COMPUTERS & SECURITY 53 (2015) 79–108

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Security of Software Defined Networks: A survey

Izzat Alsmadi[*], *Dianxiang Xu*

Department of Computer Science, Boise State University, 1910 University Drive, Boise, ID 83725, USA

ARTICLE INFO

Article history:
Received 21 January 2015
Received in revised form
19 March 2015
Accepted 21 May 2015
Available online 4 June 2015

Keywords:
Software defined networking
Security
Software Defined Security
Networking
Network security

ABSTRACT

Software Defined Networking (SDN) has emerged as a new network architecture for dealing with network dynamics through software-enabled control. While SDN is promoting many new network applications, security has become an important concern. This paper provides an extensive survey on SDN security. We discuss the security threats to SDN according to their effects, i.e., Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, and Elevation of Privilege. We also review a wide range of SDN security controls, such as firewalls, IDS/IPS, access control, auditing, and policy management. We describe several pathways of how SDN is evolving.

1. Introduction

Today Internet-based systems, such as cloud services and social networks, change their network requirements (e.g., bandwidth demand, topology, and routing information) dynamically. Hardwired networks, however, have very limited ability to cope up with such frequent changes. To address this issue, Software Defined Networking (SDN) has emerged as a new network architecture that allows for more flexibility through software-enabled network control. The basic idea is to separate control plane from data plane into a program, called controller, for dynamic orchestration of network components.

While SDN is enabling new network applications, security has become an important concern as security is not yet a built-in feature in the SDN architecture. Research has shown that various security attacks can be conducted against SDN through different network components. As SDN relies on software, code vulnerabilities also have an important impact on SDN security. Moreover, SDN offers abundant opportunities for implementing security controls as SDN controller applications. Such software solutions can enable more flexible security controls in dynamic and virtualized network environments. They provide a practical means for software-defined security control.

In this paper, we conduct an extensive survey on SDN security. We study the security threats to SDN according to their effects, i.e., Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service (DoS), and Elevation of Privilege (STRIDE). This classification of security threats, known as STRIDE (Howard and LeBlanc, 2001), has been widely applied to threat modeling of computer, software, and network systems. We also review a wide range of SDN security control applications, such as firewalls, Intrusion Detection/Protection

* *Corresponding author*. Tel.: +1 2089726299.
E-mail address: izzatalsmadi@boisestate.edu (I. Alsmadi).
http://dx.doi.org/10.1016/j.cose.2015.05.006

System (IDS/IPS), access control, auditing, and policy management. In addition, we discuss several open issues and research topics that worth further investigation.

The rest of the paper is organized as follows. To facilitate discussions on SDN security, Section 2 briefly introduces the architecture of SDN. Section 3 reviews SDN security threats and countermeasures according to the STRIDE classification. Section 4 focuses on SDN security controls. Section 5 concludes this paper.

2. SDN architecture

SDN aims at providing open, centralized, decoupled, programmable, flow-based, and dynamic network switching mechanisms.

- **Open**: Traditional networking components such as switches and routers are vendor specific. They provide limited ability for users to experiment their own networking protocols on live networks with real traffic. With SDN, developers can develop middle-boxes that interact with the controller and network switches. Many controller platforms are open source, such as OpenDayLight, Floodlight, Ryu, and Beacon.
- **Centralized**: The control of different switches is co-allocated in one logical place, i.e. the controller. In design terms, this is about splitting "the what" from "the how". Such architecture is capable of handling very dynamic network situations. For example, network traffic based on dynamically changing usage demands may require switches to suddenly join or leave a particular virtual network.
- **Decoupled:** Network functionalities include tasks related to two in-cohesive components: control and data. Splitting data from control improves overall reusability and maintainability of network systems. Policies are decoupled from switches' rules. User level security policies should be expressive and close to users' language and terms, whereas network level information (i.e. Flow or firewall rules) should be simple and close to network attributes. Furthermore, in SDN, virtual or logical network is decoupled from the physical network.
- **Programmable:** Controller can be accessed and programmed by user level applications or middle-boxes. Such programmability is considered a major characteristic of SDN. Developers can modify open source controller modules. Programmability in SDN can be extended far more than just writing applications or modifying controller functionality. It can offer network administrators the ability to write policies and monitor OpenFlow networks.
- **Flow-based management**: SDN shifts networks from IP-based to flow-based management and control. While flow-level control is technically possible in traditional networks, routing protocols make decisions based on IP addresses. SDN is a flow-based architecture, where forwarding decisions in switches are made according to flows. Records or rules in switches and firewalls are per flow. This will impact many applications that depend on network traffic. For example, typical firewall rules deny or permit packets based on source or destination IP, MAC addresses or ports. Future firewall rules may become more dynamic and be updated frequently based on real time traffic.
- **Dynamic**: A major advantage of software over hardware is that it can accommodate frequent changes for more dynamics and flexibility. Configuration or reconfiguration of hardware is labor intensive. Software can be programmed to respond to activities and make decisions dynamically. This is extremely important to those applications with highly dynamic bandwidth demand, such as cloud computing, dynamic datacenters, smart devices, and social networks.

Fig. 1 shows an overall architecture of SDN, with the consideration of the most recent technological advances. We will use this architecture as a reference throughout the paper.

SDN architecture can be divided largely into controller core or internal modules and external modules. The core modules can be seen in the central or inner rectangle of Fig. 1 including: Network manager, APIs, Network Operating System (NOS), internal services and drivers. A bare bone controller should at least contain those modules that represent main functionalities. External modules of SDN controller can be divided logically into 4 parts interacting with the controller from the 4 different directions:

- Southbound section: This can be considered as the most currently popular direction of interaction between the controller and its switches. The OpenFlow protocol is a southbound interface to the controller that represents the connecting bridge between the controller and forwarding elements such as switches. Having a non-vendor-specific protocol is important to allow all vendors join this open architecture.
  Recent improvements on SDN architecture proposed a service abstraction layer (SAL) or a hypervisor in this section to enable controller and protocols to evolve without impacting each other. Open source FlowVisor can be considered as an instance of SALs.
- Northbound section: All types of applications (also called middle-boxes) that want to interact with the controller and underlying network or traffic can be typically designed in this section. There are some proposals of a standard protocol or interface similar to OpenFlow. REST API can be considered a significant achievement toward that goal although it does not represent a standard and a secure method to communicate with the controller. In addition, there are no clear agreements regarding handshaking methods between the northbound applications and the controller, how to manage communicative permissions or authorization, or how to handle decisions' conflicts. SDN-specific policy programming languages such as Pyretic and Frenetic also communicate with the controller through northbound section. It is highly desirable that all the security applications such as firewall, access control, IDS/IPS use a common API for interactions with the controller.

From security perspectives, many concerns are raised that enabling applications to interact with controller that may have special privileges can cause several security risks.

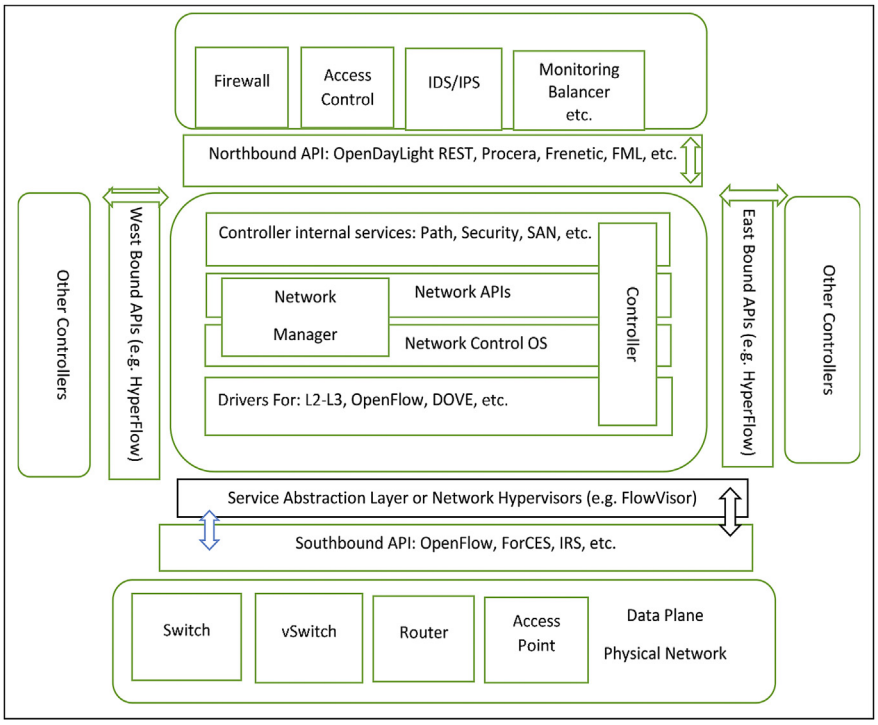

Fig. 1 – SDN architecture.

Security may come unintentionally from normal applications or network users tampering network configuration or architecture. Risks may also come intentionally from hacking methods that can possibly compromise those applications.

- East and West sections. Currently both east and west bound sections are used for the same purpose; management of an SDN distributed architecture. For examples different instances of controllers in a distributed architecture should communicate frequently and pass control and management information. There are different possible SDN distributed architectures to implement. In one type, a vertical or hierarchical architecture may exist where a top layer controller may have several low level controllers. Upper layer controller main functionalities include control, management, monitoring and tasks' distribution for the different SDN low level instances. In other types, different controllers may perform different functionalities (e.g. load balancing, security control, layer 3 switching, layer 4 routing, etc.). Further, controllers can share same tasks and load can be allocated to new instances of controllers in real time based on resources' needs, consumptions, etc.

There are different kinds of flows that travel through SDN networks. From a high level perspective, they can be classified into messages for configuration, feature requests, flow/port/table modification, installation of forwarding entries, statistics, control-plane protocols and packets' punting. Here is a list of those flow messages:

1. OpenFlow messages: Controllers use those messages to define switches' policies. There is a mode called "in-band" wherein a virtual switch; vSwitch can include hidden flows that neither controller nor user can override.
2. Packet in messages: Destiny of new packets entering the network and reaching a switch is decided by the controller. When an OpenFlow switch receives packets, it tries to match them with its flow table rules. If there is no match, packets are forwarded to the controller to make a decision about them. The controller makes decisions on all new packets or packets that do not match any flow rule in the switch flow table(s). New flow rules are then added dynamically to switches based on controller decisions.
3. Datapath flows: Those are vSwitch internal flows and can be used for caching.
4. Controller-to-switch messages: They are initiated or triggered by the controller. Those may include asking a switch about its features or sending packets to a switch for forwarding, and flushing earlier packets.
5. Switch to controller notification messages: Those are usually called event-based messages, such as: packet-in, flow-removed, port status and errors.
6. Symmetric messages: Two-way messages between controller and switches such as hello, echo and vendor messages.
7. Flow statistics: Those are generated by forwarding devices and collected by the controller.

The virtual switch is a program that processes network traffic between the Ethernet Network Interface Card (NIC) and the Virtual Machines (VMs). With virtual switches, virtual machines can act like real hosts in networks. The open vSwitch works with a centralized controller to manage virtual switches as one logical switch. In addition to OpenFlow,

OVSDB is another type of southbound APIs, designed to provide advanced management capabilities for Open vSwitches [Kreutz et al., 2013]. Some controllers support only OpenFlow in the southbound APIs section. As an open architecture, OpenDayLight controller tries to extend this limitation to include a wide range of APIs including, in addition to OpenFlow and OVSDB: NETCONF, PCEP, SNMP, BGP and LISP. Most of those protocols were included to cover backward compatibility with traditional networks.

The forwarding engine in SDN switches has no local control. It is remotely controlled by the central controller. Switches may lose communication with the controllers. Their message requests may time out. In some cases, they may try to communicate with back up controllers, if exist. They can be switched to "fail secure mode" or "fail standalone mode" states. Those are the same states that a switch is in when it just starts a new fresh connection with the controller (i.e. has an empty flow table). An OpenFlow switch can be set to different modes. In reactive mode, no default set of flow entries are specified in OpenFlow for new switch connections. Behaviour of the switch in such cases can depend on whether it is an OpenFlow only or a hybrid (i.e. OpenFlow and legacy). As an alternative, some initial default rules can be added or flushed to the switch where such mode can be a mixed between passive or traditional switching and new or reactive switching modes.

Digital certificates are communicated between switches and the controller. They usually include embedded information from hardware components and are signed by site-specific private keys. This occurs from both ways; the controller and each switch. Digital certificates, widely used in e-commerce, are electronic identity credentials that use encryption methods. The strength of encryption scheme depends on whether the users use strong encryption keys or not. Switches communicate by default with the controller through port 6653. Controllers include, or should include, a certificate manager, to issue and maintain certificates for authenticated users and switches. When switches join the network or the controller, they may have flow records in their flow tables. Controller can have the option to delete all those flows or keep them.

Security testing of communication channels in SDN should include the evaluation of several aspects. Handshaking or communication methods between the different parties (e.g. controller, switches, and top level applications) should be thoroughly evaluated to make sure that attacks such as Man in the Middle (MiM) are not possible. Control and conflict management should be also evaluated to make sure that messages from different parties will not cause any conflict or at least there is a clear conflict resolution scheme. All security applications/controls such as those listed later in this paper should eventually have one common Application Peripheral Interface (API), with all issues we mentioned earlier formalized. This API/protocol should provide interactions between those security, control or management applications and the controller.

Given that SDN is a new architecture that is expected to impact several traditional applications or environments, we will focus on showing how security threats and controls will be influenced by SDN. We will then show how such components are going to be implemented in SDN environment and what are some of the distinguished characteristics in comparison with traditional systems or networks.

3. SDN security threats and countermeasures

In this section, we will use traditional STRIDE threats model [Howard and LeBlanc, 2001] to analyze the type of threats that SDN network can be exposed to. While this model is proposed based on traditional networks, threats described below can be generic and be applicable to networks in general. As an alternative, threats on SDN can be classified based on SDN major functional components described earlier in Fig. 1 and the type and nature of attacks that each component can be subjected to. Attacks on SDN can be also classified based on the type of assets or resources a typical SDN may have. For example, attacks can be focused on switches' flow tables where those flow tables include information related to network management; switching, routing and access control. Attacks can be also focused on the controller as the central location for management and control. The channel between the controller and the switches is another major attacks' target where such channel involves important messages that can be hijacked. At the top level, controller communicates with high level applications using a standard interface (e.g. REST). Such interface can be also attacked in order to trick the controller to allow malicious applications to join the network and interact with the controller, the network and its traffic.

3.1. *Spoofing*

Spoofing refers to a process where network information (e.g. IP, MAC, ARP, etc.) is forged intentionally to hide the actual identity of traffic originator or attacker. For example, users may use spoofed IP addresses to access network resources. Spoofing is often part of a larger attack, such as SYN flooding, Smurf, and DNS amplification [Yao et al., 2011]. Spoofed addresses can also be part of a botnet or a zombie network to launch Distributed DoS; (DDoS) attacks. Currently spoofing threats in SDN primarily include Address Resolution Protocol (ARP) spoofing and IP spoofing.

3.1.1. *ARP spoofing*

ARP spoofing involves linking an attacker MAC address to a legitimate IP address. The original purpose of ARP is to resolve IP to MAC addresses. The ARP spoofing attack may cause traffic to be hijacked from the original intended receiver and, as a result, a legitimate user or host is knocked out of the network. IP to MAC mapping tables can be used to detect ARP spoofing.

[Matias et al., 2012] proposed an Address Resolution Mapping (ARM) module in the controller that tracks MAC addresses from authorized users or hosts. Controller then consults this ARP module and discards ARP responses that are not verified by the ARP module. In OpenFlow, ARP poisoning is possible between the controller and switches if the optional SSL encryption is not used. ARP cache poisoning occurs when an attacker is located in the same subnet of the victim

network (e.g. internal attacks). Attacker can use scanners to listen to network traffic between network components. [Al-Shabibi, 2014] has developed an anti ARP poisoning switch application in the POX OpenFlow controller.

ARP spoofing attacks can be countered with packet level information. [Zaalouk et al., 2014] divided attacks' detection methods into high and low resolution methods based on the amount of information given to it as input. Low resolution attacks require information at the flow, not the packet level. Packet level details are only required at high resolution level attacks. For example, attacks like DoS and Domain Name Server (DNS) amplification can be handled with information at the flow level details. On the other hand, ARP spoofing and cache poisoning attacks require information at the packet level.

3.1.2. *IP spoofing*

IP spoofing is usually used as an opening to other types of security attacks, such as DNS tampering or amplification. A DNS is a directory that associates IP addresses to domain names. To reroute traffic to illegitimate websites, an attacker may manipulate DNS directory. This can be also part of a large flooding or worm spreading attacks. What all spoofing methods have in common is that they try to redirect traffic to illegitimate hosts. They can also be considered to achieve Man in the Middle (MiM) attacks. Spoofing can be mitigated by a proper authentication scheme. Strong password and encryption methods should be enforced to avoid unauthenticated intrusion.

IP address validation methods can be also used to counter IP spoofing. Internet Engineering Task Force (IETF) formalized a standard for Source Address Validation Improvement (SAVI). SAVI verifies addresses of packets based on a binding validation. Based on OpenFlow Virtual source Address Validation Edge (VAVE), [Yao et al., 2011] extended SAVI to solve the address resolution problem. VAVE module, embedded in the controller, verifies the address of external packets that have no records in the flow table. Flow entries are inspected based on the validation module and based on a dataset of white lists to judge whether to allow or drop the flows. An explicit rule to drop the flow is added in the flow table once spoofing is detected. [Feng et al., 2012] extended the solution of VAVE using OpenRouter; an OpenFlow extension of traditional routers. Each router has a collective network view of address assignment and routing. Software Defined Filtering Architecture (SEFA); another countermeasure of the same IP based or router based spoofing, is a successor to VAVE [Yao et al., 2014]. In addition to collecting and building flow rules in routers, SEFA adds filtering rules based on spoofing occurrences.

Existing research discussed hiding hosts' identities to protect them against several types of attacks including spoofing [Jafarian et al., 2012; Kampanakis et al., 2014, and Yao et al., 2011]. This is a form of dynamic network configuration where network information is either frequently changing or is hidden from externals. Network information not only includes IP or MAC addresses, but also topology and routing tables. In an attempt to hide end users' identities and protect them from scanners and spoofing, [Jafarian et al., 2012] proposed a moving target defence approach. End hosts and their identities should be continuously and randomly changed to avoid being targeted by adversaries. OpenFlow assigns virtual IP addresses to end hosts that can be mapped to actual or physical IP addresses.

In SDN, controller should have a method to isolate its local network information from the external networks. Similar to Network Address Translation (NAT), controller can have tables to transform external to internal addressing. In fact, OpenFlow can do this natively as OpenFlow devices can rewrite the packets' header fields which will make them appear as coming from external addresses. This NAT translation should be communicated with many middleboxes while at the same time be hidden from externals [Fayazbakhsh et al., 2013]. OpenFlow networks and other network virtualization methods allow users to divide the network into slices and make flows behave differently in the different slices regardless if they will have the same real IP addresses or not. For example, it may be required for business purposes to direct some particular flows to a security control (e.g. a firewall). Alternatively it may be required to allocate more bandwidth and resources to some particular traffic.

Spoofing or forged IP addresses may occur from within the network. Based on the nature of IP address forging, source address validation from within the network can be difficult to detect. [Xiao et al., 2013] extended an earlier research about OpenRouter. In general, an SDN approach may not need a dedicated router where routing functionalities are included in OpenFlow controller and switches.

3.2. *Tampering*

Tampering is the deliberate and unauthorized modification or destruction of network information, such as topology, flows in flow tables, policies, and access lists. For example, an intruder may try to inject flow rules that will cause network misbehaviour. They may inject flow table or firewall rules that will deny legitimate hosts or allow illegitimate hosts. Intruders may also try to tamper topology information and consequently cause some traffic to be hijacked. In SDN controller distribution, different controllers communicate with important information. It is very important to secure this communication channel from being hijacked or tampered [Othman and Okamura, 2013].

Security threats may target firewall or flow table rules. [Porras et al., 2012a,b] described the security problem of dynamic flow tunnelling related to conflicts in interpreting flow rules. This problem occurs since rules are evaluated one by one. An attacker may try to orchestrate more than one rule where all those flows collectively violate firewall rule(s) while on the other hand, no single flow violates any firewall rule. In their proposed solution, they tried to check the conflict between flows and firewall rules based on all possible combinations of incoming flows. Nonetheless, this may not be scalable or applicable to complex scenarios.

As a flow-based traffic management, SDN can help preventing unintended traffic tampering. Packets can be inspected before they go on the wire for their destination for some integrity attributes. Validation results can be carried out with the traffic to be checked at the destination point. Tampering can be mitigated by distributing auditing and monitoring

across several network points [Bellessa et al., 2011]. If one point is attacked, the rest of the network points can be used to detect and correct such tampering.

To protect against tampering, controller should manage and routinely check encryption methods and legitimate connections. The main limitations of Transport Layer Security (TLS) encryption used in OpenFlow are that first it is optional to use or enforce by users and second is that many actual controllers are not even implementing or adopting it. Another related issue is controller failure modes (fail safe and fail secure modes). It is possible to compromise integrity or confidentiality when controller or switches are pushed to switch to one of those failure modes.

In virtual environments, different logical networks share the same physical or network resources. As a result, there is a serious concern about the level of correctness and integrity not only from external editing or tampering but also from internal modifications. Existing experiments showed that slices or VMs in the same tenant or cloud datacentre have a possibility that one VM may access resources from the other VMs sharing the same physical resources [Ristenpart et al., 2009; Zhang et al., 2012]. Same concern can be also mentioned in other scenarios where virtual separated resources share same physical resources (e.g. different testbed experiments, different wireless or home network users, etc.). Can security tools verify beyond doubt that logical separation or isolation guarantees no interactions between the different virtual networks?! How could this be guaranteed and what kinds of tests to be conducted to proof or certify that? It is expected in future that such certificates will be possibly required or demanded from cloud or ISP service providers.

3.3. *Repudiation*

"Repudiation is the denial by one of the entities involved in a communication of having participated in all or part of the communication" (ISO, 1989). Non-repudiation, which is considered as a legal rather than technical concept, tries to make sure that such denial does not occur. The receiver needs to verify that packets are sent from the actual sender included in the packet header and the sender needs to verify that packets sent to the actual receiver included in the packet header. Non-repudiation is often related to accountability, which is about holding individuals or entities accountable or liable for their actions.

3.3.1. *Non-repudiation verification*

In current web, e-commerce, etc. indirect or remote types of communication, this repudiation or denial from one party that they were part of this communication can be caused typically as a result of Man in the Middle (MiM) attacks in which an intruder in the middle masquerades to both parties that he/she is the other party. As such, encryption can be an effective MiM counter measure and hence repudiation. Based on this assumption, we describe encryption based methods in this section for non-repudiation verification.

Encryption methods are used to verify to communication partners that messages were authenticated from originating sources and were not tampered throughout the network. Research has showed security problems with Secure Socket Layer/Transport Layer Security (SSL/TLS) encryption that is used in OpenFlow algorithm for the communication between controller and switches. [Namal et al., 2013] proposed alternative encryption schemes, HIP-BEXv1 and HIP-EEX that offer better security features for non-repudiation, DoS, and MiM threats.

Third party verifications (e.g. Public Key Encryption PKE and digital certificates) can be used to eliminate repudiation. Currently, such security mechanism and architecture (i.e. PKE) is used widely in e-commerce systems and business transactions. Message transfer digests are also used to provide digital receipts regarding a message. Signal chaining can be also used to provide non-repudiation. For example, an audit system should include the step routes or sequence that a message or a packet went through between original sender and final receiver.

Proper auditing and logging methods for all types of activities that occur in flow tables can help in non-repudiation. Those can be provided as proofs about traffic activities. However, trade-off between performance and logging should be put in place to properly select what exactly to audit. Even logging and auditing methods themselves can be tampered by some security attacks. [Porras et al., 2012a,b] proposed FortNox, a flow-based authentication system, to provide a security audit trail for flow rule commands, rules' conflicts, and resolution outcomes. It is not clear; however, what Meta information is included in the audit or how conflicts can be handled. In addition to flow attributes described in OpenFlow specifications, for auditing we may need to know other information such as application ID, privilege level, flow time and date. If a security breach occurs, this information is useful for incident investigation.

[Andersen et al., 2008] proposed Accountable Internet Protocol (AIP) as a replacement to the Internet Protocol IP. The goal was to add more information in addition to those typically exist in packet headers that can uniquely identify the sender application, user, machine, etc. [Bifulco and Karame, 2014] proposed a location based identification of hosts in addition to the IP address that is associated with the public key encryption of the user or the host.

Non-repudiation verification in very agile and dynamic networks can be difficult to achieve. There are many SDN use cases (e.g. Bring You Own Device (BYOD), campus networks, peer to peer networks) that require non-repudiation-related qualities where current networks can hardly provide [Feamster et al., 2013a,b; Bakshi, 2013]. In these networks, users and their network preferences vary all the time. Handling non-repudiation is nontrivial given the large number of users and the agility of the network. SDN programmability and its ability to define users or hosts based on flows can be important tools to achieve such quality attributes more robustly.

3.3.2. *Accountability*

Current SDN architecture holds each controller accountable for its own switches. Inter-domain communication exchange between the different controllers is not supported [Huang et al., 2013; Huang et al., 2014]. Packets related to switches in other controllers' networks are going to be dropped by the local controller. However, there are many use cases that

justify the need for different controllers to exchange information especially as a single controller network is not a practical network design for most production networks. Different controllers should exchange information through well defined-interfaces so that they will not interfere with each other or cause security problems.

[Karame, 2013] discussed accountability issues related to QoS. Communication partners need to exchange information related to: Response time, error rate, etc. This is one of the current serious challenges in cloud computing related to Service Level Agreement (SLA). Network attacks can have a direct impact on network metrics where they may delay response time or cause some traffic not to be sent on time or correctly. Who should be liable in such cases?! How could issues related to conflicts in SLA and accountability be solved?! Karame, 2013 proposed a security approach based on OpenFlow that can handle some of the concerns in this specific aspect.

Accountability can be challenged by several network components including security controls. Security controls in most cases act as barriers that limit the ability to audit traffic resources. For example, NAT or proxy systems hide the identity of internal hosts where a firewall may not be able to know traffic source or actual host IP address. This is because there is an internal mapping in the NAT/proxy between internal to external IP addresses. In this scope, [Fayazbakhsh et al., 2013] proposed FlowTags as a system to allow security middle-boxes to identify applications. Tagging information should be integrated with flow information. Different applications which generate flows are expected to add this flow tag information based on a uniform standard (i.e. Through a FlowTag controller module). FlowTag module should handle rewriting packets' headers to include FlowTag information from originating middle-box.

3.4. *Information disclosure*

Information disclosure attacks have no direct intention to destroy or disrupt the network but to spy on its information. In addition to the sensitive information that attackers try to get, they will initially try to sniff network information such as topology, nodes' features, or communication details among nodes. The impacts of SDN architecture on scanning attacks can be mixed. The controller is a central location for control of all network switches. Being able to invade the controller, the attacker can have a tremendous network access. On the other hand, data is isolated from the controller, unlike traditional switches where control is co-located with the data inside the switch. In SDN, flow rules exist in switches' flow tables. If intruders succeed to access those switches directly, they can tamper flow rules and cause traffic to go to wrong destinations. If they succeed to disconnect a switch from communicating with the controller, they can assume control and cause a significant traffic miss-direction. If they could hijack traffic from a legitimate host, they can impersonate that host and join the network as a spy. Man in the Middle (MiM) attack is an information disclosure attack that targets information in transit and not in premises. MiM attacks are currently seen to be significantly possible in the current OpenFlow architecture [Benton et al., 2013]. Current encrypted scheme in OpenFlow communication, TLS, is optional. In addition, communication in the northbound interface with the controller is not yet standardized. Added applications with possible vulnerabilities can be used to launch MiM attacks and access controller resources.

3.4.1. *Scanning countermeasures*

Scanning methods and tools are often used in the initial steps of information disclosure attacks. Network scanners search through the network for potential information leakage and vulnerability.

Encryption methods can be used to counter scanning based attacks. The fact that the switches in SDN are remotely controlled can be a security threat by itself. As mentioned before, TSL/SSL encryption between the controller and its switches is left optional as of the last visited version of OpenFlow (i.e., 1.4). It is not clear whether OpenFlow switches only ensure one-to-one control relation (i.e. between each switch and its controller) and how it is enforced. In other words, can an intruder succeed in having a secondary control rule on the switch without disconnecting the actual controller? Does the switch or the controller guarantee that there is only one controller connection coming to or going from the switch?

Active security methods can be used to detect scanners (i.e. if external tools are used to scan the target network). [Mehdi et al., 2011] described using OpenFlow flow information for traffic anomaly detection including the detection of scanning worms. [Schehlmann and Baier, 2013] extended the approach and made it more scalable at ISP level networks. They used NetFlow to filter initial suspicious traffic and then redirect it for further analysis to OpenFlow based detection system.

Several methods have been proposed to prevent OpenFlow network scanning. Some of them used an active approach to counter sniffer/scanner attackers [Hand et al., 2013]. Some are based on responding to the scanner with incorrect or fake traffic or fighting back by flooding the attacker with large traffic. Other methods continuously change the identity of hosts [Jafarian et al., 2012; Kampanakis et al., 2014].

Some methods for hiding local identities from external users can also be used to counter sniffers or scanners. They include Virtual Private Networks (VPNs), Network Address Translation (NAT), and proxy, although their original purposes are not related to hiding hosts identities. For example, [Mendonca et al., 2012] introduced AnonyFlow, an OpenFlow-based anonymization service. Unlike traditional NAT where translation occurs between virtual and real IP addresses, AnonyFlow uses special anonymity IDs that other parties can only see instead of the IP addresses.

3.4.2. *Information disclosure countermeasures*

To protect private information, there are other actions to consider. White listing and black listing can be used to filter traffic. White and black listings in traditional networks are defined based on IP and MAC addresses. They can be also used in OpenFlow networks. As OpenFlow switches can interact with flow level information, we can define metrics based on flows and then define black and white listings based on flow level information. This may prevent some attacks that use for example large traffic where some attributes can't be identified based on IP or MAC addresses.

[Kloeti et al., 2013] proposed several recommendations to reduce information disclosure in OpenFlow networks. For example, intelligent rules for time out randomization can make it difficult for scanners or sniffers to understand network patterns. Similarly this can be applied to the response time between controller and switches. A monitoring tool can detect the difference in response time between sending a new and an existing flow rule. The existence of such difference in response time is an indicator of an OpenFlow network. Countering this type of information sniffing can take several scenarios. In one option, this can be countered using a fully proactive approach where all flow rules are installed by network administrators. Directed and intelligent counter measure methods can be also effective in making fake response time based on the nature of the network attack. Attack tree models proposed by [Kloeti et al., 2013] and several other researchers can be used to automatically detect the type of network attack. However, such models still seem to be highly semantic and do not include metrics that can be directly interpreted or related to flow or packet level data. Some useful flow or packet metrics are APf (Average Number of Packets in Per Flow (ANPPF), Average of Bytes per flow (ABf), Average of Duration per flow (ADf), Percentage of Pair-flow (PPf) and Growth of Single flow (GSf) [Feng et al., 2009].

3.5. *DoS*

DoS attacks are among the most serious threats because they affect network performance, increase latency, and drop of legitimate packets. They may even disable the whole network or stop it from functioning. For OpenFlow networks, DoS can be more devastating as there is a continuous flow between controller and switches. The continuous communication between controller and switches can tempt attackers to push flows between the controller and the switches and interrupt the normal network activities. Flooding and DNS amplification are considered as flow level resolution attacks because flow-level information is enough for detecting such attacks [Zaalouk et al., 2014]. Flow level information is usually enough to detect most types of DoS attacks. Typically traffic information collected from flow header is at the flow level. Flow based attack detection systems that rely only on the header information can address the following network threats: DoS, scans, worms and botnets. Those four types of attacks have some common signatures. For example, they have a large unbalanced traffic between fan in and fan out where most traffic is going in one direction. In most cases large traffic will be coming in. However, if the local machine is a botnet or a victim, it can be sending a large volume of traffic. Port number can be also a valuable information in those types of attacks where there are known ports to be widely used. Other types of network attacks may require packet level information in order to be detected. As flow-based networks, SDN provides native methods for DoS detection [Sperotto et al., 2010]. Information extracted from flow headers is valuable for DoS detection. Some DoS attacks cause variation in traffic volume that is visible from flow view. Semantic-based DoS attacks, however, may not be detected by traffic volume change.

The main distinguished feature of DoS attacks is the large traffic size. Methods to detect the large size of traffic are the most popular techniques used to detect flooding or DoS. However, false positive alarms may arise where such large traffic is coming from or going to legitimate hosts. There are other methods to detect possible flooding. One method is related to studying the difference in volume between incoming and outgoing traffic. Typically in communication between a source and a destination, there will be traffic going from the two sides. If traffic is large and going from one side with no single response from the other side, this can be an indication of a flooding case. In TCP transmission for example, even if data is from one side to the other, receiver will send ACK messages periodically. Even UDP transmission will have a request/response from the application layer.

In DNS amplification, public DNS servers can be used to increase the effect of DDoS. This can cause a very large scale network or Internet disruption. Recent reports (2013) showed one of the largest DoS attacks in history on the website (www.Spamhaus.Org) that is launched based on DNS amplification. Monitoring and continuously retrieving the top DNS queries can help us detect DNS amplification. For example a query that asks the name server for all the records in that domain results in a large response that causes traffic amplification. Controller (limit traffic) decision choice can be designed in a way that limits such cases.

Loops can cause DoS or can be used for network attacks. In such loops, packets travel from one switch to another without reaching their final destination. [Kazemian et al., 2013; Kordalewski and Robere, 2012] have discussed how to handle loops in OpenFlow networks.

To conduct DoS attacks in SDN, attacker may push a large volume of traffic that keeps randomly changing flow attributes [Shin and Gu, 2013; Shin et al., 2013a,b]. This is to ensure that every flow is new, from the switch perspective, and hence will be sent to the controller for making a decision about. An attacker can use a traffic generator that ensures to change attribute values per flow. As each attribute has a wide range of valid and invalid inputs (e.g. IP address: 0.0.0.0 to 255.255.255.255), the number of possible flows can be enormous. Such attack can have two goals: First, it will flood the switch flow table and saturate it with illegitimate rules. This may disable the flow table ability to accept legitimate rules. The second goal for attackers to send this large amount of flows is that this flood of flows will keep the controller busy from responding to legitimate flows from other switches and may bring it to a failure. Strong and reliable encryption methods can help in securing the private communication between switches and the controller. However, they cannot prevent flooding or DoS as those are launched from hosts sending traffic to OpenFlow networks. The Avant-Guard system proposed in [Shin et al., 2013a,b] as an enhancement to OpenFlow, showed that it is possible to handle DoS attacks and eliminate their negative impact on the network.

3.5.1. *Detection of DoS*

[Braga et al., 2010] discussed a lightweight method to detect DDoS attacks in SDN. The main challenge was to distinguish normal packets from DDoS flooding packets. They classified network traffic into an attack or normal traffic based on Self Organizing Maps (SOM). The flow features selected were based on earlier approaches [Feng et al., 2009], including APf

(Average Number of Packets in Per Flow (ANPPF), Average of Bytes per flow (ABf), Average of Duration per flow (ADf), Percentage of Pair-flows (PPf) and Growth of Single-flows (GSf). Those metrics or attributes are continuously collected and monitored for detection of possible DDoS. A major concern is that monitoring and maintaining such huge amount of data will significantly degrade controller performance which is already overwhelmed with other tasks. Having a dedicated separate module or controller to perform such task can be a more realistic solution. [Shirali-Shahreza and Ganjali, 2013a,b] proposed to sample traffic to reduce controller traffic overhead from the monitoring process.

A simple method to detect possible DoS attacks is to keep monitoring the volume of traffic flows. Threshold can be specified on what can be considered large or abnormal traffic. Once this threshold is exceeded, a DoS occurrence can be triggered [Chu et al., 2010] and controller inserts a flow rule to drop packets. Similarly, traffic map or patterns can be analyzed frequently to predict if some traffic is abnormal or large [Braga et al., 2010]. [Suh et al., 2010] proposed a content based networking architecture. Controller triggers DoS alert if traffic exceeds a certain threshold. Rules are then inserted in switches by the controller to eliminate source of DoS.

[Schehlmann and Baier, 2013] proposed an OpenFlow based approach to detect and mitigate botnets. Botnets are networks or groups of compromised hosts that are used to launch attacks such as DDoS, to propagate worms or send spams. Their proposed solution, COFFEE, utilizes SDN ability to have access to all traffic to reduce rate of false detections.

In TCP connections, acknowledgement message (TCP ACK) is required to verify communication between senders and receivers. However, it can also be triggered by a flooding or DoS attack. [Shin et al., 2013a,b] proposed a simple algorithm to handle TCP ACK packets. [Liyanage et al., 2014] proposed a security layer or interface to coordinate the communication between OpenFlow switches and the controller. A show case of TCP SYN DoS attack is used to evaluate the model. Attack includes occupying all packets and IP address possible combinations. Network performance is measured through the attack to evaluate the time it takes the network to figure and clear out the attack.

[Benton et al., 2013] evaluated OpenFlow vulnerabilities for DoS and integrity attacks. They showed that OpenFlow protocol and its communication mechanism between controller and switches should be thoroughly investigated. [Dover, 2013] conducted an experiment to simulate DoS attacks on Floodlight controller using methods such as TCP SYN or ARP cache poisoning. A vulnerability discovered in Floodlight that disconnects an old switch if a new switch is registered with the same data path ID (DPID) as of the old one. Such vulnerability can be used by malicious switches to claim to be legitimate. The only information attackers need is the DPID which can be acquired from the controller REST API.

Yuzawa, 2013 presented a simple use case for using sFlow monitoring tool for DDoS attacks' detection in OpenFlow. The goal was to counter DDoS without disrupting normal traffic. They used the module "static flow pusher" from Floodlight controller and claimed that no commercial virtual switch showed the same expected response as the open source virtual switch (vSwitch).

[YuHunag et al., 2010] proposed an autonomic DDoS detection system based on OpenFlow. The system uses the simple volume count (i.e. flows/packets per time) to judge the occurrence of DoS or DDoS. The problem with such simple metric is that many false positive alarms may occur where large volume traffic can be legitimate.

While some studies argued that OpenFlow networks have more problems with DoS than traditional networks, Yuzawa (2013), Dillon and Berkelaar (2014) showed that SDN can produce a better way of handling Remote Triggered Black Hole (RTBH). This is a technique in traditional WAN networks to countermeasure DoS attacks by instructing routers to drop all traffic to the target. They used OpenFlow traffic flow statistics to monitor traffic volume and alert for a significant increase in size attributes (e.g. byte and packet counters). They used the mathematical standard deviation measure to evaluate whether certain flows are significantly above average. Packet symmetry is also used as an indicator of DoS in that the difference between incoming and outgoing flows for a particular host is very high.

3.5.2. *Countermeasure of DoS*

DoS can be handled by effective and dynamic response methods to handle occurrences of DoS. Rate or limit traffic by the controller and monitor abnormal traffic behaviours are also important countermeasures. We discussed in an earlier section some countermeasures for spoofing. Similarly, there are some proposals for active countermeasures of DoS or flooding attacks in SDN networks specifically [Koponen et al., 2011]. Active response means to take an offensive action to counter an attack.

An attacker can focus DoS on the messages from data plane or switches to the controller and try to saturate both switch flow table and controller resources; data-to-control saturation attacks [Wang et al., 2014]. Protection mechanisms should ensure that controller and switches have the ability to quickly recover from such flooding. The mechanism should also be able to distinguish legitimate from fake traffic. Passive or dormant monitoring agents are proposed to be triggered only when they see the occurrence of fake flooding.

[Koponen et al., 2011] proposed FII (Framework for Internet Innovation) to deal with inter domain DoS, based on the IP addressing scheme AIP [Andersen et al., 2008]. AIP includes information about hosts in packet header related to the host with a global ID, rather than an IP address. This may help eliminating attacks that hijack hosts based on their IP addresses. They claimed that their approach is focusing on availability to ensure that each participant in a communication can reach the destination. The approach divides handling DoS into two parts: Inter and intra-domain attacks. For local, or intra-domain attacks, each domain should be given the choice to select their own way of validating local hosts. On the other hand, FII provides a united method to handle DoS inter-domains attacks. As a countermeasure, a shut up message (SUM) can be issued to intruders attacking the network with DoS attacks.

Flooding or DoS can also be solved by flow rules optimization or rule-merging in flow tables. Flow tables can be dynamically flooded with rules to cause buffer overflows or to saturate switch memory and cause it to be closed down or

deny service for legitimate hosts or traffic. Hence, it is necessary for switches to have a dynamic ability to continuously re-evaluate flow table rules and merge flows that can be merged. However, such evaluation and decision process itself is intelligent and complex. Based on the current OpenFlow architecture, such intelligence does not exist in switches. Further, if the controller will do this, it can add an extra overhead on controller resources.

3.6. *Elevation of privilege*

Once entering the system, attacker tries to elevate their access privilege to access system resources and applications that require special permissions. The ability to detect privilege elevation attacks requires a robust and intelligent auditing process. For example, [Ramachandran et al., 2009] proposed Pedigree, a system to trace executed applications through tagging them with special identification. A major problem with logging or auditing methods is scalability because they store a large amount of data which may affect storage, memory, and bandwidth. Since privileges are allocated in authorization or access control modules, escalation attacks often target those modules and try to tamper information in those modules. Several approaches have been proposed to give users the right level of permission [Clark et al., 2009; Naous et al., 2009; Foster et al., 2011; Porras et al., 2012a,b; Katta et al., 2012; Wen et al., 2013].

[Porras et al., 2012a,b] proposed a fine grained RBAC system based on OpenFlow. The idea is to give privilege on a flow-basis rather than on a user or host basis. One advantage of this flow-based authentication is that a user needs to be verified frequently on a flow by flow basis. In other words, a user is not always guaranteed or denied. This may reduce the problem of privilege escalation as users are frequently screened for possible privilege escalation. Another advantage is that controller can be isolated from all other flows. In addition, internal flows can be distinguished from external flows. Permissions can then have a lifecycle that starts and ends with the flow life cycle. A source authentication module is included to allow each flow rule insertion in switches to be verified through a digital signature. If no signature is provided lowest priority is given. However, this may open the opportunity for privilege escalation later on (i.e. within the switch flow table). Using default or least privilege approach has an advantage of not dropping flows if authentication failed. However, it does not solve the security attacks coming from privilege escalation. In addition, many current attacks start their intrusion by attacking a legitimate application and compromising it. The victim application privileges are then used for further attacks. Perhaps a hybrid approach is necessary to combine between such privilege or permission system in addition to another module that can track applications' "usage profile". A legitimate application that is suddenly changing the way it communicates with other applications or destinations should trigger a security alert.

[Wen et al., 2013] proposed PermOF, a fine grained access control management system that includes comprehensive access levels for controller and network resources. Simple limited authorization levels with only two or three authorization levels (including the administrator) can be an easy target to tamper with or cause privilege escalation. Including a relatively large number of access levels should result in limiting the use of high level administration capabilities that can have very powerful access and modification privileges. The proposed approach provides a set of 18 possible permission levels. A default minimum privilege is given to applications. Controller API calls trigger communication with the applications. One challenge for such approach is whether different operating systems can generate the same process IDs (which they don't) or else we need to tag process ID per operating system or in a separate special tagging system.

3.7. *SDN attacks vs classical attacks*

In this closing section of attacks, we will focus on how attacks are going to be different in SDN in comparison with attacks on classical networks. As a new architecture, SDN can expose both new security opportunities and challenges. Attackers will eventually investigate SDN strengths and weaknesses and will try to maximize exploits based on vulnerabilities. For example, zero-day attacks refer to attacks committed based on newly discovered vulnerabilities. A significant amount of such attacks are expected to be exposed in the coming years impacting SDN. Examples of some of those zero-day attacks are discussed in some research papers (e.g. [Kloeti et al., 2013]).

SDN can be categorized as a dynamic network where a significant amount of traffic is exchanged between controller and its switches. DoS attacks can be a significant threat to SDN in comparison with classical networks that do not have a central controller frequently exchanging data and control with switches. DoS attacks are expected to be larger in numbers in SDN. However, if SDN is implemented or designed correctly, their security controls should be dynamic and autonomous in a sense that they will eventually discover and eliminate DoS attacks. On the other hand, spoofing may have less chances of occurrence in SDN when compared with classical networks. This is since all spoofing techniques depend on tricking a network service (e.g. DNS, ARP, etc.) based on obsolete information. Updates in SDN are dynamic and frequent and hence changes in the network such as the inclusion/exclusion of hosts, IP addresses, MAC addresses, etc. should be quickly discovered and accommodated.

SDN controller and its channel of communication with its switches (i.e. OpenFlow) will be the most vulnerable points that are expected to be attacks' targets. SDN depends on splitting data from control in switches and allocate switches control remotely and centrally in a software controller. SDN has several positive goals for such direction. However, this direction has also its payoffs. From a security perspective, MiM attacks can possibly occur between controller and its switches where a switch can be compromised or controlled by an intruder. SDN architecture tries to approach this problem by dedicating a special connection between the controller and switches in a separate physical and logic subnet from the rest of switch ports. However, that does not eliminate completely the possibility of compromising the communication between the controller and its switches.

One of the ambitious goals of SDN is to be able to design dynamic and programmable security controls that can fully

operate with least or no human interaction. Those security controls can respond in real time to network changes and security threats and respond accordingly. However, once those controls exist we may see new types of attacks that were unconventional in classical networks. For example, it may become possible to create ghost or fake network nodes that are intelligently crafted by expert hackers based on their knowledge of software controllers, their APIs, middle-boxes, etc.

There is also a serious security threat from high level or northbound middle-boxes or applications that can be developed to communicate with the controller. Those user defined and controlled applications can interact with the controller to provide commands or pull information from the underlying network. Those applications exist in typical users environments, operating systems, Internet connected hosts. Such environment has a significant amount of threats where it is possible to attack and compromise a middle-box application that is interacting with the controller. The special relation and privileges given to such application from the controller can be a significant power exposed by attackers. In principle while one of the major goals of SDN was to enable users to interact with and control underlying network, however, a possible payoff is that such privilege can be abused intentionally by attackers or unintentionally by network users.

Table 1 below summarizes contributions in SDN-based attacks or threats. In comparison with classical networks, while the general types of attacks are not expected to change significantly, however, opportunities for some types of attacks are expected to increase.

Table 1 – SDN attacks research progress.

Threat	Detection methods and challenges	Counter measures
ARP Spoofing	ARM [Matias et al., 2012]	Packet level information [Zaalouk et al., 2014], Anti ARP switch application [Al-Shabibi, 2014]
IP Spoofing	SAVI [Yao et al., 2011]	OpenRouter [Feng et al., 2012], SEFA [Yao et al., 2014], Moving target defense [Jafarian et al., 2012]
Tampering	Problems with TLS and encryption methods [Namal et al., 2013], [Meyer and Schwenk, 2013], Tampering in virtual environments [Ristenpart et al., 2009; Zhang et al., 2012]	Distributed auditing and monitoring [Bellessa et al., 2011]. Improve
Repudiation	TLS encryption problems and alternatives [Namal et al., 2013], [Meyer and Schwenk, 2013], BYOD [Feamster et al., 2013a,b; Bakshi, 2013], SDN distributed controllers [Huang et al., 2013; Huang et al., 2014]	Flow-based authentication methods [Porras et al., 2012a,b], AIP: [Andersen et al., 2008], Location based identification [Bifulco and Karame, 2014], FlowTags [Fayazbakhsh et al., 2013]
Information disclosure	OpenFlow security challenges; MiM [Benton et al., 2013]	Flow based anomaly detection [Mehdi et al., 2011] [Schehlmann and Baier, 2013], Hiding identity methods: [Jafarian et al., 2012, Kampanakis et al., 2014, [Mendonca et al., 2012], Intelligent crated flow rules [Kloeti et al., 2013]
DoS	SDN can be more attractive to DoS [Shin and Gu, 2013; Benton et al., 2013, Shin et al., 2013a,b]. DoS detection based on Flow level information [Braga et al., 2010, Sperotto et al., 2010, Chu et al., 2010, Suh et al., 2010, Yuzawa, 2013, Zaalouk et al., 2014], lightweight DDoS detection, reduce SDN traffic [Shirali-Shahreza and Ganjali, 2013a,b]	Flow based DoS mitigation [Schehlmann and Baier, 2013], DoS active countermeasures [Koponen et al., 2011], FIT inter-domain DoS counter Koponen et al., 2011]
Privilege Escalation	Rule based access control [Porras et al., 2012a,b], PermOF; fine grained access control management system [Wen et al., 2013]	

4. SDN security controls

Security controls aim at providing access to legitimate users, protecting systems from attacks, and providing mitigation and countermeasures when attacks occur. Complexity and exact duties of each control can vary from one domain to another. Control main tasks can generally include detection, logging, protection and counter measures.

4.1. *SDN firewalls*

Firewall is one of the most popular security mechanisms. Firewalls are responsible for monitoring network traffic to allow or prevent their passage or intrusion based on certain criteria specified by users or network administrators. Typically, they work in layers 2–3 (i.e., data-link and network layers) of the OSI 7-layers model. Firewall rules can be defined to prevent or permit traffic based on IP addresses, ports, protocols, and MAC addresses. While traditional firewalls have been well-studied, the research on SDN firewalls is still evolving. An SDN controller itself performs some of the tasks that are typically accomplished by traditional firewalls. For example, controllers in SDN make decisions related to the fate of flows and write relevant flow rules in switches' flow tables.

4.1.1. *SDN firewalls vs traditional firewalls*

In terms of attributes used in firewall rules, the current implementations of SDN firewalls are similar to traditional

firewalls. On the other hand, recent versions of OpenFlow have expanded the list of attributes that can be included in flow rules. This will eventually impact future implementations of SDN firewalls.

The major impact that SDN has on firewalls is that the SDN controller make decisions on flows fate. In traditional networks, this was the main role of the firewall. In SDN, controller acts as a firewall (coarse grain firewall). Controllers continuously evaluate or know current topology by using a link discovery module. Controller generates LLDP and broadcasts packets routinely to neighboring switches. Based on response from those switches, controller can frequently predict current network topology. Controller also includes a learning switch module that learns about new devices based on their MAC addresses. Rules can be added dynamically by the controller to the switches' flow tables. If a new flow is added to the network, the learning switch checks input and output switches of the flow and also the best route for the flow. This is then added as a new rule to the proper switch.

In SDN, controllers store rules or Access Control Lists (ACL) for all network switches. Such connection (i.e. between firewall and switches) does not exist in traditional networks. As a result, firewall rules in traditional networks are static and are not connected to network traffic. Those rules are added and evaluated manually by network administrators. It is hence possible that some rules in traditional firewalls are obsolete or inapplicable. On the other hand, flow table rules in SDN are very dynamic. Obsolete rules are eventually removed from flow tables.

As the recent versions of OpenFlow have extended flow attributes, SDN based firewalls can be more specific and deal with flow or packet level attributes. OpenFlow 1.0 includes 12 header fields. In addition to those fields, there are new fields related to IP protocol, VLAN, etc. OpenFlow 1.2 and above includes 40 header fields, giving users more ability to control or interact with network flows. Having control at the flow level enables network administrators to perform tasks that were not possible using traditional networks. In some cases, they want to perform traffic redirection through middle-boxes. This problem is quite common in the cloud environment where the automatic configuration of a new instance of a VM or tenant will not be completed as network administrators have no control on middle-boxes (e.g., a firewall) to instruct those middle-boxes to allocate resources to the new VM or tenant [Sherry et al., 2012; Gibb et al., 2012; Gember et al., 2013; Mysore et al., 2013].

4.1.2. *SDN-based firewalls*

In SDN, a firewall module can be added typically as a northbound (REST) API to the controller. REST API is a standard add-on environment for interacting with most SDN controllers. It allows user-developed applications to communicate with the controller. Firewall rules are different from flow table rules although they may look similar.

Several papers have discussed how to implement SDN-based firewall modules. [Casado et al., 2006] proposed SANE, as a protection architecture for enterprise networks through defining a single protection layer. This is one of the early contributions to centralized control in the network operating systems or the SDN controller. Switches and other network components have simple and minimally trusted forwarding elements. In this early SDN architecture, controller includes access control rules instead of having them in firewalls in traditional networks. One of the explicit stated goals related to centrality is to unite all security effort and information in one place. This, however, may have different interpretations. The centrality of rules' decisions in the controller should not be mixed with combination of functions as different security controls are not cohesively performing the same tasks. The idea of a central controller offers another advantage because a firewall module interacting with the controller can have a global view of the whole network.

[Hu et al., 2014a,b] proposed FlowGuard; an SDN based monitoring framework for detecting possible conflicts between firewall rules and flows. Whenever network state changes occur, FlowGuard checks path spaces to see if a firewall policy is violated. In this study, several challenges and opportunities of SDN-based firewalls are discussed, such as the ability to dynamically evaluate policy changes, conflict issues in flow table rules, the centrality of the controller, and the firewall ability to perform stateful traffic inspection.

[Jia and Wang, 2013] proposed SDN based firewalls for P2P networks. The firewall module is provided as an API to be integrated with SDN. P2P networks' use case may benefit from SDN because P2P networks have very dynamic users (who frequently enroll and leave). Bandwidth or network demand varies also frequently. The flexibility that SDN has over traditional networks and its ability to dynamically accommodate users' demands fit most of P2P use cases. As security is always a major concern of P2P networks, SDN solutions need to provide security mechanisms to prevent possible intrusions.

[Suh et al., 2014] presented an SDN based firewall over POX controller. They used attributes from OpenFlow 1.1 to allow users to add firewall rules. They showed preliminary experimental results based on generated flows. [Sethi et al., 2013] formally modeled SDN controller behavior. They assessed the validity of their model using a simple stateful firewall module. An instance of a simple scenario to prevent direct connection from the Internet to the enterprise network is used in the evaluation. The activities or processes between the firewall, controller and switches are formally defined. In general formal modeling approaches are applied on low scales and have scalability limitations.

4.1.3. *SDN-based stateful firewalls*

With the ability of SDN to have a global view of the network, it is hoped that stateful analysis of the network or particular flows will be possible. Stateless network analysis studies packets or flows individually without considering other packets, flows, or flow rules and without looking at some other network, system or environmental variables. On the other hand, stateful analysis takes combined views of the whole rules or traffic in the network. A stateful firewall should be able to record and keep track of traffic history. It may also need to handle different protocols together (e.g. TCP, UDP, ARP, ICMP, etc.). Controller can trigger stateful packet inspection by ordering switches to send all packets to it (i.e. all attributes with wild card values). However, there are several challenges in implementing such feature. For example, real

time scenarios make it hard to observe many packets over a period of time. Reconstructing a complete stream may not be possible given that some content may change across the network between forward and reverse traffics or due to forwarding. The dynamic change of the topology makes the verification of current/historical variables very complex. Switches may dynamically change or rehash some header entries when they forward packets to destinations. Those are some examples of the challenges and the open research areas on how to conduct stateful firewall tasks based on SDN.

A complete stateful packet inspection in the whole network can occur only through the controller and not switches. Packet level information in the controller is provided with a limited access [Shirali-Shahreza and Ganjali, 2013a,b]. Forwarding planes are stateless and without the controller active monitoring of flows stateful inspection is impossible [Song, 2013].

Stateful firewalls can be used to detect security attacks. [Katta et al., 2012] presented Flog, in which a stateful firewall application can be built using programming languages, in few lines. They used stateful firewalls for detecting possible malicious code from insiders. However, their approach represents only a small example of what a stateful firewall should do. Flog saves senders and receivers' addresses and assume that externals are trusted if they previously received packets from network internals.

[Zhu et al., 2014] introduced SFA, stateful forwarding abstraction in SDN data plane. The goal is to provide packets stateful network processing that may require upper layers (L4-L7) information. A forwarding processor (FP) is proposed to extend SDN controller functionality. Packets are forwarded to this processor which will perform further processing on those packets, including state related storage and inspection. FP module can also interact with events or triggers from the controller itself such as network or topology related changes.

[Stoenescue et al., 2013] proposed using symbolic execution for networks' stateful checking. They developed a tool called Sym-Net to model basic stateful middle-boxes. Network stateful checking can help in making contextual firewall decisions. Those decisions do not depend only on L2-L3 information but can have information from possibly all network layers. While progresses in this area are very premature, however SDN features promise the expansion and advances in this area. Similar to most security challenges that face SDN solution, robustness and scalability are major issues. In the case of stateful inspections, heavy memory resources, storage and network resources are all required and necessary to conduct stateful inspection at mature levels or cases. State explosion is also another challenge. If we consider the network state as the traffic flows and rules in the network, this means that any single change in one of those flows or network elements will cause a state change. This can produce a tremendous amount of possible states.

[Fayaz and Sekar, 2014] proposed FlowTest to test stateful network cases of firewalls and policies in SDN. They focused on data plane testing to systematically test stateful behaviors. Policies typically include high level stateful instructions. For example, a policy may say "Block unsolicited connections from the Internet". Such policy has no reference to any (L2-L3) information. This requires firewalls to work beyond L2-L3 layers. States are specified per TCP connections (i.e., null, new, established, or invalid). They represent traffic states, not network states. A proxy module that operates at the session level is also proposed to support in the process of stateful inspection. A proxy state is expressed based on HTTP objects.

4.1.4. *Hybrid firewalls*

Hybrid firewalls refer to firewalls that work in an environment with mixed SDN and traditional networks. [Pan et al., 2013] proposed FlowAdapter to handle flows in heterogeneous OpenFlow switches. Flow tables in OpenFlow switches should be able to deal with legacy hardware. In addition some field types exist in flow tables have no equivalents in legacy switches. In fact, OpenFlow protocol itself is evolving where earlier versions have 12 attributes and new versions have 40. There is always a need to support backward compatibility and at the same time ensure that valuable information is not dropped or ignored due to such transformation. Adaptors are necessary to provide such transformation dynamically.

The process of transforming firewall ACLs from one system to another or from one domain to another can be time consuming. Typically security administrators use the expression "The devil is in the details" to indicate that the real complex and time consuming part of the process is not the technical part. Existing research discussed migrating firewall ACLs from traditional networks to SDN [Gamayunov et al., 2013]. Those reports claim that the process can take less time and effort given the ability of SDN or OpenFlow network to evaluate policy rules automatically.

[Shin et al., 2013a,b] proposed a security framework to allow legacy security systems interact with OpenFlow network. [Hand et al., 2013] introduced "active security" as a programming environment to configure and evaluate firewalls' configurations. They extended Floodlight by connecting it to open source IDS Snort along with some other applications. Active detection means combination of monitoring and prevention or detection with protection. However, this interaction between Snort and SDN is primitive and not coordinated (i.e. no real time interaction). Snort log output is extracted when alerts occur and is then added as an input to the controller. In typical complex scenarios, major concerns will be related to detection accuracy and also performance or network overhead. There are some other trials to integrate Snort with OpenFlow. The challenge is that SDN collects and inserts flows in a structure that is not compatible with traditional networking that current Snort versions are adopting.

4.2. *Access control*

SDN is a candidate to offer flexible and dynamic access control solution. [Casado et al., 2009] proposed Ethane SDN architecture that allows managers to enforce hosts' controls through fine grained access control policies. Ethane represents an early effort in SDN that inspired the OpenFlow protocol and central management of network or global policies. Ethane used flow based networks and a central controller. Switches direct flows to the controller to make decisions about. Policies are held in a central controller.

Inspired by Ethane, [Nayak et al., 2009] discussed dynamic access control and monitoring in SDN networks. An access

control system called Resonance is connected directly with real time monitoring which can accelerate the cycle from getting information alerts to taking actions. Access control system can be closer to the action points and can respond and take actions in real time based on current traffic. Traditional middle-boxes such as firewalls, etc. are often placed at the edge of the network. The study showed that dealing with access control dynamic interactions in SDN can be easier than that in traditional networks. Access control policies are enforced based on flow level information and real time alerts. Monitoring subsystems are integrated with the controller to assist in the access control process. Similar to Ethane, the controller enforces access controls through policies that are installed in switches.

[Wen et al., 2013] proposed PermOF, a fine-grained access control system in SDN. The major goal is to secure the controller and secure communication with the controller. PermOF includes a list of 18 possible permission levels for minimizing possible intrusion or privilege escalation. The permission system is combined with run time isolation (between controller and applications). It offers a default least privilege permission for OpenFlow applications. The ability to successfully isolate applications from the controller is a key for the applicability of such approaches.

[Yamasaki et al., 2011] proposed an SDN based VLAN solution for campus networks. In addition to the VLAN IDs problem, authors indicated an overhead problem related to the extensive time required to implement and maintain VLAN database. Access Management Function (AMF) module is added to track and authenticate users or hosts. System is evaluated with 10,000 IDs. Evaluations showed that SDN based solution can outperform traditional solution. SDN solution is also dynamic and is expected to reduce a significant amount of maintenance overhead.

[Kinoshita et al., 2012] proposed an approach for OpenFlow based access control and authentication system for wireless campus networks. They pointed out two limitations in [Yamasaki et al., 2011] approach and proposed enhancements on those limitations. The first limitation is related to the inability of the earlier system to work in anonymous user authentication mode. The second limitation is related to the cost of users' DB maintenance. Rather than dealing with individuals, they can be clustered into groups and authentication can be made based on the users' groups. This can reduce the size of the DB that authentication system needs to search through. Authentication system needs not to look for names, but rather for groups which may also help in dealing with anonymous users.

[Wu et al., 2013] discussed programmable virtual networks (PVN) in the cloud based on MAC isolation. A PVN server is proposed in OpenStack to act as an OpenFlow controller. Local agents are delivered in the network to support PVN controller to filter traffic based on MAC addresses.

4.3. IDS/IPS

Intrusion detection/protection systems (IDS/IPS) stop or allow packets based on thorough investigation of packets using data mining, pattern recognition, signature matching with existing inventory of threats, etc. Unlike traditional IDS, SDN IDS can utilize the tremendous amount of flow information in real time. SDN can change the way security mechanisms are distributed. For example, an IDS exists in one location in traditional networks (usually in the network premises). In SDN, IDS tasks can be distributed through the switches or agents in the network. Controller or one of its modules can orchestrate the process [Rothenberg et al., 2012].

4.3.1. *Integration with traditional tools*

Existing research has tried to integrate some popular IDSs such as Snort with SDN [Ballard et al., 2010 and Xing et al., 2013]. Integrating Snort with SDN faces several challenges. SDN controller typically receives samples, not complete flows which contradict with how Snort works. A common way to set things up is for the controller to receive the first packet or the first few packets of a given flow. Once having received those, the controller installs rules in the switches that will handle the rest of the packets in that flow. This is done because typically sending each packet to the controller is impractical. Since Snort expects to see every packet in a flow, we will not be able to put Snort inside the controller effectively without vastly impacting the performance and the structure of OpenFlow network. An alternative design would be to create a service in the controller to manage a set of machines running Snort and to install rules that redirect traffic to the machines running Snort.

Snort has its own limitations when it comes to the type of attacks it can detect. While being a good open source IPS/IDS (with a rule based language combining signature, protocol and anomaly based inspection), Snort is still reliant on regular signature updates. It has no way to detect higher level exploits such as web exploits (e.g. malicious Java Scripts). Snort may not also help with attacks such as: Advanced Persistent Threats (APTs). SDN and Snort differ also in the way they collect, reroute and monitor traffic. In traditional networking span ports are used to reroute traffic for monitoring or security applications. In SDN, data can be extracted from the controller through northbound APIs. Filters can be applied in SDN to extract traffic based on certain criteria and command controller to rewrite traffic based on those criteria.

[Xing et al., 2013] investigated integrating Snort with OpenFlow networks. SnortFlow is capable of reconfiguring the cloud system on the fly to detect and counter intrusions. This work came as an extension or enhancement for NICE system in [Chung et al., 2013a,b]. It uses Snort for coordinated attacks' detection. SnortFlow includes three components: A daemon to collect alerts data from Snort agent, an alert interpreter to parse alerts and decide which traffic to target, and finally rules' generator that will inject rules in OpenFlow switches. Changes caused by the new rules are saved to allow possible roll-back or restoration. Countermeasures to take are classified based on cost and intrusiveness. Careful consideration should be made on the proper counter measure to take so that it will not interrupt normal operations.

FRESCO and its successor project SE-Floodlight [Shin et al., 2013a,b] produced several security applications related to SDN. One of those recent extensions is FlowBoss. FlowBoss is hosted in SE-Floodlight and it imitates in OpenFlow what Snort is doing in traditional networks. Network policies can be specified to prevent unauthorized access. Policies can be also

specified to rate limit traffic in certain times, or filter traffic based on certain characteristics.

4.3.2. SDN IDS implementation

[Goodney et al., 2010] presented an implementation of SDN-based NIDS on the NetFPGA networking platform. It can be used to test FPGA algorithms for Deep Packet Inspection (DPI) or high speed programmable packet processing. The module can conduct network intrusion detection through DPI.

[Skowyra et al., 2013a,b] discussed OpenFlow based NIDS in embedded mobile devices and Cyber-Physical Systems (CPS). Applications or case studies include robotic transport and biomedical devices as they have similar threat models. In general, mobile devices are subjected to attacks within the device coverage range (i.e. modem, Wi-Fi or Bluetooth). Encryption is usually suggested as the main security mechanism to eliminate such attacks. However, for some small commercial applications, strong encryption methods can be infeasible or expensive. Location based security mechanisms may not protect from local users or insiders. Proposed IDS or Learning IDS (L-IDS) can be used to support encryption or location based security mechanisms. Anomalies are defined based on several characteristics: Packets' sent, position, time passed, size, etc. For each one of those characteristics, normal range is specified. Deviation from such normal range can be classified as an anomaly.

[Kerner, 2012] represents Indiana University experience with building an (Intrusion Protection System) IPS based on SDN. Major advantages of the new system were related to load balancing and the ability of the network to handle and distribute traffic based on security controls. Global policy based network security management is another important goal that SDN based IPS is expected to achieve.

[Chung et al., 2013a,b] presented a system on Network Intrusion Detection/Protection System (NIDS/NIPS) in the virtual networks using OpenFlow based programmable APIs. A graph based analytical attack model is proposed to detect and counter attacks on VMs. The system periodically scans VMs and decides based on the severity of detected vulnerability to put the VM in an inspection state or not. In the attack graph, each node represents either a pre- or post-condition of an exploit. The graph can provide details of connectivity between different vulnerabilities or exploits. An Alert Correlation Graph (ACG) is mapped to a Scenario Attack Graph (SAG) that includes the exploit, steps to reach the exploit and its post-condition or results. They focused on a small subset of flow information including only five attributes: Source and destination MAC and IP addresses in addition to the protocol.

[Heorhiadi et al., 2012] discussed NIDS problems from scalability perspective. SDN flexibility methods can offer promising solutions to this challenge and load can be distributed or sliced among different controllers. Traditionally, IDS needs to monitor all traffic which is very time consuming and produce a large volume of traffic for analysis. IDS or NIDS load can be reduced by replicating the traffic to the nodes that are off-path after making sure that they are available and have free resources. In addition, the fact the SDN controller can aggregate data from different switches in one location can also be an important characteristic to intrusion detection or protection systems. The central NIDS module periodically collects information about traffic and policies. It can be also triggered by certain traffic changes or events. The traffic itself needs to be classified or categorized into different classes. Each class can be subjected to different types of NIDS analysis. An intelligent engine can be used for initial analysis of traffic to specify the type of analysis to subject the traffic to. This can be an evolutionary process that learns from past experience or traffic and improve accuracy in future.

[Braga et al., 2010] is an example of using SDN for the assessment of security vulnerabilities. This work focused on the detection of DDoS attacks. Similarly, [Mehdi et al., 2011] focused on anomaly detection methods based on SDN in home networking. They evaluated the impact of SDN on traditional anomaly detection methods and measured the efficiency of intrusion detection methods based on low traffic rates.

[Giotis et al., 2014] proposed combining OpenFlow with sFlow to improve flow-based anomaly detection. Flow statistics can be a good source for inspecting possible anomaly behaviours in the network. Collecting statistical data through the controller faces a serious scalability issue. Consequently there are many research proposals to outsource this task to a separate supporting module. [Giotis et al., 2014] conducted a study with high packet rates (up to 130,000 packets per second). Flow information collected is based on a subset of attributes from the old version of OpenFlow that includes only 12 attributes. They evaluated data collection based on native OpenFlow and also using sFlow. Native methods can be applicable in low to medium size traffic. This is since there are some limitations on the size of flow entries in the switches' flow tables. The sFlow approach decouples the flow collection process from the forwarding logic where packet samples provide all necessary information. This can show a significant reduction in size.

The above work focused on information related to (L2-L3) layers without looking at the actual packets' contents. [Shirali-Shahreza and Ganjali 2013a,b] proposed an extension to current OpenFlow protocol to allow controller to have access to packets' contents. Current information exchanged between the controller and switches is largely related to routing information. The goal is to use such information for security applications including NIDS/NIPS or anomaly detection tools. In some cases, samples rather than the complete traffic are sent to the controller. Different sampling methods (e.g. deterministic or stochastic sampling) can be requested by the controller based on the nature of the security application or middle-box. Full packets are only requested under certain conditions.

4.4. SDN policies

SDN is expected to facilitate automatic configuration, assessment, and enforcement of network policies. While policies in traditional networks are embedded in firewalls and their ACLs, SND allows for policies at different levels of abstraction. Thus, we separate our discussion on SDN policies from that on firewalls.

Policies that regulate operational activities are considered high level guidelines that can be translated and enforced by

low level security mechanisms such as firewalls, proxies, etc. Traditionally, the translation from high level policies to concrete security mechanisms is mostly conducted manually by network administrators. SDN brings a new opportunity for security policies to be interpreted, updated, evaluated and enforced by automatic tools with least human intervention.

4.4.1. *SDN policy languages*

Policy languages have been proposed for writing formal or semi-formal policies. The main goal is to bridge the gap between two different levels of abstraction: Human natural languages in which administrators start writing high level policies and low level rules that machines can understand or interpret.

[Hinrichs et al. (2008, 2009)] proposed Flow based Management Language (FML) to express access lists and policies for NOX controller. FML itself is based on DATALOG; a declarative logic language used in the connection with databases. In the core of policies there are rules and Access Control Lists (ACLs). Policy enforcement is implemented using a decision tree to reach the right matching rule for the current flow or the packet based on the rules in the flow table and/or the firewall. FML maintains states related to lists of users and their devices or hosts. Access control decision is then made based on the values of those flow fields or attributes. In addition to "allow" and "deny", there are other decisions in FML: Waypoints, avoid, and rate limit. Waypoints or reference points are defined to mark certain known points (e.g. hosts, a server, a gateway). "Waypoint" and "avoid" are opposite to each other (i.e. to order traffic to reroute or skip those network points). Rate limit indicates a rate limitation (i.e. maximum allowance) on the traffic.

[Ballard et al., 2010] proposed ALARMS, a flow-based specification language to interact with OpenFlow flows. This can be used as a tool with administrators to enforce policies through controlling and routing traffic. The work extended earlier FML to include attributes related to the flow content. This enables access control and manipulation beyond L2-L3 layers. For example, a security administrator may want to limit chat or peer to peer applications. They may want to limit programs that consume a large amount of bandwidth. Previous FML fields have little abilities to allow administrators to make policies based on actual packets' contents.

[Foster et al., 2011] introduced the Frenetic language for programming network switches. Frenetic is developed not only for policies but also to generally assist in network services; routing, access control and traffic monitoring. It has two levels of abstraction; high level to construct and manipulate network traffic and low level to interact with switches. This may solve the contradictory constraints that policies need to handle: on one hand they need to be expressive enough to cover administrative high level requirements, and on the other hand they need to implement these requirements as rules in switches in their terms (i.e. flows). One problem with NOX controller [Gude et al., 2008] that Frenetic tried to solve is the modularity issue of policy rules if written through controller program. In general, it is not modular or reusable to write policy rules inside the controller program. Rules are expected to be very modular as they may change frequently. Hence, it is very important to separate them from the controller code. A Frenetic program can be developed to represent an instance including network policies. This program will be able to enforce policies through the controller. [Monsanto et al., 2012] introduced the NetCore language for expressing packets' forwarding policies in SDN. It includes constructs to analyze packets and historical traffic patterns. New algorithms are designed in Frenetic for compiling rich policies and for managing controller and switches' interactions.

[Foster et al., 2013] contribution is an improvement on Frenetic. It showed a rich query syntax (e.g., Not equal, GroupBy, Select, Limit, Every) that can help optimizing policy rules and allow administrators to have more control and semantic in writing policy rules. The controller can then handle transforming those policy rules into low level details understood by switches. [Katta et al., 2012] proposed (Flog), a network programming language that can be considered as hybrid between FML and Frenetic.

To improve expressiveness in network and security policies, [Voellmy et al., 2012] introduced Procera; a control architecture that includes a declarative policy language based on functional reactive programming. Procera tries to help network designers to implement expressive policies without the need to use programming languages. Procera includes signals and signal functions as reactive concepts. Signals are like transient functions where functions are attached with a period of time. Signal functions or constructs cause transformations on signals. There are other research papers related to Procera and network programming. [Voellmy and Hudak, 2011] discussed examples of applications using network programming including a learning switch and traffic monitoring applications. [Kim and Feamster, 2013] extended the work of Procera and described how it can help in network management. Main goal was to propose a solution that can compromise between the need for rich and expressive high level policy features and at the same time the need to interact with low level details at the switches or networking components' level.

[Anderson et al., 2014] proposed NetKAT network programming language based on a mathematical structure called Kleene Algebra with Tests (KAT). NetKAT can be used to express OpenFlow requirements through adding and interacting with flows. It provides a high level algebra for complex reasoning and query of flows.

4.4.2. *SDN security and network policies*

[Casado et al., 2007] described the interactions between the controller and security policies as rules injected by the controller in switches. Those however were imitating traditional ACLs. [Nayak et al., 2009] proposed the Resonance security mechanism for dynamic access control evaluation based on flow level information. Resonance interacts with high level policies to make decisions on flows. It uses a policy specification framework based on traditional or existing access control frameworks.

[Feamster et al., 2010] used OpenFlow to solve policy problems in campus and enterprise networks. They tackled two challenges; access and information flow controls. The gap between high level expressive policies and low level access controls exist in switches or firewalls continue to be a serious

challenge for administrators in dealing with large networks. For information flow control, traditional approaches are host based. If the host is compromised, the information flow can go out of control.

[Ferguson et al., 2012] introduced the concept of Hierarchical policies. Policies can be composed of or contained in other policies. A policy tree is then constructed based on the hierarchical relations between the different policies.

[Wang et al., 2012] presented a security management architecture with interactive policy enforcement. Their version of the controller is supported with a policy table that interacts with packets while they are traversing the network. Performance can be an issue here especially as the model is applied on a small size network with only 50 users. They utilized different security service elements including: IDS, protocol identification, virus scanning, load balancing, traffic monitoring and content inspection. [Son et al., 2013] introduced Flover model checking system to verify OpenFlow-based flow policies. They focused on testing for non-bypass-ability or to test that current flows adhere to firewall rules. They extended earlier approaches by including new features such as "set" and "goto".

Policy migration from traditional networks to SDN was also the subject of many research papers [Vanbever et al., 2013; Vanbever et al. 2014; Zhang et al., 2014]. Policy migration is considered another advantage for using SDN where it is expected that the process of migrating policies is less time consuming in SDN in comparison with traditional networks. Traditionally, the migration process which is manually implemented can take a long time and man power specially to address out of date or conflicting rules. Typically, the term "The devil is in the details" is used to show that the process is not complex from a technical but from an operational or practical perspective. Policy migrations can be a good show case of SDN use cases.

[Gibb et al., 2012] proposed outsourcing some network functionalities from the controller to external components. A policy API is included with several enabled features that are location independent from the controller. Those features can be called on demand whenever needed. The management and control of those features are outsourced from the controller to improve performance and reduce centrality. This proposal is close to the concept of web services offered in Service Oriented Architecture (SOA). Services are known and accessed by their public interfaces. Service providers are separated from consumers where the same service can be used in different contexts. From security perspectives, communication between service providers and the controller is critical. It should be developed with security in mind as those interfaces can act as back doors to access the controller and its core modules.

4.4.3. *Policy enforcement*

Automatic enforcement of security policies is an important task that SDN can achieve. [Bellessa et al., 2011] presented an approach to dynamically enforce flow level policies in cloud networks. Policies are written by administrators in high level languages. Those policies are then interpreted by the policy evaluation and compliance monitoring system (ODESSA) based on the network components and actual flows. In other words, ODESSA is responsible to transfer abstract policies into concrete implementation based on network specifications.

[Fayazbakhsh et al., 2013] focused on the issue of consistent policy enforcement and flow tracing or tracking. In their proposed enhancement to SDN architecture, they proposed to add contextual information to flows. In this proposal a southbound-controller middle-box will add tags to outgoing packets where those tags can be used for systematic policy enforcement. Currently OpenFlow is the only standard protocol in the southbound-controller communication. Those applications may dynamically change packets' headers. Such headers' modifications may have a negative impact on policy enforcement and may mislead the process that enforces those policies. In some cases, the same application that is supposed to perform policy enforcement may change those headers and consequently make the process difficult on itself.

[Qazi et al., 2013] proposed a middle-box layer to deal with traffic steering for those middle-boxes. A flow correlation mechanism is proposed to handle the issue of packets' induced transformation mentioned in the previous research [Fayazbakhsh et al., 2013]. There proposed solution; called SIMPLE, tried to deal with existed OpenFlow architecture and constraints. This solution represents a policy enforcement layer to manage communication between middle-boxes and the data plane. This design however, imposes a special purpose controller to interact with switches and middle-boxes. However, it is not clear how this special purpose controller is going to communicate, with OpenFlow protocol and with the main controller.

[Kazemian et al., 2013] discussed another challenge in policy enforcement and evaluation; real time issues. Due to the rapid change of network state, performing policy checking frequently can be resource and time consuming from practical considerations. In addition, such network rapid change may require policies to be frequently reevaluated. NetPlumber is proposed as a possible solution to this challenge. NetPlumber utilized a previous approach on static checking for the same authors called "Header Space Analysis, HSA". NetPlumber frequently checks for state change based on studying rules' change from a dependency graph modeled from those rules. In the graph, nodes represent forwarding rules from switches' flow tables and edges represent next hop dependency in those rules. Probe or check nodes can be used to incrementally check policy or invariants for possible modifications. Different events trigger changing the rules' graph and consequently require policy reevaluation.

[Bari et al., 2013] presented PolicyCop as an OpenFlow based interface or policy management framework. The framework allows monitoring specific parameters in the network and adjusting them based on Service Level Agreement (SLA). The framework contains several functional components including: A policy validator, checker, enforcer, traffic monitor, topology manager, etc.

4.5. *Monitoring and auditing*

Monitoring and auditing are very important tools for many security controls. A significant opportunity in SDN networks is related to the amount of details that can be gathered at the flow and even the packet level. This was difficult or resource consuming to achieve in traditional IP networks.

4.5.1. *Traffic monitoring tools*

[Nayak et al., 2009] proposed Resonance; an OpenFlow based solution that provides continuous monitoring distributed across the network. Network elements or switches forward traffic to the controller. As an early contribution in OpenFlow, this paper shows how the nature of OpenFlow architecture or process flow can help, natively, the monitoring process.

[Ballard et al., 2010] proposed OpenSAFE (Open Security Auditing and Flow Examination). This is a tool that leverages SDN to improve network monitoring. Monitoring tools use Span ports in order to create copies of network traffic for monitoring purposes. Usually network tools allow a limited number of Span ports. Firewall modules and IDS in traditional networks usually use one or more of those Span ports. This is why such monitoring tools cause significant network overhead if fully implemented. Filters that are used to reroute traffic can look similar to those firewall rules or flow tables. They can be built using the same match fields or features. However, there should be more expressive tools/mechanisms to reconstruct packets than those available in firewall or flow table rules. They include mathematical operations such as less than, more than, and sorting options related to the collection, query and statistics of traffic. With the use of OpenFlow networks, OpenSAFE can direct spanned traffic in arbitrary ways while such traffic can be used by several simultaneous services or security controls such as IDS and firewalls.

[Huang et al., 2011] proposed implementing dynamic measurements aware routing or forwarding for traffic monitoring. They discussed three challenges related to traffic monitoring: The dynamic assessment of traffic importance, flow aggregation, and finally how to perform traffic monitoring with least network disruption or network overhead? Information from OpenFlow switches are used in those tasks. Flow importance is estimated through its size. Controller can retrieve size of flows using flow-query/flow-expire messages.

[Shin and Gu, 2012] introduced CloudWatcher as a monitoring tool for cloud services. Some packets, based on security concerns, will be detoured to a security check point for further security inspection. The application includes three modules: Device and policy manager, routing rule generator and flow rule enforcer. This work addressed two issues in cloud traffic monitoring: The need to consider both insider and outsider threats and to consider that cloud networks are very dynamic where hosts or network components may change frequently. CloudWatcher is proposed as a controller northbound application. It should provide this monitoring service to different available security mechanisms. Security Aware Routing (SAR) is also proposed in CloudWatcher paper. SAR is used in some traditional networks such as Ad-hoc networks. SAR algorithms try to ensure that packets should go through certain security check points (e.g. firewalls, access control, etc.) for packets' checking.

[Argyropoulos et al., 2012] proposed PaFloMon passive slice-based monitoring tool for OpenFlow networks. The target is OFELIA European open SDN testbed. Slices in this case represent different users or experiments that are using the testbed. The monitoring tool sFlow is used and integrated with OFELIA. SFlow can help in monitoring and statistics as well as instrumentation for conducted experiments.

The level of details a monitoring tool can collect related to flow information is very important. [Shirali-Shahreza and Ganjali, 2013a,b] proposed FleXam; an extension to current OpenFlow protocol. The goal is to allow controller to have access to packet level information and packets' contents. Such information is necessary and required by most security mechanisms. Current information exchanged between the controller and switches is at the flow level that does not include packets' contents and is related to routing information only. Some approaches such as that of [Mehdi et al., 2011] proposed a solution where controller will not install flow based rules in the switches. This causes the switches to send packet level information to the controller. However, this approach may not be realistic given that the controller will be overwhelmed with packets and the overall network delivery time will be slow. As an alternative, those packets can be sent to a special monitoring tool. A compromised solution is proposed between those two alternatives. FleXam enables the controller to access samples of packet level information in switches. Those samples are selected based on controller choice using some statistical algorithms (i.e. statistical sampling).

[Raumer et al., 2014] discussed also sampling in OpenFlow traffic. They differentiate between security monitoring and Quality of Service (QoS) monitoring. In security monitoring, we are looking for possible traffic patterns that may indicate an attack. Sampling methods have no significant impact then on security monitoring, given a detection of an attack. However, in QoS monitoring, sampling can show incorrect picture of network health. For example a particular sample we are investigating may show good performance while the rest of traffic is facing an opposite situation. Same thing can be said given other quality attributes.

[Qazi et al., 2013] pointed out that SDN firewalls and other security controls are not going to strictly work in the L2/L3 layers as in traditional networks. They presented a policy enforcement layer specifically for traffic steering or monitoring. This policy can manage traffic steering based on users' or applications' requirements.

[Chowdhury et al., 2014] discussed quality factors and trade-off in traffic monitoring. The trade-off is usually between monitoring accuracy, timeliness and network overhead. Optimizing one of the three quality attributes can be at the account of the other two factors. Payless is proposed as a monitoring framework. The goal is to best optimize monitoring given the three quality attributes mentioned earlier. Monitoring information from Payless can be exchanged with different security controls or applications. Payless provides a standard RESTful API that allows tools to make and retrieve traffic queries.

[Yu et al., 2013] focused on performance monitoring in OpenFlow networks. They proposed a push-based approach where the network switches initiate information related to performance degradation or problems. Such approach can reduce overhead where information is only sent when performance degradation occurs. Calibration is required to compute normal performance range in a particular network. Any deviation from that range can then be reported by the network. FlowSense tool is developed to measure flow-based bandwidth consumption. They focused on two OpenFlow

messages between switches and the controller: PacketIn and FlowRemoved. Those are relevant to measuring flow bandwidth as they represent the start and the end of a flow stream, respectively.

[Karame, 2013] focused on security issues in network measurement tools and the impact of OpenFlow switches. Author investigations showed that most measurement tools are not developed with security in mind. Host or end to end measurements trust hosts and ignore possible insiders' threats or threats that compromise hosts. Author analyzed several examples of network security threats. Author also showed examples of how OpenFlow can be better improved in terms of security in response to selected security threats. Namely author selected two issues: Bottleneck bandwidth estimation and network coordinate measurements. Author proposed a scheme based on OpenFlow to secure communication or flow traffic from being attacked or compromised.

[Zaalouk et al., 2014] evaluated SDN features such as network visibility and control centralization as possible solutions for some security vulnerabilities. They proposed OrchSec; an orchestrator that utilizes network monitoring and SDN control to develop security applications. This architecture may mitigate some attacks that do not need a deep look at packets' contents (e.g. Worms, DoS, etc.).

4.5.2. *Traffic management*

[Curtis et al., 2011a,b] proposed Mahout, a traffic management system for dealing with large or elephant traffics. There are many security threats or attacks that push large traffics. Examples of those include: Worms, DoS or flooding. Mahout is a controller based on OpenFlow architecture where hosts, rather than the switches, are expected to monitor possible large traffics. Hosts monitor such large traffic in coordination with the controller which manages the process of handling this large traffic. Each host monitors possible large traffics and communicates with the controller once a large traffic is detected. Native single OpenFlow controllers suffer from scalability issues especially in dealing with large traffic. Using the host to handle large traffics can relieve the network from handling and waiting for traffic in progress which can be delayed for several possible reasons or problems. To detect possible large traffics from end hosts, their socket buffers can be used. A threshold is set as a variable which can define the edge of a large traffic volume.

[Jain et al., 2013] discussed a WAN SDN based solution for routing and traffic engineering in B4 Google data centres. OpenFlow is used to manage switches and optimize bandwidth usage. Google SDN solution includes OpenFlow controller and also Network Control Applications (NCAs). NCA directives' and switches' events are used by controller to maintain network state.

[Wang et al., 2013] discussed the problem of traffic load variation and how to handle it in OpenFlow networks. Traffic is continuously studied and investigated to predict possible traffic overload. NetFuse causes little overhead over the network or the controller as it uses traffic data already collected by the controller. It is important to predict whether a large traffic is related to normal or intrusion causes. Such problem can be classified as difficult; Nondeterministic Polynomial (NP) especially as the term (large traffic) is subjective and vary in size and threshold based on the nature of the traffic and the reason for packets' aggregation. Machine learning classification algorithms can be applied in this particular subject. NetFuse tries to find the best reasonable flow aggregation and the possible overloading reason.

[Jose et al., 2011] proposed a solution based on OpenFlow switches. The process is based on switches to count packets in traffic as traffic traverses. Once the size passes a certain threshold, a flag can be raised that this traffic is large. One problem with this solution is that it delays the discovery of a large traffic till it passes. This is justified by authors as a trade-off between accuracy and low overhead. Examples of several attacks that may use large traffics are shown and how their proposed solution can be used to handle those types of attacks. As mentioned earlier, one problem with identifying a large traffic is in selecting the threshold itself. This is since this threshold value is subjective and context dependent. In other words, what can be considered large for some applications will be considered normal for others.

[Sun et al., 2014] proposed a traffic management solution (HONE) based on joint information from OpenFlow network and end hosts. Data is processed locally in the end hosts. A trade-off between host and network elements is discussed. Applications in the host have a better visibility into applications' behaviours while have little knowledge about the network behaviour. On the other hand, network switches have the opposite picture. Consequently an effective monitoring or traffic management approach should try to integrate information from both sides together. One challenge with handling host-based network management however is that hosts have several components and applications that interact with the network and information about the network exist in several locations. HONE agents run on hosts in addition to a module that interacts with OpenFlow switches.

[Choi et al., 2014a,b] discussed OpenFlow management and control challenges given the centrality and scalability issues. They proposed an SDN monitoring agent or middle-box (SUMA). SUMA is proposed to integrate logically management, control and monitoring services. SUMA takes the overhead of the monitoring process from the controller. It alerts the controller in case anomaly behaviours occur. They demonstrated some attack scenarios and how they can be detected using SUMA. This middle-box acts in the southbound section between the controller and switches. One problem with such approach is that it changes significantly SDN architecture that currently has only OpenFlow as the only adopted protocol in this side.

[Rasley et al., 2014] introduced Planck, traffic management framework for providing scalable traffic data with short or small time scales using port mirroring mechanisms. Traffic from switches is mirrored to a designated port. Traffic data is an important asset for all security applications. Collecting and analyzing such data with high accuracy, real time and least network overhead contribute to improving security controls and attacks' detections. Mirroring in OpenFlow has an advantage over traditional mirroring using Span ports. This is since mirroring in OpenFlow can be customized. We can specify or extract certain traffic based on customized criteria or query. Different security tools can extract different information based on their needs. This makes the mirroring process very focused and optimized.

4.6. Security control for mobile SDN

The idea of separating control plane from data plane can be extended to wireless networks. Traditionally control is mixed between software and hardware where different mobile vendors have different architectures. The process of adding new functionalities can be very slow and complex. Wireless or cellular SDN refers to extending SDN architecture to wireless or cellular networks. In this section, we will focus on research contributions in SDN mobile networks in particular. Terms such as: SDW, SDC, SD Mobile Networks (SDMNs), or even CellSDN are used to refer to SDN-based mobile implementations. For simplicity and consistency, we will use SDW.

SDW controller is expected to provide fine-grained policies based on subscribers' attributes. Controller northbound APIs or middle-boxes provide many services such as: Mobility manager, accounts or subscribers' manager, radio source manager, policy and charging managers, interference manager, and infra-structure routing. It may also include security applications such as firewalls and IDSs.

One of the major SDN goals was to come up with a uniform or a standard switch architecture for the different networking vendors. In wireless, there are typically different cellular service providers as well as different manufacturing companies. Having a standard communication architecture can be consequently very important and useful.

[Li et al., 2012] proposed an approach for SDN wireless (SDW) with fine grained policy management. The SDW approach is expected to be cheaper than traditional wireless approaches due to the ability to use OpenFlow switches that *should be cheaper* than traditional ones. As traditional wireless switches are expensive, carriers may overuse them based on load requirements. Interference can be also reduced in SDW where controllers of different carriers can have a global view of their network and consequently can better communicate and improve issues related to interference. Different technologies and carriers may communicate effectively through this common new open architecture. New features such as usage based cost or pricing are possible using SDW. Virtualization in wireless and cellular networks will have new challenges to deal with related to: Billing, interference, radio signals, etc. where such issues should be revisited given the new architecture for both possible benefits and challenges.

[Hampel et al., 2013] argued that SDN can be an effective architecture to solve problems in mobile networks. They proposed vertical forwarding as an extension to current OpenFlow to handle mobility policy or access control management or in particular fine grained forwarding. The concept of vertical forwarding is used to distinguish their proposal from OpenFlow forwarding schemes between switches and their controller (i.e. horizontal controller). Vertical forwarding is proposed to extend this forwarding to include legacy elements of network components.

[Namal et al., 2013] discussed the idea of switch mobility and the secure change of IP addresses. They presented a system to perform this change employing IPsec encapsulated security payload. Current issues in OpenFlow which make it inapplicable in its current format to wireless networks are discussed. For example, first, changing addresses will disrupt flow processing. It should be ensured that, such process occurs very fast and dynamic. This may also impact secure session management required for secure communication between switches and the controller. The fact that mobile networks are very dynamic and very fast moving in terms of active users can both give opportunities and impose challenges to applying SDN on mobile networks. HIP (RFC 5201: Host Identity Protocol) security method is adopted to be used in OpenFlow connection to enhance existing communication method from a security perspective. HIP identifies a host either by a host identifier or a host identity tag. An extension to wireless on switches called (flow control agent) is also proposed which should update controller of the new location information for location based services.

[Skowyra et al., 2013a,b] discussed security issues in embedded mobile devices and Cyber-Physical Systems (CPS). A Learning Intrusion Detection System (LIDS) is proposed based on OpenFlow networks for detecting and mitigating security attacks. Anomaly behaviour is defined as statistically *different* flow traffic from a user defined normal traffic.

[Ding et al., 2014] discussed how SDN can benefit mobile networks in terms of security aspects in particular. One of the major challenges in mobile networks is that there is an increasing bandwidth demand that current infrastructure is not keeping up with. Mobile services are also evolving rapidly in terms of nature and complexity. They also presented some SDN security solutions in general and classified those solutions into five categories: Enterprise, home networks, edge access, cloud, and general. SDN can provide the virtualization abstraction layer necessary to integrate different Internet Service Provider (ISP) platforms or services. Complexity and details of different wireless protocols can be shielded behind this abstraction layer. Programmability feature should be able to greatly improve policy management and administration not only related to security functions but also to business functions such as billing, accounting, service subscription, etc. Several design challenges are described in order to implement SDN in mobile networks including: Mobility, roaming, monitoring overhead, multi-access, multi-operators issues, interoperability, responsiveness, compatibility, adaptation, simplicity and finally deployment and how SDN can be compatible with current mobile technologies. Paper proposed a security enhancement framework dedicated for SDN solutions in mobile networks.

Strong security mechanisms in mobile applications may cause significant resources and performance overhead. To deal with such problem, [Hurel et al., 2014] proposed outsourcing security controls in mobiles to the cloud so that they can work as security services on demand. This may clearly solve the issue of using mobile resources. However, in terms of performance, it may go from one challenge to another. This is since, those security controls need to be very transparent and fast. Once those security mechanisms are in the cloud, they will be accessed through the Internet or public networks which can suffer from periodic traffic jams. In the same problem (i.e. mobiles resources' constraining), [Gember et al., 2012] proposed an enterprise centric offloading framework that leverages SDN.

As a new architecture, SDN in mobile networks can bring both opportunities and challenges. [Liyanage et al., 2014] focused on security challenges that SDW may bring. They

focused on the issue of channel or communication security. Current SSL/TLS optional encryption method in OpenFlow has some security concerns [Meyer and Schwenk, 2013]. An alternative communication security mechanism is proposed based on Host Identity Protocol (HIP) that is independent from OpenFlow protocol. A security interface or gateway (SecGW) is proposed as a layer between the controller and OpenFlow switches to hide the identity of the controller. Each switch is embedded with a local security agent (LSA) that should allow communication of the switch with SecGW. This may prevent types of Man in the Middle (MiM) attacks as SecGW is communicating with the switches through those embedded security agents. IPSec tunnels encryption is used for the communication between SecGW and LSAs or the switches.

4.7. *SDN Wi-Fi networks*

In this section, we discuss SDN in home networking and campus networks. Many early SDN use cases that were used to promote SDN were related to this category in particular. The wireless networks are very agile. Consequently security control and management are far more important and complex if compared with wired networks. Wi-Fi has some unique issues (e.g. billing management) to handle in comparison with other networks. In addition, security problems in Wi-Fi and the use of illegitimate intrusion to private networks are very popular. Handling AAA (Authentication, Access and Accounting) and differentiating them from each other is another challenge that faces current systems that manage Wi-Fi networks. This is since current methods assume one user account and management systems for the three functions. Existing research proposed solutions for this problem based on SDN [Suresh et al., 2012; Kang et al., 2013; Pentikousis et al., 2013; Dangovas and Kuliesius, 2014].

In SDN Wi-Fi security in particular, once SDN is extended to cover wireless routers, access points and switches, existed encryption algorithms such as WEP, WAP 1 and 2 should be revisited and assessed based on this new architecture. This is since those encryption methods assume that home access points or routers include data and control. However, based on SDN, controller will be separated from data and will not exist within those devices. Encryption methods should exist between users and the controller from one side and also between the controller and the access points from another side.

One of the most serious security concerns in Wi-Fi or home networking is dealing with third party applications. Those can pose threats to both local users and ISPs. On the other hand, ISPs, for many reasons, can't enforce strict security policies on their customers to control what they can download, use, etc. Home users vary widely on the nature of applications they install or use or on the type of environments they use those applications in. A security model should then handle very agile and largely ambiguous spectrum of possible threats.

[McKeown et al., 2008] discussed some of the values of using OpenFlow networks in Universities or campus networks. Those are typical examples of large networks of users and hosts. In addition, those networks are very dynamic; many new students enroll each year and many others leave. Typically Universities allow students to bring their own devices and most users access the network through wireless access points or smart phones. Such networks can be a good show case for OpenFlow; programmable networks. Several examples on how OpenFlow can be a good solution are presented.

Feamster with several other colleagues have several papers in SDN in general and in home networking in particular [Feamster et al., 2004; Mundada et al., 2009; Nayak et al., 2009; Ramachandran et al., 2009; Anwer et al., 2010; Voellmy et al., 2010; Feamster et al., 2010, 2013a,b; Koponen 2011; Voellmy et al., 2012]. Some of those papers [e.g. Feamster et al., 2004] represent early proposals to change traditional networking architecture to a programmable architecture. Those contributions along with several others contributed to emerging SDN. SDN Research Group RG was formulated and a project called Bismark (http://projectbismark.net/) was established in coordination with Internet Engineering Task Force (IETF). Their main focus was the applications of SDN in home networking. The project proposed a security architecture in home networking for monitoring traffic and dealing with security attacks. SDN and its programmability nature can introduce great benefits to home networking management. Examples of management aspects that can be utilized based on SDN architecture for home networking in particular include: Usage cap management, parental control and bandwidth management. Those can all be offered for users in real time. Procera event based programming language is proposed to write and evaluate policies. Policy language and layer can be used to facilitate communication with the controller. Upper layers or layers in the northbound section can use Procera and other policy languages such as FML to communicate with the controller.

[Mehdi et al., 2011] focused on anomaly detection methods based on SDN in home networking. It is advocated that deploying SDN anomaly detection solutions at home networking is capable of detecting more and accurately malicious codes in comparison with those deployed in ISP premises. The impact of SDN on traditional anomaly detection methods is evaluated. They measured the efficiency of intrusion detection methods based on low traffic rates.

[Schulz-Zander et al., 2014] proposed AeroFlux; scalable wireless SDN architecture to support large Wi-Fi enterprises and carrier deployments. Application aware services are implemented to optimize resource allocation based on applications' requirements or needs. Controller tasks can be divided between local controllers and a global one. Global controller control tasks that are not time critical.

Yap et al. have published several papers on SDN Wi-Fi networks [Yap et al., 2009; Yap et al. 2010, 2011]. [Yap et al., 2011] discussed security issues in Wi-Fi and a solution based on OpenFlow. The proposed solution can accomplish the logical separation of three different functionalities: Authentication, access and accounting. Wi-Fi users should not be held accountable for information content that was downloaded from unaccounted users using their Wi-Fi networks. This is a simple example to show that there is a need to decouple the three previously mentioned aspects and identify users for each one of the three separate from each other. For example, let's consider a public restaurant that provides access to the Internet. Users should have their own accounts that distinguish them from each other. Future implementations of such

separation may help achieving (open Wi-Fi) for public services where users who have Wi-Fi in their homes can open it for public services. However, there are some challenges that SDN are expected to address, such as mapping login to traffic and traffic to authentication services, rate limiting, etc.

The Clome project described a migration process from home networking to the cloud [Nabi and Alvi, 2014]. Several advantages as well as challenges are presented in this migration. OpenFlow solution can make the transformation process faster and easier. Management tasks such as: Accounting, auditing, billing, etc. can be implemented in very flexible and customized based on SDN programmability.

4.8. *Privacy protection*

Protecting privacy is important because many security attacks (e.g. information disclosure, tampering, and non-repudiation) target users' private information. The impact of SDN on privacy is two-fold: On one hand, SDN, through its programmability and flexibility features, can give ISPs the ability to customize privacy and control services to different users based on their preferences [Stallings, 2013]. For example, users can provide their ISP with their preferences on what websites or service to permit, block, or log. On the other hand, such wealthy information can be used by ISPs for marketing purposes. The evolution of network technologies such as SDN is expected to isolate the dependency of policies on low level layers' information [Paterson, 2014]. This will eventually give ISPs the ability to track their customers based on high level applications, usage profile, traffic, data etc. While such information can provide rich wealthy information from commercial perspectives, on the other hand, it may cause significant privacy concerns. Cloud, ecommerce and health information systems are important network environments in which customers' or users' information is sensitive from privacy as well as cost perspectives [Thuemmler et al., 2013].

Network Address Translation (NAT) and Carrier Grade NAT (CGN) are techniques used to provide mass network services. They can hide internal hosts' identities from externals. NAT is used to allow many users to be able to use the same real or registered IP address over the Internet. It resolves IP addresses' conflicts through replacing unregistered IP addresses with registered IP ones. It is used in: Servers, routers, firewalls, etc. where the device maintains a state table to translate unregistered to registered IP addresses. Packets are then translated from the unregistered address to the registered one or the opposite before moving inward or outward. Logging and tracing private addresses should be handled separately as typically those will not be available based on NAT. Similarly, CGN acts like a proxy to allow several users share one public IP address. It is offered as one of the solutions for IP address exhaustion. Typically it is implemented in mobile dedicated hardware. OpenFlow can offer a better solution to CGN with a flexible, robust and cost-effective approach [Donley, 2013]. Traffic can be also better monitored and managed from the different subscribers. [Olteanu and Raiciu, 2012] focused on a case study for using OpenFlow to isolate and distinguish traffic from different hosts in CGN for stateful network processing purposes.

[Mendonca et al., 2012] introduced AnonyFlow; an OpenFlow based "in-network" anonymization service. AnonyFlow is claimed to cause less network overhead in comparison with other approaches. AnonyFlow is claimed to be able also to perform intra-domain anonymity. Users' privacy protection through the Internet can be a significant safe guard from many security attacks targeting users and their private information. Primary target for this approach is end-point logging (e.g. from third parties). IP addresses are translated to anonymity IDs that other (i.e. destination) parties can only see. AnonyFlow is responsible for the translation between those IDs and hosts or IP addresses.

[Kopsel and Woesner 2011; Kotronis et al., 2013; Suñé et al., 2014] discussed privacy issues in SDN testbeds. They proposed a privacy and availability layer in those testbeds to act as a proxy for managing the connections with remote users. Networks of different users can be separated using slicing techniques (e.g. FlowVisor).

[Khan et al., 2013] focused on P2P traffics detection and privacy protection based on OpenFlow switches. P2P traffic is usually characterized by two things. First, large files are exchanged in those networks and consequently a large bandwidth is required. Second, users vary frequently which makes privacy and anonymity critical issues.

4.9. *Security controls of BYOD*

Bring Your Own Device (BYOD) is widely popular in companies, schools, or public places. BYOD is about users who have their own: Laptops, tablets, or smart phones. They want to use those devices for both business and home or personal purposes. There is no clear distinction inside those devices of what is personal and what is for business activities. Accounts in emails, social networks, etc. are usually used for both purposes. Currently, enterprises struggle to find the best way to handle dealing with BYOD situation. Preventing such devices is not any more feasible given that all users have one or more of those devices with them all the time and given that preventing and monitoring users not to carry those small devices is hard to implement or enforce. Preventing users or asking them to dedicate different devices for work and personal usage is also hard to implement or guarantee. Finally, accepting and embracing the usage of those devices without any security control is very risky. The major security threat in BYOD is related to the data in the device and the risk that it can be hacked by intruders. A somewhat similar case is how to handle guests' access accounts to enterprise networks in companies, universities, hotels, airports, etc. Security and accountability are two major concerns for allowing users to access the network and its services. Preventing them to use the network is not a proper alternative. The commonality in both situations is the need for security controls that are very agile and temporal. Current security controls are static and do not have the ability to continuously screen and change policies based on real time scenarios.

An SDN based solution to BYOD by HP is implemented in Ballarat Grammar school in Australia [HP, 2013]. HP SDN Sentinel security solution is shown to provide a realistic access control solution for such dynamic situations. Security threats from users and devices were detected and countered in real time.

[Hand et al., 2013] proposed using SDN to handle security problems when dealing with BYOD. While proposed solution

may not be able to access and install security agents in users' mobile devices, however SDN based active security approaches can continuously monitor those mobile devices and disconnect them in case of any security concern.

SDN has some features that can make it a candidate to solve dealing with BYOD; however, SDN may have a scalability problem given the very frequent network and control update information required when using a large number of mobile devices [Awobuluyi, 2014]. [Shoji et al., 2014] introduced the concept: Bring You Own Network (BYON). The goal is to optimize mobile networks' resource utilization. A solution based on programmable network is presented. To avoid compromising security, network should align each user or device to a particular network slice. This requires fine-grained and dynamic access control to manage and keep tracking of those users and devices.

[Suresh et al., 2012; Schulz-Zander et al., 2014] argued that the existing solutions for programmable Wi-Fi networks depend only on client-side modification which may not handle situations such as BYOD. They proposed Odin to provide features that enterprise and service providers need to implement from their server side.

4.10. *Security control of open SDN labs*

SDN open labs or testbeds are open network labs where users internally or remotely are given access to perform their lab experiments using the network resources. Those open research labs were early motivators for SDN. This is because traditional switches and routers are rigid and vendor closed. Those traditional networking components do not give experimenters the ability and flexibility to try their own algorithms and test them on production networks. Currently, SDN makes it possible to have such open networking labs (e.g. GENI, OFELIA, PlanetLab, VINI, and G-Lab). However, security controls and mechanisms are still evolving in those labs and there are many open issues and serious concerns. A similar case to open labs is related to conducting locally networking experiments. Researchers in the networking or related areas may want to conduct experiments using their own machines. Conducting some experiments while connecting to the production network can cause a risk to the network. On the other hand, isolating them completely in a static manner may not allow them to use the network and its resources that they need for their routine tasks. A solution to security and access control in open labs should be easy to implement. The solution should be also easy to disable (e.g. when the experiment is completed, etc.). Further, monitoring and tracking for those devices should be simple to initiate or update and largely conducted automatically with little or no human intervention.

[Kleban et al., 2013] demonstrated that security controls in open labs are unique and has no current "off-the-shelf" solution. In those labs, there is a concern of confidentiality problem not only from externals but also from internals. Network virtualization methods that allow different users in those systems to share the same physical or network resources while having different logical environments may fail in certain scenarios. The goal of security testing in those environments is then to evaluate how much such vulnerable scenarios are real or how often they can happen.

Open labs may not be necessary open for public. They can be a company or vendor open lab which can be used for cloud data centres [DeCusatis et al., 2013]. In this specific paper [DeCusatis et al., 2013], an SDN based lab is proposed for re-provisioning and reuse in cloud datacentres.

Open labs can be illegally accessed using spoofed identities. Users may try to change their identity to get legitimate access to those labs. Some papers evaluated security vulnerabilities in those Open or public labs [Siaterlis and Masera, 2009; Li and Hong, 2011; Li et al., 2011]. Spoofing is inevitable as those networks require a pre-authorization process in which many users may not qualify. Typically, two way certificates are constructed to ensure that only authorized users can access Open labs through encrypted channels. A user who has access to one node can, theoretically access and intrude experiments in other nodes. This may harm the integrity of the results of the experiments. If a node in the Open labs is infected with malicious code, this may also spread to hosts of users who are conducting experiments. A user who has an access to such network may run a sniffing tool to get network information about connected users. Sniffing the information of both IP and MAC addresses is an important asset to conduct ARP cache poisoning attacks.

[Li et al., 2011] discussed quality requirements in Open labs. In terms of security and accountability, several security control mechanisms and vulnerabilities are discussed. Several challenges that make security control mechanisms on those Open labs very difficult are described. One of the serious challenges in those Open labs is that ownership of resources, users and groups is distributed in a complex unaccountable manner. Further, experiments are conducted through the Internet with a large amount of information or data exchanged. A threat model is proposed of three layers: Control framework and administration, slices and experiments and finally the Internet or outsiders. Typically, different experiments are conducted on different slices. While slices can share the same physical architecture, each slice should be allocated its own virtual: Memory, switches, topology, etc. Several types of security attacks such as: DoS, spoofing, cache poisoning and flooding are investigated and how such attacks can be mitigated. Fears also exist where legitimate users may intentionally or unintentionally spread attacks through the lab. For example, when their own machines are possibly victims of botnets or worms and without their knowledge they could be contributing to an attack. Breaking the isolation between the different slices or experiments may cause security problems or may risk the integrity of the experiments.

[Moraes et al., 2014] proposed FITS; a secured and flexible architecture for Open labs based on OpenFlow networks. FITS (Future Internet Open labs with Security) provides low cost smart cards for authenticating users. TLS encryption is also used over the communication channel. The other security features adopted in the Open labs include: Strong slices' isolation of the four main resources (i.e. network, topology, bandwidth and memory or forwarding tables) and also VPN-based interconnections. FITS uses open source Xen for network virtualization. To ensure virtualization and isolation between the different experiments a VLAN tag is inserted in each packet.

4.11. SDN Vs classical security controls

In Section 4 we described different types of security controls and how those controls are going to evolve based on SDN. Some security controls (e.g. firewalls, IDS/IPS, access controls) are known in classical networks. However the functionality of those controls will change based on SDN specially as those controls will be able to dynamically interact with the underlying networks in real time and hence can make proactive or responsive decisions in response to network changes, needs, attacks, etc. For example, one of the most significant problems in classical firewalls is related to how to configure/reconfigure and maintain firewall rules or access control lists. Typically a large company firewall can include a very large number of firewall rules. Different rules can have different target access entry/exits (e.g. in/out: ports, IP addresses, MAC addresses). In classical networks rules are written manually by network administrators and can be only evaluated, or updated manually also by network administrators or users. No tools can effectively exist to screen out frequently rules for possible conflicts or obsoleteness. Rule conflicts are typically solved by short-cut solutions such as: first match or priority based rules overriding. All those examples of problems in classical firewalls are expected to be solved in SDN firewalls. Ultimately an SDN firewall should be fully programmable and autonomous in a sense that it can adjust its rules to accommodate network changes, new threats or attacks, etc. While research is already going in this direction, it is acknowledged that reaching the goal of such fully programmable firewalls requires solving several hard problems or challenges. From a security perspective and sense control decisions are going to be taken by unattended firewalls, risks should be assessed that such decisions are not going to cause a serious network change, failure, etc. Safe and static rules should also exist to take controls if a monitoring system decides that firewall decisions are not *realistic*.

The amount of information collected from the network is tremendous. SDN opens the possibility of exposing and using such information. Research in artificial intelligent, data mining, etc. should take place to evaluate best methods to use in making decisions, in real time, related to several hard problems. For example those hard problems include whether a certain traffic is a threat or not, whether a certain topology change is necessary or not, what security counter measure to take and how to select one control measure over the others based on impact, cost leverage, etc. While all such types of problems are hard in terms of the amount of information and variables to collect, time is very critical where it is important to make real time decisions or else decisions can be invaluable if they are late.

In the second category of SDN controls, we described several examples of security controls that were not popular or used in traditional networks and where we believe that they will be used to a significant extent in SDN (e.g. security for: BYOD, BYON, testbeds, home networking). Those types of security controls or architectures become viable with the evolving SDN architecture. They become also viable with the evolution of the networks, Internet, etc. By having the network and its traffic controllable by software applications, security controls can be more flexible and agile. Their rivals (i.e. attacks and threats) are also expected to evolve to better challenge the new architecture and the new security controls. For example, recent years showed a significant rise in smart phones or Wi-Fi attacks. Such rise is expected to continuously grow especially as SDN is expected to spread to wireless, mobile, etc.

Table 2 summarizes research focus in security controls in SDN environment and how they are going to evolve in comparison with security controls in classical networks.

5. Conclusion

We have presented an overview of the existing research in SDN security, focusing on security threats, and security controls. It is important to note that the landscape of SDN security changes with the advances in SDN research and development. For instance, a new protocol or API introduced to SDN may incur particular security threats and thus require specific countermeasures.

To conclude this paper, we discuss several SDN security issues and research topics. While they suggest directions of further research, they are by no means an exhaustive list.

Insider threats: Insiders often have more privileges particularly when they have access to the controller modules or resources in SDN. Several insider intrusions in SDN are studied in [Juba et al., 2013; Popa et al., 2010; Shin and Gu, 2012; Duncan et al., 2012; He et al., 2014]. In virtualized SDN environments, security attacks can propagate intentionally or unintentionally from within the same physical network. A compromised VM can escalate problems to other VMs especially as they run on the same physical elements. Security measures should be continuously evaluated to ensure that logically isolated tenants sharing the same physical network are completely isolated from each other. Compromising the controller resources is another type of insider threats. Applications interacting with the controller through controller APIs can be used as back doors. Given that the controller has tremendous privileges, such attacks can cause serious network damages.

Virtual attacks: In principle, a new virtual network can be established without following a certain network topology or IP addressing space. Such black or *dark network* can exist without being physically noticed. Special vulnerability assessment tools should be developed to evaluate how likely such scenarios can be real for a particular network. In SDN, migration of hosts should be implemented automatically in addition to accompanied tasks such as triggering the proper ports or network topology elements. While this is considered a significant advantage, if implemented improperly, it may imply serious security risks.

Security-embedded routing: SDN makes it possible to embed security in traffic flows that helps transport traffic in a secure manner. It goes beyond the concept of security aware routing used to direct internal or external traffic for security checking or auditing [Shin and Gu, 2012]. In security-embedded routing, a routing protocol can act as a carrier that helps not only in guiding the traffic but also in protecting it. Similar to telecommunication carriers, for example, a routing protocol can be multiplexed or modulated with the

Table 2 – SDN security controls research progress.

Security control	Research evolution trends
Firewalls	Dynamical allocations of firewalls [Sherry et al., 2012; Gibb et al., 2012; Gember et al., 2013; Mysore et al., 2013], SDN-Based Firewalls [Hu et al., 2014a,b], [Jia and Wang, 2013], [Suh et al., 2014] [Suh et al., 2014], Stateful firewalls access [Shirali-Shahreza and Ganjali, 2013a,b], [Katta et al., 2012], [Zhu et al., 2014], [Stoenescue et al., 2013], [Fayaz and Sekar, 2014], Hybrid SDN-classical firewall issues [Pan et al., 2013], [Gamayunov et al., 2013], [Shin et al., 2013a,b], [Hand et al., 2013]
Access Control	SDN-dynamic access control [Casado et al., 2009], [Nayak et al., 2009], Fine grained access control [Wen et al., 2013], SDN VN [Kinoshita et al., 2012], [Yamasaki et al., 2011], [Wu et al., 2013]
IDS/IPS	Integration with classical tools [Chung et al., 2013a,b], [Xing et al., 2013], [Shin et al., 2013a,b], SDN IDS/IPS implementation [Goodney et al., 2010], [Kerner, 2012], [Heorhiadi et al., 2012], [Skowyra et al., 2013a,b], [Giotis et al., 2014], Applications [Braga et al., 2010], [Mehdi et al., 2011], [Shirali-Shahreza and Ganjali, 2013a,b]
Policy Management	SDN policy languages [Hinrichs et al. (2008, 2009)], [Ballard et al., 2010], [Foster et al., 2011], [Gude et al., 2008], [Voellmy and Hudak, 2011], [Monsanto et al., 2012], [Voellmy et al., 2012], [Katta et al., 2012], [Foster et al., 2013], [Anderson et al., 2014], Migration from classical network [Vanbever et al., 2013; Vanbever et al., 2014; Zhang et al., 2014], policy enforcement [Bellessa et al., 2011], [Fayazbakhsh et al., 2013], [Qazi et al., 2013], [Kazemian et al., 2013], [Bari et al., 2013]
Monitoring and Auditing	Traffic monitoring tools [Nayak et al., 2009], [Ballard et al., 2010], [Huang et al., 2011], [Jose et al., 2011], [Shin and Gu, 2012], [Argyropoulos et al., 2012], [Yu et al., 2013], [Karame, 2013], [Shirali-Shahreza and Ganjali, 2013a,b], [Raumer et al., 2014], Traffic management [Curtis et al., 2011a,b], [Jain et al., 2013], [Wang et al., 2013], [Sun et al., 2014], [Choi et al., 2014a,b], [Rasley et al., 2014]
Mobile Security Control	[Li et al., 2012], [Gember et al., 2012], [Hampel et al., 2013], [Namal et al., 2013], [Skowyra et al., 2013a,b], [Ding et al., 2014], [Hurel et al., 2014], [Liyanage et al., 2014],
Wi-Fi Networks	[Feamster et al., 2004; [McKeown et al., 2008], Mundada et al., 2009; Nayak et al., 2009; Ramachandran et al., 2009; Anwer et al., 2010; Voellmy et al., 2010; Feamster et al., 2010, 2013a,b; Koponen, 2011; Voellmy et al., 2012], [Suresh et al., 2012; Kang et al., 2013; Pentikousis et al., 2013; Dangovas and Kuliesius, 2014]
Privacy Protection	[Mendonca et al., 2012], [Stallings, 2013], [Thuemmler et al., 2013], [Donley, 2013], [Suñé et al., 2014]
Security Controls of BYOD	[HP, 2013], [Hand et al., 2013], [Awobuluyi, 2014], [Shoji et al., 2014]
Security Control of Open Labs	[Kopsel and Woesner, 2011], [Li et al., 2011], [Kleban et al., 2013], [DeCusatis et al., 2013], [Kotronis et al., 2013], [Suñé et al., 2014], [Moraes et al., 2014]

traffic in a way that makes it not readable in transit. When it reaches the destination, a demodulation process can demodulate or decrypt the traffic at the destination premises.

Policy life cycle: SDN offers a potential for automatic implementation of policy life cycle activities. The major problem with policy management is related to the gap between low level mechanisms and high level user requirements. Policies need to be richer in context than low level firewall rules in traditional networks. They need to be more expressive and comprehensive to cover a wide range of possible packets. They should be managed by the controller and supporting modules, and accessible and extendable by administrators. Policies should be evaluated automatically with high levels of performance, reliability and scalability. While the existing research has discussed some of these issues, full support of policy life cycle remains to be seen.

On-demand security services: One of the potentials of SDN is enabling Internet or cloud service providers to provide customized on-demand security services. Customers may decide the details of security services that they want from those service providers. An inventory of security services can be provided where customers can select from. Not only customers can select to opt-in or out those security services, they can also decide their parameters. For example, an ISP may provide a website blocker or parent control service. Customers can select to enable it at a certain time, for a certain period or for a particular host or user. They may also decide the nature of the websites they want to allow or block. Those can be available on their accounts and they can frequently view and/or change.

Application access control: Network security controls permit/deny traffic based on network level information (i.e. IP, MAC addresses, port number or protocol). A limitation is that many applications can't be prevented from using the network without denying them based on their host. Most security attacks compromise certain applications and it is impractical to deny the host completely. SDN global policies can have the ability to perform access control based on two levels of information: User/host and switch/network. A central access control module can be developed as part of the controller to keep tracking of the information from those levels and consequently permit/deny traffic. Such fine-grained access control system can update access control information dynamically.

Internet security check points: One evolution that SDN may bring to the Internet is the transference from IP to flow based traffic management. This can enable the implementation of Internet check points. Many use cases have the need to check security threats in flows through the communication channels, not only at source and destination premises. Such

central check points can be a first defense layer on national or enterprise gateways. Countries, states, and companies at large may decide to have a central security gateway to screen certain traffic coming to or going from their premises. An inventory of on-demand security services can be offered by those security check points. Sample security services are border control, pinholing (e.g. timed open ports), translation services (e.g., IPv4-IPv6 exchanging), QoS marking and verification, and traffic metering.

REFERENCES

Al-Shabibi, Ali. POX Wiki, Stanford University, [Online]. Available: https://openflow.stanford.edu/display/ONL/POX+Wiki, last edited by Murphy McCauley on Aug 11, 2014.

Anderson Carolyn Jane, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, et al. NetkAT: semantic foundations for networks. POPL; 2014. p. 113–26.

Andersen David G, Hari Balakrishnan, Feamster Nick, Teemu Koponen, Daekyeong Moon, Scott Shenker. Accountable internet protocol (AIP). In: SIGCOMM'08; August 17–22, 2008. Seattle, Washington, USA.

Anwer Muhammad Bilal, Motiwala M, Tariq M, Feamster N. Switchblade: a platform for rapid deployment of network protocols on programmable hardware. ACM SIGCOMM Comput Commun Rev 2010;40(4):183–94.

Argyropoulos Christos, Dimitrios Kalogeras, Georgios Androulidakis, Vasilis Maglaris. PaFloMon - a slice aware passive flow monitoring framework for OpenFlow enabled experimental facilities. EWSDN 2012:97–102.

Awobuluyi Olatunde. Periodic control update overheads in openflow-based Enterprise networks. In: IEEE 28th International Conference on Advanced Information Networking and Applications; 2014.

Bakshi Kapil. Considerations for software defined networking (SDN): approaches and use cases. In: Aerospace Conference. IEEE; 2013. p. 1–9.

Ballard Jeffrey, Ian Rae, Aditya Akella. Extensible and scalable network monitoring using OpenSAFE. In: Proceedings of the 2010 Internet Net- work Management Conference on Research on Enterprise Networking, ser. INM/WREN'10. Berkeley, CA, USA: USENIX Association; 2010.

Bari Md Faizul, Shihabur Rahman Chowdhury, Reaz Ahmed, Raouf Boutaba. PolicyCop: an autonomic QoS policy enforcement framework for software defined networks. In: Software defined networks for future networks and services (SDN4FNS); 2013.

Bellessa John, Evan Kroske, Reza Farivar, Mirko Montanari, Kevin Larson, Roy Campbell. NetODESSA: dynamic policy enforcement in cloud networks. In: Proceedings of the 2011 IEEE 30th Symposium on Reliable Distributed Systems Workshops (SRDSW'11); 2011.

Benton Kevin, Jean Camp, Chris Small. OpenFlow vulnerability assessment. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM; 2013. p. 151–2.

Bifulco Roberto, Karame Ghassan. Towards a richer set of services in software-defined networks. In: SENT'14; 23 February 2014. San Diego, CA, USA.

Braga Rodrigo, Edjard Mota, Alexandre Passito. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: Proceedings of the IEEE Conference on Local Computer Networks (LCN), Denver, CO, USA, 11–14 October 2010; 2010. p. 408–15.

Casado Martin, Tal Garfinkel, Aditya Akella, Freedman MJ, Boneh D, Nick McKeown, et al. SANE: a protection architecture for enterprise networks. In: Proceedings of the 15th conference on USENIX Security Symposium. ser. USENIX-SS'06, Berkeley, CA, USA, vol. 15; 2006.

Casado Martin, Michael Freedman, Justin Pettit, Jianying Luo, Nick McKeown, Scott Shenker. Ethane: taking control of the enterprise. ACM SIGCOMM Comput Commun Rev 2007;37(4):1–12. ACM.

Casado Martin, Michael Freedman, Justin Pettit, Jianying Luo, Natasha Gude, Nick McKeown, et al. Rethinking enterprise network control. IEEE/ACM transactions on Networking (TON) 2009;17(4):1270–83. August 2009.

Choi Taesang, Song Sejun, Park Hyungbae, Yoon Sangsik, Yang Sunhee. SUMA: software-defined unified monitoring agent for SDN. NOMS; 2014a. p. 1–5.

Choi Taesang, Saehoon Kang, Sangsik Yoon, Sunhee Yang, Sejun Song, Hyungbae Park. SuVMF: software-defined unified virtual monitoring function for SDN-based large-scale networks. CFI; 2014b.

Chowdhury Shihabur Rahman, Bari Md Faizul, Ahmed Reaz, Boutaba Raouf. PayLess: a low cost network monitoring framework for Software defined Networks. NOMS; 2014. p. 1–9.

Chu YuHunag, MinChi Tseng, YaoTing Chen, YuChie Chou, YanRen Chen. A novel design for future on-demand service and security. In: 12th IEEE International Conference on Communication Technology (ICCT); 2010.

Chung Chun-Jen, Khatkar Pankaj, Xing Tianyi, Lee Jeongkeun, Huang Dijiang. NICE: network intrusion detection and countermeasure selection in virtual network systems. IEEE Trans Dependable Secure Comput - TDSC 2013a;10(4).

Chung Chun-Jen, Cui JingSong, Khatkar Pankaj, Huang Dijiang. Non-intrusive process-based monitoring system to mitigate and prevent VM vulnerability explorations. In: 9th IEEE International Conference on Collaborative Computing Networking Applications and Worksharing (CollaborateCom 2013); 2013.

Clark Russ, Feamster Nick, Nayak Ankur, Reimers Alex. Pushing Enterprise security down the network Stack. GT-CS-09–03. Georgia Institute of Technology; 2009. Tech. Rep.

Curtis Andrew, Mogul Jeffrey, Tourrilhes Jean, Yalagandula Praveen, Sharma Puneet, Banerjee Sujata. DevoFlow: scaling flow management for high-performance networks. SIGCOMM Comput Commun Rev 2011a;41(4):254–65.

Curtis Andrew, Kim W, Yalagandula P. Mahout: low-overhead datacenter traffic management using end-host-based elephant detection. In: IEEE INFOCOM'11; 2011.

Dangovas Vainius, Kuliesius Feliksas. SDN-driven authentication and access control system. In: The International Conference on Digital Information, Networking, and Wireless Communications (DINWC); 2014. p. 20–3.

DeCusatis C, Haley M, Bundy T, Cannistra R, Wallner R, Parraga J, et al. Dynamic, software-defined service provider network infrastructure and cloud drivers for SDN adoption. In: IEEE International Conference on Communications 2013: IEEE ICC'13–2nd Workshop on Clouds. Networks and Data Centers; 2013.

Dillon C, Berkelaar Michael. OpenFlow (D)DoS mitigation. Technical report. February 9, 2014., http://www.delaat.net/rp/2013-2014/p42/report.pdf.

Ding Aaron Yi, Jon Crowcroft, Sasu Tarkoma, Hannu Flinck. Software defined networking for security enhancement in wireless Mobile networks. Elsevier computer networks (COMNET), vol. 66; 2014.

Donley Chris. Leveraging openflow in DOCSIS® networks. CableLabs; 2013.

Dover Jeremy. A denial of service attack against the open floodlight SDN controller. Dover Networks LLC; Dec 2013. Research report.
Duncan Adrian, Creese Sadie, Goldsmith Michael. Insider attacks in cloud computing. In: TrustCom; 2012.
Fayaz Seyed, Vyas Sekar. In: Testing stateful and dynamic data planes with FlowTest HotSDN'14; August 22, 2014. Chicago, IL, USA.
Fayaz Seyed, Sekar Vyas, Yu M, Mogul J. FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions. In: Proceedings of the Second Workshop on Hot Topics in Software Defined Networks. ACM; 2013.
Feamster Nick, Hari Balakrishnan, Jennifer Rexford, Aman Shaikh, van der Merwe Jacobus. The case for separating routing from routers. In: Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture, ser. FDNA'04. New York, NY, USA: ACM; 2004. p. 5–12.
Feamster Nick, Nayak Ankur, Kim Hyojoon, Clark Russel, Mundada Yogesh, Ramachandran Anirudh, et al. Decoupling policy from configuration in campus and enterprise networks. In: Local and Metropolitan Area Networks (LANMAN), 17th IEEE Workshop on. IEEE; 2010.
Feamster Nick, Jennifer Rexford, Ellen Zegura. The road to SDN: an intellectual history of programmable networks. Technical Report. Princeton, NJ, USA: Princeton University; 2013a.
Feamster Nick, Rexford Jennifer, Shenker Scott, Clark Russel, Hutchins Ron, Levin Dave, et al. SDX: a software-defined internet exchange, ONS '13 research track, Apr. 2013b.
Feng Tao, Bi Jun, Hu Hongyu, Yao Guang, Xiao Peiyao. InSAVO: Intra-AS IP source address validation solution with OpenRouter. In: INFOCOM2012; 2012.
Feng Yifu, Dongqi Wang, Bencheng Zhang. Research on the active DDoS filtering algorithm based on IP flow. In: 2009 Fifth International Conference on Natural Computation. IEEE; 2009. p. 628–32.
Ferguson Andrew, Arjun Guha, Chen Liang, Rodrigo Fonseca, Shriram Krishnamurthi. Hierarchical policies for software defined networks. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ser. HotSDN'12. New York, NY, USA: ACM; 2012. p. 37–42.
Foster Nate, Rob Harrison, Michael Freedman, Christopher Monsanto, Jennifer Rexford, Alec Story, et al. Frenetic: a network programming language. In: ICFP; Sep 2011.
Foster Nate, Arjun Guha, Mark Reitblatt, Alec Story, Michael Freedman, Naga Praveen Katta, et al. Languages for software-defined networks. IEEE Commun Mag 2013;51(2):128–34.
Gamayunov Dennis, Platonov Ivan, Smeliansky Ruslan. Toward network access control with software-defined networking. In: SDNFW; 2013.
Gember Aaron, Grandl Robert, Junaid Khalid, Shen Shan-Hsiang. Towards software-defined middlebox networking. In: HotNets'12; 2012.
Gember Aaron, Grandl Robert, Junaid Khalid, Akella Aditya. Design and implementation of a framework for software-defined middlebox networking. In: SIGCOMM; 2013. p. 467–8.
Gibb Gibb, Hongyi Zeng, Nick McKeown. Outsourcing network functionality. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ser. HotSDN'12. New York, NY, USA: ACM; 2012. p. 73–8.
Giotis K, Argyropoulos C, Androulidakis G, Kalogeras D, Maglaris V. Combining OpenFlow and sFlow for an effective and scalable Anomaly detection and mitigation mechanism on SDN Environments. Comput Netw 2014;62:122–36.
Goodney Andrew, Narayan Shailesh, Bhandwalkar Vivek, Cho Young. Pattern based packet filtering using NetFPGA in DETER infrastructure. In: 1st Asia NetFPGA developers workshop; Daejeon, Korea; June 14, 2010.
Gude Natasha, Koponen Teemu, Pettit Justin, Pfaff Ben, Casado Martin, McKeown Nick, et al. NOX: towards an operating system for networks. Comput Commun Rev 2008;38(3):105–10.
Hampel Georg, Steiner Moritz, Bu Tian. Applying software-defined networking to the telecom domain. In: Proceedings of the 16th IEEE Global Internet Symposium in Conjunction with IEEE Infocom, Turin, Italy; April 2013.
Hand Ryan, Michael Ton, Eric Keller. Active security. In: Twelfth ACM Workshop on Hot Topics in Networks (HotNets-XII), College Park, MD; November 2013.
He Jin, Mianxiong Dong, Kaoru Ota, Mingyu Fan, Guangwei Wang. NetSecCC: a scalable and fault-tolerant architecture without outsourcing cloud network security. 2014. CoRR abs/1405.0660.
Heorhiadi Victor, Reiter Michael, Sekar Vyas. New opportunities for load balancing in network-wide intrusion detection systems. CoNEXT 2012:361–72.
Hinrichs Timothy, Natasha Gude, Martin Casado, John Mitchel, Scott Shenker. Expressing and enforcing flow-based network security policies. Technical report. University of Chicago; 2008.
Hinrichs Timothy, Natasha Gude, Martin Casado, John Mitchel, Scott Shenker. Practical declarative network management. In: Proceedings of the 1st ACM Workshop on Research on Enterprise Networking, WREN '09. New York, NY, USA: ACM; 2009. p. 1–10.
Howard Michael, LeBlanc David. Writing secure code. Redmond, WA: Microsoft Press; 2001.
HP. Ballarat Grammar secures BYOD with HP Software defined network and Sentinel SDN security application. 2013. http://h17007.www1.hp.com/docs/byod/Ballarat%20Grammar%20v9.pdf.
Hu Hongxin, Han Wonkyu, Ahn Gail-Joon and Zhao Ziming. FlowGuard: building robust firewalls for software-defined networks, HotSDN'14, August 22, 2014a, Chicago, IL, USA.
Hu Hongxin, Gail-Joon Ahn, Wonkyu Han, Ziming Zhao. Towards a reliable SDN firewall. ONS; 2014b.
Huang Guanyao, Chuah Chen-Nee, Raza Saqib, Seetharaman Srini. Dynamic measurement-aware routing in practice. Netw IEEE 2011;25(3):29–34.
Huang Wun-Yuan, Hu Jen-Wei, Chou Ta-Yuan, Liu Te-Lung. Design and implementation of real-time flow viewer across different domains. In: Advanced Information Networking and Applications Workshops (WAINA), 27th International Conference on; 2013. p. 619–24.
Huang Wun-Yuan, Chou Ta-Yuan, Hu Jen-Wei, Liu Te-Lung. Automatical end to end topology discovery and flow viewer on SDN. In: 28th International Conference on Advanced Information Networking and Applications Workshops; 2014.
Hurel Gaëtan, Rémi Badonnel, Abdelkader Lahmadi, Olivier Festor. Outsourcing Mobile security in the cloud. Lect Notes Comput Sci 2014;8508:69–73.
ISO. Information processing systems – open systems interconnection – basic reference model – part 2: security architecture. 1989. ISO 7498–1.
Jafarian Jafar Haadi, Ehab Al-Shaer, Qi Duan. Openflow random host mutation: transparent moving target defense using software defined networking. In: Proceedings of the ACM Workshop on Hot Topics in Software Defined Networks (HotSDN), Helsinki, Finland; 13–17 August 2012. p. 127–32.
Jain Sushant, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, et al. B4: experience with a globally-deployed software defined WAN. In: Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Hong Kong, China; 13–17 August 2013. p. 3–14.

Jia Xinpei, Wang J. Distributed firewall for P2P network in data center. In: ICCE-China Workshop (ICCE-China). IEEE; 2013. p. 15–9.
Jose Lavanya, Yu Minlan, Rexford Jennifer. Online measurement of large traffic aggregates on commodity switches. In: Proc. of the USENIX HotICE Workshop; 2011.
Juba Yutaka, Hung-Hsuan Huang, Kyoji Kawagoe. Dynamic isolation of network devices using OpenFlow for keeping LAN secure from intra-LAN attack. In: 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013. Procedia Computer Science, 22; 2013. p. 810–9.
Kang Joon-Myung, Hadi Bannazadeh, Hesam Rahimi, Thomas Lin, Mohammad Faraji, Alberto Leon-Garcia. In: IEEE International Conference on Communications Workshops (ICC). Software-Defined Infrastructure and the Future Central Office; 9–13 June 2013.
Kampanakis Panos, Perros Harry, Beyene Tsegereda. SDN-based solutions for moving target defense network protection. In: IEEE SDN Architecture and Applications. Sydney: Australia; June 16, 2014.
Karame Ghassan. Towards trustworthy network measurements. In: TRUST; 2013. p. 83–91.
Kotronis Vasileios, Schatzmann Dominik, Ager Bernhard. On Bringing Private Traffic into Public SDN Testbeds. HotSDN'13. August 16, 2013. Hong Kong, China.
Katta Naga Praven, Jennifer Rexford, David Walker. Logic programming for software-defined networks. In: ACM SIGPLAN Workshop on Cross- Model Language Design and mplementation, ser. XLDI; 2012.
Kazemian Peyman, Chang Michael, Zeng Hongyi, Varghese George, McKeown Nick, Whyte Scott. Real time network policy checking using header space analysis. NSDI; 2013.
Kerner Sean Michael. OpenFlow can provide security, too. May 14, 2012. http://www.enterprisenetworkingplanet.com/datacenter/openflow-can-provide-security-too.html.
Khan Hassan, Khayam Sayed, Golubchik Leana, Rajarajan M, Michael Orr. Wirespeed, Privacy-preserving P2P traffic detection on commodity switches. SPS; 2013.
Kim Hyojoon, Feamster Nick. Improving network management with software defined networking. Commun Mag IEEE 2013;51(2):114–9.
Kinoshita Shunichi, Toshiki Watanabe, Junichi Yamato, Hideaki Goto, Hideaki Sone. Implementation and evaluation of an OpenFlow-based access control system for wireless LAN roaming. In: Computer Software and Applications Conference Workshops (COMPSACW), IEEE 36th Annual. IEEE; 2012. p. 82–7.
Kleban Janusz, Marc Bruyere, Clegg Richard G, Landa Raul, Grzegorz Danilewicz, Janusz Kleban, et al. Abstraction layer for implementation of extensions in programmable networks, collaborative project co-funded by the European commission within the seventh framework Programme. 2013.
Kloeti Rowan, Vasileios Kotronis, Paul Smith. OpenFlow: a security analysis. In: Proceedings of the 8th Workshop on Secure Network Protocols (NPSec), Part of IEEE ICNP, Göttingen, German; October 2013.
Koponen Teemu, Scott Shenker, Hari Balakrishnan, Nick Feamster, Igor Ganichev, Ali Ghodsi, et al. Architecting for innovation. Comput Commun Rev 2011;41(3):24–36.
Kopsel Andreas, Woesner Hagen. OFELIA - Pan-European test facility for OpenFlow experimentation. ServiceWave 2011:311–2.
Kordalewski Dave, Robere Robert. A dynamic algorithm for loop detection in software defined networks. Technical report. Fall 2012.
Kreutz Diego, Fernando Ramos, Paulo Verissimo. Towards secure and dependable software-defined networks. In: Proceedings of the second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM; 2013. p. 55–60. HotSDN'13, August 16, Hong Kong, China.
Liyanage Madhusanka, Ylianttila Mika, Gurtov Andrei. Securing the control channel of software-defined mobile networks. In: Proc. of 1st IEEE WoWMoM Workshop on Software Defined Networking Architecture and Applications. Sydney, Australia; June 2014.
Li Erran, Morley Mao, Jennifer Rexford. Toward software-defined cellular networks. In: Software Defined Networking (EWSDN), 2012 European Workshop on; 2012. p. 7–12.
Li Dawei, Hong Xiaoyan, Jason Bowman. Evaluation of security vulnerabilities by using ProtoGENI as a launchpad. Houston, USA: IEEE Globecom; Dec. 2011.
Matias Jon, Borja Tornero, Alaitz Mendiola, Jacob Eduardo, Nerea Toledo. Implementing layer 2 network virtualization using OpenFlow: challenges and solutions. In: European Workshop on Software Defined Networking; 2012.
McKeown Nick, Anderson Tom, Balakrishnan Hari, Parulkar Guru, Peterson Larry, Rexford Jennifer, et al. Openflow: enabling innovation in campus networks. ACM SIGCOMM Comput Commun Rev 2008;38(2):p.69–74.
Mehdi Sayed Akbar, Khalid Junaid, Ali Khayam Sayed. Revisiting traffic anomaly detection using software defined networking. In: Recent advances in intrusion detection. Springer; 2011. p. 161–80.
Mendonca Marc, Seetharaman S, Obraczka K. A flexible in-network IP anonymization service. In: The IEEE ICC Workshop on Software Defined Networks; 2012.
Meyer Christopher, Schwenk Jörg. Lessons learned from previous SSL/TLS attacks - a brief chronology of attacks and weaknesses. IACR Cryptology ePrint Archive. 2013.
Monsanto Christopher, Foster Nate, Harrison Rob, Walker David. A compiler and run-time system for network programming languages. In: Principles of Programming Languages (POPL); 2012.
Moraes Igor, Diogo Mattos, Lyno Ferraz, Elias Campista, Marcelo Rubinstein, Luís Costa, et al. FITS: a flexible virtual network testbed architecture. Comput Netw 2014;63:221–37.
Mundada Yogesh, Sherwood Rob, Feamster Nick. In: An openflow switch element for click, in symposium on click modular Router. Citeseer; 2009.
Mysore Radhika, Porter George, Vahdat Amin. FasTrak: enabling express lanes in multi-tenant data. In: CoNEXT'13; December 9-12, 2013. Santa Barbara, California, USA.
Nabi Zubair, Alvi Atif. Clome: the practical implications of a cloud-based smart Home. 2014. CoRR abs/1405.0047.
Namal Suneth, Ahmad Ijaz, Gurtov Andrei, Ylianttila Mika. Enabling secure mobility with OpenFlow. In: Proc. IEEE software defined networks for future networks and services (SDN4FNS); November 11-13, 2013. Trento, Italy.
Naous Jad, Ryan Stutsman, David Mazieres, Nick McKeown, Nickolai Zeldovich. Delegating network security with more information. In: Proceedings of the 1st ACM Workshop on Research on Enterprise Networking, ser. WREN'09. New York, NY, USA: ACM; 2009. p. 19–26.
Nayak Ankur, Reimers Alex, Feamster Nick, Clark Russ. Resonance: inference-based dynamic access control for enterprise networks. In: Proceedings of the Workshop on Research on Enterprise Networking (WREN), Barcelona, Spain; 21 August 2009. p. 11–8.
Olteanu Vladimir, Raiciu Costin. Efficiently migrating stateful middleboxes. In: SIGCOMM'12; August 13–17, 2012. Helsinki, Finland.
Othman Othman, Okamura Koji. Securing distributed control of software defined networks. IJCSNS Int J Comput Sci Netw Secur September 2013;13(9).
Pan Heng, Guan Hongtao, Liu Junjie, Ding Wanfu, Lin Chengyong, Xie Gaogang. FlowAdapter: enable flexible multi-table processing on legacy hardware. HotSDN; 2013. p. 85–90.

Paterson Nancy. End user privacy and policy-based networking. J information Policy 2014;4:28–43.
Pentikousis Kostas, Wang Yan, Hu Weihua. MobileFlow: toward software- defined mobile networks. Commun Mag IEEE 2013;51(7):44–53.
Popa Lucian, Yu Minlan, Ion Stoica Steven, Sylvia Ratnasamy. CloudPolice: taking access control out of the network. In: Proceedings of the 9th ACM Workshop on Hot Topics in Networks (HotNets); 2010.
Porras Phillip, Shin Seungwon, Vinod Yegneswaran, Martin Fong, Mabry Tyson, Guofei Gu. A security enforcement Kernel for OpenFlow networks. In: HOTSDN; 2012.
Porras Phillip, Shin Seungwon, Vinod Yegneswaran, Martin Fong, Mabry Tyson, Guofei Gu. A framework for enabling security controls in OpenFlow networks. ACM; 2012b.
Qazi Zafar, Cheng-Chun Tu, Luis Chiang, Rui Miao, Vyas Sekar, Minlan Yu. SIMPLE-fying middlebox policy enforcement using SDN. In: ACM SIGCOMM; August 2013.
Ramachandran Anirudh, Mundada Yogesh, Bin Tariq Mukarram, Feamster Nick. Securing enterprise networks using traffic tainting. Technical Report GTCS-09–15. Atlanta, GA: Georgia Institute of Technology; 2009. Tech. Rep.
Rasley Jeff, Stephens Brent, Dixon Colin, Rozner Eric, Felter Wes, Agarwal Kanak, et al. Low-latency network monitoring via oversubscribed port mirroring. ONS; 2014.
Raumer Daniel, Schwaighofer Lukas, Carle George. MonSamp: a distributed SDN application for QoS monitoring. In: 1st Workshop on Software-defined Networking (SDN'14), Warsaw, Poland; 2014.
Ristenpart Thomas, Tromer Eran, Shacham Hovav, Savage Stefan. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proc. ACM CCS; 2009. p. 199–212.
Rothenberg Christian Esteve, Nascimento Marcelo Ribeiro, Salvador Marcos Rogerio, Correa Carlos, de Lucena Sidney, Raszuk Robert. Revisiting routing control platforms with the eyes and muscles of software-defined networking. In: HotSDN; 2012.
Schehlmann Lisa, Harald Baier. COFFEE: a concept based on OpenFlow to filter and erase events of botnet activity at high-speed nodes. In: INFORMATIK; 2013.
Schulz-Zander Julius, Nadi Sarrar, Stefan Schmid. AeroFlux: a near-Sighted controller architecture for software-defined wireless networks. ONS; 2014.
Sethi Divjyot, Narayana Srinivas, Malik Sharad. Abstractions for model checking sdn controllers. In: FMCAD; 2013.
Sherry Justine, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, Vyas Sekar. Making middleboxes someone Else's problem: network processing as a cloud service. In: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication; 2012.
Shin Seungwon, Porras Phillip, Yegneswaran Vinod, Fong Martin, Gu Guofei, Tyson Mabry. FRESCO: modular composable security services for software-defined networks. In: NDSS; 2013.
Shin Seungwon, Gu Guofei. Cloudwatcher: network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?). In: Proceedings of the 20th IEEE International Conference on Network Protocols (ICNP), ser. ICNP'12. Washington, DC, USA. IEEE Computer Society; 2012. p. 1–6.
Shin Seungwon, Yegneswaran Vinod, Porras Phillip, Gu Guofei. Avant-guard: scalable and vigilant switch flow management in software-defined networks. In: Proceedings of the ACM Conference on Computer and Communications Security, ser. CCS'13. ACM; 2013b.
Shin Seungwon, Gu Guofei. Attacking software defined networks: a first feasibility study. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking; 2013. p. 165–6.
Shirali-Shahreza and Ganjali Sajad Yashar. Efficient implementation of security applications in OpenFlow controller with FleXam, HotSDN'13, August 16, 2013a, Hong Kong, China.
Shirali-Shahreza, Ganjali Sajad Yashar. FleXam: flexible sampling extension for monitoring and security applications in OpenFlow, HotSDN'13, August 16, 2013b, Hong Kong, China.
Siaterlis Christos, Masera Marcelo. A review of available software for the creation of testbeds for internet security research. In: International Conference on Advances in System Simulation - SIMUL; 2009.
Skowyra Rick, Bahargam Sanaz, Bestavros Azer. Software-defined IDS for securing embedded Mobile devices. In: Proceedings of HPEC'13: The IEEE high Performance Extreme Computing Conference, Waltham, MA, September 2013a.
Skowyra Rick, Lapets Andrei, Bestavros Azer, Kfoury Assaf. Verifiably-safe software-defined networks for CPS. In: Proceedings of HiCoNS'13: The ACM International Conference on High Confidence Networked Systems. Philadelphia, Pennsylvania: CPS week; May 2013b.
Shoji Yozo, Manabu Ito, Kiyohide Nakauchi, Lei Zhong, Yoshinori Kitatsuji, Hidetoshi Yokota. Mobility management in the networks of the future world. In: IEEE Consumer Communications and Networking Conference (CCNC 2014); 2014.
Son Sooel, Shin Seungwon, Yegneswaran Vinod, Porras Phllip, Gu Guofei. Model checking invariant security properties in OpenFlow. In: IEEE International Conference on Communications (ICC), Budapest, Hungary; June, 2013.
Song Haoyu. Protocol oblivious forwarding: unleash the power of SDN through a future-proof forwarding plane. In: Proceedings of HotSDN; 2013.
Sperotto Anna, Schaffrath Gregor, Sadre Ramin, Morariu Cristian, Pras Aiko, Stiller Burkhard. An overview of IP flow-based intrusion detection. IEEE Commun Surv TUTORIALS 2010;12(3). THIRD QUARTER.
Stallings William. Software-defined networks and OpenFlow. Internet Protoc J 2013;16(1).
Stoenescu Radu, Popovici Matei, Negreanu Lorina, Raiciu Costin. SymNet: static checking for stateful networks. In: HotMiddlebox'13; December 9, 2013. Santa Barbara, CA, USA.
Suh Junho, Choi Hoon-gyu, Yoon Wonjun, You Taewan Ted, Kwon Taekyoung, Choi Yanghee. Implementation of content-oriented networking architecture (CONA): a focus on DDoS countermeasure. In: Proc of 1st European NetFPGA Developers Workshop, Cambridge, UK; Sep. 2010.
Suh Michelle, Park Sae Hyong, Lee Byungjoon, Yang Sunhee. Building firewall over the software-defined network controller. In: February 16~19, ICACT2014; 2014.
Sun Peng, Yu Minlan, Michael J Freedman, Rexford Jennifer, Walker David. HONE: joint host-network traffic management in software-defined networks. J Netw Syst Manag (JNSM) April 2015;23(2):374–99.
Suñé Marc, Bergesio Leonardo, Woesner Hagen, Rothe Tom, Köpsel Andreas, Colle Didier, Puype Bart, Simeonidou Dimitra, Nejabati Reza, Channegowda Mayur, Kind Mario, Dietz Thomas, Autenrieth Achim, Kotronis Vasileios, Salvadori Elio, Salsano Stefano, Körner Marc, Sharma Sachin. Design and implementation of the OFELIA FP7 facility: The European OpenFlow testbed. Comput Netw 2010;61:132–50.
Suresh Lalith, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, Teresa Vazao. Towards programmable Enterprise WLANs with Odin. In: Proc. HotSDN; 2012. Helsinki, Finland.
Thuemmler Christoph, Müller Julius, Covaci Stefan, Magedanz Thomas, De Panfilis Stefano, Jell Thomas, et al.

Applying the software-to-data paradigm in next generation e-health hybrid clouds. ITNG; 2013. p. 459–63.
Vanbever Laurent, Stefano Vissicchio. Enabling SDN in old school networks with software-controlled Routing Protocols. In: Open network Summit (Research track). Santa Clara, CA, USA; March 2014.
Vanbever Laurent, Reich Joshua, Benson Theophilus, Foster Nate, Rexford Jennifer. HotSwap: correct and efficient controller upgrades for software-defined networks. In: ACM SIGCOMM HotSDN 2013 Workshop. Hong Kong, China; August 2013.
Voellmy Andreas, Kim Hyujoon, Feamster Nick. Procera: a language for high-level reactive network control. In: Proceedings of the 1st Workshop on hot topics in software defined networks; August 2012. p. 43–8.
Voellmy Andreas, Agarwal Ashish, Hudak Paul, Feamster Nick, Burnett Sam, Launchbury John. Don't configure the network, program it!, Domainspecific programming languages for network systems. Tech. Rep. YALEU/DCS/RR-1432. Yale University; July 2010.
Voellmy Andreas, Paul Hudak. Nettle: functional reactive programming of OpenFlow networks. In: PADL; Jan 2011.
Wang Haopei, Xu Lei, Gu Guofei. Of-guard: a DoS attack prevention extension in software-defined networks. ONS; 2014.
Wang Juan, Yong Wang, Hongxin Hu, Qingxin Sun, He Shi, Longjie Zeng. Towards a security-enhanced firewall application for OpenFlow networks. CSS; 2013. p. 92–103.
Wang Xiang, Liu Zhi, Qi Yaxuan, Li Jun. LiveCloud: A Lucid Orchestrator for Cloud Datacenters. In: 2012 IEEE 4th International Conference on Cloud Computing Technology and Science; 2012.
Wen Xitao, Chen Yan, Hu Chengchen, Shi Chao, Wang Yi. HotSDN, towards a secure controller platform for openflow applications. In: HotSDN; 2013.
Wu Yong-juan, Liang Jun-xue, Zhang Hua, Lin Zhao-wen, Yan Ma, Zhong Tian. Programmable virtual network instantiation in IaaS cloud based on SDN. J China Universities Posts Telecommun 2013;20(Suppl 1):121–5.
Xiao Peiyao, Jun Bi, Tao Feng. O-CPF: an OpenFlow based intra-AS source address validation application. In: CFI'13, June 05 – 07; 2013. Beijing, China.
Xing Tianyi, Huang Dijiang, Xu Le, Chung Chun-Jen, Khatkar Pankaj. SnortFlow: a openflow-based intrusion prevention system in cloud environment. In: Second GENI Research and Educational Experiment Workshop; 2013.
Yamasaki Yashuiro, Miyamoto Yoshinori, Yamato Junichi, Hideaki Goto, Hideaki Sone. Flexible access management system for campus VLAN based on OpenFlow. In: Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on. IEEE; 2011. p. 347–51.
Yao Guang, Bi Jun, Xiao Peiyao. Source address validation solution with OpenFlow/NOX architecture. In: Proceedings of the IEEE International Conference on Network Protocols (ICNP), Vancouver, BC, Canada; 17–20 October 2011. p. 7–12.
Yao Guang, Bi Jun, Tao Feng, Xiao Peiyao, Duanqi Zhou. Performing software defined route-based IP spoofing filtering with SEFA. In: ICCCN2014; 2014.
Yap Kok-Kiong, Sherwood Rob, Kobayashi Masayoshi, Huang Te, Chan Michael, Handigol Nikhil, et al. Blueprint for introducing innovation into wireless mobile networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures; 2010. p. 25–32.
Yap Kok-Kiong, Kobayashi Masayoshi, Underhill David, Seetharaman Srinivasan, Kazemian Peyman, McKeown Nick. The Stanford OpenRoads deployment. In: WiNTECH'09: Proceedings of the Fourth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization. ACM; 2009.
Yap Kok-Kiong, Yiakoumis Yiannis, Kobayashi Masayoshi, Katti Sachin, Parulkar Guru, McKeown Nick. Separating authentication, access and accounting: a case study with OpenWiFi. Open Networking Foundation; 2011. Tech. Rep.
Yu Curtis, Lumezanu Cristian, Zhang Yueping, Singh Vishal, Jiang Guofei, Madhyastha Harsha. FlowSense: monitoring network utilization with zero measurement cost. In: Passive and active measurement. Springer; 2013. p. 31–41.
YuHunag Chu, Tseng Min-Chi, Cen YaoTing, Chou YuChieh, Chen YanRen. A novel design for future on-demand service and security. In: Proceedings of the International Conference on Communication Technology (ICCT), Nanjing, China; 11–14 November 2010. p. 385–8.
Yuzawa Tamihiro. OpenFlow 1.0 actual use-case: RTBH of DDoS traffic while keeping the target. 2013. Online, http://packetpushers.net/openflow-1-0-actual-use-case-rtbh-of-ddos-traffic-while-keeping-the-target-online.
Zaalouk Adel, Rahamatullah Khondoker, Ronald Marx, Kpatcha Bayarou. OrchSec: an orchestrator-based architecture for enhancing network-security using Network Monitoring and SDN control functions. In: NOMS; 2014. p. 1–9.
Zhang Shuyuan, Laurent Vanbever, Sharad Malik. In-band update for network routing policy migration. In: IEEE ICNP 2014 (Concise papers track). Raleigh, North Carolina, USA; October 2014.
Zhang Yinqian, Juels Ari, Reiter Michael K, Ristenpart Thomas. Cross-VM side channels and their use to extract private keys. In: ACM Conference on Computer and Communications Security; 2012. p. 305–16.
Zhu Shuyong, Bi Jun, Sun Chen. SFA: stateful forwarding abstraction in SDN data plane. USENIX/Open Networking Summit Research Track (ONS14), Santa Clara, USA 2014.

Izzat Alsmadi: An associate professor in software engineering. He is currently working as an assistant research professor in the department of computer science at Boise State University (BSU). He is also in a leave from department of CIS at Yarmouk University, Jordan. He has his master and PhD in Software Engineering from North Dakota State University. His research interests include: Software security, software engineering, software testing, social networks and software defined networking.

Dianxiang Xu: Professor and graduate coordinator in the department of computer science at Boise State University (BSU). He joined Boise State University in August 2013. His prior teaching and research experience has included posts at Dakota State University, North Dakota State University, Texas A&M University, Florida International University, and Nanjing University. He is a senior member of the IEEE. His research interests include: Software security, software engineering, security policy, software testing, computer forensics and software defined networking.