Homework 5 – CSC/ECE 573-001, Fall, 2015

This is a team homework, assigned to the same teams as the project teams. Project leads should submit a single deliverable on behalf of the team.

There are two mini-projects in this homework – one focused on OpenFlow, and one on kernel modules. Each team may submit any one they choose. However, it must be the one focused on the topic that their course project does not focus on.

The OpenFlow mini-project has three questions. The Kernel Modules mini-project has a total of seven questions, two for Task 1, and five for Task 2. For questions that require you to write down an answer, or submit logs etc., submit your answers as a PDF file in accordance with instructions on the course website. In addition, see specific instructions in some questions regarding submitting additional deliverables such as code (also through the WolfWare submit locker).

A team may choose to submit both mini-projects, for extra credit. The normalized score on the mini-project which scores less will be used to replace the worst score of each team member, if that worst score is lower.

OpenFlow Mini-Project

1. The main goal of this exercise is to introduce to OpenFlow, its basic semantics, understand OpenFlow specific terminologies and mainly to facilitate you in choosing your course project.

Read OpenFlow specification v1.3.0 (at least sections 1 through 6) and answer the following questions briefly. You should add citations or references as you see fit.

(a) What is a pipeline processing?

(b) What are flows? What is a flow table?

(c) What's the difference between local and normal ports? In what situation one might use it?

(d) What does hard/soft timeout mean in a flow? Where would they be useful?

(e) What is meant by standalone and secure modes in OpenFlow switch?

2. Create a topology as given in the "Topology" section of the How-to OpenFlow document. Install Open vSwitch (see Installation section for more details). The distribution comes with ovs-ofctl, an utility to program/query the OpenFlow switch.

Query the switch for available features. Interpret and explain the output. Refer to the OpenFlow specification document.

Install a flow to enable ICMP echo requests/replies. Hint: This would involve flows for ARP resolution and ICMP packets.

Modify the flow's idle and hard timeout to 20/30 seconds. Stop sending ICMP traffic from h2 to h1 for 10 seconds. Observe the switch flow table state after 30 seconds.

Explain your interpretation of flow table before changing the flow timeouts and after changing it (changing it + waiting for 30s).

In your answer, include the OpenFlow utility commands you used, the actual output of the command, the flows you have installed, and the flow table output.

3. Locate information about various OpenFlow controllers – such information is openly available in the Internet. Write a brief note on 5 such controllers (3 open source and 2 commercial grade) with proper citations/references. The answer should include the OpenFlow versions that the controllers can support. Add citations and references as you see fit.

Kernel Module Mini-Project

This mini-project exposes you to the power of Linux kernel modules and the Netfilter framework, by asking you to implement a simple yet useful firewall. You are asked to create a network topology consisting of a LAN connected to the "internet" through a firewall. You will implement a simple firewall that will protect the LAN from specific threats.

Consider the following network topology that shows the network at a small organization.

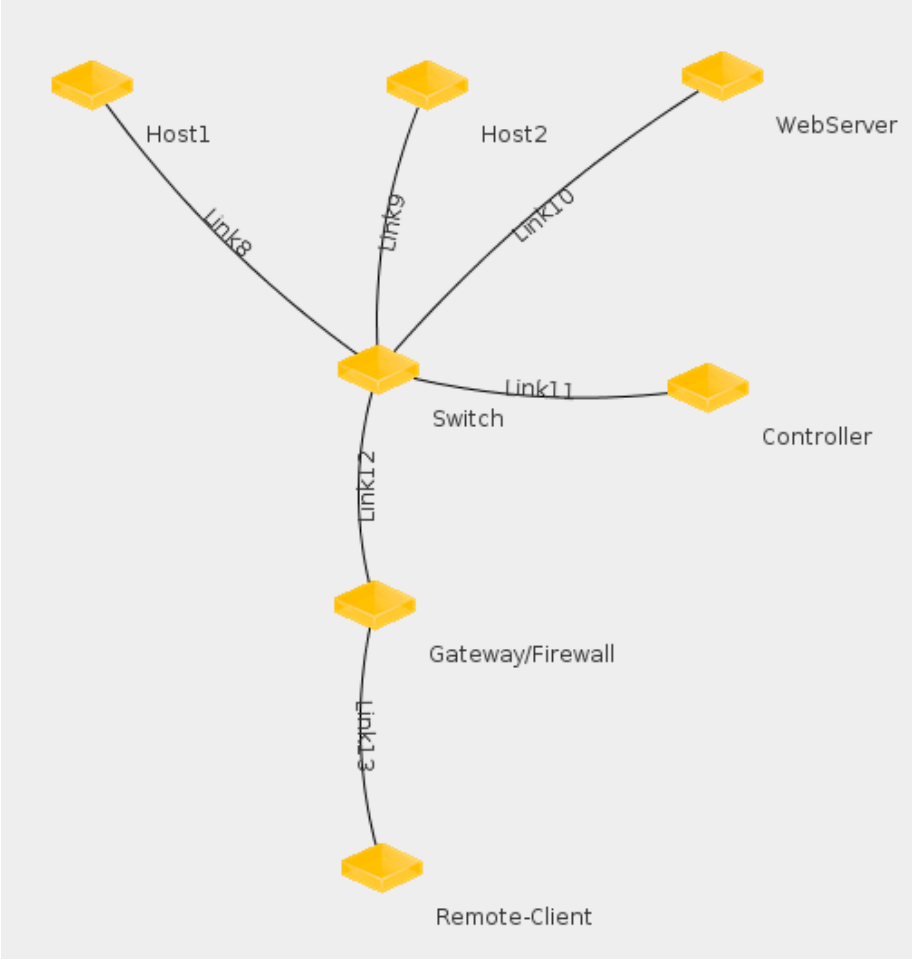

Figure 1: A LAN connected to Remote Client through a switch and a gateway

The local network (LAN) consists of a switch, two hosts and a web server that hosts the organization's public website. The LAN is connected to the outside (Internet) through a single Linux host serving as a gateway. Since the switch is an OpenFlow switch there is also a controller node. This part of the network topology is similar to the OF homework topology (Homework 4). ***This assignment is expected to be completed using GENI.*** Please see the latest updated instructions on using ExoGENI.

Important:

1. Please use the GENI controller previously assigned to your team leader based on their last name. If the team-lead's last name starts with letters A-C use 1st controller, if it is from D-K use 2nd, L-O use 3rd, P-S use 4th, and T-Z use the 5th controller.
2. Please remember to change the ORCA actor registry port number in your .flukes.properties file if you have not already done so (refer to email sent to the class earlier). Otherwise you will not be able to see image options when you create the slice.

Task 1:

1. Configure the hosts, switch, and gateway so that the remote client can ping the web server. Use instructions given with Homework 4 "How to OpenFlow" if you need help to configure the switch.
2. Now install and configure apache web server on WebServer, create a simple html webpage with your name(s), unity id and student id. Access this webpage from Host1 and Remote-Client using curl (curl "http://<webserver-ipadr>/<webpage>") and capture the output.
3. Repeat step 2 for the host "Host2" as the web server.

Questions:

1. Provide summary of the configuration you needed at each node.
2. Provide logs of the pings, and (snippet of) curl output.

Task 2:

You have recently become part of this small organization and want to protect its network from some common attacks. After reviewing common threats you decide that a simple firewall with the following rules will provide your network sufficient protection.

Firewall Rules:

1. Block all ICMP packets coming in from outside except the ones going to the web-server. Though the local hosts should be able to ping outside.
2. Block all ssh attempts from outside.
3. Block port 80 (http) access from outside except for the web-server and test that an internal website on a local host is only accessible from inside.

Implement a simple firewall as a linux kernel module with only 3 rules identified above. Get familiar with the Netfilter framework that provides hooks for you to filter packets and then take actions like drop or accept.

Resources:

1. How to implement a firewall with Netfilter: http://www.linuxjournal.com/article/7184 Note that some of the API given in their code is for old kernel versions and may not work with your Linux kernel. Use Linux source browser link below to lookup the header file and the exact API for your kernel version.
2. Netfilter Project: http://www.netfilter.org/. Also see links to more documentation.
3. Linux Source Browser: http://lxr.free-electrons.com/source/. You can use this as a reference to the Linux headers for your version of Linux. Don't forget to choose the right version of Linux near top.
4. Other examples of Netfilter use: http://www.roman10.net/how-to-filter-network-packets-using-netfilterpart-1-netfilter-hooks/ http://www.roman10.net/how-to-filter-network-packets-using-netfilterpart-2-implement-the-hook-function/

The APIs used in example code here are more up-to-date but the code is more complicated than the linuxjournal.com article.

Required steps:
The article at the first link provided above provides an example (with code) of how to implement a mini-firewall with netfilters. You can use that code as a basis for your firewall. You will have to do the following modifications:

1. Change the code to make it compatible with your version of Linux kernel. Build and run the module to test that it works.
2. Add code to implement the 3 firewall rules provided above. Use printk's with appropriate log level to log details of any packets dropped by your firewall.
3. Test your firewall by sending traffic that exercises the 3 rules. Check logs to see that the packets are dropped appropriately. Now disable firewall and confirm that the packets go through.

Questions:

3. What information can you obtain about the LAN using only ICMP messages from outside (i.e. from client)?
4. Are you able to ssh from one host to another using the interfaces/links in the provided topology? If not, please explain why.
5. For each firewall rule, describe one or more threats against which the rule provides protection.
6. Submit the code you have written along with the makefiles and a README describing your linux distribution (see "uname -a" in HW2). Make sure that the code and makefiles you submit are building.
7. Provide descriptions and logs of your test cases. Each test case should have a title, a description, and the result (log) from running the test. The logs should include logs from the kernel module that shows dropped packets and the reason they are dropped. e.g "Dropped: cause: icmp, interface eth0, dest 10.0.2.5".