SDN ENHANCED CAMPUS AUTHENTICATION SYSTEM

Prince Antony L
Student BE 4th year I
Dr. Mahalingam college of Engineering and Technology

To secure the enterprise network, even when using strong Authentication Techniques .Providing an way to strengthen the network security level to next generation level which is more effective. In traditional network architecture these solutions are difficult to implement where there are already many loopholes in this network security system . One of the possible ways to implement the controllability of traffic flow in standard networking method by applying OpenFlow driven SDN architecture and commodity access switches, is described in this paper. The centralized control enabled by SDN will ultimately result in security-defined routing and other SDN security strategies that could forever change how we defend the network and the applications or data running across it

Keywords: *SDN , OpenFlow*

I. INTRODUCTION

The aim of the SDN is to provide open interfaces enabling development of software that can control the connectivity provided by a set of network resources and the flow of network traffic though them, along with possible inspection and modification of traffic that may be performed in the network. SDN will be used to describe an innovative approach to network construction, configuration and monitoring.SDN is a novel network architecture in that it provides a segmented design where the controlling entities of the network (control plane) are detached from the forwarding functionality (data plane), allowing for efficient and granular control over the data flow. As result, a logical centralization of the network is possible.

A. Advantages of Software Defined Networks

[1] Highly available bandwidth on demand
[2] Cost-effective operations.
[3] True, multivendor interoperations/connectivity
[4] Improved Command and Control.
[5] Automatic Provisioning

These improvements are difficult to achieve in traditional networks. Traditional networking and systems administration refers to the static and laborious techniques for designing, managing and deploying network elements, servers and services. Examples include the need to interface with each element manually, attempting to determine traffic patterns via telemetry from servers or protocols, deploying virtual machines and applying security mechanisms.

II. OBJECTIVE OF THE STUDY

[1] To create a network topology with SDN controllers, Servers and nodes .
[2] Using Simulator to perform the Authentication process which is executed by controller
[3] Analyze how the communication is established between the nodes after the authentication

III. METHODOLOGY

The simulation tools such as ns3 is used for the simulation and OpenFlow switch such as ofswitch13 is used. The module provides a complete OpenFlow switch device, and the OpenFlow controller interface. The switch is fully functional, while the controller interface is intended to allow users to write more sophisticated controllers to exploit the real benefits offered by SDN paradigm.

The OpenFlow 1.3 controller application interface, namely OFSwitch13Controller, provides the basic functionalities for controller implementation. It can handle a collection of OpenFlow switches, as illustrated in Figure The OFSwitch13Controller internal structure. For constructing OpenFlow configuration messages and sending them to the switches, the controller interface relies on the dpctl utility provided by the ofsoftswitch13 library.

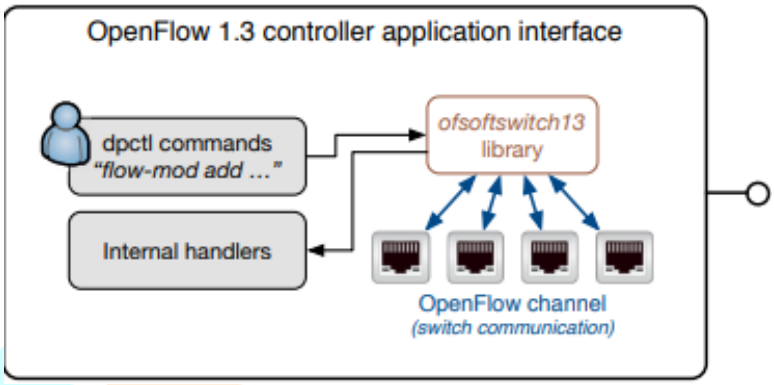

Figure 1: The OFSwitch13Controller internal structure

IV. EXPERIMENTAL SETUP

The OFSwitch13 module was designed to work together with the ofsoftswitch13 library, providing an interface for interconnecting the ns-3 simulator to the library. To this end, the ofsoftswitch13 project must be compiled as a static library and get proper linked with ns-3 simulator. Currently, the user must download and compile the code manually. Follow the instructions below to compile and link the ns-3 simulator to the ofsoftswitch13 library. These instructions have been tested on Ubuntu 16.04 LTS. Other distributions or versions may require different steps, specially regarding library compilation

A. Compiling the library

Before starting, install the following packages on your system:
$ sudo apt-get install build-essential gcc g++ python git mercurial unzip cmake
$ sudo apt-get install libpcap-dev libxerces-c-dev libpcre3-dev flex bison
$ sudo apt-get install pkg-config autoconf libtool libboost-dev

First, it is necessary to compile the ofsoftswitch13 as a static library. The ofsoftswitch13 code relies on another library, called NetBee (http://www.nbee.org), which is used to parse the network packets. So we need to compile and install them in the proper order.
Download the NetBee and unpack the source code:

$ wgethttps://bitbucket.org/ljerezchaves/ofswitch13-module/downloads/nbeesrc.zip
$ unzip nbeesrc.zip
Create the build system and compile the library:
$ cd netbee/src/
$ cmake .
$ make

Add the shared libraries built to your library directory, configure dynamic linker run-time bindings, and copy the include files:
$ sudo cp ../bin/libn*.so /usr/local/lib $ sudo ldconfig
$ sudo cp -R ../include/* /usr/include/

We are done with the NetBee library. Now, let's proceed with the ofsoftswitch13 code. Clone the repository and update to proper (preferably latest) release tag at the ns3lib branch (here, we are using v3.1.x): 8 OpenFlow 1.3 Module Documentation, Release 3.1.0

$ git clone https://github.com/ljerezchaves/ofsoftswitch13
$ cd ofsoftswitch13
$ git checkout v3.1.x
Configure and build the library (don't forget to add the --enable-ns3-lib during configuration process):
$ ./boot.sh $ ./configure --enable-ns3-lib
$ make

B. Linking the library to the simulator

It's time to download a recent (preferably stable) ns-3 code into your machine (here, we are going to use the mercurial repository for ns-3.26):

$ hg clone http://code.nsnam.org/ns-3.26
$ cd ns-3.26

Before configuring and compiling the simulator, download the OFSwitch13 code from the module repository and place it inside a new /src/ofswitch13 folder. Update the code to the latest stable version (here, we are using 3.1.0):

$ hg clone https://bitbucket.org/ljerezchaves/ofswitch13-module src/ofswitch13
$ cd src/ofswitch13
$ hg update 3.1.0 $ cd ../../

Also, you need to patch the ns-3 code with the appropriated patches available under the ofswitch13/utils directory (use the patches for the correct ns-3 version):

$ patch -p1 < src/ofswitch13/utils/ofswitch13-src-3_26.patch
$ patch -p1 < src/ofswitch13/utils/ofswitch13-doc-3_26.patch

The ofswitch13-src-3_26.patch creates the new OpenFlow receive callback at CsmaNetDevice and virtualNetDevie, allowing OpenFlow switch to get raw packets from these devices. These are the only required change in the ns-3 code to allow OFSwitch13 usage. The ofswitch13-doc-3_26.patch is optional. It instructs the simulator to include the module in the ns-3 model library and source code API documentation, which can be helpful to compile the documentation using Doxygen and Sphinx.

Now, you can configure the ns-3 including the --with-ofswitch13 option to show the simulator where it can find the ofsoftswitch13 main directory:
$ ./waf configure --with-ofswitch13=path/to/ofsoftswitch13
Check for the enabled ns-3 OpenFlow 1.3 Integration feature at the end of the configuration process. Finally, compile the simulator:
$ ./waf
That's it! compilation with OpenFlow 1.3 capabilities is successfully completed
After execution completed ..use following command to run the simulation file
$./waf - - run ofswitch13-qos-controller
It will execute the code ..after that it will create animation file named qosctrl-netanim.xml

Run the above file using netanim tool as below

$./NetAnim

It will open window ..now select the .xml file ..it will show the simulation

V. FIGURES

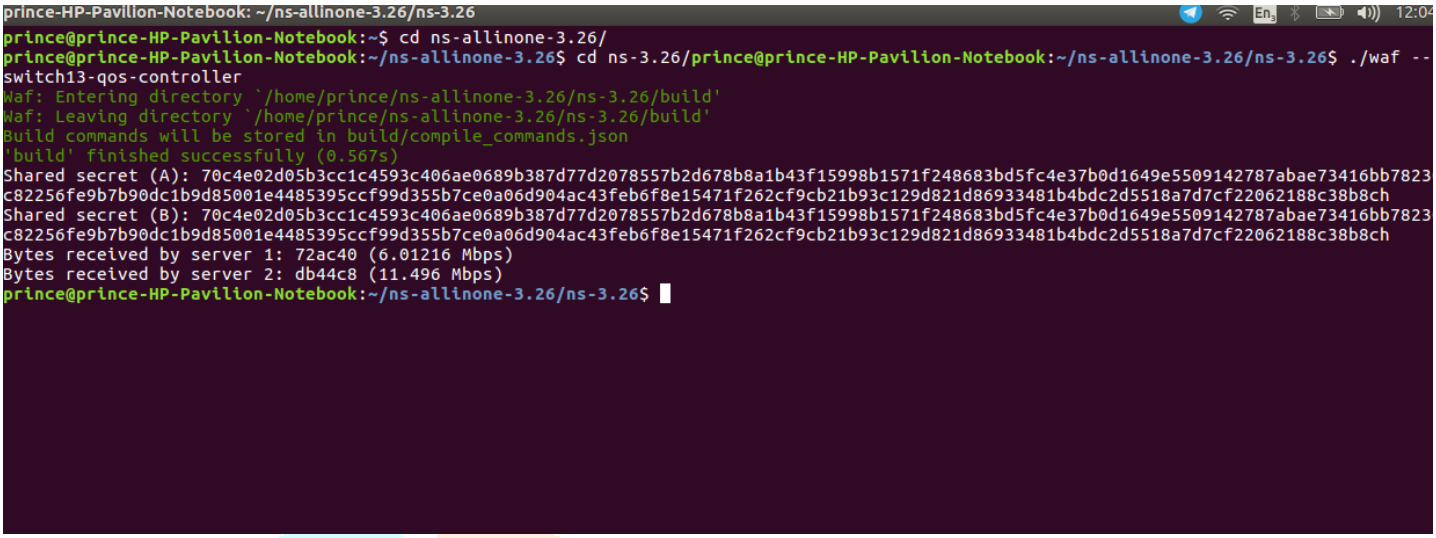

Figure 2: Compiling library files to run the Simulator

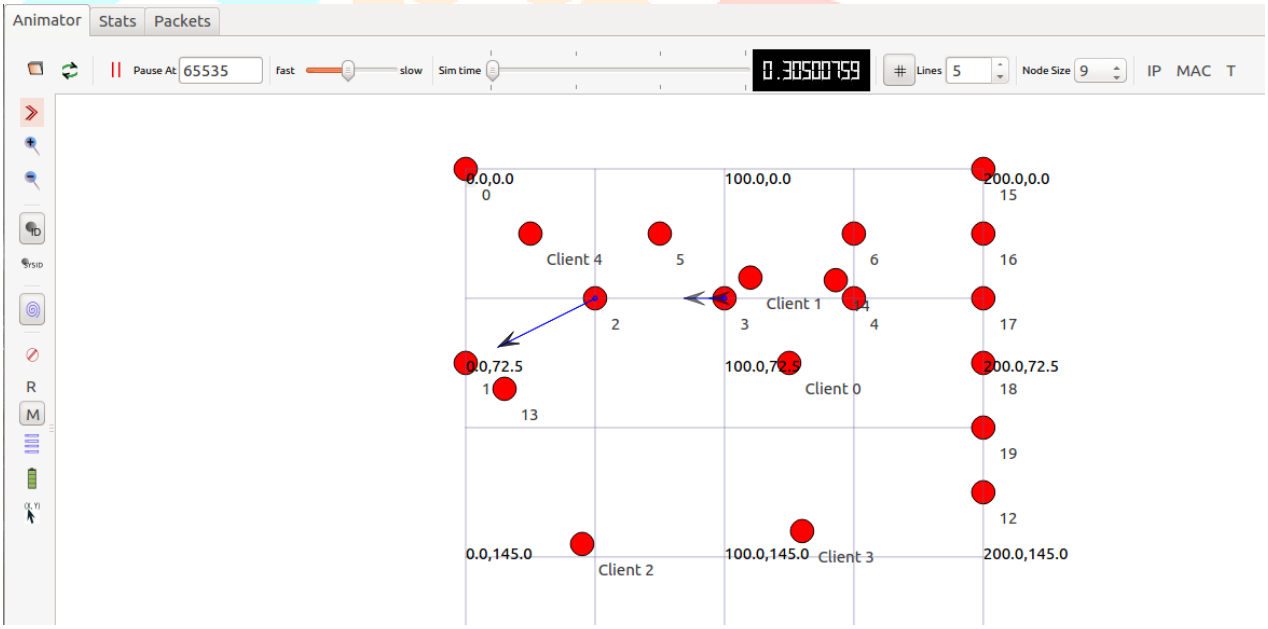

Figure 3: The simulation of Authentication system using SDN Controllers

VI. RESULTS

The Simulation of Network is established and this module can be implemented in the real case scenario.

V. CONCLUSION

[1] To register and authenticate the switches to the controller.

[2] To authenticate the hosts and to bind them to the switches (and ports).

[3] To provide the authentication of users.

[4] To manage data flows and users/hosts mobility, were developed.

REFERENCE

[1] The reference [Fernandes2014] (in Portuguese) describes the details on the ofsoftswitch13 software switch implementation

[2] The reference [Chaves2016] presents the OFSwitch13 module, including details about module design and implementation. A case study scenario is also used to illustrate some of the available OpenFlow 1.3 module features

[3] The reference [Chaves2015] is related to the integration between OpenFlow and LTE technologies. The ns-3 simulator, enhanced with the OFSwitch13 module, is used as the performance evaluation tool. This is the first published work including simulation results obtained with the OFSwitch13 module.

[4] H. Okhravi and D. Nicol, "Applying trusted network technology to process control systems", in IFIP Int. Federation for Information Processing, vol. 290, Critical Infrastructure Protection II, M. Papa and S.Shenoi, Eds., Springer, pp. 57-70, 2008.

[5]Ch. Leidigh, "Fundamental Principles of Network Security", APC ,pp. 14, 2005.

[6]"802.1X: Port-Based Authentication Standard for Network Access Control (NAC)", Juniper Networks, 2010.