Team ID 8: Sprint

SDN Threat Vector And

Security Solutions

# Team Member

Anand Kumar

Amogh Eshappa Setra

Nandan Goudar

Sachin Uplaonkar

# References

1. S. Hong, L. Xu, H. Wang, and G. Gu, “Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures,” in NDSS’15"
2. G. Pickett, Abusing Software Defined Networks [Online]. Available: <https://www.blackhat.com/docs/eu-14/materials/eu-14-Pickett-Abusing-Software-Defined-Networks-wp.pdf>
3. "Dabbagh, M.; Hamdaoui, B.; Guizani, M.; Rayes, A., ""Software-defined networking security: pros and cons,"" in Communications Magazine, IEEE , vol.53, no.6, pp.73-79, June 2015 doi: 10.1109/MCOM.2015.7120048,URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7120048&isnumber=7120004>"
4. "Zhiyuan Hu; Mingwen Wang; Xueqiang Yan; Yueming Yin; Zhigang Luo, ""A comprehensive security architecture for SDN,"" in Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on , vol., no., pp.30-37, 17-19 Feb. 2015 doi: 10.1109/ICIN.2015,URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7073803&isnumber=7073795>
5. "Kandoi, R.; Antikainen, M., ""Denial-of-service attacks in OpenFlow SDN networks,"" in Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on , vol., no., pp.1322-1326, 11-15 May 2015 doi: 10.1109/INM.2015.7140489,URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7140489&isnumber=7140257>
6. Ashraf, J.; Latif, S., ""Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques,"" in Software Engineering Conference (NSEC), 2014 National , vol., no., pp.55-60, 11-12 Nov. 2014, doi: 10.1109/NSEC.2014.6998241 URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6998241&isnumber=6998226>
7. Mousavi, S.M.; St-Hilaire, M., ""Early detection of DDoS attacks against SDN controllers,"" in Computing, Networking and Communications (ICNC), 2015 International Conference on , vol., no., pp.77-81, 16-19 Feb. 2015 doi: 10.1109/ICCNC.2015.7069319, URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7069319&isnumber=7069279>
8. CMPE 220 proposal document
9. <https://www.opennetworking.org/images/stories/downloads/sdn-resources/IEEE-papers/evolution-of-sdn-and-of.pdf>

# Code Description

**Algorithm**

Step 1: For each PacketIn event

Step 2: #Calculate entropy H after each packet is received

Step 3: If H < threshold then

#Update dictionary for each packet

#Register TimerFunction (first time only)

Else

#Normal execution

Go to Step 2

Step 4: TimerFunction: Begin #access dictionary every 5 seconds

Step 5: If numberOfPackets > 500 then

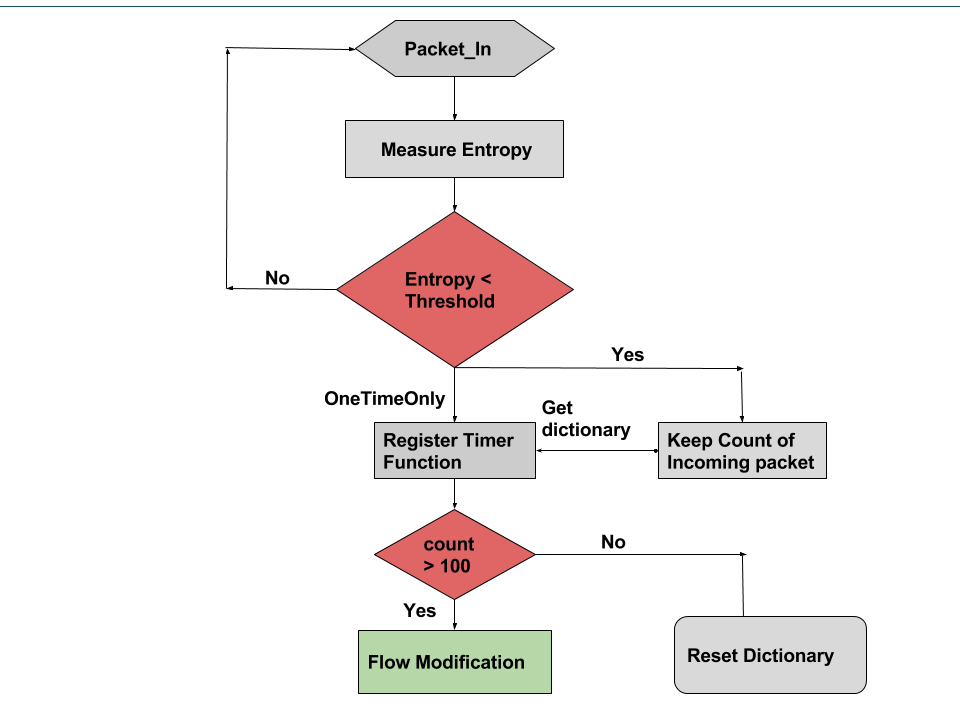
FlowMod message

Else

Reset dictionary

Step 6: TimerFunction: End

**Flow chart**



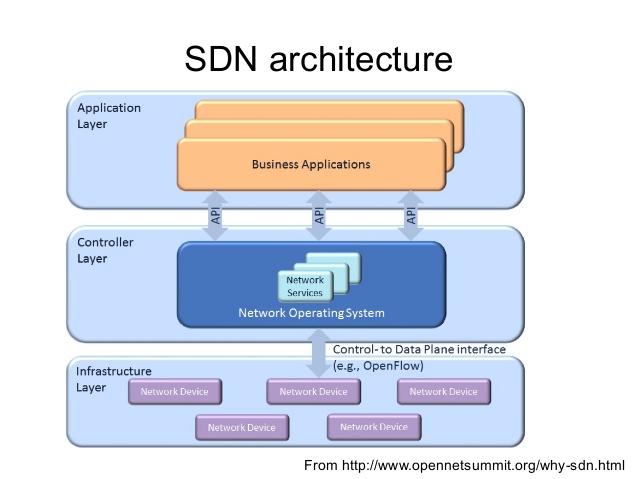
# Design

**Objective**

# In this paper our aim is to acquire in depth knowledge of Software Defined Network (SDN). Learn the SDN security attacks performed on SDN at different layers. Explore the existing prevention techniques designed, researched and implemented. We are interested in exploring one particular type of attack i.e. DDOS that can launched at different levels. We also intend to provide a solution to resolve or mitigate this attack as early as possible.

**Introduction**

The figure below illustrates the architecture of SDN. It consists of 3 layers, the application layer, the control layer and the infrastructure layer. This structural engineering decouples the system control and forwarding capabilities, ensuring the system control to be programmable and the underlying network infrastructure to be abstract for developing applications to provide network services. The controller layer is core of the SDN network which has the map of the entire network to track the resources, it can gather the inventory and the traffic statistics. The infrastructure layer hosts the network devices which are considered to be dumb as they involved in only forwarding the packets. On top of the controller resides the application layers where application are developed to monitor the network, load balance the controller, take appropriate actions on the detection of an attack.



**SDN Security**

Ensuring security in SDN is critical to provide secure communication because,

* The honesty and security of SDNs stay problematic with regards to the position of administration usefulness in a solitary concentrated virtual server.
* The programmability part of SDNs additionally makes them more defenseless against various vindictive code endeavors and assaults.
* Southbound interface of a SDN can likewise effortlessly be focused with various denial of service and side channel attacks.
* Design mistakes of SDNs can have a bigger number of genuine outcomes than in conventional systems.
* Establishing trust between the controller, infrastructure device and hosts is crucial.

**Attacks**

In this paper, the each layer and services provided in SDN are studied to find the vulnerabilities that can be used to launch the attacks. Some of the possible attacks are listed below,

1. **Host Spoofing**

This attack makes use of the Host tracking service, which keeps track of each host connected to the network by associating profile to each host. It either creates a new Host profile or Updates existing profile whenever it receives a Packet-In Message. Current implementations of SDN do not authenticate the Host Update (Via Packet-In Message) messages i.e. they do not verify the authenticity of the host. Any spoofing attacks can exploit this information and change the Host Profile information stored in the controller, consequently affecting the forwarding information. In order to successfully launch the attack, only the attacker needs to know about the MAC address, IP address, VLAN ID of the victim host. He can easily obtain MAC address using ARP requests, since MAC address is not changed in SDN.

1. **Link Fabrication Attack**

This attack exploits the link discovery services which is used to detect the links between the switches to create a map of network topology. The controller sends out LLDP packets to switches which are broadcasted to all the ports. The LLDP packets received by the other switches forward the same to controller thereby detecting the link. There is no mechanism to verify the integrity or origin of the LLDP packets and their propagation. The attacker can exploit this to launch inject a fake LLDP packet i.e. the attacker can modify the specific contents of the LLDP packet, e.g., the DPID field or the port number field, and launch the Link Fabrication Attack. He can also create a false topology network, by relaying LLDP packets from one switch to another

Physical link or tunneling

**DDoS Attacks**

In this paper we are concentrating on DDoS attacks on the data plane.

1. Bandwidth exhaustion

In the event that the switch receive large number of packets in a brief time and on the off chance that it doesn't sufficient buffer limit, then it advances the whole packet to the controller using the entire bandwidth the transfer speed. Thus the switch may drop the bundles.

* Switch buffer size is restricted – If the connection between the switch and the controller is congested, the switch's queue buffer size is filled up, a Packet-In message can't be sent consequently bringing about packet loss.
* High inertness – Congestion may present inactivity in the control channel. In the event that the switch does not get a Flow-Mod reaction inside of a certain time span, it disposes buffered packet.

1. **Flow table**

Switches are restricting in memory to store the flow rules. At the point when the switch memory is full. As the switch can't introduce any new rule, it sends an OFPT ERROR message to the controller with error code OFPFMFC TABLE FULL. It then drops this packet. The timeout values for the flow rules have a noteworthy effect. A low-timeout guarantees that the table is not full but rather includes an overhead as it corresponds as often as possible with controller while a high-timeout rapidly tops off the flow table.

**Suggested Solutions**

1. To keep an assault on the control plane transfer speed: pose a limit on the quantity of packets sent to the controller.
2. To keep an assailant from topping off switches' flow tables: configure an optimum idle time out for the flow rules and flow aggregation.
3. Host spoofing attack can be avoided by confirming the Legitimacy of Host Migration. The controller must get a Port Down sign before the host movement wraps up and later the host element should be inaccessible in the past area after the host movement
4. Authenticate the LLDP packets and the switch should only forward the packets to the other connected switches instead of broadcasting on all ports.

**DDOS Simulation and Prevention**

DDoS attack is simulated using tools like Mininet to create a network topology, scapy to create network packets and Pox to run the controller. The main idea behind detection of DDoS attack is entropy. Entropy is measure of disorder ness in a system. Under normal operation the entropy of the network falls within a certain acceptable range, but when under DDoS attack there is sudden and large drop in the entropy value. This sudden drop indicate suspicious behavior which is used here.

Along with pox controller, a python script is run which continuously monitors the entropy of the network and also keeps track of the switches and the number of packets received on each of the ports. When entropy reduces below a certain permissible threshold, then depending on the number of packets transferred via the port of a switch, port can disabled or a flow medication message could be sent to drop all the incoming packets for a certain period of time.

# Component1

1. Read the IEEE papers to gather the nature of attacks and current techniques to overcome the attacks.
2. Read the recent tech talks and conference outcome to address new threats and possible solutions.
3. We also documented the highlights of each paper pros and cons.
4. Explored OpenFlow protocol and Pox controller.

# Component2

1. Used Mininet tool to simulate network environment.
2. Used scapy to create network packets.
3. Carried out demo exercises and simulated SDN network topology to analyze the working and performance of each component host and switch
4. Used Pox as the SDN controller to connect with network created using Mininet

# Component3…

1. Learn about the flow tables.
2. Python script to register for PacketIn, FlowStatistics event.
3. Python script to measure entropy of the network for each packet
4. Create a dictionary to keep track of number of packets received on a particular port
5. A timer function to periodically check the number of packets and the entropy threshold.
6. Send flow modification message for the incoming packets of a particular port in a switch

# Weekly Progress

## Week1 Progress

1. Read 5 IEEE papers each on discovering how prone/vulnerable SDN is for security threats.
2. Read and discussed in brief the possible attack impact and nature of cause.
3. Read the upcoming new attacks and possible measures the SDN security is industry trying to overcome.
4. Discussed among the team based on understanding the strengths and weakness of each paper.
5. Discussed how things work and the protocol used in the north bound traffic which is the flow of packets between SDN controller and Switch[9].
6. Read and became familiar on specification of “Openflow” protocol version 1.4.
7. Analyzed how the south bound traffic works between network devices and virtual switch.
8. Gathering latest discussions from conferences and tech talks.
9. Attending Cisco SECCON event on 11/04/2015 in the hope of new idea to solve the problem and provide a solution to SDN security.

## Week2 Progress

1. Read the next set of 5 IEEE papers on SDN attacks.
2. Look in each paper to discover how prone/vulnerable SDN is for security threats.
3. Read and discussed in brief the possible attack impact and nature of cause.
4. Read the upcoming new attacks and possible measures the SDN security is industry trying to overcome.
5. Discussed among the team based on understanding the strengths and weakness of each paper.
6. Discussed on the nature of attack performed and possible outcomes.
7. Discussed the current mechanism used to control and defense to the attack.
8. Simulated the network topology by creating mesh, tree network in mininet environment.
9. Analyzed the flow rules and switch configurations.
10. Simulated demo architecture in mininet tool to understand things better.
11. Studied the linux/unix configuration commands[refer command slides in shared folder]
12. Analyzing how the threat vectors are breaking into SDN security layer.
13. Working extensively on making the SDN secure from attacks.
14. We are extensively working and concentrating on host spoofing (host-host) attacks, DDOS and Man in the middle attacks.
15. Trying to mock the attack and give the possible robust solution to make the network topology not to be targeted by attackers.

## Week3 Progress

1. Read the next set of 5 IEEE papers on SDN attacks.
2. Look in each paper to discover how prone/vulnerable SDN is for security threats.
3. Read and discussed in brief the possible attack impact and nature of cause.
4. Read the upcoming new attacks and possible measures the SDN security is industry trying to overcome.
5. Discussed among the team based on understanding the strengths and weakness of each paper.
6. Concentrating to generate Eavesdropping and DDOS attack in the SDN.
7. Created the topology like 2 SDN controller with 2 switches each and each switch has 2 hosts
8. Checked flow rule.
9. Exploring the DDoS attacks that can be launched on the data plane and control plane, which is discussed in brief in the design.
10. Simultaneously we are working to simulate DDOS attack in the SDN network using Mininet and Pox controller.
11. Learning how entropy can be used to detect DDoS attack.

● Week4 Progress

* 1. Discussed in brief the possible attack impact and nature of cause.
  2. Concentrating on OpenFlow protocol to understand the different types of packets that are exchanged between the controllers and networking devices.
  3. Learning about the Open Daylight controller which is the most widely used open source controller for SDN applications.
  4. We were not able to do a host spoofing attack, so we are still working on that.
  5. Run sample examples from pox controller to under the basic flow.
  6. Write python script to create network packets using scapy.
  7. Used random function to create random source addresses so that the switch forward the packets to the controller.
  8. Updated python script to register for the packetIn event, so that information about the each packet like the port, source and destination mac address, switch ID are tracked.
  9. Updated script to measure entropy for each packet.
  10. Register for timer function, to periodically check the incoming packet count
  11. Create a dictionary to have mapping between the switch ID, ports and the corresponding number of packets on the respective ports.
  12. Updated the script to send a flow modification message to ignore the incoming packets to prevent DDoS attack.