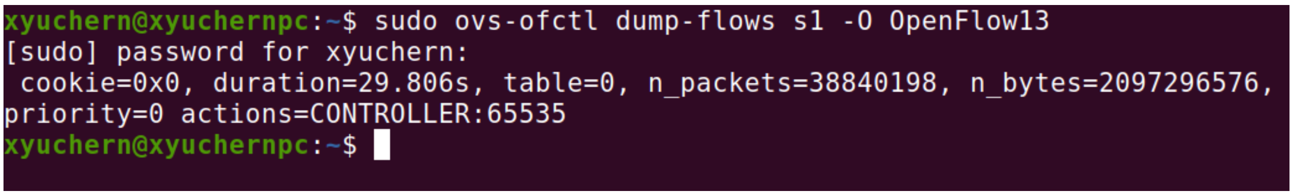
CAN201 In-Class Test 3

DoS attacks against the Data Plane of an SDN Network

Xu Chen   2257453

1. What is the initial flow entry (before attacking) installed in the switch of this lab?

Before the attack, the switch contains the following initial flow entry:



```
xyuchern@xyuchernpc:~$ sudo ovs-ofctl dump-flows s1 -O OpenFlow13
[sudo] password for xyuchern:
 cookie=0x0, duration=29.806s, table=0, n_packets=38840198, n_bytes=2097296576,
priority=0 actions=CONTROLLER:65535
xyuchern@xyuchernpc:~$ 
```

This flow entry is a default rule installed when the switch connects to the controller. It matches all packets not handled by other rules and sends them to the controller for further processing. This ensures that no packets are dropped due to the lack of matching flow entries, and it allows the controller to dynamically analyze traffic and install new rules.

2. Explain what this initial flow entry does ?

The initial flow entry is a default rule with the lowest priority (priority=0). It matches all packets not handled by other rules and forwards these packets to the controller using the action CONTROLLER:65535. This ensures that unmatched traffic is not dropped, allowing the controller to dynamically analyze packets and decide how to handle them.

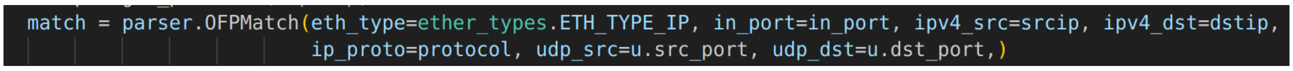
Each component of the flow entry contributes to the rule's functionality:

- cookie=0x0 is the identifier for the flow.
- duration=29.806s indicates the lifetime of the rule.
- n_packets=38840198 and n_bytes=2097296576 represent the volume of traffic matched by the rule.
- priority=0 ensures this rule is only applied as a last resort.
- actions=CONTROLLER:65535 sends unmatched traffic to the controller for further handling.

This mechanism allows flexible and dynamic network management while ensuring that no packets are dropped unnecessarily.

3. Show the flow 'match' rule (in the lab11.py) used to cope with the above flooding traffic.

The flow match rule in the lab11.py file used to cope with the above flooding traffic is as follows:



```
  match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP, in_port=in_port, ipv4_src=srcip, ipv4_dst=dstip,
                          ip_proto=protocol, udp_src=u.src_port, udp_dst=u.dst_port,)
```

This match rule is designed to handle very specific traffic by precisely matching multiple fields, including the packet's Ethernet type, input port, source and destination IP addresses, transport layer protocol, and UDP source and destination ports. Such granular matching is useful for highly controlled traffic management but can lead to issues (like flow table exhaustion) during attacks like UDP Flooding.

4. Explain how the flooding command exhausts the switch’s flow table?

Flooding attacks exploit the mechanism of spoofing source IPs combined with sending packets at an extremely high frequency. This forces the switch to continuously generate Packet-In messages to the controller and request the installation of new flow rules.

As the number of these rules grows rapidly, the limited capacity of the switch's flow table is quickly exhausted. Once the flow table is full, the switch can no longer handle new or legitimate traffic effectively, leading to severe performance degradation.

This situation not only consumes the switch’s resources but also severely disrupts normal traffic flow. It causes significant delays, packet drops, and eventually a complete denial of service, making the network unresponsive to valid user requests.