Risk based Security Enforcement in Software Defined Network

Bata Krishna Tripathy, Debi Prasad Das, Swagat Kumar Jena, Padmalochan Bera

PII: S0167-4048(18)30191-3
DOI: 10.1016/j.cose.2018.07.010
Reference: COSE 1372

To appear in: *Computers & Security*

Received date: 5 March 2018
Revised date: 1 July 2018
Accepted date: 24 July 2018

Please cite this article as: Bata Krishna Tripathy, Debi Prasad Das, Swagat Kumar Jena, Padmalochan Bera, Risk based Security Enforcement in Software Defined Network, *Computers & Security* (2018), doi: 10.1016/j.cose.2018.07.010

Risk based Security Enforcement in Software Defined Network

Bata Krishna Tripathy[a,*], Debi Prasad Das[b], Swagat Kumar Jena[a], Padmalochan Bera[a]

[a]*Indian Institute of Technology Bhubaneswar, India*
[b]*National Institute of Technology Rourkela, India*

Abstract

Software Defined Network (SDN) paradigm provides intelligent and efficient management of different network control functions (NF) depending on changes in traffic behavior, service providers' requirements and application context. However, the logical centralization of controllers' functions opens up challenges towards enforcing security perimeter over the underlying network and the assets involved. In this paper, we propose a risk assessment model for pro-active secure flow control and routing of traffic in SDN. The proposed model determines threat value of different SDN entities by analyzing vulnerability and exposure with respect to Common Vulnerability Scoring System (CVSS). The risk of a given traffic is calculated as cumulative threat values of the SDN entities that guides the flow and routing control functions in generating secure flow rules for the forwarding switches. The efficacy of the proposed model is demonstrated through extensive case studies of an enterprise network.

Keywords: Software Defined Network (SDN), Network control functions (NF), Common Vulnerability Scoring System (CVSS), vulnerability, exposure, threat, risk

1. Introduction

Software Defined Networking (SDN) is an emerging networking paradigm that allows intelligent and efficient management of network control functions. It allows execution of different network control functions as a logically centralized controller by decoupling them from the underlying forwarding network [1] consisting of various network devices, e.g., switches, routers, access points, etc. The controller provides better flexibility and configuration control to the users with easy and on-demand dynamic configuration of the network and its resources [2]. The SDN model shown in Figure 1 provides several benefits over the traditional network such as (i) Simple and reliable network, (ii) Programmability feature, (iii) Flexible device configuration and troubleshooting, and (iv) Virtualization of the network. Due to these extensive features, Software Defined Networking has attained significant attention to research community starting from academics to industries and has several applications in ranging from data centers to wide area networking, cloud computing, Internet of Things, Mobile Ad hoc Networking, cellular networking, etc. [3].

*Corresponding author: Bata Krishna Tripathy
Email addresses: bt10@iitbbs.ac.in (Bata Krishna Tripathy), debi.das@itron.com (Debi Prasad

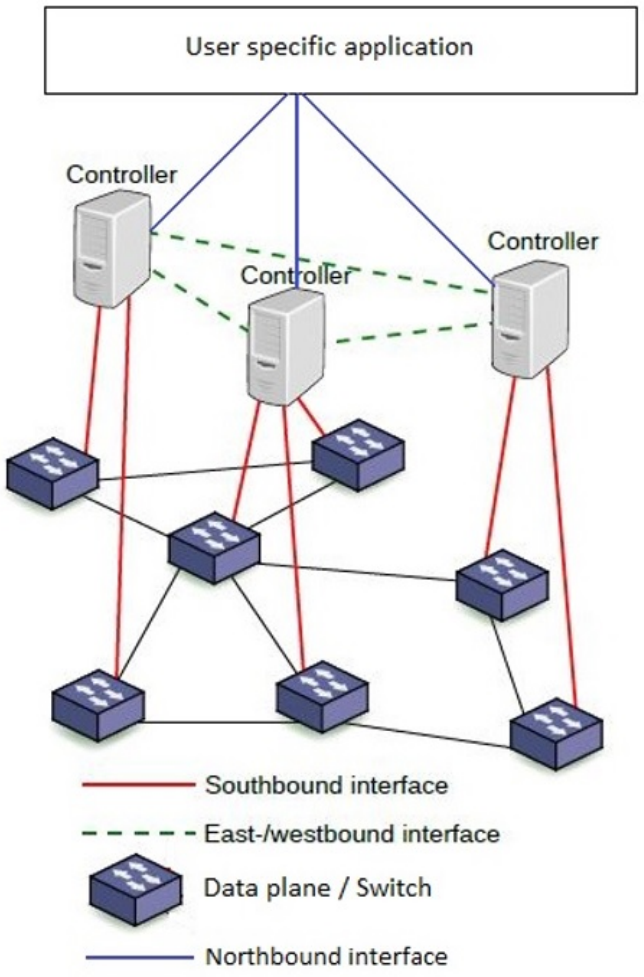

Figure 1: A model of Software Defined Networking

Software Defined Networking (SDN) enables efficient service provisioning to end users from various enterprise applications based on Service Level Agreements (SLAs) and dynamic requirements in terms of policies by maintaining the global view of the network. Among various SDN enabled communication protocol, the majority of the SDN applications implement OpenFlow protocol [48] to support necessary interaction between the controllers and the switches as OpenFlow is a vendor-independent standard and hence allows for interoperability between heterogeneous networking devices.

Despite the advantages provided by SDN, various performance and security challenges have been major concerns since its evolution. The performance and security issues those need to be addressed for efficient deployment of SDN are the management of complex policies

Das), skj11@iitbbs.ac.in (Swagat Kumar Jena), plb@iitbbs.ac.in (Padmalochan Bera)

in a simple interface, networking delay, lack of standardized interfaces between SDN layers, load balancing, packet scheduling, etc. In addition, there exist various open problems to utilize the benefits of SDN [5]. The most important problems lie in: (i) usable, reliable and efficient network service offerings [6]; (ii) extensible control function execution platform with the change in network size and requirements [7]; and (iii) end-to-end security enforcement to network services. *The thrust of this paper is to provide a seamless solution for enforcing end-to-end security in SDN.*

A number of security solutions have been contributed by the research communities to address these security issues. These vary from access control mechanisms to encryption-based authentication schemes. On the other hand, a number of security approaches have been proposed to enhance the SDN framework using middlebox architectures. In addition, few researchers focus on developing Intrusion Detection System and Intrusion Prevention System to ensure security in SDN. Whereas other classes of researchers aim at different aspects of security approaches such as developing a secure flow specification language, inspecting packet level threats, certificate authentication etc. to detect various attacks such as Denial of Service, Spoofing, Man-in-the-middle attacks, etc.

However, the state-of-art SDN security solutions do not provide an end-to-end secure flow of packets across the network segments by assessing the behavior of the traffic with respect to different SDN components as the internal architecture of SDN as well as the heterogeneous policy rules enforced by distributed policy enforcing servers impose several security challenges.

In the next subsection, we discuss the challenges of security enforcement in SDN with a motivating example.

1.1. Motivation

The open user-control, ubiquitous execution of network functions and centralized control management introduce various security threats in different levels of Software-defined network architecture. The major security challenges in SDN are demonstrated with an example here.

Figure 2 shows an SDN environment for a segment of an enterprise network. Here, the Network Application Server (AP) generates network access control and flow configuration policies based on requirements and those policies are realized dynamically in the corresponding network control functions. At the Northbound interface (controller-application interface), security challenges with respect to giving permission to alter the network policies to an application may cause harm to the whole network. In this scenario, there is a potential possibility of AP being compromised as it is exposed to external users. This might lead to altering the policy rules for the network functions. As a result, the network functions might be realized incorrectly that may allow propagation of various attacks across the network. These attacks can extend to Denial of Service (DoS), code injection, and hidden tunnel.

Now, consider the scenario, where the flow control function is driven by the rules generated from Network Application Server (AP) and Network Management Server (NMS). In such ubiquitous computing scenarios, there is a possibility of conflicts amongst the flow rules generated by these servers. This, in turn, might cause compromising the NMS by the AP leading to security and performance violations in the network.

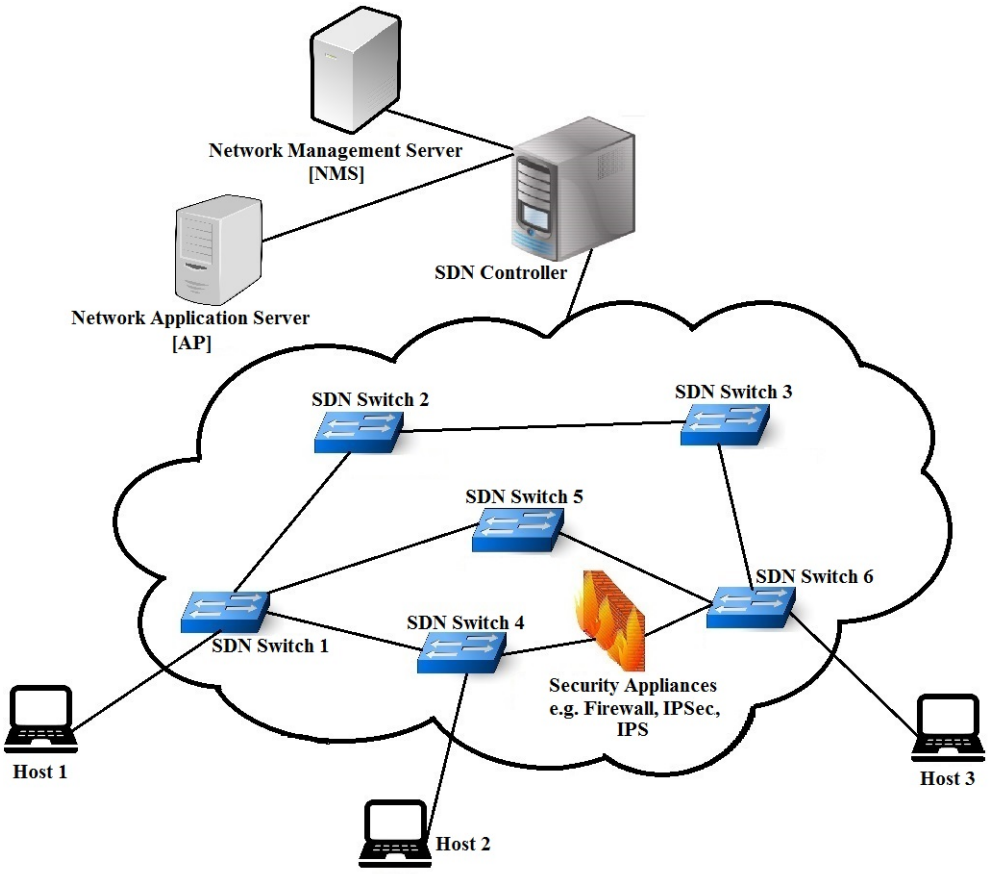

Figure 2: A Security Threat Scenario

The controller generates appropriate flow entries from the set of network policy rules and pushes them into the flow tables. An SDN cannot distinguish the flow entry of a newly generated application from an older one. So, there is a possibility that a new network policy generated by an application server can bypass and override the security policy predefined by the security administrator.

Furthermore, there may be vulnerable applications running in end hosts, controllers, and switches leading to various state-of-art attacks. In addition, the attack paths may be created across the network segment as the switches along the traffic route may have various open ports. In some cases, the network traffic containing malware, trojan or spams originating from corrupted hosts (due to vulnerabilities) might result in compromised network functions that can be manifested as different end-point attacks in the networks.

However, the existing security solutions for SDN do not support analyzing the behavior of the traffic with respect to different SDN components that may lead to various vulnerabilities and risks which may have a huge impact on the organizational assets. Therefore, an appropriate security enforcement mechanism needs to be introduced to pro-actively analyze the risks of traffics and creating strong isolation between the network functions. The main

objective of this paper is to provide an end-to-end security mechanism for SDN. The major contribution of our work lies in developing a novel network security enforcement function for SDN based on risk assessment.

The proposed security function determines threat value of different SDN entities by analyzing vulnerability and exposure with respect to Common Vulnerability Scoring System (CVSS) [39]. The risk of a given traffic is calculated as cumulative threat values of the SDN entities that guides the flow controller in generating secure flow rules for the forwarding switches.

The remainder of the paper is organized as follows. In Section 2, we discuss the relevant works related to the security enforcement mechanisms in SDN. In Section 3, we present an overview of our proposed security enforcement functions for SDN briefly. In Section 4, we discuss the *Risk_Analyze* function in detail, that assesses the risk of different SDN entities with respect to a given traffic request. Then, we evaluate our proposed security enforcement functions with a case study in Section 5. Finally, we conclude the paper in Section 6.

2. Related Work

Software Defined Network simplifies network deployment and operation along with providing a programmable platform for managing enterprise and carrier networks. However, with the introduction of open interfaces in SDN and knowledge of protocols in different Application programming interfaces, the door is thrown open to the attackers. A number of research works have been contributed in the area of security enforcement in Software Defined Networking environment.

The research communities focus on exploiting SDN to improve network security with dynamic detection and mitigation of suspicious activity in a network. Kreutz et al. [4] in 2013 reported that the centralized nature of controller and the network programmability scope introduce new threats in SDN. Mehdi et al. [9] propose the implementation of SDN to provide better security in terms of intrusion detection in a small enterprise environment. OpenSAFE [10] addresses security issues for an arbitrary direction of traffic with a novel flow specification language, namely ALARMS. FRESCO [11] presents a security application development framework with different security enabled modular functions that implement python based flow monitoring and threat detection functions. FleXam [15] facilitates packet level inspection to detect and mitigate security threats like botnet and worm propagation.

In addition, various security issues of SDN itself have attracted the research communities. NICE [16] secures OpenFlow Applications using intrusion detection with the help of attack graphs based analytical models. However, such path exploration approaches do not cope well with extensible applications. FlowChecker [17] exploits FlowVisor [18] to detect inconsistencies in policies from multiple applications or the multi-controller environment. FLOVER [19] deals with conflicting flow entries with predefined network policies using assertion sets and modulo theories. FortNOX [20] resolves conflicts in flow entry installation with role-based and signature-based authentication.

Another class of researchers aims at developing encryption approach based security solutions for SDN. The authors in [27] proposed an Identity-Based Cryptography (IBC) protocol

based mechanism for ensuring secure southbound and east/west-bound communication in a multi-domain distributed SDN environment. The work [28] presented a security solution in which the controller provides network information only for authorized programs through a safe REST API [30] based on the NTRU algorithm [29] and the NSS digital signature [31]. By analyzing various threats originating from different SDN layers, the literature [32] presented a different SDN security framework based on attribute-based encryption that enables an improved access control method. Another work [33] developed a different authentication scheme by using the Elliptic Curve Cryptography on the basis of PKI certificate verification for ensuring a secure communication in distributed controller environment.

Gabriel et al. [21] proposed a technique for handling Distributed Denial of Service (DDoS) attack in an SDN environment, by assessing risk through the means of a cyber-defense system. HiFIND [8] is a highly secured technique that prevents SDN platform from DDoS attacks for high-density data packets providing protection to customers and service provider. SN-SECurity Architecture (SN-SECA) [22] presents a formal security framework that integrates and validates different security parameters in the SDN/NFV design and implementation. The work in [25] presented a detect and defense control function called as SDN sEcure COntroller (SECO) to protect against Denial of Service (DoS) attack originating from switches or hosts. SECO uses the switch port statistics performance and the attack source positioning feature from the global view of the network for this purpose. The authors in [26] proposed a security scheme for SDN that consists of different components, i.e., a mapping algorithm, take-over process, synchronization messages, heartbeat messages, and protective mode to prevent against DoS attack.

Avant guard in literature [12] uses connection migration and actuating triggers as security features to provide security between the forwarding and control plane along with improving the response rate of the controller to the forwarding plane traffic requests. The authors in [14] proposed a model to analyze the potential threats when data plane communicates with the controller using the OpenFlow protocol using STRIDE [13] and attack trees to detect various attacks. The authors in [23] proposed a different approach to analyze and model Forwarding and Control planes Separation Network Structure (FCSNS) in SDN based on STRIDE [13], Petri net, and Attack tree models. In another work, Controller DAC [24] presented a controller-independent dynamic access control scheme to ensure protection of SDN controllers against malicious access by applications through different APIs (east/westbound and north/southbound).

On the other hand, few researchers focused on developing security middle-boxes for SDN execution platform accomplishing its programmability feature. Slick framework [36] presents a centralized controller for implementing and migrating functions onto customized middle-boxes allowing applications for routing security requirements based traffic request. FlowTags architecture [37] is another minimally modified middle-box that interacts with an SDN controller through a FlowTags (traffic flow information embedded in packet headers) Application Programming Interface (API). Another work, the SIMPLE policy enforcement layer [38] requires no modification to middlebox or SDN in contrast to [36] and [37]. However, middlebox security solutions incur significant overhead for security enforcement in SDN.

However, the state-of-art security mechanisms don't ensure end-to-end security enforce-

ment with the systematic analysis of real-time and heterogeneous traffic, and assessment of the behavior of different SDN entities. Our proposed network security enforcement functions will effectively and efficiently address these problems.

In the next section, we present an overview of our proposed security enforcement mechanism for SDN.

3. Proposed Security Enforcement Framework for SDN

This section presents our proposed SDN security enforcement framework. We propose a novel security enforcement function called *Risk_Analyze* that along with the policy control functions: *Trust_Verify*, *Policy_Conflict_Resolve*, and *Policy_Consistency_Check* [34] provides end-to-end security in SDN. Here, the main emphasis of the paper is on the *Risk_Analyze* function. Figure 3 shows the overall architecture of the SDN control functions and the process of traffic flow through these functional modules. The proposed security enforcement functions are briefly discussed in following subsections.

3.1. Trust_Verify

In SDN, different network application servers and management servers generate policy rules for different network applications dynamically as per the requirements. The *Trust_Verify* function [34] checks the following properties of the policy enforcing servers and certifies them.

- The certificate has been issued by a valid certificate authority (CA).
- The certificate has not been expired (or been revoked).
- The names listed in the certificate match the domain names.

In addition, it performs verification of PKI (Public Key Infrastructure) certificate and trust of the links (interfaces/ports). Hence, the *Trust_Verify* function ensures defense against security violations through any compromised application or management server. The detailed algorithm is demonstrated in our previous work [34].

3.2. Policy_Conflict_Resolve

Policy_Conflict_Resolve function [34] detects the potential conflicts between the heterogeneous policy rules received by SDN control plane using pattern matching and resolves them by using the concept of rule override. The override function uses artificial intelligence based algorithms to resolve the conflicts between heterogeneous policies. The *Policy_Conflict_Resolve* function manages these heterogeneous conflicts by prioritizing the levels of administrators in correspondence to different Application and Management Servers based on their roles. For example, a Network Management Server (NMS) usually has higher priority than an Application Server (AP) to serve the traffic as per the requirements. Hence, policy rules set by NMS overrides the policy rules set by AP. The detailed algorithm is presented in our previous work [34].

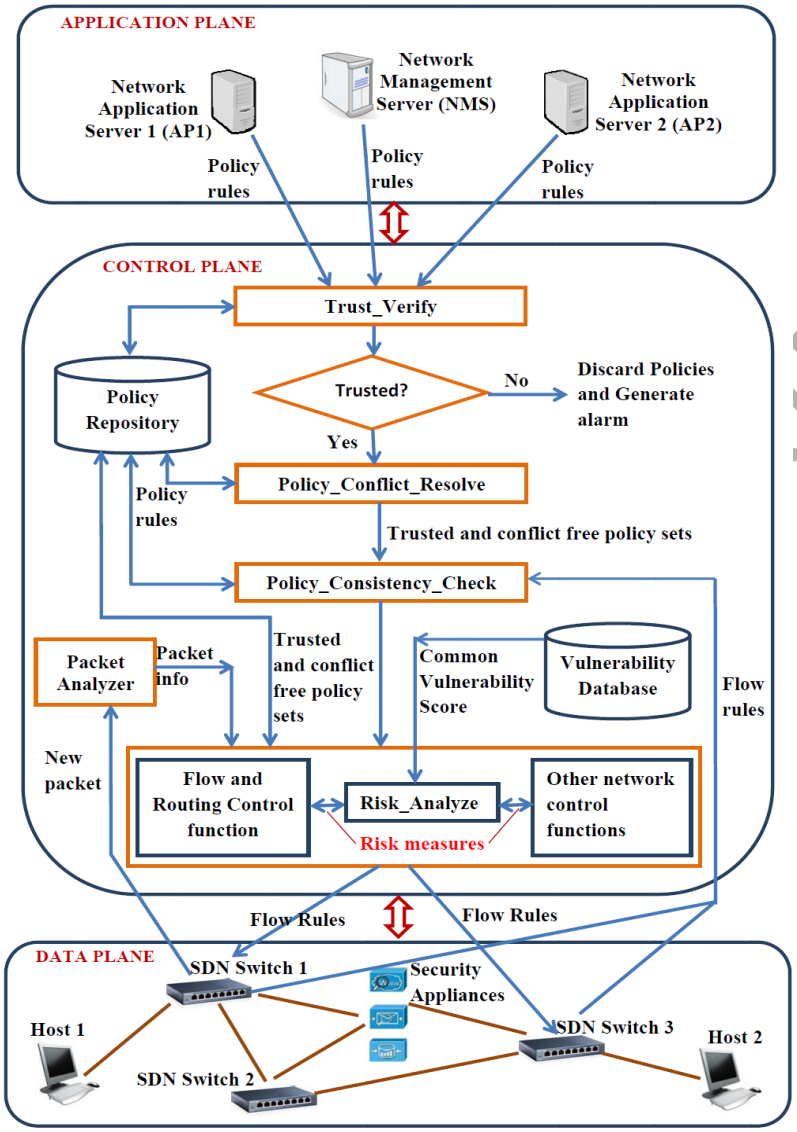

Figure 3: Proposed Risk based Security Enforcement Framework for SDN

3.3. *Policy_Consistency_Check*

SDN platform supports dynamic changes in the application layer by different administrators in the form of requirements or high-level Service Level Agreements (SLAs). The changes may be due to altering existing policies, adding or removing policies, installing or removing applications, etc. So, if any policy is modified in the application layer, it must be reflected in the flow tables of the data plane switches in order to adapt to the changes in requirements and to allow correct propagation of traffic across the network.

The *Policy_Consistency_Check* function proposed in our previous work [34] detects the inconsistency between the application layer and data layer. It checks the satisfiability of the

existing flow rules in the SDN switches and the updated conflict-free policy rule sets. This process is triggered by any change in the Policy repository and at each packet arrival in the SDN switches.

3.4. Risk_Analyze

The three secure control functions, i.e., *Trust_Verify*, *Policy_Conflict_Resolve*, and *Policy_Consistency_Check* compositely enable secure and efficient policy enforcement in SDN. However, there is possibility of security breaches due to misconfigurations, vulnerabilities, and backdoors in different entities (i.e., *compromised end hosts*, *compromised switches* and *compromised controller*). Our proposed *Risk_Analyze* first extracts threat models for different entities for a given traffic request considering vulnerability and exposure of each of the entities. Then, it calculates the overall risk of the corresponding traffic involved using the weighted sum of the threat values and criticality of the traffic request. The calculated risk measures for the related traffic are fed to the routing and flow control functions for generating correct routing and flow rules. This will ensure secure flow and routing of traffic in SDN.

We have already discussed the three policy control functions, i.e., *Trust_Verify*, *Policy_Conflict_Resolve*, and *Policy_Consistency_Check* in details in our previous work [34]. The main thrust of this paper lies in developing *Risk_Analyze* function that ensures end-to-end security for a given traffic request in SDN environment. The next section describes our proposed *Risk_Analyze* function in detail.

4. *Risk_Analyze*: The Risk Assessment Function for SDN

This section explains our proposed risk assessment model for SDN using the *Risk_Analyze* function in detail. Our proposed *Risk_Analyze* function considers the following parameters in order to assess the risk of various types of traffic.

(a) Vulnerability: It is defined as a software and hardware level weakness in the network entities, which may allow an attacker to reduce the information assurance of the entities and the underlying network [40]. We use Common Vulnerability Scoring System (CVSS) [39] for defining vulnerability of the network entities.

(b) Exposure: It is defined as the state or condition of a network being unprotected and open to the risk of suffering the loss of information [41]. We determine the threat of a network entity as the ratio of the potentially unprotected portion of the entity to the total entity size.

(c) Threat: Threats are potential events for vulnerabilities that might lead to exposure of the network and adversely impact the organizational assets [42]. Vulnerability and exposure of an entity are used to determine its threat value.

(d) Risk: It is a qualitative measure of potential security threat and its impact on the network [43].

The Common Vulnerability Scoring System (CVSS) [39] plays an important role for risk assessment of the network entities in order to ensure secure flow of traffic across the network

segment. The proposed risk assessment module uses a data structure called vulnerability database for this purpose. The vulnerability database is a local repository (offline) stored in the controller and is periodically updated with the recent Common Vulnerability Score (CVS) values of the applications or protocols or services running in different SDN components or entities (controller, end hosts, and switches). The CVS values are computed by extracting necessary metrics from online National Vulnerability Database (NVD) [44] using a script (python).

The recent vulnerability values available in NVD are in XML format which contains two standard scores: V2 and V3 in the form of Common Vulnerability and Exposure (CVE) measures. The detailed process of parsing CVE values from NVD and storing in local vulnerability database as CVS values inside the controller is explained in Figure 4. It is to be noted that in the vulnerability database, there exists exactly one entry of CVS value for an application with its version and the Operating System platform as it is the updated CVSS value of the application parsed from NVD's recent XML file using the script. The structure of an entry in the vulnerability database is $<Application/service/protocol, Version, Operating\ system, CVS\ value>$.

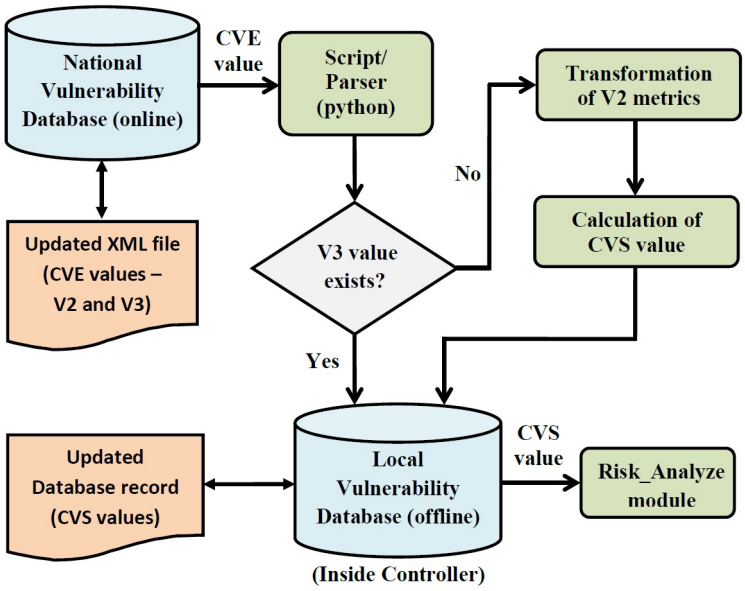

Figure 4: Parsing CVE values from NVD and storing as CVS values in local vulnerability database for risk assessment

Generally, the V3 standard is an improvement over V2 standard as V3 considers the context of attacker's access rights to read/write/execute to exploit the vulnerability and physical manipulation of the affected components. Hence, our proposed risk assessment module uses the V3 version of CVE as its CVS value for necessary risk assessment for secure processing and flow a given traffic request. However, for some older vulnerabilities there exist only V2 values in NVD. In such case, the CVS value for a vulnerability is calculated

in two steps from the available V2 metrics in NVD as discussed below.

(i) **Step 1 - Transformation of V2 metrics:**
In order to compute the overall vulnerability value, CVSS considers certain metrics which define the hardware, software and network level vulnerabilities. The V2 version differs from the V3 version in terms of the metrics and their values considered for overall vulnerability score computation. However, for some older vulnerabilities, V3 value is not available in the NVD. In this scenario, the CVS value for a vulnerability in our solution is estimated from the V2 metrics available in the XML file by appropriately transforming the metrics and their values as shown in Table 1. The transformation is performed as per the CVSS V2 and V3 standards [45], [46].

These metrics after the transformation process are then used for the necessary CVS computation in the proposed mechanism. The estimation of CVS value for a vulnerability is performed as explained below in the subsequent step.

(ii) **Step 2 - Calculation of CVS values:**
The CVS value for a vulnerability is determined from the desired metrics obtained in the previous step, using the standard equations for the overall V3 version of CVSS computation [46] with optimization in order to minimize the overhead of the CVS computation process. The procedure of the overall CVS value calculation is illustrated in Figure 5.

The entities of the Software Defined Network architecture might be the potential sources of vulnerabilities in the network [35]. In this paper, we have considered the following SDN entities for risk assessment.

- *compromised end hosts*
- *compromised switches*
- *compromised controller*

In our *Risk_Analyze* function, we extract threat models for different entities with respect to a given traffic request using vulnerability and exposure analysis of those entities. Then, the overall risk of a given traffic request is calculated as cumulative threat values of the SDN entities and criticality of the traffic request.

The following subsections illustrate the threat model and the risk assessment model for different entities of Software Defined Network.

4.1. Threat model for SDN Entities

This subsection presents the threat model for different SDN entities for a specific traffic request. The threat associated with different SDN entities are modeled using the vulnerability and exposure of the entities as follows.

Table 1: Transformation of V2 metrics and their values for CVS computation

V2 metric		Value	Transformed metric	Value
Base metrics				
Exploitability group	Access vector	Local: 0.395	Attack Vector	Local: 0.55
		Adjacent network: 0.646		Adjacent: 0.62
		Network: 1.0		Network: 0.85
	Access complexity	High: 0.35	Attack complexity	High: 0.44
		Medium: 0.61		Medium: 0.62
		Low: 0.71		Low: 0.77
	Authentication	Multiple: 0.45	Privileges Required	High: 0.27
		Single: 0.56		Low: 0.62
		None: 0.704		None: 0.85
Impact group	Confidentiality, Integrity, and Availability	None: 0.0	Confidentiality, Integrity, and Availability	None: 0.0
		Partial: 0.275		Low: 0.22
		Complete: 0.66		High: 0.56
Temporal metrics				
	Exploitability	Unproven: 0.85	Exploitability	Unproven: 0.91
		Proof-of-concept: 0.9		Proof-of-concept: 0.94
		Functional: 0.95		Functional: 0.97
		High: 1.0		High: 1.0
	Remediation level	Official fix: 0.87	Remediation level	Official fix: 0.95
		Temporary fix: 0.90		Temporary fix: 0.96
		Workaround: 0.95		Workaround: 0.97
		Unavailable: 1.0		Unavailable: 1.0
	Report confidence	Unconfirmed: 0.90	Report confidence	Unknown: 0.92
		Uncorroborated: 0.95		Reasonable: 0.96
		Confirmed: 1.0		Confirmed: 1.0
Environmental metrics				
General Modifiers	Collateral Damage Potential	None: 0	Attack Vector	None: 0
		Low (light loss): 0.1		Physical: 0.2
		Low-medium: 0.3		Local: 0.55
		Medium-high: 0.4		Adjacent network: 0.62
		High (catastrophic loss): 0.5		Network: 0.85
	Target Distribution	None: 0	Attack complexity	None: 0
		Low: 0.25		Low: 0.77
		Medium: 0.75		Medium: 0.62
		High: 1.0		High: 0.44
Impact subscore modifier	Confidentiality, Integrity, and Availability requirements	Low: 0.5	Confidentiality, Integrity, and Availability requirements	Low: 0.5
		Medium: 1.0		Medium: 1.0
		High: 1.51		High: 1.5

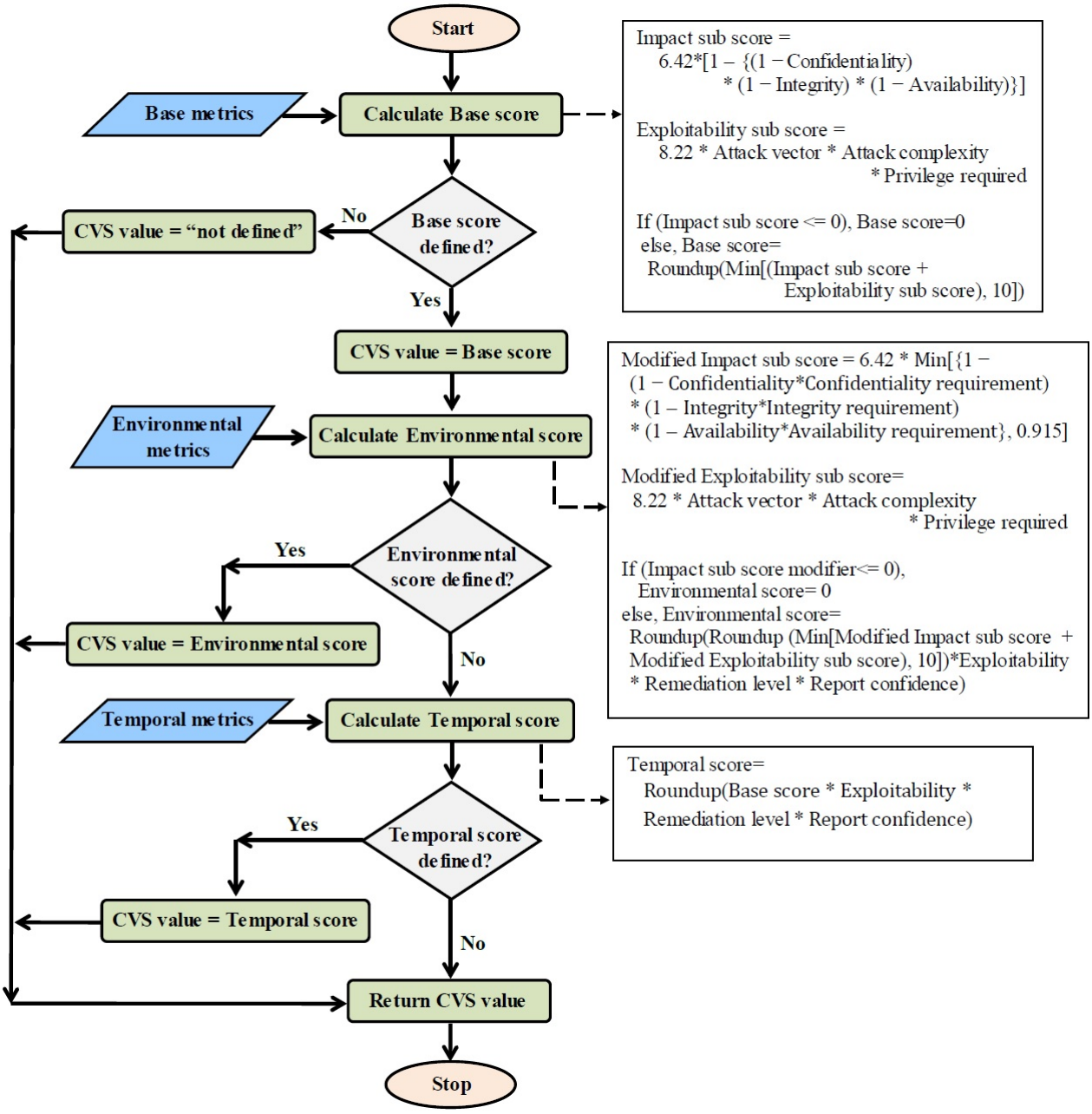

Figure 5: CVS computation of vulnerabilities from the transformed metrics in case of nonavailability of V3 value in NVD

4.1.1. Threat Model for an end host

The threat level of an end host is modeled using the vulnerability and exposure of the host. The model is presented as follows.

(i) Vulnerability of an end host: A number of vulnerable applications such as FTP,

RSH, nmap etc may be running in an end host for processing a specific traffic request. The vulnerability of an end host V_h is calculated as the average of the Common Vulnerability Scores (CVS) of all the applications running on the host extracted from the vulnerability database, i.e.,

$$V_h = \frac{1}{10} * \frac{\sum_{i=1}^{k} CVS_i}{k} \tag{1}$$

where $h\epsilon\{x, y\}$. Here, x and y are the source and destination hosts respectively. CVS_i is the Common Vulnerability Score of the ith application running in the host h, and k is the number of applications running in the host. The average value of the CVS of all applications is divided by 10 in order to normalize the value of V_h to 1 as the CVS of the applications lie between 0 to 10.

(ii) Exposure of an end host: The exposure of an end host E_h is determined considering the number of hosts that may be affected because of the vulnerability in the target end host. Hence, it is computed as,

$$E_h = \frac{n}{N} \tag{2}$$

where n is the number of hosts communicating with the target host and N is the total number of hosts in the network.

(iii) Threat value of an end host: Threat value of an end-host τ_h is calculated as a product of vulnerability and the exposure of the end host for a traffic request t. Mathematically it is defined as follows.

$$\tau_h = V_h * E_h \tag{3}$$

τ_h lies between 0 and 1. $\tau_h = 1$ indicates the entity has high risk of being compromised.

Similarly, we can define threat model for switches and controller with respect to a given traffic request.

4.1.2. Threat Model for a switch

The threat model for an SDN switch is assessed using its vulnerability and exposure analysis as follows.

(i) Vulnerability of a switch: The vulnerability of a switch V_s is determined as the ratio sum of the Common Vulnerability Scores(CVS) of the protocols running on the switch. For example, protocols are Multiprotocol Label Switching (MPLS) protocol, Label Distribution Protocol (LDP), etc. Our model can flexibly accommodate any new protocol used in switches. Mathematically, it is determined as,

$$V_s = \frac{1}{10} * \frac{\sum_{i=1}^{q} CVS_i}{q} \tag{4}$$

where CVS_i is the Common Vulnerability Score of the ith protocol running on the switch s ($0 \leq CVS_i \leq 1$), and q is the number of protocols running on the switch s. Similar to

determining the value of V_h, the ratio sum of the CVS of all protocols running on the switch is divided by 10 so as to normalize the value of V_s to 1.

(ii) Exposure of a switch: Exposure of a switch E_s is defined using the probability of malfunctioning involved with an active connection and total number of ports on that switch. E_s is calculated as,

$$E_s = \frac{x^p * (1-x)^{P-p}}{P} \tag{5}$$

where p is the number of active connections, P is the total number of ports on the switch and x is the probability of an active port getting malfunctioned. For standard ports, i.e., [0-1023], x is varied between 0 and 0.5. On the other hand, for various application specific non-standard ports, $0.5 < x \leq 1$. For example, for running FTP application on standard ports such as 989 and 990, we use $x = 0.3$ whereas for running the same FTP application on a non-standard port such as 4901, we consider $x = 0.6$.

(iii) Threat value of a switch: Vulnerability and exposure of a switch are used to determine its threat value. Threat value τ_s of a switch s is determined as,

$$\tau_s = V_s * E_s \tag{6}$$

In addition, the proposed model determines the threat value of a traffic route. An access route $r(x, y)$ is defined as a sequence of switches $< s_1, s_2, ..., s_m >$ from source host x to destination host y in the network, where $Interface(s_i, s_{i+1}) = T$ and $reachable(x, y) = T$. Threat value τ_r of the route r associated to a traffic t is calculated using the threat values of the switches along that route as follows.

$$\tau_r = \frac{\sum_{i=1}^{m} \tau_{s_i}}{m} \tag{7}$$

where m is the number of switches in the access route r. The threat value of a route lies between 0 and 1.

4.1.3. Threat Model for network controller

The threat model for an SDN controller is evaluated using the vulnerability and exposure analysis of the control functions in the SDN control plane as follows.

(i) Vulnerability of a controller: A controller may run various network control functions such as Load Balancing, Analysis Engine, Control-Plane MainApp, etc. Hence, the vulnerability V_c of a controller c is estimated using the mean of Common Vulnerability Score (CVS) of the control functions running on it. If CVS_i is the Common Vulnerability Score of the ith control function running on the controller c, and j is the number of protocols running on the controller c, then V_c is calculated as,

$$V_c = \frac{1}{10} * \frac{\sum_{i=1}^{j} CVS_i}{j} \tag{8}$$

Similar to equation (1) and (4), V_c is normalized so that its value lies between 0 and 1 as the CVS of the control functions running the controller lie between 0 to 10. The proposed model can be extended to include any new control functions.

(ii) Exposure of a controller: Exposure of a controller E_c is defined as the ratio between the number of dependent network functions (l) (as execution of the control functions depend on the output of other control functions) and the total number of network functions (L). Mathematically, it is calculated as,

$$E_c = \frac{l}{L} \tag{9}$$

(iii) Threat value of a Controller: Threat value of a controller τ_c for a given traffic request t is determined as the product of its vulnerability and exposure measures, i.e.,

$$\tau_c = V_c * E_c \tag{10}$$

The threat value of a controller lies between 0 and 1.

The following subsection illustrates the *Risk_Analyze* that calculates the risk for a given traffic request as the cumulative threat values of the different SDN entities.

4.2. Risk_Analyze

Our proposed risk assessment model first evaluates threat model for different SDN entities as discussed in the previous subsection. Then, the overall risk measure τ_t of a given traffic t is calculated as the cumulative threat values of the end hosts, access route and the control functions involved. Algorithm 1 presents the risk assessment procedure associated with a traffic.

Algorithm 1 uses weights: w_x, w_y, w_r, and w_c for source host x, destination host y, access route r, and the controller c respectively in order to consider the criticality of each SDN entity. w_r is the sum of weights of all the switches $< s_1, s_2, ..., s_m >$ along the route r.

$$w_r = w_{s_1} + w_{s_2} + ... + w_{s_m} \tag{11}$$

These weights are selected based on criticality of the nodes and should be chosen such that their sum must be equal to 1. i.e.,

$$w_x + w_y + w_r + w_c = 1 \tag{12}$$

Total threat value τ_t associated with a traffic t always lies between 0 and 1.

The threat value (τ_t) associated to a traffic t and its criticality (I_t) are used to define the risk (R_t) of the traffic. The criticality of the traffic can be High (H), Medium (M) or Low (L). The criticality of a traffic depends on the impact of the traffic in a specific application context. For example, in a Banking application, transactions have high impact and hence have High importance whereas the generation of logs has medium impact leading to Medium importance. On the other hand, simple query processing has low impact in the context and hence have low importance. So, we consider three different traffic criticality levels; i.e., High(H), Medium(M) and Low(L) respectively for these three types of traffic.

Algorithm 1 *Risk_Analyze* Algorithm

procedure RISK_ANALYZE
  for each traffic t **do**
    analyze packet header
    Entity set E= $\{x, y, r, c\}$ and Route r=$\{s_1, s_2, ..., s_m\}$
    for each entity $i \in E$ - $\{r\}$ **do**
      find V_i
      find E_i
      calculate τ_i=V_i*E_i
    end for
    for each switch $s \in r$ **do**
      find V_s
      find E_s
      calculate τ_s=V_s*E_s
    end for
    calculate $\tau_r = \frac{\sum_{i=1}^{m} \tau_{s_i}}{m}$
    calculate $\tau_t = w_x{}^*\tau_x + w_y{}^*\tau_y + w_r{}^*\tau_r + w_c{}^*\tau_c$
  end for
end procedure

The mapping function for assessing the risk of a specific traffic is expressed as:

$$f : \tau_t \times I_t \longrightarrow R_t \tag{13}$$

Table 2 shows the Risk assessment model of SDN with respect to the criticality of the traffic and threat level with respect to the specific traffic. For example, if criticality of a traffic is High(H) and its threat value is 5.5, then the risk associated with the traffic is High(H).

Table 2: Risk Assessment Model of SDN

Traffic Criticality	Total Threat Value		
	$\leqslant 0.39$	0.4 to 0.69	≥ 0.7
H	M	H	C
M	L	M	H
L	L	L	M

Note: C- Critical, H - High, M - Medium and L - Low

The calculated risk measures determined by the proposed *Risk_Analyze* function, are used in the flow and routing control functions to extract an optimal route with minimal

risk for the given traffic request. This process is executed recursively until an optimal route with minimal risk is found. Then, the flow rule with this route and other flow attributes are populated in the flow tables of corresponding forwarding switches.

In the next section, we demonstrate our proposed security enforcement function with an extensive case study of an SDN based enterprise network.

5. Experimental Results: Case Study

This section presents the performance of our proposed security enforcement functions with a case study. Figure 6 shows a segment of an SDN based large enterprise network with four subnets corresponding to different departments, i.e., Research and Development (R & D), Finance, Sale, and Customer Relationship (CR). The underlying SDN platform executes a set of control functions such as Wireshark, mainApp, OpenSSL, etc. inside the controller. The end hosts run applications FTP, RSH, nmap etc. In addition, the switches run protocols, e.g., MPLS, LDP, etc.

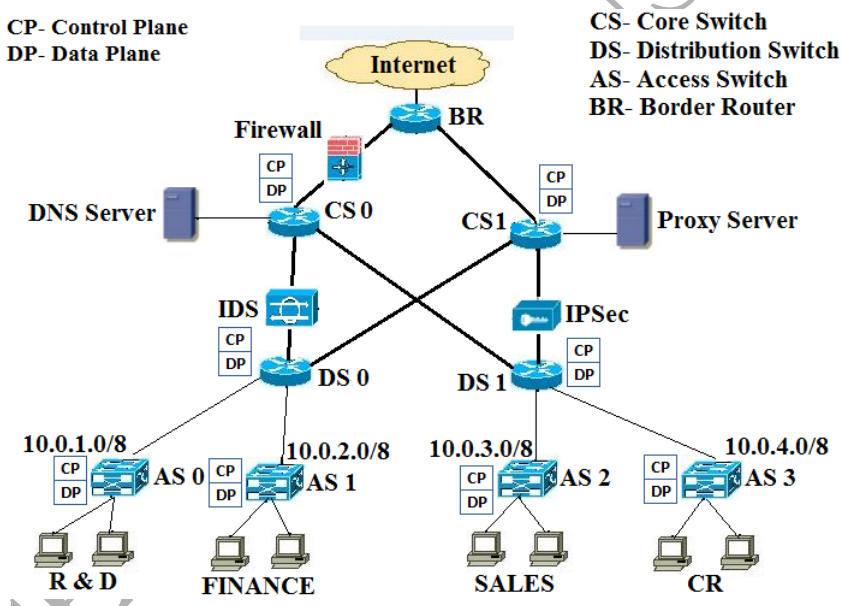

Figure 6: A segment of an Enterprise Network

The requirements on various network traffic are stated as follows:

Req_1: All outbound traffic from CR department should go through Proxy server (PS).

Req_2: All traffic from employees of R & D working outside should be served through Virtual Private Network (VPN) with high Bandwidth.

Req_3: Any incoming traffic to R & D and Finance should be forwarded through policy check and security compliance.

Req_4: Any outbound traffic from R & D and Finance should guarantee high bandwidth and availability.

Req_5: Customers should be able to communicate to CR through a secure channel and remote shell service (RSH). Let they connect through port number 8091.

Req_6: The employees of Finance and Sale should perform file transfer or transaction with proper authentication.

Req_7: All traffic related to financial transaction from Finance to employees of R & D should be authenticated.

In order to process the traffic requests with respect to these requirements, the SDN controller must ensure secure flow of traffic across the network segment. The next subsection presents the efficacy of our proposed risk assessment model with different test cases considering the above-mentioned requirements.

5.1. Efficacy of the Security Control Functions

We have implemented our proposed security enforcement control functions in mininet simulator [47] with OpenFlow controller [48], [49]. We have generated different traffic with varying traffic rates from 10 traffic per sec (TPS) to 100 traffic per sec (TPS). Our proposed functions *Trust_Verify*, *Policy_Conflict_Resolve*, *Policy_Consistency_Check*, and *Risk_Analyze* are applied on these traffic to generate secure flow and routing control rules.

In order to process the traffic requests as per the above-mentioned requirements, our proposed risk assessment model first extracts the CVS scores or values of various related applications, protocols, and services running in different layers of the network from local vulnerability database. Table 3 shows the CVS scores of some of the applications, protocols, and services running in different components of the SDN environment under test with their vulnerability details. For example, OpenSSL v1.0.2b is a network control function running in the controller and has CVS score 5.9 that leads to confidentiality violation. On the other hand, nmap service running on end host with Android 6.1 Operating system has CVS score 7.8 and has impact like DoS attack, Arbitrary controller code execution. Similarly, LDP is a protocol running on an OpenFlow switch has CVS value 7.5.

Based on the CVS Score of the related applications, protocols and services with respect to a given traffic request the *Risk_Analyze* function determines the threat model for different SDN entities. The overall threat value is calculated as the cumulative threat values of the SDN entities. Finally, the risk of the specific traffic request is determined with respect to the final threat value and the criticality of the traffic. These risk measures guide the flow and routing controller to decide the flow rules (routing path along with other flow attributes) to be populated in the flow tables of the switches.

Table 4 shows the results of *Risk_Analyze* function for few sample traffic requests taken as test cases. For example, the traffic request t_1 associated to Req_1 has medium risk with route r_1 and low risk with route r_2. Hence, the routing path decided by the flow and routing control function is r_2. Similarly, the traffic request t_2 associated to Req_2 has high risk with route r_1 and medium risk with route r_2. Therefore, the route r_2 is chosen for traffic t_2. The abbreviations used in the Table 4 are as follows.

- C- Critical, H - High, M - Medium, and L - Low.
- CR - Customer Relationship, and R & D - Research and Development,

Table 3: Common Vulnerability Score (CVS) of applications/services/protocols

Applications/ Services/ Protocols	version	Operating System	Bug details	CVE id (online NVD) [44]	Vulnerabilities	CVS (local Vulnerability database)
OpenSSL	1.0.2b	Cisco NX-OS 6.0	Internal bug - "error state" mechanism with direct call of SSL_read() or SSL_write() functions	CVE-2017-3737	Confidentiality violation	5.9
Wireshark	2.2.11	Cisco NX-OS 6.0	Internal bug - improper use of new line characters in File_read_line function	CVE-2017-17935	DoS attack	7.5
RSH	-NA-	Linux	compatibility with fileio.c in Vim 8.0.10	CVE-2017-17087	Obtaining root/admin privileges	5.5
SSL	-NA-	Linux	Lack of server's host name verification with subjectAltName field in X.509 certificate	CVE-2017-1000209	man-in-the-middle attack, spoofing	5.9
VPN	-NA-	Linux	Bug in the web-based management interface of Cisco Adaptive Security Appliance (ASA) Software	CVE-2017-12265	Arbitrary web script injection	6.1
nmap	-NA-	Android 6.1	elevated privilege vulnerability in System Server in Android	CVE-2016-6707	DoS attack, Arbitrary controller code execution	7.8
FTP	Light FTP v1.1	Windows	Buffer overflow in the "writelogentry" function	CVE-2017-1000218	DoS attack, Arbitrary controller code execution, and Obtaining root/admin privileges	9.8
MPLS	-NA-	Cisco NX-OS 6.0	Bug in Cisco Carrier Routing System (CRS) 5.1 for processing IPv6-over-MPLS packets	CVE-2016-6401	DoS attack, Arbitrary web script injection	5.3
LDP	-NA-	Cisco NX-OS 6.0	infinite loop due to a bug in print-lldp.c function of LLDP parser of tcpdump 4.9.1	CVE-2017-12997	DoS attack, Malicious code execution	7.5
MainApp	-NA-	Cisco NX-OS 6.0	hardcoded credentials for TELNET and SSH sessions	CVE-2016-1329	Obtaining root/admin privileges	9.8

- AS - Access Switch, DS - Distribution Switch, CS - Core Switch, BR - Border Router, FW - Firewall, IDS- Intrusion Detection System, PS - Proxy Server, and DNS - Domain Name Server.

Table 5 shows the statistics of the output of the proposed security enforcement functions

Table 4: Result of the *Risk_Analyze* function

Traffic	Criticality	Route	Risk	Selected Route
$t_1 \Leftrightarrow Req_1$	L	r_1: *CR-AS3-DS1-CS0-FW-BR-Internet*	M	r_2
		r_2: *CR-AS3-DS1-IPSec-CS1-PS-BR-Internet*	L	
$t_2 \Leftrightarrow Req_2$	M	r_1: *Internet-BR-CS1-PS-DS0-AS0-R&D*	H	r_2
		r_2: *Internet-BR-FW-CS0-DNS-IDS-DS0-AS0-R&D*	M	
$t_3 \Leftrightarrow Req_3$	H	r_1: *Internet-BR-CS1-DS0-Finance*	H	r_2
		r_2: *Internet-BR-FW-CS0-DNS-IDS-DS0-Finance*	M	
$t_4 \Leftrightarrow Req_4$	M	r_1: *R&D-AS0-DS0-IDS-CS0-DNS-FW-BR-Internet*	H	r_2
		r_2: *R&D-AS0-DS0-CS1-PS-BR-Internet*	M	
$t_5 \Leftrightarrow Req_5$	L	r_1: *Internet-BR-CS1-PS-IPSec-DS1-AS3-CR*	M	r_2
		r_2: *Internet-BR-FW-CS0-DNS-DS1-AS3-CR*	L	
$t_6 \Leftrightarrow Req_6$	H	r_1: *Finance-AS1-DS0-IDS-CS0-DS1-AS2-Sales*	H	r_2
		r_2: *Finance-AS1-DS0-CS1-IPSec-DS1-AS2-Sales*	M	
$t_7 \Leftrightarrow Req_7$	H	r_1: *Finance-AS1-DS0-IDS-AS0-R&D*	M	r_1
		r_2: *Finance-AS1-DS0-AS0-R&D*	C	

in SDN with respect to the total number of traffic requests. The abbreviations TREQ, CNF, BLK, CONS, RISK in the table indicate the number of traffic requests, number of conflicts detected and resolved, number of applications blocked, number of inconsistencies identified between existing flow rules and updated policy rules, and number of risks mitigated respectively. Our previous work [34] already discusses the number of conflicts detected, potential compromised applications blocked, and the number of inconsistencies resolved for a specific traffic request. In this paper, we determine the number of potential risks associated with a given traffic request. The number of risks in this context means the number of vulnerabilities determined to have either one of the risk values: Low, Medium, High or Critical associated with a specific traffic request. There can be more than one vulnerabilities related to a given traffic request in this case. While finding the overall vulnerability of end hosts, switches and controllers respectively in the equations 1, 4 and 8 for a specific traffic request, we maintain a data structure to keep track of the individual vulnerabilities associated to the applications with their risk levels. The same is reported with their impact with respect a specific traffic request and the risks of different levels are mitigated appropriately.

The efficacy of our proposed security functions is evaluated in terms of execution time with varying traffic rate. Figure 7 shows the execution time for different security enforcement functions. In our previous work [34], we already observed that, the execution time of *Trust_Verify* and *Policy_Consistency_Check* functions almost remains constant with increase

Table 5: Output of Proposed Security Enforcement Functions

TREQ	**CNF**	**BLK**	**CONS**	**RISK**
15	5	3	2	25
30	16	3	2	40
50	30	10	12	70
100	60	20	25	160

in traffic rate. On the other hand, the execution time of *Policy_Conflict_Resolve* function increases quadratically with the increase in traffic rates as reported in [34]. This is due to increase in the number of policy and flow control rules with the increase in traffic rate. In this paper, we evaluate the execution time of *Risk_Analyze* function and report that it increases almost linearly with the increase in traffic rate. This is due to the use of a fixed threat model with a specific number of parameters used (vulnerability, exposure, traffic criticality).

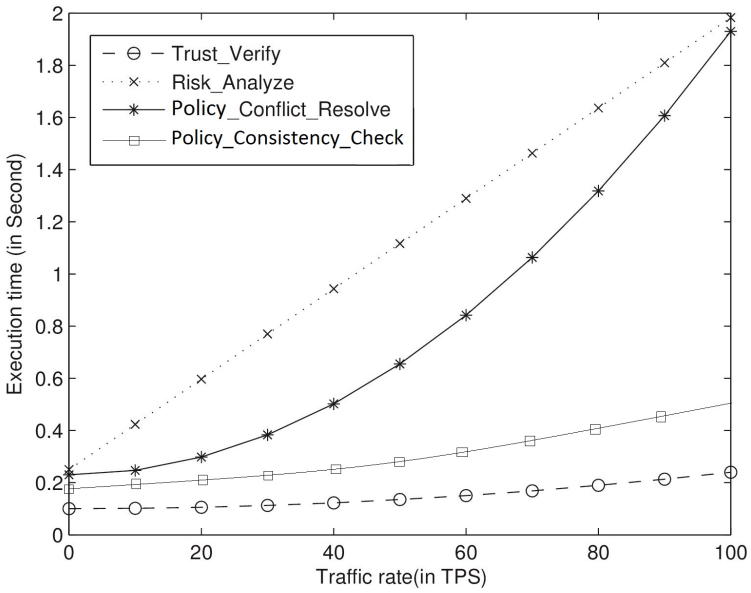

Figure 7: Execution time of the proposed Network Security Functions w.r.t. traffic rate

It is observed that the overall execution time of our proposed security enforcement functions lies within 2 seconds for traffic rate of 100 TPS which imply that these functions incur less overhead and is considerably reasonable for any kind of network. Our proposed functions incorporated in the SDN control plane ensures end-to-end secure transmission of packets across the network control execution environment and thereby strengthens the security perimeter over the underlying network.

In future, we plan to extend our proposed security enforcement functions for heterogeneous network environments with varying requirements and application context. In addition, sensitivity analysis will be incorporated in the proposed security solution as it helps in identifying the sources of the vulnerabilities in the system. Sensitivity [50] is mathematically defined as the ratio of the number of correctly predicted vulnerabilities those may be potentially exploitable to the number of actually exploitable vulnerabilities.

6. Conclusion

The Software Defined Network (SDN) paradigm might suffer from various security threats due to its inherent characteristics such as open user-control, ubiquitous execution of network functions and centralized control management. In this paper, a risk assessment model is proposed where the core component, i.e., the *Risk_Analyze* function analyzes the risk of a specific traffic request based on the threat models of related SDN entities. Then, these risk values are communicated to flow and routing control function for generating secure flow rules with no or minimal risk. The proposed security solution enables a pro-active end-to-end security assessment of the traffic requests and accordingly ensures a secure flow and routing process. The efficacy of our proposed functions has been evaluated through a case study of an enterprise network and the results have been reported. In future, the proposed security mechanism will be enhanced with appropriate sensitivity analysis to find out the potential sources of the vulnerabilities. In addition, the security functions will be verified with varying requirements and context.

References

[1] Nishtha, M. Sood, *Software defined network-Architectures*, in Proceedings of 2014 International Conference on Parallel, Distributed and Grid Computing (PDGC), pp. 451-456, December 2014.

[2] Ghodsi et al., *Intelligent design enables architectural evolution*, in Proceedings of 10th ACM Workshop Hot Topics in Networks, Article No. 3, pp. 1-6, 2011.

[3] A. Kumar, S. Jain, U. Naik et al., *BwE: flexible, hierarchical bandwidth allocation for WAN distributed computing*, in Proceedings of the ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15), pp. 1-14, London, UK, August 2015.

[4] D. Kreutz, F. Ramos, and P. Verissimo, *Towards secure and dependable software-defined networks*, in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, pp. 55-60, 2013.

[5] M. Jammal et al., *Software defined networking: State of the art and research challenges*, Computer Networks, Elsevier, vol. 72, pp. 74 - 98, 2014.

[6] D. Kreutz et al., *Software-Defined Networking: A Comprehensive Survey*, Proceedings of the IEEE, Vol. 103, No. 1, pp. 14-76, January 2015.

[7] Metzler, *Research: Understanding Software-Defined Networks*, Information Week Reports, pp. 125, October 2012. [Online] Available: http://reports.informationweek.com/abstract/6/9044/Data-Center/research-understanding-software-defined-networks.html.

[8] Z. Li, Y. Gao, and Y. Chen, *HiFIND: A high-speed flow-level intrusion detection approach with DoS resiliency*, Comput. Netw., vol. 54, no. 8, pp. 1282-1299, 2010.

[9] S. A. Mehdi, J. Khalid, and S. A. Khayam, *Revisiting traffic anomaly detection using software defined networking*, in Recent Advances in Intrusion Detection. Springer, pp. 161-180, 2011.

[10] J. R. Ballard, I. Rae, and A. Akella, *Extensible and scalable network monitoring using opensafe*, Proc.INM/WREN, 2010.

[11] S. Shin et al., *FRESCO: Modular composable security services for software-defined networks*, in Proceedings of Network and Distributed Security Symposium, 2013.

[12] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, *AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks*, in Proc. ACM SIGSAC Conf. Comput. Commun. Security, pp. 413-424, Berlin, Germany, 2013.

[13] H. Shawn, L. Scott, O. Tomasz, S. Adam, *Uncover Security Design Flaws Using The STRIDE Approach*, Mar. 2015. [Online]. Available: http://msdn.microsoft.com/en-gb/magazine/cc163519.aspx

[14] R. Kloti, V. Kotronis, and P. Smith, *OpenFlow: A security analysis*, in Proc. 21st IEEE Int. Conf. Netw. Protocols (ICNP), pp. 1-6, Gttingen, Germany, 2013.

[15] S. Shirali-Shahreza and Y. Ganjali, *Efficient Implementation of Security Applications in OpenFlow Controller with FleXam*, in 21st IEEE Annual Symposium on High-Performance Interconnects. IEEE, pp. 49-54, 2013.

[16] M. Canini, et al., *A NICE way to test OpenFlow applications*, in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, 2012.

[17] E. Al-Shaer and S. Al-Haj., *Flowchecker: configuration analysis and verification of federated OpenFlow infrastructures*, in Proceedings of the 3rd ACM workshop on Assurable and Usable Security Configuration, 2010.

[18] R. Sherwood et al., *Flowvisor: A network virtualization layer*, OpenFlow Switch Consortium, Tech.Rep, 2009.

[19] S. Son et al., *Model Checking Invariant Security Properties in OpenFlow*. [Online]. Available: http://faculty.cse.tamu.edu/guofei/paper/Flover-ICC13.pdf.

[20] P. Porras et al., *A security enforcement kernel for OpenFlow networks*, in Proceedings of the first workshop on Hot topics in software defined networks, ACM, pp. 121-126, 2012.

[21] Gabriel et al., *Achieving DDoS Resiliency in a Software Defined Network by Intelligent Risk Assessment Based on Neural Networks and Danger Theory*, 15th International Symposium on Computational Intelligence and Informatics, IEEE, pp. 319-324, 2014.

[22] D. V. Bernardo and B. B. Chua, *Introduction and Analysis of SDN and NFV Security Architecture (SN-SECA)*, 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, Gwangiu, pp. 796-801, 2015.

[23] L. Yao, P. Dong, T. Zheng, H. Zhang, X. Du and M. Guizani, *Network security analyzing and modeling based on Petri net and Attack tree for SDN*, 2016 International Conference on Computing, Networking and Communications (ICNC), pp. 1-5, Kauai, HI, 2016.

[24] Y. Tseng, M. Pattaranantakul, R. He, Z. Zhang and F. Nat-Abdesselam, *Controller DAC: Securing SDN controller with dynamic access control*, 2017 IEEE International Conference on Communications (ICC), pp. 1-6, Paris, 2017.

[25] S. Wang, K. G. Chavez and S. Kandeepan, *SECO: SDN sEcure COntroller algorithm for detecting and defending denial of service attacks*, 2017 5th International Conference on Information and Communication Technology (ICoIC7), pp. 1-6, Melaka, 2017.

[26] W. Etaiwi, M. Biltawi, and S. Almajali, *Securing Distributed SDN Controllers against DoS Attacks*, 2017 International Conference on New Trends in Computing Sciences (ICTCS), pp. 203-206, Amman, 2017.

[27] Jun-Huy Lam, Sang-Gon Lee, Hoon-Jae Lee, and Y. E. Oktian, *Securing distributed SDN with IBC*, 2015 Seventh International Conference on Ubiquitous and Future Networks, pp. 921-925, Sapporo, 2015.

[28] S. B. H. Natanzi and M. R. Majma, *Secure northbound interface for SDN applications with NTRU public key infrastructure*, 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), pp. 0452-0458, Tehran, 2017.

[29] J. Hoffstein, J. Pipher, J. H. Silverman, *NTRU: A Ring-Based Public Key Cryptosystem, Algorithmic Number Theory*, ANTS, Lecture Notes in Computer Science, Springer, pp. 267-288, 1998.

[30] S. M. Sohan, F. Maurer, C. Anslow and M. P. Robillard, *A study of the effectiveness of usage examples in REST API documentation*, 2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 53-61, Raleigh, NC, 2017.
[31] J. Hostein, J. Pipher, J. H. Silverman, *NSS: The NTRU Signature Scheme*, 2000.
[32] Y. Shi, F. Dai, and Z. Ye, *An enhanced security framework of software defined network based on attribute-based encryption*, 2017 4th International Conference on Systems and Informatics (ICSAI), pp. 965-969, Hangzhou, 2017.
[33] S. B. H. Natanzi and M. R. Majma, *Secure distributed controllers in SDN based on ECC public key infrastructure*, 2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA), pp. 1-5, Ras Al Khaimah, 2017.
[34] B. K. Tripathy et al., *A Novel Secure and Efficient Policy Management Framework for Software Defined Network*, IEEE 40th Annual Computer Software and Applications Conference, IEEE, pp. 423-430, 2016.
[35] A. S. Prasad, D. Koll and X. Fu, *On the Security of Software-Defined Networks*, Fourth European Workshop on Software Defined Networks, Bilbao, pp. 105 - 106, 2015.
[36] B. Anwer, T. Benson, N. Feamster, D. Levin, and J. Rexford, *A Slick Control Plane for Network Middleboxes*, Open Networking Summit, 2013. [Online]. Available: http://nextstep-esolutions.com/Clients/ONS2.0/ pdf/2013/research track/poster papers/nal/ons2013-nal51.pdf
[37] S. Fayazbakhsh, V. Sekar, M. Yu, and J. Mogul, *FlowTags: Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions*, in Proceedings of the second workshop on Hot topics in software defined networks. ACM, 2013.
[38] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, SIMPLE-fying Middlebox Policy Enforcement Using SDN, ACM SIGCOMM, August 2013.
[39] Mell P., Scarfone K. and Romanosky S., *Common Vulnerability Scoring System*, Security & Privacy, IEEE, Vol 4, Issue 6, pp. 85 - 89, December 2006.
[40] *Vulnerability (Computing)*, Wikipedia, [online].
Available: https://en.wikipedia.org/wiki/Vulnerability_(computing)
[41] *Exposure*, [online]. Available: http://www.businessdictionary.com/definition/exposure.html
[42] *Threat (Computer)*, Wikipedia, [online]. Available: https://en.wikipedia.org/wiki/Threat_(computer)
[43] *Cybersecurity Risk: A Thorough Definition*, Bitsight, [online].
Available: https://www.bitsighttech.com/blog/cybersecurity-risk-thorough-definition
[44] *National Vulnerability Database (NVD)*, [online]. Available: https://nvd.nist.gov/
[45] *A Complete Guide to the Common Vulnerability Scoring System Version 2.0*, [online]. Available: https://www.first.org/cvss/v2/guide
[46] *Common Vulnerability Scoring System v3.0: Specification Document*, [online]. Available: https://www.first.org/cvss/specification-document
[47] *Mininet Documentation*, GitHub. [Online].
Available: https://github.com/mininet/mininet/wiki/Documentation
[48] *OpenFlow White Paper*, [Online]. Available: http://archive.OpenFlow.org/
[49] *OpenFlow Specification*, [Online]. Available: http://archive.OpenFlow.org/
[50] A. A. Younis and Y. K. Malaiya, *Comparing and Evaluating CVSS Base Metrics and Microsoft Rating System*, 2015 IEEE International Conference on Software Quality, Reliability and Security, pp. 252-261, Vancouver, BC, 2015.

Bata Krishna Trpathy is a PhD student in School of Electrical Sciences, Indian Institute of Technology Bhubaneswar, India. He received his M Tech degree from Indian Institute of Technology Kharagpur, India in 2014. He has completed his B Tech from Institute of Technical Education and Research, Bhubaneswar, India in 2007. He has expertise in Mobile Ad hoc Networking, Software Defined Networking, and Network Security.

Debi Prasad Das has received his B Tech degree from National Institute of Technology Rourkela, India in 2017. He was working as an intern at Indian Institute of Technology Bhubaneswar, India in summer and winter vacations during B Tech. He is currently placed in Itron American Technology company, Bangalore, India as a Software Engineer. He has expertise in Software Defined Networking and Network Security.

Swagat Kumar Jena is a Project in School of Electrical Sciences, Indian Institute of Technology Scholar Bhubaneswar, India. He received his M Tech degree from Indian Institute of Technology Kharagpur, India in 2014. He has completed his B Tech from Seemanta Engineering College, Jharpokharia, India in 2008. He has expertise in Internet of Things, Wireless Sensor Network, and Network Security.

Padmalochan Bera received his PhD from Indian Institute of Technology Kharagpur, India in 2011. He completed his ME degree from West Bengal University of Technology, Kolkata, India and his BE from Jadavpur University, Kolkata, India in 2001. He is currently working as an Assistant Professor in School of Electrical Sciences, IIT Bhubaneswar, India. He has expertise in Network and Cyber Physical Systems Security, Software Defined Networking, Cloud Computing, Formal Verification and optimization.