Big Data Analytics for Real-time DDoS Detection and Mitigation in SDN

Cloud 9:
Sile Hu
Rahul Prabhu
Umar Majeed
Sanil Sinai Borkar

Agenda

-Distributed Denial of Service

-SDN and OpenFlow

-System Architecture

-System Design and Implementation

-Experiments

-Results

Distributed Denial of Service

- A Cyber Attack carried out by multiple Computers
- DDoS that Knocked Spamhaus offline
- There is an exponential Increase in the number of DDoS Attacks of over 20Gb

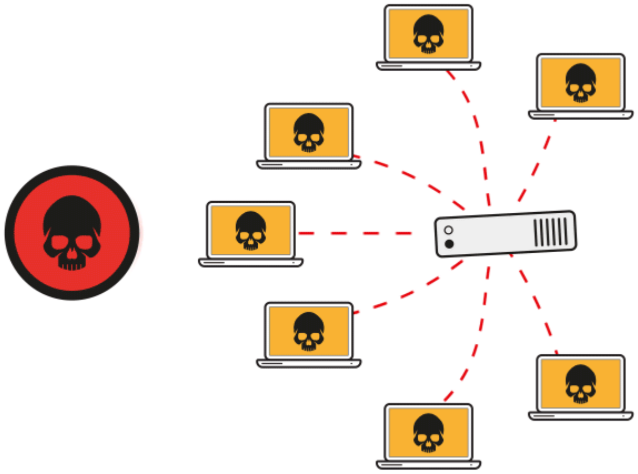


Image Copyrights: https://www.tsohost.com/blog/internet-warfare-the-ddos-arms-race

Software Defined Networking

- Provide centralized view of the overall network
- Separation of Control and Forwarding (Data) plane.
- OpenFlow: Standard for communication protocol that enables control plan to interact with forwarding plane.

Why SDN for DDoS defense

SDN Features help solving the DDoS problem:

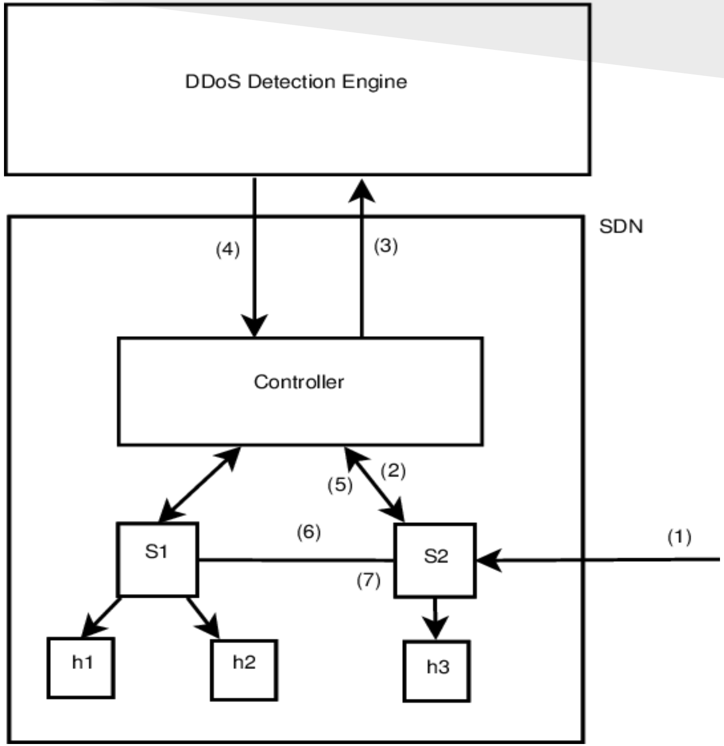
- The SDN Switches can act as a firewall when deployed at the edge of a network.
- With the help of the control plain, dynamically change the route of the packet and balance the load when an attack occurs.

Limitations & Assumptions

- Can not simulate an actual DDoS attack due to university’s network security policies.
- Controller is assumed to be secure.
- Network Latency
- We consider only TCP SYN flooding attack.

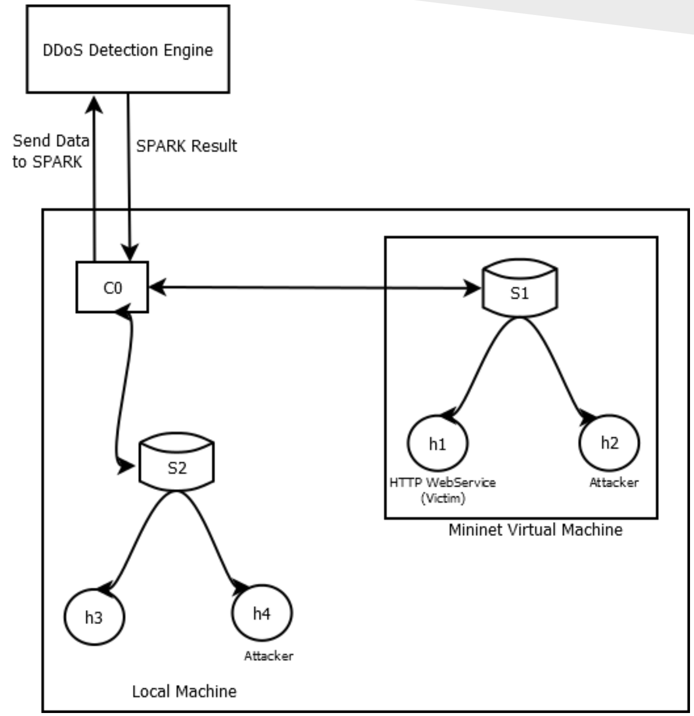
System Architecture

1. Packet reaches the boundary switch.
2. Switch asks controller for a decision on what to do with the packet.
3. Controller sends packet information to DDE
4. DDE classifies the packet as malicious or non malicious and sends information to controller
5. Controller adds flow rules to the switch based on the result from DDE
6. If the packet is not malicious, it's sent to its destination
7. Malicious packets are dropped by blocking the host.



System Design and Implementation

- Mininet as the host platform
- Components:
  - A local Machine hosting the controller (c0)
  - A guest mininet vm with two hosts and a switch.
  - Attackers (h2) and (h4) and a victim (h1).
  - Openflow enabled s1 switch, controlled by a remote controller (c0)
  - DDoS Detection Engine running on a SPARK Cluster



Design and Implementation - TCP SYN Flooding

- Host 1 is set up to host a python web HTTP server
- Host 2, the attacker carries out a TCP SYN Flooding attack on the Host 1.
- TCP SYN Flooding Attack:
  - Attacker sends a succession of SYN Requests to server
  - Server accepts the requests. The connection is established.
  - Attacker consumes server resources
  - Server becomes unresponsive to legitimate traffic

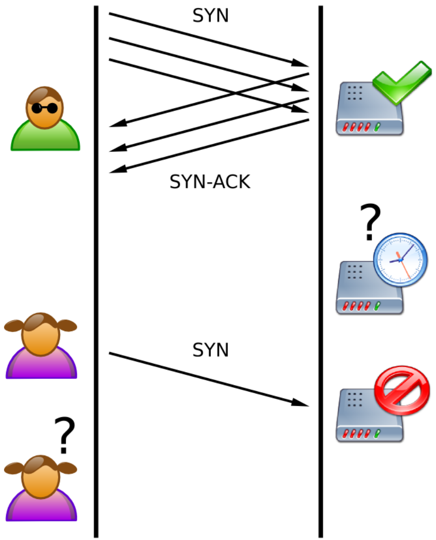


Image from www.wikipedia.com

Design and Implementation - DDE

- Main computation engine of the system.
- Runs on a SPARK Cluster.
  - Extends the functionality of the Hadoop
    - In memory Cluster Computing
    - Streaming Interface
    - Machine Learning Library (MLlib)
  - Spark is not tied to the two stage map reduce paradigm and performs hundred times faster than hadoop

DDE - Training

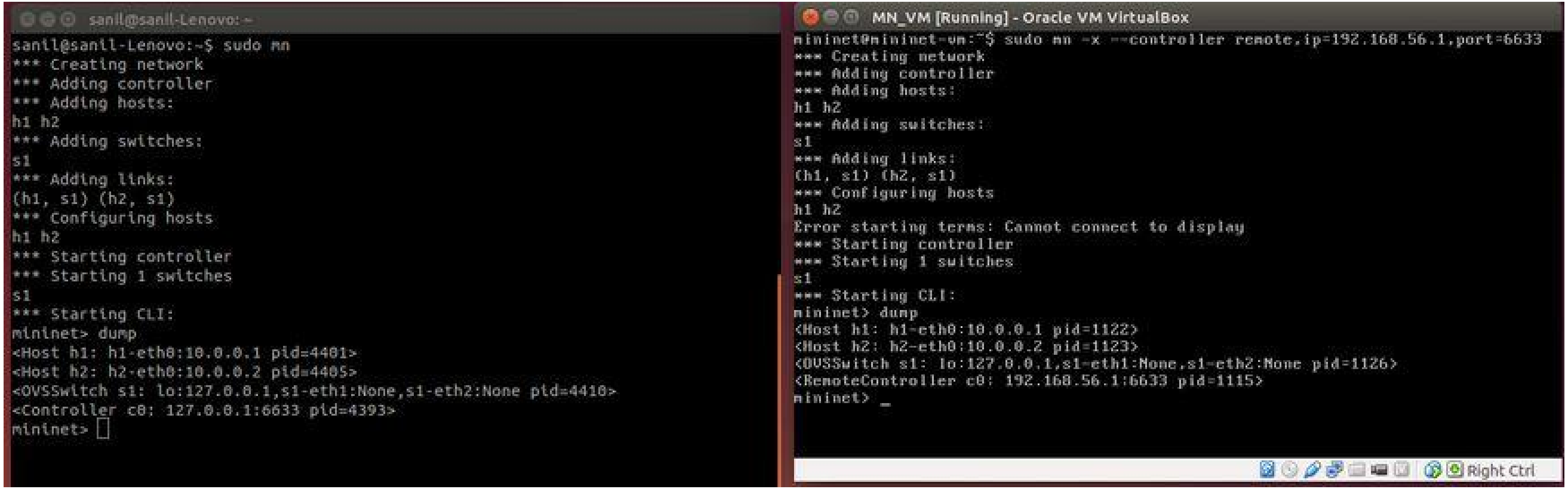
- MLLib: in built machine learning library framework provided by Apache SPARK
- DDE Training data used a good mix of normal and attack data provided by CAIDA.
- Training makes use of the “Decision Tree” machine learning algorithm.

Experiments & Results - Overview

- Simulation of a SDN Network setup ⟸
- Classification of the packets using SPARK EC2 Cluster.
- Using *OVS-OFCTL* commands to push flow rules to the switch.

Experiments & Results - Simulation of a SDN Network

- Openflow enabled Switch s1 is connected to a remote controller c0



Experiments & Results - Overview

- Classification of the packets using SPARK EC2 Cluster.

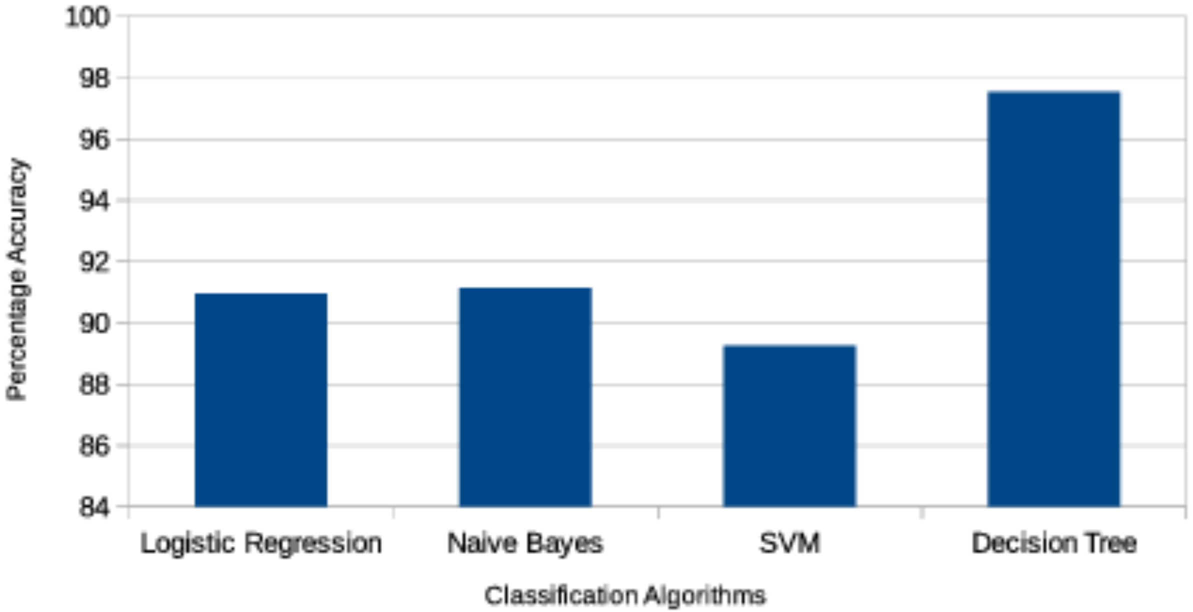
Experiments & Results - Classification

- Experimented with several classification algorithms
- The time for prediction is cumulative for all dataset rows.
- Decision Tree performs better than others even with a small feature set

Algorithm	Feature Set	Dataset Records	Time for Training (s)	Time for Prediction (s)	Error
Decision Tree	14	36,646	0.0002520084	0.0040941238	2.46%
SVM	43	36,646	0.0002040863	0.0044519901	10.75%
Naive Bayes	43	36,646	0.0005002022	0.0016739368	8.84%
Logistic Regression	43	36,646	0.0003650188	0.0011081696	9.04%

Performance of Various MLlib Machine Learning Algorithms

Experiments & Results - Classification



Percentage accuracy of the Decision Tree compared to other classification algorithms.

Experiments & Results - Overview

- Using *OVS-OFCTL* commands to push flow rules to the switch.

Experiments & Results - Mitigation

- Used OVS-OFCTL commands to add flow rules to the switch.
  - To enable forwarding in the switch:

ovs-ofctl add-flow s1 priority=10,action=normal

10.0.0.1 is the ip address of the host.

  - To block traffic from a malicious host

ovs-ofctl add-flow s1 priority=11,dl_type=0x0800,nw_src=10.0.0.1,action=drop

  - To restore the traffic back.

ovs-ofctl --strict del-flows s1 priority=11,dl_type=0x0800,nw_src=10.0.0.1

Experiments & Results - Mitigation

mininet@mininet-vm: ~/control

```
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '(\w+-eth\w+)'
*** Cleanup complete.
mininet@mininet-vm:~/control$ vim SimpleTopology.py
mininet@mininet-vm:~/control$ sudo python SimpleTopology.py
Flow Rule Added
*** Ping: testing ping reachability
h0 -> h1 h2
h1 -> h0 h2
h2 -> h0 h1
*** Results: 0% dropped (6/6 received)
Stop the flow from host 0  with ip 10.0.0.1
*** Ping: testing ping reachability
h0 -> X X
h1 -> X h2
h2 -> X h1
*** Results: 66% dropped (2/6 received)
Restore communication with the host 0
*** Ping: testing ping reachability
h0 -> h1 h2
h1 -> h0 h2
h2 -> h0 h1
*** Results: 0% dropped (6/6 received)
mininet>
```

OVS - OFCTL commands at test. Blocking data from the malicious host (h0) and restoring it after host quarantine.

Conclusion

- We develop a prototype to detect and counter DDoS in SDN at real-time using SPARK with MLlib.
- Empirical analyses show that Decision Tree is the best algorithm for packet classification.
  - Accuracy of over 97% using CAIDA dataset.

Related Work

- Various techniques exist to defend attacks from independent routers point of view
  - lack of real time response
  - need to be applied per router bases
- Brocade and Radware provide detection and control over the SDN
  - Detection and access rules are largely packet filtering based.

Future Work

- System could further be enhanced to detect and counter other types of DDoS attacks.
- Robustness and security can be further improved by making the controller resistant to DDoS attacks.
- Latency can be improved by reducing the network bottleneck between the SDN controller and the DDE.