss message. On the generation of flow miss messages, the OFSwitch SW1SW2 forwards such packet to C0. The controller C0 verifies the packet's source host address with HostTable and identifies it as a suspicious or genuine packet. C0 forwards the authentic packet as per the standard procedure of the network. C0 drops the suspicious and malicious packet and sets flow entry into SW1 and SW2 to discard the subsequent packets at the OFSwitch port level. The prevention of source spoofed attack activity is a periodic operation. As shown in the event sequence diagram, most of the activities are repetitively executed.

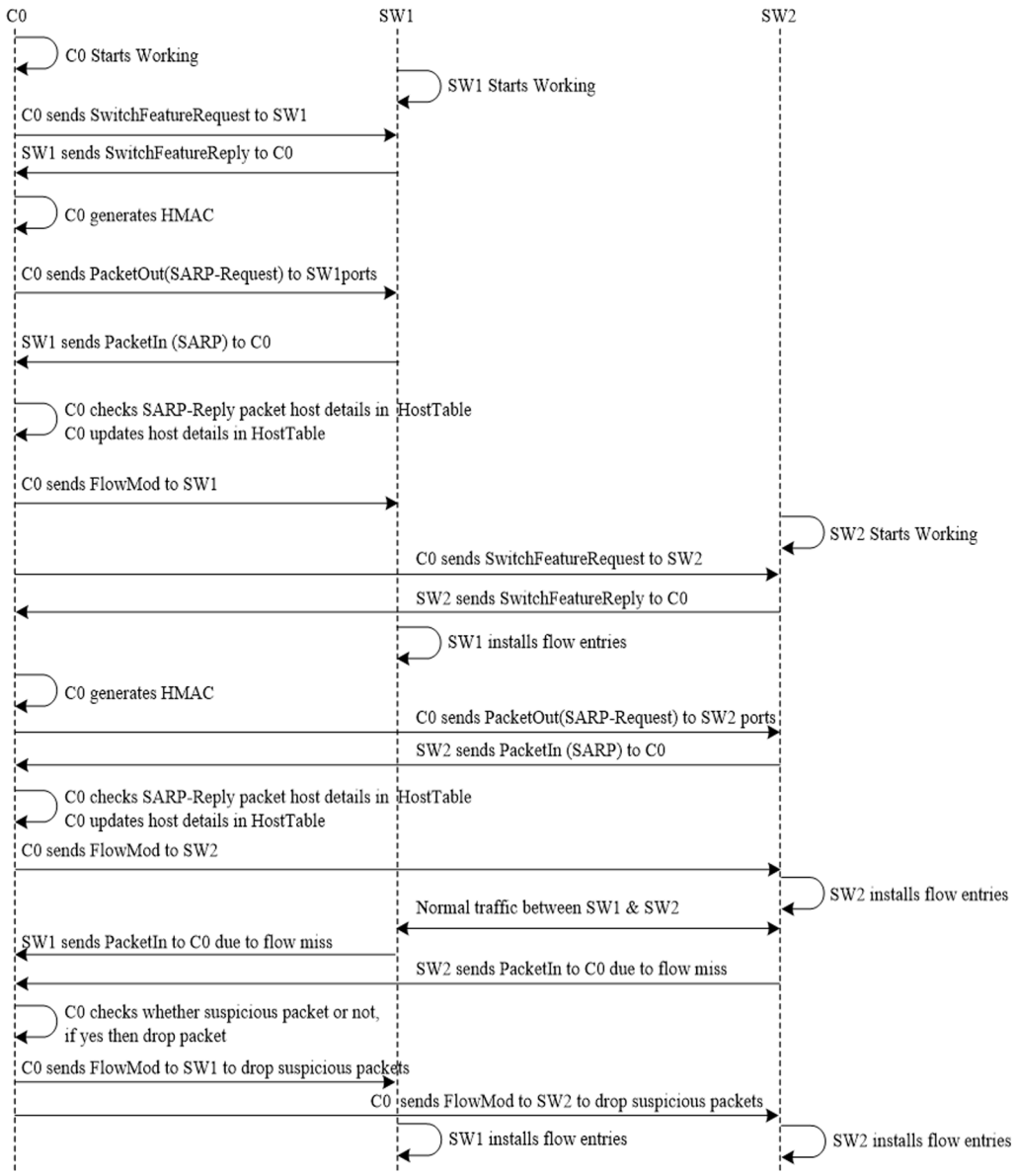

Fig. 9. HyPASS event sequence.

4. Hypass simulation

This section examines the proposed techniqueś correctness by experimenting and analyzing its results. Typologies and testing setups are described to validate the experiments. We carry out all the Mininet [37] network imitator tests using four different network scenarios given in Figs. 10–13. Table 6 presents the statistic of network scenarios. We believe that network scenarios, which have 121, 127, 31, and 10 switches with ports 483, 380, 185 and 48, respectively, are justifiable for the experiment. The network scenarios like 'Tree,5,3', 'Tree,7,2', 'Tree,3,5' and 'Hybrid' mentioned in Table 6 are from Mininet setup. We consider that HyPASS will also work with other network scenarios to support the data center and large corporate environments.

In Fig. 10, Tree,5,3 is designed with five layers of switches, and every OFSwitch is linked to six switches, excluding leaf and root OFSwitches. SW34-SW36 depicts three linked OFSwitches at layer three; SW5 and SW4 are OFSwitches at Layers five and four serially. Hosts are connected to level five OFSwitches. In Fig. 11, Tree,7,2 is designed with seven layers of OFSwitches, and every OFSwitch is linked to three switches, excluding leaf and root OFSwitches. SW33–SW34 depicts two linked OFSwitches at layer three, and SW4, SW5, SW6, and SW7 are OFSwitches at layers four, five, six, and seven serially. Hosts are connected to level seven OFSwitches. In Fig. 12, Tree,3,5 is designed with three layers of OFSwitches, and every OFSwitch at tier two is connected. SW21–SW25 depicts five linked OFSwitches at stratum two, and SW3 is OFSwitches at layer three. Hosts are linked to third-level OFSwitches.

In Fig. 13, the Hybrid network [8] scenario is designed with 8 OFSwitches and two legacy switches. It is a mixture of OFSwitches and NonOFSwitches, so we called it a Hybrid. The scenario is used to test the performance of HyPASS with NonSDN switches. In the scenario diagram, S1S2, S4S7, and S9S10 are OFSwitches, and S3 and S8 are NonOFSwitches. Every switch is connected with at least one OFSwitch. H1H30 represents linked hosts to switches at different levels. Table 7 summarizes the environment information for HyPASS validation, like details of hardware and software, SDN controllers, Software Library, Network Analyzer tools, and other resources. We use the Scapy software library to perform various attacks in testing.

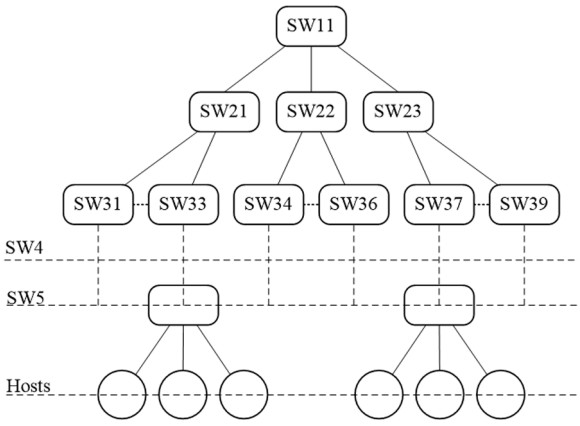

Fig. 10. Network Scenario 1: Tree,5,3.

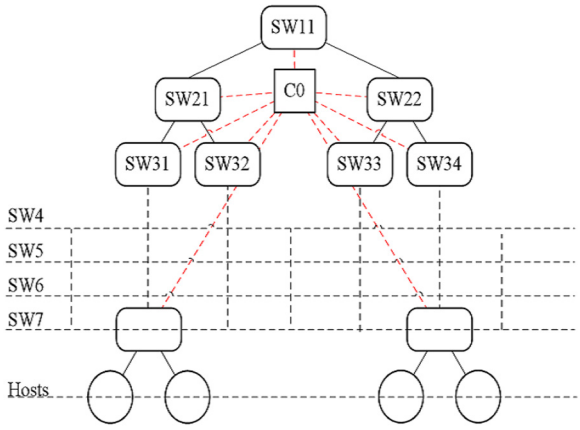

Fig. 11. Network Scenario 2: Tree,7,2.

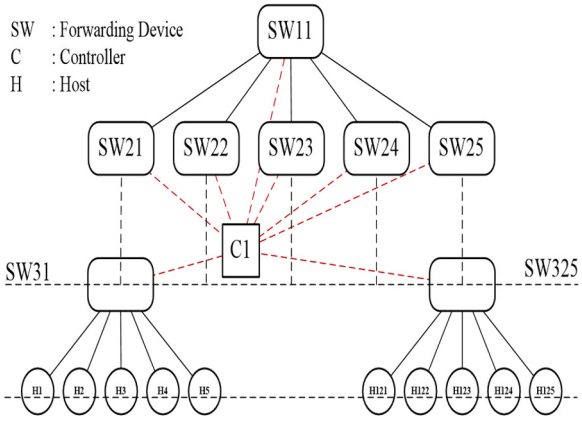

Fig. 12. Network Scenario 3: Tree,3,5.

Table 6
No of Switches, Hosts, Ports and Links.

Net Scenario	SW	Ports	Links	Hosts
Tree,5,3	121	483	363	243
Tree,7,2	127	380	254	128
Tree,3,5	31	185	155	125
Hybrid	10	48	39	30

Generation of outcomes in selfcreated setup forms doubts. In this work, we put the print() function at many required places

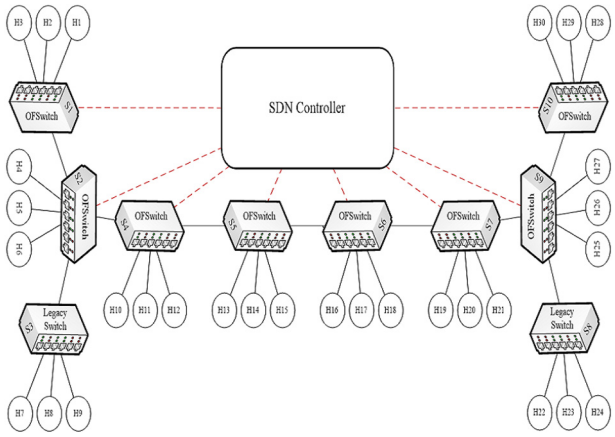

Fig. 13. Network Scenario 4: Hybrid.

Table 7
Test setup for HyPASS.

Device/Asset	Configuration
Network emulator	Mininet
OS of Attacker/Victim	Ubuntu-18.04LTS
Hardware of Attacker/Victim	i5-6200u 4 Core CPU & 4GB RAM
Controller	POX & RYU
OpenFlow Switch	V2.9.5
Programming language	Python 2.7.12
Network libraries	Scapy
Network analyzer	Wireshark
Network Scenarios	Tree,5,3, Tree,7,2, Tree,3,5 & Hybrid

in the python code to display outcomes in relative conditions instead of absolute. The same systemś implementation is regarded for comparing two controllers' outcomes, i.e., RYU and POX controllers. The controllerś choice for an experiment environment depends on features such as support of programming language, availability of documentation, updates and resource requirements of the controller. The RYU [38] and POX [39] are supporting python programming language; the ONOS [40], HPEVAN [41], FloodLight [42], and OpenDayLight [43] are supporting JAVA. Some controllers (i.e., RYU, FloodLight, and POX) are prepared for educational use, while few controllers (i.e., HPEVAN, ONOS, and OpenDayLight) are for industrial use. We use POX, and RYU controllers in the system test as both controllers are welldocumented, freely available, and support python programming [44,45].

HyPASS is a system for protecting source spoofing attacks, and it proactively discovers host information at handshaking and validates the address of suspicious packets. Fig. 8 depicts our system's modules to protect against source spoofing attacks proactively, efficiently, and securely. It prepares the environment and information for security; even its proactive detection of the host increases the period of handshaking. But it improves the forwarding time of the first actual packet of the host by setting flow entry proactively.

Host Detection: When HyPASS is being tested, it looks for and records information about linked hosts in the host table. Examples of data from the identified hosts are shown in Table 8. Wiresharkś output has been used to confirm the accuracy of the HostTable data. According to the results, HyPASS is discovering and updating the HostTable for all hosts [46,47].

Filtering Accuracy: To produce a variety of traffic, we utilize Scapy to forge addresses at a specific rate. SDN-specific security measures are implemented on OFSwitches, but not on traditional devices. (nonSDN Switches). FIGURE 14 depicts the accuracy of the system's filtering using POX and RYU controllers in four

Table 8
HostTable data sample of network scenario 4: Hybrid.

OpenFlow SwitchID	OpenFlow SwitchPortID	HostMAC	HostIP
	2	00:00:00: 00:00:01	10.0.0.1
S1	3	00:00:00: 00:00:02	10.0.0.2
	4	00:00:00: 00:00:03	10.0.0.3
....	....	....	....
....	....	....	....
	2	00:00:00: 00:00:1c	10.0.0.28
S10	3	00:00:00: 00:00:1d	10.0.0.29
	4	00:00:00: 00:00:1e	10.0.0.30

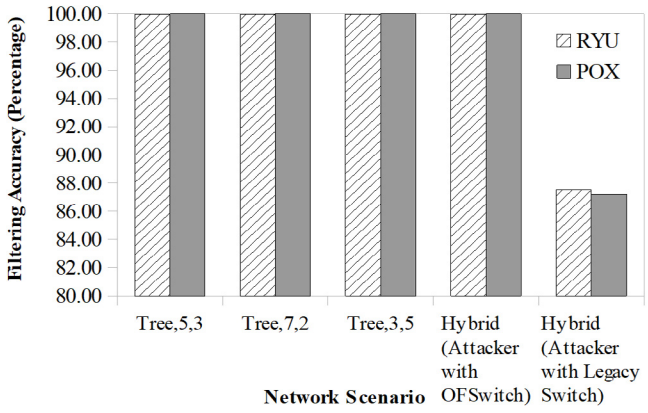

Fig. 14. Address spoofing attack filtering accuracy.

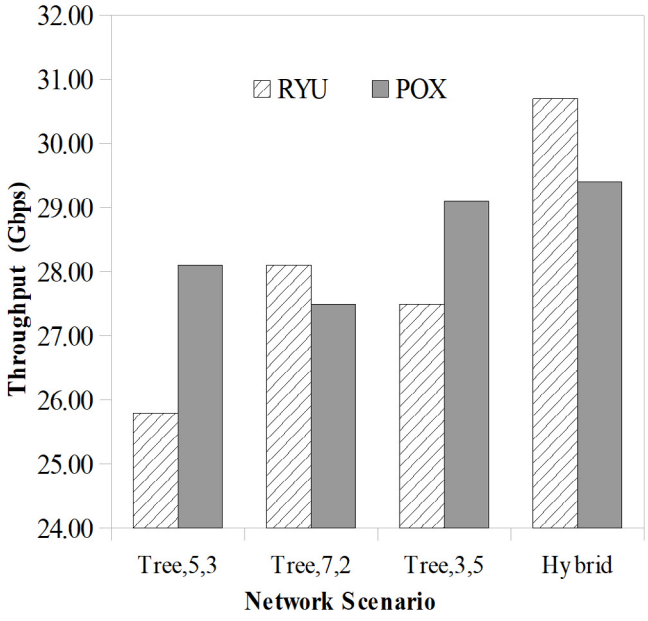

Fig. 15. Throughput in different network scenarios.

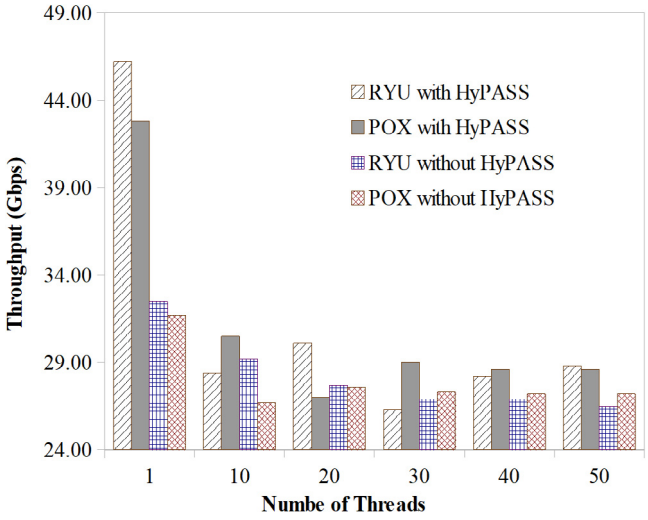

Fig. 16. Throughput with different number of threads.

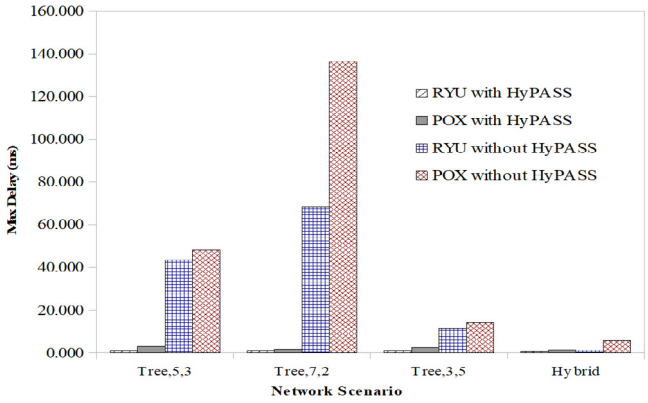

Fig. 17. End to end delay (maximum).

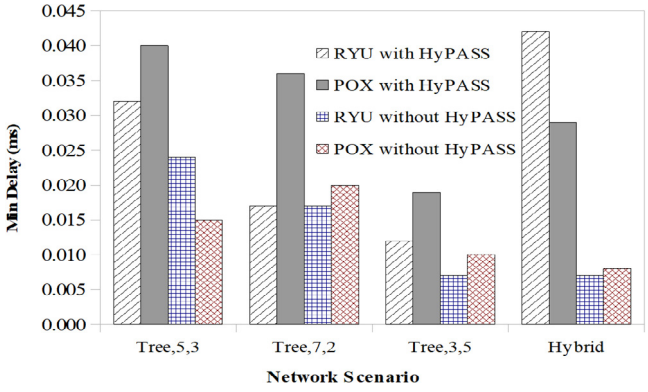

Fig. 18. End to end delay (minimum).

different network circumstances. There is a 99.999% chance that HyPASS's filtering accuracy with both controllers for 'Tree,5,3,7,2' is 99.999% accurate. Every single one of the OFSwitches may be found in these network configurations. All the hosts use OF-Switch's port. The 'Hybrid' network setup shown in Fig. 13 shows legacy switches S3 and S8. Because SDN security regulations do not apply to older switches, the accuracy of address forgery attack screening in hybrid topologies varies. The host's traffic linked to the legacy switch connected to the same switch is not controlled by the SDN controller [48]. The HyPASS firewall cannot identify and filter assaults originating from the same legacy switch as the victim hosts (examples: H8 and H9) (examples: H7 and H8 and H9). We identify and filter out fraudulent packets when the attacker, victim, or both are connected to OFSwitch in a hybrid topology. Fig. 14 depicts an OFSwitchconnected attacker, victim, or both, and the 99.99 percent filtering accuracy for a 'Hybrid' topology. Another factor to consider is the number of hosts linked to the legacy switch and the amount of faked packets sent by the attacker to the victim's computers connected to the switch. Our system has a filtering accuracy of above 80% in this mix mode test. Throughput: Fig. 15 indicates the actual data

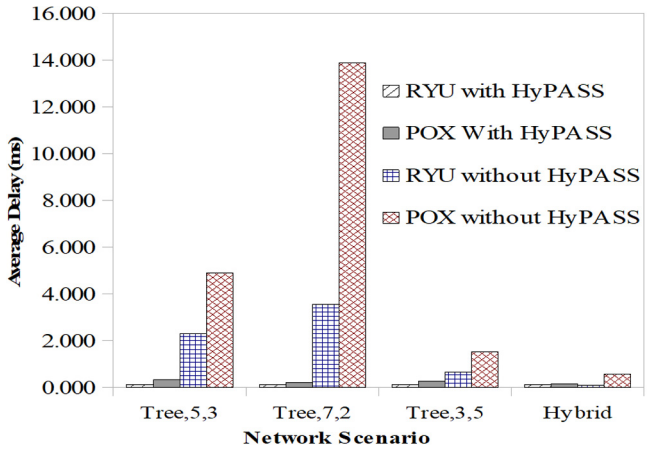

Fig. 19. End to end delay (average).

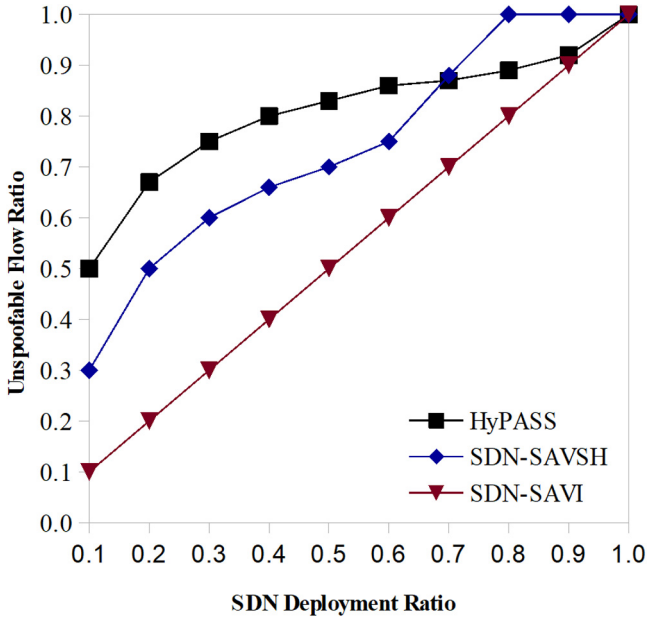

Fig. 20. SDN deployment and flow generation ratio.

transfer rate achieved after implementing HyPASS on POX and RYU controllers. We use the iPerf network utility for measuring the system throughput. The graph shows that the systemś throughput depends on the complexity of network topology. In 'Tree,5,3', 'Tree,7,2', 'Tree,3,5' and 'Hybrid', the POX controller provides throughput 28.10, 27.50, 29.10 and 29.40 Gbps respectively and the RYU controller provides throughput 25.80, 28.10, 27.50 and 30.70 Gbps respectively. The POX controller performs better in complex network scenarios, but RYU performs better in less complex networks [49,50].

Fig. 16 shows the throughput with POX and RYU controllers using concurrent threads for bandwidth transfer. We have used iPerf with Parallel option to make simultaneous links between 1 and 50. The more concurrent threads create sessions and occupy the resources (i.e., CPU, buffers) of the client and server, and the throughput goes down. In the parallel sessions, our system performs batter with RYUcontroller. We also have conducted an overhead analysis of HyPASS. The throughput test has been conducted without HyPASS, and Fig. 16 gives the improved performance of throughput of the network with HyPASS [51].

End-to-End Delay: We generate 1k packets to test end-to-end delay in its transmission. The system records the starting time and reaching time of transfer of a packet from source to destination. It subtracts starting time from reaching time to measure the time taken to traveling of a packet. Figs. 17 and 18 present the maximum and minimum time taken by a packet in transmission. Fig. 19 shows the average time packets travel of the duplicate size packets. Our system performs better with the RYU controller than the POX in end-to-end delay test results. However, without the HyPASS system, the test results show more delay in packets' delivery on the RYU and POX controllers [52].

Efficiency Comparison: Fig. 20 compares HyPASS with SDN-SAVSH [1] and SDN-SAVI [6] on the generation ratio of filtering flow for various deployment ratios of the OpenFlow network. We take the ratios of SDNSAVSH and SDNSAVI from research [1]. The graph depicts that HyPASS generates more flows to secure the networking with the binding of IP and MAC with OFSwitch Port than other approaches.

Packet Performance: In the RTT, the HyPASS reduces the first packet delivery time in testing. It is calculated by issuing an echo request until the destination host receives an echo reply message. The system achieved more than 83% time-saving in sending and processing the first packet with different network scenarios. Figs. 16, 17 and 19 prove that our system saves data and controls bandwidth and the controller CPU power, and minimizes the waiting time to deliver the first packet [53].

5. Discussion

SARP Message: The proposed system generates SARP messages in the network for each IP in the IP range at handshaking or status change of any OFSwitch port. It discovers the host details proactively. It generates lots of SARP messages. We have to minimize the propagation by verifying the target IP of SARP-Request with HostTable. The system does not generate SARP-request for the IP, which is available in the HostTable, and it saves data and controls plane bandwidth and CPU power.

Address Assignment: Static IP addressing strategies are used in the HyPASS discovery process. Static and Dynamic Host Configuration Protocols (DHCP) are supported by IPv4 technology. We recommend forwarding DHCP messages (i.e. Discover, Offer, Request and Ack) as a best practice. Data from the DHCPRequest is retrieved and stored in the binding HostTable by the system. This flow entry installs essential flow entries with host details for forwarding DHCPRequest, DHCP-Release and DHCPAck upon receipt of the IP detail in the HostTable of the linked MAC address.

Flow Table Size: HyPASS is binding the MAC and the IP of the source host with the switch port. It generates binding rules for source IP, MAC, and switch ports to prevent addressing spoofing attacks. However, this increases the number of entries in the flow table and can cross table size in an extensive network. We suggest using the multiple tables in OFSwitch. In this process, the first table is used for matching with source IP, MACand switch port of the packet. It directs to sub-tables for onward checking and processing of packets. It also removes the flow rule dependency and makes them separate.

6. Conclusion

In this paper, a new HyPASS design is proposed, which uses proactive host discovery and address validation to provide the protect against address-forgery attacks in SDN networks. Source Address- forgery attacks lot of packet which result as loss of network bandwidth, computing power, and other network resources that user are not able to access their resource. There is host recognition, address validation, and identification of the system's critical modules of malicious packets. We use POX and RYU controllers in the Mininet for latency, throughput, and attack

prevention testing. The accuracy and efficiency of the system are examined in four SDN scenarios classified as OpenFlow-enabled and Hybrid. At the controller handshaking, the HostTable is updated, and all address-forgery threats are prevented by verifying the source address in different SDN settings. Using an OpenFlow-enabled system, it achieves 99.999% filtering accuracy. Due to the use of older switches in a hybrid network, this might vary.We recommend releasing a solution that prevents address spoofing attacks without a hitch after testing in a real SDN environment. With this system, future modifications and testing of the IPv6 protocol and multiple concurrent controllers are possible.

Abbreviation

SDN: Software-Defined Network

DHCP: Dynamic Host Configuration Protocol

OFC: open Flow controller

SD-WAN: Software-Defined Wide Area Network

MPLS: Multiprotocol Label Switching

VPN: Virtual Private Network

QoE: Quality of Experience

CDN: Content Delivery Network

CRediT authorship contribution statement

Ramesh Chand Meena: Conceptualization, Project administration. **Surbhi Bhatia:** Formal analysis. **Rutvij H. Jhaveri:** Investigation. **Long Cheng:** Writing – review & editing. **Ankit Kumar:** Data curation, Methodology, Writing – original draft. **Arwa Mashat:** Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have shared the data in zip file, Python code is also available.

Acknowledgments

This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia [GRANT453]. All authors have read and agreed to the published version of the manuscript.

References

[1] G. Chen, G. Hu, Y. Jiang, C. Zhang, SAVSH: IP source address validation for SDN hybrid networks, in: 2016 IEEE Symposium on Computers and Communication, ISCC, 2016, pp. 409–414.
[2] C. Zhang, et al., Towards an SDN-Based Integrated architecture for mitigating IP spoofing attack, IEEE Access 6 (2017) 22764–22777.
[3] S. Deng, X. Gao, Z. Lu, X. Gao, Packet injection attack and its defense in software-defined networks, IEEE Trans. Inf. Forensics Secur. 13 (3) (2018) 695–705.
[4] S. Deng, X. Gao, Z. Lu, Z. Li, X. Gao, DoS vulner abilities and mitigation strategies in software-defined networks, J. Netw. Comput. Appl. 125 (2019) 209–219.
[5] A.S. Alshra'a, J. Seitz, Using INSPECTOR device to stop packet injection attack in SDN, IEEE Commun. Lett. 23 (7) (2019) 1174–1177.
[6] B. Liu, J. Bi, Y. Zhou, Source address validation in software-definednetworks, in: SIGCOMM 2016 - Proceedings of the 2016 ACMConference on Special Interest Group on Data Communication, (Dc) 2016, pp. 595–596.
[7] R.C. Meena, M. Nawal, M.M. Bundle, SIPAV-SDN: Source internet protocol address validation for software-defined network, Int. J. Innov. Technol. Explor. Eng. 8 (12) (2019).
[8] M.Z. Asghar, F. Subhan, H. Ahmad, S. Hakak, T.R. Gadekallu, M. M. lazab, Senti-eSystem: A sentiment-based eSystem-using hybridized fuzzy and deep neural network for measuringcustomer satisfaction, Softw. Pract. Exp. 51 (3) (2021) 571–594.
[9] P. Manzanares-Lopez, J.P. Muñoz-Gea, F.M. Delicado-Martinez, J. Malgosa-Sanahuja, A.F. De La Cruz, Host discovery solution: An enhancement of topology discovery in OpenFlowbased SDN networks, in: ICETE 2016 - Proceedings of the 13thInternational Joint Conference on E-Business and Telecommunications. Vol. 1, Icete, 2016, pp. 80–88.
[10] R.C. Meena, M. Nawal, M. Bundele, Instant detection of host in SDN (IDH-SDN), Int. J. Recent Technol. Eng. 8 (3) (2019) 5603–5608.
[11] F. Pakzad, M. Portmann, W.L. Tan, J. Indulska, Efficient topology discovery in OpenFlow-based software DefinedNetworks, Comput. Commun. 77 (2016) 52–61.
[12] G. Tarnaras, E. Haleplidis, S. Denazis, SDN and ForCES basedoptimal network topology discovery, in: 1st IEEE Conference OnNetwork Softwarization: Software-Defined Infrastructures for Networks, Clouds, IoT and Services, NETSOFT 2015, 2015.
[13] L. Ochoa-Aday, C. Cervello-Pastor, A. Fernandez-Fernandez, Self-healing topologydiscovery protocol for software-defined networks, IEEE Commun. Lett. 22 (5) (2018) 1070–1073.
[14] Y. Jiménez, C. Cervelló-Pastor, A. García, Dynamic resource discovery protocolfor software defined networks, IEEE Commun. Lett. 19 (5) (2015) 743–746.
[15] T. Alharbi, M. Portmann, F. Pakzad, The (In) security of topology discovery in open flow-based Software Defined network, Int. J. Netw. Secur. Appl. 10 (3) (2018) 01–16.
[16] X. Zhao, L. Yao, G. Wu, ESLD: An efficient and secure link discovery scheme for software-defined networking, Int. J. Commun. Syst. 31 (10) (2018) 1–18.
[17] M. Dhawan, R. Poddar, K. Mahajan, V. Mann, SPHINX:Detecting security attacks in software-defined networks, in: NDSS, (February) 2015, pp. 8–11, 1–15.
[18] S. Hong, L. Xu, H. Wang, G. Gu, Poisoning network visibility in software-Defined Networks: New attacks and counter measures, in: NDSS, (February) 2015, pp. 8–11, 1–15.
[19] D. Hasan, M. Othman, Efficient topology discovery in software defined networks: Revisited, Procedia Comput. Sci. 116 (December) (2017) 539–547.
[20] A. Nehra, M. Tripathi, M.S. Gaur, R.B. Battula, C. Lal, SLDP: A secure and light weight link discovery protocol for software defined networking, Comput. Netw. 150 (2019) 102–116.
[21] A. Nehra, M. Tripathi, M.S. Gaur, R.B. Battula, C. Lal, TILAK: A token-based prevention approachfor topology discovery threats in SDN, Int. J. Commun. Syst. 32 (17) (2019) 1–26.
[22] X. Huang, P. Shi, Y. Liu, F. Xu, Towards trusted and efficient SDN topology discovery: Alightweight topology verification scheme, Comput. Netw. 170 (2020).
[23] N. Hubballi, N. Tripathi, An event based technique for detecting spoofed IP packets, J. Inf. Secur.Appl. 35 (2017) 32–43.
[24] F. Ubaid, R. Amin, F. Bin, M. Muwar, Mitigating address spoofing attacks in hybrid SDN, Int. J. Adv. Comput. Sci. Appl. 8 (4) (2017) 562–570.
[25] B. Liu, J. Bi, A.V. Vasilakos, Toward incentivizinganti-spoofing deployment, IEEE Trans. Inf. Forensics Secur. Vol. 9 (3) (2014) 436–450.
[26] C.P.G. Machado, FCFSSAVI: First-come, first-served source address validation improvement forlocally assigned IPv6 addresses, Internet Eng. Task Force 3 (September) (2012) 1–47.
[27] L. Zhou, X. Yin, Z. Wang, Protocol security testing with SPIN and TTCN-3, in: 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, 2011, pp. 511–519.
[28] J. Bi, J. Wu, G. Yao, F. Baker, Source address validation improvement (SAVI) solutionfor DHCP, Internet Eng. Task Force (2015) 1–54.
[29] A. Bremler-Barr, H. Levy, Spoofing prevention method, in: Proceedings - IEEE INFOCOM, Vol. 1, 2005, pp. 536–547.
[30] M. Jhaveri, R. H., N.M. Patel, Attack-pattern discovery based enhanced trust model for secure routing in mobile ad-hoc networks, Int. J. Commun. Syst. 30 (7) (2017) e3148.

[31] M. Casado, et al., SANE : A protection architecture for enterprise networks, in: Proc. USENIX Secur. Symp, Vol. 49, 2006, p. 50.
[32] X. Liu, A. Li, X, W.D. Yang, Passport :Secure and adoptable source authentication university of California, Irvine, in: USENIX Symposium on Networked Systems Design and Implementation, 2008, pp. 365–378.
[33] M. Liyanage, Q.V. Pham, K. Dev, S. Bhattacharya, P.K.R. Maddikunta, T.R. Gadekallu, G. Yenduri, A survey on zero touch network and service (ZSM)management for 5G and beyond networks, J. Netw. Comput. Appl. (2022) 103362.
[34] R. Jhaveri, S. Ramani, G. Srivastava, T.R. Gadekallu, V. Aggarwal, Fault-resilience for BandwidthManagement in industrial software-defined networks, IEEE Trans. Netw. Sci. Eng. 8 (4) (2021) 3129–3139.
[35] R.H. Jhaveri, N.M. Patel, Attack-pattern discovery based enhanced trust model for secure routing in mobile ad-hoc networks, Int. J. Commun. Syst. 30 (7) (2017) http://dx.doi.org/10.1002/dac.3148.
[36] A. Kukec, M. Bagnulo, M. Mikuc, SEND-based source address validation for IPv6, in: Telecommunications, 2009. ConTEL 2009. 10th InternationalConference on, 2009, pp. 199–204.
[37] Online. Available: https://github.com/mininet/mininet/wiki/Documentation.
[38] Online. Available: https://github.com/osrg/ryu.git.[39] POX.
[39] Online. Available: https://github.com/noxrepo/pox.[40] ONOS.
[40] Online. Available: http://onosproject.org/. [41] Hpevan.
[41] Online. Available: https://marketplace.saas.hpe.com/sdn/content/sdn-controller-free-trial.
[42] Online. Available:http://www.projectfloodlight.org.
[43] OpenDayLight. Online. Available: https://www.opendaylight.org/.
[44] R.U. Khan, X. Zhang, M. Alazab, R. Kumar, An improvedconvolutional neural network model for intrusion detection in networks, in: 2019 Cybersecurity and Cyberforensics Conference, CCC, IEEE, 2019, pp. 74–77.
[45] Z. Guo, L. Tang, T. Guo, K. Yu, M. Alazab, A. Shalaginov, Deep graph neural network-based spammer detection under the perspective of heterogeneous cyberspace, Future Gener. Comput. Syst. 117 (2021) 205–218.
[46] J. Ali, B.H. Roh, B. Lee, J. Oh, M. Adil, A machine learning framework for prevention of software-defined networking controller from DDoS attacks and dimensionality reduction of big data, in: 2020 International Conference on Information and Communication Technology Convergence, ICTC, IEEE, 2020, pp. 515–519.
[47] J. Ali, B.H. Roh, Quality of service improvement with optimal software-defined networking controller and control plane clustering, Comput. Mater. Contin. 67 (2021) 849–875.
[48] J. Ali, B.H. Roh, An effective hierarchical control plane for software-defined networks leveraging TOPSIS for end-to-end QoS class-mapping, IEEE Access 8 (2020) 88990–89006.
[49] B. Sundaravadivazhagan, V. Malathi, V. Kavitha, A novel credit grounded job scheduling algorithm for the cloud computing environment, in: 2022 International Conference on Inventive Computation Technologies, ICICT, 2022, pp. 912–919, http://dx.doi.org/10.1109/ICICT54344.2022.9850650.
[50] A.R. Khan, S.M. Bilal, M. Othman, A performance comparison of open source network simulators for wireless networks, in: 2012 IEEE International Conference on Control System, Computing and Engineering, IEEE, 2012, pp. 34–38.
[51] S. Mustafa, B. Nazir, A. Hayat, S.A. Madani, Resource management in cloud computing: Taxonomy, prospects, and challenges, Comput. Electr. Eng. 47 (2015) 186–203.
[52] A.N. Khan, M.L. Kiah, M. Ali, S.A. Madani, S. Shamshirband, BSS: block-based sharing scheme for secure data storage services in mobile cloud environment, J. Supercomput. 70 (2) (2014) 946–976.
[53] B.R. Sathishkumar, B. Sundaravadivazhagan, B. Martin, et al., Revisiting computer networking protocols by wireless sniffing on brain signal/image portals, Neural Comput. Appl. 32 (2020) 11097–11109.

Dr. Ramesh Chand Meena is working as a Scientist in Software Technology Parks of India, Ministry of Electronics and Information Technology, Government of India, He obtained a Ph.D. degree in Computer Engineering from Poornima University, Jaipur, India. He completed Master of Technology (MTech.) in Computer Engineering in 2006 from Sam Higginbottom University of Agriculture, Technology & Sciences (formerly Allahabad Agricultural Institute), Allahabad, India, Master of Science (M.Sc.) in Computer Science in 2003 from Maharshi Dayanand University, Rohtak, India. He has more than 24 years of working experience at various organizations of the Government of India in the field of Computer Software Development, Computer Network Management, and Implementation of Government of India schemes like Software Technology Park (STP) & Electronic Hardware Technology Park (EHTP). He has published research papers in international/national journals and conferences. His research interests include Network Architecture and Security, Computer Programming, Software Defined Networks, IoT, etc.

Surbhi Bhatia received her doctorate in Computer Science and Engineering from Banasthali Vidyapith, India, in 2018, and did her Masters in Technology from Amity University in 2012 and Bachelors in Information Technology in 2010. She is also PMP certified from PMI, USA. She is currently an Assistant Professor in the Department of Information Systems, College of Computer Sciences and Information Technology, King Faisal University, Saudi Arabia. She is associated with reputed journals and has published many research papers in high indexing databases. She has been granted patents from USA, Australia and India. Her research interests are Artificial Intelligence, Sentiment Analysis and Information Systems.

Dr. Rutvij H. Jhaveri (Senior Member, IEEE) is an experienced educator and researcher working in the Department of Computer Science & Engineering, Pandit Deendayal Energy University, Gandhinagar, India. He conducted his Postdoctoral Research at Delta-NTU Corporate Lab for Cyber–Physical Systems, Nanyang Technological University, Singapore. He completed his Ph.D. in Computer Engineering in 2016. In 2017, he was awarded with prestigious Pedagogical Innovation Award by Gujarat Technological University. Currently, he is co-investigating a funded project from GUJCOST. He was ranked among top 2% scientists around the world in 2021. He has 2050+ Google Scholar citations with h-index 23. Apart from serving as an editor/guest editor in various journals of repute, he also serves as a reviewer in several international journals and also as an advisory/TPC member in renowned international conferences. He authored 110+ articles including several IEEE Transactions published by prominent publishing houses. He also possesses memberships of various technical bodies such as ACM, CSI, ISTE, IDES and others. He is a member of the Research Advisory Board in Symbiosis Institute of Digital and Telecom Management since 2021. He is an editorial board member in several Hindawi and Springer journals. He also served as a committee member in "Smart Village Project" - Government of Gujarat, at the district level during the year 2017. His research interests are SDN, network security/resilience, IoT systems and eHealth.

Dr. Cheng is a Full Professor in the School of Control and Computer Engineering at NCEPU in Beijing and a Visiting Professor at Insight in Dublin. His research mainly lies in distributed systems, deep learning, cloud computing and process mining. Specifically, he is interested in performance optimization for parallel and distributed data applications and systems. He was an Assistant Professor in the School of Computing at Dublin City University (DCU), and a Marie Curie Fellow at The Performance Engineering Laboratory (PEL) directed by Prof. John Muphy in University College Dublin (UCD), Ireland. Before joining UCD in 2018, he was a postdoctoral researcher at the Architecture of Information Systems Group (AIS) in Eindhoven University of Technology (Netherlands), worked with Prof. Wil van der Aalst for about two years focusing on new process mining techniques that are able to deal with huge event logs. During 2014 and 2016, he worked as a postdoctoral researcher in the Knowledge-based System Group led by Prof. Markus Krötzsch at TU Dresden (Germany). He received his Ph.D. degree from National University of Ireland Maynooth under the supervision of Prof. Tomas Ward in 2014. During his Ph.D., he was also a research assistant at IBM Research Dublin under the supervision of Prof. Georgios Theodoropoulos and Dr. Spyros Kotoulas. Currently, aside from the main focus on advancing network-aware computing systems, high-performance DNN inference and energy data analytics, Dr. Cheng also has some collaborative projects in intelligent IoT edge systems, cloud-based process mining, high-performance privacy-preserving data processing, in-network computing, and AI accelerator. He is currently a Senior Member of the IEEE.

Ankit Kumar is an Assistant professor at the Department of Computer Engineering & Applications, GLA University, Mathura, India. He completed M. Tech from IIIT Allahabad and received Ph.D. from Satya Sai University of Medical Science, Sehore, Bhopla. His research is in wireless sensor networks & Machine Learning. His articles are published in over 25 in SCI journals and 39 in Scopus index journals. He has received 8 patents. He has received a Research Grant from TEQIP. His work has been profiled broadly in information security, cloud computing image processing, neural networks, and networks. His research interests include computer network information security, computational models, compiler design, and data structures. He is a reviewer and editor at numerous reputed journals.

Arwa Mashat is an assistant professor in the Information System department in the Faculty of Computing and Information Technology, King Abdulaziz University, Rabigh, Saudi Arabia. Dr. Arwa received her Ph.D. in Instructional Design & Technology from Old Dominion University in USA. Her research interests include eye tracking, education technology, online learning and internet of things.