International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Instant Detection of Host in SDN (IDH-SDN)

Ramesh Chand Meena, Meenakshi Nawal, Mahesh Bundele

Abstract: ***SDN features are making it more popular day by day: centralized monitoring, control of network equipments, increased performance and flexibility in designing network policies as per organization requirements. The SDN controller deals with data & control plane separately. The SDN switches are simply data forwarding devices and controller decides control over forwarding data through them. Controller has a technique to identify the network switch and router nodes; but it does not identify the presence of hosts before they generated network traffic and is not able to create the packet forwarding rules, security policies. The objective of this paper is to detect connected host before they generate any traffic and store host details at controller level for future researches in area of development of new network tools, applications, optimizations techniques and security. Here, we propose Instant Detection of Host in SDN (IDH-SDN) to detect host before transmission of any data packet and store host details in a HostTable at controller level. In our experiment, various network topologies have been used to test host detection and data collection algorithm and results of all experiments verified with Wireshark network packet analyzer. The HostTable data may be used for various purposes such as development of new network tools, policies, security approaches in OpenFlow network.***

Keywords: Software Defined Networking, Host, Detection, Controller, Instant, SD, IDH

I. INTRODUCTION

SDN is becoming more prefixed technology for network designing and controlling centralized and handle separately data plane and control plane. These planes were implemented in firmware of network hardware of conventional network; and they used routing paths and decisions to forward the data packet further or not. SDN implemented the data plane and control plane at switch (packet forwarding device) level and SDN controller level respectively [14]. Their separation provided flexibility for network administrator to control centrally and design network policies programmatically. The data packet forwarding network devices are responsible to forward to packets and control plane is to make rules for data packets to be forwarded one or more network nodes/hosts and sending them to data forwarding devices for execution[16]. The SDN controller [27] needs network topology to make data forwarding rules and control data paths. The correct topology detail is needed to monitor, manage and control network traffic. The topology includes the details of data forwarding devices (network elements – switches & routers) and connected hosts.

The majority of SDN controllers have topology detection systems [27] limited to switches, routers and connectivity linking them. The recent studies shown that controller of SDN do not identify the presence of hosts before they generated network traffic and the controller is not able to create the packet forwarding rules, security policies. In this situation, the packet forwarding devices do not have any policy for the data traffic comes first time from a host. The device sends host's first packet to the SDN controller and waits for rules and policies to handle this type of traffic. The controller examines packet received from packet forwarding device, extracts the host information from packet and creates rules & policies for it and send them to packet forwarding device for acting accordingly with traffic from the host. Receiving a packet from forwarding device, examination of the packet, extraction of host details from packet, creation of rules & policies for host and sending to the forwarding device involve many processes. These processes are time consuming and till the time, the packet is in waiting state and first packet always takes time to reach to its destination. In this paper we propose Instant Detection of Host in SDN (IDH-SDN) to detect host before transmission of actual data packets and to store host details in a HostTable at controller level. This method has been coded for RYU SDN Controller and tested in various network scenarios.

This remaining paper is having following divisions. In Div.-II we describe OpenFlow background. In Div.-III has a detail of Network Topology Detection in OpenFlow. In Div.-IV we described Related Works. In Div-V describes proposed IDH-SDN. In Div-VI we described Implementation & Testing of IDH-SDN and in Div.-VII Conclusions.

II. OPENFLOW BACKGROUND

Software-Defined Networking (SDN) is a rising creation that is dynamic, adaptable, cost-effective and allow it perfect for the dynamic, high-bandwidth nature of current requirement. SDN separates the forwarding and network control actions to make possible directly programmable to network control. The functions and services of network are abstracted by the underlying infrastructure. The OpenFlow protocol is an essential and main element for building SDN solutions. Open Network Foundation has promoted the OpenFlow as a standard protocol for providing south-bound interface among SDN switches and controller.

The Controller and switches perform a greeting to start an OpenFlow connection. Switches talks with its controller on related IP and TCP port to establish an encrypted TCP connection.

Manuscript published on 30 September 2019
* Correspondence Author
Ramesh Chand Meena*, Research Scholar, School of Engineering & Technology, Poornima University, Jaipur, India. Email: rameshrmz@yahoo.com, ORCID:0000-0002-2649-1384
Meenakshi Nawal, Associate Professor, Computer Science and Engineering, School of Engineering & Technology, Poornima University, Jaipur India. Email: meenakshi.nawal@poornima.edu.in
Mahesh Bundele, Director & Principal, Poornima College of Engineering, Jaipur, India Email: maheshbundele@poornima.org

The controller sends an OFPT_FEATURE_REQUEST to switch which supports the OpenFlow version and switch sends an OFPT_FEATURE REPLY message to complete establishment of OpenFlow connection with information such as unique identifier of switch as datapath_id, MAC addresses of switch active ports. Handshaking between controller and switch reveals the existence of switches but not interconnections of the network devices.

The controller uses network topology details to perform various network control and management tasks. The most important task is to detect and setup proper paths to enable the switches to forward network traffic.

Each OpenFlow switch has flow tables along with a group table. The controller used OFPT_FLOW_MOD messages to make manipulations in flow tables. It involves addition, deletion and modification in the flow table entries. Every flow entry has a structure of fields, actions set and counters set. These fields are used to match field values with packet on its arrival. On matching a packet with any entry, the respective counter entry is increased and related activities are executed. Table miss event is raised when entry does not match and it instructs for packet to drop or send to other table or send to controller. In last case, switch raises an OFPT_PACKET_IN messages along with packet to controller to evaluate packet's contents. After deciding actions for packet, it is returned to the switch along with list of required actions and the controller transmits an OFPT_FLOW_MOD message to insert an entry into flow table of a switch for processing related packets in future.

III. NETWORK TOPOLOGY DETECTION IN OPENFLOW

Link Layer Discovery Protocol (LLDP) is a single-hop protocol and used by network devices to advertise their presence, properties and their neighbors in wired LANs. The network switch sends LLDP messages from its ports at a fixed time period [27]. The message is framed by setting hex 88cc value in ethertype field for sending it at address of multicast MAC (i.e. 01:80:c2:00:00:0E) of bridge. Since, SDN switch do not send LLDP messages by itself. The following steps are used to detect the network topology:-

- SDN controller creates an entry into table flow of each SDN switch using OFPT_FLOW_MOD for any LLDP frame received from any port of the switch to forward to the controller. This entry orders to the SDN switch to communicate with controller using OFPT_PACKET_IN message on receipt of any LLDP message from any port of switch.
- Controller creates OFPT_PACKET_OUT message with a LLDP frame payload & instructs to forward this frame to the respective port.
- The SDN switch extracts the LLDP frames payload form OFPT_PACKET_OUT messages on receipt of the message and forwards to every switch port except in-coming port.
- The SDN Switch LLDP packets received from port to the controller by OFPT_PACKET_IN messages.
- The SDN controller extracts network topology related information of network devices.

Above procedure detects only switches and connecting links but it does not detect hosts. Most of the SDN controllers implements basic host detection method. This method uses table miss flow entry of switches that forces the switch to forward packets to controller. In event of a host generates an ARP or IP traffic and switch do not have any existing flow entries for such traffic, then first packet of host traffic is forwarded to controller by SDN switch. Controller completes host detection by extracting host details from this first packet. Since LLDP is single hop protocol still it may also be used to detect host but it needs to implement this protocol at host level. This implementation is not easy at host level because hosts and network devices are monitored by the different entities.

IV. RELATED WORK

In study [21] described that host discovery activity depends on details contained in ARP and DHCP packets available via the PacketIn function at layer 2 protocol. It also stated that OpenFlow Discovery Protocol refers to controller learning about the presence of network nodes (SDN switches) in OpenFlow using LLDP frame format. These do not suggest any host discovery mechanism before generation of traffic by host itself. In [16] authors have been inspired by nmap utility and proposed a proactive host discovery algorithm using ARP requests created at controller level to get host information after session establishment. This approach used OFPT_PACKET_OUT messages targeted to an edge port, SDN switch dispatches out ARP-requests to the mentioned switch port. In case a host is live with edge port, first ARP-reply will be dispatched to controller through switch and controller will extract the required information from the ARP reply packet. Controller will have entire network topology details (i.e. switches, ports of switches, link connectivity between them, host IP and MAC) using both LLDP and host discovery module.

It has been observed that approach proposed in [16] is working properly when switch is connected with running controller and then making any change on a switch port. But this approach will not be working properly in two situations: first, when switch is started before controller and second, when switch is disconnected from controller, any change made on a switch port and reconnected with the controller. Authors designed the approach to implement at OFPT_PORT_STATUS message, which is raised by switch to connected controller when any change happens on a switch port. The change may be addition/deletion/status change of a port.

V. PROPOSED INSTANT DETECTION OF HOST IN SDN (IDH-SDN)

Network topology detection in SDN controllers is limited to switches, routers and connectivity linking them and controller do not identify the hosts' presence before they generated network traffic. Without hosts information the controllers are not able to create the packet forwarding & filtering rules, security policies [30]. The packet forwarding devices do not have any forwarding & filtering rules, security policies for the data traffic comes first time from a host. The network device sends such first packet to the SDN controller and waits for rules and policies to handle this type of traffic.

The controller inspect packet received from packet forwarding device, extracts the host information from packet and creates rules & policies for the packet and add into flow table of packet forwarding device so that device can act accordingly with forthcoming same type of traffic from the host. This activity involves many processes at controller level such as receiving a packet from forwarding device, inspection of the packet, mining of host details from packet, creating of rules & policies and adding into flow table of forwarding device for the traffic of host.

Every process is time consuming. Till these processes are completed by controller, the packet has a long wait and the first packet always takes long time to reach to its destination. In this section we have proposed Instant Detection of Host in SDN Controller. As soon as any switch/switch port/host is added into the SDN network, our following instant detection algorithm identifies the SDN Host and adds its details into HostTable at controller level:

1. Controller generates ARP request packets using Source IP as 0.0.0.0 and Source MAC as switch port MAC with target IP x.x.x.x and MAC "FF:FF:FF:FF:FF:FF" through switch port.
2. OFSwitches broadcast ARP request packet.
3. If any host having x.x.x.x IP address will respond to ARP Request message through an ARP reply message otherwise it will be ignored.
4. Received ARP reply packet will be forwarded by OFSwitch to controller using PacketIn message and the packet has IP, MAC address and DataPath has Switch No and Switch Port No.
5. The Controller extracts IP, MAC, Switch ID and Switch Port ID from ARP Reply packet and DataPath to add into HostTable. After extracting required information the packet is forwarded to its destination in a normal way after making required entry in flow-table. In [29] authors have discussed about a table to keep track of IP and MAC of every host very similar to our HostTable.
6. The Controller also forwards ARP Reply packets through Packet_Out message to OFSwitch.

Block diagram of IDH-SDN and format of HostTable are given in figure 1 and figure 2 respectively:-

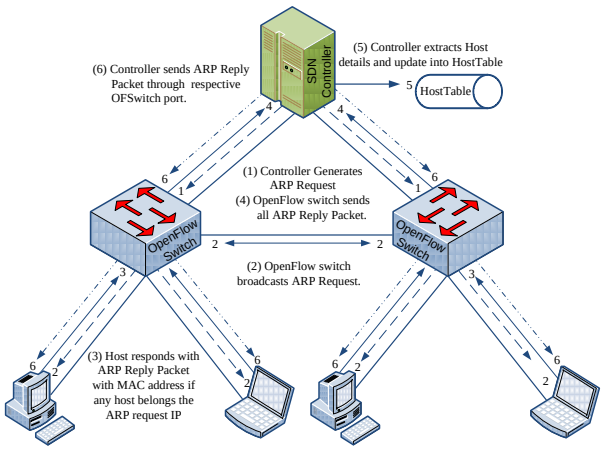


Figure 1: Block Diagram of IDH-SDN

Switch ID	Switch Port ID	Host MAC	Host IP

Figure 2: HostTable format

First of all in proposed approach, we need changes and implementations at various events of controller for a simple switch. When controller and switch are started, the switch do not have any flow entries so all ARP packets will be forwarded to the controller. Controller examines packets and extracts required information such as Host MAC, Host IP, Switch ID, Switch Port ID. Software module for the algorithm has been coded for RYU SDN Controller and tested in various network scenarios.

Architecture of the proposed IDH-SDN is given in figure-2. It has hosts, switches, Controller as components of SDN network. SDN switches are connected with hosts, controller and have interconnection links between forwarding devices (Switches, routers, hubs). RYU Controller is used to connect with switches through Open Southbound API channel. On top of the controller, IDH-SDN is implemented and connected through Open Northbound API channel to achieve the object of detection of host on the event of handshaking of SDN Switch with Controller and on event of change notified on switch port/host.

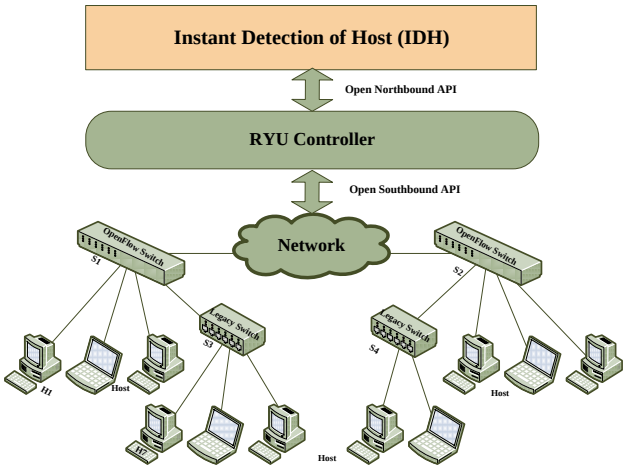


Figure 2: Architecture of IDH-SDN

VI. IDH-SDN IMPLEMENTATION AND TESTING

In this section we have discussed the implementation and testing of proposed IDH-SDN algorithm. We used RYU SDN controller and OpenFlow switches for implementation of proposed system. Mininet network emulator was setup in Ubuntu version 18.04 64 bit OS environment to create virtual OpenFlow switches and hosts using Intel-i5 6200U CPU @2.30 GHz with 4 GB RAM and 512GB SSD hardware configuration. In this implementation, we have used OpenFlow 1.2 version switches and Python programming language.

RYU SDN framework provided simple_switch12.py sample application program has been used for implementation of proposed algorithm. First of all, we used EventOFPSwitchFeatures and EventOFPPortStatus event of RYU controller. Controller forwards feature request message to switch at the time of handshaking to get features of the switch. EventOFPPortStatus event is raised by OpenFlow switch to the controller to notify the change of ports.

With above messages, the features of the switch or change of switch ports are passed through ev.msg object and this information is used to take required details (such as Switch ID, Switch Port ID & MAC) for detection of host. Above both messages are also used to make necessary entries in flow table of switch so that all ARP packets should be forwarded to controller.RYU controller raises EventSwitchLeave when connected OpenFlow switch link is terminated. In our research this event is used to remove the connection terminated switch details from the HostTable. EventOFPPortStatus is used in the proposed algorithm for removing host from HostTable when switch port is turned down.

We used EventOFPPacketIn event to examine ARP reply packets and extract host information from the packets. Host IP address and MAC address from ARP reply packet, Switch ID and Switch Port ID from datapath are taken and checked with HostTable and if this information does not exist in the HostTable than it is added to the table otherwise it will be ignored. After completion of above activity, the ARP reply packet is forwarded to its destination by making required changes in switch flow table. Implementation of proposed IDH-SDN at EventOFPSwitchFeatures and EventOFPPortStatus is shown in flow charts shown in figure 2 & 3 respectively.

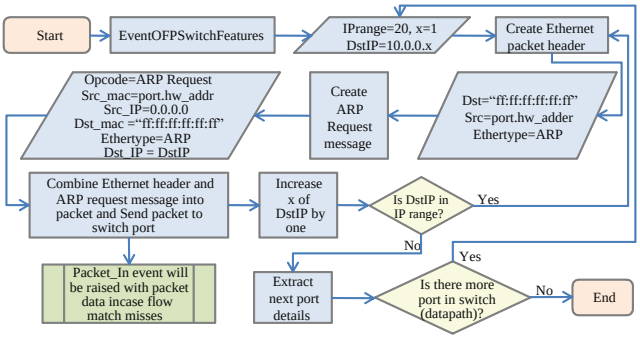


Figure 3: Flow Chart for IDH-SDN Implementation on Event of OFPSwitchFeatures

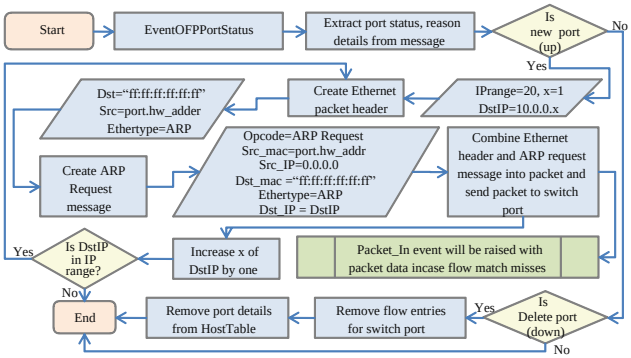
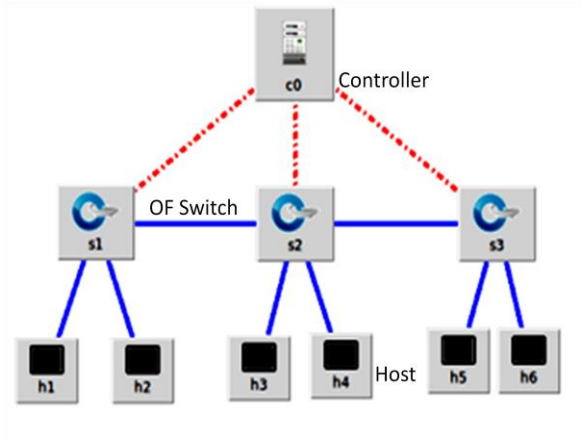
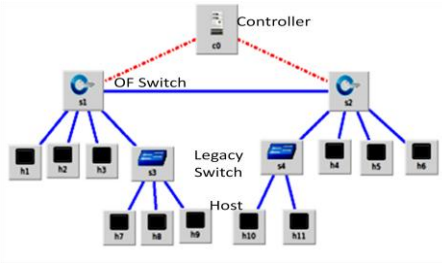


Figure 4: Flow Chart for IDH-SDN Implementation on Event of OFPPortStatus

SDN Topologies Used for Testing: Two SDN topologies were used for testing of proposed IDH-SDN. First SDN topology has 3 OpenFlow switches, 6 hosts and an SDN Controller. Second topology has 2 OpenFlow Switches, 2 legacy switches, 11 hosts and an SDN controller. The diagram of above mentioned different topology configuration is shown in figure 4:



(A)



(B)

Figure 4: SDN Topologies

Collection of Host Information: In the first experiment, we emulated a network topology which consists of 6 hosts with 3 numbers of switches and 1 SDN Controller as given in figure 4(A). Information collected by approach during experimentation related to host such as MAC, IP, Switch ID and Switch Port ID is given in following table-1:

Table 1: Data from SDN Topology (A)

Switch ID	Switch Port ID	Host MAC	Host IP
1	2	['fa:a2:6a:ff:ef:ff']	['10.0.0.2']
	3	['c2:fa:72:35:67:e1']	['10.0.0.1']
2	2	['0a:d4:e8:9e:26:6c']	['10.0.0.3']
	3	['3a:a8:1b:85:36:a1']	['10.0.0.4']
3	1	['3e:c8:6b:6c:a3:7e']	['10.0.0.5']
	2	['2a:1e:a8:ef:47:56']	['10.0.0.6']

In second experiment, we emulated a network topology which consists of 11 hosts with 2 numbers of OpenFlow switches, 2 numbers of legacy switches and 1 SDN Controller as shown in figure 4(B). Information collected by system during experimentation related to host such as MAC, IP, Switch ID and Switch Port ID is given in following table-2:

Table 2: Data from SDN Topology (B)

Switch ID	Switch Port ID	Host MAC	Host IP
1	3	['de:d9:f0:e3:17:5a']	['10.0.0.1']
	4	['a2:6d:5a:0d:a5:13']	['10.0.0.2']
	5	['b2:cc:45:87:91:07']	['10.0.0.3']
	1	['1a:6c:0b:21:24:ad']	['10.0.0.7']
		['ee:52:76:df:36:93']	['10.0.0.8']
		['ba:71:c4:1e:2b:58']	['10.0.0.9']
2	1	['2a:4c:99:63:16:98']	['10.0.0.4']
	2	['9a:3a:08:9b:82:93']	['10.0.0.5']
	3	['6e:17:56:5a:85:80']	['10.0.0.6']
	4	['16:be:9f:6b:ff:ad']	['10.0.0.10']
		['76:a2:6d:44:69:ed']	['10.0.0.11']

Our approach was also tested by starting switch first and controller later. In second case, we disconnected the working switch from controller,

made a change on switch port and reconnected the switch with controller. In both the situations our approach worked properly and collected host information of various network scenarios as shown above and stored in HostTable at controller level.

Table 3: HostTable record having multiple IPs for same MAC

Switch ID	Switch Port ID	Host MAC	Host IP
1	2	['fa:a2:6a:ff:ef:ff']	['10.0.0.2']
	3	['c2:fa:72:35:67:e1']	['10.0.0.1']
2	2	['0a:d4:e8:9e:26:6c']	['10.0.0.3']
	3	['3a:a8:1b:85:36:a1']	['10.0.0.4']
3	1	['3e:c8:6b:6c:a3:7e']	['10.0.0.5']
	2	***['2a:1e:a8:ef:47:56']***	***['10.0.0.6']*** ***['10.0.0.8']***
	4	***['0a:10:a8:eb:70:f2']***	***['10.0.0.7']***

We also tested our system by adding new hosts and assigning multiple IPs to single MAC. New hosts were detected by the system and multiple IP assignment to a single MAC was properly reflected in the HostTable. This experiment results are shown in table-3 in bold and italic font face. Collected test results were verified with Wireshark open-source packet analyzer and it was found that host related information collected during above experiments were correct.

VII. CONCLUSION

Our paper has proposed instant host detection in Software Defined Networks OpenFlow Network. We have implemented as a module in the RYU controller and collected required data at the handshaking of OpenFlow switch with controller. The algorithm has also been tested for adding new hosts into network and removing existing host from network. The approach generated ARP-Requests and ARP-Reply packets transmitted by hosts through switch to controller were examined; required information was extracted from ARP reply packets and stored into HostTable. In our experiment, various network topologies have been used to test host detection and data collection algorithm and results of all experiments verified with Wireshark network packet analyzer. The HostTable data may be used for various purposes such as development of new network tools, policies, security approaches in OpenFlow network.

REFERENCES

1. A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran and S. Guizani, "Securing software defined networks: taxonomy, requirements, and open issues," in IEEE Communications Magazine, vol. 53, no. 4, pp. 36-44, April 2015.
2. Pradeepa R and Pushpalatha M, “Artificial Neural Network (ANN) BasedDDoS Attack Detection Model on Software Defined Networking (SDN)”, International Journal of Recent Technology and Engineering (IJRTE), Volume-8 Issue-2, July 2019, pp. 4887-4894
3. B. H. Lawal and A. T. Nuray, "Real-time detection and mitigation of distributed denial of service (DDoS) attacks in software defined networking (SDN)," 2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, 2018, pp. 1-4.
4. C. Zhang, G. Hu, G. Chen, A. K. Sangaiah, P. Zhang, X. Yan and W. Jiang, "Towards a SDN-Based Integrated Architecture for Mitigating IP Spoofing Attack," in IEEE Access, vol. 6, pp. 22764-22777, 2018
5. D. Satasiya and Raviya Rupal D., "Analysis of Software Defined Network firewall (SDF)," 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, 2016, pp. 228-231
6. E. Nordmark, M. Bagnulo, E. Levy-Abegnoli "FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses", Internet Engineering Task Force, RFC-6620 ISSN:2070-1721 May 2012
7. Gian M. di Marzo and Francesco Benedetto, “Software Defined Networks for Data Center Optimization”, Recent Patents on Computer Science (2014) 7: 24.
8. Guolong Chen, Guangwu Hu, Yong Jiang and Chaoqin Zhang, "SAVSH: IP source address validation for SDN hybrid networks," 2016 IEEE Symposium on Computers and Communication (ISCC), Messina, 2016, pp. 409-414
9. H. T. Nguyen Tri and K. Kim, "Assessing the impact of resource attack in Software Defined Network," 2015 International Conference on Information Networking (ICOIN), Cambodia, 2015, pp. 420-425
10. K. Guerra Pérez, X. Yang, S. Scott-Hayward and S. Sezer, "A configurable packet classification architecture for Software-Defined Networking," 2014 27th IEEE International System-on-Chip Conference (SOCC), Las Vegas, NV, 2014, pp. 353-358.
11. A. ARIVOLI, Eniyan Manivasagam, Ashwin K and Nirmal Kuma, “SDN-Ddos -Detecting Ddos Attack in SDN using Sflow Mitigation Technology”, International Journal of Engineering and Advanced Technology (IJEAT), Volume-8 Issue-6, August 2019, pp. 302-306
12. Kwon, Jonghoon, Dongwon Seo, Minjin Kwon, Heejo Lee, Adrian Perrig and Hyogon Kim. “An incrementally deployable anti-spoofing mechanism for software-defined networks.” 2015 Computer Communications 64: pp.1-20.
13. L. M. van Adrichem, Niels & Doerr, Christian & A. Kuipers, Fernando. (2014). “OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks.” IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World 10.1109/NOMS.2014.6838228: pp. 1-8.
14. M. Dabbagh, B. Hamdaoui, M. Guizani and A. Rayes, "Software-defined networking security: pros and cons," in IEEE Communications Magazine, vol. 53, no. 6, pp. 73-79, June 2015.
15. M. Z. Masoud, Y. Jaradat and I. Jannoud, "On preventing ARP poisoning attack utilizing Software Defined Network (SDN) paradigm," 2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Amman, 2015, pp. 1-5.
16. Manzanares-Lopez, Pilar & Muñoz-Gea, Juan & Manuel Delicado-Martinez, Francisco & Malgosa, Josemaria & Flores de la Cruz, Adrian. (2016). Host Discovery Solution: An Enhancement of Topology Discovery in OpenFlow based SDN Networks. 80-88.
17. Ramesh Chand Meena, Mahesh Bundele, "A Review on Implementation Issues in IPv6 Network Technology”, International Journal of Engineering Research and General Science(2015), Vol.3, Issue 6: pp.800-809
18. O. Chippalkatti and S. U. Nimbhorkar, "An approach for detection of attacks in software defined networks," 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, 2017, pp. 1-3.
19. O. Strugaru, A. D. Potorac and A. Graur, "The impact of using Source Address Validation filtering on processing resources," 2014 10th International Conference on Communications (COMM), Bucharest, 2014, pp. 1-4
20. P. B. Pawar and K. Kataoka, "Segmented proactive flow rule injection for service chaining using SDN," 2016 IEEE NetSoft Conference and Workshops (NetSoft), Seoul, 2016, pp. 38-42
21. P. Zanna, S. Hosseini, P. Radcliffe and B. O'Neill, "The challenges of deploying a software defined network," 2014 Australasian Telecommunication Networks and Applications Conference (ATNAC), Southbank, VIC, 2014, pp. 111-116.
22. K. K. Karmakar, V. Varadharajan and U. Tupakula, "Mitigating attacks in Software Defined Network (SDN)," 2017 Fourth International Conference on Software Defined Systems (SDS), Valencia, 2017, pp. 112-117
23. S. Murtuza and K. Asawa, "Mitigation and Detection of DDoS Attacks in Software Defined Networks," 2018 Eleventh International Conference on Contemporary Computing (IC3), Noida, 2018, pp. 1-3.
24. S. Nadar and S. Chaudhari, "Proactive-routing path update in Software Defined Networks(SDN)," 2017 International Conference on Intelligent Computing and Control (I2C2), Coimbatore, 2017, pp. 1-3.
25. S. T. Ali, V. Sivaraman, A. Radford and S. Jha, "A Survey of Securing Networks Using Software Defined Networking," in IEEE Transactions on Reliability, vol. 64, no. 3, pp. 1086-1097, Sept. 2015.
26. A. MahaLakshmi, N. Swapna Goud and Dr. G. Vishnu Murthy, “A Survey on Phishing And It’s Detection Techniques Based on Support Vector Method (SVM) and Software Defined Networking(SDN)”, International Journal of Engineering and Advanced Technology (IJEAT), Volume-8, Issue-2S,December 2018 pp. 498-503

27. T. Alharbi, M. Portmann and F. Pakzad, "The (in)security of Topology Discovery in Software Defined Networks," 2015 IEEE 40th Conference on Local Computer Networks (LCN), Clearwater Beach, FL, 2015, pp. 502-505.
28. T. Chin, X. Mountrouidou, X. Li and K. Xiong, "Selective Packet Inspection to Detect DoS Flooding Using Software Defined Networking (SDN)," 2015 IEEE 35th International Conference on Distributed Computing Systems Workshops, Columbus, OH, 2015, pp. 95-99
29. T. Javid, T. Riaz and A. Rasheed, "A layer2 firewall for software defined network," 2014 Conference on Information Assurance and Cyber Security (CIACS), Rawalpindi, 2014, pp. 39-42.

AUTHORS PROFILE

Ramesh Chand Meena is currently pursuing the Ph.D. degree in Computer Science & Engineering with the Poornima University, Jaipur, India. He is also working as Scientist with Software Technology Parks of India, Ministry of Electronics and IT, Government of India. He completed Master of Technology in Computer Science and Engineering in 2006, Master of Science in Computer Sciences in 2003. He has more than 20 years of working experience at various organizations and departments of Government of India in the field of Computer and IT. His research interests include Network Architecture and Security, Computer Programming, Software Defined Networks etc.

Dr. Meenakshi Nawal is currently working as Associate Professor with the Poornima University, Jaipur, India. She is Gold Medalist in M.Sc (IT) from MDS University Ajmer. She has completed Ph. D (Computer Science) from Banasthali Vidhyapith, Jaipur. She is having 13 years experience of Academics and Research. Her area of research is "Patient Authentication and Security measures in Remote Health Monitoring". She has attended many National and International Workshops, Conferences and Published papers in National conference. Her areas of research interest are Machine Learning, Deep Learning, Image Processing, Big Data Analytics and Software Defined Networks etc.

Dr. Mahesh Bundele is currently working as Principal and Director of Poornima College of Engineering, Jaipur since 1st September 2018. He has total 33 years experience in teaching and research. He has developed many unique research methodology concepts and implemented. He is the mentor and controller of quality research and publications at the University. He is also responsible for inculcation of innovative and critical analysis concepts across the University and across the Poornima Foundation involving three other campuses. He did his doctoral degree in Wearable Computing and guiding research in Pervasive & Ubiquitous Computing, Computer Networks, Software Defined Networking. His areas of interests are also Wireless Sensor Networks, Algorithmic research, Mathematical Modeling, and Smart grids. He has more than 50 publications in reputed journals and conferences. He has been reviewer of few IEEE Transactions. He is actively involved in IEEE activities in Rajasthan Subsection and Delhi Section and holding the responsibility on Standing Committee of IEEE Delhi Section for Technical & Professional activities for controlling quality of conferences and publications in IEEE. He is also holding position of vice chair in Jaipur ACM professional chapter.