TRUST ENHANCED SECURE ROUTING IN SDN

by

Nurefşan Sertbaş

B.S., Computer Engineering, Istanbul Technical University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2018

ACKNOWLEDGEMENTS

There are many people I would like to thank due to their support and encouragement during my master studies. First of all, I would like to express my deepest gratitudes to my co-supervisor Prof. Ufuk Çağlayan for his support. He gave me a chance to be a part of Department of Computer Engineering Department. I also would like to express my special appreciation and thanks to my supervisor Prof. Fatih Alagöz, for all his support, patience and guidance during the last year of my MSc. studies.

I would like to offer my heartfelt thanks to Orhan Ermiş for his guidance during my masters studies. I would like to thank him for his kindness and helping me out through some rough times. This thesis would not be possible without him. Many thanks also go to my thesis jury members: Prof. Tuna Tuğcu and Assist. Prof. Şerif Bahtiyar for their participation in my thesis committee in their rare time and their invaluable comments on my thesis. I also would like to thank SATLAB members for their help and invaluable advices during my MSc studies. In particular, I would like to thank Samet Aytaç for his great support and encouragement to finish my thesis.

Last, but not the least, I would like to express my gratitude to my parents, friends, and colleagues for their support from the beginning till the end of my thesis. My apologies go to those I may miss, a collective thanks to all.

This thesis work has been fully supported by the Scientific and Technical Research Council of Turkey (TÜBİTAK) under BİDEB 2210-A (National MSc Scholarship Programme).

ABSTRACT

TRUST ENHANCED SECURE ROUTING IN SDN

The presence of programmable entities, namely controllers and switches, in Software Defined Networks (SDNs) provides a hierarchical architecture to achieve dynamic and manageable networks for easily utilizing high bandwidth communications. However, such architecture may cause unsubstantiated packet dropping and incorrect packet forwarding due to the inability of current networking protocols in SDN. For instance, the detection of a compromised switch, which can be used for manipulating the data plane operation, is not possible with OpenFlow. One of the potential candidates to overcome such vulnerabilities is to use an approach to reflect subjective behaviors of entities for detecting the compromised ones. The provision of computational trust based solution for evaluating subjective behaviors is expected to help determine compromised switches. Therefore, in this thesis, we propose a Trust Enhanced Secure Routing (TESR) for switches to be used in secure routing. The proposed model provides three different trust computations in order to find the most suitable trust level for different states of a network. To show the applicability of the proposed approach, we demonstrate a set of simulations for the detection of compromised switches. Simulation results show that TESR operates effectively to detect and eliminate compromised nodes while selecting secure paths.

ÖZET

YAZILIM TABANLI AĞLARDA GÜVEN İLE İYİLEŞTİRİLMİŞ GÜVENLİ YÖNLENDİRME

Yazılım tabanlı ağlarda programlanabilir varlıkların yani denetleyici ve anahtarlayıcıların varlığı, yüksek bant genişlikli iletişimi kolayca kullanımı için dinamik ve yönetilebilir ağlar için hiyerarşik bir yapı sağlar. Bununla birlikte, yazılım tabanlı ağlardaki mevcut protokollerin yetersizlikleri nedeniyle bu tür yapılar doğrulanmamış paket düşmeleri ve yanlış paket yönlendirmeleri gibi ataklara neden olabilir. Örneğin, veri düzleminin çalışmasını manipüle etmek için bir anahtarlayıcının ele geçirildiğinin tespiti OpenFlow ile mümkün değildir. Bu tür güvenlik açıklarının üstesinden gelmek için kullanılabilecek potansiyel adaylardan birisi ele geçirilmiş olan anahtarlayıcıları tespit etmek için varlıkların öznel davranışlarını yansıtacak bir yaklaşım kullanmaktır. Kişisel davranışları değerlendirmek için hesaplamalı güven temelli çözümün sağlanması, tehlikeye giren anahtarlayıcıların belirlenmesine yardımcı olacaktır. Bu nedenle, bu tezde, güvenli yönlendirmede kullanılacak anahtarlayıcılar için bir Güven ile İyileştirilmiş Güvenli Yönlendirme'yi öneriyoruz. Önerilen model, bir ağın farklı durumları için en uygun güven düzeyini bulmak için üç farklı güven hesaplaması sunar. Önerilen yaklaşımın uygulanabilirliğini göstermek için, ele geçirilen anahtarlayıcıların tespiti için bir dizi simülasyon gösteriyoruz. Simülasyon sonuçları Güven ile İyileştirilmiş Güvenli Yönlendirme'nin güvenli yolları seçerken risk altındaki düğümleri tespit etmek ve ortadan kaldırmak için etkin bir şekilde çalıştığını göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A	An ACTION filed of flow entry
ADJ	(NxN) Adjacency matrix
BC_{v_i}	Betweenness centrality of *node i*
a	Destination switch
b	Source switch
c	The number of edges in ADJ
$E(p)$	The pdf of observing x as an outcome
f	The number of maliciously altered flow entry in each switch
f_i	Flow rule to be used for probing
f_{test}	Flow to be tested
h	The number of flow entries in each switch
$itcScore$	The set of initial indexes for all switches in S
k	Number of shortest paths
K	A set of neighbors of the S_{test}
l	The number of selected switch
m	Number of compromised switches
M	A Match filed of flow entry
M_{test}	A Match filed of probe packet
N	A number of switches
p	Probe packet
P	Attack detection probability
P_{best}	The best path of all paths from a to b for time $t_{current}$
$P_{\texttt{fl}}$	The probability of selecting maliciously altered flow entry
P_k	Result of the k th iteration probing
P_{paths}	Set of possible paths from a to b
r	The number of successfully forwarded packets to the next hop
$P_{\texttt{sw}}$	The probability of selecting compromised switch
r_k	The value of r at k^{th} iteration

r_{new}	Updated value of r after k^{th} iteration
r_{old}	Previous value of r before k^{th} iteration
S	The set of OpenFlow switches in data plane
s	The number of failed packets
s_k	The value of s at k^{th} iteration
s_{new}	Updated value of s after k^{th} iteration
s_{old}	Previous value of s before k^{th} iteration
S_i	i^{th} switch in S
SR_c	Application security requirements
S_{test}	Switch to be tested
T	The set of β trust indexes for all switches
T_c	Minimum required trust level
T_P	The set of trust indexes of the k paths
T_S	The set of trust indexes of the switches in S
T_{S_i}	Trustworthiness of *switch* i
$t_{current}$	Current time
t_p	Polling interval of switches
T_v	Varying feedback value based trust index
Tp_i	Probability-based trust index of *node* i
t_{sat}	Network saturation time
$t_{S_i,1}$	Initial index
$t_{S_i,2}$	Probing-based index
$t_{S_i,3}$	Statistics-based index
U	The set of users in the network
v	The feedback value
VID	VLAN_ID
α	The set of trust index weights
α_1	Initial index weight
α_2	Probing-based index weight
α_3	Statistics-based index weight

σ_{jk}	The number of shortest paths between all node pairs
$\sigma_{jk}(v_i)$	The number of shortest paths passing through *node* i to connect *node* j and k

LIST OF ACRONYMS/ABBREVIATIONS

AODV	Adhoc On-demand Distance Vector Routing
CLI	Command Line Interface
DoS	Denial of Service
FN	Forwarding Nodes
GUI	Graphical User Interface
HDB	History DataBase
IDS	Intrusion Detection System
LKSON	Local Knowledge Sharing Overlay Network
MAC	Media Access Control
MANET	Mobile Ad hoc Network
NFN	Non Forwarding Nodes
NI	Northbound Interface
ONF	Open Networking Foundation
OS	Operating System
P4	Programming Protocol-independent Packet Processors
QoS	Quality of Service
RADIUS	Remote Authentication Dial-In User Service
SDN	Software Defined Network
SD-VANET	Software-defined Vehicular Ad Hoc Network
SFC	Service Function Chaining
SI	Southbound Interface
TCAM	Ternary Content Addressable Memory
TESR	Trust Enhanced Secure Routing
TLS	Transport Line Security
ToCP	Trust Oriented Controller Proxy
VM	Virtual Machine
WSN	Wireless Sensor Networks

1. INTRODUCTION

Traditional networks are complex and very hard to manage. The control logic and the forwarding logic are bundled in each networking device that reduces flexibility and slows down the innovation. Moreover, it is needed to configure thousands of network devices and protocols individually in order to add new functionalities to the network. Software Defined Networks (SDN) has emerged to solve suhc drawbacks of traditional networks such as configuration challenges, the lack of agility and the scalability.

SDN provides programmable networks which refer to the ability of control, change, and manage network behavior dynamically by using software programs. An underlying idea is to decouple controlling and forwarding functions from each other. Such decoupling enables centralized control of forwarding elements, global visibility, programmability as well as a dynamic and flexible network management. Also, SDN enables more flexible resource allocation by adding or removing devices on the fly. The global knowledge of the centralized controller simplifies the development of networking functions and services. These functions and services are dynamically managed by the centralized entity called the controller. Moreover, SDN architecture enables more flexible routing approaches than traditional networks. For instance, a controller directs data plane entities to forward network traffic through the certain nodes depending on the need. Therefore, SDN promises simplified and utilized network management and improved performance. However, the standardization of SDN is still incomplete. Such incompleteness makes SDN open to some security vulnerabilities by introducing new attack surfaces due to separation of control and data plane.

There have been many studies regarding the security of SDN [4–6]. In general, attackers use vulnerabilities of switches to compromise nodes in the data plane. When some of switches are compromised in the network, constructing correct forwarding paths becomes a challenging task. Attackers target data plane by using these switches to manipulate data plane operations. Accordingly, compromised switches follow com-

mands given by the attacker and cause security attacks such as incorrect forwarding attack or dropping packets attack. One of the In incorrect packet forwarding attack, commands given by the controller may not be executed correctly [4]. For instance, an attacker instructs a switch to duplicate packet rather than applying forwarding rules commanded by the controller [4–6]. It is possible to redirect sensitive data to untrusted parties or attackers. In addition, some packets can be duplicated and redirected by using more than one ports of a switch. Although the original packet may arrive at the specified destination as instructed, the duplicated packet can be redirected to unauthorized and untrusted third parties. Detection of incorrect packet forwarding attacks is not possible through OpenFlow stats because a compromised node may give deceptive information that seems node behaves correctly. The compromised node may flood the packet, manipulate the packet content or send the packet to the wrong destination port. In packet dropping attack, the compromised switch may drop packets passing through it. Such attacks cause performance degradation and suspension of communication [5,6] due to the re-transmission cost of the communication.

Existing mechanisms for compromised switch detection in the literature either consider only simple threads as in [5] or more complex as in [7–10]. The main security specification for the security of SDN communications between control and data plane is to use SSL/TLS protocol. However, this approach is only used for providing the confidentiality and authentication for communications not for providing the availability of a network itself. In addition, SDN standards such as OpenFlow also suffer from detecting compromised switches in the network. Accordingly, attackers may retrieve necessary information by using such compromised switches. Moreover, such switches may pursue leaking information without being detected as defined in incorrect packet forwarding and dropping packet attacks. Therefore, there is a need for an additional mechanism to analyze and identify switch behaviors by observations, which is our main motivation for this study. Using computational trust is one of the most appropriate candidates to subjectively analyze switch behavior in the network by considering the communication of a switch with other switches in its neighborhood. The study in [11] uses packet forwarding statistics to detect security attacks. However, to best

of our knowledge, using only such statistics are not only enough to detect attacks. Additional information such as initial security precautions of a switch and detailed probing analysis have to be considered to correctly detect the compromised switches, which is our another motivation for the study.

One of the potential candidates to overcome security vulnerabilities in SDN is to use an approach to reflect behaviors of entities in the network. For this reason, Using computational trust is one of the most appropriate candidates to subjectively analyze switch behavior in the network by considering the communication of a switch with other switches in its neighborhood. In general, computational trust is defined as a trustor trusts a trustee to perform a specific action or provide a specific service. In this study, we define trust as a measure of how confident switch about to deliver a packet to its instructed next hop. The study in [11] uses packet forwarding statistics to detect security attacks. However, to the best of our knowledge, using only such statistics are not only enough for correctly detecting compromised switches. Additional information such as the initial security precautions of a switch and detailed probing analysis have to be considered for the detection, which is our another motivation for the thesis.

Our main contributions are as follows:

- We propose Trust Enhanced Secure Routing (TESR) for SDN, which provides a computational trust model to detect compromised switches in the data plane. The proposed three dimensional trust model evaluates switches based on the initial configuration, forwarding behavior and packet forwarding statistics.
- TESR has an initial trust index assignment algorithm to evaluate the trustworthiness of switches at the initial state of the network. We check an initial security precautions of every switch by considering the use of access control mechanisms, current firmware version, etc. Thus, we provide initial trust index for the nodes.
- In addition, TESR provides two different detection algorithm for incorrect packet forwarding and packet dropping attacks. In the first one, the algorithm polls switches to evaluate their forwarding behavior by sending probe packets to the

network. In the latter one, collected switch statistics such as port statistics are used for evaluating the node.

- TESR provides a path construction algorithm. This algorithm uses trust indexes of nodes to determine the correct routing path, namely initial trust index, probing-based index and statistics-based index. By using this indexing approach, it is also possible to determine adaptive routing for different applications with different trust expectations.
- We provide simulations for TESR in order to evaluate the applicability. Moreover, we show the detection performance of TESR comparing with the models used in literature.

The organization of the thesis is as follows:

In Chapter 2, we present a comprehensive overview on SDN, computational trust.

In Chapter 3, we introduce TESR. In addition, we provide a discussion for the performance of TESR with respect to the computational complexity of the proposed algorithms and their detection rates.

In Chapter 4, we present numerical evaluations for the performance and the efficiency of TESR by using simulations on an example scenario. Then, by considering the simulation results, we give a discussion on the practicality and applicability.

Finally, in Chapter 5, our conclusions and future research directions are drawn.

2. LITERATURE REVIEW

In this chapter, we review the literature with respect to software-defined networking and trust approach. First, we explain the basic concepts related to SDNs together with security issues in SDN. Then, we give basic definitions and a brief review of the trust mechanisms in different environments.

2.1. An Overview of Software Defined Networks

Nowadays, the Internet becomes an essential tool for our daily life. However, it was not designed to meet the requirements of current traffic. In other words, handling with increasing number of users and huge growth in data traffic become a significant problem for traditional networks. Therefore, there exist several researches about the future Internet architectures to provide better performance, security, and scalability. Software-defined networking is proposed as a candidate for the next generation of the Internet.

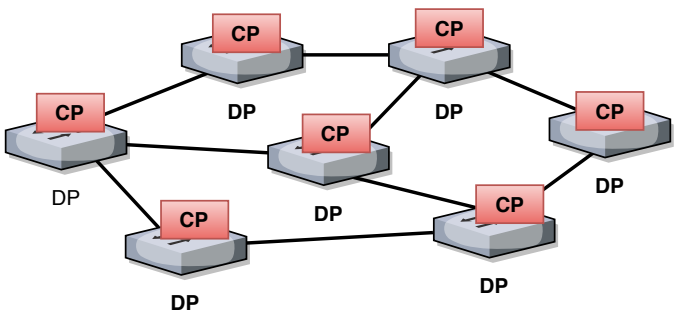

Figure 2.1. Traditional Networks

SDN is a solution to problems that could not be handled by traditional networks. SDN differs from traditional networks by the control logic. The control and data planes are embedded in the same networking devices in traditional networks while they are separated in SDN architecture as shown in Figure 2.1 and Figure 2.2 respectively.

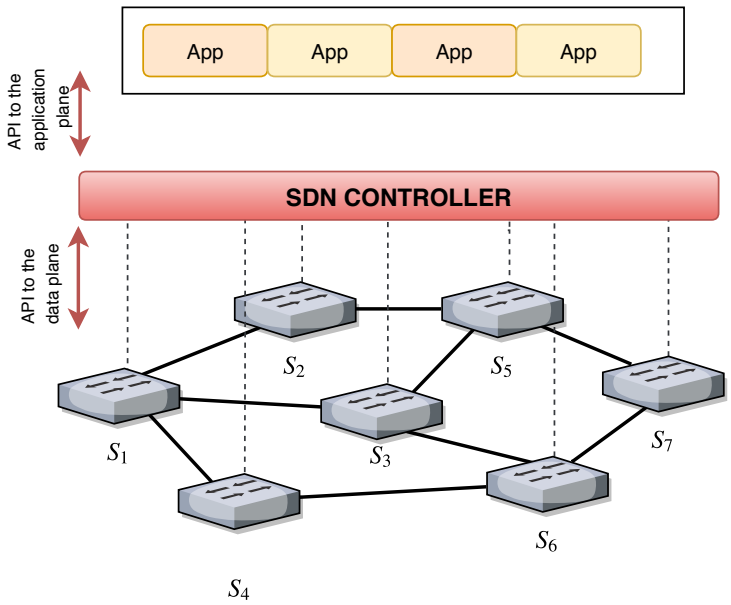

Figure 2.2. Software Defined Networks

Ethernet switch in traditional networks is build up multiple ports and a control logic. Control logic operates based on the forwarding table which contains a list of MAC addresses and associated ports. When a packet comes to the switch, the destination address is searched in the forwarding table. If the address is found, the packet is forwarded from the associated port. Otherwise, the packet is flooded to all ports. This means that control logic is completely based on the predefined static configuration.

Such decentralized and static structure in traditional networks was considered as an optimum way to make the network resilient [12]. However, traditional networks are not able to meet the current market requirements. Limitations of traditional networks are listed as follows:

- Configuration challenges: In traditional networks, the configuration of network devices is done manually by using Command Line Interface (CLI). Such a process is very slow and time-consuming even with the use of network automation tools. To bring new functionality to the network, it is needed to configure individual

devices and wait for vendor releases. Therefore, the network configuration should be automated and independent from the vendor.

- Multi-vendor devices: Organization of the multi-vendor environment is a difficult task and requires experience. An administrator should have a knowledge of all device types to manage the network properly.
- Slow deployment: Introducing new services takes a long time in traditional networks. The changes needed to be tested and re-tested again. Also, it is important to implement new services without interrupting the system connectivity.
- Virtualization challenges: Currently, applications are distributed across multiple virtual machines (VMs), which provides several services without knowing the physical location of the hosts. Therefore, traditional networks should be adapted to support such virtualized architectures.
- Inability of agility and flexibility: Managing and controlling devices according to the needs of the businesses is a significant task. The system should be flexible to respond to these requirements on time by monitoring load and changes in the network. Also, such architectures should be capable of reacting faults in real time.
- Lack of optimization: In order to respond requirements of the network, applications should be able to directly program the network so that the network parameters are optimized for any network condition.

As mentioned above, traditional approaches are not able to meet the current requirements of the market. Therefore, it is needed to find new approaches to handle those requests. SDN is a promising solution which is proposed to overcome the drawbacks of traditional networks. Based on the white paper from Open Networking Foundation (ONF) [3], the main benefits of SDN are as follows:

(i) Decoupled control and data planes: Control logic moves from forwarding entities to the entity called controller. Therefore, control functions operate at this layer.

(ii) Logically centralized architecture: Controller plane is considered as a network intelligence with global network view. The controller aware of the topology and

configures switches to meet the dynamic requirements of the network.

(iii) Abstraction of underlying network infrastructure: Such an abstraction enables controllers to automatically react and adopt network to changing business needs.

SDN is a new networking approach that handles the complex nature of the traditional networks. Data plane is very similar to the traditional networks by including a set of networking equipment which is used as forwarding elements. The difference from the traditional networks is that the embedded control is shifted to the logically centralized control plane.

The forwarding plane entities inform the controller about topology changes and traffic statistics. With the help of updated network states, switches can be reconfigured by the controller to meet the dynamic requirements of the network. Moreover, such architecture supports the diversity of the network elements by allowing the network administrator to use equipment from different vendors. Global network view and the software-based structure of the control plane makes easy to manage the network.

An architecture of SDN composed of three layers as follows:

- Data/Forwarding Plane: Data plane consists of several forwarding devices which can carry out some specific operations that are defined in the instruction set. Forwarding entities take an action on the incoming packet based on their forwarding table. These actions may be drop packet, forward packet to some specific ports, forward to the controller or rewrite some part of the header. The connection between these devices may be wireless through the radio channel or wired.
  Communication between a controller and data plane carry out through the Southbound Interface (SI). First and probably most well-known southbound interface is OpenFlow, which was developed by ONF. Forwarding entities will inform controller through the SI about the current state such as statistics, resources. Then, the controller can make changes dynamically according to real-time demands and needs by installing the necessary instructions in forwarding devices.

- Control Plane: A controller is seen as a network intelligence and can program the forwarding devices depending on its own logic. While the controller can communicate with a data plane through the southbound interface, it also communicates with the management plane through Northbound Interface (NI). It helps to abstract low-level instruction sets as in the southbound interface. An SDN controller operates as a broker between the applications and the forwarding elements. Some specific functions can be requested through the application layer and the controller can carry out that functions by instructing forwarding elements if it is necessary.
- Application/Management Plane: Application plane includes several applications with different functions such as network virtualization, intrusion detection and prevention, network monitoring, and flow balancing. Each application may request different functions from the network. Such requirements are transformed to the low-level instructions by the controller to program the behavior of the forwarding devices [12].

2.1.1. OpenFlow Protocol

OpenFlow is a well-known communication protocol to manage an OpenFlow switch from a remote controller in SDN [13]. It is responsible for the communication of forwarding elements with the controller by specifying the form of exchanged messages.

A typical OpenFlow switch has at least one flow table and a group table as shown in Figure 2.3. An OpenFlow switch can communicate with a controller through the OpenFlow channel. A controller uses OpenFlow channel to program the behavior of data plane by adding/deleting or modifying the entries in the flow tables.

An OpenFlow switch process packets based on OpenFlow pipeline to take an appropriate action. The OpenFlow pipeline contains a set of linked flow tables as shown in Figure 2.4. Each flow table has several flow entries that contain match and

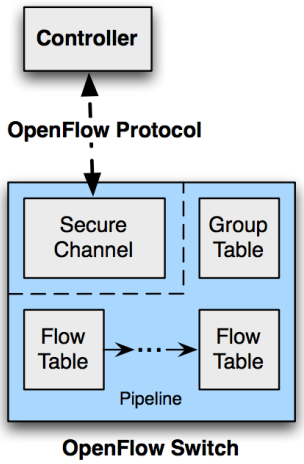

Figure 2.3. Main Components of an OpenFlow Switch [1]

action fields.

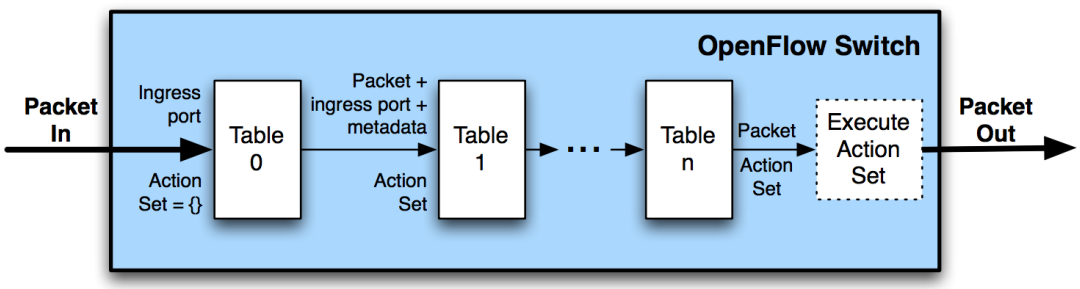

Figure 2.4. Overview of Operations in OpenFlow Switch [1]

The pipeline is triggered with an incoming packet to the switch. The process starts from Table 0 by searching lookup key which is generated from header fields in the table. In case of the key exists in the table, a corresponding action is performed by the switch. Such an action may be a modification of packet or update the action set. Then, the packet is directed to another table. This procedure repeats with the remaining tables depending on the specified action in the previous table. In case of matching flow entry does not direct packets to another flow table, pipeline processing stops at that table. The packet is processed with its associated action set and the process is over. The flowchart of the packet flow through an OpenFlow switch is shown

in Figure 2.5.

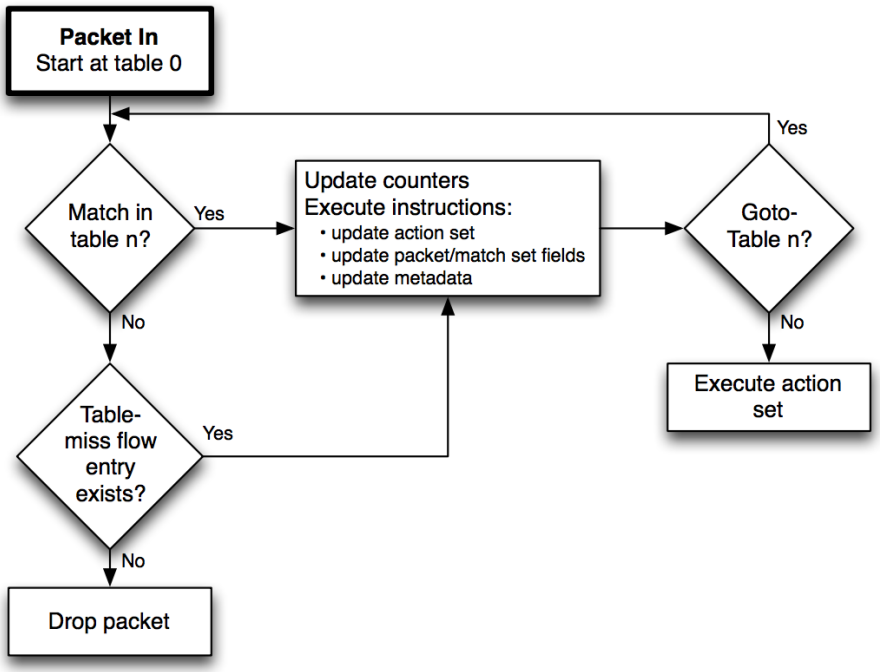

Figure 2.5. Flowchart Detailing Packet Through an OpenFlow Switch [1]

If a packet does not match with any entry in a flow table, the switch may drop the packet, pass the packet to another table or asks controller what to do with the packet. The controller runs its own logic and responds to the switch as "Send packet out of port x". Then, the switch will perform that request. The behavior of the switch in case of miss-match depends on the configuration.

2.1.2. Applications of SDN

SDN can be deployed on several different environments such as home and enterprise networks, data centers. SDN architecture provides an opportunity to add some additional functionalities to applications such as adaptive energy preservation and security mechanisms. Dynamicity and programmability enable adjusting network according to the environmental changes. By considering all features that come with SDN, we can say that SDN has a wide array of applications. SDN applications are

grouped into five categories as following [12]:

- Traffic Engineering: There exist several traffic engineering studies for minimizing power consumption, increasing the network utilization and balancing the server load. Such applications are useful for large-scale organizations for avoiding bottlenecks and optimizing the performance.
- Mobility and Wireless: Management of limited wireless spectrum and radio resources is critical issue in communication networks. SDN-based approaches provide more optimal solutions with dynamic and flexible operations and fine-grained logically centralized decisions.
- Measurement and Monitoring: Measurements from the network can be useful for deploying new services as well as improve the existing ones. For instance, collection of data plane statistics in SDN helps controller to estimate the loads in the data plane. There exist different sampling and estimation studies to reduce the load of the statistics collection process and increase the estimation performance.
- Data Center Networking: SDN provides solutions for problems of large-scale organizations such as QoS-based applications, management difficulties of large-scale networks, resource constraints. Dynamic and flexible nature of SDN helps data center networks in several cases such as improving the efficiency of the system, live network mitigation, intelligent resource utilization and fast reactions to failures.
- Security and Dependability: There exist two main issues for SDN and security. First issue is the use of SDN to improve the security of current networks. It is easier to detect the problems in the network and mitigate the attack of an attack by using programmable SDN approach. Second issue is improving the security of SDN itself. As stated before SDN is a new paradigm and brings several facilities but it is not a silver bullet. It has several vulnerabilities and there is a long way to go in deploying secure SDN implementations.

2.1.3. Security in SDN

Security has become very significant for any network with the evaluation of the Internet. Ensuring the protection and the prevention of the data and system from unauthorized access and/or manipulation become essential for each network. Even though SDN brings several powerful managing and monitoring facilities, there are some challenges should be considered. One of the challenges in the current SDN implementations is the standardization procedure that has not been completed yet. Standards provide compatibility and improve the interoperability by enabling deployment of equipment from different vendors. Coordination and cooperation between different entities become a problem for networks that have diversity in the equipment such as SDNs. Lack of standardization may cause security vulnerabilities in the network.

2.1.3.1. Threat Vectors in SDN. Centralization of control logic in SDN leads to new security threats and an increases attack surfaces [14]. Such attacks may target the planes or the interfaces of the SDN. The threat vectors are shown in Figure 2.6 including both control and data plane are as following:

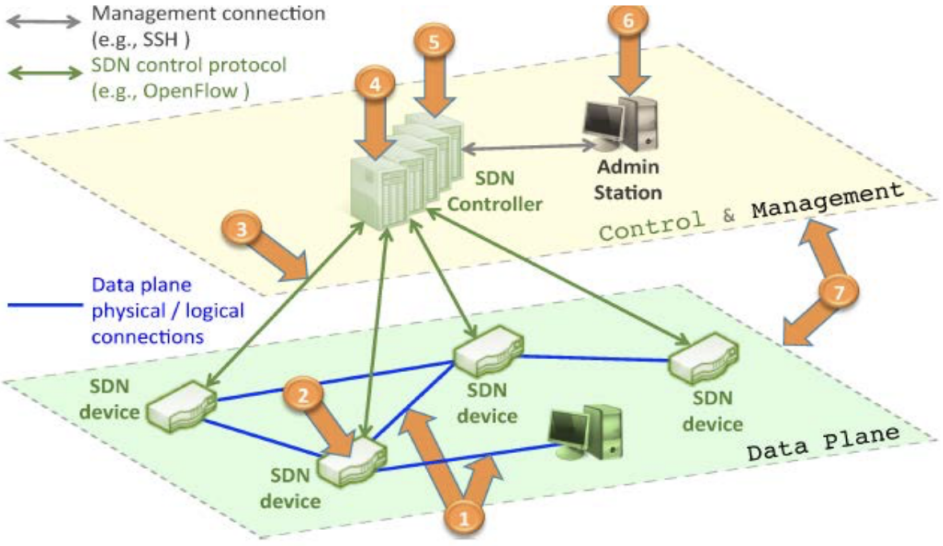

Figure 2.6. Threat Vectors of SDN [2]

- Threat Vector 1: Generation of fake traffic flows

  Such threats are not SDN specific, they are also a problem in traditional network approach. Generating fake traffic flows may causes a Denial of Service attacks (DoS) by filling the TCAM capacity of the switch.
- Threat Vector 2: Attacks by using switch vulnerabilities

  Attacks that derived from data plane vulnerabilities may affect on the whole network even though the entities are not in the control plane. There should be an attack detection mechanism to overcome such problems derived from attacked data plane entities.
- Threat Vector 3: Control plane communication

  Attacks targeting the control plane communication aims to exploit the packet transmission between the control plane and data plane. There exist some cryptography-based approaches to avoid such attacks.
- Threat Vector 4: Attacks by using controller vulnerabilities

  In SDN, compromising the controller means that compromising the network therefore, such attacks may critical for the SDN environment. Replication, diversity and recovery mechanisms will be used for avoiding controller vulnerabilities.
- Threat Vector 5: Trust between controller and management applications

  There may exist malicious applications that deployed on the controllers. In order to ensure the trust between controller and management applications, there should be additional mechanisms to measure how much the application is trusted.
- Threat Vector 6: Attacks by using admin station vulnerabilities

  Admin stations may also be malicious due to human factor. Such vulnerability can be used by an adversary to attack the controller and compromise the network. Therefore, authentication and authorization mechanisms should be deployed to handle such attacks.
- Threat Vector 7: Lack of trusted recovery mechanisms

  In case of any fault, there should be fast and trusted recovery mechanisms to report and diagnose the problem. Collecting reliable information from the system is significant for correct detection and fast recovery.

2.1.3.2. Literature Research for SDN security. There exist several studies in the literature about the SDN security. Such studies can be examined into two category as control plane and data plane. While control plane security focuses on the interactions with both upper and the lower planes, data plane security focuses on the reliability of the forwarding entities and their interactions with the controller.

Control plane is the brain of the network therefore, it is very significant to prevent controller from the attacks. As addressed in Threat Vectors 4-7, such attacks can be derived from the controller and admin station vulnerabilities, lack of trusted recovery mechanisms and communication between controller and application layer. In [15], policy checking framework is proposed to check the validity of the policies instructed by the applications. The proposed model operates as an intrusion detection system and aims to detect malicious policies. Thus, it is assured that all policies in the system are verified properly. To improve the network performance multiple controller scenario may be deployed. However, such an architecture provides opportunity for attackers to use controller vulnerabilities. The framework proposed in [16] deploys network hypervisor-based approach by adding an intermediary layer between control layer and network equipment named Trust Oriented Controller Proxy (ToCP). Such a proxy is used for compare the instructions from different controllers and determine the consistency. Then, ToCP installs rule on forwarding devices if rules are validated as consistent and trusted enough.

There exist several works in the literature that expose security vulnerabilities of SDN referring to Threat Vectors 1-3. Addressing Threat Vector 1, there exist attacks targeting the availability of the system. The reactive flow installation could be used by an attacker to create massive table-miss traffic. For instance, an attacker may send huge number of unique packets to its target switch. Then, the switch installs a great number of flow rules and the flow tables of the switch overloads with the installation of useless flow rules. The switch may ignore the new flow entries or remove the last one in the tables depending on the policy [17]. That type of attacks causes an decrease in the throughput of the packet forwarding and packet processing [18].

There are also some studies focuses on the security of the communication between the controller and the data plane elements as given in Threat Vector 3. The current implementation of OpenFlow does not work with Transport Layer Security (TLS). Due to the need of multiple certifications in TLS, vendors do not prefer to use it. Certification-based solutions authenticates the owners' identity but they do not guarantee the trustworthiness of the key owner. Therefore there should be additional mechanism to deal with that problem [19]. In [20], cryptography-based model is proposed to avoid impersonation attack that distorts the controllers' global network view by impersonating network elements. The model verifies the integrity of the switches and uses short time flow specific keys to establish a protected channel for the communication.

Even if the communication channel between the controller and the switch is secured, there exist still some issues derived from switch vulnerabilities [21,22]. Referring to Threat Vector 2, there exist different attacks aims to insert malicious entries into the switches. In [11] an identity-based rule source authentication is proposed by dynamic trust evaluation mechanism with the help of network security vitalization. In [23], they aimed to avoid attacks derived from the installation of low priority rules in a way that bypasses the high priority administrator rule. The study proposes a new switch entry mechanism which describes hosts with their positions in the topology aiming to overcome that type of attacks. In [24], a mechanism including symmetric-key cryptography is used for ensuring the integrity of the forwarding rules at switches. The studies [15, 16, 25] for ensuring the integrity, trustworthiness, and consistency of the inserted rules in the switches.

An attacker can use vulnerabilities of data plane to compromise the switch. Such nodes follow commands given by the attacker and cause security attacks such as incorrect forwarding attack or dropping packets attack. In incorrect packet forwarding attack, commands given by the controller may not be executed correctly [4]. Then, it is possible to redirect sensitive data to untrusted parties or attackers. In addition, some packets can be duplicated and redirected by using more than one ports of a switch.

In such case, the original packet may arrive to the specified destination as instructed however the duplicated packet can be redirected to unauthorized and untrusted third parties.

There exist several studies that use probing approaches to detect incorrect forwarding by tracing the forwarding path. Service Function Chain (SFC) path tracer [26], uses a probing mechanism that checks the path taken by the SFC to identify problems in SFC configuration. However, the mechanism requires that packets sent for probing should not be dropped or IP header should not be modified. In literature, different probing mechanisms are proposed to overcome that problem [27, 28]. SDN traceroute [29] is another tool that shows the traversed path by the packet using probing mechanism. The proposed method sends a probe packet to the controller after each hop which brings significant latency overhead to the network. Netography [30] uses a troubleshooting approach that injects probe packets into network and then, collects and analyzes copies to find out root causes of the problem. In [4], probing is used for verifying the forwarding behavior of switches by testing entries in the flow table. However, node collaboration to manipulate the decision of a controller has not been taken into account.

In addition to the probing approaches, there exist also cryptography-based approaches. For instance, we can avoid tampering of routing information through these methods with additional cryptographic operations. However, implementation of such mechanisms highlights the trade-off between security and performance for the most of cases. ShortMAC [10], requires every node on the forwarding path adds an authenticator to each packet. In [24], a signature is embedded in packets and packet is checked by the nodes on the forwarding path to see whether it has been correctly forwarded. In [31], the path validation mechanism requires that each forwarding entity verifies cryptographic markings in the packet. There are other studies [7,8] focus on the detection and mitigation of attacks targeting the data plane entities. However, scalability of such approaches is limited due to cryptographic operations per packet and increased packet size.

Packet dropping attacks can be divided as black hole attack, selective forwarding attack and ONOFF attack as given in [32]. In the black hole attack, a node drops all packets received from its neighbor. Selective forwarding attack occurs when a node forwards the received packet to next hop with a certain probability. In the ONOFF attack, a node initially forwards all packets until it gets high trust value then, periodically it forwards data (attacker is OFF) and drops packets (attacker is ON).

It is challenging to distinguish whether packet drops arise from an attack or a congestion in the network [33]. There exist approaches that collect switch statistics via OpenFlow for detection of packet dropping attacks. FlowMon [5], aims to detect malicious activities such as packet dropping or packet swapping using statistics reports. In [6], both active probing and statistics checking is used for fault localization and packet obfuscation for fault mitigation in the data plane.

As a consequence, SDN is prone to several attacks due to lack of standardization and inability of current networking protocols. Therefore, it is a huge research field for academy and industry partners.

2.2. An Overview of Trust

Trust has always been a part of the human's life by social interactions. It is originated from the interactions between people in the society. In current relationships, trust mainly depends on the face to face meetings and recommendations, especially in business context. However, it is needed to make specific definition and model the trust to support same services without including human interactions. There should be an abstract way of specifying and managing trust for any kind of platform or application field. Formal trust specification is required several fields such as medical systems, e-commerce, mobile computing platforms, telecommuting and content selection for web documents.

2.2.1. Defining Trust

Trust is a multi disciplinary term that has been studied by several number of disciplines from different perspectives such as marketing, management, psychology and philosophy [34]. It is important to define trust but it is hard to make a certain definition about it because it is an abstract and complex quantity that has diverge meanings and properties depending on the context. In other words, there is no consensus definition of the trust in the literature. In [35], trust is defined as believing someone is good and honest and will not harm you, or that something safe and reliable. Another definition in [36]defines trust as an assured reliance on the character, ability, strength, or truth of someone or something. In [37], trust refers to the subjective probability of an entity that evaluates a future action of another entity. In [38, 39], trust has been defined as a subjective quantity between two agents that one's actions affect the other agent's decisions.

As a consequence, trust is a multi-dimensional term and the definition is highly dependent on the concept. Common to those definitions are the notions of confidence, belief, faith, hope, expectation, dependence, and reliance on the goodness, strength, reliability, ability, or character of an entity.

2.2.2. Trust Models in Computer Science

Definitions are as diverse as sociology, psychology, economics, philosophy, politics, and now computer science. With the development of the Internet-based applications, information systems become distributed instead of central. Distributed nature of the systems requires subjective quantity to be used for decision-making process for any circumstances. Despite of that trust is significantly important for modern systems, there is no accepted technique or tool for specification.

Due to the lack of consensus definition, authorization and authentication are used interchangeably to define the trust in computer science. Authorization can be defined

as an assignment of access control rights for a subject to perform specific actions under predefined constraints while authentication is defined as a verification of an identity of an entity by several features. Starting from this point of view, Grandison and Sloman define trust for Internet applications [19] as:

"The firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context."

In [40], trust includes identification and reliability of business partners, the confidentiality of sensitive information, the integrity of valuable information, the prevention of unauthorized copying and use of information, the guaranteed quality of digital goods, the availability of critical information, the management of risks to critical information, and the dependability of computer services and systems.

There should be support for different trust relationships and different type of security policies. Moreover, entities in trust relationship are distributed and there is no direct knowledge of each other. As a result, there is a need for a mechanism that establish trust relationship between entities. Also, trust decisions are hard-coded in many case which makes the trust mechanism in-scalable and in-flexible.

There exist several implementations of trust such as by reputation, subjective logic, fuzzy logic based models and game theory based models. Bayesian theory is frequently used for trust management and applied broadly. The mechanism based on the statistical analysis of the data recorded from the object's behaviors. While there exist different versions of different distributions such as Dirichlet distribution and max entropy, Beta Distribution is the most common distribution function. It is used for representing and deriving the reputation score which is calculated as an expectation value of beta probability density function.

In subjective logic based trust models, subjective beliefs about the truth are expressed by subjective opinions with the degree of uncertainty. In [41], subjective

logic is be used for calculative analysis of trust networks. Let x be the proposition and A is the belief owner, the subjective opinion can be formulated as follows:

$$W_x^A = (b, d, u, a) \tag{2.1}$$

where b is a belief that the specified proposition is true, d is the belief that the specified proposition is false, u is the amount of uncommitted belief and a is the a priori probability in the absence of evidence.

Fuzzy logic based trust models are widely used since such models can handle uncertainty and more suitable for imprecise data inputs like linguistics terms rather than numerical values [42, 43]. Such models use a series of IF-ELSE rules instead of modeling the system mathematically.

Game theory based models are proposed for capturing the behavior of the node mathematically but it is not an exact predictive tool, it gives suggestions about how participants ought to behave in the future. In [44, 45] game theory based model is used for selecting useful nodes and eliminate the uncooperative ones. The critical prerequisite is that this model is used for bidirectional behavior which is not suitable for one-way transmission.

2.2.3. Trust Mechanisms for Routing

Traditional security mechanisms such as cryptography and authentication provide data confidentiality and identification of the owner. However, such mechanisms cannot detect compromised nodes in the system. Trust can be seen as an additional mechanism to enhance the security of a system. Nodes are evaluated by trust model and trusted nodes are involved in the routing process. There are several trust mechanisms deployed in different networks such as Mobile Ad-Hoc Networks (MANETs) and wireless sensor networks (WSNs).

2.2.3.1. Trust based routing in Mobile Ad-Hoc Networks. Mobile Ad-Hoc Networks are highly challenging environment with decentralized architecture, dynamically changing topology, no infrastructure, and high mobility. Combination of such factors makes MANETs vulnerable to various malicious attacks. Therefore, trust is an efficient mechanism to enhance the security of the MANETs.

In literature, there exist several trust-based mechanisms to evaluate the behavior of the nodes by observing the network. Most of the studies modify the current routing protocol, named Adhoc On-demand Distance Vector Routing (AODV) protocol to enhance the security. In [46, 47], the proposed trust model uses control packets received/forwarded, data packets received/forwarded, route/streams established, and data forwarded/received as events collected from the network to be used for trust calculation. After trust calculation of nodes, route trust calculation process starts with the modified version of the AODV by adding a simple ACK mechanism and a new field to the routing table.

There are several works which use the modified version of the update mechanism of the Beta trust model for trusted routing operation. In Beta trust model, trust index of *node i* is computed as follows:

$$T_{ij} = \frac{r_{ij} + 1}{r_{ij} + s_{ij} + 2} \tag{2.2}$$

where the number of successfully forwarded packets from *node i* to *node j* is represented with s_{ij} and the number of failed packets is f_{ij}. Then after a certain time parameters will be updated with the use of an aging parameter β as following:

$$s_{ij}^{new} = \beta s_{ij}^{old} + s_{ij}^{\Delta t} \tag{2.3}$$

$$f_{ij}^{new} = \beta f_{ij}^{old} + f_{ij}^{\Delta t} \tag{2.4}$$

where $\beta \in [0, 1]$ In [48], the update mechanism is modified considering consecutive drop cases as following:

$$s_{ij}^{new} = \beta s_{ij}^{old} + s_{ij}^{\Delta t} \tag{2.5}$$

$$f_{ij}^{new} = (1 - \beta) f_{ij}^{old} + f_{ij}^{\Delta t} \tag{2.6}$$

where $\beta \in [0, 0.4]$.

In addition to trust computation, different punishment mechanisms are considered as a supplementary process for trusted routing [49, 50]. Before the punishment phase, malicious nodes are determined with a certain threshold. The malicious ones are isolated from the system for a certain time. After a certain time, the trust of the node is set to the threshold and the node is included in the system again. In [33], nodes are classified as forwarding node (FN) or non-forwarding node (NFN). Then, the modified Beta trust model is used. Moreover, a punishment mechanism for consecutive drop cases is proposed. The trust value of the node increases slowly for consecutive success while decreases fast in case of consecutive failures. Such mechanisms give an opportunity to a node to prove itself. In other words, the trust of the node increases when the node behaves correctly. In case of detection of malicious activity, the node is marked as a malicious again and insulated for another term of the isolated time.

Packet forwarding statistics are widely used as an input for the trust calculation. In [51], both direct and indirect trust components are used for calculating node trust. Direct trust is computed by using the number of successfully forwarded and the number of dropped packets. Indirect trust is computed by taking opinions of the neighbors. Then, the final trust of the node is computed as the weighted sum of both components. In [52], the proposed model uses only the packet forwarding ratio to compute history-based trust value of the node. Also, time decay mechanism is used for weighting the interactions differently. Node's current trust is calculated with the help of fuzzy logic

rules prediction method together with the social control theory. They describe two factors affected on current trust as node capability level including information about the CPU cycle, bandwidth, etc. and the historical trust level of the node.

Most of the studies use packet forwarding ratio as a metric for trust calculation but it is possible to use different metrics as well. In [53], new probability-based trust metrics to be used in ad hoc networks based on transit time variation, link failures, dropped packets, packet losses, congestion, and delays in the path are proposed. It is stated that the traffic patterns will be used for detecting the alteration of the communication status. Thus, there are several traffic parameters to observe the behaviors of the nodes. The parameters can be the probability of link or path failure, the probability of transit time variation, intermediate node congestion probability and end-node delay, the probability of delays at intermediate nodes, the probability of lost, inserted and multiplied packets, and the probability of normal traffic. Selection of appropriate metrics depends on the environment and the definition of trust.

2.2.3.2. Trust based routing in Wireless Sensor Networks. In recent years, the trust becomes important for wireless sensor networks so as to increase reliability among the nodes. Due to the resource constraint environment, the complexity of the operations has a significant importance. Such conditions require to deploy simple and practical mechanisms which cause several security vulnerabilities. Even if some cryptographic mechanisms are already implemented, they do not solve the entire problem. However, trust is considered as an efficient mechanism for the sensor environment.

In [33], Bayesian-based trust model is proposed to calculate the trustworthiness of the nodes. Then, the model is extended to cover all layers in the network protocol stack aiming to support several applications. Each node observes the neighbors in terms of location information, cryptographic correctness, communication parameters, energy usage, data forwarding ratio and the number of recommendation packets sent or received [32].

In [54], both the energy consumption of the sensor node and data forwarding statistics are used for node trust calculation. The model uses historical behaviors in direct trust component by using nodes forwarding statistics and current behaviors in indirect trust component by using neighbors recommendations. According to the Bayesian beta distribution family based model, the overall trust of *node i* on *node j* can be computed as follows:

$$R_{ij} = \frac{R_{ij}(D) + R_{ij}(ID)}{1 + \sum_k R_{ik}} \tag{2.7}$$

where j is the index of routing node, k is the index of neighbor of both i and j, $R_{ij}(ID)$ is indirect and $R_{ij}(D)$ is direct trust component, and R_{ik} is the general trust assessment of *node i* on *node k*. The energy constraint due to the environmental conditions become a parameter in decision criteria as following:

$$\text{Decision value} = R_{ij}.W_{trust} + E_{residual}.W_{energy} \tag{2.8}$$

2.2.3.3. Trust based routing in other networks. Trust based routing mechanisms have a wide variety of applications in several networks. In [49], Beta trust model is modified for cognitive radio networks. The parameters update process is modified as follows:

$$s_{ij}^{\text{new}} = \lambda.s_{ij}^{\text{old}} + s_{ij}^{\Delta t} + \sum_{k \in \Omega} I(s_{kj}) \tag{2.9}$$

$$f_{ij}^{\text{new}} = \lambda.f_{ij}^{\text{old}} + f_{ij}^{\Delta t} + \sum_{k \in \Omega} I(f_{kj}) \tag{2.10}$$

where Ω represents the set of all neighboring nodes of *node j*. In the above equations, the first component represents the weighted version of the old records and the second component represents the new value obtained after the time period Δt. The third component is the indirect trust component. However, the indirect trust values are

obtained from the third party nodes which may cause false evaluations. Therefore, the update mechanism uses a combination of three components to get more accurate results.

In [55], a context-aware uncertainty trust model which is composed of the combination of the Bayesian approach with the cloud model is proposed. Cloud-based models are good for representing uncertain trust values but they are lack of the ability of context-aware trust calculation and dynamic update mechanism. Thus, the Bayesian approach is used for avoiding vulnerabilities in the cloud. The model has two important value that uncertainty and context information which is used explicitly in the trust.

There are also several works that use machine learning as an additional mechanism in the trust calculation procedure. In [56], generic machine learning based trust model which uses behavior history of the agents is proposed. Such models validates the potential transaction as successful or unsuccessful based on the similarities with the previous transactions. All of the past behaviors and the current collective information about the agents are used for forming a feature vector for an agent and then, the trust calculation engine selects the suitable machine learning algorithm and gives the likelihood of generating a successful/unsuccessful outcome. They also propose a mechanism named Local Knowledge Sharing Overlay Network (LKSON) in case of inexperienced agents in the evaluation of the potential transaction aiming to exchange of local information of the agents.

In [57, 58] a computational trust model has been proposed based on stereotypes similar to the real world. The model uses historical information about the agents. The agent, groups other agents using feature vectors of the agents by using entropy as decision criteria and then use these groups to estimate new agent's trust when it faces with a new agent by using the beta distribution based trust model. They also propose a d-Stereotype model for the cases that agents behave differently from the group. In order to handle that case, they modify the stereotype model with a dichotomy-based approach. Briefly, the d-stereotype model divides each group into two subgroups as

honest and dishonest ones. The only difference comes from the group trust calculations because of the subgroups.

Due to the inability of traditional mechanisms trust is widely used for different networks such as WSNs and MANETs to provide reliable routing. Thus, trust can be considered as a candidate solution for the problem of secure routing in software-defined networks.

In this chapter, we have reviewed concepts related to software-defined networking and trust as a background for the concepts proposed in the thesis. First, we have presented background information about software-defined networking architecture and applications. Then, we addressed the security vulnerabilities of the SDNs. Finally, we provide a brief literature overview about trust based secure routing approaches in different environments.

3. TRUST ENHANCED SECURE ROUTING IN SOFTWARE DEFINED NETWORKS

Lack of standardized security mechanisms and an inability of current protocols causes serious security vulnerabilities in SDN. Thus, switches are prone to several attacks and such vulnerabilities can be used by adversaries to target forwarding behavior of data plane. Therefore, adversary target data plane by using such switches to manipulate data plane operations. Then, compromised switches follow commands given by the adversary and cause security attacks such as incorrect forwarding attack or dropping packets attack. Moreover, constructing a certain forwarding path is a challenging task if some of the switches are compromised. Therefore, identifying malicious nodes and constructing secure routing path including trusted nodes are significantly important for SDN.

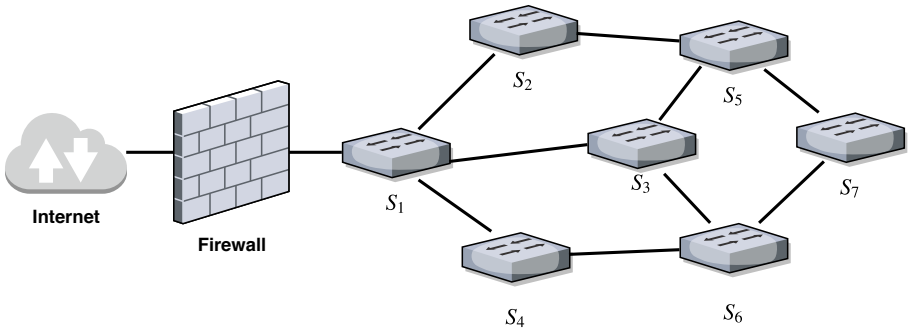

Figure 3.1. Security Approach in Traditional Networks

In traditional networks which as shown in Figure 3.1, the network traffic passes through the security device such as firewall and intrusion detection system before going to the Internet. Such a routing policy will be enforced by the security policy in the system. Passing through security device enables controlling and verifying the traffic. Therefore, malicious attacks can be detected and mitigated before introducing the network.

However, routing mechanism becomes different in SDN. As shown in Figure 3.2, traffic is forwarded based on the instructions given by the controller. A security policy may enforce flow to pass through the firewall if the flow seems suspicious. However, if S1 does not forward flow as instructed, flow could not be verified. In such a case, flow may forwarded through the red line and packet is delivered to receiver without being detected. Therefore, ensuring correct data plane operations is critical for SDN environment.

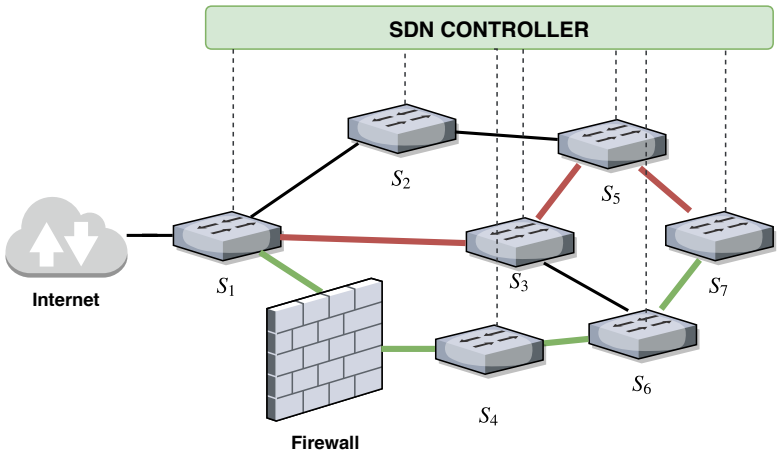

Figure 3.2. Security Approach in Software Defined Networks

The organization of the chapter is as follows. In the next section, we introduce a trust based model for SDN, namely Trust Enhanced Secure Routing (TESR) to evaluate the forwarding behavior of switches in data plane. In Section 3.2 and 3.3, we explain main modules of TESR, namely Trust Computation Module and Path Construction Module respectively. Finally, performance analysis and concluding remarks are given in Section 3.4.

3.1. TESR Overall View

An overall view of TESR is as shown in Figure 3.3. Data plane entities communicates with an Floodlight SDN controller through OpenFLow protocol. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_8\}$ be the set of OpenFlow enabled switches in the data plane and $\mathcal{U} = \{U_1, U_2, \ldots, U_{25}\}$ be the set of users in the network. Switches in S are programmed by

SDN controller to route the traffic generated by the users in U. We propose a trust based routing model which enables controller to construct secure communication path considering the requirements of an application.

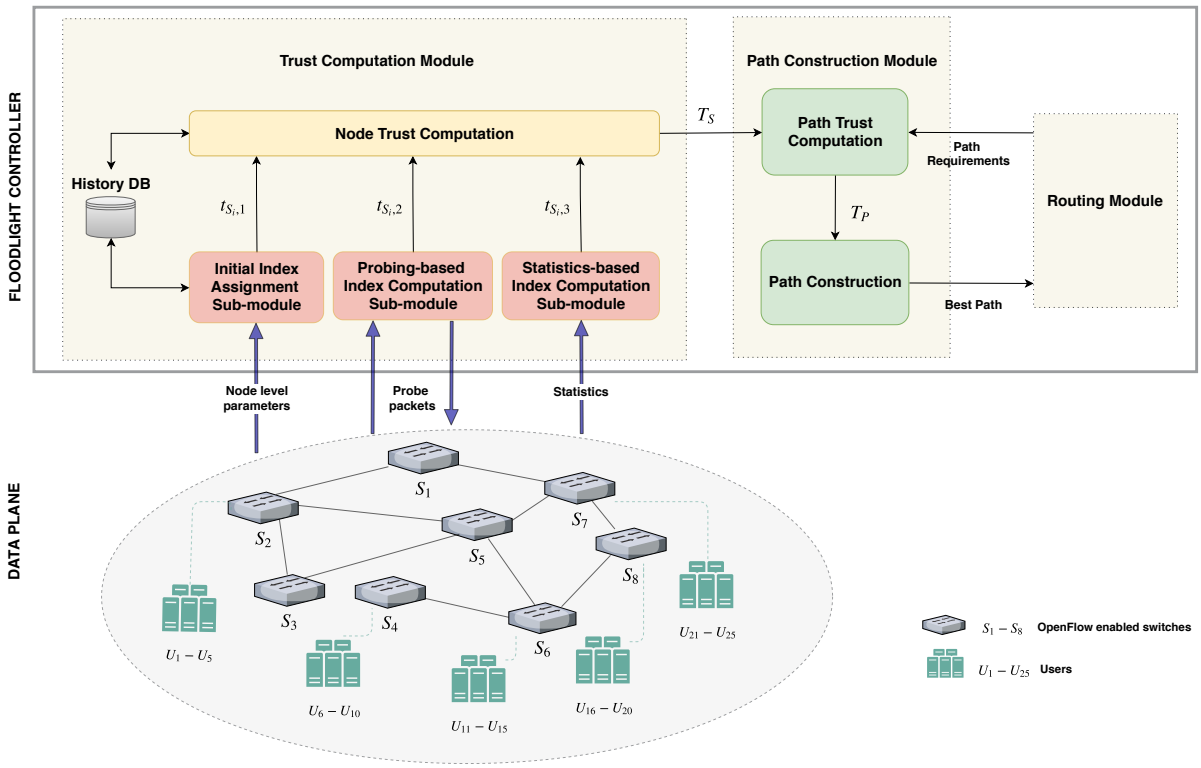

Figure 3.3. Detailed Block Diagram of TESR

TESR introduces two modules namely, Trust Computation Module and Path Construction Module. The Trust Computation Module interacts with the data plane to compute the trust indexes of data plane which are given by $t_{S_i,1}$, $t_{S_i,2}$, $t_{S_i,3}$. The module is responsible from obtaining initial configuration parameters, sending/receiving probe packets and collecting switch statistics from the nodes in S. Let T_S be the set of trust indexes of the switches in S, node trust computation block combines node trust indexes for all nodes in S and generates a set T_S. Then, the Path Construction Module uses T_S to evaluate communication paths according to their trust values. The path trust computation block finds k possible paths between given source and destination and computes path trust values using T_S. Let T_P be the set of trust indexes of the k paths, path construction block finds the best path which satisfies the path requirements. Routing module uses the best path to route the packets. In the following sections, we

give the details of Trust Computation and Path Construction modules.

3.2. Trust Computation Module

The Trust Computation Module, which is the combination of Initial Index Assignment, Probing-based Index Computation, and Statistics-based Index Computation sub-modules, is used for computing the trustworthiness of each switch in the network by monitoring the forwarding behavior of the data plane. Description of sub-modules are as follows:

- Initial Index Assignment Sub-module: We assume that there is no data plane statistics at the initial state of the network. Therefore, the module is used as a cold start approach for assigning initial trust indexes to switches.
- Probing-based Index Computation Sub-module: The module computes forwarding behavior of the data plane by sending probe packets to the network and verifies the forwarding path.
- Statistics-based Index Computation Sub-module: The module uses packet forwarding statistics of switches to estimate how a controller can trust a switch to accomplish packet forwarding successfully.

3.2.1. Initial Index Computation Sub-module:

In Initial Index Assignment sub-module, we check network level and node level parameters which are given in Table 3.1 and Table 3.2 respectively for assigning initial trust indexes to the switches. We determine the trustworthiness of the network by network level parameters. We also use node level parameters to determine the trustworthiness of the nodes.

The algorithm of the initial index assignment sub-module is given in Figure 3.4.

Table 3.1. Network Level Initial Index Assignment Criteria

Network Level Parameters	
System security architecture	Tools such as firewall and intrusion detection systems (IDS) can be used for monitoring network activities to detect malicious actions [59]. Therefore, such tools are important for networks so that, some of the attacks may be avoided before it happens.
Critical information storage protection	There may exist some critical information about the system that should be kept secure in order to avoid unauthorized use attempts [60]. For example, History DataBase (HDB) is critical for Trust Computation Module in TESR. Therefore, it is important to deploy a protection considering the physical and virtual access to the database.

Input: N: number of switches

$S = S_1...S_N$ set of switches in data plane

Output: $itcScore$: set of initial indexes for all switches

1: **for all** $S_i \in$ N **do**

2: check network and node level parameters for S_i

3: update $itcScore[i]$

4: **end for**

5: **return** $itcScore$

Figure 3.4. Initial Index Assignment Algorithm

Table 3.2. Node Level Initial Index Assignment Criteria

Node Level Parameters	
Physical access control	Switches in the network may be located in different places therefore, access control becomes important [61]. Only authorized people are allowed to access these places.
802.1x port-based authentication	802.1x protocol can be used for configuring role-based access control so that only authenticated users through RADIUS server gain access. There may exist user-specific access and resource allocation policies for authenticated users [62, 63].
MAC-based access control	Port security feature in switches can be used for restricting a port's ingress traffic by limiting the MAC addresses that are allowed to send traffic into the port [64].
Use of trusted computing module	Such platforms include hardware-based identity provisioning with public key cryptography, tampering detection, and response mechanisms. Therefore, they are useful for cases that software-only solutions fail to handle [65].
Patch level	Security patch includes few changes to correct the weakness described by a vulnerability. Therefore, it is important to be up-to-date. Even the operating system (OS) version is updated there may be some vulnerabilities which are required to be corrected by the new patch [59].
Firmware version	The firmware is a program to control the operation and the functionality of the switch. Therefore, upgrading the firmware improves the performance and brings enhanced features to device.

3.2.2. Probing-based Index Computation Sub-module:

In Probing-based Index Computation sub-module, we send probing packets to the network to validate whether the packet is forwarded to the right destination or not. The module starts with the selection of the switch and the flow to be tested. Some nodes in the system are more important than others because of their locations in the network [66]. We use such importance factor as a decision criterion for assigning probing period for switches. Therefore, critical nodes are identified and more attention is paid to these nodes before attacks. Polling important switches more frequently than the others is a very useful approach so that, system reliability can be improved.

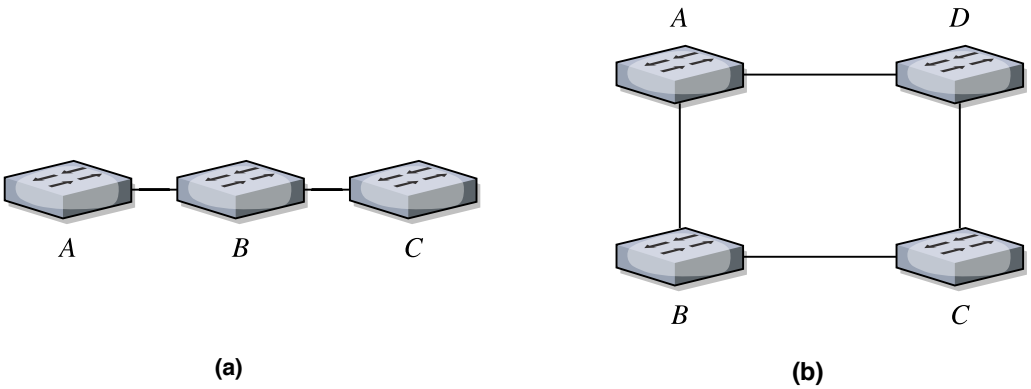

Figure 3.5. An example of Betweenness Centrality

An illustration of an example topology is given in Figure 3.5. In the first case, B has significant importance for the communication between A and C. There is no any node that enables such communication without using B. The betweenness centrality index is computed as 1. On the other hand, in the second case, B or D can be used for communication between A and C. The betweenness centrality index is computed as 0.5. Thus, betweenness centrality can identify critical critical nodes considering the communication paths.

We use betweenness centrality to determine the importance of the node in the topology [67]. The betweenness centrality index is given as follows [68]:

$$BC_{v_i} = \sum_{j,k \in N j \neq k} \frac{\sigma_{jk}(v_i)}{\sigma_{jk}} \tag{3.1}$$

where BC_{v_i} is the betweenness centrality of *node i*, N is the number of nodes in the topology, σ_{jk} is the number of shortest paths between all pairs and $\sigma_{jk}(v_i)$ is the number of shortest paths passes through *node i* to connect *node j* and *node k*.

Input: S_{test}: switch to be tested
K:set of neighbors of the S_{test}
Output: P_k: a result of the k th iteration probing
1: poll S_{test} for flow stats and select random flow f_{test}
2: set match M and action A as the same as in f_{test}
3: generate random `VLAN_ID` and set VID
4: build a match field M_{test} as the union of the M and VID
5: **for all** $S_i \in K$ **do**
6: insert flow rule f_i whose match field is M_{test} and action is controller
7: **end for**
8: create probe packet p as a packet out whose match field is M_{test}, the action is TABLE and payload data is randomly generated nonce
9: inject p into the NW
10: **if** collected copies come from the right switches as given in A and the payload data is not modified **then**
11: $P_k = 1$
12: **else**
13: $P_k = 0$
14: **end if**
15: **return** P_k

Figure 3.6. Probing-based Index Computation Algorithm

The algorithm of the probing-based index computation sub-module is given in Figure 3.6. Switches are polled with different frequencies depending on their locations in the network. After switch selection, random test flow is chosen and probing procedure is followed. If the controller verifies the probe packet is correctly handled, probing-based index is increased.

3.2.3. Statistics-based Index Computation Sub-module:

In Statistics-based Index Computation sub-module, we use Beta trust model which is applicable for both centralized and distributed settings [33, 48, 54]. We follow centralized approach that trusted controller determines the trust values of every switch in the data plane using statistics. The model basis on statistical derivations by using Beta probability density function [69]. Considering a process with two possible outcomes as $x, \overline{x}$, let r be the observed number of outcome x and s be the observed number of outcome $\overline{x}$. Then, the probability density function of observing x as an outcome in the future given using past observations as follows:

$$E(p) = \frac{r+1}{r+s+2} \tag{3.2}$$

We use the same binary approach by defining that r is the number of successfully forwarded packets to the next hop and s is the number of failed packets. We obtain r and s by polling the switches periodically. After k^{th} iteration, r and s values are updated as follows:

$$r_{new} = r_{old} + r_k \tag{3.3}$$

$$s_{new} = s_{old} + s_k \tag{3.4}$$

There exist several statistics which is specified in OpenFlow v1.3 specification as given in Table 3.4. We use such counters to obtain r and s by polling the switches periodically. The network wide statistics are collected in real time. Therefore, TESR can aware of the topology and react changes dynamically. In case that a switch is powered off or inaccessible for some time, TESR keeps the last statistics and mark that node as passive. When the switch becomes active, it is marked as active and started to be polled periodically.

The algorithm of the statistics-based index computation sub-module is given in Figure 3.7. Port statistics are requested from switches by statistics request messages and trust index is updated based on the new statistics values.

Input: $S = S_1...S_N$ where N is the number of switches
t_p: polling interval of switches
Output: T :a set of β trust indexes for all switches computed for each k seconds

```
1: while 1 do
2:    for S_k in S do
3:       poll S_k for port stats
4:       update r_k and s_k based on current stats
5:       T_k = (r_k+1)/(r_k+s_k+2)
6:    end for
7:    sleep(t_p)
8: end while
```

Figure 3.7. Statistics-based Index Computation Algorithm

Our trust model, TESR, combines three trust indexes as a vector to determine the trustworthiness of a switch. Let i be the index of a switch, trustworthiness of *switch* i is given as follows:

$$T_{S_i} =< t_{S_i,1}\, t_{S_i,2}\, t_{S_i,3} > \tag{3.5}$$

Table 3.4. List of OpenFlow Counters [3]

Counter	**Bits**	
Per Flow Table		
Reference count (active entries)	32	Required
Packet Lookups	64	Optional
Packet Matches	64	Optional
Per Flow Entry		
Received Packets	64	Optional
Received Bytes	64	Optional
Duration (seconds)	32	Required
Duration (nanoseconds)	32	Optional
Per Port		
Received Packets	64	Required
Transmitted Packets	64	Required
Received Bytes	64	Optional
Transmitted Bytes	64	Optional
Receive Drops	64	Optional
Transmit Drops	64	Optional
Receive Errors	64	Optional
Transmit Errors	64	Optional
Receive Frame Alignment Errors	64	Optional
Receive Overrun Errors	64	Optional
Receive CRC Errors	64	Optional
Collisions	64	Optional
Duration (seconds)	32	Required
Duration (nanoseconds)	32	Optional

where $t_{i,1}$ stands for initial index, $t_{i,2}$ stands for probing-based index, $t_{i,3}$ stands for statistics-based index, and $t_{i,j} \in [0,1]$ where $i \in \{1,..,N\}$ and $j \in \{1,2,3\}$.

In this section, we showed how the proposed module accomplishes the trust computation for a switch. The module communicates with the data plane to collect data from the switches and the collected data is processed into the sub-modules to generate trust index for the switch. In the next section, we explain how computed trust indexes are used for constructing secure routing path that satisfies the required trust level.

3.3. Path Construction Module

We use Path Construction Module to determine the route that satisfies the requirements of an application. Let T_c be the minimum required trust level and $\alpha =< \alpha_1, \alpha_2, \alpha_3 >$ be a set of weights for the initial, probing and statistics-based indexes respectively. Then, the given application security requirements are represented as follows:

$$SR_c = [\alpha_1, \alpha_2, \alpha_3, T_c] \tag{3.6}$$

The algorithm of the path construction module is given in Figure 3.8. The module starts finding possible routes from given source to destination using k shortest path algorithm. Then, path trust indexes are computed by using the switch trust indexes. The module returns with the maximum trusted path that satisfies the security requirements of the application, SR_c.

Let t_{sat} be the network saturation time which is highly correlated with the density of the network traffic. When there is a heavy traffic, t_{sat} becomes smaller. TESR uses only initial indexes until $t = t_{sat}$ to compute node trust. After $t = t_{sat}$, it includes weighted probing and statistics based indexes in trust computation procedure. Such an approach provides a secure communication even there is no switch statistics by using

Input: a: source switch

b :destination switch

$T_S = T_{S_1}...T_{S_N}$: a set of trust indexes of switches

t_{sat}: network saturation time

$SR_c = [\alpha_1, \alpha_2, \alpha_3, T_c]$ security requirement of client

Output: P_{best} : best path of all paths from a to b for time $t_{current}$

1: Compute possible paths $P_{paths} = P_1, ..., P_k$ from a to b using k-shortest path algorithm
2: **for** P_i in P_{paths} **do**
3: **if** $t_{current} < t_{sat}$ **then**
4: $T_{P_i} = \Pi_{S_m \in S_1 ... S_N}(t_{S_m,3})$
5: **else**
6: $T_{P_i} = \sum_{n=1}^{3} \alpha_n . \Pi_{S_m \in S_1 ... S_N}(t_{S_m,n})$
7: **end if**
8: **end for**
9: Find and return $min(||P_i - T_c||)$, $\forall P_i \in P_{paths}$

Figure 3.8. Path Computation Algorithm

the initial index assignment. Also, it guarantees that a switch cannot be fully trusted if some basic security features are not enabled.

3.4. Comments on the Performance of TESR

In this section, we discuss the time complexity of the proposed algorithms. Then, we evaluate the detection probability.

- Let N be the number of switches in the data plane, the initial index assignment algorithm runs in regular periods to check the existence of critical parameters with the $\mathcal{O}(N)$ time complexity.
- Let k be the number of neighbors of the tested switch, the time complexity of the incorrect forwarding detection algorithm for one iteration is $\mathcal{O}(k)$.

- The packet dropping attack detection algorithm which polls data plane in certain periods runs in $\mathcal{O}(N)$ time.
- The time complexity of the path construction algorithm mostly depends on the k-shortest path approach which is implemented with Yen's algorithm [70]. Let ADJ be the adjacency matrix of switches in S and c be the number of edges in ADJ, the algorithm runs in $\mathcal{O}(kN(c + NlogN))$ time.

The proposed algorithms do not cause performance overhead since all of the algorithms run with polynomial time.

3.4.1. Detection Probability Analysis

An incorrect forwarding attack occurs when an attacker compromises a switch and alters flows in the compromised switch maliciously. Malicious behavior can be the modification of either the match fields or action fields of an entry. Therefore, the detection performance of such attacks depends on both selecting compromised switch and selecting maliciously altered flow entry.

We evaluate the attack detection probability of probing approach and show an effect of the parameters on the detection performance. Let N is the number of OpenFlow switches in data plane and h is the number of flow entries in each switch. Suppose that an attacker compromises m switches and maliciously alters f flow entries in each switch. Let l be the number of selected switches where $l<m$, the probability of selecting one compromised switch is given as follows:

$$P_{\texttt{sw}} = \frac{C_1^m C_{l-1}^{n-m} + C_2^m C_{l-2}^{n-m} + \ldots + C_l^m C_0^{n-m}}{C_l^n} \tag{3.7}$$

$$= \sum_{j=0}^{l-1} \frac{C_{j+1}^m C_{l-j-1}^{n-m}}{C_l^n} \tag{3.8}$$

Then, the probability of selecting maliciously altered flow entry given as follows:

$$P_{\texttt{fl}} = (1 - \frac{C_1^{h-f}}{C_1^h}) + 2(\frac{C_1^{h-f}}{C_1^h})(1 - \frac{C_1^{h-f}}{C_1^h}) + \ldots + l(\frac{C_1^{h-f}}{C_1^h})^{l-1}(1 - \frac{C_1^{h-f}}{C_1^h}) \tag{3.9}$$

$$= \sum_{j=0}^{l-1} C_j^{j+1}(\frac{C_1^{h-f}}{C_1^h})^j(1 - \frac{C_1^{h-f}}{C_1^h}) \tag{3.10}$$

$$= \sum_{j=0}^{l-1}(j+1)(\frac{h-f}{h})^j(1 - \frac{h-f}{h}) \tag{3.11}$$

The probability of detecting an incorrect attack depends on both selecting compromised switch and selecting the maliciously altered flow entry from the selected switch. By combining Equation 3.8 and 3.11, attack detection probability is calculated as follows:

$$P = P_{\texttt{sw}}.P_{\texttt{fl}} = \sum_{j=0}^{l-1} \frac{C_{j+1}^m C_{l-j-1}^{n-m}}{C_l^n}.(j+1)(\frac{h-f}{h})^j(1 - \frac{h-f}{h}) \tag{3.12}$$

Then, the detection probability in case that selecting one switch is given as follows as expected.

$$P = \frac{m}{n}.\frac{f}{h} \tag{3.13}$$

As a consequence, attack detection probability is highly correlated with the number of compromised switches, the number of switches in the data plane, the number of maliciously altered flow entries and the number of flow entries in each switch. Selecting multiple switches and multiple flow entries for testing improves the detection probability.

3.4.2. Beta Trust Model Performance Analysis

In SDN network, we model packet forwarding statistics of the nodes as feedbacks which can take values 0 or 1. In Figure 3.9, we show how the trust index evolves as a function of accumulated feedback with varying feedback values.

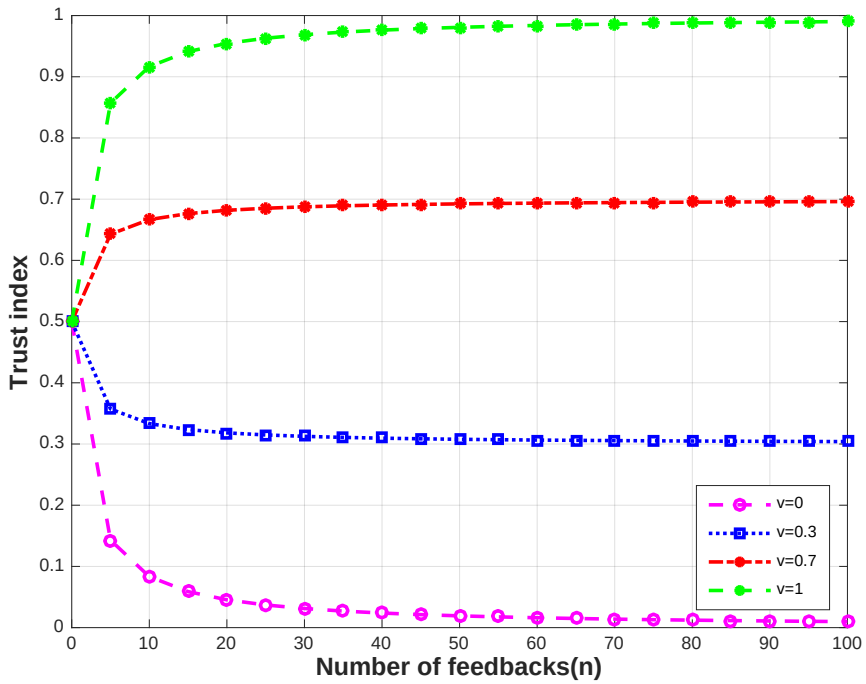

Figure 3.9. Trust Index as a Function of n with Varying Feedback v

However, in our simulation, feedbacks take binary values which shows successful transmission or failure as 1 or 0. Let a node provides n identical feedback values v. Mathematically, the r and s values can be expressed as following:

$$r = nv \tag{3.14}$$

$$s = n(1 - v) \tag{3.15}$$

Then, the trust index is computed as following:

$$T_v = \frac{r+1}{r+s+2} = \frac{nv+1}{n+2} \tag{3.16}$$

As shown in Figure 3.9, trust index approaches to 1 when $v = 1$ and approaches to 0 when $v = 0$ as expected. Consecutive successful transmissions ($v = 1$) increases trust gradually.

In this chapter, we proposed a trust based routing model for SDN, called TESR. The proposed model uses data plane observations to detect compromised nodes by evaluating the forwarding behavior of the switches. We explained the modules in TESR named, Trust Computation Module and Path Construction module in detail. Then, we discuss the performance of TESR in terms of the computational complexity and efficiency of the proposed algorithms.

4. SIMULATIONS ON THE APPLICABILITY OF TRUST ENHANCED SECURE ROUTING

In this chapter, we illustrate the simulation of our proposed model on different scenarios. Then, we evaluate the performance of TESR and show its behavior under different test cases. Finally, we discuss the practicality of TESR in terms of capturing real switch state, performance overhead and resource overhead.

4.1. Simulation of TESR on an Example Scenario

We simulated TESR in an emulated network using Mininet. Our prototype runs in Floodlight controller which is an open source SDN controller written in Java programming language by Big Switch Networks organization [71]. The current version of the Floodlight supports OpenFlow protocol version 1.3. It is also possible to use the Web GUI (Graphic User Interface) of Floodlight, that is an easy way to see what is going on the network and debugging purposes. In order to simulate the real SDN environment, we use Scapy as a traffic generator for the evaluation of our model. We configured the simple scenario shown in Figure 4.1, with a Floodlight controller and seven OpenFlow enabled switches.

In the initial state of the network, the Initial Index Assignment sub-module initializes trust indexes for the nodes by checking the parameters given in Table 3.1 and 3.2. For instance, S1 satisfies all network level parameters but MAC-based access control feature is not supported. Then, the initial trust index for S1 is set as 0.88 since we have eight parameters and each has an equal contribution to the computed index.

Trust indexes assigned by the Initial Index Assignment sub-module is used until the network saturation time, t_{sat}. After $t = t_{sat}$, probing and statistics based indexes are started to compute.

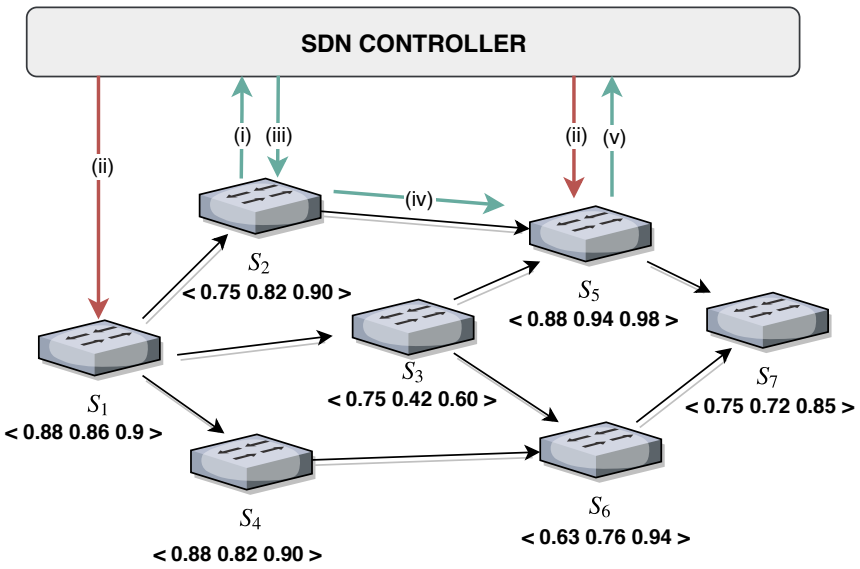

Figure 4.1. An Illustration of Example Scenario

In our scenario, the Probing-based Index Computation sub-module selects S2 to test, its operation steps are as follows:

(i) Random flow from S2 is selected to test.

(ii) New flow rule is inserted into S1 and S5 to trap the probe packet comes from S2.

(iii) Controller creates a probe packet by using `VLAN_ID` header field to differentiate the packet from the regular network traffic whose `VLAN_ID` is 0. Match fields are set as specified in the algorithm (Figure 3.6). The action field is set as TABLE to treat packet as any packet received from an input port. Then, the probe packet is sent as a packet-out message to S2.

(iv) The received packet is processed by S2 and forwarded to its next hop, S5.

(v) Since we configure neighbors to listen to probe packet comes from S2, S5 traps that packet and send to the controller.

Controller checks received packets to find out where the original probe packet is forwarded. Therefore, we can detect two incorrect forwarding behaviors as follows:

- If S2 floods the packet, S1 also sends the trapped probe packet to the controller. Therefore, we can detect whether the packet is flooded to the network or not.
- If controller gets a response from S1 rather than S5 means that packet is forwarded to the wrong destination. Therefore, we can detect incorrect forwarding behavior of S2.

The proposed probing approach against replay attacks and compromised switch collaboration. In case of a malicious node collaboration, a tested switch may disturb the packet content, incorrectly forward the packet or drop. If S2 and S5 are both malicious, S5 may pretend as it received the packet from S2 to manipulate the controller decision. We use random nonce as a payload data and random `VLAN_ID` for probe packets to avoid such attacks.

The Statistics-based Index Computation sub-module polls switches with regular periods. For instance, the controller polls S3, S3 replies back using `STATISTICS_REPLY` message including its port statistics. Then, the forwarding counters are updated and related trust index is computed.

Trust computation module monitors the data plane continuously and updates related trust indexes. The path construction module is triggered when the controller receives a packet with the given source and destination nodes. For instance, if a packet source is S1 and the destination is S7, the packet reaches to the network. Then, the path construction process starts with finding k shortest routes. There are four possible paths between S1 and S7 as illustrated in Table 4.1. According to the specified path requirements, we compute path trust indexes and the most trusted path is chosen as P2.

4.2. Performance Evaluation of TESR

We have tested our model over randomly generated topologies including 50 switches. We evaluate the attack detection performance of TESR in terms of both malicious rate

Table 4.1. Path Trust Computation Example

Path	**Path Trust Vector**	**Path Trust (α=<0.2 0.3 0.5>)**
P1:S1-S4-S6-S7	<0.37 0.39 0.65>	0.52
P2:S1-S2-S5-S7	<0.44 0.48 0.67>	0.57
P3:S1-S3-S5-S7	<0.44 0.58 0.45>	0.49
P4:S1-S3-S6-S7	<0.31 0.20 0.43>	0.34

in the network and the number of iterations. Detection performances of the betweennes centrality based polling and random polling are shown in Figure 4.2 and 4.3 respectively.

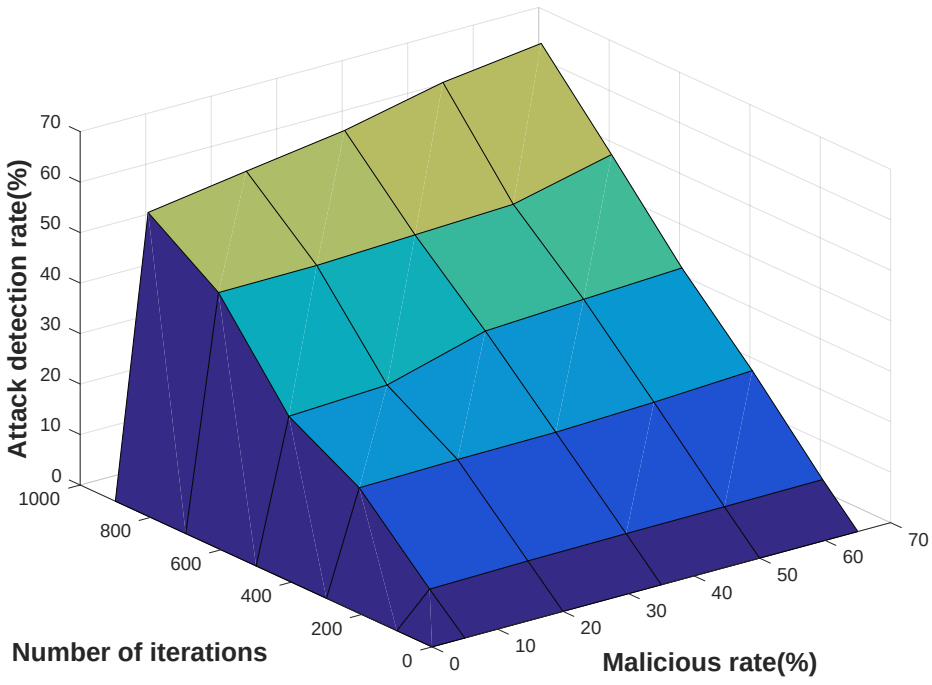

Figure 4.2. Attack Detection Performance of Betweennes Centrality Based Polling Approach

Results shows that betweennes centrality based polling performs better than the random polling approach. While betweennes centrality based polling approach gets its highest value 67.83%, the highest value of random polling approach is 34.05%.

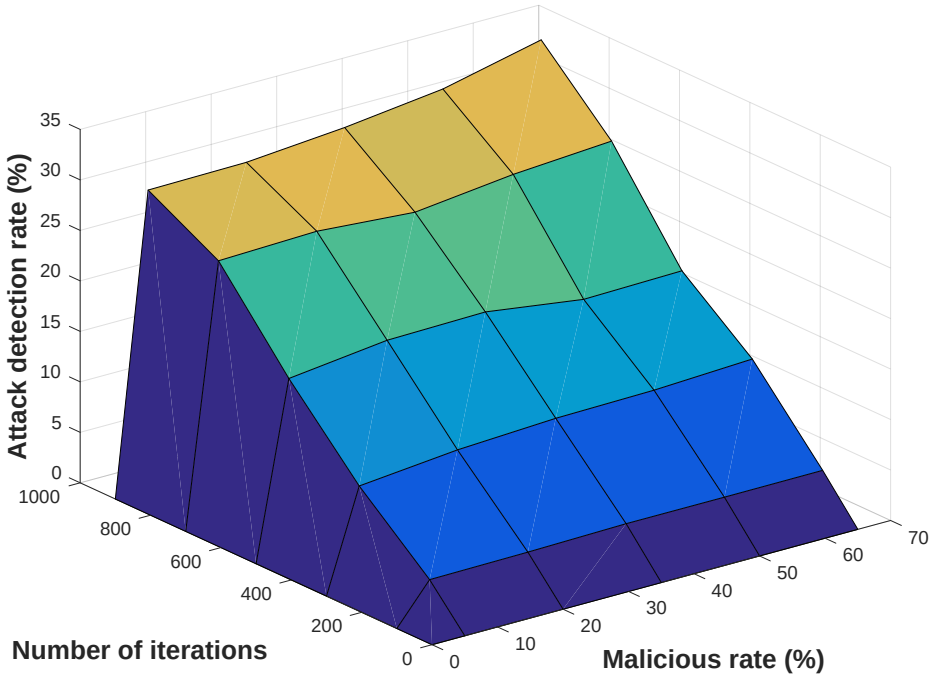

Figure 4.3. Attack Detection Performance of Random Polling Approach

Increasing number of iterations and malicious rate increase the attack detection rate as expected. However, detection rate is not affected from the increasing malicious rate below 200 iterations.

In Figure 4.4, the x-axes represent the number of iterations that simulation runs and y-axes represent the attack detection rate. We set the malicious rate to 65%. We have tested our approach against the random selection approach for both densely compromised and sparsely compromised cases namely, type 1 and type 2. Simulation results show that the idea of sampling important switches more frequently improves the detection of malicious behavior.

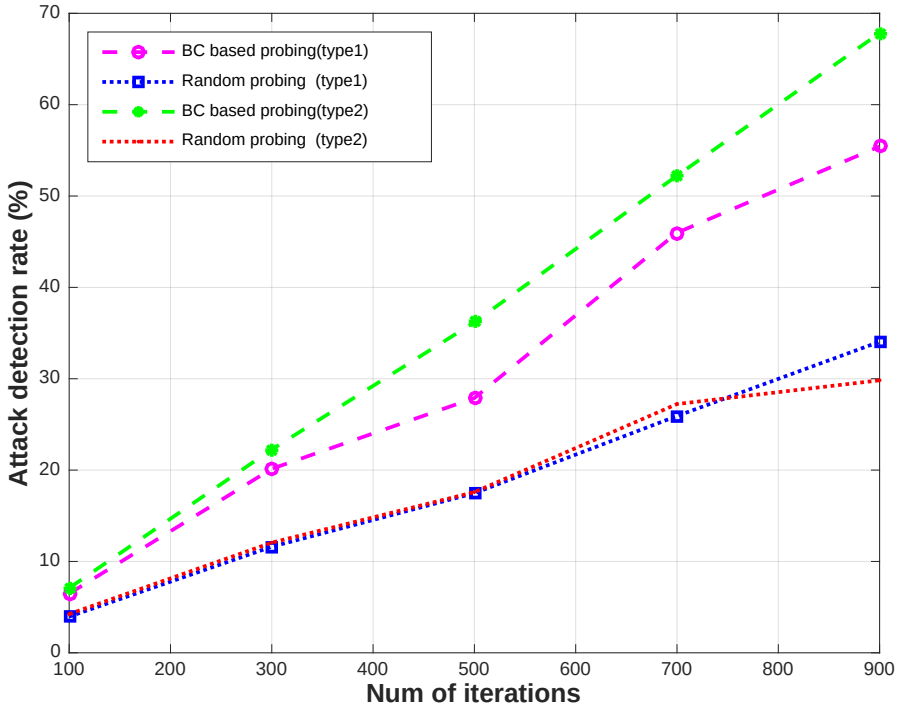

Figure 4.4. Detection Probability vs. Number of Iterations in the Densely/Sparsely Compromised Case

We also investigate the attack detection performance by increasing malicious rate as shown in Figure 4.5. We set the number of iterations at 900 and we run our prototype to see how the attack detection rate changes with the network maliciousity of the network. As shown in the simulation results, our model is able to detect most of the attacks even if the network is sparsely compromised.

In Figure 4.6, we compare the behaviors of Beta trust model and probability-based trust model. We use Equation 3.2 for the Beta trust model while we are using Equation 4.1 for the probability-based trust model.

$$Tp_i = \frac{r}{r+s} \tag{4.1}$$

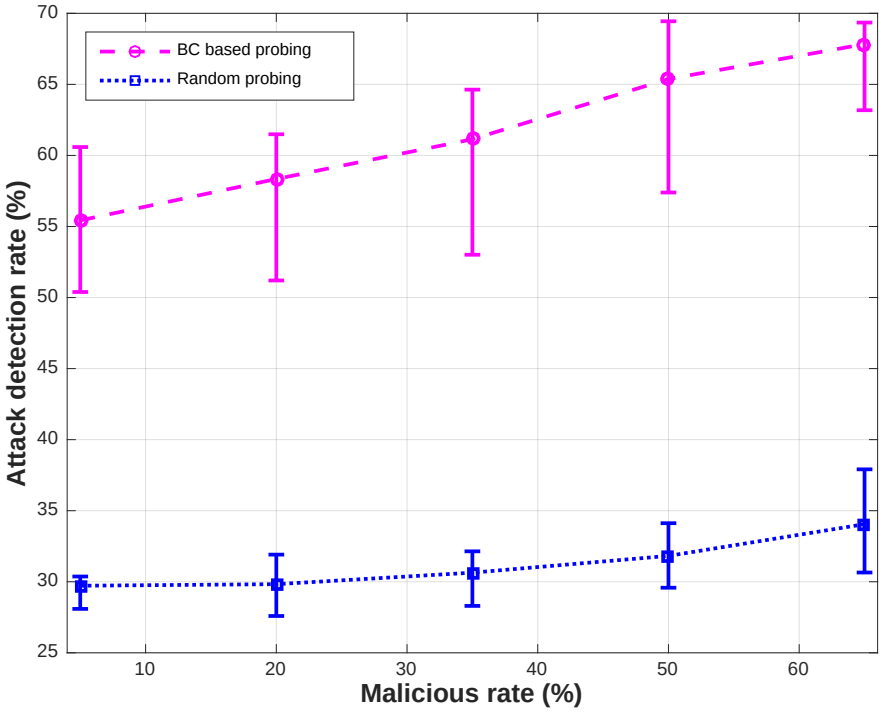

Figure 4.5. Detection Probability vs. Malicious Rate

where r is the number of successfully forwarded packets to next hop and s is the number of failed packets for *node i*. As shown in simulation results, Beta trust model and the probability-based model both converge to 1 when number of successfully transmitted packets increases. However, Beta trust model provides fine-grained trust index for the initial state of the network.

In order to show the behaviors of the models, we simulate a simple scenario as shown in Figure 4.7. The packet drop rate is constant during the simulation. The only exception occurs in case that the total number of transmitted packets is between 50 and 80. In such interval drop rate is fixed to 0. Therefore, trust indexes are both increasing. Both models behaves similar but the Beta trust model enables fine-grained index especially for low drop cases.

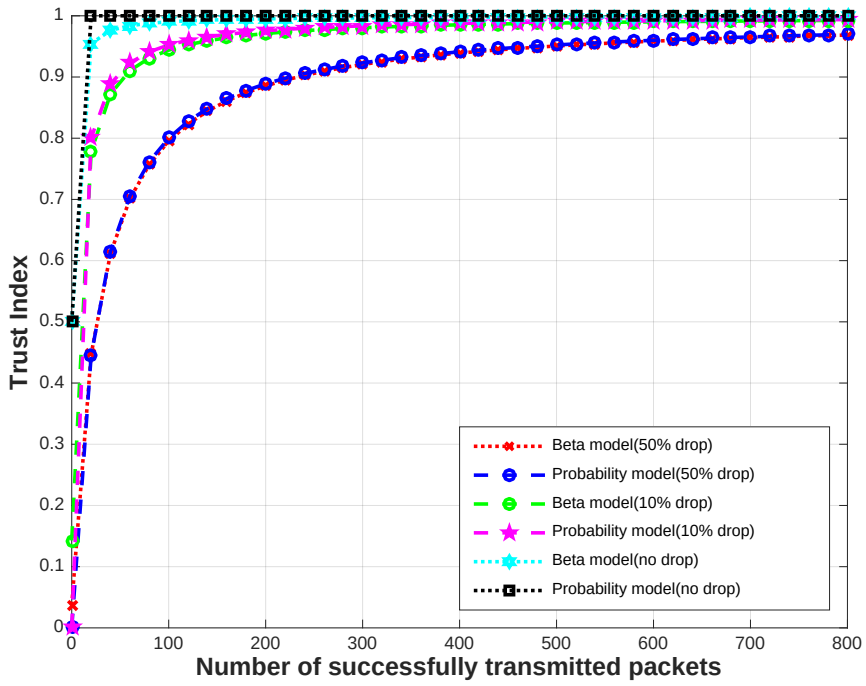

Figure 4.6. Comparison of Beta Trust Model vs Probability Based Trust Model

4.3. Applicability and Practicality of TESR

Comments for evaluating the applicability and the practicality of TESR are as follows:

- Performance Overhead and Resource Overhead: TESR is practical and efficient in terms of performance overhead and resource overhead. First, TESR requires adding one flow rule to the neighbors to trap the probe packets. It is extremely less than the TCAM capacity of switches. Second, TESR monitors and evaluates the forwarding behavior in an effective manner. An alternative to TESR would be continuous monitoring of the data plane. However, such mechanisms cause too much resource overhead and decrease the performance. Instead, we use the betweenness centrality based sampling approach to determine the polling frequency of switches. TESR supports dynamic topology changes, therefore, changes in the

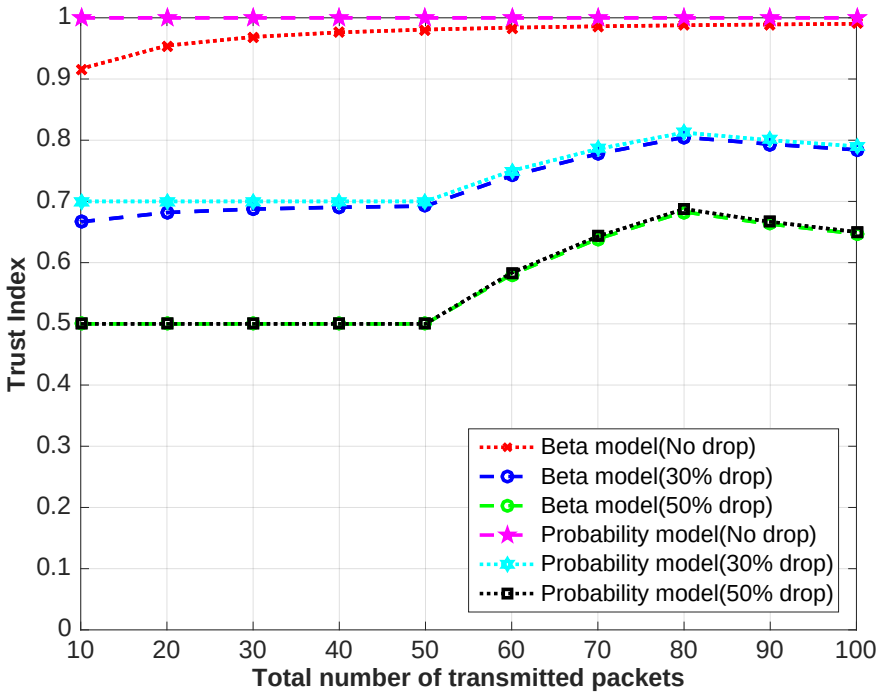

Figure 4.7. Comparison of Beta Trust Model vs Probability Based Trust Model

topology trigger TESR to recalculate polling frequencies.

- Capturing real switch state: Switches have limited TCAM space which prevents installing several predefined static rules. Therefore, the most of the SDN controllers install flow rules reactively into switches. Instead of creating artificial flows, we get flow samples actively from the switches to evaluate the forwarding behavior of the switch. Therefore, we are able to analyze switches with more realistic test cases. Additionally, TESR runs in real time, therefore, it reflects switch behaviors better than offline approaches.

As shown in the simulation results, TESR is an effective attack detection approach for SDN. Our model uses computational trust model for evaluating switches by regular monitoring of the data plane.

In this thesis, we use OpenFLow protocol for the communication between the controller and data plane entities. There exist other tools such as Programming Protocol-independent Packet Processors (P4) [72]. Unlike OpenFlow, P4 enables advanced packet processing and more fine-grained matching capabilities [73]. However, as far as we know P4 is not supported by the Floodlight controller.

In this chapter, we simulate TESR on an example scenario and we show the interaction of the modules with data plane. Then, we run several experiments to evaluate the performance and accuracy of our detection approach. Finally, we give a brief discussion about the practicality and applicability of our model.

5. CONCLUSION AND FUTURE WORKS

In this thesis, we investigate the problem of compromised switch detection in SDN. We propose Trust Enhanced Secure Routing by observing the forwarding behavior of the data plane. Observations are used for determining the trustworthiness of the switches in the data plane by computing the trust indexes. Then, the computed indexes are used for constructing secure routing path that satisfies requirements of an application. Our model helps controller to ensure correct data plane operations such as forwarding packet to the intended destination without modification. In addition, our model is able to detect compromised nodes successfully even if the network is sparsely compromised. Therefore, a controller is able to isolate compromised nodes from the network and offers special routing services depending on the requirements of an application. Moreover, simulation results show that we increase the detection rate of compromised switches by using adaptive sampling approach by polling important switches that involve in more communications than others increases the detection rate of compromised switches. Furthermore, as shown in simulation results, Beta trust model provides finer-grained indexes for TESR to increase the detection rate.

Future directions to improve TESR are as follows:

- We addressed two malicious actions as packet dropping and incorrect packet forwarding attacks in this thesis. Analysis and implementation for detecting different attacks rather than incorrect packet forwarding and packet dropping is left as a future work.
- We select a random flow per switch in each iteration, therefore, detection of compromised ones may take several iterations. TESR can be extended to make such selection in an efficient way so as to early detection of compromised switches.
- We determine the polling period of a switch by betweenness centrality. We plan to add different parameters to decide more efficient polling period for the switches. Therefore, the detection mechanism spends more time on significant switches

which increases the detection rate.

REFERENCES

1. Open Networking Foundation, *OpenFlow Switch Specification*, June 2012, version 1.3.0.

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

3. The Open Networking Foundation, *Software Defined Networking: The New Norm for Networks*, ONF white paper, 2012.

4. Chi, P.-W., C.-T. Kuo, J.-W. Guo and C.-L. Lei, "How to detect a compromised SDN switch", *2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pp. 1–6, IEEE, 2015.

5. Kamisiński, A. and C. Fung, "Flowmon: Detecting malicious switches in software-defined networks", *Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense*, pp. 39–45, ACM, 2015.

6. Chao, T.-W., Y.-M. Ke, B.-H. Chen, J.-L. Chen, C. J. Hsieh, S.-C. Lee and H.-C. Hsiao, "Securing data planes in software-defined networks", *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pp. 465–470, IEEE, 2016.

7. Khurshid, A., W. Zhou, M. Caesar and P. Godfrey, "Veriflow: Verifying network-wide invariants in real time", *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 49–54, ACM, 2012.

8. Zhang, X., C. Lan and A. Perrig, "Secure and scalable fault localization under dynamic traffic patterns", *IEEE Symposium on Security and Privacy (SP)*, pp. 317–331, IEEE, 2012.

9. Kim, T. H.-J., C. Basescu, L. Jia, S. B. Lee, Y.-C. Hu and A. Perrig, "Lightweight source authentication and path validation", *ACM SIGCOMM Computer Communication Review*, Vol. 44, pp. 271–282, ACM, 2014.

10. Zhang, X., Z. Zhou, H.-C. Hsiao and T. H.-J. Kim, "ShortMAC: Efficient Data-Plane Fault Localization", *Proc. NDSS*, 2012.

11. Wang, M., J. Liu, J. Mao, H. Cheng and J. Chen, "NSV-GUARD: constructing secure routing paths in Software Defined Networking", *Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom)*, pp. 293–300, IEEE, 2016.

12. Kreutz, D., F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-defined networking: A comprehensive survey", *Proceedings of the IEEE*, Vol. 103, No. 1, pp. 14–76, 2015.

13. Pfaff, B., B. Heller, D. Talayco, D. Erickson, G. Gibb, G. Appenzeller, J. Tourrilhes, J. Pettit, K. Yap, M. Casado *et al.*, *OpenFlow Switch Specification*, 2009.

14. Eldewahi, A. E., A. Hassan, K. Elbadawi and B. I. Barry, "The Analysis of MATE Attack in SDN Based on STRIDE Model", *International Conference on Emerging Internetworking, Data & Web Technologies*, pp. 901–910, Springer, 2018.

15. Paladi, N., "Towards secure SDN policy management", *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pp. 607–611, IEEE, 2015.

16. Betgé-Brezetz, S., G.-B. Kamga and M. Tazi, "Trust support for SDN controllers and virtualized network applications", *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pp. 1–5, IEEE, 2015.

17. Dao, N.-N., J. Kim, M. Park and S. Cho, "Adaptive Suspicious Prevention for

Defending DoS Attacks in SDN-Based Convergent Networks", *PloS one*, Vol. 11, No. 8, 2016.

18. Shang, G., P. Zhe, X. Bin, H. Aiqun and R. Kui, "FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks", *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, May 2017.

19. Grandison, T. and M. Sloman, "A survey of trust in internet applications", *IEEE Communications Surveys & Tutorials*, Vol. 3, No. 4, pp. 2–16, 2000.

20. Paladi, N. and C. Gehrmann, "TruSDN: Bootstrapping Trust in Cloud Network Infrastructure", *arXiv preprint arXiv:1702.04143*, 2017.

21. Ulema, M., *IEEE CQR 2014 International Workshop*, p. 21, Manhattan College, May 2014.

22. Open Networking Foundation, *Principles and Practices for Securing Software-Defined Networks*, January 2015.

23. Tseng, Y., Z. Zhang and F. Naït-Abdesselam, "SRV: Switch-based rules verification in software defined networking", *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*, pp. 477–482, IEEE, 2016.

24. Sasaki, T., C. Pappas, T. Lee, T. Hoefler and A. Perrig, "SDNsec: Forwarding Accountability for the SDN Data Plane", *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*, pp. 1–10, IEEE, 2016.

25. Porras, P., S. Shin, V. Yegneswaran, M. Fong, M. Tyson and G. Gu, "A security enforcement kernel for OpenFlow networks", *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 121–126, ACM, 2012.

26. Eichelberger, R. A., T. Ferreto, S. Tandel and P. A. P. Duarte, "SFC path tracer: a troubleshooting tool for service function chaining", *2017 IFIP/IEEE International*

Symposium on Integrated Network Management (IM2017), pp. 568–571, IEEE, 2017.

27. Zhang, Y., L. Cui, F. P. Tso and Y. Zhang, "Track: Tracerouting in SDN networks with arbitrary network functions", *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, pp. 1–6, IEEE, 2017.

28. Fayazbakhsh, S. K., V. Sekar, M. Yu and J. C. Mogul, "Flowtags: Enforcing network-wide policies in the presence of dynamic middlebox actions", *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 19–24, ACM, 2013.

29. Agarwal, K., E. Rozner, C. Dixon and J. Carter, "SDN traceroute: Tracing SDN forwarding without changing network behavior", *Proceedings of the third workshop on Hot topics in software defined networking*, pp. 145–150, ACM, 2014.

30. Zhao, Y., P. Zhang and Y. Jin, "Netography: Troubleshoot your network with packet behavior in SDN", *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, pp. 878–882, IEEE, 2016.

31. Naous, J., M. Walfish, A. Nicolosi, D. Mazières, M. Miller and A. Seehra, "Verifying and enforcing network paths with icing", *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, p. 30, ACM, 2011.

32. Geetha, V. and K. Chandrasekaran, "A distributed trust based secure communication framework for wireless sensor network", *Wireless Sensor Network*, Vol. 6, No. 09, p. 173, 2014.

33. Geetha, V. and K. Chandrasekaran, "Enhanced Beta Trust Model for Identifying Insider Attacks in Wireless Sensor Networks", *International Journal of Computer Science and Network Security (IJCSNS)*, Vol. 13, No. 8, p. 14, 2013.

34. Wang, Y. D. and H. H. Emurian, "An overview of online trust: Concepts, elements,

and implications", *Computers in human behavior*, Vol. 21, No. 1, pp. 105–125, 2005.

35. Cambridge, *Cambridge Dictionary*, 2017, `http://dictionary.cambridge.org`, accessed at July 2018.

36. Webster, *Merriam-Webster Dictionary*, 2017, `https://www.merriam-webster.com`, accessed at July 2018.

37. Gambetta, D., ""Can We Trust Trust?"", *Trust: Making and Breaking Cooperative Relations*, pp. 213–237, 1988.

38. Dasgupta, P., "Trust as a commodity", *Trust: Making and breaking cooperative relations*, Vol. 4, pp. 49–72, Citeseer, 2000.

39. Mui, L., M. Mohtashemi and A. Halberstadt, "A computational model of trust and reputation", *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pp. 2431–2439, January 2002.

40. Jones, S., M. Wilikens, P. Morris and M. Masera, "Trust requirements in e-business", *Communications of the ACM*, Vol. 43, No. 12, pp. 81–87, 2000.

41. Jøsang, A., R. Hayward and S. Pope, "Trust network analysis with subjective logic", *Proceedings of the 29th Australasian Computer Science Conference-Volume 48*, pp. 85–94, Australian Computer Society, Inc., 2006.

42. Nefti, S., F. Meziane and K. Kasiran, "A fuzzy trust model for e-commerce", *Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*, pp. 401–404, July 2005.

43. Luo, J., X. Liu and M. Fan, "A trust model based on fuzzy recommendation for mobile ad-hoc networks", *Computer Networks*, Vol. 53, No. 14, pp. 2396–2407, 2009.

44. Komathy, K. and P. Narayanasamy, "Trust-based evolutionary game model assisting AODV routing against selfishness", *Journal of Network and Computer Applications*, Vol. 31, No. 4, pp. 446–471, 2008.

45. Molva, P. M.-R., *Game theoretic analysis of security in mobile ad hoc networks*, Tech. rep., Citeseer, 2002.

46. Pirzada, A. A. and C. McDonald, "Establishing trust in pure ad-hoc networks", *Proceedings of the 27th Australasian conference on Computer science-Volume 26*, pp. 47–54, Australian Computer Society, Inc., 2004.

47. Pushpa, A. M., "Trust based secure routing in AODV routing protocol", *Internet Multimedia Services Architecture and Applications (IMSAA), 2009 IEEE International Conference on*, pp. 1–6, IEEE, 2009.

48. Chen, H., H. Wu, X. Zhou and C. Gao, "Agent-based trust model in wireless sensor networks", *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, Vol. 3, pp. 119–124, IEEE, 2007.

49. Zhang, G., Z. Chen, L. Tian and D. Zhang, "Using Trust to Establish a Secure Routing Model in Cognitive Radio Network", *PloS one*, Vol. 10, No. 9, p. e0139326, 2015.

50. Duraimurugan, A., S. Karthi and D. Kirupagaran, "A Trust Based Secure Source Routing using Fuzzy Logic Rules Prediction in Mobile Ad Hoc Networks", *International Journal of Engineering and Innovative Technology (IJEIT*, Vol. 2, No. 10, 2013.

51. Sardar, M. and K. Majumder, "A new trust based secure routing scheme in manet", *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013*, pp. 321–328, Springer, 2014.

52. Xia, H., Z. Jia, X. Li, L. Ju and E. H.-M. Sha, "Trust prediction and trust-based source routing in mobile ad hoc networks", *Ad Hoc Networks*, Vol. 11, No. 7, pp. 2096–2114, 2013.

53. Umuhoza, D., *Metric of trust for mobile ad hoc networks using source routing algorithms*, Ph.D. Thesis, University of the Western Cape, 2006.

54. Chen, D., X. Yu and X. Dong, "Research on Beta Trust Model of Wireless Sensor Networks Based on Energy Load Balancing", *Wireless Sensor Network*, Vol. 2, No. 05, p. 373, 2010.

55. Jin, B., Y. Wang, Z. Liu and J. Xue, "A Trust Model Based on Cloud Model and Bayesian Networks", *Procedia Environmental Sciences*, Vol. 11, pp. 452 – 459, 2011, `http://www.sciencedirect.com/science/article/pii/S1878029611008954`, 2011 2nd International Conference on Challenges in Environmental Science and Computer Engineering (CESCE 2011).

56. Liu, X., G. Tredan and A. Datta, "A generic trust framework for large-scale open systems using machine learning", *ArXiv e-prints*, March 2011.

57. Liu, X., A. Datta and K. Rzadca, "Trust beyond reputation: A computational trust model based on stereotypes", *Electronic Commerce Research and Applications*, Vol. 12, No. 1, pp. 24–39, 2013.

58. Liu, X., A. Datta and K. Rzadca, "Trust beyond reputation: A computational trust model based on stereotypes", *arXiv preprint arXiv:1103.2215*, 2011.

59. McBride, M., M. Cohn, S. Deshpande, M. Kaushik, M. Mathews and S. Nathan, "SDN Security Considerations in the Data Center", Open Networking Foundation, October 2013.

60. Nickolov, E., "Critical information infrastructure protection: analysis, evaluation and expectations", *CRITIS 2016 11th International Conference on Critical Infor-*

mation Infrastructures Security, pp. 105–119, 2016.

61. Hogg, S., "SDN Security Attack Vectors and SDN Hardening", *Artikkeli Network*, 2014.

62. Brocade, *Software-Defined Networking in the Campus Network*, Tech. rep., Brocade Communications Systems, 2014.

63. Benzekki, K., A. El Fergougui and A. El Belrhiti El Alaoui, "Devolving IEEE 802.1 X authentication capability to data plane in software-defined networking (SDN) architecture", *Security and Communication Networks*, Vol. 9, No. 17, pp. 4369–4377, 2016.

64. Cisco, *Cisco IOS Software Configuration Guide: Configuring Port Security*.

65. Jacquin, L., A. Lioy, D. R. Lopez, A. L. Shaw and T. Su, "The trust problem in modern network infrastructures", *Cyber Security and Privacy Forum*, pp. 116–127, Springer, 2015.

66. Hu, P., W. Fan and S. Mei, "Identifying node importance in complex networks", *Physica A: Statistical Mechanics and its Applications*, Vol. 429, pp. 169–176, 2015.

67. Raghavan Unnithan, S. K., B. Kannan and M. Jathavedan, "Betweenness centrality in Some classes of graphs", *International Journal of Combinatorics*, Vol. 2014, 2014.

68. Wang, J., L. Rong and T. Guo, "A new measure of node importance in complex networks with tunable parameters", *4th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM'08*, pp. 1–4, IEEE, 2008.

69. Ismail, R. and A. Josang, "The beta reputation system", *In Proceedings of the 15th bled electronic commerce conference*, Vol. 5, pp. 2502–2511, 2002.

70. Yen, J. Y., "Finding the k shortest loopless paths in a network", *management Science*, Vol. 17, No. 11, pp. 712–716, 1971.

71. Big Switch Networks, *Floodlight SDN Controller*, `https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview`.

72. Bosshart, P., D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors", *ACM SIGCOMM Computer Communication Review*, Vol. 44, No. 3, pp. 87–95, 2014.

73. Von Tüllenburg, F. and T. Pfeiffenberger, "Concepts for Reliable Communication in a Software-Defined Network Architecture", S. Tonetta, E. Schoitsch and F. Bitsch (Editors), *Computer Safety, Reliability, and Security*, pp. 173–186, Springer International Publishing, Cham, 2017.