

DDoS Attack Recognition and Mitigation

SakethGajawada

IMT2020531

Saketh.Gajawada@iiitb.ac.in

1. Introduction

Distributed denial of service (DDoS) attacks pose a grave threat to the stability and security of online networks. By hijacking vulnerable devices and forming powerful "botnets", malicious actors can unleash a crippling barrage of traffic to cripple even the most robust of systems. As our digital lives continue their rapid migration online, ensuring uninterrupted access and dependable service has never been more important.

2. Motivation

In today's hyperconnected world, even moments of downtime can have outsized consequences. When bad actors bring servers to their knees with floods of malicious traffic, more is disrupted than just bits and bytes - businesses suffer, communication halts, and vital services risk outage. Traditional defenses have struggled to keep pace with the evolving tactics of sophisticated DDoS attackers. There is an urgent need for innovative solutions capable of detecting and mitigating these advanced threats at the speed and scale demanded in modern networks. By leveraging the programmability of Software Defined Networking, this project aims to develop a proactive, prevention-oriented approach for thwarting DDoS attacks before damage can be done. With SDN's centralized control and real-time traffic analysis, anomalous patterns indicative of an incoming attack can be identified with unprecedented acuity. Countermeasures can then be deployed with lightning speed through dynamic, automated rule adjustments.

3. Tools Used

The key tools used in this project are:

1. Mininet - For emulating a virtual SDN network topology
2. sFlow - For collecting flow-level network traffic statistics
3. Ryu Controller - For implementing the SDN control logic
4. hping3 - For generating attack traffic in the Mininet topology

4. Working

1. Ensure the Installation is properly done without any errors.
2. Run the following command to start sFlow-RT and run the ryu.js script
3. Start the Mininet Topology. Either run the Topology.py or directly make a simple mininet topology using the below command
4. Run ryu application
5. Enable the Sflow in the switch s1
6. Open the sflow-rt Dashboard. Open the mininet dashboard from apps in the menu.
7. Run hping3 to attack the network
8. Check dump flows

```
sakethg@sakethg-VirtualBox: ~  
sakethg@sakethg-VirtualBox:~$ env "RTPROP=-Dscript.file=$PWD/ryu.js" sflow-rt/start.sh  
2023-12-09T21:17:34+05:30 INFO: Starting sFlow-RT 3.0-1695  
2023-12-09T21:17:34+05:30 INFO: Version check, running latest  
2023-12-09T21:17:34+05:30 INFO: Listening, sFlow port 6343  
2023-12-09T21:17:35+05:30 INFO: Listening, HTTP port 8008  
2023-12-09T21:17:35+05:30 INFO: /home/sakethg/ryu.js started  
2023-12-09T21:17:35+05:30 INFO: app/mininet-dashboard/scripts/metrics.js started
```

Fig. 1. SFLOW-RT Intialization

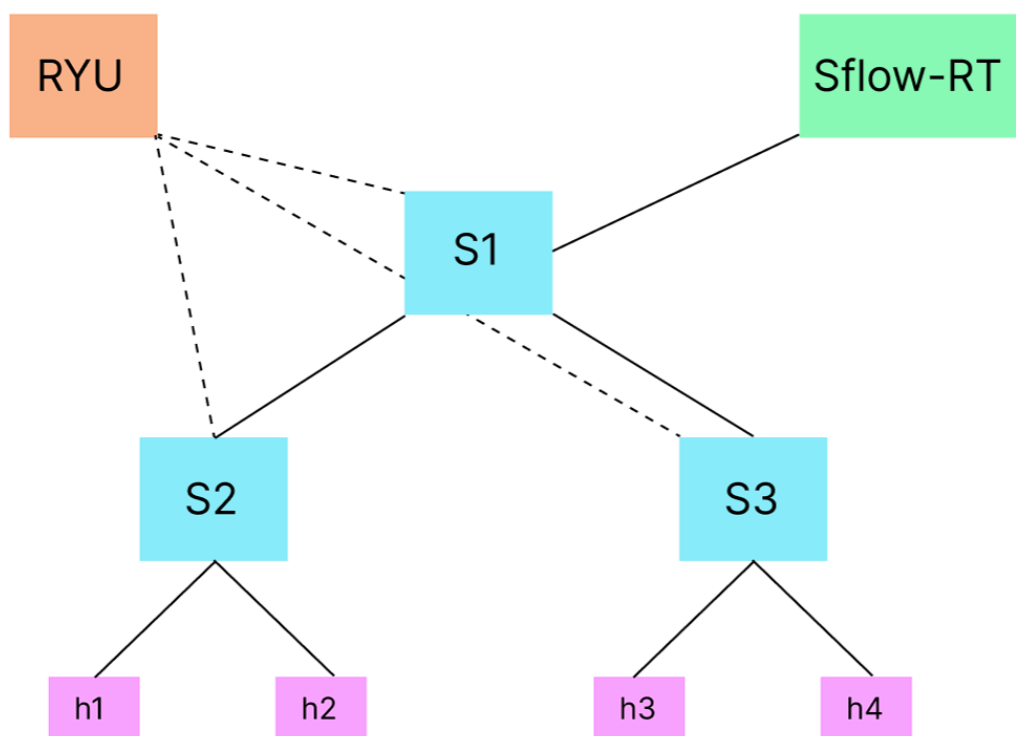


Fig. 2. Simple Topology

```
sakethg@sakethg-VirtualBox: ~  
sakethg@sakethg-VirtualBox:~$ sudo mn --custom sflow-rt/extras/sflow.py --link tc,bw=10 --  
controller=remote,ip=127.0.0.1 --topo tree,depth=2,fanout=2  
*** Creating network  
*** Adding controller  
Unable to contact the remote controller at 127.0.0.1:6653  
Unable to contact the remote controller at 127.0.0.1:6633  
Setting remote controller to 127.0.0.1:6653  
*** Adding hosts:  
h1 h2 h3 h4  
*** Adding switches:  
s1 s2 s3  
*** Adding links:  
(10.00Mbit) (10.00Mbit) (s1, s2) (10.00Mbit) (10.00Mbit) (s1, s3) (10.00Mbit) (10.00Mbit)  
(s2, h1) (10.00Mbit) (10.00Mbit) (s2, h2) (10.00Mbit) (10.00Mbit) (s3, h3) (10.00Mbit) (10  
.00Mbit) (s3, h4)  
*** Configuring hosts  
h1 h2 h3 h4  
*** Starting controller  
c0  
*** Starting 3 switches  
s1 s2 s3 ...(10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00  
Mbit) (10.00Mbit)  
*** Enabling sFlow:  
s1 s2 s3  
*** Sending topology  
*** Starting CLI:  
mininet> 
```

Fig. 3. Mininet

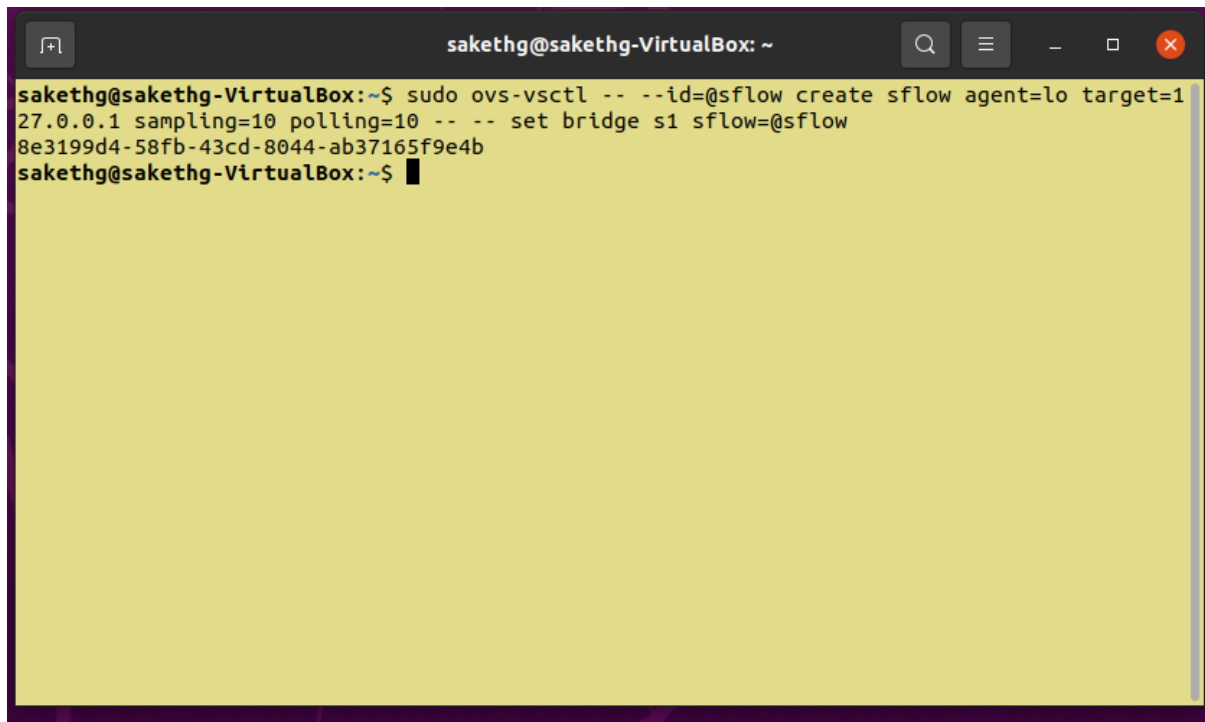
```
sakethg@sakethg-VirtualBox: ~  
sakethg@sakethg-VirtualBox:~$ ryu-manager ryu.app.simple_switch_13 ryu.app.ofctl_rest  
loading app ryu.app.simple_switch_13  
loading app ryu.app.ofctl_rest  
loading app ryu.controller.ofp_handler  
instantiating app None of DPSet  
creating context dpset  
creating context wsgi  
instantiating app ryu.app.simple_switch_13 of SimpleSwitch13  
instantiating app ryu.app.ofctl_rest of RestStatsApi  
instantiating app ryu.controller.ofp_handler of OFPHandler  
(6909) wsgi starting up on http://0.0.0.0:8080  
packet in 0000000000000002 32:4e:65:4d:7a:bf 33:33:00:00:00:02 2  
packet in 0000000000000001 5a:f3:03:04:cb:63 33:33:00:00:00:02 1  
packet in 0000000000000003 aa:89:87:82:78:41 33:33:00:00:00:02 1  
packet in 0000000000000003 5a:f3:03:04:cb:63 33:33:00:00:00:02 3  
packet in 0000000000000001 aa:89:87:82:78:41 33:33:00:00:00:02 2  
packet in 0000000000000001 32:4e:65:4d:7a:bf 33:33:00:00:00:02 1  
packet in 0000000000000002 aa:89:87:82:78:41 33:33:00:00:00:02 3  
packet in 0000000000000003 32:4e:65:4d:7a:bf 33:33:00:00:00:02 3  
█
```

Fig. 4. Ryu Manager

5. Results

1. The image 1 shows the initialization of sflow-rt.

2. Construct the topology using mininet either using direct command or using the topology.py. The diagrams [2 3](#) is an example of a simple topology with 4 hosts and 3 switches.
3. Start Ryu application [4](#) simple switch and ofctl rest, Simple switch is for switching and ofctl rest to communicate to ryu application for adding the blockage of flow when an attack is detected/recognized.
4. Configure s1 flow (include SFlow)[5](#)



```
sakethg@sakethg-VirtualBox: ~  
sakethg@sakethg-VirtualBox:~$ sudo ovs-vsctl -- --id=@sflow create sflow agent=lo target=1  
27.0.0.1 sampling=10 polling=10 -- -- set bridge s1 sflow=@sflow  
8e3199d4-58fb-43cd-8044-ab37165f9e4b  
sakethg@sakethg-VirtualBox:~$
```

Fig. 5. Flow-Rule

5. Pingall to ensure our topology is connected to ryu controller and working.
6. Open localhost:8008 and view mininet dashborad.
7. Start attacking/flooding the network. Either run the the hping3 command mentioned earlier or any one of the generate_ddos_traffic.py/generate_ddos_traffic_1.py.
8. Below Digaram of mininet dasbhboard shows the sudden spike in the transfer rate of incoming packets.
9. The diagram [6](#) shows that whenever there is flooding happening at a node(host), It is blocked with the help of sflow.
10. We can also see the POST command [7](#) from RYU controller terminal given below communicating with SFlow
11. Check by dumping flows.[8](#) As you can see when we flood, the output action is automatically configured to drop by Sflow.

```
sakethg@sakethg-VirtualBox: ~
sakethg@sakethg-VirtualBox:~$ sflow-rt/start.sh
2023-12-09T22:42:30+05:30 INFO: Starting sFlow-RT 3.0-1695
2023-12-09T22:42:31+05:30 INFO: Version check, running latest
2023-12-09T22:42:31+05:30 INFO: Listening, sFlow port 6343
2023-12-09T22:42:31+05:30 INFO: Listening, HTTP port 8008
2023-12-09T22:42:31+05:30 INFO: app/mininet-dashboard/scripts/metrics.js started
2023-12-09T22:42:42+05:30 INFO: blocking 10.0.0.3,53
```

Fig. 6. Blocking

```
sakethg@sakethg-VirtualBox: ~
packet in 0000000000000003 46:a8:04:da:bb:fe 33:33:00:00:00:02 2
packet in 0000000000000001 46:a8:04:da:bb:fe 33:33:00:00:00:02 2
packet in 0000000000000002 46:a8:04:da:bb:fe 33:33:00:00:00:02 3
packet in 0000000000000002 2e:fc:d8:47:75:e8 33:33:00:00:00:02 1
packet in 0000000000000001 2e:fc:d8:47:75:e8 33:33:00:00:00:02 1
packet in 0000000000000003 2e:fc:d8:47:75:e8 33:33:00:00:00:02 3
packet in 0000000000000001 92:24:1b:27:7b:c5 33:33:00:00:00:02 2
packet in 0000000000000002 92:24:1b:27:7b:c5 33:33:00:00:00:02 3
(16859) accepted ('127.0.0.1', 33784)
127.0.0.1 - - [09/Dec/2023 17:26:17] "POST /stats/flowentry/add HTTP/1.1" 200 134 0.040701
packet in 0000000000000001 5a:f3:03:04:cb:63 33:33:00:00:00:02 1
packet in 0000000000000002 32:4e:65:4d:7a:bf 33:33:00:00:00:02 2
packet in 0000000000000003 5a:f3:03:04:cb:63 33:33:00:00:00:02 3
packet in 0000000000000001 32:4e:65:4d:7a:bf 33:33:00:00:00:02 1
packet in 0000000000000003 32:4e:65:4d:7a:bf 33:33:00:00:00:02 3
packet in 0000000000000003 aa:89:87:82:78:41 33:33:00:00:00:02 1
packet in 0000000000000001 aa:89:87:82:78:41 33:33:00:00:00:02 2
packet in 0000000000000002 aa:89:87:82:78:41 33:33:00:00:00:02 3
packet in 0000000000000002 5a:ce:63:61:c8:2c 33:33:00:00:00:02 3
packet in 0000000000000003 d6:fc:33:c7:97:eb 33:33:00:00:00:fb 3
packet in 0000000000000001 5a:f3:03:04:cb:63 33:33:00:00:00:fb 1
packet in 0000000000000001 92:24:1b:27:7b:c5 33:33:00:00:00:fb 2
packet in 0000000000000003 5a:f3:03:04:cb:63 33:33:00:00:00:fb 3
packet in 0000000000000002 92:24:1b:27:7b:c5 33:33:00:00:00:fb 3
```

Fig. 7. Flowentry recorded to block

```
0FPST_FLOW reply (0F1.3) (xid=0x2):
  cookie=0x0, duration=40.881s, table=0, n_packets=1115082, n_bytes=46833444, priority=4000,udp,in_port=1,nw_
dst=10.0.0.3,tp_src=53 actions=drop
  cookie=0x0, duration=104.057s, table=0, n_packets=3, n_bytes=238, priority=1,in_port=2,dl_src=42:9f:26:cc:0
3:2e,dl_dst=b2:cf:1e:7c:66:f9 actions=output:1
  cookie=0x0, duration=104.055s, table=0, n_packets=2, n_bytes=140, priority=1,in_port=1,dl_src=b2:cf:1e:7c:6
6:f9,dl_dst=42:9f:26:cc:03:2e actions=output:2
  cookie=0x0, duration=104.047s, table=0, n_packets=924, n_bytes=61040, priority=1,in_port=3,dl_src=aa:c9:aa:
6c:42:b2.dl_dst=b2:cf:1e:7c:66:f9 actions=output:1
```

Fig. 8. DumpFlows

References

1. <https://github.com/chiragbiradar/DDoS-Attack-Detection-and-Mitigation>
2. <https://sflow.org/>
3. <https://ernie55ernie.github.io/sdn/2019/03/25/install-mininet-and-ryu-controller.html>
4. https://ryu.readthedocs.io/en/latest/ryu_app_api.html#create-a-ryu-application