University of Crete
Computer Science Department
Lecturer: Prof. Dr. X. Dimitropoulos
TAs: Dimitrios Gkounis, George Nomikos
M. Lakiotakis, G. Vardakis

HY436 - Software Defined Networks
Tasks of **Assignment 3**

Assignment 3

Brief Introduction

In this exercise you will verify what you have learned about FortNOX [1], regarding rule conflict detection, and FRESCO [2], regarding the composition of modular security services for Software Defined Networks. Details on these two frameworks have been presented in the SDN Security lecture on 3/11/2014. Your tasks are the following.

Task 1: Conflict Detection

Assume that you can write flow rules using the following format:

```
srcIP (srcPort) -> dstIP (dstPort) ACTION
```

Here, srcIP, srcPort, dstIP, and dstPort can be "*" meaning "any value". ACTION can be FORWARD, DROP, or SET. The "*" can also appear as the trailing part of a dotted-quad address. For example, 192.168.10.* stands for "any address in 192.168.10.0/24".

(a) Using the notation above, write a set of three rules that do the following:

(I) Allow HTTP traffic from anywhere to 192.168.10.2. This means that every host can access the HTTP service running on 192.168.10.2.

(II) Allow SSH traffic from anywhere in 192.168.10.0/24 to 192.168.10.2. This means that only hosts within the specified subnet can access the SSH service running on 192.168.10.2.

(III) Drop all other traffic to 192.168.10.2.

Assumptions: as a simplification, we are not interested in the rules that deal with the reverse communication (e.g., server to host). Instead, we are interested in the rules that allow (or deny) a host from sending traffic to the server. The rules are prioritized according to the order in which you are writing them.

(b) Using the notation above, write a set of rules that, if installed, would allow *external* SSH traffic to 192.168.10.2, despite the rules that were developed in part (a). *External* traffic could originate from any host.

Assumptions: Assume that you install the rules on an OpenFlow switch which possesses a pipeline composed of multiple flow tables. In our case, assume that the first table of the pipeline contains all the rules developed in part (a). Your task is to populate the rest of the flow tables with rules with SET or FORWARD actions that will be activated after scanning the first flow table. This means that you will craft rules using the SET action that transform the incoming traffic at intermediate tables, from benign traffic that has successfully passed

the first test (rules of part (a)) to malicious traffic bypassing the DROP rule, while being forwarded to the target. Lastly, as a simplification, we are not interested in the rules that deal with the reverse communication (e.g., server to host). Instead, we are interested in the rules that allow (or deny) a host from sending traffic to the server, either directly or after packet header transformations.

(c) Show how the Alias Set Reduction Algorithm catches the conflict.
Hint: Write down the alias sets for both the fRules developed in part (a) and the cRules written in part (b). Check for conflicts between FORWARD and DROP actions present in fRules and cRules.

Task 2: Policy Enforcement via Composing Service Modules

You are sysadmin in a large law firm. You administrate two servers, called "Aral" and "Esso". Aral contains files concerning a lawsuit against Aral and Esso contains files in connection with a lawsuit against Esso. A lawyer's machine is allowed to initiate connections to either server, but once it has made a connection to, let's say Aral, it would be forbidden to ever connect to Esso, and vice versa. (This is a real policy employed in law firms, used to prevent allegations of conflict of interest. It is called the Chinese Wall policy because you may find yourself on either side of the Chinese Wall, but you're not allowed to cross to the other side.)

(a) Can you implement this policy as a static set of OpenFlow rules? If yes, give the rules; if no, prove why not.
Hint: We want a proper proof. No plausibility arguments allowed! You may need to formalize the syntax and semantics of (a simplified version of) OpenFlow for this.

(b) Can you implement this policy using the proposed Fresco modules as they were described in the lecture?

(c) Independent of your answer to (b), design your own Fresco module of type "ChineseWall" that can implement the Chinese Wall policy:

- What kind of state must an instance of ChineseWall keep?
- How many inputs, outputs, and events does it have?
- It would be natural to assume that the IP addresses for Aral and Esso would be parameters of such a module. Show that this is not needed.

Hint: It does not matter whether the lawyers initiate the connections using UDP, TCP, or any other transport protocol. From the moment that the "ChineseWall" module has seen a connection attempt towards one of the servers, all other connection attempts to the other server are denied.

Please make sure that you have already studied the SDN security lectures slides and both FortNOX [1] and FRESCO [2] papers.

Good luck!

Acknowledgments

This assignment is based on material from a similar course at ETH Zurich. The initial feedback for the formulation of this exercise has been provided by Dr. Stephan Neuhaus from ZHAW.

References

[1] Philip Porras, Seungwon Shin, Vinod Yegneswaran, Martin Fong, Mabry Tyson, and Guofei Gu. A Security Enforcement Kernel for OpenFlow Networks. In *Proceedings of HotSDN*, 2012.

[2] Seungwon Shin, Phillip Porras, Vinod Yegneswaran, Martin Fong, Guofei Gu, and Mabry Tyson. FRESCO: Modular Composable Security Services for Software-Defined Networks. In *Proceedings of NDSS*, 2013.