TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of computer science

Center of digital forensics and cyber security

ITC70LT

Morteza Fakoorrad (132133 IVCMM)

Application Layer of Software Defined Networking: pros and cons in terms of security

(Master Thesis)

Supervisor: Pr. Dr Olaf M. Maennel

Ph.D.

Professor at Tallinn University of Technology

Tallinn 2016

Declaration

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

……………………………… ………………………………

(Author's signature) **(Date)**

Abstract

As time goes on “Software Defined Networking” as a new buzz phrase is becoming more popular; but this paradigm is still at an early stage of development and carries with itself new security concerns and capabilities which do not exist in traditional networking. For this reason, a security analysis of the application layer of software defined networking including north bound interface has been done. Known vulnerabilities have been addressed and a security flaw in the design of one of the northbound interfaces has been discussed. Finally, a method has been proposed to provide an effective network security defence by coupling the SIEM and northbound interfaces of software defined networking. This thesis is written in English and is 46 pages long, including 5 chapters, 7 figures and 7 tables.

Keywords:

Software Defined Networking, Application Layer, Northbound Interface, Security Information and Event Management, Security Requirements

Annotatsioon

Tarkvaraliselt määratletud võrgu rakenduskihi turvalisuse tugevused ja nõrkused

Aja jooksul on muutunud käibefraas "programmeeritav võrgustus" aina populaarsemaks, kuid sellegi poolest on see paradigma varajases arengujärgus ja võrreldes traditsioonilse arvutivõrguga kannab endas uusi seni tundmatuid turvaohte. Sel põhjusel tegin turvaanalüüsi programmeeritava võrgustuse rakenduskihile koos ülesliidesega. Käsitletud on teadaolevadid turvaauke ja ülesliidese disainist lähtuvaid turvariske. Lõpptulemusena pakun välja effektiivse võrgu turvakaitse sidudes turvamooduli programmeeritava võrgustuse ülesliidesega. Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 46 leheküljel, 5 peatükki, 7 joonist, 7 tabelit.

Märksõnad:

Programmeeritav võrgustus, Rakenduskiht, Ülesliides, Turvateabe ja turvasündmuste haldus, Turvanõuded

Acknowledgment

In the name of God, the Most Compassionate, the Most Merciful

I would like to express my deep appreciation to my supervisor Pr. Dr Olaf M. Maennel for his guidance and persistent help. I am fortune to have him as my thesis's supervisor. I would not be able to finish this research without his advice and supports.

I would like to thank my friend, Nabil Haqjoo, for his encouragement and advices for editing this thesis. I would never forget the help I got from him. I would also like to thank my friend Seifollah Akbari for providing and helping me regarding the testbed machine in the chapter four of this study. Finally, I would like to thank my parents and my wife who support me and trust me all the time.

List of Abbreviations

SIEM	Security Information and Event Management
SDN	Software Defined Networking
IDS	Intrusion Detection System
OF	OpenFlow
USM	Unified Security Management
OSSIM	Open Source Security Information Management
NBI	Advanced Persistence Technique
STRIDE	Spoofing, Tampering, Reputation, Information Disclosure, Denial of service, Elevation of privilege
DDOS	Distributed denial of services
DOS	Denial of service
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
RBAC	Role Based Access Control
CLRF	Carriage Return Line Feed
IP	Internet Protocol
HTTP	Hypertext Transfer Protocol
ARP	Address Resolution Protocol
VLAN	Virtual Local Area Networks
MPLS	Multiprotocol Label Switching
ToS	Type of service

OPCODE	Operational Code
ONF	Open Networking Foundation
ICMP	Internet Control Message Protocol
TLS	Transport Layer Security
IPv4	Internet Protocol Version 4
SAL	Service Abstraction Layer
REST	Representational State Transfer
API	Application program interface
IPS	Intrusion Prevention System
OSGI	Open Service Gateway Initiative
CLI	Command Line Interface
OPEX	operating expenditure
CAPEX	capital expenditure
GUI	Graphical user interface

Table of Contents

List of Figures

List of Tables

1- Introduction

This study is presented in five chapters. The first chapter describes the motivation and scope of the thesis. It also covers a summary of related work.

In the second chapter, the architecture and elements of software defined networking are presented. In the third chapter, a methodology has been described for eliciting security requirements which are then applied to the application layer of SDN. As a result of applying this methodology, a defensive system is proposed in the fourth chapter. The final chapter analyzes the results of the described and simulated network attacks, draws conclusions and identifies additional future work to be carried out for this research topic.

1.1 Motivation and problem statement

During the past few decades computer networks' technology has developed rapidly; this rapid development has led to a new buzz phrase called 'software defined networking' henceforth known as 'SDN'. This paradigm is based on the idea of decoupling the network's control plane from the data plane [1]. It converts the typical network's complicated routing devices into simple switches whose function is to extract and then execute the policy of the programmable logically centralized controller. The dynamic ability of SDN distinguishes it from traditional networks in which routing devices are performing both forwarding and control functions in a distributed fashion using static protocols.

Security concerns associated with SDN vary in some aspects from those of a classical network due to the specific network implementation, the software based controller and its programmable attributes. Within the scope of SDN the network architecture shifts to software defined networking. As a consequence there is a need for software defined security to be utilized in most cases. Hard-wired security approaches are seldom used.

SDN controllers are the heart of the networks and can be an easy target for attackers; they are logically organized between network infrastructures and the SDN applications. The controller provides an interface between the two. Its centralized nature enables it to prepare other SDN elements with a global vision of what is going on in the network [2].

SDN controllers manage the control plane and provide the possibility for SDN applications to enhance the functionality of SDN based networks. SDN applications include virtualized network functions and business applications that retrieve data from external appliances, such as a book keeping system in an internet service providing company, to enforce business policies on a network level. These applications interact with the controller and have capability to program the underlying infrastructure via the so called SDN northbound interfaces (NBI). However, the current NBIs have several security weaknesses.

Despite the importance of these critical interfaces, by default, they do not have enough security capabilities to monitor SDN applications and apply countermeasures in case of any possible threat. Without these security attributes, all SDN applications and anyone from a trusted boundary of SDN controller, have a full access to the underlying network enabling the possibility for potential intruders. As a result of this study, a penetration test has been done to show how SDN controller's security is affected by SDN application layer and northbound interface.

Additionally, the lack of NBIs' standards for developing SDN applications escalates the security concerns in the scope of SDN. Hence, the flexibility required by developers to design SDN applications connected to controller platforms is not provided well enough.

In this thesis, for the mentioned reasons, vulnerabilities of the SDN application layer and northbound interface of SDN controller are addressed and investigated. The most critical security weaknesses of the SDN application layer have been addressed.

Moreover, since there are no unique and specific standards for developing SDN controllers, the identification of attack methods, eliciting security requirements and security controls have been discussed in this study.

Finally, In order to illustrate the capabilities in terms of security provided via a SDN controller and the security requirements described in this study, a platform is introduced to simulate intelligent network security with the integration of SDN and SIEM.

1.2 Research questions

For this thesis, the main question is: How do the SDN application layer and NBI affect security of SDN?

The following detailed questions are also provided to answer to the main question.

1-What are the main vulnerabilities of the SDN controller and application layer?

2-What are the security requirements for the SDN controller northbound interface and application layer?

3-How can security requirements be ascertained for the SDN application layer and SDN northbound interface?

4-How can manual intervention be reduced with SDN controllers' northbound interface in order to countermeasure detected threats and attacks?

1.3 Scope

This thesis concentrates on the security of the SDN controller's northbound interface and the SDN controller application layer. The rest of SDN network elements such as the data plane and the open flow protocol are not within the scope of this paper. The security status of current SDN controllers' northbound interface and their application layers, their possible vulnerabilities and the corresponding mitigation techniques for these vulnerabilities are also addressed. Finally, as a result of eliciting security requirements, a method is proposed to provide an automated defensive system by integrating SIEM with SDN.

1.4 Related work

SDN carries remarkable potential to be an interesting research topic. Experts of the Open Networking Foundation (ONF) [3] published an article to prove that the SDN approach is a cost-effective way to implement automated malware quarantine (AMQ). This solution monitors network devices and in case of any threat and malicious activity the AMQ detects and isolates those that have become compromised before they are able to have a negative impact on the network. The AMQ is designed so that is integrated with the SDN controller. However, with the more centralized SDN management, although it can be a defense method, it can also bring with it new vulnerabilities. A very powerful SDN controller of the network could still be vulnerable to attacks from hackers because of single points of failure. Hackers will still be able to access all kind of important information. To fix this issue, in this study a defensive system by means of northbound interface APIs is proposed which is not depended on controller.

Sandra et al [4] proposed a system that provides the restriction of SDN applications by means of the API of the open source controller Floodlight. However, this method relays on controller and also does not have a mechanism to verify the authenticity of external applications.

Kevin et al [5] provided a brief overview of the vulnerabilities present in the OF protocol as it is currently deployed by hardware and software vendors. They identified a widespread failure to adopt Transport Layer Security (TLS) for the OF control channel by both controller and switch vendors, thus leaving the OF protocol vulnerable to “man-in-the-middle” attacks. However, the rest of vulnerabilities and security requirements regarding SDN application layer and northbound interface are not investigated well enough. Hence, in this study, those issues are deeply addressed.

In [6], the authors addressed the problem of DDoS attacks and provided the theoretical schema, the concept, and solutions of FireCol. The kernel of FireCol uses intrusion prevention systems (IPSs) placed at the Internet service providers (ISPs) level. The IPSs form virtual protection supports the hosts to protect and communicate by exchanging defined traffic information. However, this solution is not a lightweight method and the flexibility and dynamism is limited in this system and the deployment and management is also complicated.

In [7], challenges and threats of SDN were investigated. Authors mainly
emphasized the importance of several security objectives such as confidentiality, integrity and availability of various entities within the SDN architecture. However, they do not propose any concrete security mechanisms required to achieve these goals.

Rowan et al [8] performed a security analysis of OF using STRIDE and attack tree modeling methods. They evaluated an approach on an emulated network testbed. The evaluation assumed an attacker model with access to the network data plane. Finally, they propose appropriate counter-measures that are able to potentially mitigate the security issues associated with software defined networks. Their analysis and evaluation methods are not exhaustive, but are aimed to be adaptable and extensible to new versions and deployment contexts of OF.

2- Background

In this chapter the theoretical background for understanding the research and its motivation is explained. The structure of a software defined network, the ideas behind it together with more details about the northbound interface are described. Lastly, the current SDN controllers is briefly introduced.

2.1 Software-Defined Networking

Software-Defined Networking (SDN) idea was first time announced in 2010 [9] as a networking paradigm which intends to simplify the control and the management of a computer network environment.

SDN is an architecture in which the network control and the management are centralized and decoupled from data plane to grant the network programmable features. In the current networking paradigm, the data and the control are placed together. This means that the prevailing operating system and its attributes with the hardware are deployed in a single device. Hence, network devices, such as switches, routers and security appliances are designed with the intelligence of handling traffic relative to the adjacent devices. This causes the intelligence to be distributed in the network [10]. Furthermore, most of the network devices are Command Line Interface (CLI) based and configuration is handled separately per device, leading to time-consuming manual configuration that is prone to errors. Hence the networking industry is prevented to react quickly to feature requests or innovate new management capabilities.

2-2 Architecture

SDN architecture decouples controller from data plane and provides it a new layering model [11] . The Open Networking Foundation (ONF) [12] is a non-profit industry consortium that is leading the standardization of OpenFlow protocol which is described in the next section. The ONF defined the following layering scheme for SDN structure.

As illustrated in the Figure 1, SDN paradigm is split into three layers: application layer, control layer, and infrastructure layer. This paradigm and arrangement brings the possibility to centralize the state of the network and the intelligence into one part.

This improves the programmability feature of a network which is carried out via APIs, enabling the networking industry to innovate different solutions in the development process.

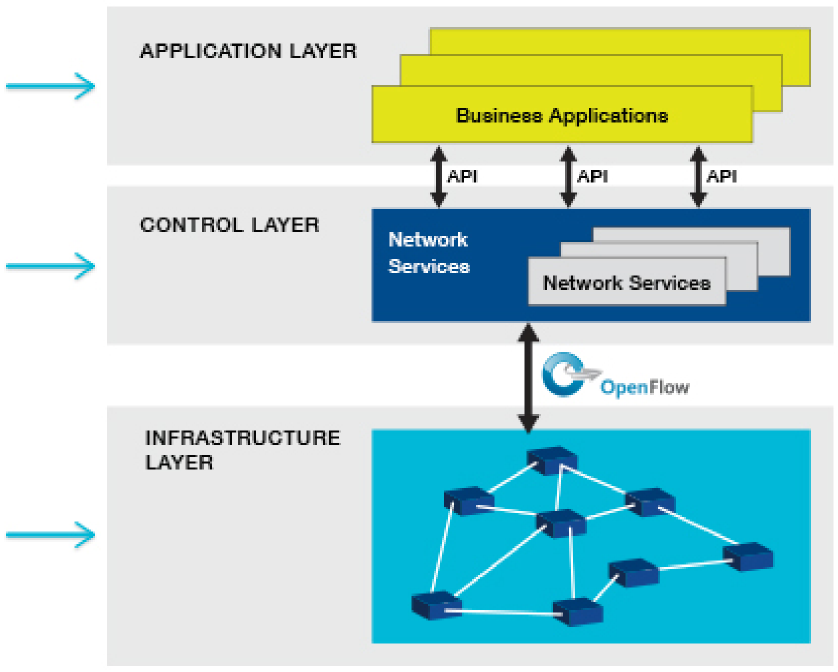


Figure 1: The SDN schema [1]

Additionally, programmability facilitates creativity and extraction of new network features and services. By means of centralization, SDN eases the process of preparing and equipping the network to allow it to provide (new) services to its users while optimizing performance and granularity of policy management. Therefore, SDN gives networks the ability to become more scalable, flexible and proactive. The control layer communicates with the infrastructure layer through the so called OpenFlow protocol and southbound interface.

2-2-1 OpenFlow

OpenFlow (OF) [13] is a communication protocol that interconnects the forwarding instructions with the data plane. The foundation is based on the SDN paradigm. It checks flow tables are updated on switches and also checks that the controller has a secure protocol for effective communication with its switches (encrypted using TLS). Figure 2 shows the basic concept of an OF enabled switch.

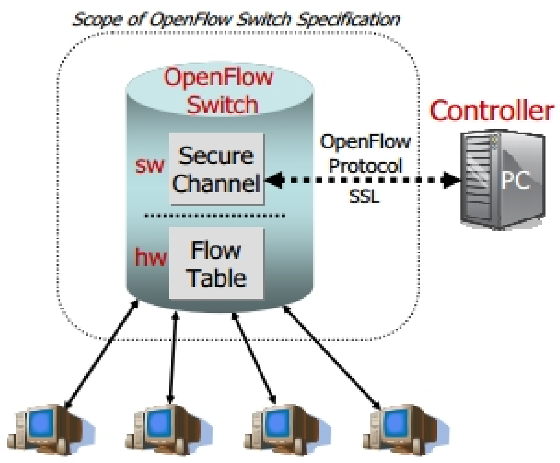


Figure 2: Scope of OF switch [14]

The goal of the OF enabled switch is to manage the switch flow tables by receiving controller rules to perform the required updates via the OpenFlow protocol. This protocol gives the ability to the controller to update, delete and insert rules in flow tables. In the following section each of these components are described separately namely the data plane, the control plan and their interaction.

2-2-2 Data-Plane

The flow table of each switch includes a set of match fields, counters and instructions. The Ingress port along with the header of each packet that the switch receives is checked against the flow table. If the packet header's important parameters such as the Ethernet port number, the source IP port, VLAN tag, destination Ethernet or IP port are found inside the flow table then there is an occurrence of a match.

If there is a match, the switch then runs the specified actions (e.g. drop the packet, block the packet or forward the packet) [15]. If there is no match the controller makes a decision and inserts a new flow table entry [15]. The switch sends a "flow request" message to the controller that includes the new packet header. The controller must verify the received flow request and decide what action should be performed on the packet. After a decision is made it must send the appropriate instructions back to the switch. This new rule indicates the actions that need to be chosen for the packet [15]. The entire process is illustrated in Figure 3.

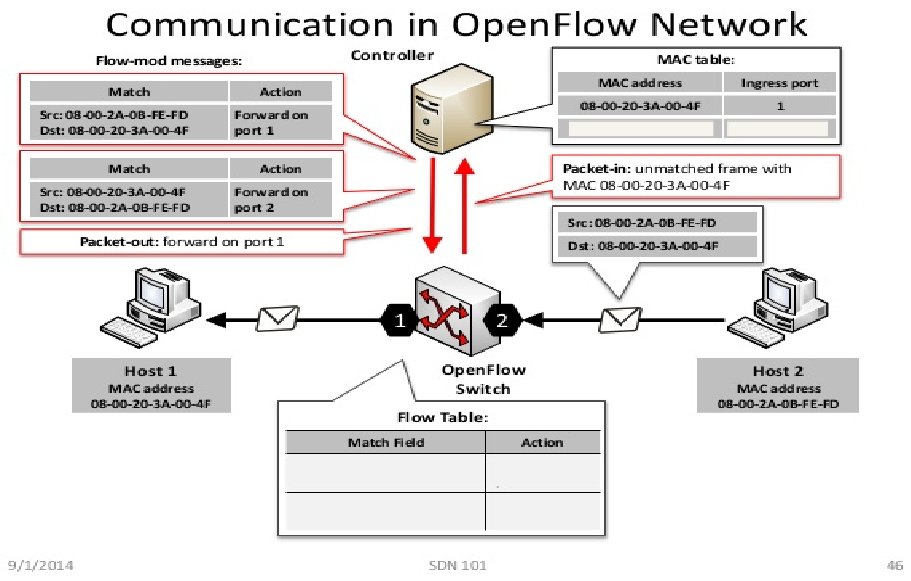


Figure 3: Communication in OF [16]

When the values in the header of the new packet are equal to those defined in the flow table then a match event occurs [17]. If a field equals the value of “ANY” it is treated like a wildcard and will match with any header value. Table 1 shows the fields of the new packet which are compared against the flow tables. In case of multiple matches, the entry with the exact match has higher priority over the wildcard entries. In case of multiple entries [17] with the same priority, the preferred flow entry is explicitly undefined. In such cases the switch is allowed to choose only one of them [17].

Type
Transport source port/ICMP type
IPv4 protocol/ARP opcode
MPLS traffic class
Ethernet destination address
Ethernet type
IPv4 source address
VLAN priority
ToS bits
Ethernet source address
VLAN id
IPv4 destination address
Metadata
MPLS label
Ingress Port
Transport destination port/ICMP code

Table 1: The fields which are compared against flow table [17]

During the last few years the research community concentrated their efforts on SDN and the OpenFlow protocol. A non-profit and user-driven organization dedicated to the promotion and adoption of SDN through open standards development was founded in 2011 as the Open Networking Foundation (ONF). Its main goal is maintaining the SDN ecosystem by introducing standards and solutions. By the March of 2015 it had already published the OF specification version 1.5.1 [17], describing the requirements for an OF switch.

Multiple vendors (like HP, IBM and CISCO) have introduced OF enabled switches to the commercial market. There are also some options for developing an OF switch for research purposes. Linux Ethernet Switches can easily work with the OF protocol, but not with outstanding performance. Furthermore, OF is not limited to hardware only, there are multiple projects for software switches. The one that is most comfortable and compatible with most of the other projects is the OpenVSwitch [18] , an open source software switch that can be installed on virtual server environments.

2-2-3 Control Layer

The Control layer is the most important part of the SDN paradigm. A controller is connected to all the network devices in the infrastructure layer and keeps track of the topology. While exchanging information of the network state with upper layer applications via northbound APIs, the controller converts their commands to low level language to the network devices in order to perform the desired network actions. The key task for the controller is adding and removing entries from the flow-tables which are in the switch data plane. Figure 4 illustrates a basic controller.

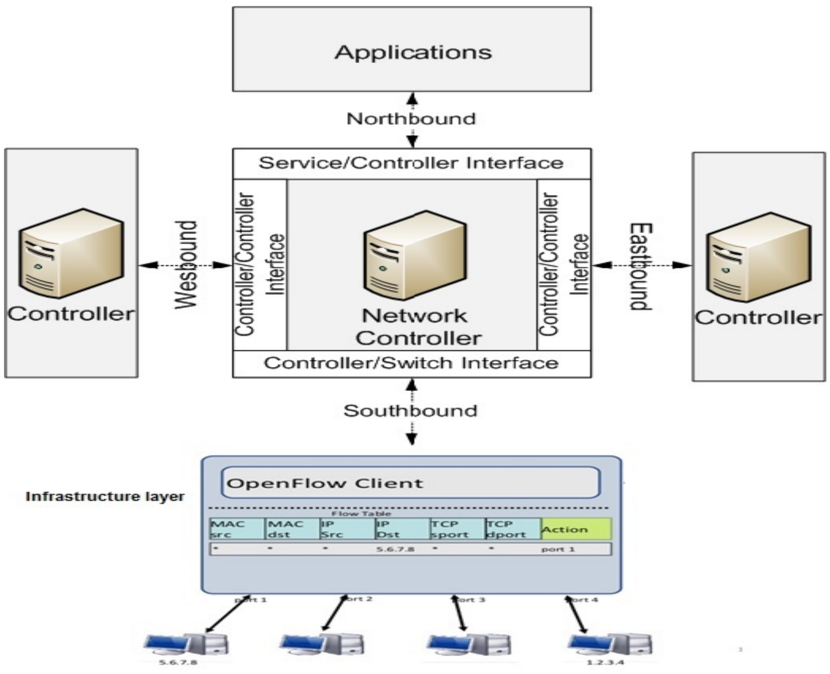


Figure 4: a basic controller

The controller communicates with switches by means of OF protocol using the so called southbound interface. Due to security reasons, OF is provided with an optional security attribute that grants the use of TLS on an OF control channel.

This method provides authentication for the switch and the controller (if certificates are appropriately checked on both sides) to stop intruders from impersonating a switch or a controller. Additionally, an encryption of the control channel to hinder eavesdropping is provided. This differs with every implementation because it is not defined.

The controller takes over of all the essential functions like topology management and service [15]. It is able to be improved with additional features; moreover it provides information to other external applications. Currently there is no unique standard for northbound communication. Some efforts have been put to enhance the abstraction level by means of designing network programming languages on application layer of the SDN. Examples of such languages are Nettle [19] and PonderFlow [20]. Finally, through the east and westbound interfaces the controller can connect to other controllers [15].

Some of the current open source controllers which are highly used released not only for commercial goals but also for academic and testing objectives are shown in the table 2.

Controller Name	Programming language	Developer
NOX	C++ & Python	Nicira
Floodlight	Java	Big Switch Networks
POX	Python	Nicira
Beacon	Java	Stanford University
Ryu	Python	NTT Communications
OpenDaylight	Java	OpenDaylight Project
Onos	Java	Open Networking Lab

Table 2: SDN Controllers

2-2-4 Infrastructure layer

The infrastructure layer is the layer in which all the network devices are placed and connected physically. On these network devices such as routers and switches, software is deployed which provides a control data plane interface (Southbound API) in order to communicate with the upper level (Control layer). It includes all equipment that enables [21] traffic forwarding.

2-2-5 Application Layer

The application layer [21]is the layer in which all the attributes, services and rules are defined. Applications demand the information of network appliances and the topology in order to react upon it. These applications are able to create end-to-end features and make decisions based on changes in the network. When the network topology, feature or policy requirements are modified, applications are able to change the network behavior dynamically. The essential communication tools between the mentioned layers are provided by means of the Application Programming Interfaces (APIs).
SDN applications [21] scope include the following areas:

- Enforcement of security and access policies
- Load balancing

- Traffic engineering
- Enforcement of QoS policy
- Network monitoring and management

The northbound APIs is provided to enable communication between the controller and its applications. Currently, there is no standard for designing this interface, yet many SDN controllers have been developed [22] with REST API. REST as an architectural approach is the most used [23] leverage for the modern services.

Other popular APIs are C++, JAVA and Python. The communication between the controller and the network data planes are done via the Southbound API.

As shown in the figure 5, the primary architectural foundation of the OpenDaylight controller consists of application and service modularity, controlled by the implementation of a service abstraction layer (SAL) [24]. The controller exposes open northbound APIs, which are used by applications. OpenDaylight, supports both the OSGi framework and the bidirectional REST APIs in the northbound layer. The OSGi framework is mainly used by applications that will run in the same address space as the controller, whereas the REST (Web-based) API is used by applications that can run on same machine as the controller or on a different machine. These applications typically realize business logic and may include all the necessary methods. In other controllers, the northbound applications use the controller to collect network intelligence, run algorithms to execute analytics, and then use the controller to orchestrate the new rules, if any, throughout the network [24].

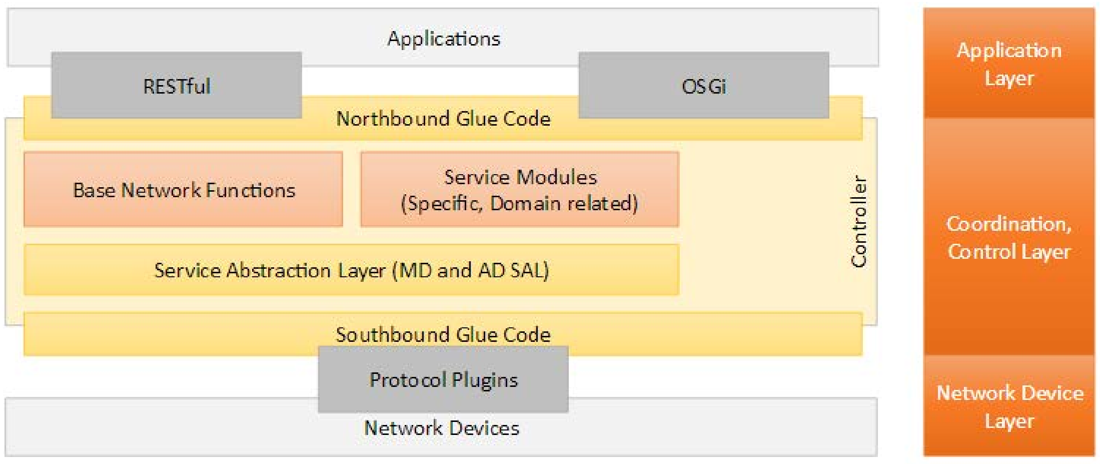


Figure 5: service abstraction layer [24]

2-2-6 Management plane:

The Management plane carries out static tasks that are better [25] to be performed outside the application, the controller and data planes. This plane should be isolated and hidden from users. The Management plane handles tasks such as setting up the network or configuration of network parameters. It should not be programmable from outside, in order to prevent any kind of network attacks and to protect the entire network.

3- SDN Application Layer and NBI Security Evaluation and Threat Modeling

In this chapter STIRDE threat modeling method is described to categorize threats regarding the SDN controller’s NBI and application layer. Moreover, the most important threats are listed along with security requirements to countermeasure the threats. Additionally, a methodology is described to elicit security requirements for the SDN controller’s northbound interface and SDN application layer.

3-1 STRIDE Threat Modeling

STRIDE [26] is a classification scheme for specifying known threats according to the types of exploit that are used. The STRIDE acronym is formed from the first letter of each of the following categories.

- **Spoofing** is an attempt to gain access to a system using a forged identity. A compromised system would give an unauthorized user access to sensitive data.
- **Tampering** is data corruption during network communication, where the data’s integrity is threatened.
- **Repudiation** is a user’s refusal to acknowledge participation in a transaction.
- **Information disclosure** is the unwanted exposure and loss of private data’s confidentiality.
- **Denial of service** is an attack on system availability.
- **Elevation of privilege** is an attempt to raise the privilege level by exploiting some vulnerability, where a resource’s confidentiality, integrity, and availability are threatened.

3-2 SDN controller’s NBI and application layer’s threat model

For identifying threats affecting the SDN controller’s NBI and application layer, three elements are defined:

·**Threat source** [27] – a source triggering the vulnerability which can be intentional or unintentional.

· **Threat target** [28] – a SDN component in which the vulnerability exists

· **Threat action** – a malicious action by means of the threat that carries negative impact on the SDN controller and the entire network.

By considering the elements and applications connected to SDN controller, different threats from various reports and a test performed during this study are categorized in this study.

3-3 Eliciting Security Requirements

Naved et al [29] presented a method used in this study for eliciting security requirements from the business process models. The first stage is dedicated to business asset identification and security objective determination. This part begins with the analysis of the value chain that provides obtaining an understanding of organizational processes which helps to determine the assets that must be protected against security risks. In terms of the security objective: confidentiality, integrity, availability should not be intercepted or negated.

In the second step, the elicitation of security requirements is done from the system's contextual areas as below:

1- **Access Control** indicates how the business assets could be changed by individuals, applications or their groups. The major goal is to protect the confidentiality of identified business asset. A method to mitigate the security risk is the introduction of access control mechanism (SR04 – defined in the section 3.4.5), for example the Role-Based Access Control (RBAC) model [29].

2- **Communication Channel** is used to exchange data between business partners. The communication channel could be intercepted by the threat agent and the captured data could be misused. This could lead to the loss of the channel reliability, and could negate the confidentiality and integrity of information. A security requirements implementation could be fulfilled by the standard transport layer security (SR06 – Latest TLS version defined in the section 3.4.5) (TLS) protocol [29].

3- **Input interfaces** should be used to input data submitted by business partners. The threat agent can exploit the vulnerability of the input interfaces by submitting the data with a malicious script. If it happens then the availability and integrity of any activity may be negated. To avoid this risk the following security requirements must be implemented for the input interface: The input interface should canonicalize the input data to verify against its canonical representation [29].

   Input-filtration validates (SR01 in section 3.4.5) the input data against the secure and correct syntax. The string input should potentially be checked for length and character set validity (e.g., allowed and blacklisted characters). The numerical input should be validated against their upper and lower value boundaries. Input sanitization should check for common encoding methods used (e.g., HTML entity encoding, URL encoding, etc) [29].

4- **Business Service** is an activity executed within an enterprise on behalf of the business partner. The goal is to guarantee availability of the business services. I.e. when receiving simultaneously multiple requests, the server will not be able to handle them; hence, the services will be unavailable [29]. To mitigate this risk, one could use packet filtering/dropping (SR07 – Automate packet dropping/blocking in controller) mechanism via security appliances which is also performed in the chapter four of this study.

5- **Data Store** is used to define how data are stored and retrieved to/from the associated databases. If the threat agent is capable of accessing and retrieving the data, their confidentiality and integrity would potentially be negated, thus, resulting in the harm of the business asset. One mitigation can be auditing [30] which is the process of monitoring and recording selected events and activities. It determines who committed what operations on what data and when; this is useful to detect and trace security violations performed. Potentially, the data store auditing could be supported by the access control policy. Moreover, important data can be encrypted. The encryption offers two-fold benefits: (i) the data would not be seen by the unauthorized users (e.g. database administrator) where the circumstances do not permit one to revoke their permissions; (ii) due to any reasons if someone gets physical access to the database (s)he would not be able to see the confidential data stored [29].

3-4 Case study

In this study, to illustrate security threats compromising northbound interface and application layer, an internet service provider with minimum required applications for specific tasks is assumed as a case study. Using the information received from the controller, possibly combined with information from other sources, it can make decision about changing the network. This is a data traffic application in an ISP shown in the diagram 1. An accounting system holds the data traffic limit of a certain user and users' financial statements. The application extracts the data usage of this user by means of the northbound interface. Once it reaches the limit defined in the accounting system, the application is able to create a rule blocking or limiting data transfer from / to that user.

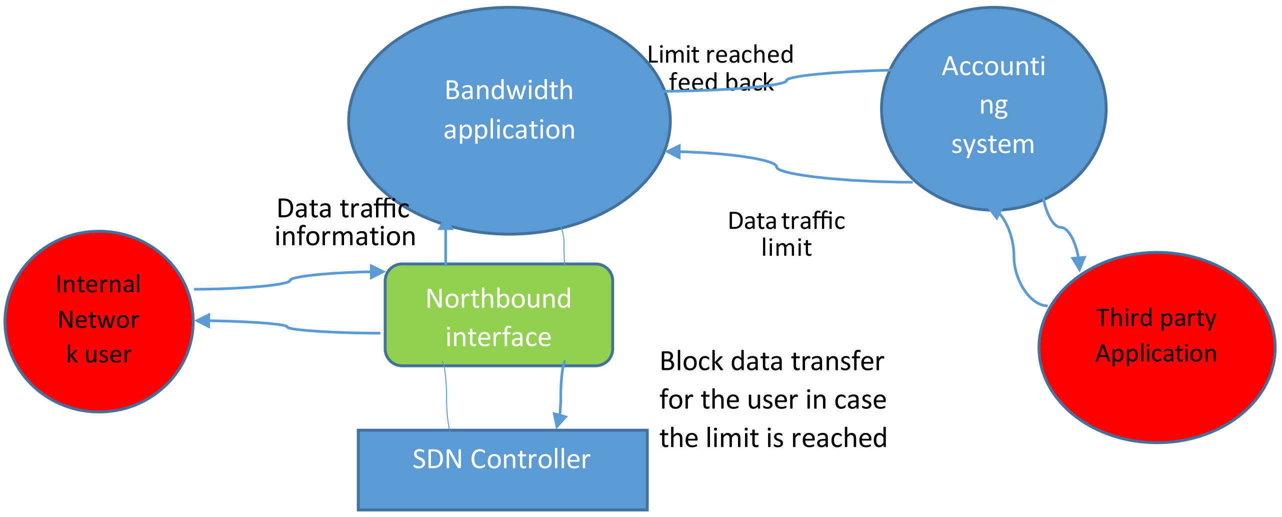


Diagram 1 Internet service provider

3.4.1 Parties (stakeholders and their goals) in the case study

Stakeholder	**Goal**
Network User	wants to have access to Internet
Network Owner	Wants the network and all processes run smoothly without any problem and grants limited traffic to users.

Table 3: Parties

3.4.2. Business assets:

Immaterial assets such as information, process essential for running the business of the organization that has value in terms of its business model and is vital for obtaining its goals [31].

Business asset	What value does each business asset bring?	What are security criterions of each asset?
BA1 – Network traffic	Updating network traffic usage automatically based on network users' information will save much time as no-one will have to do it manually. If network usage statistic is not available there will be incorrect information in book keeping system. Therefore, it has negative impact on business.	Availability. Network usage data must be available any time to update information about user accounts' availability
BA2 – Network usage data in Bandwidth application	Having up-to-date data in bandwidth application has direct impact on business. Also it is possible to generate reports on this data.	Integrity. Statistic also has to be correct and not modified.
BA3 – Network Users' information	User information is very important for checking the correctness of the network usage statics. If data (for example the limit of each user) is changed it is possible that user will pay the amount of money marked on order but get actually less traffic.	Availability. User data must be available any time when accounting system needs it. Confidentiality: unauthorized user/ application must not have access to network users' information
BA4 – Data in the accounting system	Data such as payment status in accounting system is very important for running the business and has direct impact on business	Integrity. Data in accounting system should be correct and not modified

Table 4: Business assets

3.4.3. IS (Information System) assets:

Material assets with exception of software which are part of IS that have value to the organization and support business assets to obtain the goals [31].

IS asset	**How this IS asset support business asset(s)**
IS1 – SDN controller	**IS1** Is the key of the network. It is the strategic control point in the SDN based business, relaying information to/from the switches/routers via southbound APIs and the applications and business logic via northbound APIs. All **BAs** rely on **IS1**
IS2 – Northbound Interface and its APIs	Communication between **IS1** and SDN applications is done via **IS2** ; So network statistic (**BA2**) in Bandwidth application is supported by **IS2**
IS3 – Bandwidth Application	Gathers network traffic statistic (**BA2**) and deliver it to **IS4**;
IS4 – Accounting system	**IS4** holds the data traffic limit of a certain user (**BA4**) and allows making reports regarding network usage and statistics for each user.
IS5 – Controller web interface	Has a complete view of network traffic (**BA1**) and direct access to **IS1**. And is able to change or block network traffic to/ from a user.
IS6 – flow table	Network traffic **(BA1)** information such as destination IP address are included in header fields of a flow table and routing decisions are based on these information

Table 5: Information system assets

3.4.4. Identified Risks:

Risks	Threat source , target and their expertise	Attack methods	IS asset vulnerabilities	Impacts	Threat	Event
R01 – Multi partial requests to NBI of the SDN controller	An attacker can perform application layer DDOS against northbound interface which works as a service.	Attacker can try to establish multi connections to the target NBI, but sending only a partial request. Then gradually sends more HTTP headers, but never completes a request. The targeted NBI keeps each of these wrong connections open. Afterwards, it will be unavailable. This attack is discribed and shown in the section 3.5 and appendix 1.	IS2 Accepts unlimited connections and does not have input validation against incoming packets and headers of the packets.	**Impact to business asset:** BA4 and BA2 **Impact to security criterion:** The availability of data will be intercepted;	**Denial of service**	The controller will be unavailable to all applications
R02 – Applications with chained execution [32]:	Malign applications may cause serious interference and security issues in control messages .	Malign applications [33] that participate in a service chain can drop control messages before the target applications, thus leading to extra interference. Additionally, interference may occur when a malicious application faces with an infinite loop to stop applications with chained execution.	Lack of control to authorizing to what resources an application is permitted to see and manipulate (IS1 and IS2). Lack of isolation of resources between applications.	**Impact to business asset:** BA4 and BA2 **Impact to security criterion:** The availability of data will be intercepted; The integrity will be negated	**Denial of service / Elevation of privilege**	Comminucation between SDN controller and target application will be messed up.
R03 – Gateway to unauthorized access [32]:	A malevolent nested application create an instance of another class application.	A malevolent [33] application can bypass the access control via creating an instance of another class application and can be a gateway to unauthorized access.	Lack of authorization for each instance of nested applications and weak RBAC (IS1, IS3, IS4)	**Impact to business asset:** BA2 and BA4 **Impact to security criterion:**The confidentiality of data will be egated	**Spoofing / Elevation of privilege**	SDN application will be misused due to lack of RBAC

Risks	Threat source , target and their expertise	Attack methods	IS asset vulnerabilities	Impacts	Threat	Event
R04 – unauthorized access SDN internal database [34]:	A SDN application get access to controller internal database.	An application can receives certain privileges to access the underlying resources; thus, the SDN controller shares internal storage [33] among various SDN applications. Hence, applications can access and manipulate the internal database of a SDN controller, which might be used for many subsequent attacks, such as manipulating network behavior.	Privilege abuse problem (IS1, IS3, IS4)	**Impact to business asset:** BA2 and BA3 **Impact to security criterion:** The integrity, availability and confidentiality will be negated	**Elevation of privilege**	An unauthorized access to SDN controller database via SDN application
R05 – Applications abusing SDN control messages [35]:	A control message is responsible for the two-way communication between the data plane and application plane. An arbitrarily issued control message of SDN via an application may have negetive impact in flow table.	A malicious [33] application which overwrites an existing flow rule in the controller switch flow table may cause unexpected network behavior; this event is known as flow rule modification. A malicious application may block all communication by sending a control message that clears the flow table records of a SDN switch [33].	Lack of constraint and functional requirement (IS7)	**Impact to business asset:** BA1 **Impact to security criterion:** The availability of network traffic will be negated	**Denial of service**	Network traffic will not be available if flow table is cleared or modified wrongly.

Risks	Threat source , target and their expertise	Attack methods	IS asset vulnerabilities	Impacts	Threat	Event
R06 – Exhaustion of resources [34]:	Malicious applications can contribute to exhaust the available system resources and affect the performance of other applications, including the SDN controller.	Memory consumption [33]: A malicious application might be involved in continuous consumption of system memory or in memory allocation to exhaust all the available system memory. A malicious application [33] can execute a system exit command to cancel the controller instance.	Issues in system architecture and protocol design may make systems more subject to resource-exhaustion attacks. IS1,IS3, IS4	**Impact to business asset:** BA1 **Impact to security criterion:** The availability of network traffic and applicarions will be negated	**Denial of service**	A simple denial of service condition happens when the resources necessary to perform an action are entirely consumed, therefore preventing that action from taking place.
R07- Authentication	Attacker will Access the management Console(Web GUI) by misusing Authentication weaknesses.	Conduct brute force login attempts/password guessing attacks against the management Console(Web interface): or Attacker will guess weak password and steal data through social engineering.	Managment consol accepts simple password and 'https' is not supported or not enabled by default in some controllers like Ryu and Open Mul. Additionally, NBI does not generate syslog for its authentication IS6, IS1	**Impact to business asset:** BA1 **Impact to security criterion:** Confidentiality / Integrity	**Information disclosure / Spoofing**	controller web UI will be accessed by unauthorized person.

Risks	Threat source , target and their expertise	Attack methods	IS asset vulnerabilities	Impacts	Threat	Event
R08 - Poorly designed northbound APIs/interface [36]:	An unsecure designed northbound interface can be misused easily by SDN applications to manipulate the behavior of other applications.	Attacker via SDN application may intercept an unsecure designed northbound API to eliminate an ongoing application session. Furthermore, a SDN application may use a northbound interface to unsubscribe a target application arbitrary, as a result of that, rendering it unable of reaching important subscribed control messages that can be easily done via the unsubscription of an event listener.	Weak RBAC policy (IS2) and lack of standards in designing northbound interface.	**Impact to business asset:** BA2 **Impact to security criterion:** The Availabilty of data will be intercepted.	**Tampering/ Repudiation**	Data will be unavailble or tampered by a malicouse SDN application because of weak RBAC policy.

Table 6: Identified risks

3.4.5. Security requirements

Security requirements	Which risks do the security requirements mitigate?	What are the controls that implement the defined security requirements?
SR01 – Connection header validation	R01	The NBI needs to validate connection requests. It should not accept incomplete requests or it should provide higher priority to those requests which are complete in their headers.
SR02 – Strong authorizing system	R01,R02	The interface to the SDN controller needs to support authorizing specific network resources to applications and manipulating the authorizations of applications.
SR03 – Resource isolation policy	R04,R02	Nested applications need to be able to define privacy policy regarding the resources are visible to the external application.
SR04 – Secure RBAC policy	R08, R03,R04	The system must have role-based access control – every application must have right to access only sources needed for its work process. i.e. A set of access control tokens that might be either granted or denied for an application.
SR05 – Separate authorizations	R03	The controller needs to separate authorizations held by different instances of a nested application. This secures them against bugs and operational mistakes than maintaining isolation of authorizations even if the nested application tries to bypass the authorizations.
SR06 – Latest TLS version	R05	TLS authentication between controller communications and all parts of network such as switches, applications and management console can be enabled or implemented. It does not permit unauthenticated connections bump authenticated ones.

Security requirements	**Which risks do the security requirements mitigate?**	**What are the controls that implement the defined security requirements?**
SR07 – Automate packet dropping/blocking	R06,R01	Automate packet dropping/blocking at the controller plane to avoid denial of service attacks or any suspicious traffic in near real time. Specific rules need to be installed on the network elements.
SR08 – Secure password policy	R07	Demanding secure and complex passwords from administrator; storing passwords as encrypted data; not plain text.
SR09 – limit the number of connections to NBI	R01	The number of connections can be limited as described in the section 3.6.

Table 7: Security requirements

3.5 Application layer DOS attack

The goal of this test that was performed on Ryu [37] (which is a SDN framework), is to make the controller unavailable to all network users and applications; thus, they will be unable to change the network or receive any information regarding network' status from the controller. An antagonistic user will be able to send huge amount of traffic to the northbound interface, or could send multi partial requests to the controller, resulting denial of service on the controller. Since most of controllers' NBI are designed based on apache, in this test, SlowLoris [38] DDOS attack (which send partial requests) has been performed against SDN northbound interface; thus, the controller has been unavailable to the network. Ryu doesn't have authentication in NBI at all. As a result, the controller is vulnerable to DOS attack. Despite the fact that some of other controllers like Open-Daylight have authentication in NBI, they don't generate syslog for it or it is not enabled. Hence, brute-force attack might not be detected alongside with DOS attack.

Because SlowLoris exploits a vulnerability in a web server / service which waits for a complete header to be received. Apache and the web servers like dhttpd and Goahead have a feature of timeout [39]. These web servers wait for the specified timeout duration for the completion of a request. By default, the timeout value is 300 seconds, but it can be modified.

Additionally, the timeout counter is reset each time the client sends more data. But we should consider a situation if someone intentionally send partial http requests and reset the timeout counter of each request by sending some fake data continuously.

This is exactly what Slowloris performs. It is carried out by establishing connections to the target server, but sending only a partial request. Then gradually sends more HTTP headers, but never completes a request. The targeted server keeps each of these wrong connections open. This finally overflows the maximum connection pool, and leads to denial of additional connections.

SlowLoris is written in Perl as an open-source tool to show how a single computer can bring down a web server by consuming all of the resources of the server. It is aimed to show Apache how its servers/services could be vulnerable to this attack due to a design principle.

SlowLoris or low-bandwidth DOS attacks occur in the application layer against web servers or web services and does not matter whether it is a web service or it is a full blown web server; it still has the same weaknesses, and can be affected by the same attacks. Since the NBI works as a web service, this attack brings down the APIs and keeps anyone away from making changes to the network.

Details of the test have been documented in the appendix 1.

3.6 Solutions for mitigating the SlowLoris DDOS-attack

Since SlowLoris attack does not send a malformed request to NBI, usually it cannot be detected by traditional intrusion detection systems, as a result, it evades of them.

3.6.1 Connection request validation

The NBI needs to validate connection requests. It should not accept incomplete requests or it should provide higher priority to those requests which are complete in their headers.

As described above, SlowLoris sends incomplete http request with fake headers. A complete GET request should be like below:

1- GET / HTTP/1.0[CRLF]

2- User-Agent: Wget/1.10.2 (Red Hat modified)[CRLF]
3- Accept: */*[CRLF]
4- Host: 192.168.56.103[CRLF]
5- Connection: Keep-Alive [CRLF][CRLF]

What is [CRLF] in the Get request?

CRLF is acronym of CR (Carriage Return) and LF (Line Feed) [40]. CRLF is non printable, used to define end of the line. Even a text editor sets a CRLF at the end of a line when we want to put a new line after current line. And two CRLF characters define a blank line.

In the above example, there are two CRLF characters at the end of the "Connection" header (which defines an empty line). In http protocol, an empty line after the header, represent the completion of the header.

Slowloris Dos attack misuses the described CRLF. It does not send a complete empty line, which shows the end of the http header. A partial request sent by the SlowLoris, is described below.

```
  "GET /$rand HTTP/1.1\r\n"
 ."Host: $sendhost\r\n"
 ."User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;          Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.503l3; .NET CLR 3.0.4506.2152;    .NET CLR 3.5.30729; MSOffice 12)\r\n"
. "Content-Length: 42\r\n";
```

In the above code, \r\n is used to define carriage return and newline in Perl language. Two sequential "\r\n\r\n", must be used to define a blank line. Hence, this is a partial header in HTTP.

In order to fix this issue, HTTP request validation engine/approach can be implemented. Jeff [41] described and implemented a comprehensive HTTP validation engine that can be implemented on NBI to mitigate the attack.

3.6-2 limiting the number of connections to the controller from NBI

It is possible to limit the number of connections by means of iptables on port 8080 (which the NBI uses it) of the controller. As an example:

```
# iptables -A INPUT -p tcp --syn --dport 8080 -m connlimit --connlimit-above 50 -j DROP
```

In the next chapter of this study due to programmable feature of SDN, a methodology is proposed enable us to provide the security requirement SR07 (Automate packet dropping/blocking at controller which can also be considered as mitigation method for the most of the attacks) as well as an intelligent network security in the SDN world.

4- Automate packet dropping/blocking at SDN controller

Programmability and flexibility of SDN facilitates providing intelligent network security via integration of security appliances into SDN based networks, which can be implemented on the application layer of the SDN using northbound interface, rather than being implemented as split element. SDN's centralized administration provides events through the network to be gathered and normalized, thus, more precise image of the network's status provides security strategies easier to enforce and to monitor [2].

The ability to implement security approaches on top of the controller gives us possibility to dynamically place appliances and sensors at different places in the network that provides more useful network monitoring. Via a precise vision of its status, the network is able to easily detect attacks, and the number of errors in reporting and alerting can be decreased. In operation, if a security appliance shows to the SDN controller that a device is giving signs of being attacked by a botnet or any activity detected via security appliances, the SDN controller is able to reroute the potentially offending traffic to a security appliances for further analysis [2]. In case the traffic is considered malicious via the IDS or security appliances, the SDN controller can filter or drop it.

In this chapter, the following contributions are described:

1- A methodology to achieve rich concept of coupling between SDN and SIEM segment, and the relation between these segments and protocols.

2- Implementation of the idea in a virtual environment with related components with sample attack scenario. Security alert is created in SIEM and sent to SDN controller to deploy rule in the SDN network for mitigations of attacks.

3- Results of our experiences and benefits of coupling SDN and SIEM.

But before, SIEM and the type of SIEM used in this study is shortly described.

4-1 SIEM

Security information and event management (SIEM) is an approach [42] regarding security management that provides a complete picture of an organization's information technology security. The principle of a SIEM system is that relevant data about an enterprise's security is generated in different locations and being able to look at all the data from a single point of view makes it easier to spot trends and see patterns that are out of the ordinary. In fact, SIEM technology provides near real-time analysis of security alerts, which have been produced via network appliances. SIEM can be considered as software, appliance or managed service, and is also applied to log security data to generate reports for compliance goals. The aim of SIEM is to aid to companies' reaction to attacks faster in a fashion prompt.

SIEM not only can process log data but also is able to quarry other types of data. Event data includes contextual information about users, data and assets. The data is normalized, hence, events from various sources can be correlated and analyzed for different goals, such as network security event monitoring, user activity monitoring or compliance reporting. The technology enables historical analysis, real-time security monitoring, and other support for incident management and analysis and compliance reporting [43]. According to Detken et al [44], a complete SIEM approach includes various modules that support the following functionalities:

a- **Event correlation**: save, archive, normalize, and correlate log files to a data-base for analyzing additionally.

b- **Situation detection:** monitoring network condition and detection of unwanted situations. It might include anomaly detection techniques; signature based matching and configuration management.

c- **Application programming interface (API):** provides a generic interface for the integration of various devices and systems.

d- **Identity mapping:** allocating network specific information such as IP/MAC address to real entities like actual user.

e- **Key performance indication:** measurement of the IT security by central analysis of asset details regarding security information.

f- **Role based access control:** holistic view of all events under consideration of different responsibilities.

g- **Compliance reporting:** regular monitoring on the IT compliance such as integrity and risk of an enterprise.

4-1-1 OSSIM

AlienVault developed two different SIEMs, one open source and the other one is a closed source commercial version. The commercial Unified Security Management platform (USM) provides additional supports [45] for log management and scaling enhancements for the open source version.

AlienVault uses its own correlation engine to provide real time analysis of sensor data from AlienVaults agent software. OSSIM is one of the easiest SIEM products to configure and deploy along with a lower cost of deployment than other SIEM vendors' products. Both products provide a SIEM solution with file monitoring, NetFlow support, host and network intrusion detection capabilities and vulnerability assessment capabilities. Providing OSSIM as 'all in one' solution that is able to provide various methods of collecting and processing data from different sources, gives it a deep insight into the network activities and ability to react to threat as they emerge. Despite the fact that main function of a SIEM is correlation not detection, OSSIM has detector plugins [46] that make it useful in this study.

4-2 Methodology

In this method, network activity data are gathered with sensors and sent to the SIEM server in order to detect malicious activities. SIEM processes data and generates network activity information. This information provides a clear view of network activity and the critical event will result in creation of a ticket within the SIEM. At this time, the ticket can be converted to a flow that can be carried out on SDN controller automatically by system or manually by the network administrator.

These rules on SDN network may change topology and create another ticket to change network topology again. Therefore, these changes can have a feedback. Each new flow will have an assigned expiry time, to be able to effectively be checked against other existing flows as a means of preventing network conflicts, prior to activation of that flow. The event's severity can cause action on one or more network layers. For instance, flood ping may be used to check the network or device performance. But its duration and packet size can have a drastic negative impact on network performance.

As shown in the figure 6, this activity will be detected by SIEM and create an event with high severity. The resulting ticket will make a flow rule to SDN controller and drop packets of source IP or MAC or Port (layer 3, 2 or 1).

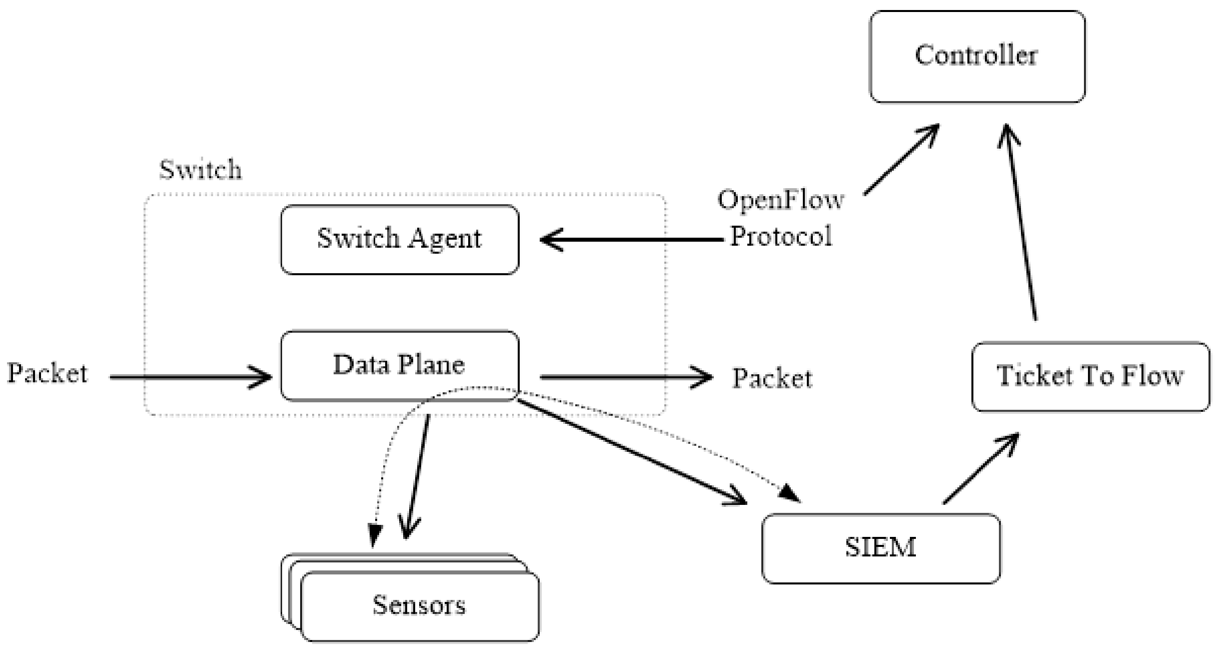


Figure 6: Coupling SDN & SIEM

An OF switch consists of two components: the switch-agent and the data plane. The switch-agent speaks [47] OF protocol to one or more controllers. Additionally, it communicates with the data plane using the requisite internal protocol [48]. The switch-agent will translate the controller issued commands into the necessary low-level instructions to send to the data plane, as well as translate internal data plane notifications into appropriate OF messages and forward to the controller. The data plane performs all packet forwarding and manipulation; however, based on its configuration, sometimes it sends packets to the switch-agent for further handling [48].

Current OF releases span versions from 1.0 to 1.5.1 OF 1.0 can perform actions against port IDs, queue IDs, and up to 4 protocols. It is important to note that support for actions against Ethernet, IPv4, TCP, and UDP, are optional. An SDN controller will check and know if a switch supports theses optional actions. In the testbed of this study OF 1.3 is used. In OF 1.3 each flow entry of a flow table has a set of instructions to apply to all matching packets. According to [49] these instructions can be used to modify the packets state, forward the packet to a particular port, forward the packet to another table or group for further processing and generate metadata in the process. Each packet maintains an action set, which contains a set of actions to apply to the packet when no further table processing can be accomplished. The write and clear instructions provide ways of manipulating the action set. The apply instruction bypasses the action set and performs actions immediately. The Goto instruction provides a mechanism to choose the next flow table for processing. The meter instruction allows the application of a rate limiter to the flow. The experimenter instructions provide a structure for custom extensions to instructions. Apply and write are the only instructions that apply actions in some way, as input they both contain action lists.

4-3 Implementation

To integrate SIEM with SDN, many different methods can be used. One such method is to use simple API. When SIEM creates an alert, API sends source IP of the attacker to SDN controller to create a new flow to block the attacker's IP or to drop the packets which are sent from specific sources. If manual control is preferred, another API may be used to send alert details to the SDN controller's GUI, to be resolved by the system's administrator. The integration is implemented using both the automatic method and manual method. As shown in the figure 7, the testbed was hosted on a laptop with a 12 GB RAM and a Core i5 CPU in a virtual environment; packet flooding was performed from a trusted boundary and denial of service was considered as the threat model. The details of the implementation are described in the appendix two.

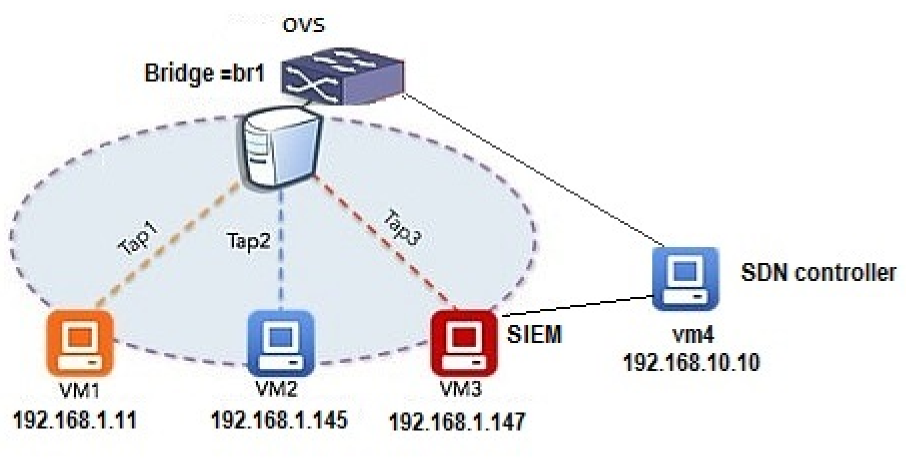


Figure 7: Testbed of Integrating SDN controller with SIEM

1. One SDN Controller based on Open-Daylight Service Provider
2. One OF enabled switch with 4 virtual ports based on OpenVswitch with OF version 1.3
3. One 'All in one' OSSIM Server for gathering data
4. Two network clients connected to switch for a simple packet flooding

5- Conclusion and Future Work

The security vulnerabilities, attacks and threats discussed in chapter three are critical and can compromise all the corresponding SDN layers and interfaces.

In order to answer the main question of the thesis, it has been illustrated that due to the lack of security standards in the design of the SDN NBI and application layer, with a low-bandwidth DOS attack against NBI from the application layer, the controller became unavailable and it was not possible to receive or send any information from the controller. Hence, the dependency between the application layer of SDN and the controller of the network is a security concern that should not be ignored in the future study. Despite the fact that the described attacks in the chapter three occur against different applications and web servers/services in our daily life and have a negative impact, they rarely could happen in traditional networking, having a negative impact on the controller in which the controller is integrated to data plane on a router.

The security of the SDN controller not only depends on NBI, but also is highly affected by other permissive applications that communicate with NBI. As described in chapter three, a malicious user by means of a permissive application can reach the controller. Therefore, if SDN is deployed in real life the business owners who deploy SDN in their organization should also consider that their network might be affected by the application that communicates with the NBI of SDN. Thus, security requirements for the applications which communicate with SDN controller via NBI should not be ignored and should be elicited with a proper method according to the application types and the business process model of the organization which its network is based on SDN architecture.

Protection against different applications will remain a challenge for the SDN paradigm. No secure architecture for various users or network service applications exists, and the access control of nested applications has not been illustrated.

On the other hand, due to programmable feature of SDN, it has successfully been demonstrated that the emerging SDN technology has provided new and effective methods to tackle network security challenges. This opens new horizons in network security disciplines and resolves problems of conflicts in the configuration of the numerous network security devices and other active network elements. The integration of SIEM and SDN technologies as demonstrated represents a unique network-wide intelligent security solution with integrated advanced detection and reaction capabilities.

Major vendors are offering OF functionality in their products. These are becoming more available on the growing SDN market. It seems very likely that the adoption of OF will continue at an increasing rate in the coming years. In the SDN network infrastructure, SIEM is able to provide an automated reaction to security threats in an effective manner due to its single point configuration of the numerous network security devices and other active network elements.

The study showed that when attack traffic was detected it could be automatically dropped or routed to a honeypot for further monitoring where it would be possible to gather data log files about the attack activity and its metrics. This data can be further analyzed by SIEM. SIEM can then deploy pro-active security rules to the entire network based on this acquired data.

Although an integrated SIEM / SDN implementation can successfully release many hardware resources from networks and greatly simplify security management, eliminating 'human-in-the-loop' intervention is often impractical; human intervention by a trained incident handler is often required. Most of the times for forensic reasons or business needs a compromised system may be kept online.

The integrated use of SIEM for security monitoring with intelligent creation and deployment of security rules in SDN based networks can effect finer grained security policies whereby each individual network port can act as a single firewall. The savings in CAPEX and OPEX can be very considerable.

Appendix 1: SlowLoris DOS attack from application layer against SDN northbound interface

At the first step, we need to install Ryu framework with the following steps:

1- git clone git://github.com/osrg/ryu.git

2- cd ryu; python ./setup.py install

For running the controller, the following file should be executed

3- ./ryu/app/sdnhub_apps/run_sdnhub_apps.sh

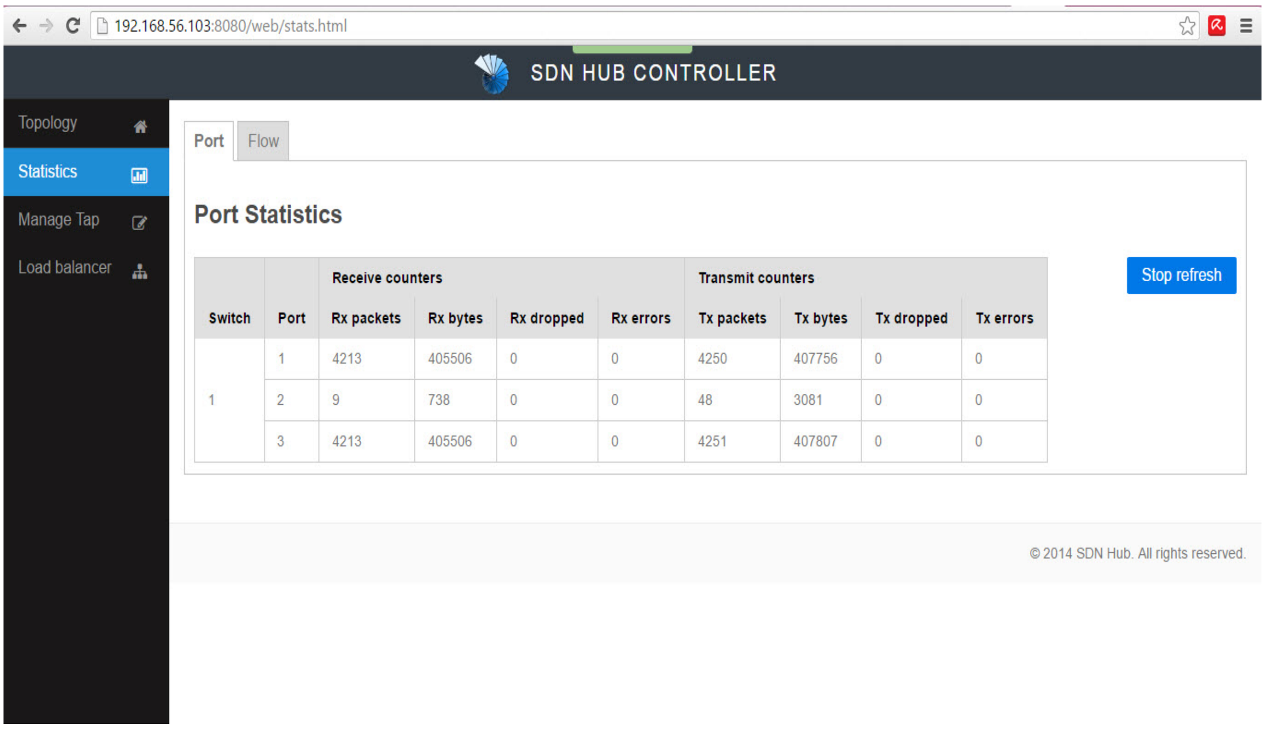
Then we can add a switch if we do have openVswitch available or installed by the following command:

4- sudo apt-get install openvswitch-switch

5- ovs-vsctl add-br s1

6- sudo ovs-vsctl set-controller s1 tcp:192.168.56.103:6633

The web GUI of the Ryu controller is available on the ip:8080 as shown in the figure below:



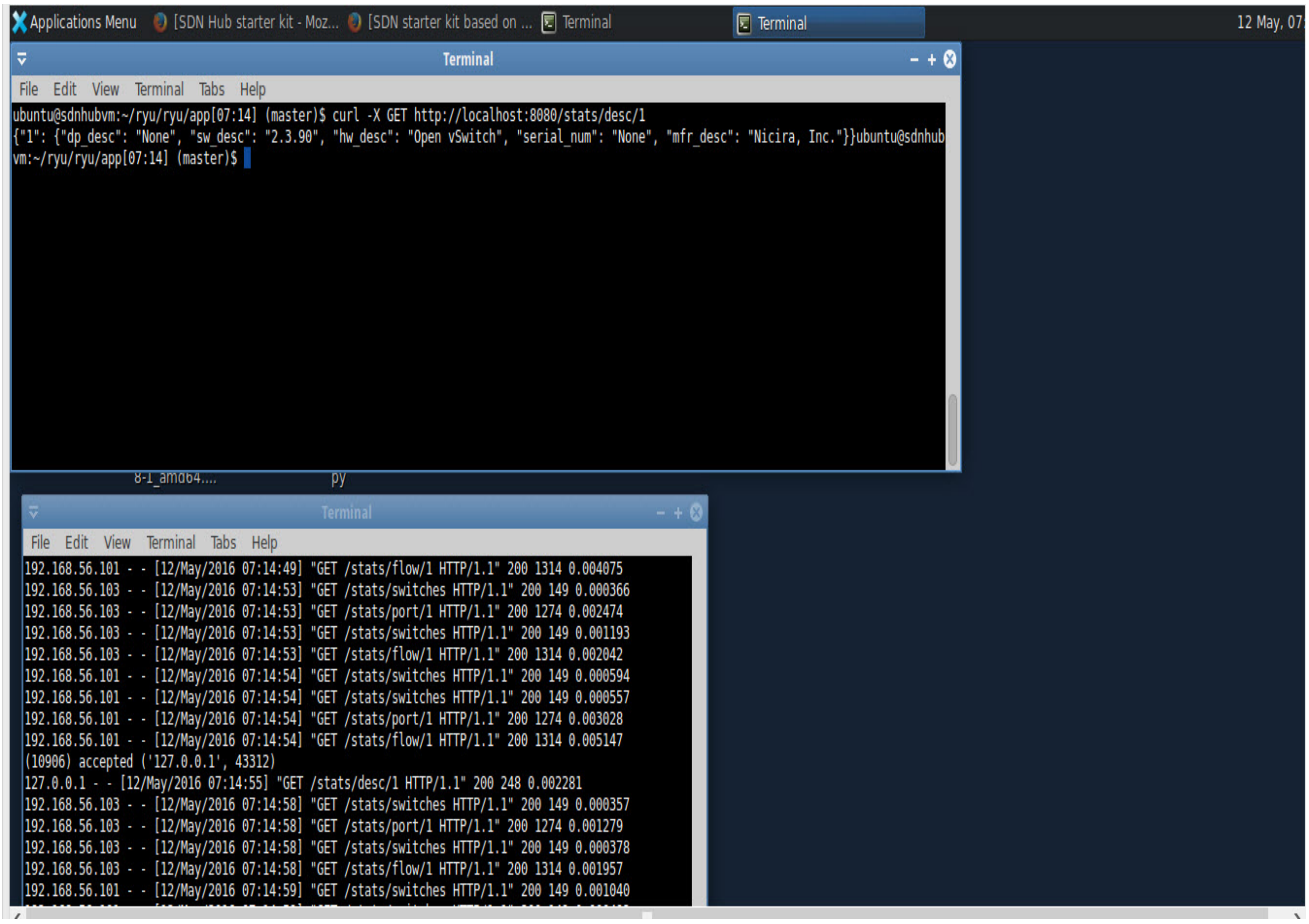
Port Statistics

		Receive counters				Transmit counters			
Switch	Port	Rx packets	Rx bytes	Rx dropped	Rx errors	Tx packets	Tx bytes	Tx dropped	Tx errors
	1	4213	405506	0	0	4250	407756	0	0
1	2	9	738	0	0	48	3081	0	0
	3	4213	405506	0	0	4251	407807	0	0

Now as an application, we want to make an inquiry about the switch information:

7- curl -X GET http://192.168.56.103:8080/stats/desc/1

The result is shown in the following figure:


Applications Menu
[SDN Hub starter kit - Moz...
[SDN starter kit based on ...
Terminal
Terminal
12 May, 07
Terminal
File Edit View Terminal Tabs Help
8-1_amd64....
py
Terminal
File Edit View Terminal Tabs Help

```
ubuntu@sdnhubvm:~/ryu/ryu/app[07:14] (master)$ curl -X GET http://localhost:8080/stats/desc/1
{"1": {"dp_desc": "None", "sw_desc": "2.3.90", "hw_desc": "Open vSwitch", "serial_num": "None", "mfr_desc": "Nicira, Inc."}}ubuntu@sdnhub
vm:~/ryu/ryu/app[07:14] (master)$
```

```
192.168.56.101 - - [12/May/2016 07:14:49] "GET /stats/flow/1 HTTP/1.1" 200 1314 0.004075
192.168.56.103 - - [12/May/2016 07:14:53] "GET /stats/switches HTTP/1.1" 200 149 0.000366
192.168.56.103 - - [12/May/2016 07:14:53] "GET /stats/port/1 HTTP/1.1" 200 1274 0.002474
192.168.56.103 - - [12/May/2016 07:14:53] "GET /stats/switches HTTP/1.1" 200 149 0.001193
192.168.56.103 - - [12/May/2016 07:14:53] "GET /stats/flow/1 HTTP/1.1" 200 1314 0.002042
192.168.56.101 - - [12/May/2016 07:14:54] "GET /stats/switches HTTP/1.1" 200 149 0.000594
192.168.56.101 - - [12/May/2016 07:14:54] "GET /stats/switches HTTP/1.1" 200 149 0.000557
192.168.56.101 - - [12/May/2016 07:14:54] "GET /stats/port/1 HTTP/1.1" 200 1274 0.003028
192.168.56.101 - - [12/May/2016 07:14:54] "GET /stats/flow/1 HTTP/1.1" 200 1314 0.005147
(10906) accepted ('127.0.0.1', 43312)
127.0.0.1 - - [12/May/2016 07:14:55] "GET /stats/desc/1 HTTP/1.1" 200 248 0.002281
192.168.56.103 - - [12/May/2016 07:14:58] "GET /stats/switches HTTP/1.1" 200 149 0.000357
192.168.56.103 - - [12/May/2016 07:14:58] "GET /stats/port/1 HTTP/1.1" 200 1274 0.001279
192.168.56.103 - - [12/May/2016 07:14:58] "GET /stats/switches HTTP/1.1" 200 149 0.000378
192.168.56.103 - - [12/May/2016 07:14:58] "GET /stats/flow/1 HTTP/1.1" 200 1314 0.001957
192.168.56.101 - - [12/May/2016 07:14:59] "GET /stats/switches HTTP/1.1" 200 149 0.001040
```

In this step, we want to launch Slowloris Dos attack from a trusted boundary against NBI:

SlowLoris-type DDOS tool are available in both Perl and Python scripts as open-source tool:

Requirements:

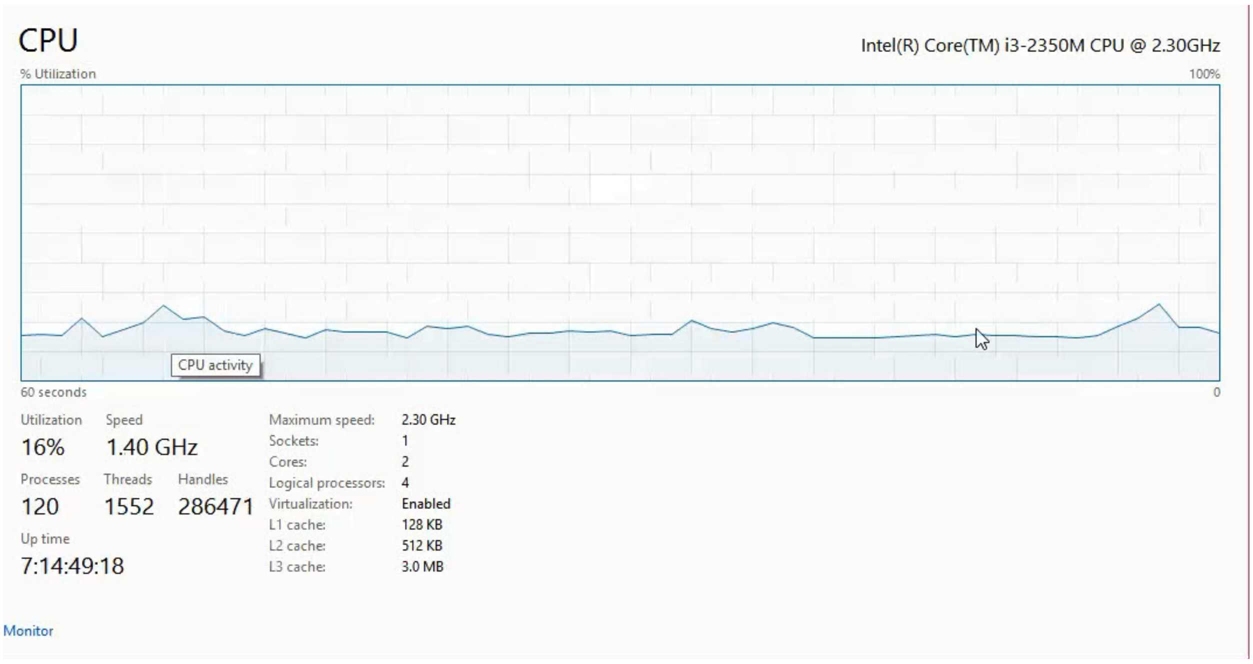
8- sudo apt-get update

9- sudo apt-get install perl

10- sudo apt-get install libwww-mechanize-shell-perl

11- wget - c https://github.com/llaera/slowloris.pl/archive/master.zip

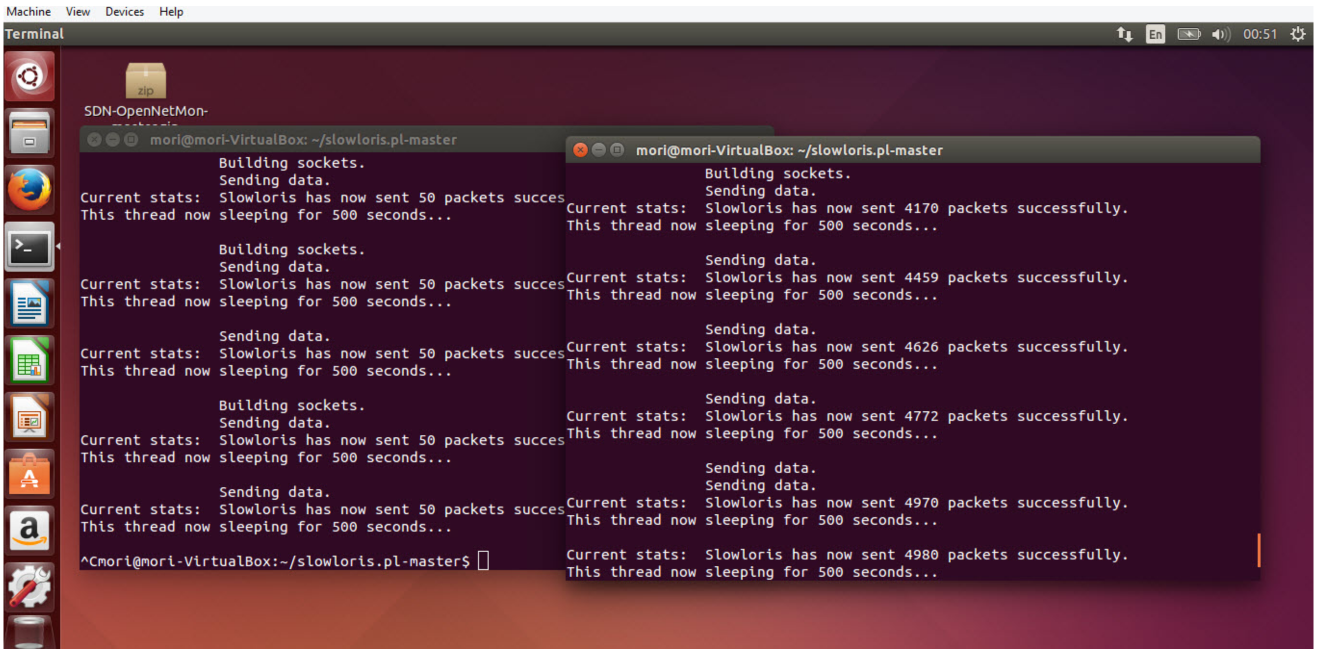
CPU usage before doing the attack is shown in the picture below:



```
Utilization   Speed          Maximum speed:       2.30 GHz
16%           1.40 GHz       Sockets:             1
                             Cores:               2
Processes   Threads   Handles   Logical processors:   4
120         1552      286471    Virtualization:       Enabled
                                L1 cache:             128 KB
Up time                         L2 cache:             512 KB
7:14:49:18                      L3 cache:             3.0 MB
```

12 - After unzipping the file, we need to run the attack with the following command:

Sudo perl Slowloris.pl - dns 192.168.56.103 - port 8080 - timeout 500 - tcpto 5 - num 1000

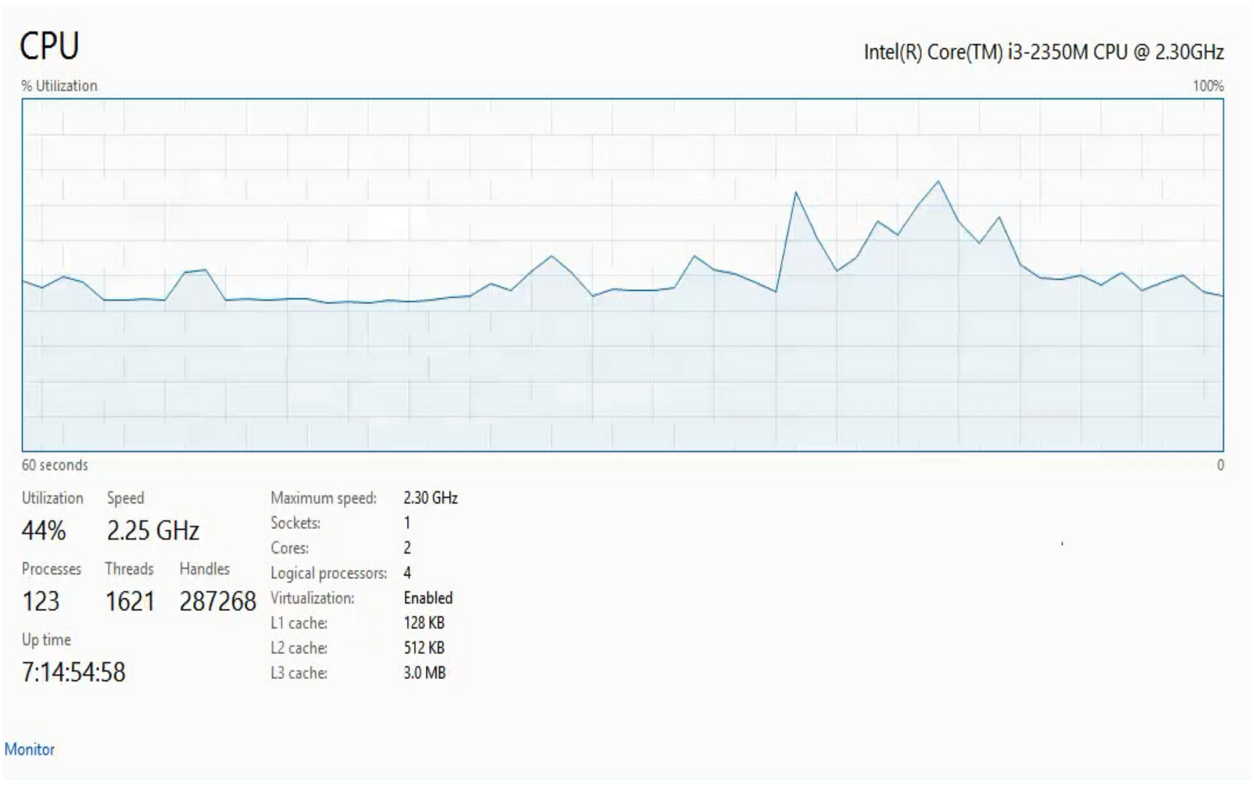


```
mori@mori-VirtualBox: ~/slowloris.pl-master
                 Building sockets.
                 Sending data.
Current stats:   Slowloris has now sent 50 packets succes
This thread now sleeping for 500 seconds...

                 Building sockets.
                 Sending data.
Current stats:   Slowloris has now sent 50 packets succes
This thread now sleeping for 500 seconds...

                 Sending data.
Current stats:   Slowloris has now sent 50 packets succes
This thread now sleeping for 500 seconds...

                 Building sockets.
                 Sending data.
Current stats:   Slowloris has now sent 50 packets succes
This thread now sleeping for 500 seconds...

                 Sending data.
Current stats:   Slowloris has now sent 50 packets succes
This thread now sleeping for 500 seconds...

^Cmori@mori-VirtualBox:~/slowloris.pl-master$
```

```
mori@mori-VirtualBox: ~/slowloris.pl-master
                 Building sockets.
                 Sending data.
Current stats:   Slowloris has now sent 4170 packets successfully.
This thread now sleeping for 500 seconds...

                 Sending data.
Current stats:   Slowloris has now sent 4459 packets successfully.
This thread now sleeping for 500 seconds...

                 Sending data.
Current stats:   Slowloris has now sent 4626 packets successfully.
This thread now sleeping for 500 seconds...

                 Sending data.
Current stats:   Slowloris has now sent 4772 packets successfully.
This thread now sleeping for 500 seconds...

                 Sending data.
                 Sending data.
Current stats:   Slowloris has now sent 4970 packets successfully.
This thread now sleeping for 500 seconds...

Current stats:   Slowloris has now sent 4980 packets successfully.
This thread now sleeping for 500 seconds...
```

Then we need to wait for some minutes; afterwards, we will be unable to get any information from the controller or make any change on it.

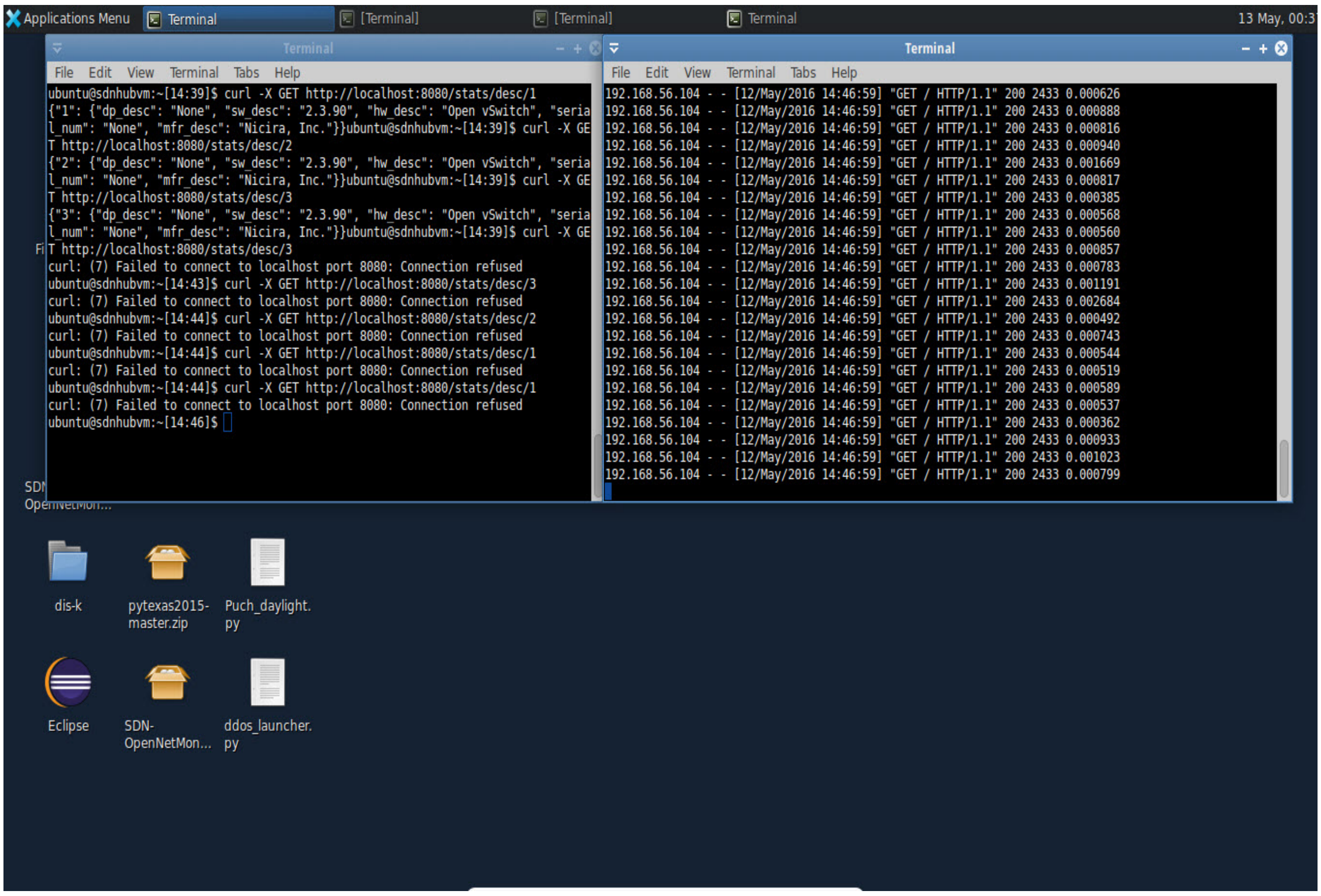
The CPU usage some minutes after launching the attack is shown below:



Utilization 44%
Speed 2.25 GHz
Processes 123
Threads 1621
Handles 287268
Up time 7:14:54:58

Maximum speed: 2.30 GHz
Sockets: 1
Cores: 2
Logical processors: 4
Virtualization: Enabled
L1 cache: 128 KB
L2 cache: 512 KB
L3 cache: 3.0 MB

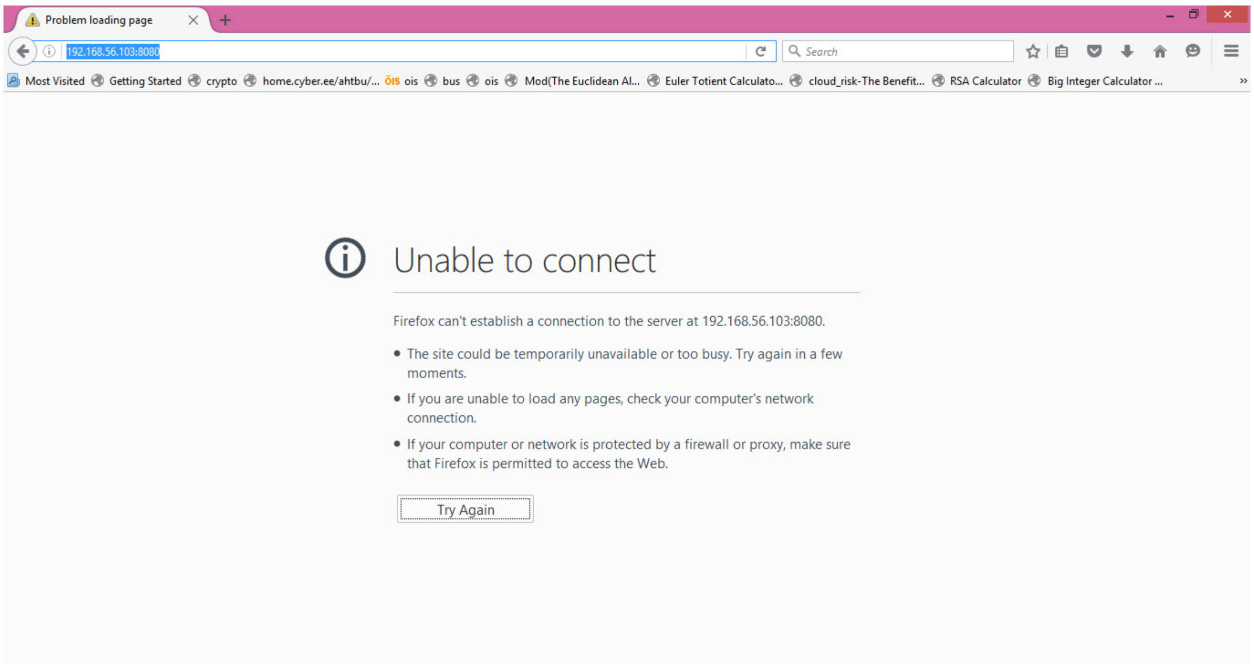
So, now it is the time to check the switch's status again as an application:



```
ubuntu@sdnhubvm:~[14:39]$ curl -X GET http://localhost:8080/stats/desc/1
{"1": {"dp_desc": "None", "sw_desc": "2.3.90", "hw_desc": "Open vSwitch", "seria
l_num": "None", "mfr_desc": "Nicira, Inc."}}ubuntu@sdnhubvm:~[14:39]$ curl -X GE
T http://localhost:8080/stats/desc/2
{"2": {"dp_desc": "None", "sw_desc": "2.3.90", "hw_desc": "Open vSwitch", "seria
l_num": "None", "mfr_desc": "Nicira, Inc."}}ubuntu@sdnhubvm:~[14:39]$ curl -X GE
T http://localhost:8080/stats/desc/3
{"3": {"dp_desc": "None", "sw_desc": "2.3.90", "hw_desc": "Open vSwitch", "seria
l_num": "None", "mfr_desc": "Nicira, Inc."}}ubuntu@sdnhubvm:~[14:39]$ curl -X GE
T http://localhost:8080/stats/desc/3
curl: (7) Failed to connect to localhost port 8080: Connection refused
ubuntu@sdnhubvm:~[14:43]$ curl -X GET http://localhost:8080/stats/desc/3
curl: (7) Failed to connect to localhost port 8080: Connection refused
ubuntu@sdnhubvm:~[14:44]$ curl -X GET http://localhost:8080/stats/desc/2
curl: (7) Failed to connect to localhost port 8080: Connection refused
ubuntu@sdnhubvm:~[14:44]$ curl -X GET http://localhost:8080/stats/desc/1
curl: (7) Failed to connect to localhost port 8080: Connection refused
ubuntu@sdnhubvm:~[14:44]$ curl -X GET http://localhost:8080/stats/desc/1
curl: (7) Failed to connect to localhost port 8080: Connection refused
ubuntu@sdnhubvm:~[14:46]$
```

```
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000626
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000888
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000816
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000940
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.001669
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000817
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000385
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000568
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000560
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000857
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000783
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.001191
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.002684
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000492
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000743
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000544
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000519
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000589
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000537
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000362
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000933
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.001023
192.168.56.104 - - [12/May/2016 14:46:59] "GET / HTTP/1.1" 200 2433 0.000799
```

Also web interface of the controller:


Problem loading page

192.168.56.103:8080

Most Visited · Getting Started · crypto · home.cyber.ee/ahtbu/... · ois · bus · ois · Mod(The Euclidean Al... · Euler Totient Calculato... · cloud_risk-The Benefit... · RSA Calculator · Big Integer Calculator ...

Unable to connect

Firefox can't establish a connection to the server at 192.168.56.103:8080.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Try Again

It is failed to connect to the localhost port 8080 which NBI was running on it; the connection was refused.

The test video has been uploaded in the following link:

https://www.youtube.com/watch?v=dDsGNUTiqVA

Appendix 2: integration SDN with SIEM (OSSIM)

SDN controller installation and configuration

1- At the first step we need to install OpenDayLight SDN controller. So, we need to download it from the link below and setup on the Ubuntu server 14 – 64 bit.

https://nexus.opendaylight.org/content/repositories/opendaylight.release/org/opendaylight/integration/distributions-serviceprovider/0.1.1/distributions-serviceprovider-0.1.1-osgipackage.zip

Or we can directly install it from linux command line with the following steps:

1- sudo apt-get update
2- sudo apt-get install maven git openjdk-7-jre openjdk-7-jdk
3- git clone http://git.opendaylight.org/gerrit/controller.git
4- cd controller/opendaylight/distribution/opendaylight
5- mvn clean install
6- export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
7- echo JAVA_HOME=/usr/lib/jvm/ java-1.7.0-openjdk-amd64 >> ~/.bashrc
8- cd target/distribution.opendaylight-0.1.0-SNAPSHOT-osgipackage/opendaylight
9- ./run.sh

After the installing, we can access to web UI via http://ip:8080

2- OpenVswitch installation

This virtual switch needs to be installed on the OSUbuntu server 14 – 64 bit with the following steps:

1- sudo apt-get install openvswitch-switch openvswitch-controller

For defining a switch, the command below needs to be executed:

2- ovs-vsctl port-br ovs-br1

For defining OpenFlow version, the following command needs to be executed:

3- ovs-vsctl set bridge br1 protocols=OpenFlow13

For adding a port to the switch:

4- ovs-vsctl add-port ovs-br1 vnet0 # OSSIM : Network based detection

5- ovs-vsctl add-port ovs-br1 vnet1 # Victim
6- ovs-vsctl add-port ovs-br1 vnet2 # Attacker

For establishing connection between controller and switch just simply, the command below should be executed in the command line:

7- ovs-vsctl set-controller ovs-br1 tcp:ip:port

OSSIM installation and configuration

First OSSIM should be downloaded from the link below:

https://www.alienvault.com/products/ossim/download

Then we need to import it into the virtual machine, afterward MySQL service for OSSIM should be started. For integrating SDN controller and OSSIM we need a script to check the OSSIM's database and in case of any incident we can access to the SDN controller and deploy the rule. In order to perform this policy, just we need to put the script below and schedule the OSSIM's crontab to execute the script in an optional interval time.

1- sudo service mysql start
2- Sudo nano SDN.sh

```
#!/bin/bash
mysql -uroot -prCra6pNEfR alienvault -e "SELECT * FROM incident_alarm" |awk '{print $3,$4}'|awk 'NR>1'|awk 'END{print}' |awk '{print $1}'
mysql -uroot -prCra6pNEfR alienvault -e "SELECT * FROM incident_alarm" |awk '{print $3,$4}'|awk 'NR>1'|awk 'END{print}' |awk '{print $2}'
/usr/bin/curl -u admin:admin -H 'Content-type: application/json' -X PUT -d "{\"installInHw\":\"true\", \"name\":\"flows1$w\", \"node\": {\"id\":\"00:00:00:10:f3:29:b1:81\", \"type\":\"OF\"}, \"etherType\": \"0x800\", \"nwSrc\": \"${SRC[$i]}\", \"nwDst\": \"${DST[$i]}\", \"priority\":\"65535\",\"actions\":[\"DROP\"]}" "http://192.168.10.10:8080/controller/nb/v2/flowprogrammer/default/node/OF/00:00:00:10:f3:29:b1:81/staticFlow/flows1$w"
```

In order to create manual rule ticket from OSSIM web UI the following code has been written in the php programming language and Linux bash script:

```
<?php
$db_ip=trim(shell_exec("/bin/cat /etc/ossim/ossim_setup.conf | grep db_ip| awk -F= '{print $2}'"));
$db_user=trim(shell_exec("/bin/cat /etc/ossim/ossim_setup.conf | grep '^user='| awk -F= '{print $2}'"));
$db_pass=trim(shell_exec("/bin/cat /etc/ossim/ossim_setup.conf | grep '^pass='| awk -F= '{print $2}'"));
$db_id=trim(shell_exec("/bin/cat /usr/share/ossim/www/forensics/net.sh | grep '^#LAST_ID='| awk -F= '{print $2}'"));
$last_id=$db_id;

if (!$con = mysql_connect($db_ip,$db_user,$db_pass)) {
   echo 'Could not connect to mysql';
   exit;
}

if (!mysql_select_db('alienvault', $con)) {
   echo 'Could not select database';
   exit;
}

$qry      = "SELECT i.id,ia.src_ips,ia.dst_ips,ie.src_ips as src_ipse,ie.dst_ips as dst_ipse FROM incident i LEFT JOIN incident_alarm ia ON i.id=ia.incident_id LEFT JOIN incident_event ie ON i.id=ie.incident_id WHERE i.id > $db_id";
$result = mysql_query($qry,$con);

if (!$result) {
   echo "DB Error, could not query the database\n";
   echo 'MySQL Error: ' . mysql_error();
   exit;
}
```

```
$id= array();
$src=array();
$dst=array();

while ($row = mysql_fetch_array($result)) {
  if (($row['id'] && $row['src_ips'] && $row['dst_ips']) || ($row['id'] &&
$row['src_ipse'] && $row['dst_ipse'])){
        $id[]=$row['id'];
        if ($row['src_ips'])
                $src[]=$row['src_ips'];
        else
                $src[]=$row['src_ipse'];
    if ($row['dst_ips'])
                $dst[]=$row['dst_ips'];
        else
                $dst[]=$row['dst_ipse'];
        $last_id=$row['id'];
  }
}
$id=implode(',',$id);
$src=implode(',',$src);
$dst=implode(',',$dst);
if (!empty($id))
        shell_exec("sudo /usr/share/ossim/www/forensics/net.sh Sdn $id $src
$dst");
shell_exec("/bin/sed -i 's/^#LAST_ID=.*/#LAST_ID=$last_id/g'
/usr/share/ossim/www/forensics/net.sh");
mysql_free_result($result);
?>
```

When the source and destination's IP address are specified from the web UI of SIEM then they are set into variables and parsed to the script below; thus, according the IP address a flow is added to the SDN controller and the defined action is executed.

Net.sh

```
#!/bin/bash

## SET VARIABLES ##
 ###################
 ID="$1"
 SRC_IP="$2"
 DST_IP="$3"
 IFS=',' read -ra SRC «< "$SRC_IP"
 IFS=',' read -ra DST «< "$DST_IP"
## ADD FLOW ##
 ##############
 i=0
 for w in $(echo "$ID" | tr ',' ' ') ; do
    /usr/bin/curl -u admin:admin -H 'Content-type: application/json' -X PUT -d
"{\"installInHw\":\"true\", \"name\":\"flows1$w\", \"node\":
{\"id\":\"00:00:2c:60:0c:94:84:99\", \"type\":\"OF\"}, \"etherType\": \"0x800\",
\"nwSrc\": \"${SRC[$i]}\", \"nwDst\": \"${DST[$i]}\",
\"priority\":\"65535\",\"actions\":[\"DROP\"]}"
"http://localhost:8080/controller/nb/v2/flowprogrammer/default/node/OF/00:00
:2c:60:0c:94:84:99/staticFlow/flows1$w"
    i=$i+1
 done
#################
```

The programming code is accessible from the following links:

https://drive.google.com/file/d/0B_aE3moMH9wwcnBBZGZUdVVFbTg/view?usp=sharing

https://drive.google.com/open?id=0B_aE3moMH9wwMXYwZmRHTkdYcTA

The test for integrating SDN with SIEM result video has been uploaded in this link:

https://www.youtube.com/watch?v=eqaUkvFQJ48&feature=youtu.be

References

[1] "Open Networking Foundation :Dedicated to SDN," Open Networking Foundation, 2015. [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition. [Accessed 25 11 2015].

[2] Kristian Slavov, Daniel Migault, Makan Pourzandi, "IDENTIFYING AND ADDRESSING THE VULNERABILITIES AND SECURITY ISSUES OF SDN," Ericsson, 2015.

[3] Mike McBride, Marc Cohn , Smita Deshpande, "https://www.opennetworking.org," 8 10 2013. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-security-data-center.pdf. [Accessed 20 11 2015].

[4] Sandra Scott-Hayward, Christopher Kane and Sakir Sezer, "OperationCheckpoint:SDN Application Control," in *IEEE 22nd International Conference on Network Protocols*, Washington, DC, USA, 2014 .

[5] Kevin Benton, L. Jean Camp ,Chris Small, "OpenFlow Vulnerability Assessment," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, New York, 2013.

[6] Jérôme François, Issam Aib, "A Collaborative Protection Network for the Detection of Flooding DDoS Attacks," *IEEE/ACM TRANSACTIONS ON NETWORKING,* vol. 20, no. 6, pp. 1828 - 1841, December 2012.

[7] Lisa Schehlmann, Sebastian Abt and Harald Baier, "Blessing or Curse? Revisiting Security Aspects of Software-Defined Networking," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, Rio de Janeiro, 2014.

[8] Rowan Kl¨oti, Vasileios Kotronis, Paul Smith, "OpenFlow: A Security Analysis," in *International Conference on Network Protocols (ICNP)*, 2013.

[9] Teemu, Martin, Natasha, Jeremy, Leon, Min,Rajiv,Yuichiro,Hiroaki,Takayuki, "Onix: A Distributed Control Platform for Large-scale Production Networks," in *in Proceedings of*, Berkeley, CA, 2010.

[10] R. Chua, "www.sdxcentral.com," 2013. [Online]. Available: https://www.sdxcentral.com/articles/news/juniper-new-sdn-strategy-contrail-role/2013/01/. [Accessed 10 12 2015].

[11] OpenDaylight, "OpenDaylight SDN Controller Platform," OpenDaylight, 19 03 2015. [Online]. Available: https://wiki.opendaylight.org/view/OpenDaylight_SDN_Controller_Platform_(OSCP):Main. [Accessed 12 12 2015].

[12] ONF, "“Open Networking Foundation.”," 2015. [Online]. Available: https://www.opennetworking.org/. [Accessed 13 12 2015].

[13] S. LLC, "what-is-openflow," Juniper networks, 2012. [Online]. Available: https://www.sdxcentral.com/resources/sdn/what-is-openflow/. [Accessed 15 12 2015].

[14] M. Kassner, "TechRepublic," 2015. [Online]. Available: http://www.techrepublic.com/blog/it-security/software-defined-networking-how-it-affects-network-security/. [Accessed 20 12 2015].

[15] G. V. Nikolaev, *Network Monitoring with Software Defined networking Towards: OpenFlow network monitoring,* Faculty of Electrical Engineering, Mathematics and Computer Science, Delft university of Techonolgy, 2013.

[16] S. Zaghloul, "SDN 101: Software Defined Networking - IBM Academic Initiatives," IBM, 2014.

[17] ONF, "OpenFlow Switch Specification (version 1.5.1)," 2015.

[18] Ben Pfaff,Justin Pettit,Teemu Koponen, "Extending Networking into the Virtualization Layer," in *HotNets*, New York , 2009.

[19] Andreas Voellmy, Ashish Agarwal, Paul Hudak, "Nettle: Functional Reactive Programming of OpenFlow Networks," Yale University, 2010.

[20] Bruno Batista, Marcial Fernandez, "A New Policy Specification Language to SDN OpenFlow-based Networks," *International Journal On Advances in Networks and Services,* vol. 7, p. 163 to 172, 2014.

[21] PLVision, "Software-Defined Networking," PLVision, 2015. [Online]. Available: http://plvision.eu/expertise/networking/software-defined-networking/. [Accessed 10 12 2015].

[22] Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Verı´ssimo,Christian Esteve Rothenberg,Siamak Azodolmolky, "Software-Defined Networking:A Comprehensive Survey," *Proceedings of the IEEE,* vol. 103, no. 1, p. 31, January 2015.

[23] Xinyang Feng, Jianjing Shen , Ying Fan, "REST: An alternative to RPC for Web services architecture," in *Future Information Networks*, Beijing, 2009.

[24] S. RAO, "OpenDaylight, the Most Documented Controller," 2015. [Online]. Available: http://thenewstack.io/sdn-series-part-vi-opendaylight/. [Accessed 10 05 2016].

[25] ONF, "SDN Architecture Overview," Open Networking Foundation, 2013.

[26] A. Shostack, "STRIDE : Chapter 3," in *Threat Modeling: Designing for Security*, John Wiley & Sons, 2014, p. 624.

[27] "Computer Security Threat Sources (Attacks)," comptechdoc, [Online]. Available: http://www.comptechdoc.org/man/Business_guide/risk-assessment/securitythreats.html. [Accessed 10 02 2016].

[28] OWASP, "Category:Threat Agent," [Online]. Available: https://www.owasp.org/index.php/Category:Threat_Agent. [Accessed 02 02 2016].

[29] Naved Ahmed , Raimundas Matulevicius, "A Method for Eliciting Security Requirements from the Business Process Models," *Joint Proceedings of the CAiSE 2014 Forum and CAiSE 2014*

Doctoral Consortium, vol. 1164, 18 06 2014.

[30] R. B. Natan, Implementing Database Security and Auditing: Includes Examples for Oracle, SQL Server, DB2 UDB, Sybase, MA, USA: Digital Press Newton, 2005.

[31] N. MAYER, "Model-Based Management of Information System Security Risk," Doctoral Thesis in Computer Science, Belgium, 2009.

[32] Christopher Monsanto, Joshua Reich, Nate Foster,Jennifer Rexford, David Walker, "Composing Software-Defined Networks," in *In: NSDI*, 2013.

[33] Adnan Akhunzada, Abdullah Gani, Nor Badrul Anuar, Ahmed Abdelaziz, Muhammad Khurram Khan, Amir Hayat, Samee U.Khan, "Secure and dependable software defined networks," *Journal of Network and Computer Applications,* 2015.

[34] Xitao Wen, Yan Chen, Chengchen Hu,Chao Shi, Yi Wang, "Towards A Secure Controller Platform for OpenFlow Applications," in *second ACM SIGCOMM workshop on hot topics in SDN*, Hong Kong, 2013.

[35] J. M. Dover, "A denial of service attack against the Open Floodlight SDN controller," Dover Networks LLC.

[36] Diego Kreutz, Fernando M. V. Ramos, Paulo Verissimo, "Towards Secure and Dependable Software-Defined Networks," in *second ACM SIGCOMM workshop on hot topics in SDN*, Hong Kong, 2013.

[37] R. S. F. Community, "COMPONENT-BASED SOFTWARE DEFINED NETWORKING FRAMEWORK," Ryu, 2014. [Online]. Available: https://osrg.github.io/ryu/.

[38] Incapsula, "Slowloris : DDoS Attack Glossary," Incapsula, [Online]. Available: https://www.incapsula.com/ddos/attack-glossary/slowloris.html. [Accessed 07 05 2016].

[39] indiandragon, "How to perform a HTTP flooding using a low bandwidth attack," [Online]. Available: http://blog.indiandragon.in/2012/04/how-to-perform-a-http-flooding-using-a-low-bandwidth-attack.html. [Accessed 38 04 2016].

[40] D. Miessler, "The Carriage Return and Line Feed Characters," [Online]. Available: https://danielmiessler.com/study/crlf/. [Accessed 18 05 2016].

[41] OWASP, "How to Build an HTTP Request Validation Engine for Your J2EE Application," OWASP, [Online]. Available: https://www.owasp.org/index.php/How_to_Build_an_HTTP_Request_Validation_Engine_for_Your_J2EE_Application#Simple_and_Obvious_API. [Accessed 13 05 2016].

[42] A. Kibirkstis, "What is The Role of a SIEM in Detecting Events of Interest?," November 2009. [Online]. Available: https://www.sans.org/security-resources/idfaq/what-is-the-role-of-a-siem-in-detecting-events-of-interest/5/10. [Accessed 15 03 2016].

[43] Sunil Gupta, Dr. Kees Leune, "Logging and Monitoring to Detect Network Intrusions and Compliance Violations in the Environment," SANS Institute InfoSec Reading Room, 2012.

[44] Prof. Dr. K.-O. Detken, T. Rix,Prof. Dr. C. Kleiner,B. Hellmann,L. Renners, "SIEM approach for a higher level of IT security in enterprise networks," in *The 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems*, Warsaw, Poland, 2015.

[45] A. :. W. PAPER, "OSSIM vs. USM: A Comparison of Open Source vs. Commercial," AlienVault, 2016.

[46] alienvault, "About the USM Plugin Types," [Online]. Available: https://www.alienvault.com/documentation/usm-v5/plugin-management/plugin-comp-types.htm#Detector. [Accessed 05 04 2016].

[47] Maciej Ku´zniary, Peter Perešíniy, Dejan Kosti´cz, "What you need to know about SDN control and data planes," EPFL Technical Report EPFL-REPORT-199497, 2014.

[48] Flowgrammable, "OpenFlow: Switch Anatomy," Flowgrammable, [Online]. Available: http://flowgrammable.org/sdn/openflow/#tab_switch. [Accessed 15 03 2016].

[49] ONF, "OpenFlow Switch Specification: Version 1.3.1," 2012.