I. J. Computer Network and Information Security, **2024, 5, 46-56**
Published Online on October 8, 2024 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2024.05.05

An Efficient Approach for Detection of Compromised SDN Switches and Restoration of Network Flow

Tinku Adhikari*
Mizoram University/Computer Engineering, Aizawl, 796009, India
Techno International Newtown/Information Technology, Kolkata, 700156, India
E-mail: tinku.adhikari@gmail.com
ORCID iD: https://orcid.org/0000-0001-6388-4307
*Corresponding Author

Ajoy Kumar Khan
Mizoram University/Computer Engineering, Aizawl, 796009, India
E-mail: ajoyiitg@gmail.com

Malay Kule
IIEST/CST, Shibpur, 711103, India
E-mail: malay.kule@gmail.com

Subhajit Das
Hiroshima University/HiSIM Research Center, Hiroshima, 739-8530, Japan
E-mail: sjdsubha@hiroshima-u.ac.jp

Received: 10 April 2023; Revised: 19 October 2023; Accepted: 27 December 2023; Published: 08 October 2024

Abstract: In Software Defined Networking (SDN) the data plane is separated from the controller plane to achieve better functionality than the traditional networking. Although this approach poses a lot of security vulnerabilities due to its centralized approach. One significant issue is compromised SDN switches because the switches are dumb in SDN architecture and in absence of any intelligence it can be a easy target to the attackers. If one or more switches are attacked and compromised by the attackers, then the whole network might be down or defunct. Therefore, in this work we have devised a strategy to successfully detect the compromised SDN switches, isolate them and then reconstruct the whole network flow again by bypassing the compromised switches. In our proposed approach of detection, we have used two controllers, one as primary and another as secondary which is used to run and validate our algorithm in the detection process. Flow reconstruction is the next job of the secondary controller which after execution is conveyed to the primary controller. A two-controller strategy has been used to balance the additional load of detection and reconstruction activity from the master controller and thus achieved a balanced outcome in terms of running time and CPU utilization. All the propositions are validated by experimental analysis of the results and compared with existing state of the art to satisfy our claim.

Index Terms: Compromised SDN Switch, Data Plane, Flow Reconstruction, CPU Overhead.

1. Introduction

Software Defined Networking (SDN) is now generally acknowledged by the academic and business community. SDN is being used in a growing number of real-world applications as a potential network virtualization technology. The detachment of the controller plane from data plane and the concentration of network controlling functions, which enable network programming capability, deliver rapid and flexible network administration, and encourage network innovation, are the main innovations of SDN [1]. But this new design also comes with significant security threats, some of which are never encountered in conventional networks, such as the susceptibility in SDN controllers, SDN switches, and SDN controller applications. Such a flaw is unavoidable and may be used by the attackers to seize switches and controllers.

This work is open access and licensed under the Creative Commons CC BY 4.0 License.

SDN switches that might be hacked could spy on private communications, block typical management and operation, and potentially harm the infrastructure. A corrupted switch can alter its own flow entries to implement malicious actions like packet dropping or unauthorized forwarding. Additionally, it can connect to a rogue controller and send fake OpenFlow messages which is one method an attacker might influence the control flow to his or her advantage. Added assault purpose is to trick the controller into thinking there are links that aren't there by taking advantage of the topology discovery function.

Therefore, it is crucial to identify compromised SDN switches as soon as possible. Otherwise, any network traffic via these corrupted switches may lead to any type of DoS or DDoS attack which eventually hamper the whole network. Isolation of compromised SDN switches and restoration of the network traffic flow is another important concern here. Maintaining the proper path from sender to receiver is a challenging task after the isolation of the compromised SDN switches because it requires a considerable amount of resource and there might be similar type of attack in the newly created path too. Therefore, only the removal of the compromised device may not be sufficient and hence we should remove all the links attached to them to maintain the network integrity. The detection of compromised switches, isolation of them and restoration of original network flow puts a considerable amount of burden on the controller and CPU but from the security perspective it is awfully important. Therefore, in our approach we have proposed deploying another secondary or backup controller along with the primary controller to work in tandem with the primary or master controller, in which all the aforesaid functionalities will be deployed. In our study we have used OpenDaylight (ODL) [2] as the master or primary controller and ONOS [3] as the slave or secondary controller for the prototype implementation and performance evaluation of our proposed algorithms. OpenFlow networks give users a single location where they can get data about participating switches, existing pathways, and flows, as well as swiftly calculate the best restoration strategies. The Path Computing Element (PCE), a specific computing platform in the SDN controller with the capability to take into account numerous restrictions for complex route calculation, is where the new pathways or diversions are computed. The SDN controller modifies the configuration of the compromised switches by adding or deleting flow entries from/to the associated switches with the flow tables of that path.

Data plane switches do not have any intelligence; all they do is transmit data packets towards the controller and other switches. This behavior creates a significant susceptibility that an attacker may take advantage of it. Network communications might be significantly disrupted by a rogue or malfunctioning SDN switch. There are five categories that may be used to group the abnormal behavior at an SDN switch.

- A corrupted or compromised switch may selectively or randomly stop passing network traffic, which is known as traffic loss.
- It is known as traffic misrouting when the traffic of a hijacked switch is sent to the incorrect address.
- Traffic reshaping, which is the process of altering the sequence of packets on a hacked switch.
- Contents of the traffic might be modified by the compromised switch which is known as traffic alteration.
- Traffic via a compromised switch might be delayed and poses a fatal threat in time bound communications known as traffic delay.

In this work an approach has been proposed to first detect the compromised switches, isolate them from the network and finally restore the network flow to maintain the sustainability of the whole network. The major contributions of this work are-

- In order to effectively detect compromised SDN switches with little overhead, we suggest using secondary controller(s) to verify the handling data of network updation data obtained from the main controller and its switches.
- We propose another solution approach to effectively restore the network flow after eliminating the compromised switches along with all subsequent connections to them.
- We outline the suggested approach, design the appropriate algorithm for detecting compromised SDN switches, and provide a thorough introduction to the prototype implementation evaluation.
- To validate the suggested techniques, we show the experimental findings and compared them with the existing state of the art solutions to establish the effectiveness of our algorithms.
- Finally, a brief security analysis has been presented to discuss the impact of our work on security aspects.

Figure 1 depicts the SDN architecture with some compromised switches. It can be seen from the figure that in a typical SDN architecture SDN controller acts as a central communicator between the applications residing in the application layer and the data plane switches. Applications communicate with the controller via the northbound interface (NBI) and controller communicates with the data plane switches via the southbound interface (SBI). Therefore, applications need controller intervention to install flow rules in the data plane switches. Now as seen in the following figure, if somehow one or more switch is compromised by the attackers the whole network becomes either disconnected or unreliable. Any traffic via these switches is vulnerable and may impact the whole network tremendously. Therefore, immediate detection of this type of compromised switches is essential as well as the subsequent activity of isolation and flow reconstruction.

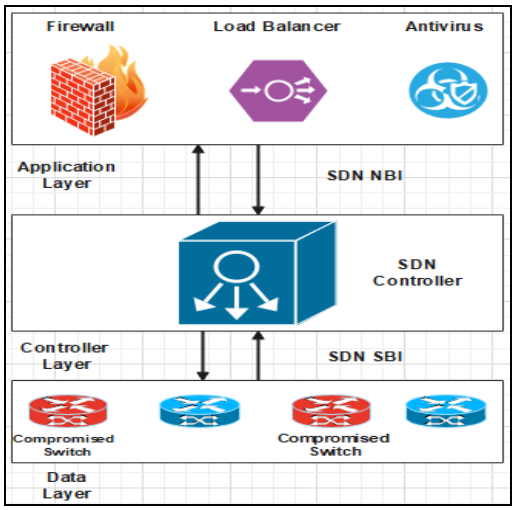

Fig.1. SDN Architecture with compromised switches

2. Related Works

SDN has faced significant concerns from attacks on the data plane. Attackers can disable the control plane or observe its activities by using numerous hosts connected to OpenFlow switches without having much knowledge of the controller apps. DoS, topological poisoning, and side-channel attacks are some of these attacks [4]. SPHINX [5], one of the existing systems, has usable implementations. By abstracting network activities using incremental flow graphs, it may identify and counteract security threats started by malicious switches. Additionally, it recognizes assaults in accordance with the administrator's defined regulations and takes appropriate action. The nearest approach to SPHINX is called WedgeTail [6], and it is known as an excellent intrusion prevention solution for the data plane of SDNs. This operation is independent of rules that have already been established by an administrator, and it is capable of quickly identifying and addressing any implanted harmful forwarding devices. Two anomaly detection algorithms—Packet Droppers and Packet Swappers are proposed in another study, FlowMon [7]. The controller examines the real forwarding pathways and the port information gathered to find malicious switches. The idea was put forth by Haeberlen et al.[8] to allow each connecting node to request the security log of the other node in order to freely assess whether the behavior of the node deviates from expected behavior. The security log of each node would record the messages received and sent by the network node. A proper node is able to defend itself against malicious allegations based on the intended temper-evident record, and a malfunctioning node will be found by executing the feedback analysis procedure on it on a regular basis.

In order to ensure a seamless transition, Botelho et al. [9] suggested designing a fault resistant SDN controller plane and storing application and network states in common data storage. In order to prevent controller hijack, Qi et al. [10] suggested a control plane design to support numerous disparate controllers and carried out a theoretical examination of its efficacy without presenting the system implementation. Every request for network update is handled by several different controllers, according to this study, and the controllers will vote to select how it will be handled in the end. Al-Shaer and Al-Haj [11] suggested a technique to identify SDN switches with numerous conflicting policies. For the purpose of identifying non-bypass property violations, Son et al. [12] suggested modelling flow tables entries with the Yices SMT solver. Between the switches and their corresponding controller, Khurshid et al. [13] introduced a layer called VeriFlow to check dynamically and in real-time for any violations throughout the entire network. By utilising a collection of conceptual tools those maintain a graph of dependency among rules, Kazemian et al. [14] introduced a real-time tool for policy checking called NetPlumber based on Header Space Analysis (HSA) [15] to gradually verify the conformity of state changes.

Throughout the years, several different path calculation methods for backbone networks have been presented. Single-failure restoration methods are categorized rather thoroughly in [16], along with a comparison of their restoration timeframes. In [17] the authors provide an illustration of how to formulate a capacity maximization issue for single-failure path reconstruction. Although while few research [18,19] have taken into account numerous failure situations, but no answers are provided for the optimization problems based on multiple failures. Internal processing time is another point of consideration here which is missing in most of the literature. In case of multiple failure scenario this processing time poses significant impact on overall path restoration process [20]. Determining the shortest path is not the main issue in this situation since rerouting a failed flow via it may not be effective in terms of lowering the flow table processing time. Moreover, it is demonstrated in [21] that restoration strategies based on shortest path that disregard the processing time of the flow table may be unable to rapidly restore network flows, particularly in big networks supporting numerous network flows.

3. Impact of Compromised SDN Switches

When a packet is sent from a host to its nearby SDN switch, the switches will first search for a comparable entry in the flow table based on the routing-related information included in the packet. If the particular switch is unable to locate a matched entry inside flow table, it will encapsulate the packet content as an OpenFlow [22] PACKET_IN message and send it to the controller. Following that, the controller will take the packet out of the PACKET_IN message and gather the routing information. The controller programs will handle the deployment of transmission route for any packet and thereafter appropriate instructions are provided to switches to deploy necessary data flow via the southbound interface.

The compromised SDN switches have the malicious impact on both the controller and switches, thereby eventually it will impact the whole network too. As we know that controller is the main component of the SDN architecture, any adverse effect on the controller will also affect the whole network. The major attack points of compromised switches on the controller as well as on the whole network are summarized below-

- A controller utilizing a compromised switch may be instructed by the attacker to deactivate its switches by erasing all the entries in its flow table.
- By using the compromised switches, the attacker can arbitrarily take and use the system resources that belong to authorized users.
- Until the attack goal is reached, the attacker may keep stealing controllers to control other switches attached to them.
- The attacker may alter the flow table entries and the network topology, which would affect regular network operations.
- The hijacking attempt might not be discovered by operators if the attacker tampers with the controller's log.

4. Insights of SDN Controllers

In our work we have used Open Daylight (ODL) as the primary controller and ONOS as the slave or secondary controller. The main responsibility of the primary controller is to maintain the normal traffic flow in the network whereas the main responsibility of the secondary controller is to dynamically check for any compromised switch(es) and to inform about this type of event(s) to the primary controller. As per the statistics provided by the backup controller, the primary controller then logically isolates compromised switch(es). Network flow restoration is the second major responsibility of the secondary controller and after creation of the new flow path this information is transmitted to the primary controller to maintain normal network traffic flow.

ODL and ONOS have been chosen as primary and secondary controller because using distributed approaches, modern distributed controllers, such as ODL and ONOS, have enabled the synchronization of network view between primary and secondary controllers. Consequently, the current synchronization technique may be used when the primary and secondary controllers are the same type of controller. If not, one of those distributed approaches must be implemented in each of the heterogeneous primary and secondary controllers to synchronize their network and application related states.

5. Proposed Solution Approach

We have bifurcated our proposed solution mechanism into two major parts- first the detection process of compromised switches is performed which triggers the next process of switch isolation and network path reconstruction. The principles of both these activities are stated below.

5.1. Principle of Compromised Switch Detection

Based on our investigation, it is essential to gather data of each network update event and the associated master controller and switch handling data in order to detect possibly compromised SDN switches in a reliable and timely manner. This is the first step to performing. Therefore, the following data are required to be collected primarily-

- Each request for network update.
- Network update instruction received by switches from both the controllers.
- Execution log of that request by the switches.

To collect this information as a whole set, an idiosyncratic ID is assigned for each network update request. The actions performed by the primary controller are mirrored in the secondary controller. Execution log of secondary controller and switches for any particular network update request ID contains information of network state update. The similar type of information is generated from master controller but is case of any comptonization of any switch its flow table entries does not match with which is obtained from the secondary controller. Therefore, these execution results

from primary and secondary controllers can be compared to find any inconsistency in the execution log of the switches. If any such inconsistency is detected, then the corresponding switch is treated as a compromised switch, and it is subject to be isolated from the network.

5.2. Principle of Switch Isolation and Network Flow Reconstruction

Once detected, compromised switches can be isolated logically by deleting their corresponding flow table entries from the primary controller. As we are maintaining similar copies of flow table entries for both the primary and secondary controller, flow table entries in the master controller get updated too. At this point the network is fully or partially unreachable due to the absence of the switches or links and therefore network reconstruction is essential at this point to maintain the normal operational flow.

To reconstruct the network flow, there are two major considerations- i.e. the link cost factor or link weightage between two switches and the flow table alteration cost (internal processing time) if any such link is inserted or removed from the existing flow table. In this study these two parameters are important because normally a network can be thought of as a connected directed graph where switches are like graph vertices and link between them are the edges. Removal or insertion of any edge means altering the existing flow table which incurs flow table alteration cost because subsequent flows must have to be altered to accommodate these changes. If there is a huge flow table modification then a considerable amount of processing power and time is wasted to perform this job, so flow table alteration cost should be minimized as far as possible. Another important concern is link weightage which might be thought as any feasible metric like distance, time etc. As we all know, that spanning tree is a tree which connects all the vertices of a graph and minimum cost spanning tree incurs minimum cost among them, MST is a good choice for network flow reconstruction. Therefore, from the set of all possible paths from source to destination, the path which will incur minimum link cost has been considered in this study assuming link cost factor as a constant, because internal processing time will pose serious time overhead on overall network and CPU performance if not tactfully managed and demands more research highlights. Detailed description of both the principles mentioned in 5.1 and 5.2 are presented in the next subsection.

5.3. System Model

According to the discussion in section 5.1 and 5.2 to detect compromised SDN switches four types of information are collected for each network update request under a single ID-

- The request for network update.
- The execution report by primary controller- E_M
- The execution report by the secondary controller- E_S
- The execution report by the switches- E_T

Based on the above collected information, the secondary controller can take the decision whether a switch is compromised or not by comparing the execution log of both the controllers. This is informed to the master controller immediately to perform the subsequent actions of switch isolation and path reconstruction. The following might be observed by the secondary controller against each ID-

- $E_M=E_S=E_T$ which implies all the network update operations have been performed successfully without any issue.
- $E_M \neq E_S$ which implies execution report of master and secondary controller are not same and switch tampering may occur.
- $E_S \neq E_T$ which implies execution report of secondary controller and switches are not matching and that switch has been compromised.

Let CON_M denote the master controller and CON_S denote the secondary controller. Req_i is the network update request for any ID denoted by ID_i where $i=1,2,3.....n$. S is the set of all switches controlled by CON_M, S_i is the set of all switches participating in Req_i.

The detection algorithm is presented in figure 2 which takes a set of network request updates from all legitimate sources and provides a set of compromised SDN switches. It can be observed from fig 2 that compromised switch detection algorithm takes the parameters mentioned above and checks the execution report of primary controller, secondary controller and the switches and loosely predicts any anomaly if execution report of primary and secondary controller des not match. Afterwards the algorithm finally detects the compromised switch if there is a mismatch between secondary controller log and the same of the switches.

At this point all the compromised SDN switches are detected and hence comes the subsequent part of the process i.e. isolation and network flow/path reconstruction. For isolation of the compromised SDN switches corresponding flow table entries are removed from master controller as well as from the secondary controller too.

Now at this point we can consider the whole network as a weighted directed graph $G(v,e)$ where v is the set of switches, e is the set of links between them, e_{ij} is a link between node v_i to v_j. Here the term switch and node are operationally synonymous and used interchangeably in this literature. Cumulative cost of link e_{ij} is denoted by $cost_{ij}>0$

which may be any metric like weightage, delay, hop distance etc. We denote nth source-target pair by S_n and D_n respectively.

Input: A pool of network update requests from legitimate sources
Output: A set of compromised SDN switches
1 **if** *Controller is* CON_M **then**
2 Send Req_i *(i=1,2,3…N)* to CON_S;
3 Execute Req_i
4 **end**
5 **else if** *Controller is* CON_S **then**
6 Generate $Record_i$ by using ID_i ;
7 Add Req_i to *Record*;
8 **if** $E_M \neq E_S$ **then**
9 The switch might be compromised;
10 **end**
11 **end**
12 **else if** $E_M = E_S$ then
13 $\forall E_T$ **if** $Req_i = ID_i \neq S_i$ *and* $E_S \neq E_T$ **then**
14 The switch has been compromised;
15 **end**
16 **end**

Fig.2. Detection algorithm of compromised SDN switches

In our approach to reconstruct the network flow we assume the following points-

- A link was there between S_n and D_n or can be established.
- P is a path between any two pair of nodes, and it contains no self-loop.
- The path cost denoted by $cost_{ij}$ is 1 if it traverses e_{ij} and 0 otherwise.
- Path cost is additive.

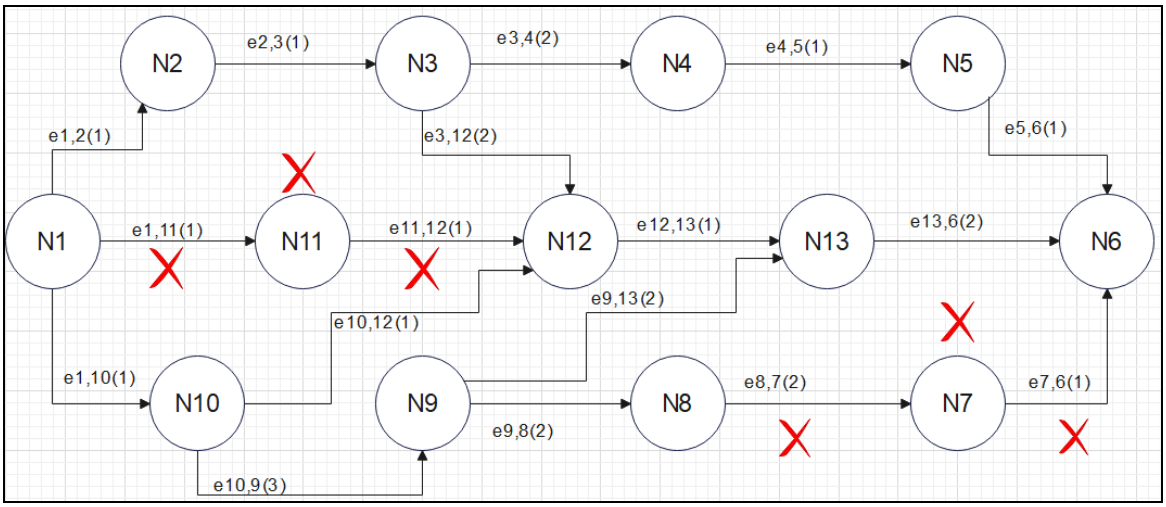

Fig.3. Network flow reconstruction with 13 nodes

An exemplary snapshot is presented in figure 3 and it can be noticed that in this network of 13 nodes there was a path between N1 and N6 via N11, N12, N13 and total link cost was 5 because $cost_{ij}$ is additive. At this point switch N11 and N7 were compromised, and their subsequent links were deleted from the flow table of the controller. Therefore, there are 4 alternatives with us to reach N6 from N1:P1=N1-N2-N3-N4-N5-N6, where $cost_{ij}$ =6, P2=N1-N2-N3-N12-N13-N6, where $cost_{ij}$ =7, P3=N1-N10-N12-N13-N6, where $cost_{ij}$ =5 and P4=N1-N10-N9-N13-N6, where $cost_{ij}$ =8. Hence P3 has been chosen as the newly constructed path because of minimum flow table alteration cost and corresponding flow table entries are updated in master and secondary controller as well.

Figure 4 depicts the network flow restoration algorithm which receives all the compromised SDN switches and associated links between them thereby providing a path by which switches can reconnect again after eliminating the compromised ones. It depicts that if there is a connection between two distinct nodes, then there might be different paths from these two nodes which may incur different cost values. Therefore, the minimum cost valued path has been chosen here as a path between two such nodes.

Input: A set of compromised switches and links in between.
Output: A path between any two pair of switches.
1 For any Source-destination pair S_n and D_n ;
2 If any switch is designated as a compromised then;
3 Calculate $Cost_{ij}$ for each path available from S_n to D_n;
4 Choose the path with smallest $Cost_{ij}$ value;
5 If there is more than one path with same $Cost_{ij}$ value then choose any one of them;

Fig.4. Network flow reconstruction algorithm

6. Testbed Setup, Implementation Result and Security Analysis

6.1. Testbed Setup

As per the prototype implementation of our proposed algorithm for detection of compromised switches, a random switch MAC and link flooding attack has been initiated from Macof tool of kali linux. Detection time and overhead on CPU performance or utilization has been measured for the compromised switch detection algorithm and flow path construction time along with overhead posed on CPU for running the second algorithm has been measured for network flow reconstruction algorithm. Similar types of metrices are used for both the algorithms to achieve better symmetrical view. After successful detection, a network flow reconstruction algorithm has been implemented on a virtual linear network of 25 hosts and 25 switches. Three virtual machines were deployed for the setup configuration and switches were deployed using Mininet emulator using a linear topology in the data layer. All the VMs were implemented on a windows 10 host running on Intel core i5 CPU 3.8 GHz with 16 GB of RAM. The snapshot of the testbed setup is given below-

- Controllers- Opendaylight version 14 (Silicon) has been used as a primary controller installed on Ubnuntu 20.04 LTS VM. The same virtual machine has been used for the secondary controller i.e. ONOS version 2.6.0 (woodpecker).
- Emulator- An emulator called Mininet has been used to simulate a total of 25 hosts, 25 switches and 625 internal paths among them in a linear topology. It is installed on another virtual machine based on a similar type of VM.
- Attacker Machine- A device which is used to initiate the switch MAC flooding and link flooding attack by Macof tool and has been installed on Kali Linux.

6.2. Prototype Implementation and Result Analysis

In the implementation scenario attack traffic has been generated by Macof tool installed in Kali linux which has generated both MAC flood and link flood traffic to overwhelm the switches and links in between. Afterwards both algorithms were implemented in the similar simulation environment described above to measure two major parameters i.e., running time and CPU overhead imposed by the algorithms. These two parameters are extremely important because larger detection time in compromised switch detection process increases the risk of overall DoS in the network and larger time in network restoration makes the whole network unusable. The major computing resource is CPU time therefore both the algorithms should be lightweight enough so that they can easily run and do not impose any major overhead on the overall network performance. We have compared both of our algorithms with the existing state of art and noticed significant improvement for both the parameters for both the algorithms which are described below.

In the experiment the PACKET_IN message has been selected as the initiator of the network update. Upon receiving the PACKET_IN message the master and secondary controllers execute this request and update their flow table. The switches send the execution reports to both controllers. For a switch whose report is different with master or secondary controller has been chosen as a compromised one. We have measured the average detection time after two successful packet and link flooding attacks and obtained 600 different values of the same. As per the experimental findings we found that all switches compromised by the attack have been detected successfully and the average time taken to achieve the highest performance is 4.85 milliseconds. A similar parametric value achieved in [1] was 7.4867milliseconds, therefore our algorithm has provided faster detection time which has been depicted in figure 5.

In the next experimental study, the impact of extra overhead on CPU usage has been studied with our algorithm and without our algorithm and it is found that 117.85% CPU overhead is achieved without the algorithm and 134.44% with our algorithm. The average overhead is therefore 17.24% approx. which is the resultant cumulative impact from all the switches. The same parameters are compared with the existing state of the art SDN-RDCD[1] which has an overhead of 136.34% approx. when the algorithm is running. Therefore, our algorithm is imposing 1.9% less overhead on overall CPU performance which has been depicted in figure 6.

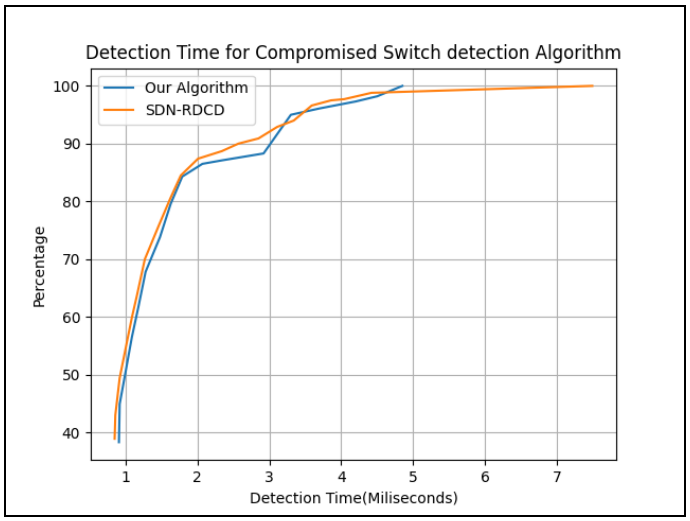

Fig.5. Compromised switch detection time

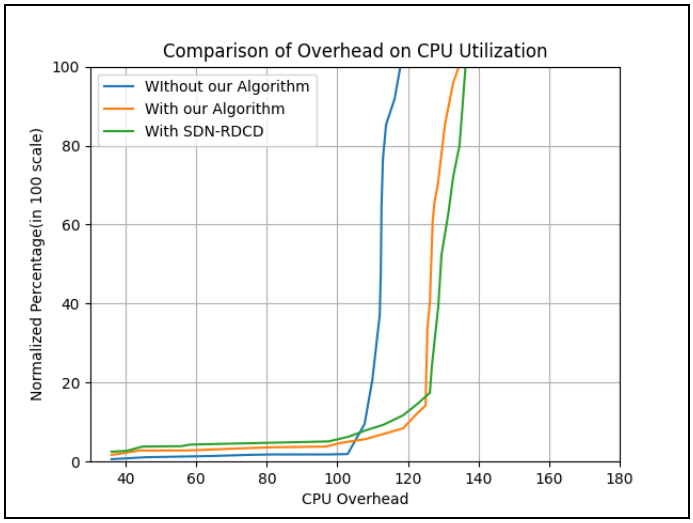

Fig.6. CPU overhead comparison of compromised switch detection algorithm

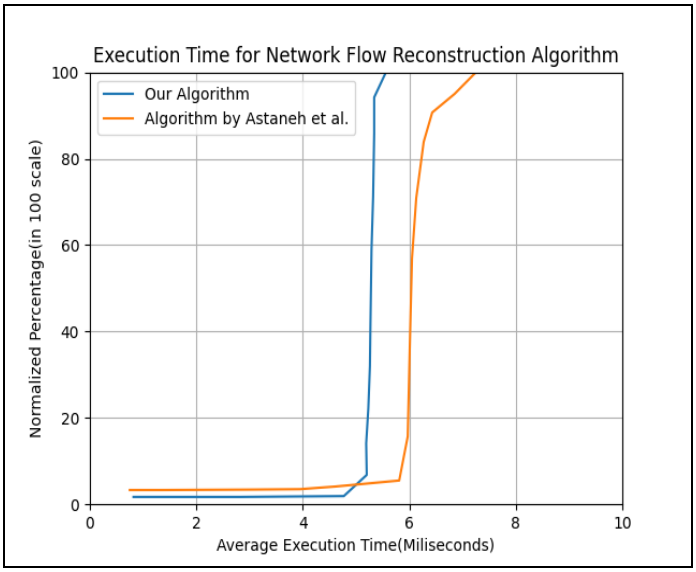

Fig.7. Execution time of network flow reconstruction algorithm

For the construction of network flow path, a sample of 25 switches and 25 hosts has been taken into consideration along-with 625 links among them. The proposed algorithm completed its execution within 5.56 milliseconds which is an average taken from 500 different execution cycles under normal operational environment. In a similar simulation

environment, we have compared our simulation result with Astaneh et al. [23] which gives an execution time of 7.25 milliseconds. Therefore, the proposed algorithm achieves a significant 1.69 millisecond faster execution time which is very crucial in real time scenario. Figure 7 presents a graphical representation of the aforesaid claim.

The similar approach has been followed for testing the CPU overhead utilization of network flow reconstruction algorithm like the previous one i.e. compromised switch detection algorithm. Therefore in this approach the extra burden or load on CPU has been measured. It is found that 135.33% CPU overhead is achieved without the algorithm and 142.47% with our algorithm. Similarly this algorithm is also compared with Astaneh et al.[23] with the same simulation testbed and it provides 144.33% overhead on CPU with the algorithm. So we can conclude that our algorithm has a very marginal amount of load addition on the CPU performance which is illustrated in figure 8.

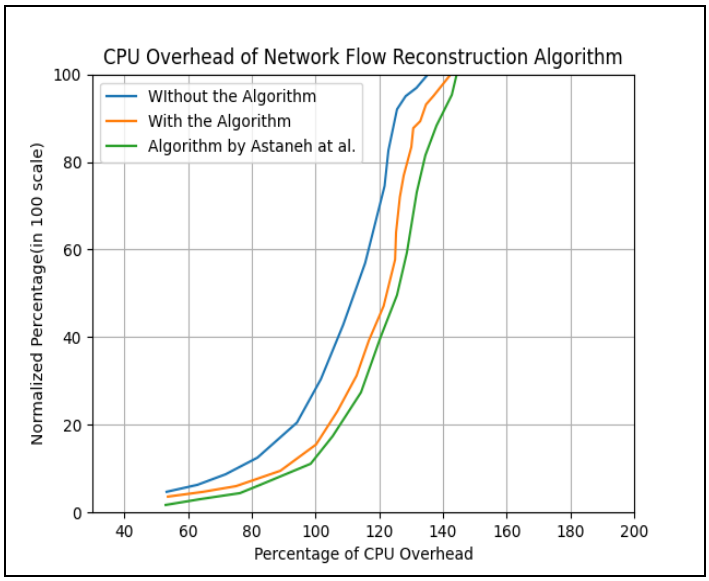

Fig.8. CPU overhead comparison of network flow reconstruction algorithm

6.3. Security Analysis

The major intention of this work is to provide an algorithmic solution for the detection of compromised SDN switches and restoration of the network flow thereafter. In the normal implementation of the SDN architecture, there is no provision of any kind of security solution to detect such type of incident and therefore any type of attack by which switch(es) might be compromised is undetected which can cause a potential damage of the whole system. Therefore to maintain normal operational flow of the network, the solution approaches proposed by us provide a holistic approach. In our work we have presented a two controller approach so that the extra load of the algorithms can be handled by the secondary controller and overall network load remains same as before.

Therefore the following security aspects have been addressed by our work-

- Switch(es) which have been attacked and compromised by the attacker has been detected and isolated from the network successfully.
- Subsequent flow table entries have also been removed for those compromised switches.
- The entire network flow have been restored again after elimination of the compromised switches.
- In short, the work proposed by us manifests a two-fold approach, i.e. detection of threat and mitigation of the ill effects caused by that threat, which is a considerable achievement and security contribution.

7. Conclusion and Future Scope

In this work an approach for the detection of the compromised SDN switches and reconstruction of network flow has been deduced which are then implemented and evaluated for their effectiveness as per their respective parameters. Two major parameters are considered in the study to evaluate the performance of both the algorithms and after comparing both of them with existing similar type of solutions it has been found that the first algorithm has achieved 35.22% of speed in running time and imposed 1.9% less overhead on computoing resource. In case of the second algorithm it has achieved 23.31% speed in running time and imposed 1.29% less overhead on computng resource. Therefore it can be concluded that both the algorithms are lightweight and efficient.

On the note of future scope there are two observations which can be addressed-firstly a controller can also be compromised like switches in a mutually trusted environment and the effects of that should be analysed properly. Secondly in the time of network flow reconstruction flow table alteration cost is an important factor which also

demands further research exposure. We believe if the compromised switch isolation controller which is deleting the entries from primary and secondary controller and updating flow table entries could be designed through FPGA more get further level of security because, to secure the proposed algorithm itself.

References

[1] Zhou, Haifeng, Chunming Wu, Chengyu Yang, Pengfei Wang, Qi Yang, Zhouhao Lu, and Qiumei Cheng. "SDN-RDCD: A real-time and reliable method for detecting compromised SDN devices." *IEEE/ACM transactions on networking* 26, no. 5 (2018): 2048-2061.

[2] Badotra, Sumit, and Japinder Singh. "Open Daylight as a Controller for Software Defined Networking." *International Journal of Advanced Research in Computer Science* 8, no. 5 (2017).

[3] Berde, Pankaj, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz et al. "ONOS: towards an open, distributed SDN OS." In *Proceedings of the third workshop on Hot topics in software defined networking*, pp. 1-6. 2014.

[4] Gao, Shang, Zecheng Li, Bin Xiao, and Guiyi Wei. "Security threats in the data plane of software-defined networks." *IEEE network* 32, no. 4 (2018): 108-113.

[5] Dhawan, Mohan, Rishabh Poddar, Kshiteej Mahajan, and Vijay Mann. "Sphinx: detecting security attacks in software-defined networks." In *Ndss*, vol. 15, pp. 8-11. 2015.

[6] Shaghaghi, Arash, Mohamed Ali Kaafar, and Sanjay Jha. "Wedgetail: An intrusion prevention system for the data plane of software defined networks." In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 849-861. 2017.

[7] Anjum, Iffat, Mu Zhu, Isaac Polinsky, William Enck, Michael K. Reiter, and Munindar P. Singh. "Role-based deception in enterprise networks." In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pp. 65-76. 2021.

[8] Haeberlen, Andreas, Petr Kouznetsov, and Peter Druschel. "PeerReview: Practical accountability for distributed systems." *ACM SIGOPS operating systems review* 41, no. 6 (2007): 175-188.

[9] Neti, Saran, Anil Somayaji, and Michael E. Locasto. "Software Diversity: Security, Entropy and Game Theory." In *HotSec*. 2012.

[10] Botelho, Fábio, Alysson Bessani, Fernando MV Ramos, and Paulo Ferreira. "On the design of practical fault-tolerant SDN controllers." In *2014 third European workshop on software defined networks*, pp. 73-78. IEEE, 2014.

[11] Al-Shaer, Ehab, and Saeed Al-Haj. "FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures." In *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration*, pp. 37-44. 2010.

[12] Son, Sooel, Seungwon Shin, Vinod Yegneswaran, Phillip Porras, and Guofei Gu. "Model checking invariant security properties in OpenFlow." In *2013 IEEE international conference on communications (ICC)*, pp. 1974-1979. IEEE, 2013.

[13] Khurshid, Ahmed, Wenxuan Zhou, Matthew Caesar, and P. Brighten Godfrey. "Veriflow: Verifying network-wide invariants in real time." In *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 49-54. 2012.

[14] Kazemian, Peyman, Michael Chang, Hongyi Zeng, George Varghese, Nick McKeown, and Scott Whyte. "Real time network policy checking using header space analysis." In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pp. 99-111. 2013.

[15] Kazemian, Peyman, George Varghese, and Nick McKeown. "Header space analysis: Static checking for networks." In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pp. 113-126. 2012.

[16] Tapolcai, János, Pin-Han Ho, Péter Babarczi, Lajos Rónyai, János Tapolcai, Pin-Han Ho, Péter Babarczi, and Lajos Rónyai. "Failure restoration approaches." *Internet Optical Infrastructure: Issues on Monitoring and Failure Restoration* (2015): 15-31.

[17] Guo, Qi, Pin-Han Ho, Hsiang-Fu Yu, Janos Tapolcai, and Hussein T. Mouftah. "Spare capacity reprovisioning for high availability shared backup path protection connections." *Computer communications* 33, no. 5 (2010): 603-611.

[18] Shah-Heydari, Shahram, and Oliver Yang. "Performance study of multiple link failure restorability of shared protection trees." In *2007 Fourth International Conference on Broadband Communications, Networks and Systems (BROADNETS'07)*, pp. 594-600. IEEE, 2007.

[19] Sinha, Rakesh K., Funda Ergun, Kostas N. Oikonomou, and K. K. Ramakrishnan. "Network design for tolerating multiple link failures using Fast Re-route (FRR)." In *2014 10th International Conference on the Design of Reliable Communication Networks (DRCN)*, pp. 1-8. IEEE, 2014.

[20] Rotsos, Charalampos, Nadi Sarrar, Steve Uhlig, Rob Sherwood, and Andrew W. Moore. "OFLOPS: An open framework for OpenFlow switch evaluation." In *Passive and Active Measurement: 13th International Conference, PAM 2012, Vienna, Austria, March 12-14th, 2012. Proceedings 13*, pp. 85-95. Springer Berlin Heidelberg, 2012.

[21] Staessens, Dimitri, Sachin Sharma, Didier Colle, Mario Pickavet, and Piet Demeester. "Software defined networking: Meeting carrier grade requirements." In *2011 18th IEEE workshop on local & metropolitan area networks (LANMAN)*, pp. 1-6. IEEE, 2011.

[22] Jarschel, Michael, Simon Oechsner, Daniel Schlosser, Rastin Pries, Sebastian Goll, and Phuoc Tran-Gia. "Modeling and performance evaluation of an OpenFlow architecture." In *2011 23rd International Teletraffic Congress (ITC)*, pp. 1-7. IEEE, 2011.

[23] Astaneh, Saeed A., and Shahram Shah Heydari. "Optimization of SDN flow operations in multi-failure restoration scenarios." *IEEE Transactions on Network and Service Management* 13, no. 3 (2016): 421-432.

Authors' Profiles

Tinku Adhikari is a research scholar of Computer Engineering department of Mizoram Central University, Aizawl, India. He has completed his M. Tech from NITTTR Kolkata and MCA from St.Xaviers College Kolkata Under IGNOU. His current area of work is Network Security, SDN Security, Quantum Cryptography. He is also associated with Techno India College of Technology, Kolkata, India as an assistant professor since last 12 years.

Dr. Ajoy Kumar Khan is presently serving as Professor and Head, Department of Computer Engineering, Mizoram University (A Central University under MHRD). He completed his B.Tech and M.Tech in Computer Science and Engineering from University of Calcutta and Ph.D from Assam University(A Central University) , Silchar. He has more than 16 years teaching experience in the field of Computer Science and Engineering. Prof. Khan published more than 36 papers in the UGC listed Journal and more than 38 papers in conference proceedings or book chapters. He served as resource person in more than 22 conference / Seminar/ FDP. Prof. Khan successfully completed three research projects sponsored by DST, DeitY and UGC, Govt. of India. Three Ph.D scholars completed , 8 pursuing and 13 M.Tech scholars successfully completed their M.Tech thesis under his supervision .His research interest field is Applied Cryptography and Information Security like Smart Card Security, IOT security, SDN Security, Cloud Data Security, IDS, Digital Forensic etc.

Dr. Malay Kule has been on the faculty of the Indian Institute of Engineering Science and Technology (IIEST), Shibpur, India since 2013, where currently, he is an assistant professor (Grade II) of Computer Science and Technology. Previously he served as an assistant professor in the department of Computer Science and Engineering of St. Thomas' College of Engineering and Technology, Kolkata, India from 2006 to 2013. He received the B. Sc. degree in physics honours, the B. Tech. and M. Tech. degrees in Computer Science and Engineering, all from the University of Calcutta, India. He received the PhD degree in Engineering from the Indian Institute of Engineering Science and Technology, Shibpur, India. His research interest includes defect tolerance of nanoscale crossbar circuits, cryptology, hardware security, and Social Network Security

Subhajit Das has over 15 years of experience in VLSI Design, Embedded Systems, and Digital Signal Processing in Industry, Research, and Academia. He began his career in the Embedded Systems domain at Wipro Technologies before transitioning to academia at Adamas Institute of Technology, India. During his doctoral research, Subhajit served the Indian Institute of Engineering, Science and Technology Shibpur, India, as an SMDP-C2SD Associate at the School of VLSI Technology. Currently, he is working as Stuff Researcher at Hiroshima University at HiSIM Research Center. His research interest includes nano-interconnect design, power device compact modeling, Internet of Things (IoT), VLSI design, signal processing, modelling of Carbon nanotube and graphene nanoribbon as nano-interconnect etc.

How to cite this paper: Tinku Adhikari, Ajoy Kumar Khan, Malay Kule, Subhajit Das, "An Efficient Approach for Detection of Compromised SDN Switches and Restoration of Network Flow", International Journal of Computer Network and Information Security(IJCNIS), Vol.16, No.5, pp.46-56, 2024. DOI:10.5815/ijcnis.2024.05.05