SDN Traffic Control

Final Presentation

Demo

Abstract

- Enterprise and campus networks install firewalls at edge of their networks to filter traffic based on policy
- The rest of the network wastes precious resources in forwarding such packets till the firewall
- We propose a SDN based distributed firewall solution which works up to the transport layer using a black list approach.
- Using appropriate flow rules on SDN switches which are present in the access layer of the network, the unwanted traffic can be blocked early.

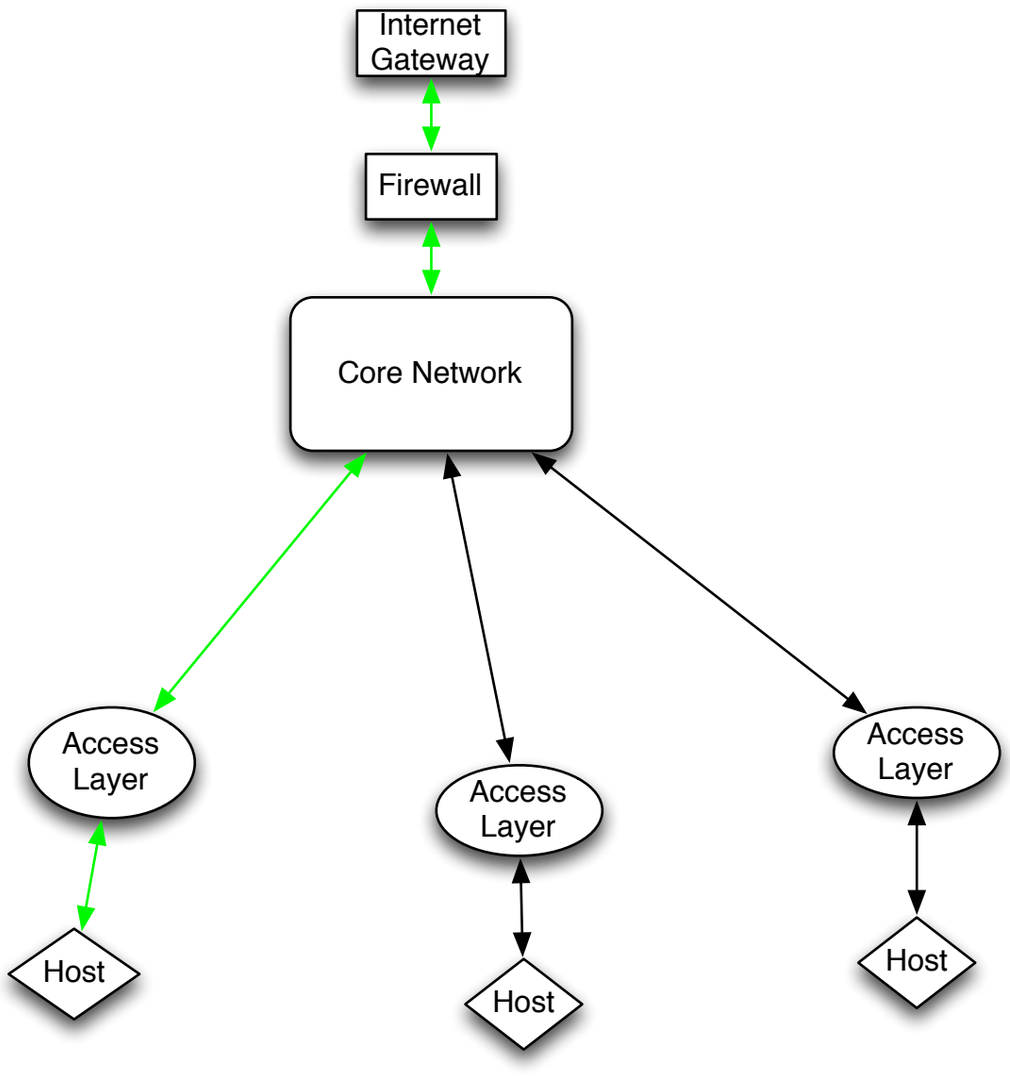
Introduction

- Enterprise class firewalls like the Cisco ASA hardware are expensive to purchase and maintain in the network.

- Due to these reasons enterprise networks do not have many such firewalls distributed throughout their network.

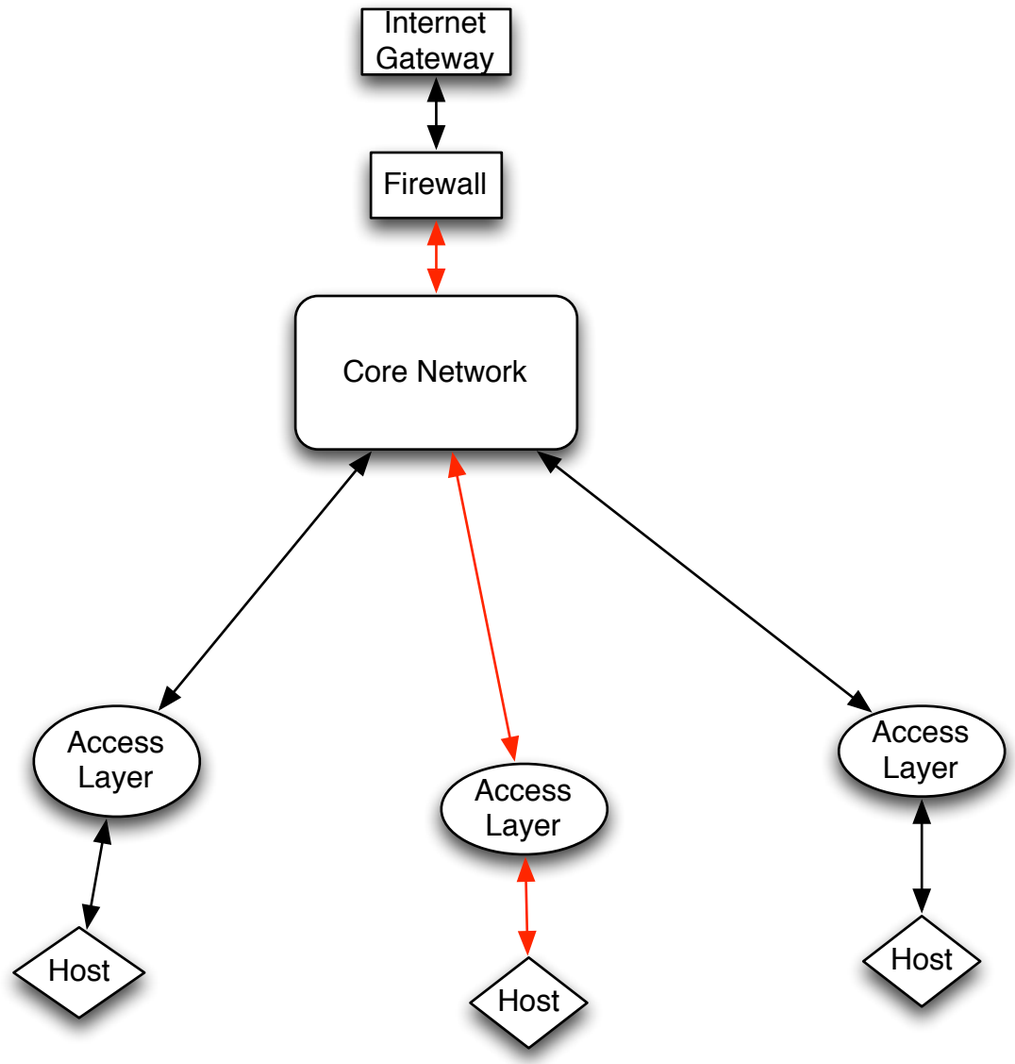
- Hence filtering of packets which might eventually be dropped at the firewall is not possible in the access layer of the network.

Idea



- Useful traffic is forwarded to gateway

Idea



- Useless traffic is dropped at firewall

Requirement Organisation

- The proposed solution would enable better utilisation of the network resources of the network.
- The network will consist of smaller amounts of unwanted traffic and useful traffic will be benefitted.

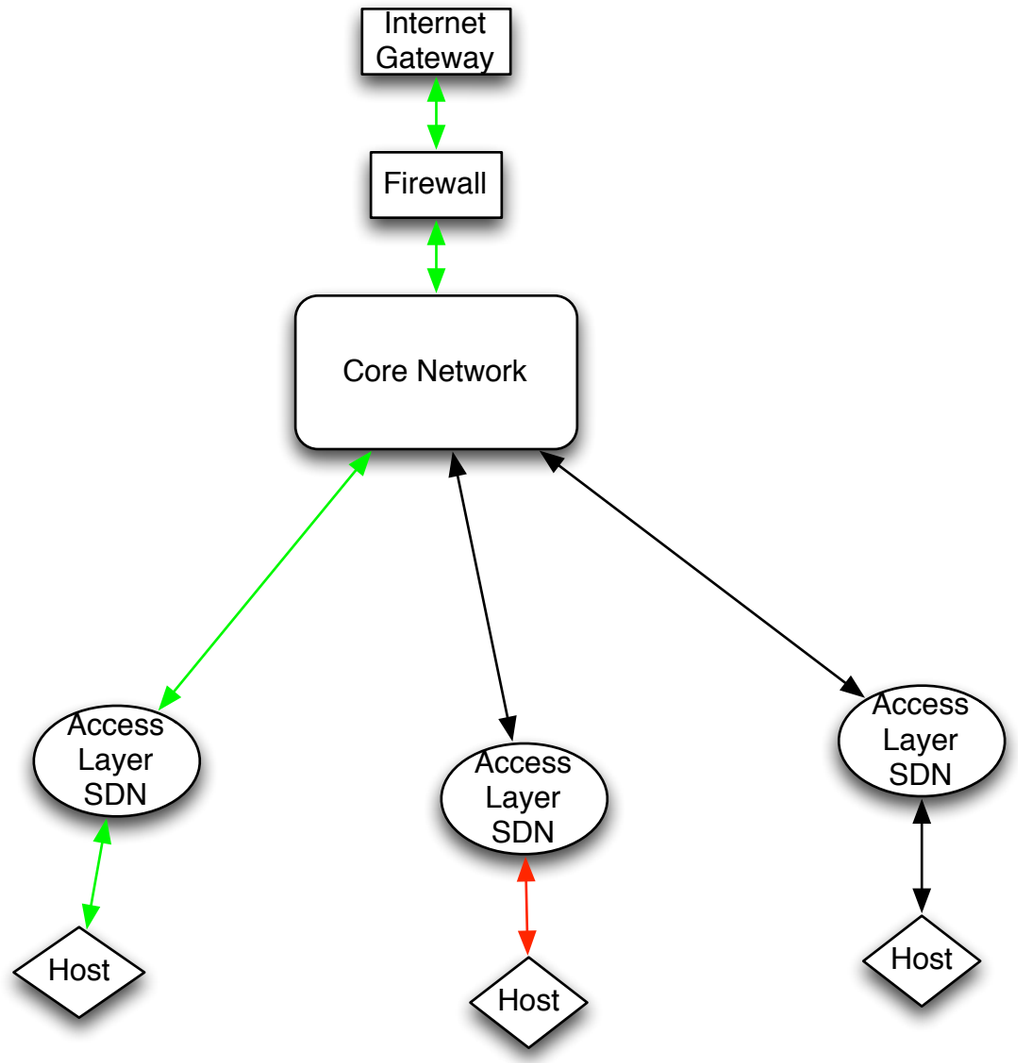
Solution

- SDN switches are much cheaper than Cisco(proprietary) firewalls.

- They have the flexibility to read packets up to the transport layer. Also all SDN switches can be controlled by a central controller.

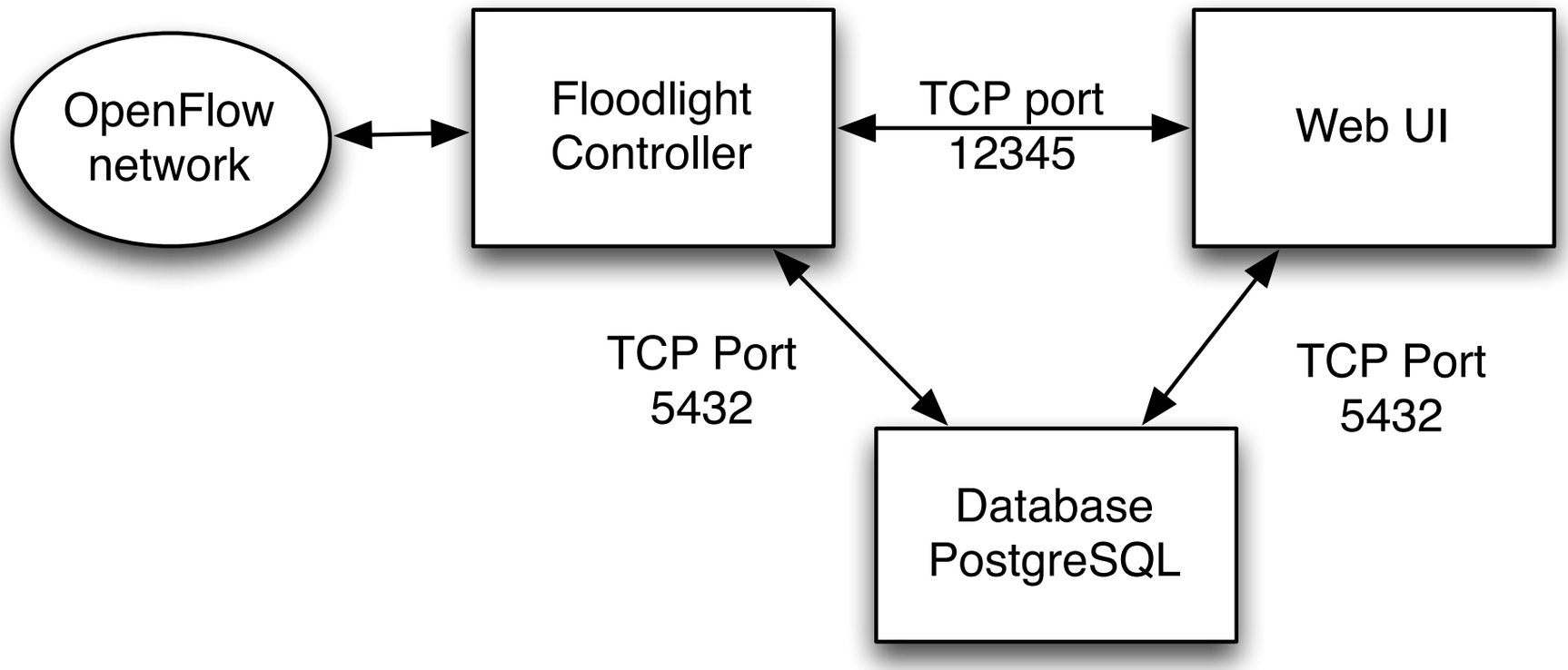
- SDN switches can hence be deployed at the access layer of the network, and appropriate flow rules added to filter out unwanted traffic.

Solution



- SDN switch can drop packet instead of firewall

Architecture



Implementation

- The solution is implemented as an additional module in the Floodlight SDN controller.

- The module has 2 parts, one part deals with handling OpenFlow messages like PACKET_IN, and the other part deals with handling messages from the Web UI.

- The module listens on TCP port 6633 for OpenFlow switches, and TCP port 12345 for update messages regarding change in firewall rules.

Implementation

- We use a database to store all the rules of the firewall. The schema is as follows:
  - (src_net,src_prefix,dst_net,dst_prefix,proto,dst_port,priority)
  - The Web UI sends the above parameters to the OpenFlow module in JSON format to add the rule to all switches immediately.

Suricata Rule Set

- To increase the usefulness of this project and also to easily manage the rules, we have added support to import rules from Suricata.

- We implement a parser for the rules, which will convert rules possible in OpenFlow to the format required by our module.

- We used the Security Onion db to obtain the most popular rules to give them higher priority in the flow rules.

- Every Suricata rule is stored independent of the rules added by the administrator. The rules also have an option to enable or disable.

System Implementation

- The source code was mainly written in Java using the Floodlight OpenFlow Controller.
- The Web UI was written in JSP.
- PostgreSQL was used as the backend database

Evaluation

- The proof of concept was first tested on Mininet simulator and also using VMs and OpenVSwitch.

- The flow rules were correctly installed for the firewall rules inserted using the WebUI, and the traffic matching the rule was blocked successfully.

- The system was then deployed in the Disanet SDN network, and one rule to block ‘iperf’ was added.

- The deployment was successful.

Evaluation

- The system is based on a blacklist approach.
- The whitelist approach has more performance drawbacks due to the nature of the OpenFlow Protocol.
- The blacklist approach requires one flow rule for every firewall rule that is to be applied.
- In the whitelist approach we need number of flow rules = (no. firewall rules)*(no. users matching the rule)
- This is due to the fact that in an OpenFlow switch, a matching rule should specify the action, and cannot refer to another rule.

Conclusion

- The distributed firewall proposed here is able to achieve required goal of reducing the unwanted traffic in the access layer.

- Due to limitation of OpenFlow protocol, not all proprietary firewall rules are possible to be implemented.(For example blocking on range of ports)

Future Work

- The Suricata rules have to better translated into flow rules.

- Automatic updating of rules from Suricata and priority values of rules based on SecurityOnion.

Drawbacks of using Suricata Rules

- The number of rules are very large in number and difficult to test

- Since we are ignoring DPI checks in rules, some rules become redundant.

- Currently no ‘block’ rule is being used in our Suricata deployment. For testing purposes we used ‘alert’ rules.

- Some rules which contain port range matches and other parameters which cant be expressed in OpenFlow rules are ignored.

Suggestions?

Thank You

ありがとう