Software Defined Networking: Attacks and Countermeasures

Nada Mostafa Abd Elazim
Computer Engineering Department.
Arab Academy for Science and Technology, College of Engineering
Cairo, Egypt
nadamostafa@aast.edu

Mohamed A. Sobh
Ain Shams University
Cairo, Egypt
mohamed.sobh@eng.asu.edu.eg

Ayman M. Bahaa-Eldin
Misr International University
On leave from Ain Shams University
ayman.bahaa@eng.asu.edu.eg

***Abstract*—Software defined networking (SDN) is an emerging network architecture; it differs from traditional networks as it separates control planes from data planes. This separation helps the network to be more flexible and easier to handle and allows faster innovation cycles at both planes. SDN has benefit over traditional networks in terms of simplicity, programmability and elasticity. Openflow protocol is a south-bound API interface; it is the most popular and common protocol that used to communicate the controller with the switches. This paper will focus on the architecture of SDN and provide some challenges faces the SDN; finally, it will discuss some security threats that face SDN and their countermeasures.**

***Index Terms*—SDN, Openflow, API interface**

I. INTRODUCTION

Traditional networks were very complex and difficult to manage. They combine the control plane with data plane that make network management difficult.

On the other hand, software defined networking (SDN) is a new networking approach to build computer networks that separates and abstracts elements of these systems to help building flexible and scalable network.

Advantages of Software defined networking (SDN) over traditional network [1]:

- It has virtual environment as it uses resources without caring about where it is located and how it is orderly.
- Monitor large number of devices by one command.
- Easy to change behaviour, size, and quantity.
- Minimize downtime, enforcement of policy, discover the faults and solve them, and add new devices, resources, sites, and workloads.
- Monitoring of resources.
- Improve the utilization of network device.
- The global vision of the network due to the centralization of the controller.

Openflow [2] is a protocol found in the southbound API interface that locates between the control and data forwarding layer. It is the way to virtualize the network. openflow is designed to be easy programmed, that helps the network manager to create new protocols for solving problems.

SDN has many applications in data centre, WAN, IoTs, cellular networks, and Wi-Fi network.

Security threats are on the rise, SDN faces many security threats in each of its layer, for example, in Data forwarding layer there are man at the end attack, DoS attack, spoofing attack, intrusion attack, scanning attack, tampering attack, hijacking attack, side channel attack, and anomaly attack. In control layer there are DoS/DDoS attack, intrusion attack, anomaly attack, threats based on distributed multi-controllers, threats from applications, packet-in attack, side channel attack, scanning attack, spoofing attack, hijacking attack, and tampering attack. In application layer there are intrusion attack, anomaly attack, illegal access, trust model, chained applications, altering SDN database, third party applications, misuse of resources, and security rules and configuration conflicts.

As software defined networking continue to increase in popularity as a flexible and dynamic approach to networking, it's going to need a flexible and dynamic approach to security. Some papers present some countermeasures for these threats and their authors try to solve them without facing overhead or performance problem in the network [34], [35], [36] and [37].

The remaining of the paper is orderly as follows. Section II discusses the architecture of SDN. Section III discusses some applications on SDN. Section IV illustrates some challenges of SDN. Section V illustrates the security threats for each SDN layer. Section VI presents some countermeasures for these security threats. Section VII Proposing a trust mechanism for software defined networking. Finally, we conclude in Section VIII.

II. ARCHITECTURE OF SDN

Software defined networking (SDN) [3, 5] consists of three layers: Application layer, control layer, and data forwarding layer from up to bottom. SDN consists of two

interfaces the first one between data forwarding layer and control layer called southbound APIs (e.g. openflow), and the second one between application layer and control layer called northbound APIs (e.g. REST) as shown in figure 1.

A. Application Layer:

Application layer is the top layer of SDN, it contains network applications that interact with network devices and manipulate the behaviour of them through the control layer. To combine applications with the underlying physical resources they need some requirements such as topology and link discovery, firewall services, domain name services, network address translation services, and deploying virtual private networks.

Application layer can compel its policies without interact directly with the data forwarding layer, through the northbound API that is supported by the control layer.

B. Northbound APIs:

It is the interface between application layer and control layer that provide universal data and functionality for network applications. There is no standard protocol for northbound SDN interface at present.

C. Control Layer:

Control layer is the middle layer of SDN; it is the heart of SDN, it is used to manage and control the network. it is consists of the SDN controller that communicates with the SDN switch through a southbound API such as openflow, the controller provide a global view of the network topology in the data forwarding layer, and also run different routing protocols like BGP and OSPF on SDN controller.

Openflow protocol support single controller for simplicity but has a problem of single point of failure, some architectures provide multiple distributed controllers such as Floodlight [31], NOX [32] and OpenDaylight [33], which communicate with each other through eastbound and westbound interfaces.

D. Southbound APIs:

It is the interface between the control layer and data forwarding layer, the most popular protocol in this interface is openflow protocol

E. Data Forwarding Layer:

Data forwarding layer is the bottom layer of SDN, it consists of number of SDN switches that connected through wired or wireless media. The main function of this layer is to forward the packets to the controller according to the assigned rules and policies provided by controller via southbound interface. Each SDN switch consists of forwarding table called flow table that contains many rules used to make forwarding decisions. Each rule in the flow table is consists of three fields: the action, the counter and the pattern. The action field is a particular rule that will be performed, the counter field is incremented by one if the rule is matched the fields, the pattern field defines the flow pattern that is the set of header field values of the packet.

If the rule is not matched, the switch will report the controller for help or discard the packet.

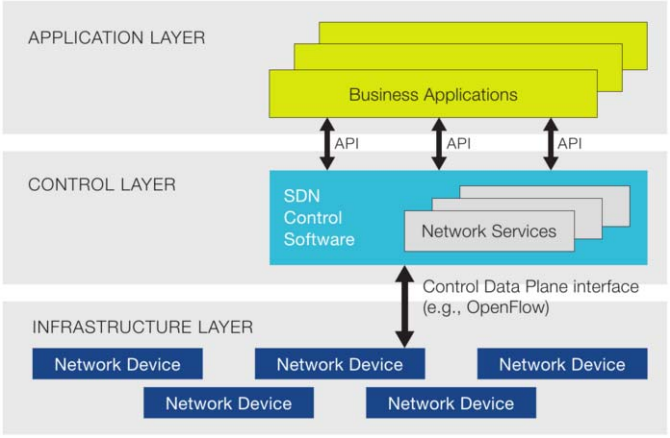

Fig.1. SDN Architecture

III. APPLICATIONS OF SDN

A. SDN in data center [5,6]:

Data center is used for storing information or data, its structure is so complex and difficult to manage. The data center in SDN has variable size based on the requirements. SDN can be able to upgrading the data centers without lost any information and insulting the overall performance.

B. SDN in WAN [5]:

Due to the lack of the network global view there is no forwarding decision of the traffic when a fail occurs in the network. SDN controller provides a central view of the network and calculates optimal path.

Google gained significant attention when using SDN in WAN for managing their data center. Google is able to gain benefits such as fault tolerance, active link utilization up to 100% promoting energy efficiency [6].

C. SDN for IOT [5]:

SDN offers large benefits to control the network in addition to flexibility, centralized control, abstraction, and scalability. Internet of things (IOTs) uses same features of SDN. Problems faced IOTs are large volume of data from many devices, devices heterogeneity and issues of the security.

IV. SECURITY CHALLENGES OF SDN

Security is very important aspect in software defined networking (SDN). SDN security needs to be built in architecture to provide availability, integrity and privacy of all elements and information. Also need to protect and secure the controller, establish trust between each component in the SDN architecture, check that the controller is doing what you want, and if there is a failure happens, the architecture must be able to identify, recover and report the problem and protect against it in the future.

Separation of the control and data planes opens the door for the security threats and increases the security

challenges in SDN. Optimal location of SDN controller, switches and other devices is an open problem in SDN that affects the overall network performance and security.

Due to the nature of SDN as it is flat, its integration is another security challenge where defense solutions and monitoring systems must be compatible to enhance the overall performance, network security, and energy saving [6].

SDN has a real challenge to address dynamic real time change and deployment of rules.

Security challenges are found in the three layers of the SDN architecture that are described below:

A. *Security challenges in data forwarding layer [6]:*

The flow tables in data plane have limited space, the storage of the flow entries in flow tables cause overhead on flow tables that results high cost and low performance. This challenge can be solved by using intelligent flow tables management methods to be able to store large number of rules with low cost and high performance [30].

One of the challenges is to recognize the true flow rules and differentiate them from false rules. Another challenge is based on the number of flow entries in the flow table of the switch.

SDN switches works on the rules generate by the SDN controller. They need high capacity processors to work efficiently. They need more power consumption if the switch includes more number of CPUs.

Configuration complexity and optional use of TLS (transport layer security) can cause different types of attacks.

B. *Security challenges in control layer [4]:*

SDN controller is the heart of SDN, but due to its centralized decision making it become a single point of failure that may cause network down in case of security compromise.

The control layer becomes attractive part for security attacks due to its visible environment.

Another challenge is that number of switches connected with the controller that send requests to the controller and waiting for reply, if the number of the switches increase the time that need for the controller to reply for a certain switch is increased, the load on the controller may cause controller failure.

Since the controller is responsible for the enforcement of the network-wide policy, the configuration conflicts can be done by multiple controllers in a single domain, the semantic gap of the controller-switch, and access control distribution.

Using multiple controllers is another challenge that faces the control layer to manage a large number of devices, these multiple controllers divide the network into multiple software defined sub-networks that have to aggregate the information and maintain different privacy rules in each sub-network.

DoS attack faces the controller that make the resource unavailable to legitimate users. When the number of flows increases in the datapath to the controller, it causes the controller to break.

C. *Security challenges in Application layer [4]:*

Applications can point to security threats to network services, resources and functions, because there is no standard or open specifications to make it easy for applications to control the services and functions of the network through the control layer [29].

Although openflow enables security detection algorithms in the form of security applications, there are no forcing openflow security applications.

V. SECURITY THREATSIN SDN

This section presents security threats in each SDN layer [3, 5, 6].

A. *Security attacks in application layer:*

The attacker is affected by the application layer; it can change the configuration of the network, gain access on network information, lock network resources and insert malicious computer programs. It affects the availability and reliability of the network.

Illegal access: done due to the vulnerabilities of software of the controller.

Security rules and configuration conflicts: due to the applications complexity conflicts may occur between security rules, resulting in network services confusion and management complexity.

Intrusion attack: the attacker gets access from the system without valid permissions and affects the security and stability of the network.

Anamoly attack: the attacker gets access of unauthorized users and injects malicious application, this attack can't be traced and difficult to detect.

Trust model: SDN security can't be compelled through physical infrastructure. If the applications are attacked the network must handle the consequences.

Chained applications: some applications have to run in chain, these nested applications can bypass access control mechanism leading to unauthorized access.

Altering SDN database: according to certain privileges for applications to run on top of SDN, they have access to the internal storage to manipulate and exploit it. This access can manipulate the network behaviour.

Third-party applications: the security issues arising from the controls of third party are interoperability problems and security policies conflict.

Misuse of resources: the resources of the system like memory and CPU can be exhausted by fraudulent applications. That leads to affect the applications performance and SDN controller.

B. Security attacks in Northbound API interface:

This interface is between the application and the control layers suffer from several attacks and security vulnerabilities such as:

Intrusion attack: the attacker gets access from the system without valid permissions and affects the security and stability of the network.

C. Security attacks in Control layer:

The control layer consists of controllers, if they are compromised, whole network may be affected. The controller becomes an attractive part for attackers. If the controller fails to do any task, so the packets will not be forwarded.

DoS/DDoS attacks: this attack happens when the attacker send a large amount of packet-in messages in a short period of time, that prevent legitimate users to use the network. This attack will consume the buffer memory spaces in network devices and reduce the network performance.

Threats on distributed multi-controller: due to the risk of single point of failure in a controller,that has shortage of scalability and reliability. To solve this problem it is proposed to use distributed control, at which each controller communicates with each other and with its switches. This distributed nature suffers from several security problems like authorization, authentication, and privacy issues during transmission of network information. Multi-controller architecture causes configuration conflicts and inconsistent configuration.

Packet-in attack: this attack happens when the switch sends fake packet-in messages that cause corruption of the controller.

Side-channel attack: this attack happens when the attacker have access on basis of information such as energy consumption and timing information.

Intrusion attack: the attacker gets access from the system without valid permissions and affects the security and stability of the network.

Anamoly attack: the attacker gets access of unauthorized users and injects malicious application, this attack can't be traced and difficult to detect.

Scanning attack: the attacker makes scan for the whole network to understand it; it can sniff information of the network through the controller.

Spoofing attack: this attack happens when the attacker create fake rules inside the flow tables inside the switches. It can gain a complete control of the network.

Hijacking attack: the attacker gains sensitive information such as passwords, communication data, etc. It can modify any information and change the traffic to any destination.

Tampering attack: the attacker wants to instantiate new flows at the controller either by spoofing northbound or southbound API messages, and allows traffic to pass across the SDN and bypass security policies.

D. Security attacks in Southbound API interface:

This interface is between the control and the data forwarding layer suffer from several attacks such as:

Man in the middle attack: this attack act as an agent node inserted between the controller and the switch, it is used to tamper and change the rules that forwarded to the flow table in the switch, and it also may insert malicious rules to the switch that affected the network.

Traffic analysis attack: an attacker can access sensitive information by sniffing the packets transferred to the switch.

Network monitoring: the attacker tries to observe the southbound interface and learn the data transfer between the switches and controllers.

E. Security attacks in data forwarding layer:

Data forwarding layer contains many of switches connected with each other, the function of this layer to forward packets. If a switch suffers from any attack, the packets that pass through it will not be forwarded correctly. Also attacker can attack the switch by attacking the interface linking to it. The switch consists of three functions called openflow client, flow table, and flow buffer. When a packet is received to the switch, the switch saves the packet in the flow buffer after that it searches the flow table if the rule is matched or not. If the rule is matched, the packet is removed from the flow buffer and forwarding to the output port. If the rule is not matched, so the switch send packet_in message through the openflow client to the controller to request for instructions. After that the controller adds the new rule in the flow table. These functions face many security attacks such as:

DoS Attack (saturation attack): denial of service attack facing flow table and flow buffer in the SDN switches. In flow table, an attackersends large number of packets in a short time to unknown hosts. These packets fill the flow table as the flow table has limited capacity. This attack makes the legal packet or traffic will not be forwarded correctly.

Flow buffer suffer also from denial of service attack, the packets first are placed in the flow buffer before forwarding them out. The flow buffer has limited storage capacity, the attacker tries to fill the flow buffer with large

amount of packets, after that the flow buffer didn't have enough space and any new packets can't be able to enter the flow buffer.

Man at the end attacks: an attacker can give a flow table of a victim switch fake topologies or make the controller in an unpredictable state.

Side-channel attacks: the attacker can determine the forwarding policy by knowing the packet processing time.

Spoofing attacks: the attacker uses legal user identity to insert fake packets and malicious applications into the network. The SDN switches can be spoofed and used for sending requests to controller in which it can't block these malicious packets due to the lack of middle boxes.

Intrusion attacks: the attacker gets access from the system without valid permissions and affects the security and stability of the network.

Anamoly attacks: the attacker gets access of unauthorized users and injects malicious application, this attack can't be traced and difficult to detect.

Scanning attack: the attacker can get access to the basic information of the whole network; it can sniff information of the network such as hosts' features, topology and communication details between switches and hosts.

Hijacking attack: the attacker takes control of the element of the communication. It can get access on communication data and all flow rules, if the switch is hijacked. It can impersonate a useful user and get communication data from other host, if any host is hijacked.

Tampering attack: the attacker modifies the network information, and inserts malicious flow rules.

VI. COUNTERMEASURES

This section discusses the countermeasures for each layer in the SDN

A. *Countermeasures for Application layer:*

PermOF [22]: is a permission system to limit the privilege on applications, it also discusses 18 permissions for applications to be enforced by the controller API entry. This paper proposes a customize isolation mechanism to provide permissions enforcement and separate the control traffic from the data traffic.

This permission system provides access control tokens to accept or deny any application. It has four permission categories, read permissions to read and manage any sensitive application information. Notification permissions notify of application for certain events. Write permissions have the ability to modify states either for controller or switches. System permissions provide access to the local resources for the application provided by operating system.The design of permission system depending on four aspects: thread models, openflow protocol control messages, applications functional requirements, and general API set of the controller implementations.

NICE[23]: is a tool for testing application of openflow. It tests the state space of the system such as the switches, the controller and the hosts.It augments model checking with symbolic execution of event handlers. Finally, it is used to detect bugs in programs of openflow, through a combination of symbolic execution and model checking.

The programmer can use NICE to check that there is no black holes or forwarding loops. NICE integrates symbolic execution, model checking and search strategies.

It is concluded that NICE when applying on small examples is five times faster. It has low overhead.

Vericon [25]: is used to verify the correctness of the controller program in software defined network on all topologies and all sequence of network events. It implements classical Floyd-Hoare-Dijkstra verification by using invariants of desired network wide and first order logic to assign permissible network topologies.

Vericon is implemented using Z3 and is applied to SDN programs.

Anteater [27]: is a tool to check invariants in data plane. Anteater translates high-level network invariants into instances of Boolean satisfiability problems (SAT). It uses SAT solver to check against network state, and if it is found violations it reports counterexamples.

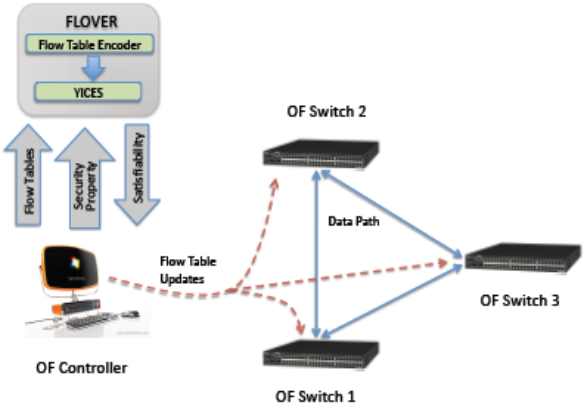

Fig 2.Flover Architecture

The goal of the anteater is to detect the problems in network by analyzing the forwarding table contents in routers, firewall, switches, and other equipment. Anteater is used by the operators to check if the network conforms to a group of invariants.

It is concluded that anteater is a system that detect real networks bugs via analysis of data plane. It gathers

information from network devices. Anteater improves the reliability of the network.

Flover [26]: is a model checking system that verifies the flow policies. It is executed as an openflow application that runs on an openflow controller. Flover verifies a group of specific non-bypass properties against the updated flow rule set. Flover consists of two parts as shown in figure 2, the first part is a flow table encoder that encodes group of flow rules, flow rule set and specified non-bypass properties into Yices code, the second part is Yices SMT solver that verifies if the non-bypass property carries on the encoded model.Flover supports two modes of execution: in-line mode the flover executes the validation of flow rule for each flow rule update, batch mode that checked periodically to improve the response time of the controller.

It is concluded that the performance of the flover can disclosure of amendment and violations coverage of the rule up to 200 rules in less than 120 ms and 131 ms.

NetPlumber [28]: is a checking tool policy for real time based on Header Space Analysis (HSA). It is used to check for state changes compliance. It is fit for SDNs.NetPlumber checks each event, if there is any violation, it makes alert to the user or block the change.

NetPlumber has faster update time than Hassel because it updates only the part of transfer function whose affected by the change.

NetPlumber checks the SDN policies as shown in figure 3. Netplumber agent is found between the switches and control plane, it sends any updates in the network to netplumber that performs the updates of the network. If any violation happens, netplumber take an action like removing the rule or announcing the administrator. Plumbing graph is the heart of netplumber that captures all flow paths through the network.

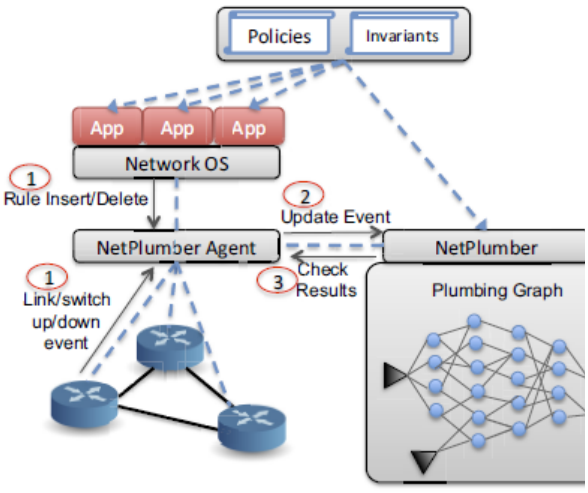

Fig 3. SDN policy checker

It is concluded that any update in the network in real time is fast by using NetPlumber.

B. Countermeasures for Control layer:

FloodGuard [15]: is a lightweight protocol that defense the SDN network from saturation (flooding) attack by using two modules: a proactive flow rule analyzer module, and a packet migration module as shown in figure 4.

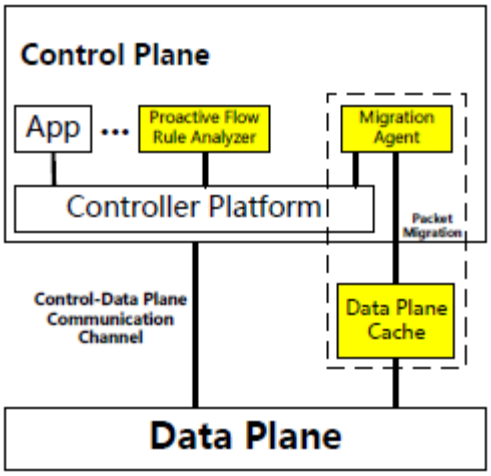

Fig 4.FloodGuard Architecture

The states of FloodGuard as shown in figure 5. starts initially from the idle state. If there is no attack, the proactive flow rule analyzer module and the packet migration module still idle. When detecting saturation attack, FloodGuard state changes to init state, and the migration agent component transfers the table miss packets to the data plane cache, and the proactive flow rule analyzer module will dynamically convert the path conditions and generate the proactive flow rule, at the same time, the data plane cache component deals with the cached packets and generates packet_in messages to the controller. When a proactive flow rules are generated, the floodGuard state changes to defense state, and the analyzer module installs these rules on the switches. When the attack is finished, the floodGuard state changes to finish state, and the migration agent component stops migrating the table miss packets, and the data plane cache continue processing the packets inside it until it finishes, then it will become idle again.

Proactive flow rule analyzer module found in the control layer above the controller platform, this analyzer module consists of three components as shown in figure 6: symbolic execution engine is a way to generate proactive rule, application tracker converts the path conditions to proactive flow rules dynamically and gets the runtime value, and proactive flow rule dispatcher. It

involves the symbolic execution and the dynamic application tracking to generate proactive flow rules in runtime. If the system detects saturation attack, it activates the analyzer module. The analyzer module generates the proactive flow rules dynamically, by combining symbolic execution with dynamical application tracking.

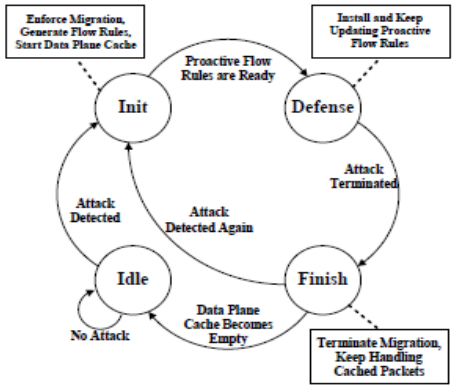

Fig 5.state of FloodGuard

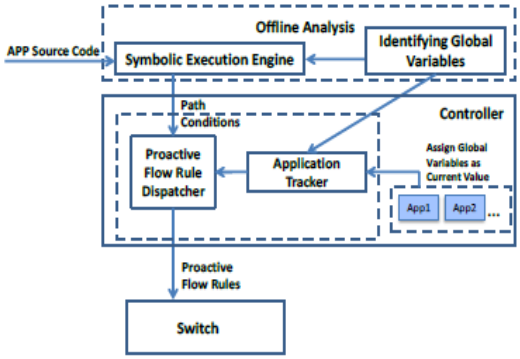

Fig 6.proactive flow rule analyzer

Packet migration module includes two components: migration agent and data plane cache. The migration agent locates in the control layer and the data plane cache locates between the data and control planes. The migration agent is the main component of the FloodGuard system. It uses to detect the flooding attack, by using both the rate of real time of packet-in messages and the infrastructure utilization that calculates capacity usage percentage. It sends table miss packets to the data plane cache when detecting DoS attack, then the data plane cache didn't send the packet in message directly to the controller it first sends it to the migration agent component and then this agent sends the packet_in events with the information of the original datapath to the controller.

The data plane cache temporarily stores all the table miss packets, it contains three functions: packet classifier that distributes the packet's header and links it to buffer queue. The packet buffer queue uses round robin scheduling algorithm to migrate the header packet to data plane and will drop any packet if it is full. The packet_in generator converts the accepted packets into packet_in messages and sends them to the controller.

It is concluded that when the attack rate increases, the bandwidth decreases. It protects the switches and the controller but suffers some time delay overhead.

DDoS Blocking Application (DBA) [16]: it is located in the SDN controller; it is used to solve DDoS attacks by blocking any malicious traffic, and differentiates the normal from malicious traffics.

The system architecture as shown in figure 7.consists of SDN controller, OpenFlow switches, number of normal users and number of bots, and protected service.

When a new client wants to enter a network, it sends a request to the switch, if this flow is mismatch in flow table, it send it back to the controller that take action to creates a flow entry of the new packet inside the flow table. When the server responds to the request from the client, another flow entry is created in the switch but in reverse direction.

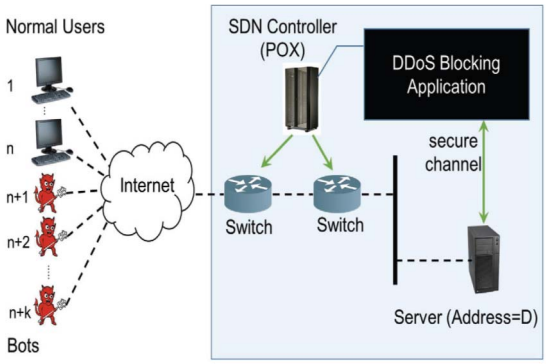

Fig 7.system architecture

DBA observes the number of flows in each switch, at the same time; the server observes metrics that indicates a possible DDoS attack. When the server discovers DDoS attack and it nearly to avalanche, it communicates with DBA through a secure channel, which provides the server with a new IP address D' at which the service continue working. The server opens a new socket with a new IP address for the service. So any client can use this new IP address as the new location for the service, and then drop the connection. In order to prevent bots to decode the message at address D', we use CAPTCHA to protect the communication.

When the server begins to receive the messages for incoming requests, the DBA orders the switch to accept flows from address D', and it increments the counter for this new destination, if the counter exceeds a given threshold it classifies it as a bot and drop the arriving packets. The client is said to be bot, if the repeated violations exceeds a given threshold, and all the flow entries create inside the flow table are deleted to avoid the overflowing of the flow table with unnecessary entries.

It is concluded that this proposed scheme in DBA regulates the defense using simple and few openflow interfaces.

The future work is to plan to minimize the dependences between SDN controller and the protected server through the pre-arranged cooperation, so that SDN can provide protection of servers inside it.

Content-oriented Networking Architecture (CONA) [17]: this architecture used to solve and reduce the security and accountability problems such as DDoS attack. The rationale behind using CONA that the user is interested in the content only rather than the location of the content.

A host uses the MAC address or a private address to communicate with its access router in CONA. The access router manages the hosts attached to its subnet. The publisher agent sends DNS query request, then the DNS send reply message containing the public IP address of the publisher's agent that wish to attach. When the host wants to set up internet connectivity, the agent sends an advertisement message to the host including the domain name and the certificate of the agent.

If the DDoS attack starts, the publisher agent detects firstly the requests toward the publisher. If the arrival rate of the requests exceed a certain threshold or its contents aren't normal so the agent detects DDoS attack. Then the agent sends notification message to the controller. Once the controller receives this notification; it performs two actions, firstly it tests the flow of the attacked publisher from the other agents, secondly, it sends to each agent connected to the attacked publisher a rate limit message to limit the rate of content request on to the other publisher. This limiting rate can be checked by the publisher easily and propagated it across ISP's boundaries in a daisy chain manner. Then, the traffic generated by DDoS attack can be contained by a back pressure mechanism.

It is concluded that in case of no limiting mechanism, the throughput of the normal host decreased when the content requests rate of the malicious host increased. In case of using limiting mechanism, the throughput of the normal host isn't affected when the normal host is attached to the same agent with the malicious host.

DISCO [18]: to solve the single point of failure, it is an open distributed SDN control plane for multi-domain SDN networks. DISCO architecture consists of two modules: intra-domain module and inter-domain module. Intra-domain module collects the main functions of the controller;it manages and monitors the network, this module reacts to any problem in the network dynamically to redirect any malicious traffic. Inter-domain module communicates with the controllers to exchange information, it consists of two modules: Messenger and agents. Messenger implements a channel between the controllers to exchange information and request action. It uses in communication open shortest path first protocol, resource reservation protocol and border gateway protocol. Messenger communication uses advanced message queuing protocol (AMQP) that offers features for orientation of the message, priority queuing, routing, reliable delivery and security.Agents are used to support QoS routing and reservation functions. It is used to exchange data between patterns.

McNettle [19]: is simple and highly scalable SDN controller framework supporting multi-core CPUs that build on OpenFlow switches to increase the throughput and decrease the latency of control servers with global visibility. To handle any demanding network control applications, these multicore architectures provide enough memory, IO and computational resources. It is used to monitor and track the traffic of the flow, and deal with any malicious attacks facing the traffic flow.

It is concluded that it has better performance. When the number of switches increases the aggregate throughput decreases over a fixed number of cores. When the number of switches more than 500 switches the latency deceases if the number of cores increases.

HyperFlow [20]: is a distributed control plane for OpenFlow, to recover the lack of scalability. It is physically distributed;it takes benefit of centralization of network control as it provides scalability. It centralizes decision making to individual controllers. The response time of the control plane is reduced to the requests of data plane.All controllers participate the same network and serve requests, finally decrease the flow setup times.

HyperFlow consists of NOX controllers that act as decision elements; OpenFlow switches that act as forwarding of data, and an event propagation system. Each switch connects to the suitable controller. If the controller is failed, so the switches connected to that controller have to reconfigure to connect to the near controller.HyperFlow uses publish/subscribe messaging model, to broadcast events of the controller to others. It uses publish/subscribe system that must provide storage of events, holds the events order published by the same controller, and operation of each partition continues independently. HyperFlow application made to make sure that all controllers have a whole view of the network. Each controller runs on the HyperFlow application.

It is concluded that HyperFlow decreases the creating and installing time of any new flow rules and improves the performance of the control layer; it also decreases the response time of control layer.

Future work is to perform Hyperflow on large testbed to further characterize its scalability, performance, and robustness.

FRESCO [21]: Is an openflow security application implemented for rapid design and modular composition for mitigation and detection modules.

FRESCO consists of library that includes 16 reusable modules, and each module consists of five interfaces: input, output, event, parameter and action. The developer may replicate security functions, such as: scan detectors, firewalls, IDS detection logic, or attack deflectors.

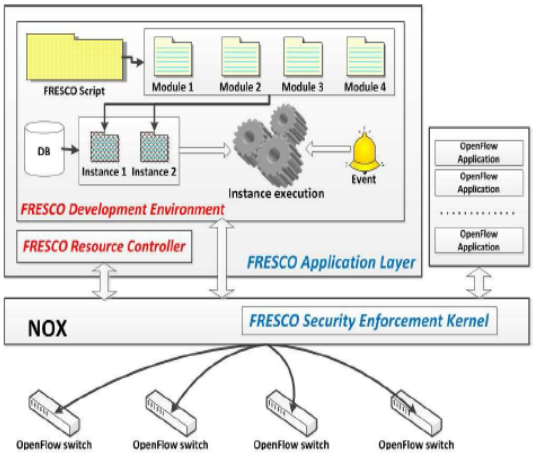

Fig 8. FRESCO Architecture

The framework of FRESCO consists of an application layer that implemented using NOX python modules, which provides two functions of key developers: resource controller and FRESCO development environment, and security enforcement kernel which enforces security applications, as shown in figure 8.

To create a security application from FRESCO internal modules, the FRESCO script language is used by the developers to determine the interactions between the security modules of the NOX python. These scripts accessed via DE database API of FRESCO. The FRESCO modules are executed by FRESCO DE; these modules create new flow rules in which controller's security enforcement kernel processed.

It is concluded that FRESCO has low overhead that enables security functions create rapidly. Also it offers a new powerful framework that prototypes and delivers security applications innovative into rapidly evolving world of software defined networks.

Future work is to release all developed code as open source software to the SDN community.

C. Countermeasures for Southbound API interface:

The most suitable solution to solve man in the middle attack is to use transport layer security (TLS) to secure the communication link between the controller and the switches. This security mechanism is optional in the later version of openflow specification.

Other alternative countermeasures are discussed in this part:

FlowChecker [8]: is a tool used torecognize any misconfiguration inside the switch in a FlowTable when the user writes any wrong or conflict rulesin the FlowTable or in the path that need number of FlowTables to be preserved at the same time. The FlowChecker is act as an independent centralized server application that receives queries from the OpenFlow applications to analyze or debug OpenFlow configuration. FlowChecker is act as a stand-alone service that communicates with the OpenFlow controllers through a secure channel over SSL. The controller communicates with the FlowChecker through a queries that specifies the topology, users, controllers IDs, and name binding.

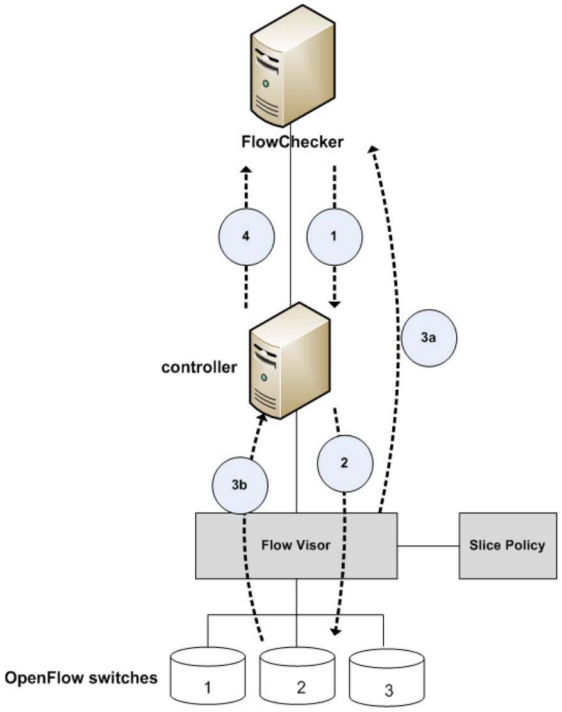

Fig 9.Flowchecker operation

The FlowChecker works as follows: for checking the validity of any properties, a user writes his query using CTL logic. After that the FlowChecker sends a request message to the controllers asking them to extract the flowtables from their switches and send them back to the FlowChecker. After the controller receives the request it sends it to the switches waiting for their reply through a FlowVisor that responds to the message by sending the slices policies for that domain to theFlowChecker. Finally the FlowChecker has slices policies comes from the FlowVisor and the FlowTables come from the switches respond, the validation is done by comparing slices polices with the flowTables and generates the result if there are conflicts and misconfiguration inside

FlowTables in the switches or not, and reply by CTL query as shown in figure 9.

The complexity to find if there is an inter-switch conflict between n OpenFlow switches is O (n^2).

It is concluded that the misconfigurations in flowtables can be done at run time.It is easier to write queries Using CTL language to extract statistics to be used for QoS analysis.

FortNOX [9]: is a software extension that provide authorization and authentication security enhancement. It enables NOX to check the conflicts of the flow rule in real time. FortNOX applies a rule conflict detection engine, this detection engine intermediates the requests of flow rule insertion. When a conflict is detected, FortNOX may accept or reject the new rule if the rule insertion requester operates with high security authorization.

FortNOX provides non-bypassable policy-based enforcement of the flow rule insertion requests from applications of openflow to expand the NOX openflow controller. Any rule is inserted by a security application to FortNOX it doesn't allow any peer to add flow rules that make conflict with these rules.

If the rules are from security applications that are privileged, the fortNOX stores these rules in the security constraints table. If the rules are from security application that are not privileged, fortNOX checks if a conflict occurs in security constraints table or not, if the conflict occurs an error message is sent to the openflow application, otherwise the rule is sent to the network switches.

VeriFlow [10]: is used to decrease the latency during the checks without any vulnerability in the network performance. VeriFlow is an intermediate layer between the controllers and switches that checks the dynamic violations in each rule in case of insertion,modification or deletion. VeriFlow makes tracking to any change in the forwarding state event, interceptsand verifies the rules before entering the network, and monitors all communication.To check for the rule if it is accepted or rejected, the new rule is entered in the VeriFlow that contains different phases;firstly, the network is sliced into set of equivalence classes, each equivalent class has a number of packets that have the same actions in the network, if there is any change in the network it affects only a small number of equivalent classes. Secondly, the VeriFlow generates forwarding graphs for every equivalent class; finally it converts this graph to query to run it to check every rule entering the network as shown in figure 10.

It is concluded that VeriFlow is the tool that makes verification of invariants in network-wide in real time, also it is able to update forwarding table in real time.

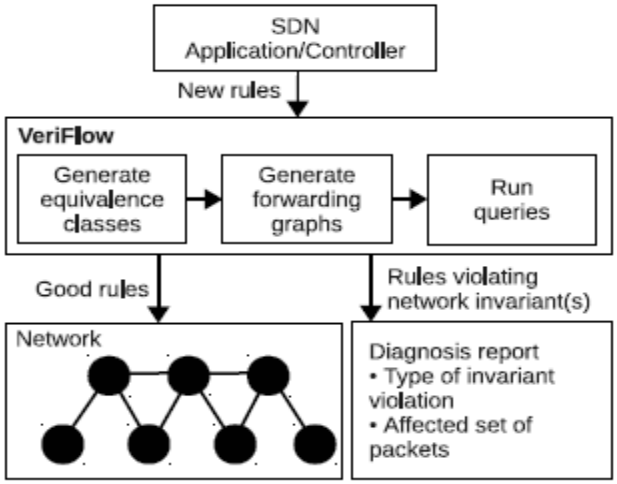

Fig 10.Veriflow architecture

Controller replication [11]: this technique focuses on single point of failure; this problem will compromise the functionality of the network. If any problem occurs in the network like the application enters an infinite loop, failure of the server, or an attack from another network. If a problem occurs in a primary controller then the switch will be connected to the backup or secondary controller to overcome any problem and complete the normal operation.

CPRecovery is used when there is a fail in the primary controller. Replication process is done between the switch that runs on the primary and secondary controller. In replication phase the following actions occur: the arrival of the packet_in event to the network OS that indicates the arrival of packet whose actions include in the flowtable of the switch. The network OS checks if the primary server is in the network, also verifies if the source table entry is related to the path that sent the packet. If the entry found in the source table, it examines a destination port, creates a flow for the packet to pass, connects this flow to actions and sends the packet. If there is no entry found in the source table, it creates a new source table entry connecting its MAC address to the path and port, then it verifies if secondary network OS is ready.If it is ready, the primary server generates a message contains the MAC address and datapath identification which insert in the source table. And send this message to the secondary network OS and waiting for acknowledgment message that confirm that the message of state update is arrived as shown in figure 11.

The experimental results for this technique will result increasing the delay during the change of the controller during the failure, the average delay will be 20 msec and

increased to 800 msec after the secondary controller take action on the network it quickly recover the delay and returns it to its normal state. Also the response time will be changed depending on the replication degrees, the response time increases by the increase in the replication degrees.

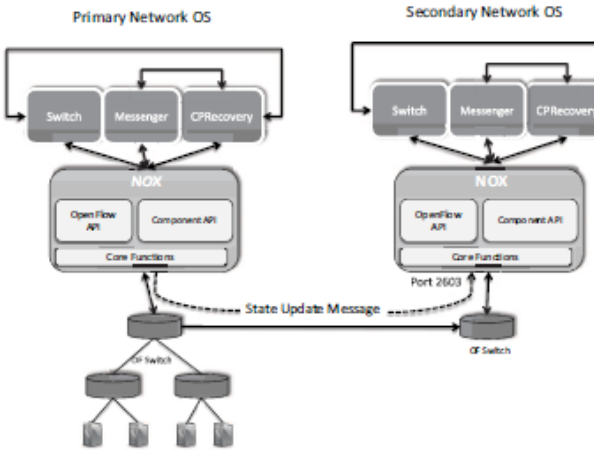

Fig 11. Controller replication

The future work of replication technique is to extend it other methods like a state machine approach, and to provide replication to other NOX components such as discovery, routing and topology.

D. Countermeasures for data forwarding layer:

FlowVisor [12]: is a network virtualization layer that lies between the switches and controller, it uses OpenFlow protocol to control the underlying physical network. FlowVisor supports multiple controllers on the same OpenFlow; it divides the network in to slices and each slice having its own set of flows running on switches, and each controller has its own slice. When a message sends from the controller to the switch it paths first to the FlowVisor to check the slice policy of the controller, if the slice policy is matched so the switch sends the message to the controller.

When the system affected by DoS attack, the FlowVisorrewrite the rule that comes from the controller to drop the UDP traffic caused DoS attack.

It is concluded that the FlowVisor makes overhead to the actions between the control and data forwarding layers.Also the FlowVisor increase the latency for new flow message from the switch to the controller by 16 msec, and for sensitive applications the latency will be 16 msec that will be much overhead over the system, without using FlowVisor the overhead will be 12 msec.

Virtual source Address Validation Edge (VAVE) [13]: VAVE over OpenFlow/NOX architecture is used to solve the source address validation problem that caused DoS attack.OpenFlow devices used as a protective area, when any packet enter this area which contain number of OpenFlow devices, it check its entry in the flow table of the device. If it isn't matched, it will be directed to the NOX controller that check if the source address of the packet is valid or not, if the source address is used forged source it will result to cut the spoofing flow and the entry will be removed if there is no packet matching.

VAVE has three additional modules in OpenFlow/NOX architecture such as filtering rule generator uses to generate rule to filter an flow paths come from illegal address space on VAVE interface, validation module uses to check if the packet is spoofing or not, and rule adaptor uses to make rule configuration on the OpenFlow device through interface supplied by NOX.

It is concluded that VAVE is very flexible and processing packet has low overhead compare with any other works. VAVE is used to increase the protectiveness of SAVI (source address validation improvement) devices that uses to prevent spoofing of SAVI-SAVI flow.

Resonance [14]: is a system to secure the network, where it enforces dynamic access control policies to the data forwarding layer based on real time alerts and packet flow information.The design of the Resonance has the following features: policy specification framework, distributed network monitoring, and the ability to take specific actions using switches.

The resonance works as follows: OpenFlow switches establish a secure communication with a controller. If new host want to enter and use the network, firstly it sends a DHCP discover message to whole elements in the network, after the switch connected to the host receives the DHCP packet, it forwards the packet to the controller through a secure channel. The controller registers the host to its database of hosts and establishes the flow table entry to help DHCP to communicate with the host. In the registration phase, the controller will drop any traffic isn't DHCP or ARP, unless the traffic is HTTP at which the controller installs an entry to redirect traffic that redirects the user to the authentication web site. In the operation phase, the host can be able to connect to any internet destination. If any alarm comes from the network to inform about an event, the controller can transfer the host to the authentication phase.

VII. TRUST MECHANISM FOR SOFTWARE DEFINED NETWORKING (SDN)

A. Criteria to calculate the trust values:

- Check the IP address of the source if it is sent large amount of packets that exceeds the threshold in small amount of time excludes it or block it.
- If the number of packets in the packet count field is so large compared with the others so mark it as malicious packet and remove it from the flow table.
- If the number of the packets in the packet count field is so small compared with the others so remove it from the flow table.
- An Echo request/reply message sent by switch or controller to check for latency and bandwidth, if the latency is high and the bandwidth is low so remove it.

B. The trust system model:

To make trust system model depending on the criteria that calculate the trust values, this can be done by discussing two models: average trust model and cumulative trust model.

Average trust model [7]:

This trust model is done by accumulate the trust values of a certain rule or node and divide them by the number of trust values.

$$AVG=\frac{T_n}{n} \quad (1)$$

Cumulative trust model [7]:

This trust model is done by accumulate the summation of a given trust values.

$$R_n = \sum_{i=1}^{n} r_i \quad (2)$$

C. Results:

It is noticed that when the experiment run without attack the number of packets drop is small and the number of packets received is small compared with the experiment run with attack. In the flow table, the duration between flow rules and the number of packets received are small compared with the experiment run with attack as shown in figure 12.

When the experiment run with attack, it is noticed that the number of packets drop become larger and the number of packets received are big. In the flow table, the duration between flow rules are too big and there is duplication in the rules as shown in figure 13.

VIII. FUTURE TRENDS

The future work of this paper is to search for trust mechanism for SDN to make authentication of the flows and the hosts and to secure them from any attacks like denial of service attack, man at the end attack, intrusion attack ...etc. that affect the network and make it behaves abnormally actions.

IX. CONCLUSION

This paper introduces a survey on software defined network (SDN). This survey includes applications and some security challenges for SDN. It surveyed many security attacks in each SDN layer and its countermeasures. Finally it introduces trust mechanism for SDN, illustrates the criteria to calculate the trust values and discusses different models for calculating the trust values.

	Flow statistics without attack					
	source_IP	in_port	duration	number of packets	number of bytes	action
S1	10.0.1.3	2	240.941	2	196	Output:3
	10.0.1.1	1	240.791	38	3724	Output:3
	10.0.1.2	3	240.763	37	3626	Output:1
	10.0.1.2	3	237.286	1	98	Output:2
	10.0.1.3	2	237.226	38	3724	Output:1
	10.0.1.1	1	237.205	38	3724	Output:2
	10.0.1.1	1	8.979	0	0	NORMAL
			248.141	44	3304	CONTROLLER:65535

Fig. 12. Flow rules of S1 without attack

	flow statistics with attack					
	source_IP	in_port	duration	number of packets	number of bytes	action
S1	10.0.1.3	2	106.139	1105	59846	Output:3
	10.0.1.2	3	104.039	1098	59468	Output:2
	10.0.1.1	1	80.215	2101	116754	Output:2
	10.0.1.1	1	80.215	4695	256830	Output:3
	10.0.1.2	3	80.215	4555	249270	Output:1
	10.0.1.3	2	69.386	1917	106818	Output:1
	10.0.1.1	1	7.749	0	0	NORMAL
			115.108	1310	70392	CONTROLLER:65535

Fig. 13. Flow rules of S1 with attack

REFERENCES

[1] Introduction to software defined networking (SDN). Washington university in saint Louis. http: //www.cse.wustl.edu/~jain/cse570-13/

[2] Prepare for software defined networking. Business white paper, Build the foundation for SDN with openflow.

[3] Jiafu wan, athanasiosvasilakos, Di Li, zhaogangshu, jiaxianglin, Muhammad imran, "Security in Software-Defined Networking: Threats and Countermeasures", Article in mobile networks and applications, Vol. 21, Issue 5, pp. 764-776, January 2016.

[4] Ijazahmed, sunethnamal, mikaylianttila, Andrei gurtov"Security in software defined networks: a survey", IEEE communication surveys and tutorials, VOL.17, NO.4, Fourth quarter 2015.

[5] Azaka, S Revathi, Angelina Geetha. “A survey of applications and security issues in software defined networking”, I.J. computer network and information security,VOL 9, issue 3, P. 21-28, 2017.
[6] Danda B. Rawat, Swetha R. Reddy,“Software defined networking architecture, security and energy efficiency: A Survey”, IEEE communications surveys and tutorials, Vol.19, NO.1, first quarter 2017.
[7] Zhuge Bin, Peng Dan, Bu Xiaobo, Wang weiming. “Research and implementation of the SDN resources transaction process based on trust mechanism”, China communications, Supplement No.1, Vol. 13, pp. 190-207, 2016.
[8] Al-Shaer E, Al-Haj S,“FlowChecker: configuration analysisand verification of federated OpenFlow infrastructures”,In: Proceedings of the 3rd ACM Workshop on Assurable and UsableSecurity Configuration, pp 37–44, 2010
[9] Porras P, Shin S, Yegneswaran V, Fong M, Tyson M, Gu G , “A security enforcement kernel for OpenFlow networks”, In: Proceedings of the First Workshop on Hot Topics in SoftwareDefined Networks, pp 121–126, 2012
[10] KhurshidA, Zhou W, Caesar M, Godfrey P,“Veriflow: verifyingnetwork-wide invariants in real time”, ACM SIGCOMMComputCommun Rev 42(4): 467–472, 2012
[11] Fonseca P,Bennesby R, Mota E, Passito A, “ A replicationcomponent for resilient OpenFlow-based networking”, In: IEEENetwork Operations and Management Symposium (NOMS), pp933–939, 2012
[12] Sherwood R, Gibb G, Yap K K, Appenzeller G, Casado M,McKeown N, Parulkar G ,“Flowvisor: a networkvirtualization layer”,OpenFlow Switch Consortium, Tech. Rep, 2009
[13] Yao G, Bi J, Xiao P, “ Source address validation solution with OpenFlow/NOX architecture”, In: 19th IEEE InternationalConference on Network Protocols (ICNP), pp 7–12, 2011
[14] Braga R, Mota E, Passito A, “ Lightweight DDoS floodingattack detection using NOX/OpenFlow”, In: IEEE 35th Conferenceon Local Computer Networks (LCN), pp 408–415, 2010
[15] Wang H, Xu L, Gu G,“FloodGuard: a dos attack preventionextension in software-defined networks”,In: 45th Annual IEEE/IFIPInternational Conference on Dependable Systems and Networks(DSN), pp 239–250, 2015
[16] Lim S, Ha J I, Kim H, Kim Y, Yang S, “ A SDN-orientedDDoS blocking scheme for botnet-based attacks”, In: IEEE SixthInternational Conference on Ubiquitous and Future Networks(ICUFN), pp 63–68, 2014
[17] Suh J, Choi H G, Yoon W, You T, Kwon T, Choi Y, “Implementation of a Content-Oriented Networking Architecture(CONA): a focus on DDoS Countermeasure”, In: Proceedings ofEuropean NetFPGA Developers Workshop, 2010
[18] Phemius K, Bouet M, Leguay J,“Disco: distributed multidomainsdn controllers”,In: IEEE Network Operations and Management Symposium (NOMS), pp 1–4, 2014
[19] VoellmyA,Wang J,“Scalable software defined network controllers”,In: Proceedings of the ACM SIGCOMM2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp 289–290, 2012
[20] TootoonchianA, Ganjali Y,“HyperFlow: a distributed controlplane for OpenFlow”,In: Proceedings of the 2010 Internet NetworkManagement Conference on Research on Enterprise Networking.USENIX Association, pp 3–3, 2010
[21] Shin S, Porras P, Yegneswaran V, Fong M, Gu G, Tyson M, “FRESCO: Modular Composable Security Services for Software-Defined Networks”, In : Proceedings of Network and DistributedSecurity Symposium, pp 1-16, 2013
[22] Wen X, Chen Y, Hu C, Shi C, Wang Y,“Towards a securecontroller platform for openflow applications”, In: Proceedings ofthe Second ACM SIGCOMM Workshop on Hot Topics inSoftware Defined Networking, pp 171–172, 2013
[23] Canini M, Venzano D, Peresini P, Kostic D, Rexford J,“ANICEway to test OpenFlow applications”, In: Proceedings of the 9thUSENIX Conference on Networked Systems Design andImplementation, 2012
[24] Skowyra R, Lapets A, Bestavros A, Kfoury,“A Verifiablysafesoftware-defined networks for CPS”, In: Proceedings of the 2ndACM International Conference on High Confidence NetworkedSystems, pp. 101–110, 2013
[25] Ball T, Bjmer N, Gember A, Itzhaky S, Karbyshev A, Sagiv M,Valadarsky A,“Vericon: towards verifying controllerprograms in software-defined networks”, ACM SIGPLAN Not49(6): 282–293, 2014
[26] Son S, Shin S, Yegneswaran V, Porras P, Gu G,“Modelchecking invariant security properties in OpenFlow”. In: 2013 I.E.International Conference on Communications (ICC), pp 1974–1979, 2013
[27] Mai H, Khurshid A, Agarwal R, Caesar M, Godfrey P, King S,“Debugging the data plane with anteater”. ACM SIGCOMMComputCommun Rev 41(4): 290–301, 2011
[28] Kazemian P, ChanM, Zeng H, Varghese G,McKeown N, Whyte S,“Real time network policy checking using header space analysis”,In: USENIX Symposium on Networked Systems Design andImplementation, pp 99–111, 2013
[29] T. Nadeau, “software driven networks problem statement”, Network working group internet-Draft, sep. 30, 2011. [online]. Available: https: //tools.ietf.org/html/draft-nadeau-sdn-problem-statement-00
[30] F. Hu, Q. Hao, and K. Bao. “a survey on software-defined network and openflow: from concept to implementation”. IEEE communication surveys tutorials. Vol. 16.No. 4. Pp. 2181-2206. 4th quarter, 2014.
[31] Floodlight controller documentation for developers [Online].Available: http: //www.projectfloodlight.org/floodlight/
[32] Gude N, Koponen T, Pettit J, Pfaff B, Casado M, McKeown N,Shenker S (2008) NOX: towards an operating system for networks.ACM SIGCOMM ComputCommun Rev 38(3): 105–110
[33]OpenDaylight.[Online].Available: http: //www.opendaylight.org
[34] Ebada Essam-Eldin ElDessouky, HasanDAĞ, Ayman M. Bahaa-Eldin, “Protecting Openflow Switches against Denial of Service Attacks” ,2017 12th International Conference on Computer Engineering and Systems (ICCES), 479-484, 2018, IEEE
[35] Mohammad Mousa, Mohmed Sobh, Ayman M. Bahaa-Eldin, “Software Defined Networking concepts and challenges”, 2016 11th International Conference on Computer Engineering & Systems (ICCES), 79-90, 2016, IEEE
[36] Mohammad Mousa, Mohmed Sobh, Ayman M. Bahaa-Eldin, “Autonomic Management of MPLS Backbone Networks using SDNs”, 2017 12th International Conference on Computer Engineering and Systems (ICCES), 169-174, 2018, IEEE
[37] Reham ElMaghraby, Nada Abd Elazim, Ayman M. Bahaa-Eldin, “A Survey on Deep Packet Inspection”, 2017 12th International Conference on Computer Engineering and Systems (ICCES), 188-197, 2018, IEEE