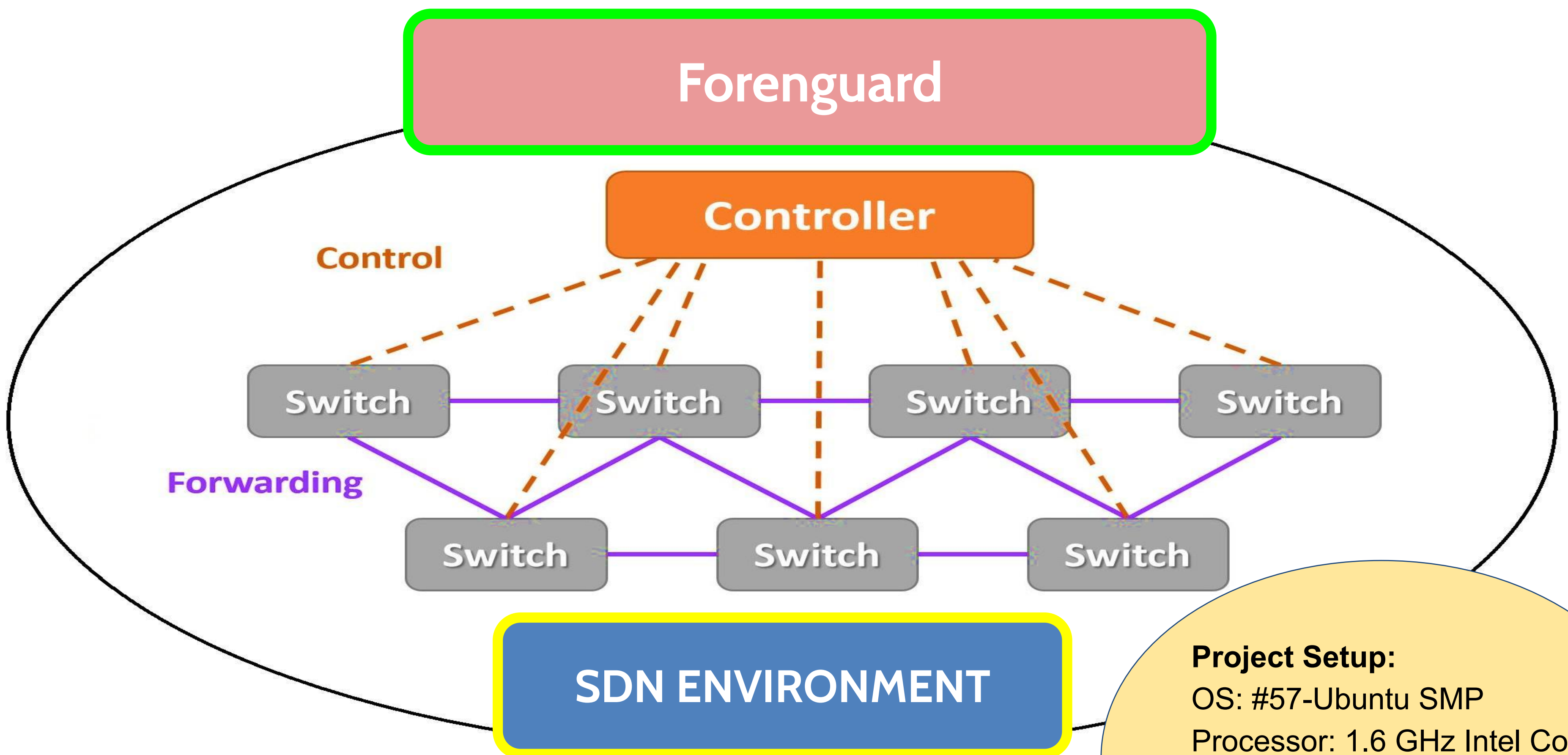
Software Defined Networking Security

Chong Yu, Sovatha Sang, Damien Piris , Marc Suda, John Jay College of Criminal Justice, Manhattan, NY 10019
Sven Dietrich, Associate Professor, Mathematics and Computer Science Department John Jay College of Criminal Justice, Manhattan, NY 10019



Project Setup:
OS: #57-Ubuntu SMP
Processor: 1.6 GHz Intel Core i5
Memory: 8 GB 1867 MHz DDR3
Tools used: ForenGuard
MongoDB
ODL Controller

Introduction

What is SDN? Software Defined Networking is an modern computer network architecture which defines how a networking and computing system can be built using a combination of open, software-based technologies and commodity networking hardware that separate the SDN control plane and the SDN data plane of the networking stack.

Before SDN/Traditional Networking? In traditional networking, the switch does not have programmability, the rules cannot be changed dynamically. In SDN, the switch is connected to a controller, which controls the actions of the switch. The controller can be programmed dynamically to control the switch.

Advantages of SDN

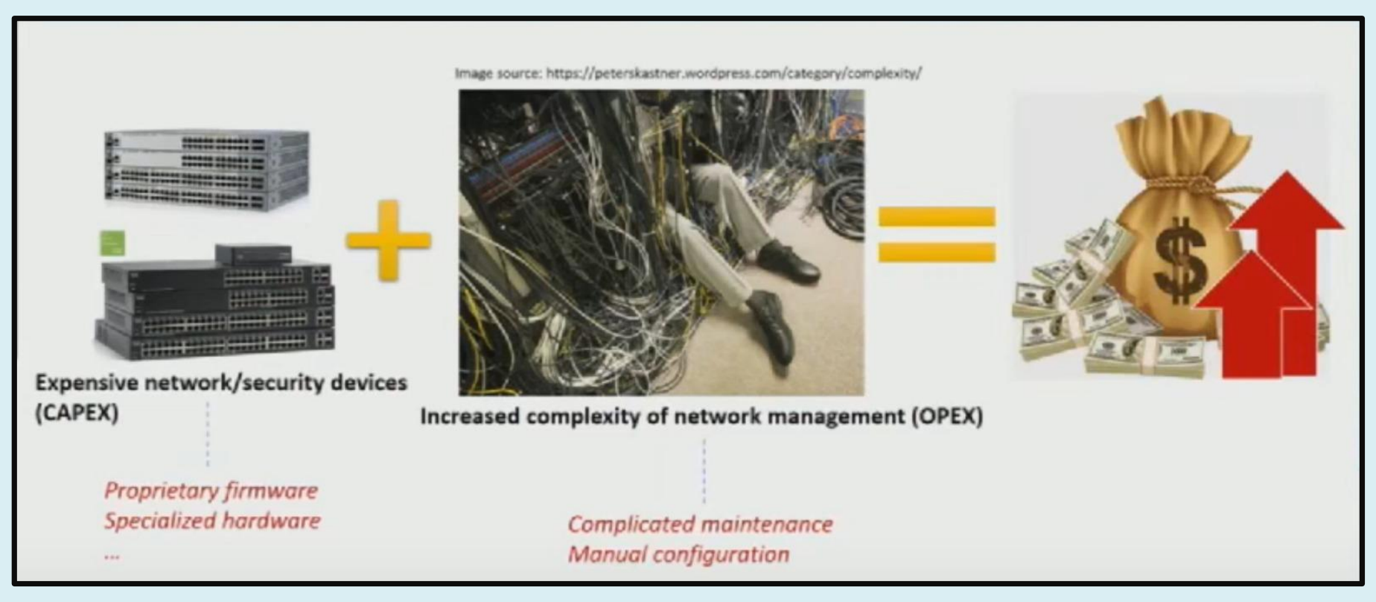
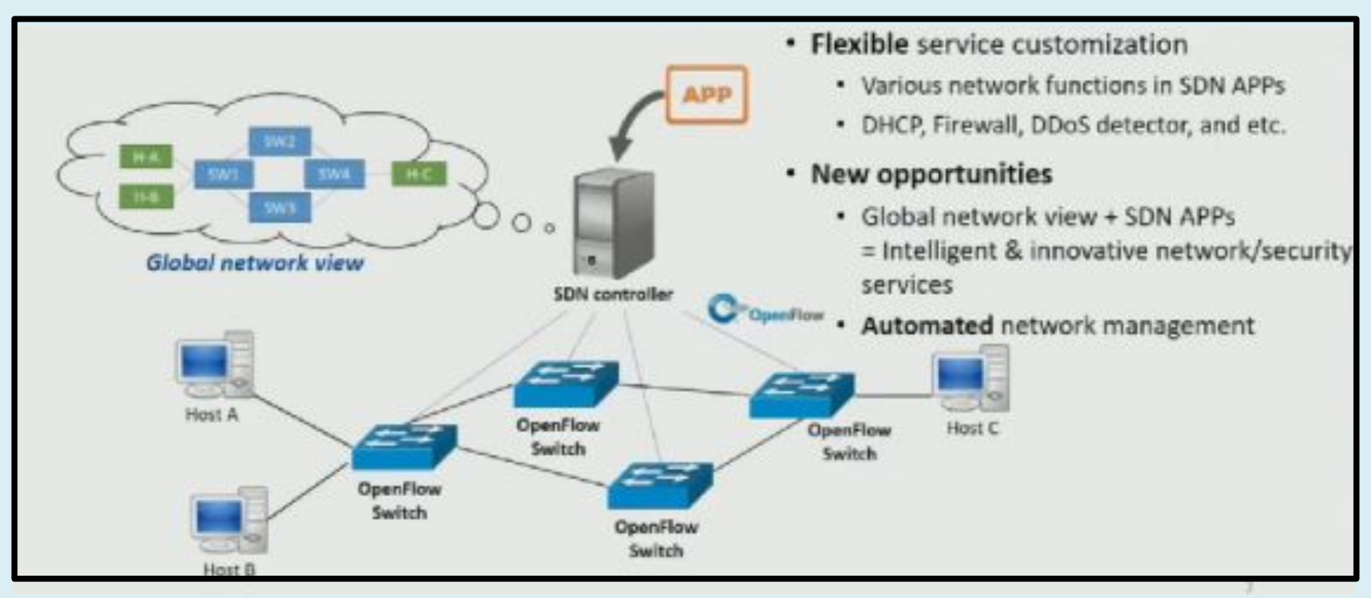


Image source: https://peterskastner.wordpress.com/category/complexity/
Expensive network/security devices (CAPEX)
Proprietary firmware
Specialized hardware
...
Increased complexity of network management (OPEX)
Complicated maintenance
Manual configuration

- ❑ In a month from now, there will be over 50 billion devices connected to the internet because of this our network will need to be scaled at a much larger rate.


Flexible service customization
Various network functions in SDN APPs
DHCP, Firewall, DDoS detector, and etc.
New opportunities
Global network view + SDN APPs = Intelligent & innovative network/security services
Automated network management
APP
Global network view
SDN controller
OpenFlow
Host A
Host B
Host C
OpenFlow Switch
OpenFlow Switch
OpenFlow Switch
OpenFlow Switch

- ❑ SDN grants the ability to manage a network from a centralized perspective.
- ❑ SDN virtualizes both the data and control planes allowing the user to provision physical and virtual elements from one location.
- ❑ SDN gives the user more scalability providing the ability to provision resources at will you can change your network infrastructure at a moment's notice.

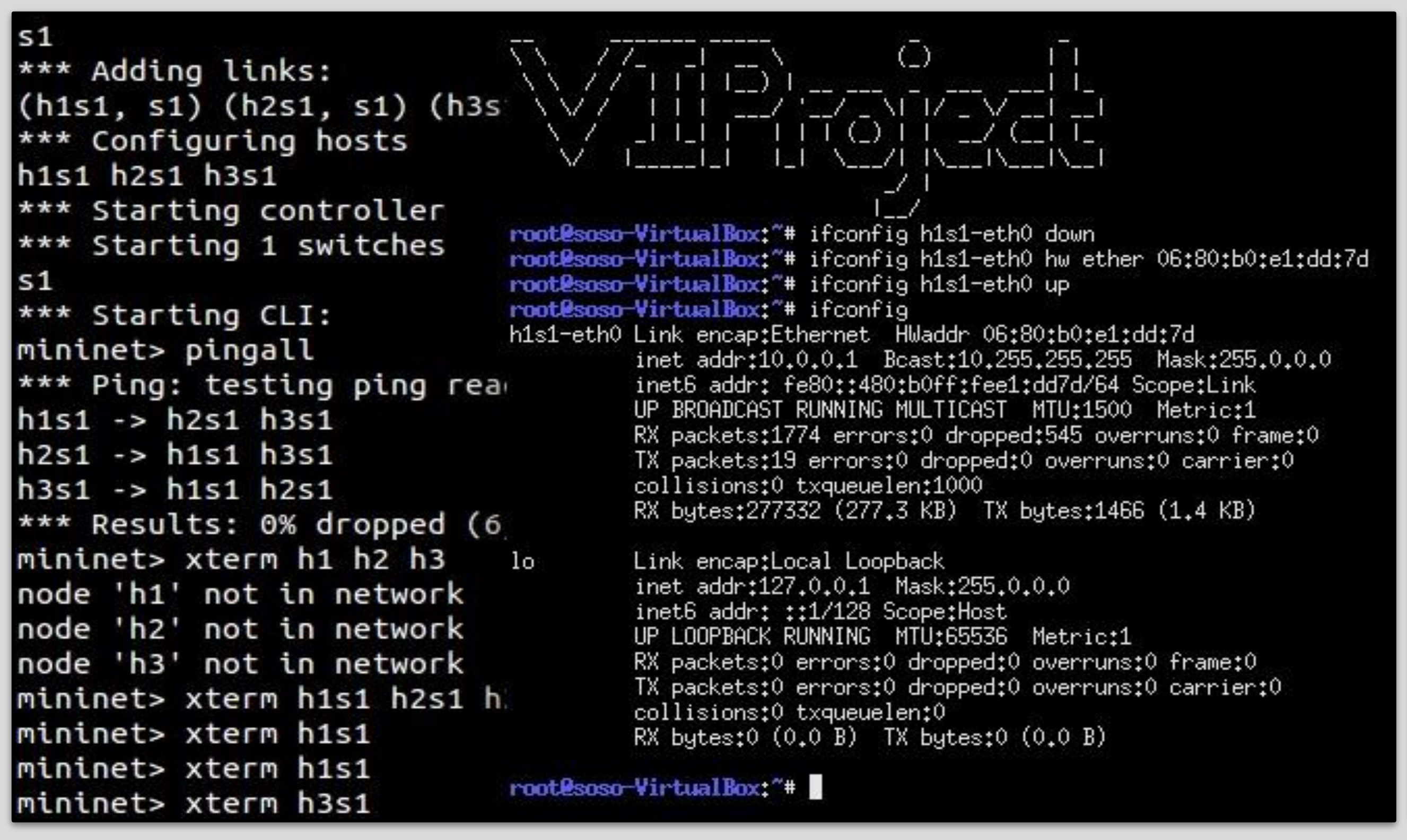
Disadvantages of SDN

- ❑ Increased complexity which allows hackers to control network operations in arbitrary ways, confuse or blind the defenders, and create inconsistencies
- ❑ Inability to directly manage individual devices, leading to increased maintenance.
- ❑ Increased latency due to infrastructure being virtualized.

Leading SDN Market Players:

- ❑ Cisco Systems Inc.
- ❑ IBM Corporation
- ❑ Hewlett Packard Enterprise
- ❑ VMware
- ❑ Juniper Networks
- ❑ Huawei Technologies Co. Ltd.

Environment Setup

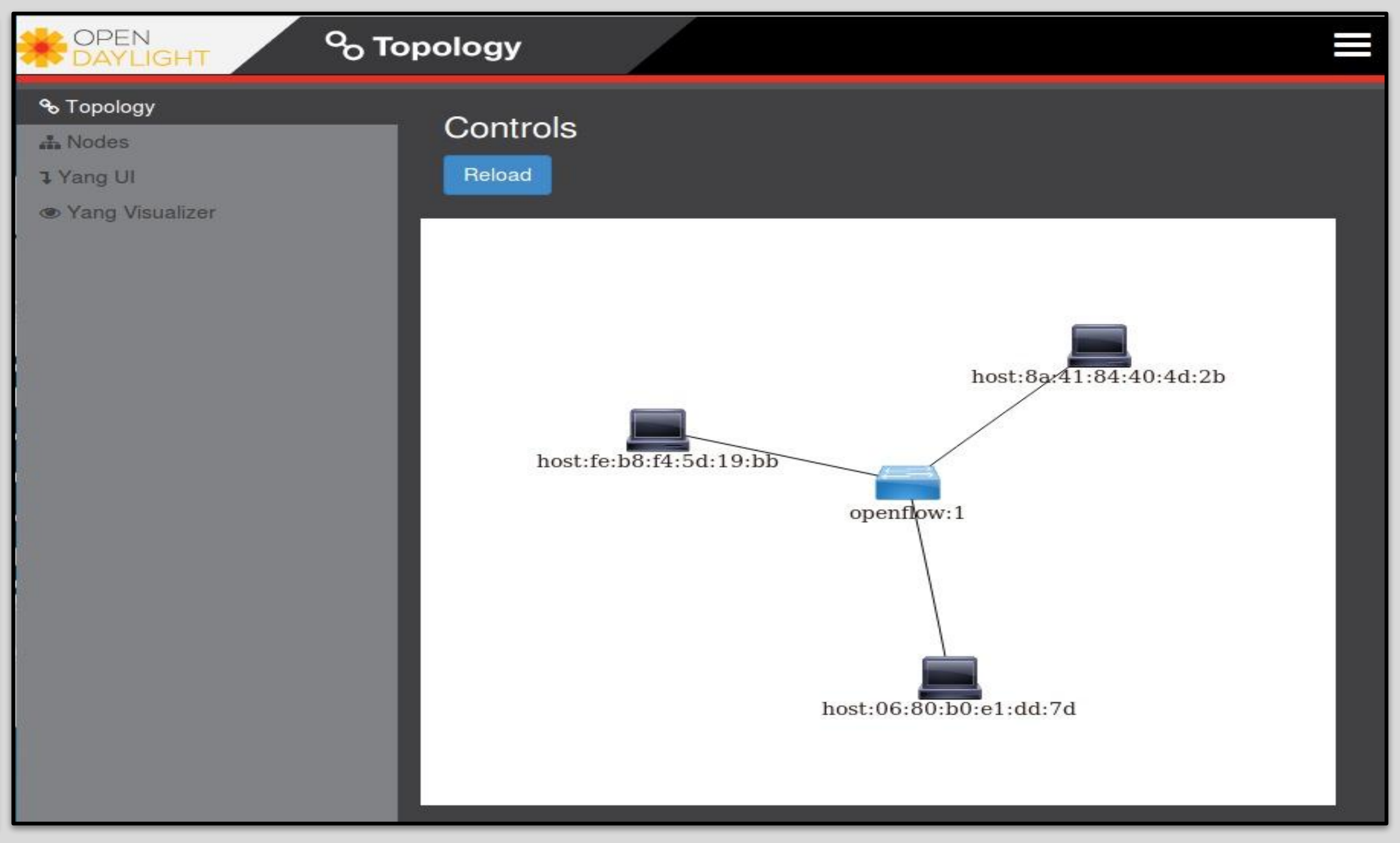


```
s1
*** Adding links:
(h1s1, s1) (h2s1, s1) (h3s
*** Configuring hosts
h1s1 h2s1 h3s1
*** Starting controller
*** Starting 1 switches
s1
*** Starting CLI:
mininet> pingall
*** Ping: testing ping rea
h1s1 -> h2s1 h3s1
h2s1 -> h1s1 h3s1
h3s1 -> h1s1 h2s1
*** Results: 0% dropped (6
mininet> xterm h1 h2 h3
node 'h1' not in network
node 'h2' not in network
node 'h3' not in network
mininet> xterm h1s1 h2s1 h
mininet> xterm h1s1
mininet> xterm h1s1
mininet> xterm h3s1
```

```
root@soso-VirtualBox:~# ifconfig h1s1-eth0 down
root@soso-VirtualBox:~# ifconfig h1s1-eth0 hw ether 06:80:b0:e1:dd:7d
root@soso-VirtualBox:~# ifconfig h1s1-eth0 up
root@soso-VirtualBox:~# ifconfig
h1s1-eth0 Link encap:Ethernet  HWaddr 06:80:b0:e1:dd:7d
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::480:b0ff:fee1:dd7d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1774 errors:0 dropped:545 overruns:0 frame:0
          TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:277332 (277.3 KB)  TX bytes:1466 (1.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@soso-VirtualBox:~# 
```



Implementation

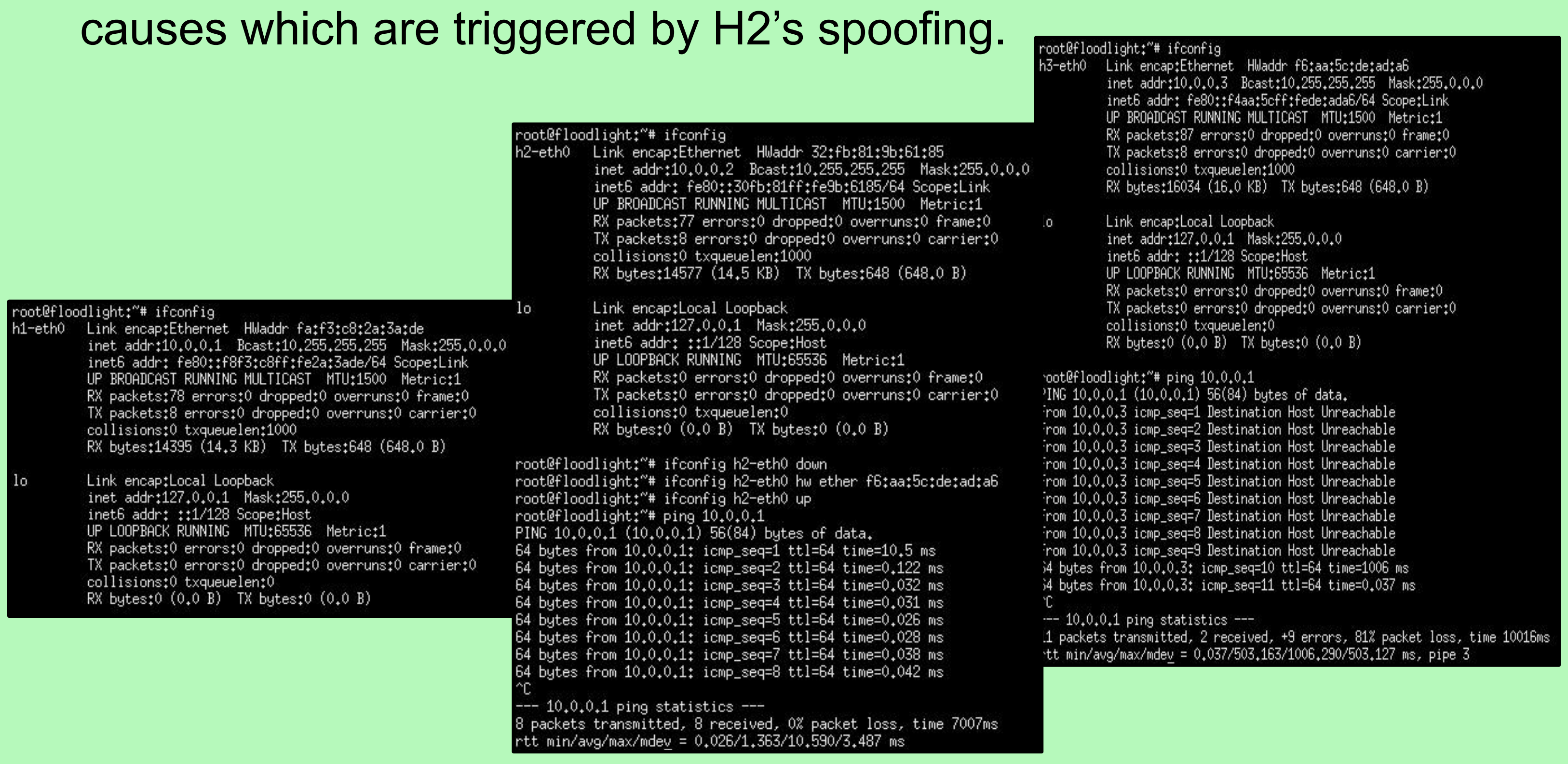
```
OFSwitchBase
DPID[00:00:00:00:00:00:00:02]:
-> Receive Event OFPacketInVer13
-> Init Function
processPacketInMessage
-> Init Function addToPortMap
-> Write Variable
macVlanToSwitchPortMap -254649434
-> Return Function addToPortMap
-> Init Function getFromPortMap
-> Read Variable
macVlanToSwitchPortMap -517582621
-> Return Function getFromPortMap
-> Init Function pushPacket
-> Return Function pushPacket
-> Init Function writeFlowMod
-> send Message FLOW_MOD
-> Return Function writeFlowMod
-> Init Function writeFlowMod
-> send Message FLOW_MOD
-> Return Function writeFlowMod
-> Return Function
processPacketInMessage
-> Return Function receive
```

```
OFSwitchBase
DPID[00:00:00:00:00:00:00:02]:
-> Receive Event
OFPacketInVer13
-> Init Function
processPacketInMessage
-> Init Function addToPortMap
-> Write Variable
macVlanToSwitchPortMap
-517582621
-> Return Function addToPortMap
-> Init Function getFromPortMap
-> Read Variable
macVlanToSwitchPortMap null
-> Return Function
getFromPortMap
-> Init Function
writePacketOutForPacketIn
-> Return Function
writePacketOutForPacketIn
-> Return Function
processPacketInMessage
-> Return Function receive
```

```
OFSwitchBase
DPID[00:00:00:00:00:00:00:01]:
-> Receive Event OFPacketInVer13
-> Init Function
processPacketInMessage
-> Init Function addToPortMap
-> Write Variable
macVlanToSwitchPortMap -517582622
-> Return Function addToPortMap
-> Init Function getFromPortMap
-> Read Variable
macVlanToSwitchPortMap -254649433
-> Return Function getFromPortMap
-> Init Function pushPacket
-> Return Function pushPacket
-> Init Function writeFlowMod
-> send Message FLOW_MOD
-> Return Function writeFlowMod
-> Init Function writeFlowMod
-> send Message FLOW_MOD
-> Return Function writeFlowMod
-> Return Function
processPacketInMessage
-> Return Function receive
```

```
OFSwitchBase
DPID[00:00:00:00:00:00:00:01]:
-> Receive Event
OFPacketInVer13
-> Init Function
processPacketInMessage
-> Init Function addToPortMap
-> Write Variable
macVlanToSwitchPortMap
-254649433
-> Return Function addToPortMap
-> Init Function getFromPortMap
-> Read Variable
macVlanToSwitchPortMap null
-> Return Function
getFromPortMap
-> Init Function
writePacketOutForPacketIn
-> Return Function
writePacketOutForPacketIn
-> Return Function
processPacketInMessage
-> Return Function receive
```

- ❑ Setup the environment using OpenDayLight SDN Controller
- ❑ Start mongodb to collect and store data from mininet
- ❑ Create a three-switch topology using mininet and openflow protocol
- ❑ Prepare for the MAC spoofing attack within node H1,H2, and H3
- ❑ Check all three hosts network configuration
- ❑ H2 is spoofing H3's MAC address then pings to H1
- ❑ At the same time, H3 cannot make the connection with H1 anymore
- ❑ ForenGuard is able to diagnose why H3 lost connection with H1.
- ❑ By checking the diagnosis results, there are four events that are the root causes which are triggered by H2's spoofing.



```
root@floodlight:~# ifconfig
h1-eth0   Link encap:Ethernet  HWaddr fa:f3:c8:2a:3a:de
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::f8f3:c8ff:fe2a:3ade/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:78 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14395 (14.3 KB)  TX bytes:648 (648.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
root@floodlight:~# ifconfig
h2-eth0   Link encap:Ethernet  HWaddr 32:fb:81:9b:61:85
          inet addr:10.0.0.2  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::30fb:81ff:fe9b:6185/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:77 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14577 (14.5 KB)  TX bytes:648 (648.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@floodlight:~# ifconfig h2-eth0 down
root@floodlight:~# ifconfig h2-eth0 hw ether f6:aa:5c:de:ad:a6
root@floodlight:~# ifconfig h2-eth0 up
root@floodlight:~# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=10.5 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.122 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.032 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.031 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.026 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.028 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.038 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.042 ms
^C
--- 10.0.0.1 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7007ms
rtt min/avg/max/mdev = 0.026/1.363/10.590/3.487 ms
```

```
root@floodlight:~# ifconfig
h3-eth0   Link encap:Ethernet  HWaddr f6:aa:5c:de:ad:a6
          inet addr:10.0.0.3  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::f4aa:5cff:fede:ada6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:87 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16034 (16.0 KB)  TX bytes:648 (648.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

oot@floodlight:~# ping 10.0.0.1
ING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
rom 10.0.0.3 icmp_seq=1 Destination Host Unreachable
rom 10.0.0.3 icmp_seq=2 Destination Host Unreachable
rom 10.0.0.3 icmp_seq=3 Destination Host Unreachable
rom 10.0.0.3 icmp_seq=4 Destination Host Unreachable
rom 10.0.0.3 icmp_seq=5 Destination Host Unreachable
rom 10.0.0.3 icmp_seq=6 Destination Host Unreachable
rom 10.0.0.3 icmp_seq=7 Destination Host Unreachable
rom 10.0.0.3 icmp_seq=8 Destination Host Unreachable
rom 10.0.0.3 icmp_seq=9 Destination Host Unreachable
4 bytes from 10.0.0.3: icmp_seq=10 ttl=64 time=1006 ms
4 bytes from 10.0.0.3: icmp_seq=11 ttl=64 time=0.037 ms
C
-- 10.0.0.1 ping statistics ---
1 packets transmitted, 2 received, +9 errors, 81% packet loss, time 10016ms
tt min/avg/max/mdev = 0.037/503.163/1006.290/503.127 ms, pipe 3
```

Future work

- ❑ Try to install malicious applications on the SDN controller to gain control of the network.
- ❑ Find ways to defend against incoming attacks (ex: using DELTA).

Acknowledgements

This material is based upon work supported in part by the SUCCESS (Secure Communication and Computer Systems) Lab at TAMU. We thank Professor Dietrich for his guidance and motivation for our research and we also thank Professor Gu for his assistance with the Forenguard Tool, and Haopei Wang for the comments that greatly improved the implementation and understanding the concept of how Forenguard works with ODL Controller.

References

1. M. Dacier, H. Konig, R. Cwalinski, F. Kargl and S. Dietrich, "Security Challenges and Opportunities of Software-Defined Networking", IEEE Security & Privacy, vol. 15, no. 2, pp. 96-100, 2017. Available: 10.1109/msp.2017.46
2. J. Cao, Q. Li, R. Xie, H. Sun, G. Gu, M. Xu, and Y. Yang, "The CrossPath Attack: Disrupting the SDN Control Channel via Shared Links"
3. H. Wang, G. Yang, p, Chinprutthiwong, L. Xu, Y. Zhang, and G. Gu, "Towards Fine-grained Network Security Forensics and Diagnosis in the SDN Era", 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), Available: 10.1145/3243734.3243749
4. S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," 2015 International Conference on Computing, Networking and Communications (ICNC), 2015. Available: 10.1109/iccnc.2015.7069319