Security Issues in Software Defined Networking

Anupama Potluri

School of Computer and Information Sciences
University of Hyderabad

Outline

Outline

The Three Planes of Networking

Networking protocols can be divided into three planes of function namely:

- Data Plane: consisting of the forwarding of actual user data via applications using TCP/IP
- Control Plane: Consisting of routing protocols that find the path to send data on such as OSPF, BGP, LDP, RSVP etc.
- Management Plane: Protocols such as SNMP that help in configuration and monitoring of network elements such as switches, routers, servers etc.

Software Defined Networking: Motivation I

SDN was triggered by many different developments in the industry: [1]

1. Availability of good switch chips that make fast switching possible
2. Big data centers
3. Frustration with big vendors that leads to tie-in with their equipment and prevents fast innovation and high costs
4. Virtualization is difficult to achieve with standard networking

Software Defined Networking: Motivation II

SDN was motivated by many factors especially those arising out of data center networking. These are:

1. **Manageability:** Manual configuration of routers/switches etc. using low-level constructs such as IP and MAC addresses instead of higher level constructs such as hostnames/user IDs for setting policies and monitoring.
2. **Control Plane Modularity:** Lack of control plane modularity results in various routing protocols gathering the network topology which consists of 95% of the code whereas the actual path calculation such as a Dijkstra algorithm takes 5% of the code.

Software Defined Networking: Motivation III

3. **Virtualization:** OpenvSwitch is a software virtualized switch that allows multiple VMs on a single server to communicate with each other while supporting typical networking policies such as access lists, firewalls etc. Virtual switches migrate along with their VMs and this requires a huge amount of work in traditional networking.
4. **Programmable and Generalised Data Plane:** A data plane that can be programmed through a well-known interface allows for generalised actions on the packets flowing through a switch. This allows for horizontal scaling and removes vendor lock-in.

Software Defined Networking: Motivation IV

5. **Innovation:** Introducing new control plane protocols is a simple matter when topology is provided as an interface to the new protocol and the programmable data plane. Allows for new data plane protocols through generalised programmable data plane.
6. **Networking Principles:** The three principles are presented in a later slide.

Software Defined Networking: Definition

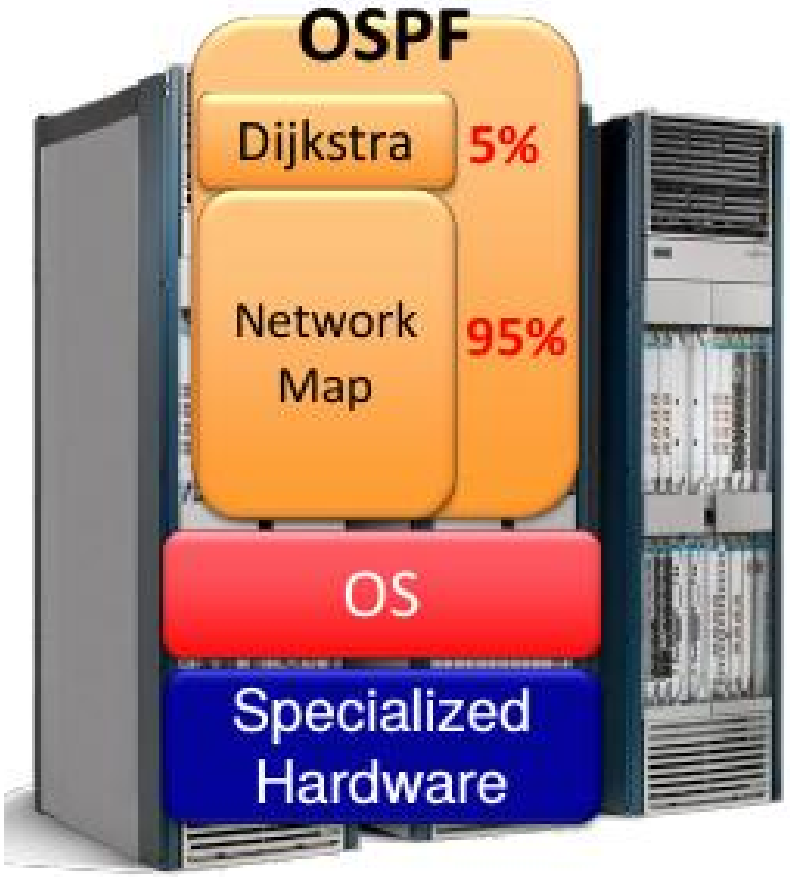
Nick Mckeown defines Software Defined Networking as: [1]

A network in which the control plane is physically separate from the forwarding plane.

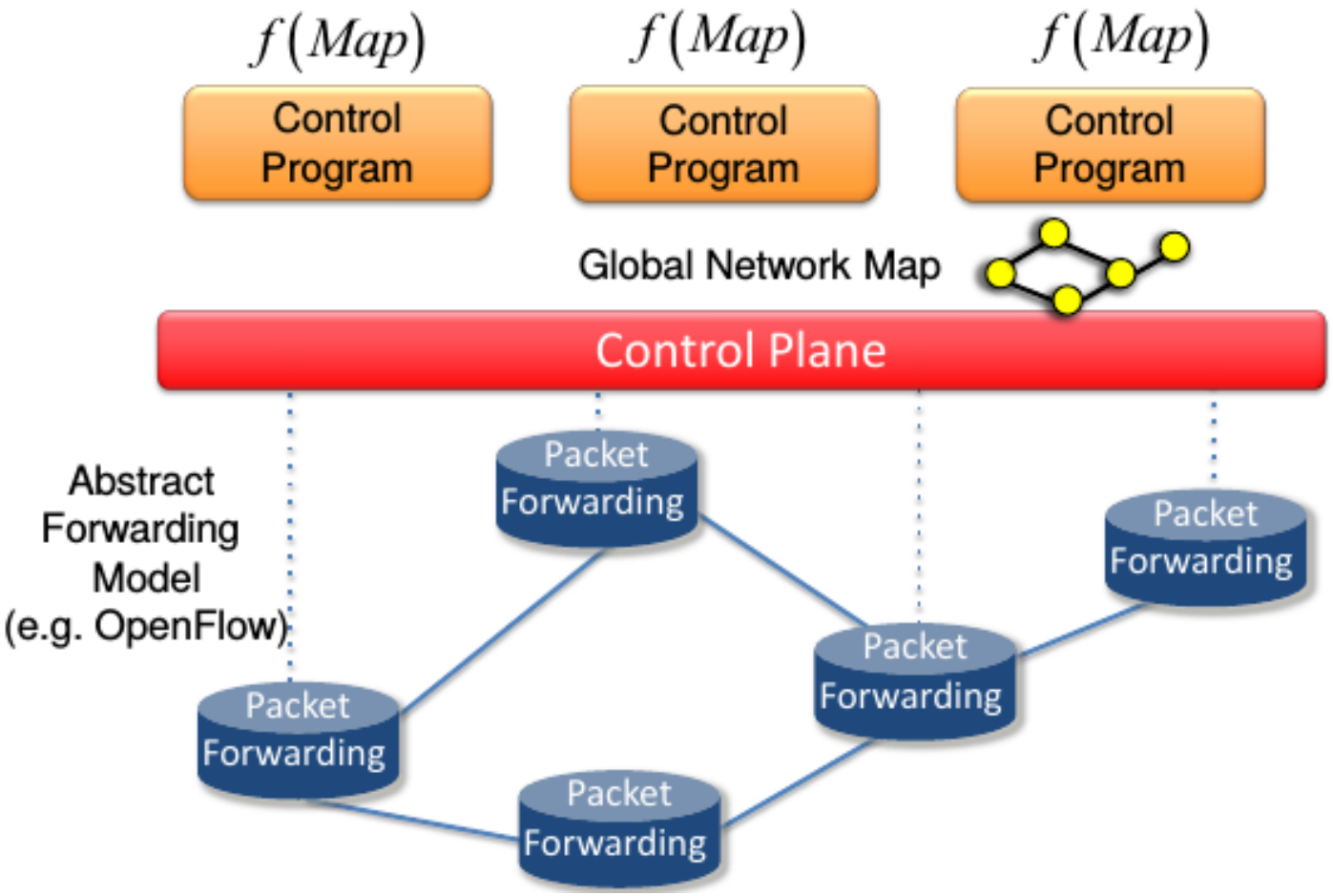
AND

A single control plane controls several forwarding devices.

Current Networking Infrastructure [1]



Software Defined Networking Architecture [1]



Networking Principles

Networking is extremely complex and is just a bag of protocols [2]. The complexity of networking comes from the following:

Operate without communication guarantees

Need Distributed State Abstraction

Compute the configuration of each physical device

Need Specification Abstraction

Operate within given network-level protocol

Need Forwarding Abstraction

Distributed State Abstraction [2]

1. Shield mechanisms from vagaries of distributed state
   - While allowing access to this state
2. Natural abstraction: global network view
   - Annotated network graph provided through an API
3. Control mechanism is now program using API
   - No longer a distributed protocol, now just a graph algorithm
   - e.g. Use Dijkstra rather than Bellman-Ford

Specification Abstraction [2]

1. Control program should express desired behavior
2. It should not be responsible for implementing that behavior on physical network infrastructure
3. Natural abstraction: simplified model of network
   - Simple model only enough detail to specify goals

This is “network virtualization”

Forwarding Abstraction [2]

1. Control plane needs flexible forwarding model
2. Abstraction should not constrain control program
   - Should support whatever forwarding behaviors needed
3. It should hide details of underlying hardware
   - Crucial for evolving beyond vendor-specific solutions

OpenFlow: Programmable Data Plane Protocol I

OpenFlow [4] [5] was proposed as a standard interface for data plane abstraction. It is still not generalised enough. An OpenFlow switch consists of three parts:

1. A *Flow Table* having rules to match, action and counters.
2. A *Channel* to communicate with a controller on which commands are sent to the switch and counters are read.
3. The *OpenFlow Protocol* which is the interface between the controller and the switches.

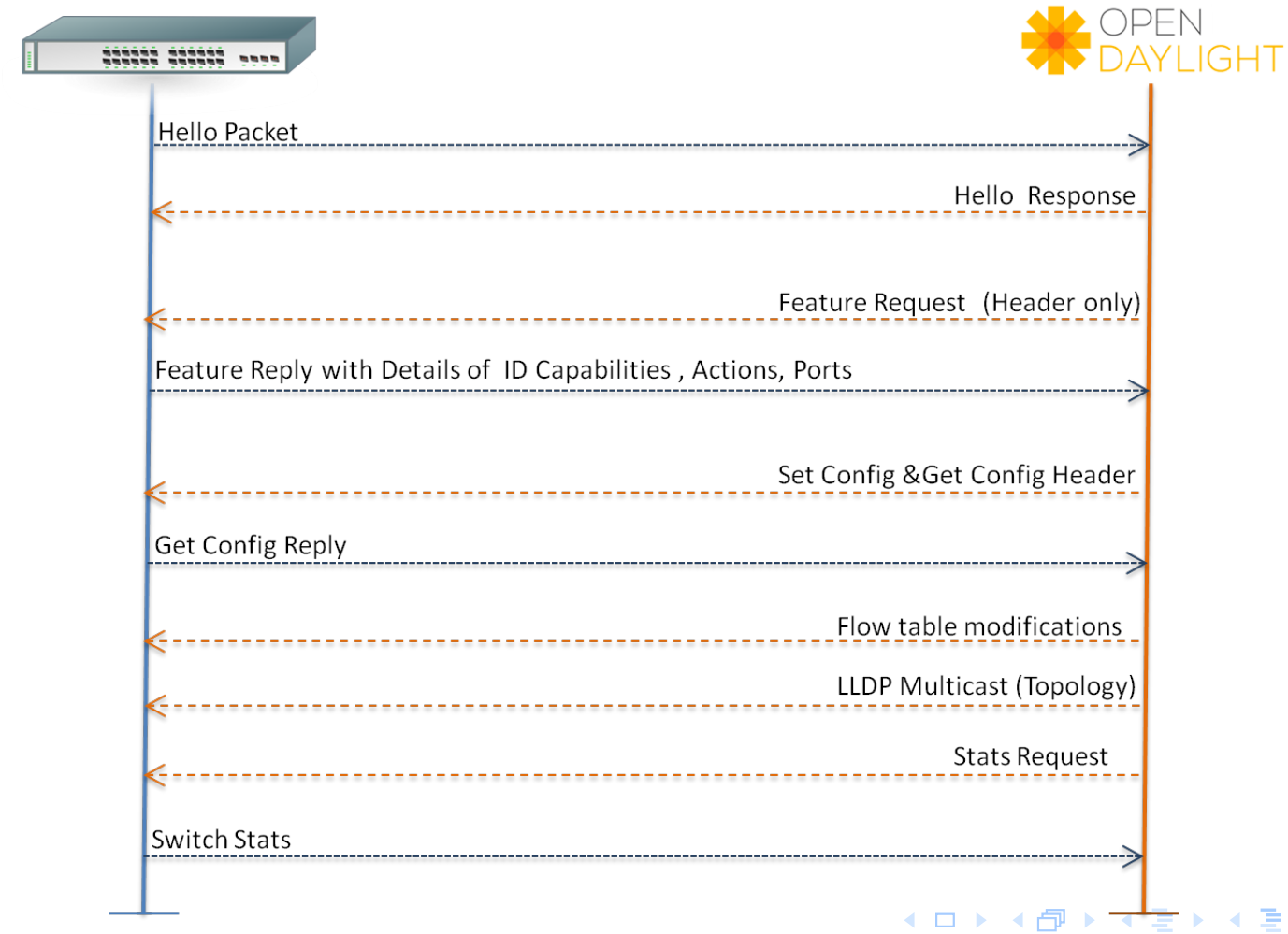
OpenFlow: Programmable Data Plane Protocol II

The three basic actions associated with each rule that all dedicated OF switches must support are:

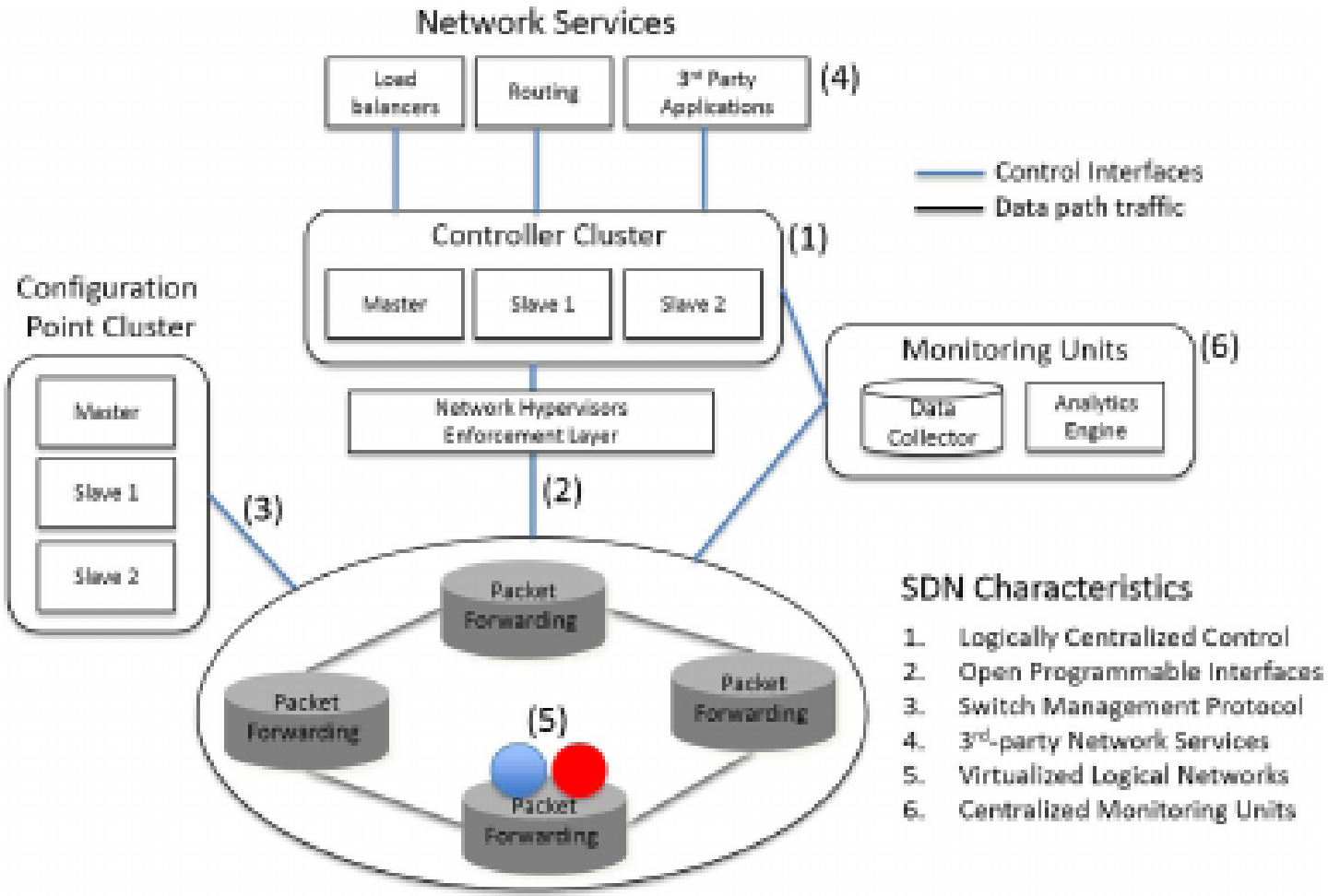
1. Forward this flow's packets to a given port (or ports). This allows packets to be forwarded along a given path to the destination.
2. Encapsulate and forward the traffic to a controller using the secure channel to the controller. This is usually done for the intiation packet of a flow so that the controller can consult the policies for this packet and install the required rules in the switches on the path.
3. Drop the flow's packets.

OpenFlow-enabled commercial switches have a fourth rule to augment the three above: forward the traffic through the switch's normal processing.

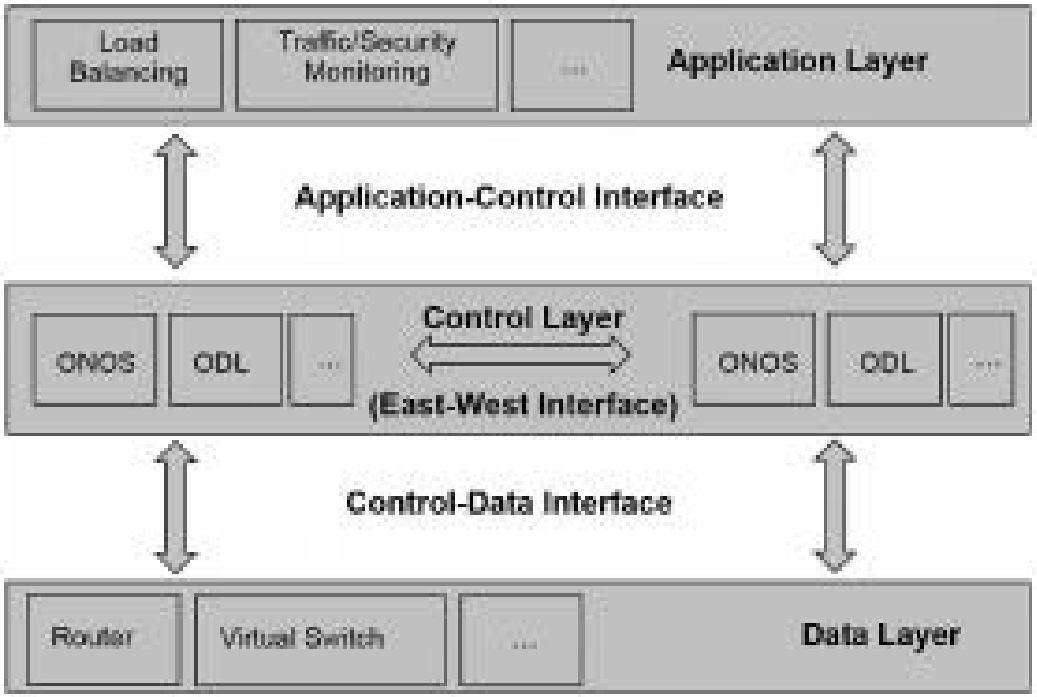
OpenFlow Message Exchange between Switch and Controller



SDN Characteristics [6]



SDN Functional Architecture [6]



Terminology of APIs

1. **Northbound API:** API between the controller and the applications
2. **Southbound API:** API between control and data planes
3. **East-West API:** API between different controllers

Outline

Basic Properties of a Secure Communications Network

Basic properties of a secure communications network are:

1. Confidentiality
2. Integrity
3. Availability of Information

These are supported by techniques of

- Authorization
- Authentication
- Encryption

The rest of the slides in this section are taken primarily from [6].

Attacks and Vulnerabilities in SDN I

1. **Unauthorized Access:** A compromised controller/application can gain access to network elements and manipulate the actions.
2. **Data Leakage:**
   - Analysis of time taken for a packet from ingress to egress port of a switch gives information regarding the proactive or reactive installation of rules. This can then be used to launch DoS attacks.
   - As part of network virtualization, when logical switches are instantiated on top of OF-enabled physical switches or in OVSes, the compromised switches can be used to extract credentials of tenants other than our own.

Attacks and Vulnerabilities in SDN II

3. **Data Modification:** Man-in-the-middle attacks are very much possible in SDN between the Controller and Data plane entities as OpenFlow has not made TLS mandatory. This allows modification of processing rules by intermediate entities that can hijack or play havoc with the flows. Further, all network virtualization layers that might exist between controller and data plane may be vulnerable adding further complexity to securing the SDN.
4. **Malicious/Compromised Applications:** Since third-party applications are integrated with the controller using Northbound APIs, any buggy/malicious/compromised application can take control of the network.

Attacks and Vulnerabilities in SDN III

5. **Denial of Service:** This is one of the easiest attacks on the SDN controller where a lot of bogus packets are sent to the switch from an attacking host which result in a flood of *packet-in* messages to the controller resulting in DoS as well as installation of rules in the switch for these packets with the *drop* action that can fill up the precious TCAM space resulting in DoS in switches.
6. **Configuration Issues:** Since SDNs provide easy methods of programming the network, when policies from multiple third party applications are installed in various network elements, it is possible this leads to inconsistencies and create vulnerabilities. Policy consistency checking is an important part of security in SDN as well as conformance to security recommendations such as use of TLS for Controller-Data plane messages.

Attacks and Vulnerabilities in SDN IV

7. **System Level SDN Security:** If an OF switch is disconnected from the controller, it switches to either a *fail secure* or *fail standalone* mode. The operator needs to understand its behaviour and be able to handle the proper accounting and auditing needs that are typical in data centers/ISPs.

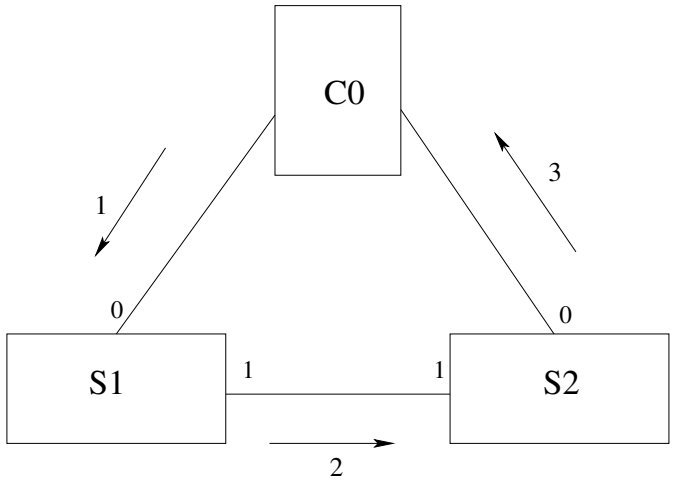
Examples of some simple attacks in SDN

The following slides will demonstrate how some simple attacks such as DoS, ARP poisoning, fake link injection etc. can be launched in SDN.

These attacks were simulated using Mininet [14].

The slides are taken from my student Sarika Gupta's M.Tech. final viva. I thank her for the use of her slides.

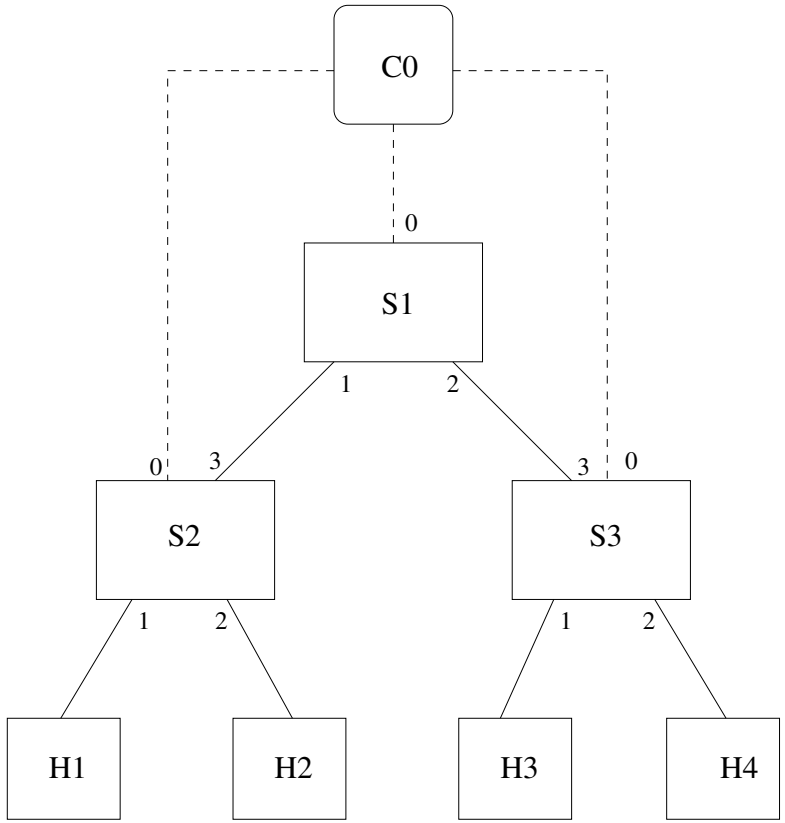
Example: LLDP Fake Link Injection Attack [13] I



1. *packet-out* message from controller to each switch with OFDP containing LLDP payload
2. *LLDP Advertisement* of this message to all the ports
3. *packet-in* message from the switch receiving the advertisement to the controller containing the DPID of the switch and the ingress port number.

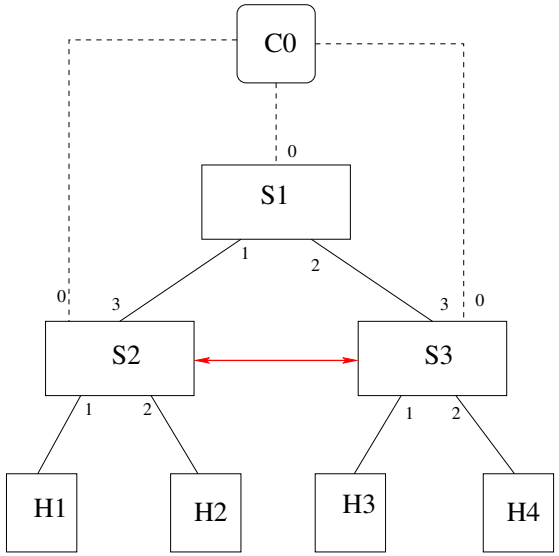
Example: LLDP Fake Link Injection Attack [13] II

Topology created for the LLDP attack



Example: LLDP Fake Link Injection Attack [13] III

Fake link between S2 and S3:

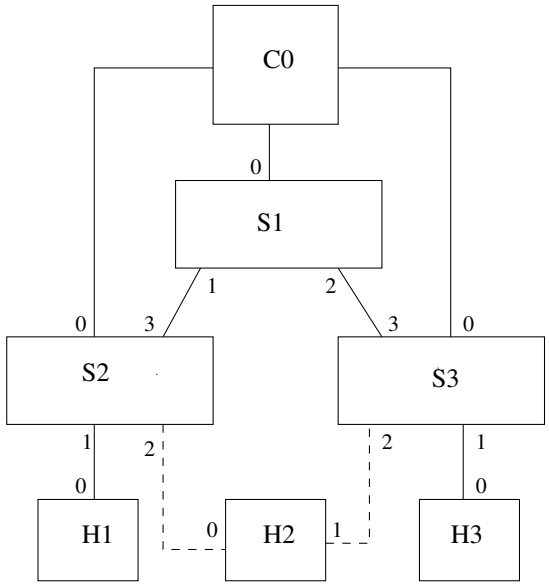


To create a fake bidirectional link:

1. H3 sends the LLDP packet received from port 1 of S3 to host H2
2. H2 sends the LLDP packet received from port 2 of S2 to host H3

Example: LLDP Fake Link Injection Attack [13] IV

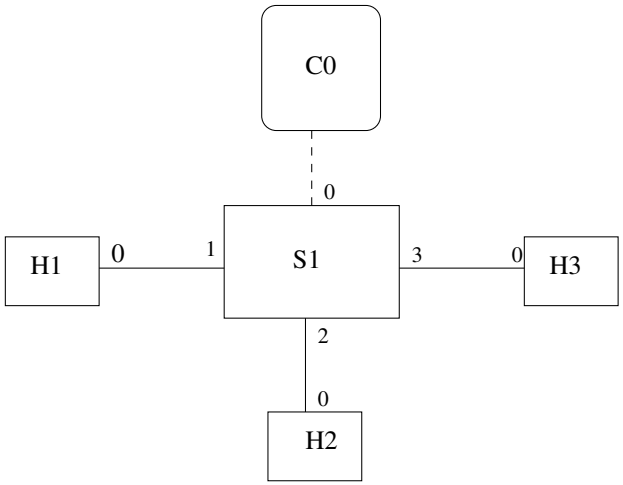
Man-In-The-Middle Attack:



Example: ARP Poisoning Attack using ARP Request Messages [15]

- ARP request is sent by a host to know the MAC address of another host on the network
- In ARP poisoning attack, falsified ARP request/reply messages are sent to a host which links the IP address of another host with the MAC address of the attacker

Topology created for the attack

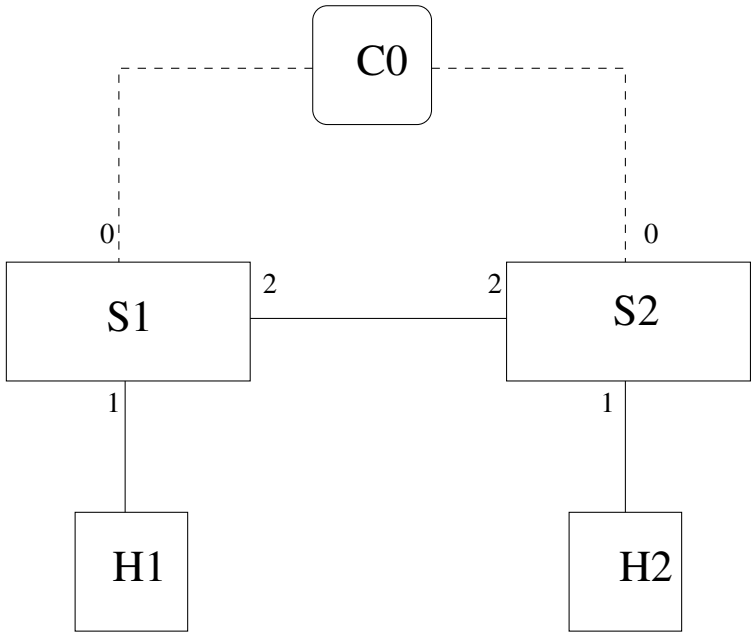


Example: DoS Attack using Multicast packets [15]

- The controller does not install flow rule for multicast packets
- Switch always forwards the multicast packets to the controller
- Attacker can utilize this and can generate *n* number of multicast packets leading to DoS at the controller

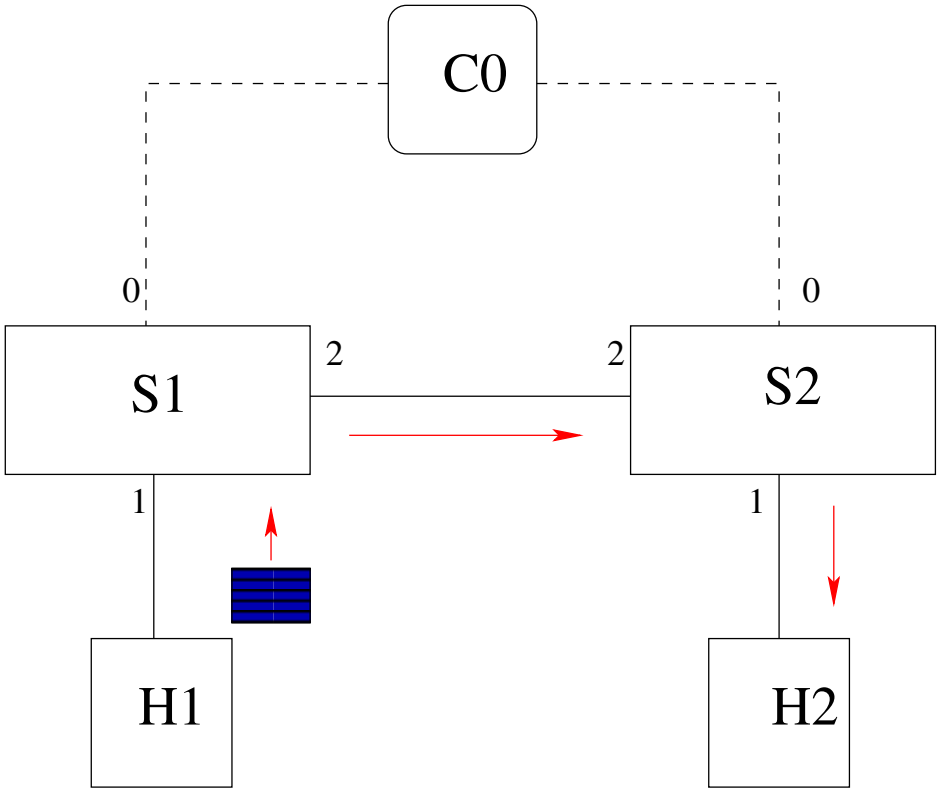
Example: Switch Black Hole Attack [15] I

Switch black hole is a type of denial of service attack, in which the switch silently discards packets without informing the source.



Example: Switch Black Hole Attack [15] II

H1 pings H2: the controller will install rules in the switches to forward the ping packets and ping replies.

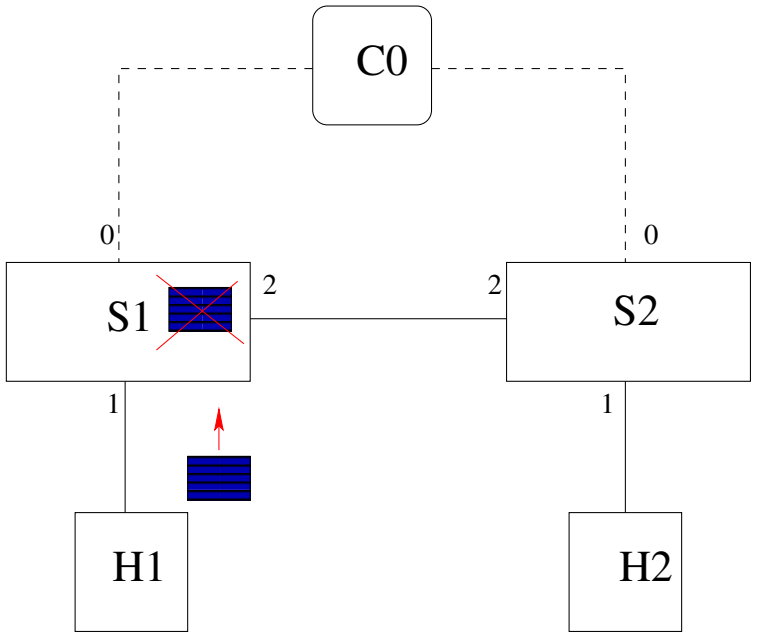


Example: Switch Black Hole Attack [15] III

Malicious switch on the flow path can drop packets leading to black hole.

Rule installed on switch S1 to drop packets :

ovs-ofctl add-flows tcp:127.0.0.1:6634 dl_src=*, priority=65535, idle_timeout=0, actions=drop



Solutions to Security Issues in SDN

Typical solutions to the security issues in SDN involve leveraging the characteristics of SDN itself:

- logically centralised monitoring via collection of statistics
- complete knowledge of topology
- ability to dynamically modify flow rules in the switches.

Solutions to Unauthorized Access

Some of the solutions proposed for unauthorized access to the network by compromising controller/application are:

- Signature-based flow rule installation
- Multiple controllers using Byzantine fault-tolerance algorithm
- Hierarchical system of controllers to reduce the possibility of single point-of-failure as well as the seriousness of compromise
- Implementation of permissions for minimum privilege to applications
- Implementation of permissions to authenticate and authorize an application
- Hosts are authenticated and authorized using IEEE 802.1x and RADIUS

Solutions to Malicious/Compromised Applications

- **FortNOX:** Uses role-based authentication to determine security authorization of applications. Policy conflicts are detected and rule updation depends on whether the new policy is being added by a higher or lower priority application.
- **ROSEMARY:** Each application is run with its own instance of a micro-Network Operating System such that the vulnerability is limited to only its instance of control plane. Monitoring the resource usage by the application also allows to detect rogue applications.
- **LegoSDN:** This looks into the problem of crash of the control layer when the application layer crashes. The proposal is to create a layer between SDN and Apps, a network-wide transaction system that does atomic updates and rollback to avoid inconsistent state on crashes.

Solutions to Denial of Service

- **AVANT-GUARD:** Focuses on TCP SYN flood attack. Solution is to not send any *packet-in* messages to the controller until the 3-way handshake has been completed and time out the other entries.
- **CPRecovery:** Uses a backup controller and provides seamless transition to it when it is detected that the primary controller has failed due to an attack. Uses the saved state to ensure the attack scenario is not transitioned.
- **Virtual source Address Validation Edge (VAVE):** Pre-emptive protection against IP spoofing. An incoming packet that does not match a rule in the OF switch is sent to the controller. If spoofing is detected, the rule is updated to drop packets from this source. Such rules are also aggregated to ensure that the switch does not run out of TCAM space.

Solutions to Configuration Issues

Typical solutions to the mis-configuration issues can be categorized as:

- Detecting Network Errors
- Language-based Resolution (e.g. Frenetic)
- Real-time Policy Checking (e.g., NetPlumber, FlowGuard)
- Consistent Abstractions/Network View
- Formal Verification Methods

Network Security Enhancement using SDN Framework

1. Collect, Detect and Protect
2. Attack Detection (Traffic Analysis) and Prevention (Rule Updating)
3. DoS/DDoS Protection
4. Secure, Scalable Multi-Tenancy
5. Authentication, Authorization and Accounting
6. Security Middleboxes – Architectures and Services

Collect, Detect and Protect

The very characteristics of SDN to collect statistics from the data plane elements as well as dynamic programmability of the data plane can be leveraged to prevent attacks.

- [7] has three modules:
  1. **Collector:** collects statistics with OpenFlow and sFlow
  2. **Anomaly Detection:** analysis of these stats is carried out
  3. **Anomaly Mitigation:** reconfiguration of data plane to mitigate the attack
- NetFuse [10] protects against traffic overloading. Using the information obtained by monitoring the OF control plane messages such as *packet-in* and *stats* messages, it determines if say, the volume of flows to a destination port exceed the normal load. If so, these are rate-limited using adaptive rate-limiting algorithms.

Attack Detection (Traffic Analysis) and Prevention (Rule Updating)

Some of the solutions target specific attacks and suggest prevention mechanisms.

- **AVANT-GUARD [8]:** An example is to collect traffic statistics from individual hosts and compare against a defined threshold. If this is exceeded, then, a flow rule dropping traffic from this host is installed.
- **SnortFlow [9]:** In this the Snort IDS is combined with OF. Snort agents collect information and send to the Snort server which analyses the data and then uses the OF controller to reconfigure the network if intrusion is detected.

DoS/DDoS Protection

- **Lightweight DDoS [12]:** Traffic information such as average packets per flow, average bytes per flow are calculated periodically using statistics gathered from OF-enabled switches. A Self-Organizing Map is run to classify flows into normal or malicious flow and make changes to flow rules accordingly.
- **DDoS Blocking Application (DBA) [11]:** Sophisticated DDoS attacks use a large number of bots (hosts) to send small amount of legitimate-looking traffic. The DBA runs on top of SDN controller and monitors the traffic pattern to determine if there is a DDoS attack going on.

Recommended Best Practices I

1. **Policy Conflict Resolution:** All controller solutions must include policy conflict resolution subsystem.
2. **Mutual Authentication:** Mutual authentication amongst all communicating entities helps prevent data manipulation attacks, impersonation and ensures secure identification of network elements.
3. **Containerized Applications:** Most applications today are bundled with the controller, instantiated as a dynamic module in the controller or integrated as a third-party software with remote access. Instead, applications must be containerized which can:
   - authenticate the application
   - control the application's access rights
   - limit, account for and isolate resource usage for every application

Recommended Best Practices II

4. **Rate-Limiting, Flow Aggregation and Short Timeouts:** Flow aggregation can avoid overflowing of TCAM space, short timeouts allow for malicious flows to timeout faster releasing TCAM space and rate-limiting control messages from data plane allows for prevention of DoS attacks etc.
5. **Logging/Forensics for IDS/IPS:** Logging of critical information is an essential part of any networking. This can be used for writing data analytics applications which can then enhance the capabilities of IDS/IPS.

Summary

SDN is an exciting new paradigm of networking that leads to dynamically programmable networks with virtualized network elements. Its logically centralized structure is both its strength and weakness as an attack on the centralized controller can bring down the whole network. On the other hand, the very same property allows a bird's eye view of the network which allows monitoring of the network easier. Dynamic programmability allows for immediate flow rule modification to drop all malicious flows.
However, there is still a lot of research needed to make SDN secure. Many industry groups are working on this:

1. ONF Security Discussion Group
2. ETSI ISG NFV Security Expert Group
3. ITU-T Standardization of SDN

References I

[1] http://www.comsnets.org/archive/2014/doc/ NickMcKeownsSlides.pdf

[2] http://opennetsummit.org/archives/oct11/shenker-tue.pdf

[3] https://www.youtube.com/watch?v=YHeyuD89n1Y

[4] OpenFlow. http://en.wikipedia.org/wiki/OpenFlow.

[5] OpenFlow: Enabling Innovation in Campus Networks. Nick McKeown et. al., SIGCOMM Comput. Commun. Rev., pages 69–74, April 2008. http://doi.acm.org/10.1145/1355734.1355746

[6] A Survey of Security in Software Defined Networks. Sandra Scott-Hayward, Sriram Natarajan and Sakir Sezer. IEEE Communications Surveys and Tutorials, Vol. 18, No. 1, First Quarter 2016.

[7] Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. Giotis, K. et. al., Computer Networks, Vol. 62, Pages 122–136.

References II

[8] AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks. S.Shin, et. al., Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (CCS'13), pages 413-424.

[9] SnortFlow: A OpenFlow-Based Intrusion Prevention System in Cloud Environment. T.Xing, et. al., Proceedings of the 2013 Second GENI Research and Educational Experiment Workshop (GREE'13), pages 89-92.

[10] NetFuse: Short-circuiting traffic surges in the cloud. Y.Wang et. al., IEEE ICC-2013, pages 3514-3518.

[11] A SDN-oriented DDoS blocking scheme for botnet-based attacks. S. Lim et. al., in Proc. 6th ICUFN, 2014, pp. 63–x68.

[12] Lightweight DDoS flooding attack detection using NOX/OpenFlow. R. Braga, E. Mota and A. Passito, in Proc. IEEE 35th Conf. LCN, 2010, pp. 408–415.

References III

[13] Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures.Sungmin Hong et. al., NDSS'15.

[14] A network in a laptop: rapid prototyping for software-defined networks. Bob Lantz, Brandon Heller and Nick Mckeown, Hotnets-IX, Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Article No. 19, 2010.

[15] SPHINX: Detecting Security Attacks in Software-Defined Networks. Mohan Dhawan et. al., NDSS'15.