Securing SDN networks using AIMPD

A.ArulSelvanGnanamonickam[1], DR.B.PARAMASIVAN M.E
Research Scholar, Bharathiar University. Coimbatore, Tamilnadu, India,

Professor , Dept. of Information Technology, National Engg. College (An Autonomous Institution), KovilpattiThoothukudi, Tamilnadu, India
aarulselvan1974@gmail.com,bparamasivam@yahoo.co.in

Abstract: Conventional networks were formerly used to transfer data between nodes. The main issue with these networks was that they weren't particularly dependable and couldn't accommodate freshly added devices. As a result, conventional networks are being replaced with SDNs (Software Defined Networks) which carry data of multiple networking applications. In essence, SDNs are dynamic and can be used as foundations for applications that need a lot of data like big data. Centralizations of SDNs are their major advantages. They distinguish switches, routers, and other elements in their environments while forwarding packet. They separate control planed from user/ data planes where the controllers are essentially server-based programmes that instruct switches or routers on how to route data packets. Open Flows are essential parts of SDN installations. They have a set of protocols for direct communications with controllers residing in control planes across SDNs. Flow controls are managed using APIs (Application Programming Interfaces) are employed. Controllers govern traffic by abiding to networking rules. One major problems envisioned in SDNs is the functionality of APIs in terms of data security making it imperative to prioritise network data security in these networks. Network data can be categorised using AIs (Artificial Intelligences) in order to identify malicious or invasive packets. To find abnormalities in SDNs, several MLTs (Machine Learning Techniques) have been employed by researches including DTs (Decision Trees), RFs (Random forests), J48, NBs (Naive Bayes) and DLTs (Deep Learning Techniques). This research work suggests AIMPD (Artificial Intelligence based Malicious Packet Detections) schema based on OANNs (Optimised Artificial Neural Networks) for classifying malicious packets early from SDN traffic information using SDN traffic dataset. The schema showed classification accuracy of above 95% which is evaluated in terms of training and validation accuracy and loss.

Keywords: Software Defined networks, Traffic dataset, Artificial Neural Networks, Malicious Packets, OpenFlow Protocol, Feature Bifurcations, Deep Learning

Introduction: Enormous expansions in usages of online apps and cloud services due to increasing wired and mobile connected devices resulting in volumes in carrier network traffics. Emerging paradigms of services including virtualized clouds, big data, data centres and dissemination of multimedia materials have resulted in network administrators encountering variety of data types, services and devices while managing their networks while ensuring availability, security, and quality of services and without increasing costs of operations or equipments. SDNs have become alternatives to loaded networks in creating flexible network infrastructures with programmable devices including dynamic network architectures [1] where new protocols and rules can be specified using only software and without involving hardware. The management tools and legacy network architectures were not made to handle such extremely elastic demands. This substantially restricts an operator's capacity to adjust scale, performances, and user experiences in a way that is cost-effective for coping with the dynamic surroundings of today. The industry's solution to overcoming these difficulties has been the emergence of SDNs. SDNs can adapt dynamically to changes in user behaviours and the availability of network resources. They allow rapid creation of services at affordable costs. Moreover, network infrastructures can be modified immediately for adapting to applications and user needs. SDNs separate activities of control planes (controllers) and data planes (switches) of networks using protocols that alter network switch’s forwarding tables forwarding tables and without the need for configure current infrastructures manually. They enable networks to be optimised on the go and to react faster to changes for network demands. SDNs split software of switches from real network hardware and management of network devices from transmitted information. Controllers of SDNs have complete knowledge about the networks they control as they encompass information from switches (network resources) and apps (network consumers). These controllers enable networks to effectively reconfigure themselves as pertinent and communicate with applications, enabling them to execute various logical network topologies just like singular networks. SDNs are designs that abstract several definable network layers to create flexible and agile networks. They provide businesses

and service providers the ability to react fast to customer needs, which helps to enhance networks' governance. Network administrators or engineers shape network traffics in SDNs from centralized control panels without physically visiting each network switch. Regardless of connections between servers and devices, centralized SDN controllers guide switches to offer network services wherever they are required. These procedures differ from conventional network topologies where traffic choices are made by individual network devices based on their set up routing tables. Many networking advancements have been influenced by SDNs in significant ways. SDN architectures consist of three levels: application, control, and infrastructure. These layers communicate with one another via north- or south-bound APIs. Figure 1 depicts the architecture of SDNs.

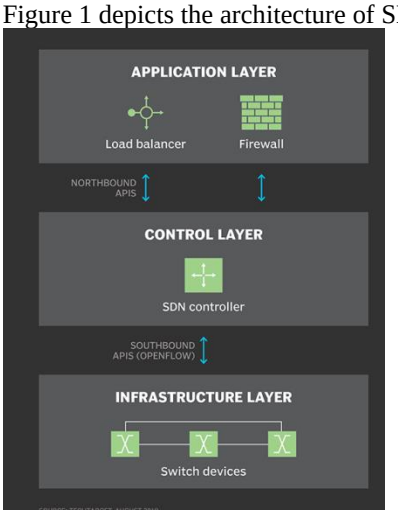

Fig 1 – Architecture of SDNs

SDNs also cover a variety of technologies, such as programmatic automation, functional segregation, and network virtualization. SDNs initially only focused on separating the control planes of networks from the data planes. Data planes transfer packets from one place to another whereas control planes decide how packets should flow via networks. In traditional SDNs, packets are sent to network switches with built-in rules for packet forwarding. Switches (data plane devices) get these rules from centralized controllers, which also receive data on network traffic. Virtual overlays, which conceptually segregate networks on top of real networks, are how SDNs are virtualized. End-to-end overlays can be used by users to segregate network traffic and abstract underlying networks. Service providers running multi-tenant cloud environments and cloud services can utilise these micro-segments to build different virtual networks with tailored rules for tenants. OpenFlows are multivendor standards created by ONF (Open Networking Foundation) for use in networking hardware when SDNs are implemented. The interface between controllers and switches is defined by the OpenFlow protocols. The OpenFlow Controller can give instructions to the OpenFlow switch using the protocol regarding how to handle incoming data packets. SDNs may both provide new risks and vulnerabilities while also preventing security breaches through symmetric and centralized controllers. Single points of failure can also occur in these central controllers. Internet industry giants like Google, Cisco, HP, Juniper, or NEC as well as standardisation bodies like the ONF (Open Network Foundation) or the IETF (Internet Engineering Task Force) were primary motivators for the introduction of OpenFlow protocol [2] in 2008. By placing sets of flow entries in switches, OpenFlow [3] enables controllers to dynamically configure network's forwarding states. Switches' behaviours are determined by flow entries kept in flow tables. IT behemoths, such Google B4 [4] and Huawei carrier networks [5], have already shown the important impact of SDNs. Other controller software for SDNs used by open source communities are NOX [6], Beacon [7], and Open Daylight [8]. SDNs are linked to many security issues with relation to data managements and application interfaces [9]. Abdul et. al. [10] proposed SDNs based on clouds where many controllers, switches and cloud servers which increase reliability. Although there is still a significant market presence for traditional networks, ideas like SDNs or NFVs (Network Function Virtualizations), many network experts are still inexperienced with this technology, and there are thousands of vulnerabilities that have not yet been identified. This paper proposes AIMPD (Artificial Intelligence based Malicious Packet Detections) schema based on ANNs (Artificial Neural Networks) for identifying malicious packets early using traffic flows and in SDNs networks with performance evaluations where accuracy above 95% is achieved in classifications. Following this introductory section, section two briefly discusses probable types of attacks followed by review of related literature. Section 3 provides an overview of the suggested AIMPD schema followed by experimental results and discussions in section 4. The paper is concluded in section 5 with future works.

Literature Review

Attacks on SDNs: Different types of attacks are possible on SDNs which are explained below. Figure 2 depicts possible attacks vectors (in red) against SDNs.

- **Network Manipulations**: An important assault on the control plane. The SDN's controller is taken over by an attacker, who then fabricates network data and launches other assaults throughout the whole network.
- **Traffic diversions**: The network components under assault are located in the data plane. The exploit compromises a network component to change traffic patterns and enable listening.
- **Side channel attacks**: This attack targets network components in data planes. Attackers can learn if flow rules are there or when timing information is not used, such as how long it takes for new network connections to be established.
- **App manipulations**: Application planes are targets of this assault. Application vulnerability attacks might lead to malfunctions, service interruptions, or data eavesdrops. Applications of SDNs can be be accessed by attackers with high privileges and perform illegal operations.
- **Denial of Services "DoS"**: One of the most frequent assaults, it can have an impact on SDNs as a whole. DoS might be used by attackers to reduce or completely stop services of SDNs.
- **ARP Spoofing Attacks**: ARP cache-poisoning assaults, commonly known as man-in-the-middle attacks. may be used by hackers to access network, sniff traffics, alter and even block them. The network topology data and topology-aware applications of SDNs are both corrupted by these kind of assaults. LLDPs and IGMPs are two other protocols that might result in poisoning.
- **API exploitations**: A software component's APIs may include flaws that might let a hacker undertake an unauthorized information leak. At the northbound interface, API abuse can also occur and result in the obliteration of network traffic.
- **Traffic sniffing**: Hackers frequently employ sniffing attacks to gather and examine network communication data. A hacker can also use sniffing to steal crucial data by listening in on network components or links. Anywhere where there is continuous traffic, sniffing can occur. In SDNs hackers can intercept traffic going to and from central controllers by taking advantage of unencrypted connections. The data gathered may provide important details about network flows or permitted traffic.
- **Password guessing or brute force**: This assault might target an element that is not SDNs. An unauthorised person might access SDNs by brute forcing passwords or through password guesses..

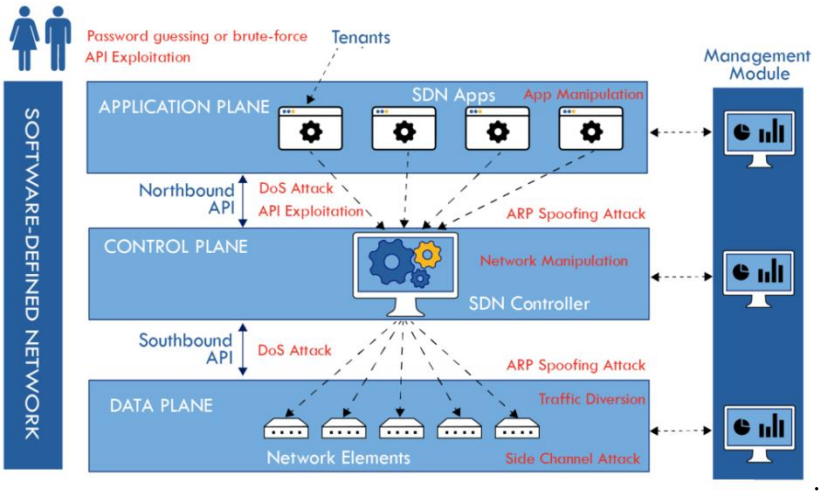

Figure 2 – Attack types on SDNs

In recent years, flow-based anomaly detection classifications have been extensively investigated. Flow-based anomaly detection system, using MLPNNs (multi-layer Perceptron neural networks) with one hidden layer and GSAs (gravitational search algorithms) proposed in [11] while flows were classified as benign and malicious with high degree of accuracy. In [12], The study introduced a novel concept for inductive network intrusions using one-class SVMs (support vector machines) for analysis of trained malicious network data in contrast to other systems which give lower false alarm rates. Multiple anomaly detection algorithms were proposed by the authors of [13] where NOX and OF compliant switches were used. The schemes successfully detected anomalies in traffics of small

networks but failed in ISPs. The study in [14] presented light weighted method for detecting DDoS attacks based on traffic flow attributes, where data was retrieved with very little efforts by utilising programmable interfaces made available on NOX platform. The study's high rate of detections were due to the usage of SOMs (self-organizing maps). SVMs were also used for analysis of DDoS attacks by Kokila et al. [15] where false positive rates were reduced and higher classification accuracies were obtained when compared to other methods. Trung et al. [16] proposed enhanced defences using SVMs with IAs (idle - timeout adjustments) to secure and preserve network resources in the event of flooding assaults in SDNs. Using entropy variations of destination IP addresses, a simple technique in [17] identified DDoS assaults within specified fraudulent traffic packet counts. Niyaz et al [18] DLTs, based on SAEs (Stacked Auto Encoders) (SAE), identified DDoS multi-vector assaults in SDNs. Tang et al. [19] modelled DNNs (deep neural networks) for intrusion detections system, trained on NSL-KDD dataset for identification of flow-based abnormalities in SDNs. However, they only utilised six of the dataset's fundamental characteristics, and shockingly, no suitable feature selection methodology was applied. Tang et al. [20] in their later studies also proposed GRU-RNNs (gated recurrent unit recurrent neural networks) which enabled intrusion detections with high degrees of accuracy and very limited features when compared to other works. Sen et al. [21] offered another method of creating controllers for SDNs based on rules during DDoS assaults which were identified using MLTs, tested on virtual network test bed SDNs. A network trace dataset based on SDNs was also produced in order to construct and test their MLTs. Vetriselvi et al. [22] suggested dual MLTs for intrusion detections in SDNs. They created an intrusion assessment systems by combining GAs (genetic algorithms) with MLTs where the first phase identified assaults followed by classifications in the second stage. Using NSL-KDD dataset, Elsayed et al. [23] thoroughly examined several MLTs capable of identifying intrusions in SDNs. To obtain improved accuracies in intrusion detections, the study by Dey in [24] used two distinct methods of feature selections based on DLTs and MLTs. Dey [28] in a study investigated performances of MLTs using various feature selection strategies in terms of intrusion detections in SDNs. Many supervised learning algorithms have been used to evaluate the quality of SDN datasets: Three tree-based algorithms namely DTs [28], RFs [29], and Adaptive Boosts; KNNs (k-nearest Neighbors) [31]; NBs and SVMs [33] based on linear kernels and radial basis functions. In addition MLPs (multi-layer Perceptrons) were also engaged to evaluate InSDN dataset. Thus, though many methods exist in literature, gaps still exist in the area of intrusion detections for SDNs.

Proposed AIMPD Schema

Traffic classification is one of the most important areas of network management. There are broadly three approaches for network traffic classification namely port-based approach that are simple and fast but can be easily manipulated and are less reliable; Deep packet inspections give good results but can only be used for unencrypted traffic and fail in real-time encrypted data/traffic and finally approaches based on AIs. The main focus of this work is to use AIs and discover malicious packets early from traffic flows (dataset). The proposed schema AIMPD is based on ONNs and classifying malicious packets early from SDN's traffic information. AIMPD's methodology follows three main stages namely data preparation, feature preparations/bifurcations and classifications. The classified samples are evaluated in terms of training and testing accuracies.

AIMPD Schema

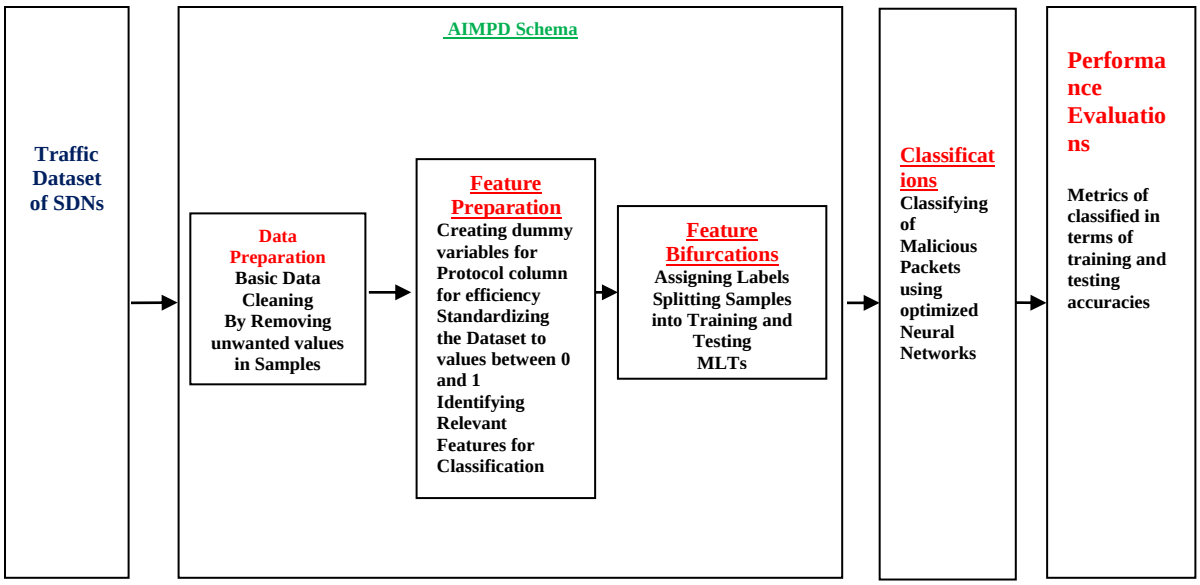

Fig. 3 - AIMPD Scheme

The necessary features were obtained from SDN's traffic dataset. The dataset included the following features: dt (date), switch (switch no), src (packet's source IP), dst (packet's destination IP), pktcount (counts of packets), bytecount (counts of bytes), dur (data flow durations), dur_nsec (data flow durations in nano seconds), tot_dur (total flow dutations), flows (counts of flows), packetins (Input Packets from Devices), pktperflow (counts of packets in flows), byteperflow (counts of bytes in flows), pktrate (rates at which packets arrive), Pairflow (flows in pairs), Protocol (UDP, ICMP, TCP), port_no (port nos), tx_bytes (counts of bytes sent by functions), rx_bytes (counts of bytes received by functions) and tot_kbps (data flows in terms of total kilobytes per second). Figure 4 shows a snapshot of the SDN traffic data set.

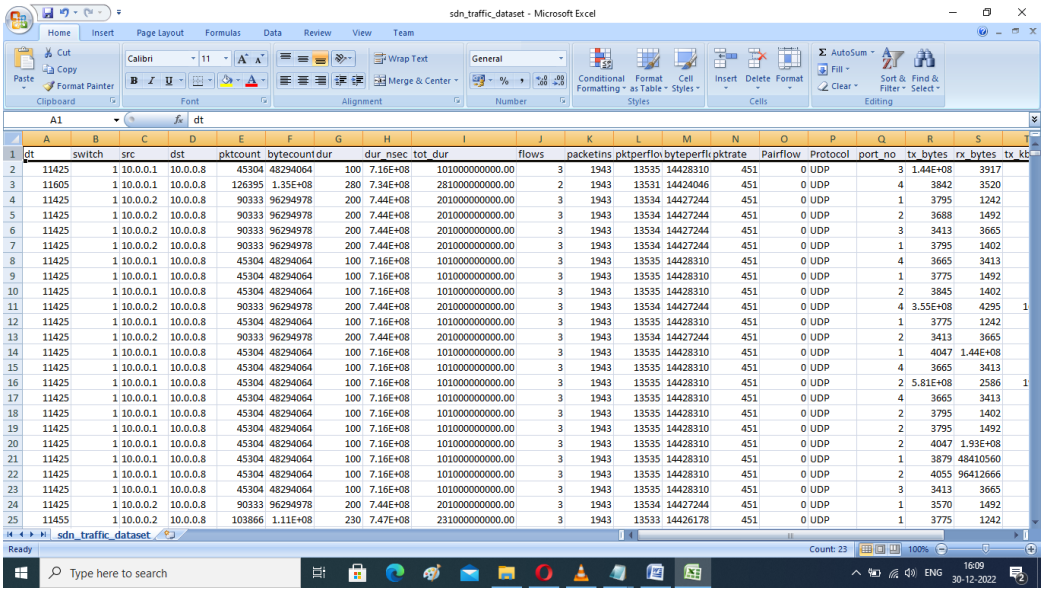

Fig. 4 – Snapshot of Traffic Dataset

AIMPD – Data preparation: Preparing input data for processing are significant steps and involves transformations of raw inputs and cleaning them to aid further processes and analyses. These preliminaries are time-consuming tasks, but necessary for getting insights or removing biases brought on by bad qualities of data. Accurate business choices can only be made with clean data, which imply its purposefulness. Processes that clean data are critical in spite of being the most time-consuming phases in data preparation processes. They eliminate erroneous data and fill gaps (missing values) while ensuring adherence to given patterns. Data inputs are validated after cleaning by

examining errors found in initial steps of data preparations. This allows for corrections in inputs. AIMPD executes basic Information Cleaning and deleting unnecessary columns/values from input samples.

AIMPD – Feature Preparation: MLTs need high-quality data for their learning and feature preparations are the next steps after data preparations which validate inputs. Feature preparations (also known as feature engineering) are processes that turn data inputs into features, making it usable by MLTs. Improper feature preparations can result in essential relationship losses including linear models with unexplored nonlinear elements or DLTs with inconsistent input data. Even minor differences in data/feature preparations can result in dramatically different outcomes. Iterative methods produce better outcomes both in feature/data preparations as they estimate model's problems and revisit them. AIMPD generates dummy variables for the protocol column for improving its efficiency while transforming dataset values in the range (0,1) using Min-Max scales. Choosing relevant features are finals steps of feature preparations.

AIMPD – Feature Bifurcations: AIMPD identifies relevant features and assigns them labels for accurate classifications of malicious samples. Data labels (annotations) are processes of target attribute additions to training samples and generating labels used by models of MLTs for predictions and and specifically supervisedMLTs.
AIMPD – Classifications: ANNs are algorithms inspired by the workings of biological neural networks found in living organisms. They are usually composed of interconnected nodes and weights. Inputs first pass through nodes (neurons) which are activated with functions and multiplied with weights to produce outputs. They function by imitating a large number of linked processing units and are similar to abstract representations of neurons. They use multiple layers for processing, but minimally have input, hidden, and output layers. These units are linked with varied degrees of strengths (or weights). Inputs (initial layers) get transmitted from neurons to additional layers before they reach output layers. ANNs learn by studying individual items and generate predictions based on their weight adjustments when inaccuracies exist while processing. These iterate their procedures for enhancing predictions before reaching stopping points. Initially, when weights are at random, the results may seem illogical, but as iterations pass networks start learning. For examples, known outcomes are compared with learnt responses whichare sent back into the networks, progressively modifying weights. As trainings advance, network's capability to replicate known outcomes improve and thus trained, ANNs can predict unknown outcomes. Figure 5 depicts ANNs.

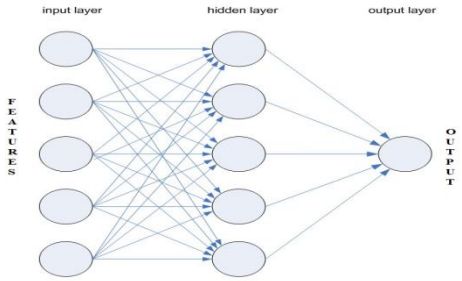

Fig.5 - , Structure of ANNs

Results
The proposed AIMPD was implemented using python 3 running on windows 10 with an AMD processor. The stage wise results of this work's suggested system are detailed in this section in the form of figures or tables wherever necessary.
AIMPD Data Preparation:
Data transformations modify formats or values in order to achieve certain results or to make data more understandable to larger audiences. Enriching data are done by additions or integrations of data with other relevant information in order to deliver deeper insights of data. Data preparations assist in rapid corrections of errors and produce high-quality data with improved scalabilities which help organizations keep in line with innovation curves and without incurring delays or additional expenditures. The columns ID, switch numbers, source/destination IP addresses, and port numbers were removed since they were unrelated to predictions.. Figure 6 depicts AIMPD output of data preparations.

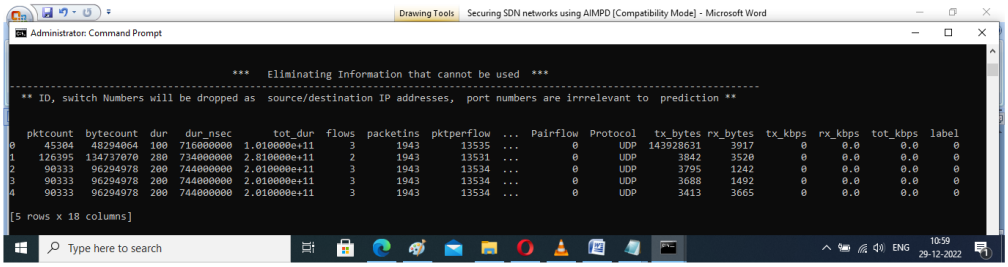

Fig. 6 - AIMPD Output of Data Preparations

AIMPD – Feature Preparation: Dummy variables for the Protocol column were produced in order to improve the efficiency of AIMPD. Creating dummy variables are similar to creating other variables, except that these variables have their values in the range (0,1) i.e. false or true. This provides enhanced possibilities for determining inclusions of dummies. Dummy variables are frequently used to include categorical variables in models.

AIMPD standardizes dataset values between 0 and 1 using MinMax Scaler where the normalizations are done using Equation (1)

$$z_i = (x_i - \min(x)) / (\max(x) - \min(x)) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

where: z_i is the i^{th} normalized value in the dataset, x_i stands for i^{th} value in the dataset, min(x) represents minimum value in the dataset and max(x) stands for maximum value in the dataset. The objective of Min-Max Normalizations are to converts each data value to a value between 0 and 100 where new values are obtained using Equation (2). Figure 7 depicts AIMPD output of Feature Preparations.

$$\text{New value} = (\text{value} - \min) / (\max - \min) * 100 \ldots\ldots\ldots\ldots\ldots\ldots\ldots..(2)$$

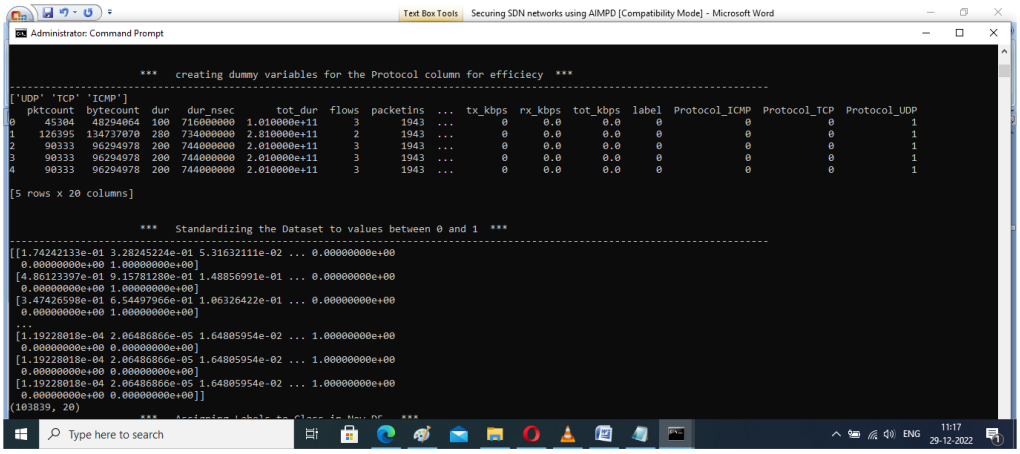

Figure 7 depicts AIMPD output of Feature Preparation.

AIMPD – Feature Bifurcations: AIMPD assigns labels to Classes in data samples. Class labels are often string values, e.g. “spam,” “not spam,” and must be mapped to numeric values before being provided to an algorithm for modeling. This is often referred to as label encoding, where a unique integer is assigned to each class label, e.g. “spam” = 0, “no spam” = 1.Figure 8 depicts AIMPD output of Label Assignments.

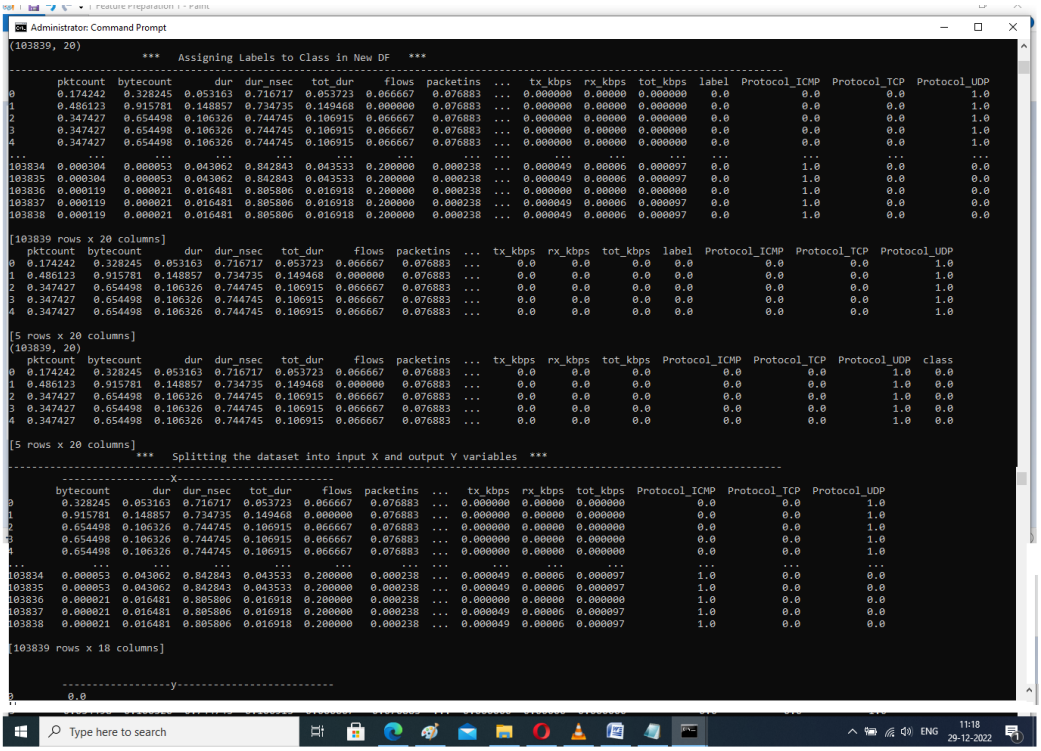

Fig. 8 - AIMPD output of Label Assignments

The dataset values are subsequently split into training set (80%) and test set (20%) both for classifications and evaluations of AIMPD. Figure 9 depicts AIMPD data sample splits. Table 1 lists the important features of SDN traffic dataset.

TABLE 1 -Extracted Traffic Features from the dataset

No.	Feature Description	SDN Derived Features	Traffic Dataset
1	Length of the connection	Flow Duration	dur
2	Protocol_type	Protocol	Protocol
3	Packets from Switches	Input Packets from Devices	packetins
4	Packet arrival rates	rates at which packets arrive	pktrate
5	Packets in flows	counts of packets in flows	pktperflow
6	Received bytes	bytes received by functions	rx_bytes
7	Sent bytes	bytes sent by functions	tx_bytes

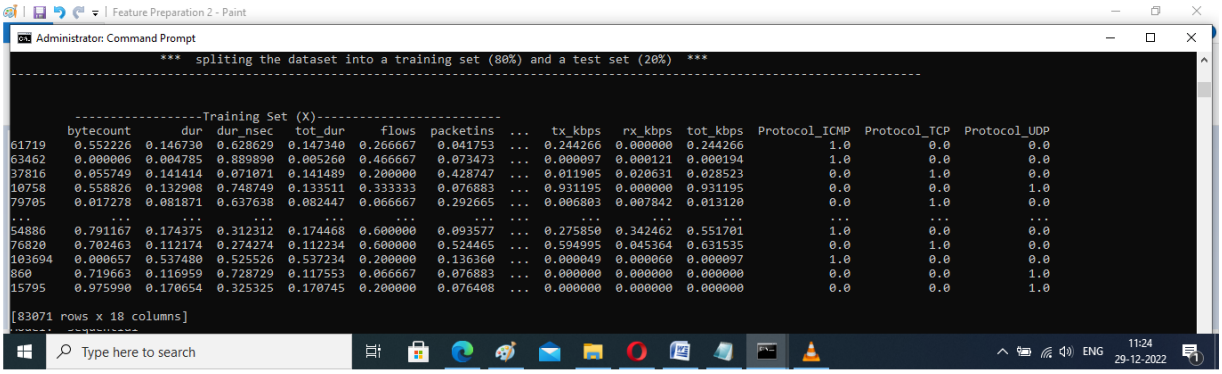

Fig. 9 - AIMPD data sample splits

AIMPD – Classifications: AIMPD uses ANNs for classifications. One major advantage of AIMPD is in defining early stopping callbacks (iterations) as it stops the learning of ANNs when no improvements in validation losses are found for 5 consecutive epochs where epochs imply training ANNs with all the training data for one cycle. Figure 10 depicts an epoch in ANNs.

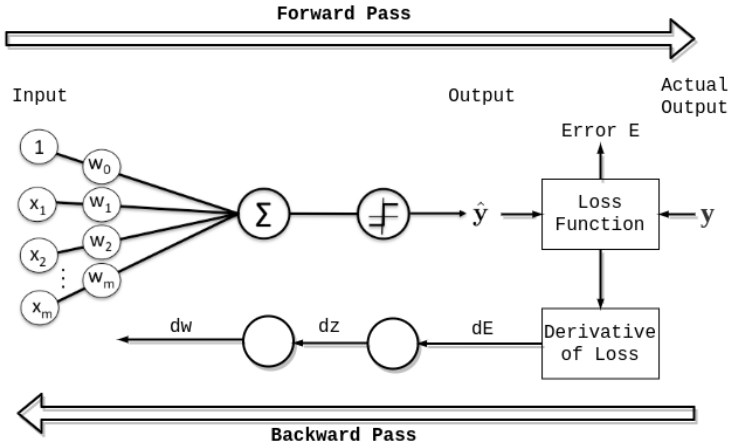

Fig. 10 – Single Epoch of ANNs

Epochs use the data sample only once in each epoch. A forward pass and a backward pass together are counted as one pass. AIMPD uses binary crosentropy to compute losses at epochs during training and Adam optimizer. Figure 11 depicts the AIMPD's model of ANNs.

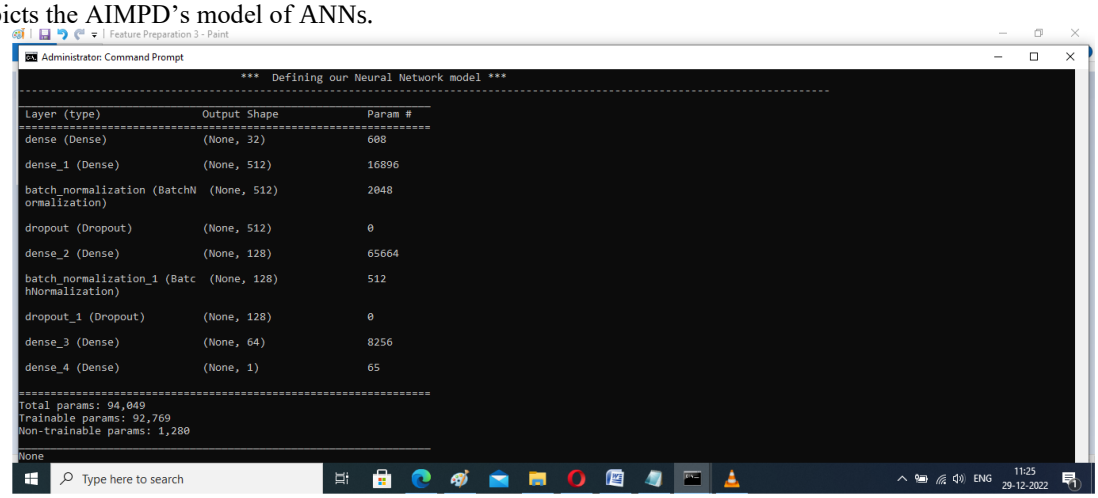

Fig. 11 - AIMPD's model of ANNs

Thus, AIMPD was trained on the SDN traffic dataset to classify malicious packets and evaluated for its efficiency. Figure 12 depicts AIMPD's optimized ANNs outputs.

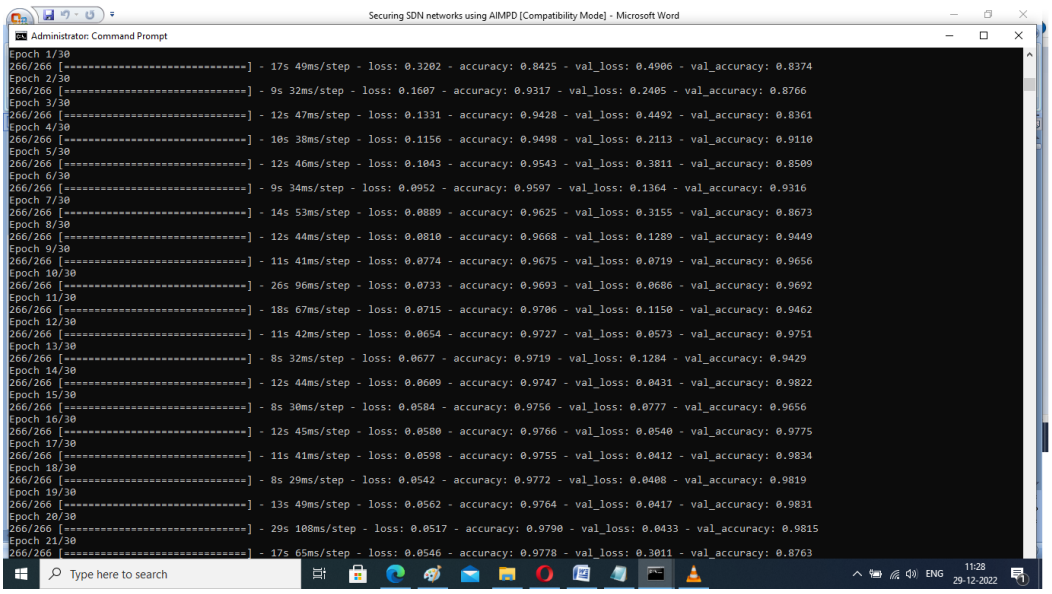

Fig. 12 - AIMPD's optimized outputs of ANNs

AIMPD Evaluations: DLTs use ANNs and can be validated using two concepts namely training and validation losses. Consequently, when DLTs are used on specific datasets, models take inputs and produce outputs where their performances are measured using the measures of loss as it quantifies errors produced by models. High loss values imply models are producing erroneous outputs, while low loss values indicate fewer errors. Moreover, losses are computed using cost functions which are dependent on the problems that need to be solved and the data fed into the networks. The errors can be measures in many ways, but generically cross entropies are used for binary classifications. The training losses are metrics used to assess how models of DLTs fit into training data or assesses errors of the models on training sets computationally; training losses are computed by taking sums of errors of training set examples. Training losses are measured for each batch and visualized by plotting curves of training losses. Validation losses are metrics used to assess performances of models based on DLTs on validation sets where validation sets are parts of the dataset set aside or split to validate model performances. These losses are similar to training losses and computed as sums of sample's errors in validation sets. Additionally, validation losses are measured after epochs which indicates if models need further tuning or adjustments. These losses can be plotted as learning curves. In most DLTs, training and validation losses are usually visualized together on graphs for diagnosing model's performances and identify which aspects that need tuning. In case both training and validation losses are high, validation losses greater than training losses indicate that models encounter under fits. i.e. training data is modelled accurately resulting in large errors. On the contrary when validation losses are greater than training losses, it implies models encounter data over fits. and cannot generalize new data. In these cases models, perform well on training data, but perform poorly on validation sets. AIMPD avoids both under and over fits due to its optimizations of ANNs as is evident from Figures 13 and 14.

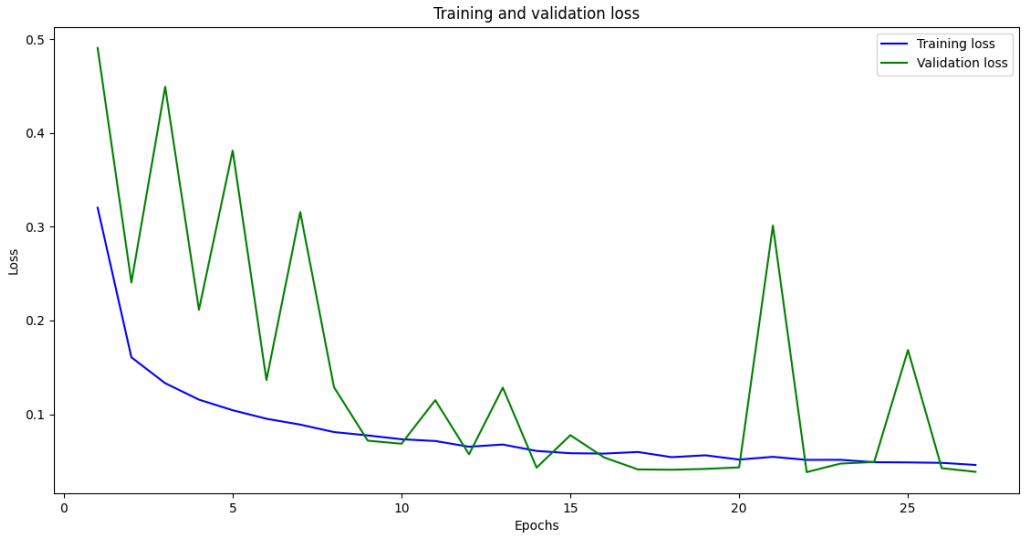

Fig. 13 – Training and Validation Loss of AIMPD on SDN traffic dataset

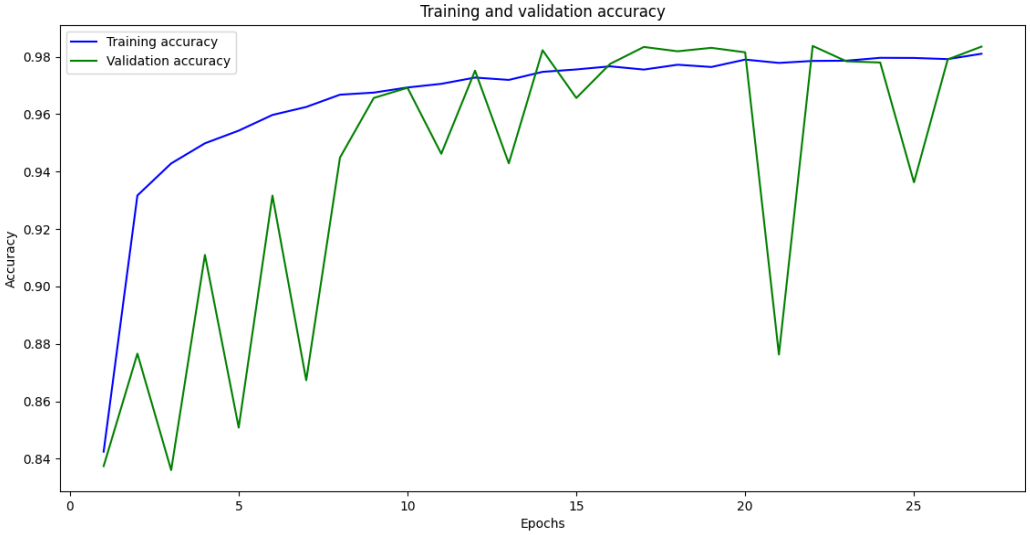

Fig. 14 – Training and Validation Accuracy of AIMPD on SDN traffic dataset

Conclusion: Conventional networks were formerly used to transfer data between nodes. The main issue with these networks was that they weren't particularly dependable and couldn't accommodate freshly added devices. As a result, conventional networks are being replaced with SDNs which carry data of multiple networking applications. In essence, SDNs are dynamic and can be used as foundations for applications that need a lot of data like big data. Centralizations of SDNs are their major advantages. Network evolutions are responsible for new kinds of assaults which stem as known and unknown hazards, and zero-day exploits. Since there are currently no histories of prior real-case attacks on SDNs, it is difficult to identify current weaknesses and create protections around these network controllers. A taxonomy of possible attacks can assist in establishing foundations of security. The centralized controllers create issues. New network technologies might pose previously unknown hazards or perhaps make matters worse, since controllers and links to control planes present novel security issues that are specific to SDNs. Using a dataset, this paper has explored a difficult area linked to malicious packets in traffics of SDNs. Popular network assaults may also affect SDNs which are more vulnerable to malicious traffics than traditional networks. In traditional

networks, assaults may only damage subsets of networks from the same vendor without bringing the entire network down. However, in SDNs, hacked switches or end-users might overwhelm controllers, causing widespread network disruptions. Hence, this study proposed AIMPD assesses this intensity, has focused on detection systems for identifying malicious packets in SDNs. This proposed work contributes to researches on SDNs by proposing a model based on DLTs for improved classifications of malicious packets.

References

[1] Software Defined Networking Definition. Available online: https://www.opennetworking.org/sdn-definition (accessed on 16 May 2017).
[2] ONF SDN Evolution. Available online: http://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2013/05/TR-535_ONF_SDN_Evolution.pdf (accessed on 25 February 2018)
[3] McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. Openflow: Enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 2008, 38, 69–74.
[4] Jain, S.; Kumar, A.; Mandal, S.; Ong, J.; Poutievski, L.; Singh, A.; Venkata, S.; Wanderer, J.; Zhou, J.; Zhu, M.; et al. B4: Experience with a globally-deployed software defined wan. SIGCOMM Comput. Commun. Rev. 2013, 43, 3–14.
[5] C.T. Huawei Press Centre and H. Unveil World's First Commercial Deployment of SDN in Carrier Networks. Available online: http:://pr.huawei.com/en/news/hw-332209-sdn.htm (accessed on 28 February 2018).
[6] Gude, N.; Koponen, T.; Pettit, J.; Pfa , B.; Casado, M.; McKeown, N.; Shenker, S. Nox: Towards an operating system for networks. SIGCOMM Comput. Commun. Rev. 2008, 38, 105–110..], Ryu [Ryu. Available online: http://osrg.github.io/ryu (accessed on 11 March 2018)
[7] Erickson, D. The beacon openflow controller. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, (HotSDN '13), Hong Kong, China, 16 August 2013; ACM: New York, NY, USA; pp. 13–18.
[8] Opendaylight: A Linux Foundation Collaborative Project. Available online: http://www.opendaylight.org (accessed on 6 March 2018)], and Floodlight [Floodlight. Available online: http://www.projectfloodlight.org (accessed on 15 March 2018)
[9] Kreutz, D.; Ramos, F.M.; Verissimo, P. Towards Secure and Dependable Software-Defined Networks. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, (HotSDN '13), Hong Kong, China, 16 August 2013; ACM: New York, NY, USA; pp. 55–60.
[10] I. Abdulqadder, D. Zou, I. Aziz, B. Yuan, and W. Dai, "Deployment of robust security scheme in sdn based 5g network over nfv enabled cloud environment," IEEE Transactions on Emerging Topics in Computing, 2018
[11] Jadidi, Z.; Muthukkumarasamy, V.; Sithirasenan, E.; Sheikhan, M. Flow-Based Anomaly Detection Using Neural Network Optimized with Gsa Algorithm. In Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops, Philadelphia, PA, USA, 8–11 July 2013; pp. 76–81.
[12] Winter, P.; Hermann, E.; Zeilinger, M. Inductive Intrusion Detection in Flow-Based Network Data Using One-Class Support Vector Machines. In Proceedings of the 2011 4th IFIP International Conference on New Technologies, Mobility and Security, Paris, France, 7–10 February 2011; pp. 1–5.
[13] Mehdi, S.A.; Khalid, J.; Khayam, S.A. Revisiting Tra c Anomaly Detection Using Software Defined Networking. In Lecture Notes in Computer Science, Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection, (RAID'11), Menlo Park, CA, USA, 20–21 September 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 161–180.
[14] Braga, R.; Mota, E.; Passito, A. Lightweight Ddos Flooding Attack Detection Using Nox/Openflow. In Proceedings of the IEEE Local Computer Network Conference, Denver, CO, USA, 10–14 October 2010; pp. 408–415.
[15] Kokila, R.T.; Selvi, S.T.; Govindarajan, K. DDoS Detection and Analysis in SDN-Based Environment Using Support Vector Machine Classifier. In Proceedings of the 2014 Sixth International Conference on Advanced Computing (ICoAC), Chennai, India, 17–19 December 2014; pp. 205–210.
[16] Phan, T.V.; van Toan, T.; van Tuyen, D.; Huong, T.T.; Thanh, N.H. OpenFlowSIA: An Optimized Protection Scheme for Software-Defined Networks from Flooding Attacks. In Proceedings of the 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), Ha Long, Vietnam, 27–29 July 2016; pp. 13–18.
[17] Mousavi, S.M.; St-Hilaire, M. Early Detection of Ddos Attacks Against Sdn Controllers. In Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC), Garden Grove, CA, USA, 16–19 Febuary 2015; pp. 77–81. Symmetry 2020, 12, 7

[18] Niyaz, Q.; Sun, W.; Javaid, A.Y. A deep learning based ddos detection system in software-defined networking (sdn). arXiv 2016, arXiv:1611.07400.
[19] Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi, S.A.R.; Ghogho, M. Deep Learning Approach for Network Intrusion Detection in Software Defined Networking. In Proceedings of the 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), Fez, Morocco, 26–29 October 2016; pp. 258–263.
[20] Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi, S.A.R.; Ghogho, M. Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, Canada, 25–29 June 2018; pp. 202–206.
[21] Sen, S.; Gupta, K.D.; Manjurul Ahsan, M. Leveraging Machine Learning Approach to Setup Software-Defined Network(SDN) Controller Rules During DDoS Attack. In Algorithms for Intelligent Systems, Proceedings of the International Joint Conference on Computational Intelligence, Dhaka, Bangladesh, 4 July 2019; Uddin, M., Bansal, J., Eds.; Springer: Singapore, 2020.
[22] Vetriselvi, V.; Shruti, P.S.; Abraham, S. Two-Level Intrusion Detection System in SDN Using Machine Learning. In ICCCE 2018, Proceedings of the Lecture Notes in Electrical Engineering, Hyderabad, India, 24 January 2018; Kumar, A., Mozar, S., Eds.; Springer: Singapore, 2019; Volume 500.
[23] Elsayed, M.S.; Le-Khac, N.A.; Dev, S.; Jurcut, A.D. Machine-Learning Techniques for Detecting Attacks in SDN. arXiv 2019, arXiv:1910.00817.
[24] Dey, S.K.; Rahman, M.M. Flow based anomaly detection in software de-fined networking: A deep learning approach with feature selection method. In Proceedings of the 2018 4th International Conference on Electrical Engineering and Information Communication Technology (iCEEiCT), Dhaka, Bangladesh, 13–15 September 2018; pp. 630–635.
[25] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," IEEE Access, vol. 6, pp. 1792–1806, 2017.
[26] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based ddos detection system in software-defined networking (sdn)," arXiv preprint arXiv:1611.07400, 2016.
[27] T. V. Phan, N. K. Bao, and M. Park, "A novel hybrid flow-based handler with ddos attacks in software-defined networking," in 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld). IEEE, 2016, pp. 350– 357.
[28] N. Frosst and G. Hinton, "Distilling a neural network into a soft decision tree," arXiv preprint arXiv:1711.09784, 2017.
[29] M. Belgiu and L. Dragu¸t, "Random forest in remote sensing: A review of applications and future directions," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 114, pp. 24–31, 2016.
[30] A. J. Wyner, M. Olson, J. Bleich, and D. Mease, "Explaining the success of adaboost and random forests as interpolating classifiers," The Journal of Machine Learning Research, vol. 18, no. 1, pp. 1558–1590, 2017.
[31] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient knn classi-fication with different numbers of nearest neighbors," IEEE transactions on neural networks and learning systems, vol. 29, no. 5, pp. 1774–1785, 2017.
[32] S.-C. Chu, T.-K. Dao, J.-S. Pan et al., "Identifying correctness data scheme for aggregating data in cluster heads of wireless sensor network based on naive bayes classification," EURASIP Journal on Wireless Communications and Networking, vol. 2020, no. 1, pp. 1–15, 2020.
[33] S. M. H. Bamakan, H. Wang, T. Yingjie, and Y. Shi, "An effective intrusion detection framework based on mclp/svm optimized by time-varying chaos particle swarm optimization," Neurocomputing, vol. 199, pp. 90–102, 2016.