SDN Firewall with POX

Implementation Overview

OMSCS/OMSCY CS 6250 Computer Networks

What are we implementing?

The purpose of this phase of the project is to code (using the POX OpenFlow API) a configurable firewall that will block or allow traffic based on various packet header parameters (please watch the Wireshark tutorial video).

These parameters will be coded in a file named configure.pol that you will create in a separate step. This file will provide a combination of packet header parameters possibly including any of the following: Source/Destination IP Address, Source/Destination MAC Address, IP Protocol, and Source/Destination Application Ports.

Your implementation code should be agnostic and handle any given input from the configure.pol file. PLEASE NOTE THAT YOU WILL NEVER BE TESTED WITH A FILE THAT HAS INVALID ENTRIES OR ILLEGAL POLICIES.

Georgia Tech
CREATING THE NEXT

How to access data from configure.pol?

The content of the configure.pol will be provided as a list of dictionaries with each entry in the list equivalent to one line from the configure.pol file.

Note that all data being passed in this list of dictionary items are strings – this will require that you change the data type to other basic types or objects depending on the particular field involved.

Remember that a "-" is valid for any field except Allow/Block and means that you do not want to match that content.

policy[] = Dictionary containing data from a line in the configure.pol file

policies = List of policy[] items

Georgia Tech
CREATING THE NEXT

policy[] dictionary contents

- policy[‘mac-src’] = Source MAC Address (00:00:00:00:00:00)
- policy[‘mac-dst’] = Destination MAC Address (00:00:00:00:00:00)
- policy[‘ip-src’] = Source IP Address (10.0.10.1/32)
- policy[‘ip-src-address'] = Source IP Address part (10.0.10.1)
- policy[‘ip-src-subnet'] = Source IP Address subnet (32)
- policy[‘ip-dst’] = Destination IP Address (10.0.10.1/32)
- policy[‘ip-dst-address'] = Destination IP Address part (10.0.10.1)
- policy[‘ip-dst-subnet'] = Destination IP Address subnet (32)
- policy[‘ipprotocol’] = IP Protocol (6)
- policy[‘port-src’] = Source Port for TCP/UDP (80)
- policy[‘port-dst’] = Destination Port for TCP/UDP (80)
- policy[‘rulenum’] = Rule Number (any)
- policy[‘comment’] = Comment (any)
- policy[‘action’] = Allow or Block

Georgia Tech
CREATING THE NEXT

What code is provided by the frameworks?

The base code frameworks provides the following:

- Import data from configure.pol, process each line into a dictionary and adds that dictionary to a list (policies) that is passed to your function.
- The code iterate through every entry in the policies list
- The variable rules is a list containing a POX flow modification object for every entry in the policies list
- rule should be a flow modification object, which would have match and action objects attached to it.
- The implementation code will automatically inject these rules into the POX OpenFlow controller.

Georgia Tech
CREATING THE NEXT

What do I need to code?

Your code needs to create an OpenFlow Flow modification object with a match and action object(if necessary) for every policy in the policies list.

Your code needs to configure these three objects to match and handle traffic based on the different configured policy[] items.

Reference the Appendix in the Project Description – it contains all that you need from the POX library to implement this project. It also contains notes that may help you decipher the POX API.

You do not need to validate input You should assume that all configuration items will be valid.

Georgia Tech
CREATING THE NEXT

What do I need to code?

Your code will go into the sdn-firewall.py file. You will be completing the firewall_policy_processing(policies) function.

You will start by commenting out the rule = None item and replacing that with an OpenFlow Flow Modification object.

You can add debug or print statements to the code. These will not have any impact on the autograder.

Please read Appendix A which is an excerpt from the POX API reference manual that contains all of information that you will need to complete this project.

Georgia Tech
CREATING THE NEXT

What process should I use?

It is recommended that you attempt to code the assignment using the OpenFlow POX API documentation and then test using the two example rules included with the project:

1,Block,-,-,10.0.0.1/32,10.0.1.0/24,6,-,80,Block 10.0.0.1 from accessing a web server on the 10.0.1.0/32 network

2,Allow,-,-,10.0.0.1/32,10.0.1.125/32,6,-,80,Allow 10.0.0.1 to access a web server on 10.0.1.125 overriding the previous rule

What do these rules do?

- The first rule basically blocks host hq1 (IP Address 10.0.0.1) from accessing a web server on any host on the us network (the subnet 10.0.1.0/24 network). The web server is running on the TCP IP Protocol (6) and uses TCP Port 80.
- The second rule overrides the initial rule to allow hq1 (IP Address 10.0.0.1) to access a web server running on us5 (IP Address 10.0.1.125)

Georgia Tech
CREATING THE NEXT

Implementation Tips and Tricks

- Remember that you use an Allow to override a previously defined Block action.
- You will need to assume that all traffic is IPV4.
- Do not hard code your solution. By hardcoding, it is meant that you should not be using actual IP/MAC Addresses and Ports in your coding. Use the policy[] dictionary items above to make the implementation generic. Failure to heed this instruction may result in the loss of 1/3 of your points when the project is graded with an alternate topology and/or ruleset.
- Hint 1: Priority is your friend. Do NOT use the rulenum as your priority. See Appendix A for a discussion on priority level. This is how you will make an Allow rule override a Block rule.
- Hint 2: Blocking a packet is a null action.
- If you use Visual Studio code, add the following to your workspace settings:

```
"python.autoComplete.extraPaths": [    "/home/mininet/pox/"   ],
```


Georgia Tech
CREATING THE NEXT

Error Conditions and How to Fix Them Part 1:

- On the topology terminal window, if you get an error message that states “Unable to contact Remote Controller”, it means that the POX controller had crashed and normally shows that there is a bug in your implementation code.
- Pay attention to the Fields ignored due to unspecified prerequisites warning. If you see this rule, you have issues with missing prerequisites in your implementation code and that the specified field will be ignored. See Appendix A Match Structure for a discussion of the prerequisites
- Watch for type mismatches.
- If you get the following error message, please run the cleanup.sh utility: "Exception: Error creating interface pair (s1-eth0,hq1-eth0): RTNETLINK answers: File exists”

Georgia Tech
CREATING THE NEXT

Error Conditions and How to Fix Them Part 2:

- If you get a struct.pack or struct.unpack error message, take a look at https://github.com/att/pox/blob/7f76c9e3c9bc999fcc97961d408ab0b71cbc186d/pox/openflow/libopenflow_01.py for more information. Also, the struct.pack error might reference how to fix (i.e., not an integer, EthAddr(), etc).
- A struct.unpack error may be ignored if it occurs on your GRE rule.

Georgia Tech
CREATING THE NEXT

How to test your implementation?

As you develop your solution, it is recommended that you test using the example rules to test your solution. Initially, you will want to test by hand using the test-server.py and the test-client.py as seen in the Wireshark demonstration video and as described in the project description.

Next, complete Part 6 and build a configure.pol file that will provide a more comprehensive test than the example ruleset. An unofficial test suite will be provided that will provide basic tests to verify your performance.

However, it will be your responsibility to make sure your implementation code can handle ANY VALID ENTRY from a configuration policy file.

Georgia Tech
CREATING THE NEXT