Intrusion Detection, Prevention and Mitigation for Software Defined Networks – using Floodlight Controller.

Author: Pratik Lotia

- Background:

Since its inception in 2007 at the Stanford University, Software Defined Networks (SDN) is the most promising concept which is expected to replace a major part of the traditional network engineering. SDN aims at enhancing programmability and automation of traditional networks. Currently, most of it is done using Network Function Virtualization (NFV) which acts as an overlay network on the traditional network. SDN provides flexibility to change network configuration for different types of network scaling [1]. The abstraction of application, control and data planes provides easy configurability and reduces total cost of ownership (TCO) by ensuring full potential use of common off-the-shelf (COTS) hardware. SDN bridges the gap between network engineers and developers. It is possible for both teams to work together to develop efficient and flexible networks. However, it has been ten long years since SDN was introduced and still it has not yet been implemented on a large scale. The problem is analogous to the problem faced for wide-scale implementation of IPv6 protocol -- Security. The problem is that, SDN is yet to be fully analyzed and understood from the Security point of view. SDN vulnerabilities are yet to be properly identified, let alone patching them and preventing exploits. This project aims at using traditional security monitoring tools to identify attacks on the Data plane and block suspicious traffic with an additional option to analyze the suspicious traffic by redirecting it to a Honeypot.

- Relevance:

According to Morgan [2], cybercrime is expected to reach $6 Trillion by 2021. On an average [3] it take over 3 months to detect a breach and network intrusions while remediation takes around 5-6 months. Around $160 million records have been compromised in 2015 alone. The cloud security market is expected to reach $12 billion by 2022. IoT (Internet of Things), Artificial Intelligence, Self-driving cars, Machine Learning, Big Data and Fin-techs (Financial technology) will be the prime target for attackers. All these sectors consist of automation and centrally administered configuration. These will be using NFV or SDN in some form or the other. Hence, it is very important to identify threats as soon as possible and also block suspicious traffic before it compromises private data.

Over the past few years, there have been several attempts to develop a plan to secure SDN architecture at various planes. Some have intended to develop intrusion detection systems (IDS) while have some have proposed developing firewall system for SDN. A majority of these have been developed to be compatible with controllers like Ryu & POX while are not widely used anymore. There have been attempts to develop such systems for commonly used open-source controllers such as Floodlight and OpenDaylight, however, they have been developed for private usage for their respective companies/projects and not much is available for free public usage.

This system can be used in Data Center networks as well as Service Provider networks. The exact working and compatibility for the respective environments needs to be analyzed properly before

implementing this. This project provides test cases to show the working of the system. For extensive and full scale usage the user needs to develop signatures for intrusion alerts and according plan to block each type of packet.

- Intrusion Detection System:

Intrusion Detection System [4] (IDS) is an application which monitors traffic on a network or a host for any type of suspicious activity. It generally consists two types – signature-based detection and anomaly based detection. The former uses predefined patterns of traffic to decide whether an incoming/outgoing packet is suspicious. It can raise alerts which can be sent to the network administrator or can be logged to a Security Information and Event Management (SIEM) system. The latter uses machine learning to detect a significant change in the traffic behavior.

This project can use either of the types. However, it has been tested using the former type, especially using a common software called Snort.

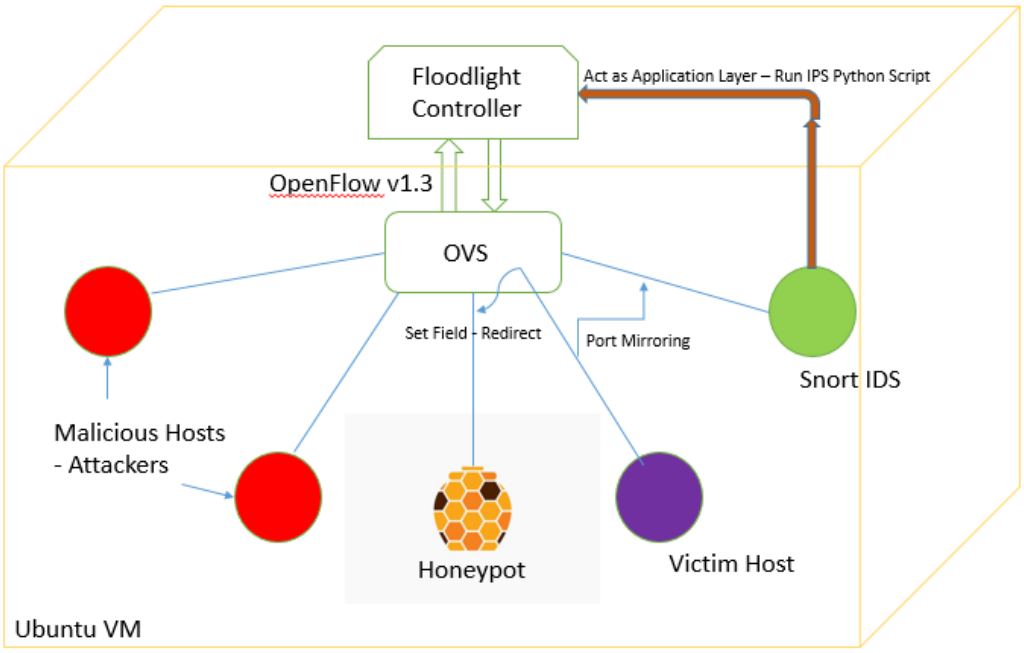
- Intrusion Prevention System:

Intrusion Prevention System [5] (IPS) makes use of IDS to identify the parameters of suspicious traffic and accordingly blocks that traffic by building Firewall rules or Access Control Lists.

- Mitigation System:

IDS system raises a lot of false positives. IPS using such IDS can block even good traffic. Also, using IDS and IPS it is impossible to know the exact intention of the attacker and its purposes to attack the network. To solve this, this project can use a Honeypot system where our IPS can redirect suspicious traffic to a Honeypot machine instead of directly blocking it. The Honeypot will in turn determine whether the traffic is really suspicious or it is just a false positive.

- Working of the project & implied Benefits:
  - The project uses an Ubuntu v16.04 64-bit Operating System – VM.
  - Floodlight, Mininet and Snort (along with dependencies) are installed and configured on the VM.
  - For testing purposes, all rules of Snort except the local user rules have been disabled.
  - Snort local rules file is configured with signatures of traffic like ICMP and HTTP.
  - Floodlight controller is started and it runs on <localhost_IP> on port 8080.
  - Mininet topology is initiated with 1 switch, 5 hosts having IP base of 10.0.0.0/24 with their respective X-terms and Open Virtual Switch interacting with Floodlight Controller using OpenFlow protocol version 1.3.
  - The switch is set as a compatible OVS bridge.
  - Port mirroring is enabled on OVS such as traffic generating from and directed to host 4 is mirrored on host 5’s network interface.



- For testing in real world environment, a hybrid router can be used in place of OVS and Router IP Traffic Export (RITE) can be enabled to mirror traffic.
- Due to mirroring, host 5 actually acts as a network sniffer.
- A security monitoring tool is started on host 5 to analyze all incoming traffic (both mirrored and non-mirrored). For this project, Snort tool has been used for testing. Alternatively tools for flow analysis like NetFlow & SiLK can be used too.
- For practical purposes, host 5 can be made inaccessible from the public environment so that only mirrored traffic is active on host 5 interface. This way there is no confusion about whether the traffic is for host 5 or for host 4. It will always be for host 4 – just mirrored on host 5.
- The alerts are directly logged into a text file.
- This Host can now act as an application layer for pushing flow entries to the switch via the Controller.
- A python script reads the alerts file and determines parameters such as source and destination IP, source and destination ports, protocols and Ethernet type, flags and other header values.
- The python script is configured to dynamically make flows entry matching fields using these parameters. These are written in JSON format. The flow entry supports all the theoretical variations supported by OpenFlow version 1.3 such as groups, buckets, timers, meter bands, modify fields, push & pop tags, timers, etc.
- The python script ensures to add a hard timeout value of a few seconds and a higher priority value in the flow entry. It is necessary to specify the switch DPID (data path identifier) and a unique name for the flow entry.
- Not specifying any action parameter will automatically notify the switch to drop any matching packet.

- This JSON format is properly encoded and sent as a POST message in a proper format using REST API directed to the controller IP and port which in turn pushes them to the OVS.
- In this way, suspicious traffic can be blocked on the Data Plane. This forms our IPS part.
- IDS can generate several false positives and this can lead to blocking of good traffic. Hence, a hard timeout of a few seconds is added so that the suspicious traffic is only temporarily blocked. This would considerably slow down attackers while only temporarily block good traffic.
- Alternatively we can have an additional functionality. Instead of blocking the traffic for a specific match field, it is possible to modify certain fields of the packet and alter its behavior. We can change the destination IP so that the traffic can be directed to a different system.
- Using this, we can redirect the traffic to a Honeypot system. The traffic can be analyzed here and we can understand the intent of the sender of the suspicious traffic. The honeypot system varies from application to application.
- Hence attacks can be mitigated using this method.
- This concludes the working of this project.

About the author:

Pratik Lotia is currently a Graduate student (Masters) in Network Security and SDN Automation in the Interdisciplinary Telecommunications Program (ITP) at University of Colorado Boulder. He has a Bachelor of Engineering (B.E.) in Electronics and Telecommunications from University of Mumbai.

Prior to this, he has been involved in several research projects ranging from Renewable Energy to Industrial Printing technology. He has been recognized in the Top 50 Innovators of India by the Govt. of Indian and Lockheed Martin.

He has been a founder of a start-up called VenTech which is involved in Automation of Embedded systems and Industrial Machinery. He has been selected in Top 25 upcoming entrepreneurs by the TATA Group.

References:

1. https://f5.com/solutions/service-provider/reference-architectures/network-functions-virtualization
2. http://www.csoonline.com/article/3153707/security/top-5-cybersecurity-facts-figures-and-statistics-for-2017.html
3. https://techbeacon.com/38-cybersecurity-stats-matter-most
4. https://en.wikipedia.org/wiki/Intrusion_detection_system
5. https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-prevention-system-ips
6. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343653/How+to+Write+a+REST+Application
7. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343539/Floodlight+REST+API

8. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/4882438/ACL+Access+Control+List+REST+API
9. https://gist.github.com/electricjay/519f7741e8e22b65a073
10. https://github.com/floodlight/floodlight/blob/master/apps/circuitpusher/circuitpusher.py
11. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343539/Floodlight+REST+API
12. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343518/Static+Entry+Pusher+API
13. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343614/Firewall+REST+API
14. http://www.cs.duke.edu/courses/fall14/compsci590.4/notes/slides_floodlight_updated.pdf

TLEN 5830-001 – Next Generation Networks

Final Project
SDN based Intrusion Detection, Prevention and Mitigation

University of Colorado Boulder
Interdisciplinary Telecommunications Program

Pratik Lotia, Graduate student

Lab Summary

Securing Networks against malicious users is very important. This is to prevent leaking any confidential data and prevent altering the normal behavior of the network/application.SDN is a networking concept that aims to centralize networks and make network flows programmable, and NFV focusses on virtualized network functions. SDNFV can be used to manage networks better and reduce CapEx/ OpEx.

SDN-based Intrusion Detection, Prevention and Mitigation system uses standard security monitoring tools to create secure and error-free systems that can be deployed, managed and manipulated with ease in the industry.

In this lab, students will deploy, study, and evaluate simple IDS to increase understanding of threats, and will develop their own IPS application how a programming application can be used to block suspicious traffic. In the end, students will attempt to mitigate threats using Honeypot system.

Objective 1 – Connect to the Ubuntu VM and Install/Run Floodlight controller and configure Snort application.

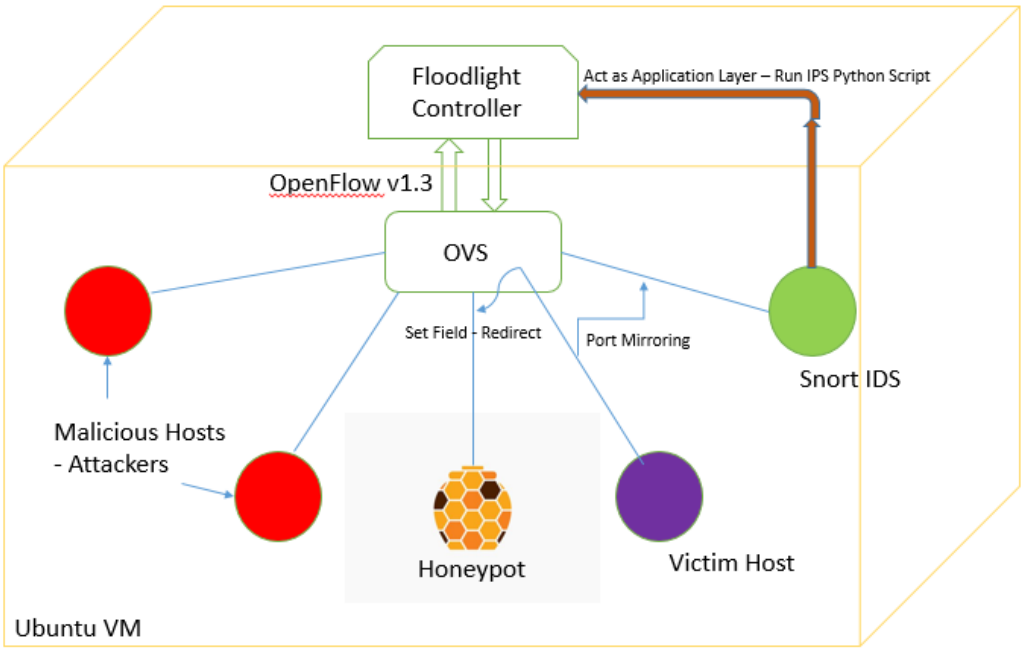
1. You will first be in the Linux GNOME environment of the Ubuntu VM. Verify that the VM has internet connectivity by pinging 8.8.8.8.
2. You can have Floodlight installed inside the Ubuntu VM or you can run it separately. If you are planning to run the Floodlight controller separately from another VM, ensure that the Floodlight controller has connectivity to the Mininet VM.
3. Ensure that you have a Floodlight controller installed. What command did you have to use to check that? **[2 points]**
4. Modify the Snort configuration file so as to disable all in-built rules and enable only local rules and the threshold configuration file.
    a. How did you do it? Post screenshots **[10 points]**

       Hint: Using '#', comment out all lines starting with 'include $SOME_VAR…' found at the end of the configuration file – except the lines having 'include $RULE_PATH/local.rules' and 'include threshold.conf'
    b. Replace 'any' with the IP address of your VM (NAT address) in the line 'ipvar HOME_NET any'. Post screenshot. **[2 points]**
    c. Also comment out lines for preprocessors, whitelists, etc starting with 'var …'. **[1points]**
    d. Check whether Snort is configured and working properly. Which command did you use? You should get a success message after execution of that particular command. Resolve errors if any. **[5 points]**
    e. Configure a file to use Snort as a daemon. How did you do it? Attach the configuration file. Post screenshots. **[15 points]**
    f. Enable the snort service. Post screenshot showing that the service is running **[5 points]**

g. Create a local rules file under /etc/snort/rules. Define signatures for HTTP and Ping traffic to or from the VM. **[16 points]**

Alternatively **SA / TA** are advised to offer pre-configured snort configuration file OR create a VM image for download having Snort working properly.

Objective 2 – Run Mininet and detect suspicious traffic using IDS.



---

1. Initialize the Mininet network topology comprising of five hosts and one switch, and with a connection to the Controller. Ensure Xterm for all nodes.
    a. What command did you use? **[2 points]**
2. On the switch X-term, configure port mirroring for traffic to and from host 4 to get mirrored to host 5. What commands did you use? **[8 points]**

3. Using respective X-term, start Snort on Host 5 such that alerts are logged to the terminal. What command did you use? **[5 points]**
4. Send/Receive Ping traffic to host 4 from host 1 or 2. Next, configure host 4 as a HTTP Web Server on port 80. Send a GET request to the Web Server from host 1 or 2. Ensure you receive 200 OK response. Post screenshots. **[4 points]**
5. Alerts should be generated and seen on host 5. Post screenshots. **[20 points]**

Objective 3 – Write a Python script to block suspicious traffic using flow entries.

1. Restart Mininet using same topology. Do port mirroring like above. Now on host 5 instead of logging alerts to terminal, log the alerts to a text file. What command did you use? **[5 points]**
2. Write a Python script to parse the log file, identify the parameters of the traffic – such as source & destination IP, ports, Ethernet type, etc. Create a JSON data in the Python script of all the required matching fields. Use suitable encoding to send the flow entries as POST message to Floodlight Controller using correct IP, port and path. Keep the priority less than 30,000. Attach the script. Execute the python file. **[40 points]**
3. Now recreate the same type of Ping, HTTP traffic of the last section. Do you get successful replies now? Post screenshots. **[5 points]**
4. What are the mandatory fields in the flow entry configuration pushed to Floodlight? **[3 points]**
5. Restart this section. However, in the python script add a hard timeout of 20 seconds in the flow entry JSON data. Execute point number 3 of this section after 20 seconds. What replies do you get? Post screenshots. **[5 points]**
6. IDS raises a lot of false positives and it might block good traffic. How did we avoid this problem in this lab? **[5 points]**

Objective 4 (Extra credit) – Delete Flow entries.

1. Remove the timer field from JSON data in Python script. Add a function so as to delete the flow entries. Attach modified python file. **[10 points]**
2. What parameter did you use to delete flows? **[5 points]**

Objective 5 (Extra credit) – Create a simple Honeypot.

1. Modify JSON data so as to redirect traffic destined for host 4 to host 3 which is your honeypot host. (This is different from port mirroring. You are not mirroring, instead you are completely redirecting the traffic such as host 4 will not receive it). Attach modified python script. **[10 points]**
2. Use your imagination and creativity to make a simple Honeypot application. It can be as simple as differentiating good traffic from bad traffic for one particular test case. Or it can be like an HTML

form on a web application such that it differentiates between humans and bots. You will have to show and explain this to SAs. Attach all relevant files and include step by step working and execution. Accordingly push flow entries to delete the blocking flow entry only if the traffic turns out to be good. Alternatively, instead of deleting add a flow entry with a higher priority to allow the traffic. **[40 points]**

Total Score = _____/221 (156 + 65 Extra credit)